

...IN  
**21**  
TAGEN

# ORACLE 8

## Datenbankentwicklung

in 21 Tagen

Bewährtes Tutorial  
für alle Spielarten des  
Datenbankdesigns mit  
dem neuen Oracle 8

Führt Sie in die  
Entwicklungstools  
Developer/2000 and  
Oracle Power Objects  
ein

Macht Sie produktiv  
für flexible Anwendungs-  
entwicklung innerhalb  
kurzer Zeit

David Lockman

**SAMS**



Auf CD-ROM  
der Source-Code  
aus dem Buch sowie  
Tutal-Produkte von Oracle

# Oracle in 21 Tagen Inhaltsverzeichnis

---

## Einleitung

1. Die Welt relationaler Datenbanken erforschen
2. Richtlinien für die Entwicklung einer Oracle-Anwendung
3. Logischer Datenbankentwurf
4. Implementierung Ihres logischen Modells: Physischer Datenbankentwurf
5. Einführung in SQL
6. Mit SQL Daten modifizieren
7. In SQL integrierte Funktionen nutzen
8. Fortgeschrittene Abfragen mit SQL
9. Programmierung einer Oracle-Datenbank mit PL/SQL
10. Programmentwicklung mit PL/SQL
11. Weitere Programmiertechniken mit PL/SQL
12. Developer/2000: Einführung in Oracle Forms
13. Developer/2000: Entwicklung einer Benutzeroberfläche
14. Developer/2000: Anwendungsentwicklung mit Hilfe von Oracle Forms
15. Developer/2000: Entwicklung von Berichten mit Hilfe von Oracle Reports
16. Developer/2000: Oracle Graphics und den Procedure Builder verwenden
17. Einführung in Oracle Power Objects
18. Eine Anwendung mit Oracle Power Objects entwickeln
19. Datenbanksicherheit und Datenbankoptimierung unter Oracle

20. [Database Designer und Designer/2000 in der Anwendungsentwicklung](#)
  21. [Oracle - Die nächste Generation](#)
  22. [Bonustag: Power-Builder für eine Oracle-Datenbank verwenden](#)
  23. [Bonustag: Oracle8 - Neue Merkmale](#)
  24. [Antworten zu den Tests und Übungen](#)
- 

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

 [Top](#) [Index](#)

[← Kapitel](#) [Inhalt](#) [Index](#) [↑ Oben](#) [Info](#) [Suchen](#) [Kapitel →](#)

---

# Einleitung

[Voraussetzungen](#)

[Die CD-ROM](#)

[Über dieses Buch](#)

[In diesem Buch verwendete Konventionen](#)

---

Willkommen zu »Oracle 8 in 21 Tagen«! Dieses Buch behandelt die Verwendung des Datenbanksystems Oracle und der zu Oracle gehörenden Programmierwerkzeuge im Prozeß der Anwendungsentwicklung. Während einer Zeitdauer von drei Wochen werden Sie mit den unterschiedlichen Aspekten der Datenbankentwicklung vertraut gemacht, einschließlich des logischen Datenbankentwurfs, des physischen Datenbankentwurfs und der Verwendung der strukturierten Abfragesprache (Structured Query Language, SQL), einer Standardsprache für die Kommunikation mit einer relationalen Datenbank.

Zusätzlich lernen Sie PL/SQL kennen, die in Oracle integrierte prozedurale Spracherweiterung zu SQL. Mit PL/SQL können Sie Anwendungslogik in eine Oracle-Datenbank einbetten. Sobald Sie die Grundlagen von PL/SQL erlernt haben, widmen Sie sich fünf Lektionen lang Developer/2000, einer Sammlung von Programmierwerkzeugen für die Anwendungsentwicklung, die Oracle Forms, Reports, Graphics und Procedure Builder umfaßt. Zwei weitere Lektionen konzentrieren sich auf Oracle Power Objects, ein anderes Tool für die Anwendungsentwicklung. Sie lernen außerdem etwas über die Datenbanksicherheit und die Datenbankoptimierung unter Oracle. Dieser Lektion folgt eine Übersicht über zwei Oracle-Produkte, die Sie möglicherweise bei der Anwendungsentwicklung einsetzen könnten, nämlich Designer/2000 und Database Designer. Die abschließende Lektion steigt in die drei Bereiche Oracle Web Application Server, Developer/2000 for the Web und die Network Computing Architecture von Oracle ein.

Zusätzlich zu den 21 Lektionen gibt es zwei Bonustage: Am ersten Tag wird die Verwendung von Powerbuilder zusammen mit einer Oracle-Datenbank erörtert, am zweiten Tag werden neue Merkmale von Oracle8 vorgestellt, die für Datenbank- und Anwendungsentwickler von besonderem Interesse sein könnten.

## Voraussetzungen

Sie ziehen den meisten Nutzen aus diesem Buch, wenn Sie folgende praktische Erfahrungen mitbringen:

- Kenntnisse mindestens einer Programmiersprache, wie z.B. BASIC oder C
- Sicherheit im Umgang mit den Windows- oder Windows-95-Umgebungen
- Vorkenntnisse im Umgang mit einer Datenbank, wie z.B. Microsoft Access, dBase, FoxPro, Paradox oder auch Excel

Die beste Möglichkeit, sich in ein neues Themengebiet einzuarbeiten, ist, sich darin zu üben und viele Dinge auszuprobieren. Um den in jeder Lektion vorgestellten Beispielen folgen sowie jeden Test und jede Übung bearbeiten zu können, benötigen Sie Zugang zu folgenden Programmen:

- einer Oracle-Datenbank. Es ist nicht entscheidend, um welche Art einer Oracle-Datenbank es sich handelt. Es kann entweder der Universal Server, der Workgroup Server oder Personal Oracle sein. Wenn Sie auf einen Oracle-Server zugreifen, benötigen Sie offensichtlich einen zuverlässigen Netzwerkzugriff auf die Datenbank. Idealerweise sollte die Oracle-Datenbank unter Version 7.1 oder höher des Oracle-RDBMS laufen.
- zu Developer/2000 oder Power Objects. Fünf Lektionen beschäftigen sich mit den Komponenten von Developer/2000, Version 1.3, nämlich mit Oracle Forms, Reports, Graphics und Procedure Builder. Zwei weitere Lektionen sind Power Objects 2.0 gewidmet. Um die Funktionsweise der in diesen Lektionen vorgestellten Beispielanwendungen nachvollziehen zu können, ist es für Sie zweifellos wünschenswert, die entsprechenden Versionen dieser Programmierwerkzeuge zur Verfügung zu haben, um praktische Erfahrungen damit sammeln zu können.
- Sie benötigen weiter einen PC, auf dem Windows 95 läuft. Obwohl die Beispiele in diesem Buch in einer Windows-95-Umgebung vorgestellt werden, können Sie für viele dieser Beispiele auch Windows NT benutzen.

## Die CD-ROM

Die CD-ROM zum Buch enthält eine Datei namens `fluggle.dmp`, welche die in diesem Buch besprochene Beispieldatenbank enthält. Indem Sie diese Datei in eine Oracle-Datenbank laden, können Sie den in jeder Lektion vorgestellten Beispielen folgen. Eine Datei `readme.txt`, die sich ebenfalls auf der CD-ROM befindet, enthält Hinweise darüber, wie man `fluggle.dmp` in eine Oracle-Datenbank lädt.

Zusätzlich gibt es zu jeder Lektion einen Ordner, der die Skriptdateien für die in jeder Lektion vorgestellten Listings enthält. In jedem Ordner befinden sich darüber hinaus möglicherweise weitere Dateien, die sich für das in der jeweiligen Lektion besprochene Oracle-Tool einsetzen lassen.

## Über dieses Buch

Dieses Buch ist als 21-Tage-Kurs zum Selbststudium gedacht, der Tests und Übungen am Ende jedes Kapitels sowie Beispiele umfaßt, die Sie selbst ausprobieren können. Das Konzept sieht vor, daß Sie drei Wochen lang an jedem Tag der Woche ein Kapitel durcharbeiten können. Sie sollten sich die Zeit jedoch selbst einteilen. Wenn Sie der Ansicht sind, daß Sie jeden Tag zwei oder mehr Kapitel durcharbeiten können, dann nur zu! Sollten Sie andererseits meinen, daß Sie sich einem Kapitel mehr als einen Tag widmen sollten, so dürfen Sie sich auch gerne so viel Zeit nehmen, wie Sie benötigen.

Jede Woche beginnt mit einem Überblick über die Woche und endet mit einem Rückblick darauf. Jeder Tag schließt mit einem Abschnitt F&A, der Fragen und Antworten zu den an diesem Tag angesprochenen Themen enthält. Außerdem gibt es am Ende des Tages jeweils einen Workshop. Ein Test überprüft Ihre Kenntnisse über die am jeweiligen Tag vorgestellten Konzepte, und eine oder mehrere Übungen dienen dazu, daß Sie Ihr Wissen in die Praxis umsetzen. Wir legen Ihnen dringend nahe, diese Abschnitte zu bearbeiten, um das erworbene Wissen zu vertiefen.

## In diesem Buch verwendete Konventionen

Dieses Buch verfügt über spezielle Icons, mit denen wichtige Konzepte und Informationen herausgestrichen werden sollen.



*Ein Hinweis enthält interessante Informationen zum behandelten Thema.*

---



*Ein Tip gibt Ihnen Ratschläge oder zeigt Ihnen einfachere Wege zur Lösung eines Problems auf.*

---



*Ein Achtungszeichen weist Sie auf mögliche Probleme hin und hilft Ihnen, schwierigen Situationen aus dem Wege zu gehen.*

---

---



*Das Symbol »Neuer Begriff« ist den Absätzen hinzugefügt, in denen ein neuer Begriff definiert wird. Der neue Begriff ist kursiv gedruckt, so daß Sie ihn leicht erkennen können.*

---

---



*Das Symbol »Eingabe« kennzeichnet einen Code-Ausschnitt oder ein Listing, bei denen Sie einen Teil oder den kompletten Code selbst eingeben müssen und in den meisten Fällen eine Ausgabe von Oracle enthält. Die Eingabe ist in Fettschrift, die Ausgabe in Standardschrift dargestellt.*

---

---



*Das Symbol »Analyse« kennzeichnet die Erläuterungen und den Zweck des zuvor vorgestellten Listings.*

---

---



*Hier finden Sie Hinweise auf Inhalte der Begleit-CD zum Buch.*

---

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

# 1

# Die Welt relationaler Datenbanken erforschen

[Structured Query Language \(SQL\)](#)

[Die Oracle-Produktreihe](#)

[Weitere Oracle-Produkte](#)

[Was bedeutet Client-Server-Datenverarbeitung?](#)

[Die Ursprünge der Client-Server-Datenverarbeitung](#)

[Die Herausforderung an die Client-Server-Datenverarbeitung: Der Network Computer](#)

[Der Zweck dieses Buchs](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

1970 veröffentlichte »Communications of the ACM«, eine renommierte Zeitschrift der Informatikwelt, einen Aufsatz mit dem Titel »A Relational Model of Data for Large Shared Data Banks«. Der von Dr. E. F. Codd, einem Mitarbeiter des IBM Research Laboratory in San José, verfaßte Aufsatz stellte die theoretische und mathematische Grundlage für das Konzept einer relationalen Datenbank dar. Kaum ein anderer einzelner Artikel aus dem Bereich der Informatik hatte einen so großen Einfluß auf Hersteller, Praktiker und Benutzer wie dieser Beitrag.

---





*Bei einer relationalen Datenbank handelt es sich um ein Informationssystem, welches Daten in Form von Zeilen in einer Reihe von Tabellen darstellt, wobei jede Tabelle über eine oder mehrere Spalten verfügt.*

---

In seinem Aufsatz beschrieb Codd die Elemente einer relationalen Datenbank: Beziehungen, Attribute, Domänen und die Vergleichsoperatoren. Codd's Abhandlung beschrieb ein Datenspeichersystem, das über drei Eigenschaften verfügte, die zu jenem Zeitpunkt dringend benötigt wurden:

- **Logische Datenunabhängigkeit:** Diese wünschenswerte Eigenschaft bedeutet, daß an einem Attribut (Spalte) vorgenommene Änderungen -- zum Beispiel Vergrößerung oder Verkleinerung - keine spürbaren Auswirkungen auf andere Attribute mit derselben Relation (Tabelle) haben. Die logische Datenunabhängigkeit war für Datenverarbeitungsunternehmen reizvoll, da hiermit die Kosten für die Softwarepflege beträchtlich gesenkt werden konnten.
- **Referentielle Integrität und Datenintegrität:** Im Gegensatz zu anderen Datenbanksystemen würde eine relationale Datenbank die Anwendungssoftware von der Last befreien, Integritätsregeln durchsetzen zu müssen. Codd schilderte zwei Merkmale, die von einer relationalen Datenbank gewahrt würden -- die referentielle Integrität und die Datenintegrität. Diese Eigenschaften werden in diesem Buch noch im einzelnen erläutert.
- **Ad-hoc-Abfrage:** Mit Hilfe dieser Eigenschaft könnte der Benutzer der Datenbank vorgeben, welche Daten abgefragt werden sollen, ohne dabei anzugeben, auf welche Weise dies erfolgen soll.

Es ist wichtig, die Grenzen der seinerzeit vorhandenen Datenbanken zu verstehen. Der Durchschnittsbenutzer -- sofern es sich nicht um Programmierer handelte - konnte keine Daten abfragen, die gewissen Kriterien entsprachen, solange kein Programm für diese speziellen Anforderungen geschrieben wurde. Um lediglich die Breite eines vorhandenen Felds, z.B. zipcode (PLZ), von fünf auf neun Zeichen zu erhöhen, mußten die mit der Wartung betrauten Programmierer zahllose Programme ändern, nur um die Feldadressen anzupassen. Viele dieser Programme verfügten möglicherweise noch nicht einmal über einen direkten Verweis auf das Feld zipcode, aber die Auswirkungen durch die Vergrößerung eines Felds machte sich bei diesen älteren Datenbanken im gesamten System bemerkbar.

Es verging noch eine gewisse Zeit, bevor einige der Funktionen der von Codd beschriebenen relationalen Datenbank tatsächlich in ein kommerzielles Produkt implementiert wurden. Anfang der achtziger Jahre wurde die relationale Datenbank zur Grundlage von Entscheidungshilfen. Durch die Leistungen der relationalen Datenbank konnten Benutzer in Unternehmen über interaktive Abfragen und Berichte Daten analysieren, ohne dabei die Hilfe von Programmierern in Anspruch nehmen zu müssen. Dank weiterer Fortschritte im Hardware-Bereich und in der relationalen Technik erfuhren relationale Datenbanken in Transaktionsverarbeitungssystemen Ende der achtziger Jahre eine größere Akzeptanz.

---



*Die heutigen relationalen Datenbanken enthalten eine Reihe besonders nützlicher Funktionen, die Codd in seinem damaligen Artikel nicht erwähnt hat. Zum Zeitpunkt der Drucklegung dieses Buches gibt es auf dem Markt jedoch noch keine Datenbank, in der alle von Codd aufgestellten Regeln für relationale Datenbanken implementiert sind.*

---

Heutzutage sind relationale Datenbanken das Kernstück der Informationssysteme vieler großer und kleiner Organisationen, sowohl auf dem öffentlichen als auch dem privaten Sektor. Viele Hersteller verkaufen heute Management-Systeme für relationale Datenbanken (RDBMS); einige der bekannteren Hersteller sind Oracle, Sybase, IBM, Informix, Microsoft und Computer Associates. Unter diesen Herstellern hat sich Oracle zum Marktführer entwickelt. Das RDBMS-Programm von Oracle wurde auf mehr Plattformen integriert als alle anderen Datenbankprodukte. Da Oracle mehrere Plattformen unterstützt, haben sich viele Hersteller von Anwendungssoftware für Oracle als Datenbankplattform entschieden.



Bei RDBMS (Relational Database Management System) handelt es sich um die von einem Hersteller wie Oracle Corporation bereitgestellte Software, mit deren Hilfe eine relationale Datenbank verwaltet wird. Ein RDBMS unterstützt die Verwendung deklarativer Anweisungen, welche die Regeln beschreiben, denen die Daten entsprechen sollen. Diese Funktion wird als deklarative Integrität bezeichnet. Bei einem RDBMS handelt es sich um die Hauptkomponente einer Client-Server-Architektur, auf die wir in dieser Lektion noch zu sprechen kommen.

---

## Structured Query Language (SQL)

Bei der Structured Query Language (SQL, strukturierte Abfragesprache) handelt es sich um eine nichtprozedurale Sprache; im Gegensatz zu C oder Cobol, in denen Sie den Zugriff auf und die Bearbeitung von Daten genau beschreiben müssen, gibt SQL an, was zu tun ist. Oracle ermittelt intern, wie die Abfrage ablaufen soll. SQL entspricht den Normen des American National Standards Institute (ANSI) und der International Standards Organization (ISO) sowie dem Industriestandard. Die Implementierung von SQL durch Oracle entspricht ANSI X3.135-1989/ISO 9075-1989, Stufe 2, wobei das Leistungsmerkmal der Integritätsverbesserung vollständig integriert ist. Wie andere Datenbankhersteller bietet auch Oracle viele Erweiterungen zur ANSI-SQL.

---



Die Structured Query Language (SQL) ist die offizielle und eigentliche Standardsprache für die Herstellung einer Schnittstelle zu einer relationalen Datenbank.

---

Außerdem entspricht die Implementierung von SQL bei Oracle den Normen der US-Regierung, wie in der Federal Information Processing Standard Publication (FIPS PUB) 127 unter dem Titel *Database Language SQL* beschrieben.

## Die Oracle-Produktreihe

Als weltweit führender Hersteller von Software für relationale Datenbanken unterstützt die Oracle Corporation ihr Aushängeschild, das Oracle-RDBMS, auf mehr als 90 Plattformen. Das Oracle-RDBMS steht in den drei folgenden Konfigurationen zur Verfügung:

- **Oracle Universal Server** kann viele Benutzer auf in hohem Maße skalierbaren Plattformen, wie z.B. Sun, HP, Pyramid und Sequent, unterstützen. Verschiedene Optionen stehen nur bei dieser Konfiguration zur Verfügung. Oracle Universal Server ist für eine Vielzahl von Betriebssystemen und Hardware-Konfigurationen erhältlich. Oracle WebServer - ein integriertes System zur dynamischen Erzeugung von HTML-Dateien aus dem Inhalt einer Oracle-Datenbank - ist ebenfalls im Universal Server enthalten.
- **Oracle Workgroup Server** ist für Arbeitsgruppen bestimmt und für NetWare, Windows NT, SCO Unix sowie UnixWare erhältlich. Oracle Workgroup Server ist ein kostengünstiges und wartungsarmes Produkt für kleine Benutzergruppen. Oracle WebServer steht unter Oracle Workgroup Server als Option zur Verfügung.
- **Personal Oracle** ist die Windows-Version des Oracle-Datenbankprogramms und bietet dieselbe Funktionalität wie Oracle Universal Server und Oracle Workgroup Server. Obwohl Personal Oracle nicht als Datenbank-Server für mehrere Benutzer funktioniert, bietet das Produkt dennoch eine hervorragende Umgebung für Experimente und die

Erstellung von Prototypen.

## Oracle Universal Server

Oracle Universal Server enthält mehrere optionale Komponenten:

- **Die Distributed Option** (Verteilungsoption) ermöglicht es, daß mehrere Datenbanken auf verschiedenen Computern als eine einzige logische Datenbank arbeiten. Im Zuge einer Transaktion können zum Beispiel mehrere verschiedene Datenbanken mit Hilfe eines als *Two-Phase-Commit* bezeichneten Verfahrens geändert werden; dadurch wird garantiert, daß die erforderlichen Änderungen entweder in allen Datenbanken erfolgreich durchgeführt oder alle rückgängig gemacht werden.
- **Die Replication Option** (Replikationsoption) ermöglicht es der Oracle-Datenbank, Änderungen einer Oracle-Datenbank an andere Datenbanken weiterzureichen.
- **Die Context Option** (Kontextoption) erweitert die Möglichkeiten des Benutzers, den Inhalt einer Oracle-Datenbank nach bestimmten Schlüsselwörtern und Themen zu durchsuchen.
- **Die Spatial Data Option** (Option räumliche Daten) ermöglicht es dem Datenbankentwickler, die Erstellung spezieller Indizes vorzusehen, welche komplizierte räumliche, zeitliche und sonstige Abfragen unterstützen.

## Oracle Workgroup Server

Oracle Workgroup Server ist für kleinere Benutzergruppen bestimmt als Oracle Universal Server. Eine typische Anlage für Oracle Workgroup Server könnte aus einem Pentium-Server der Oberklasse mit Microsoft NT und Unterstützung von bis zu 25 Benutzern bestehen. Der Workgroup Server konfiguriert sich automatisch, so daß die Aufgaben der Datenbankverwaltung auf ein Minimum reduziert werden.

## Personal Oracle

Personal Oracle stellt beinahe dieselben Funktionen bereit, die Oracle8 Server auf größeren Plattformen bietet. Somit können Sie Personal Oracle für die Erstellung einer funktionsfähigen Datenbankanwendung einsetzen, die Sie dann später auf eine Mehrbenutzerversion von Oracle8 Server übertragen können. Sie können Personal Oracle zur Erstellung aller Tabellen, Indizes, Datensichten, Sequenzen und anderer Datenbankobjekte verwenden, die in einer produktionsorientierten Datenbankanwendung existieren. Zur Zeit ist Personal Oracle für drei Plattformen verfügbar - Windows 3.11, Windows 95 und Windows NT. Personal Oracle stellt ein Werkzeug namens Navigator bereit, bei dem es sich um ein intuitives grafisches Werkzeug zur Verwaltung der Objekte in einer Personal-Oracle-Datenbank sowie in Oracle-Ferndatenbanken handelt. Mit Hilfe des Navigators können Sie Fernverbindungen definieren, mit denen Sie eine Oracle-Ferndatenbank, sei es auf einem Unix-, NT- oder sonstigen System, bearbeiten können.

## Weitere Oracle-Produkte

Außer dem Oracle-RDBMS erstellt und vertreibt die Oracle Corporation noch eine weitere Software-Produktfamilie:

- Bei **Developer/2000** handelt es sich um eine Werkzeugfamilie, welche die Entwicklung von Client-Server-Anwendungen unterstützt. Jedes dieser Werkzeuge wird von drei verschiedenen GUI-Umgebungen unterstützt: Windows, Mac und Motif. Mehr über Developer/2000 erfahren Sie an den Tagen 12 bis 16. Developer/2000 umfaßt folgende Werkzeuge:
  - Oracle Forms ist ein Unternehmenswerkzeug zur Erstellung formularbezogener Anwendungen, welche gut in die Oracle-Datenbank integriert werden.

- Oracle Reports ist ein Berichtsgenerator, der auch den komplexesten Anforderungen gerecht wird. Diese Berichte können vor dem Ausdrucken mit Hilfe der Vorschaufunktion auf dem Bildschirm angezeigt werden.
- Oracle Graphics ist ein Werkzeug, um aus einer Datenbankabfrage eine Präsentationsgrafik zu erzeugen.
- Oracle Procedure Builder wird zur Entwicklung, zur Pflege und zum Testen von Software verwendet, die in einer Sprache namens PL/SQL geschrieben wurde, welche aus einer Oracle-Datenbank heraus gespeichert und ausgeführt werden kann.



*Bei PL/SQL handelt es sich um eine von der Oracle Corporation entwickelte Programmiersprache, welche prozedurale Erweiterungen für SQL zur Verfügung stellt. PL/SQL-Module sind in Oracle-Datenbanken enthalten, um Firmenregeln durchzusetzen, Ausnahmen zu verwalten und die Funktionen bereitzustellen, die von anderen Werkzeugen für die Anwendungsentwicklung aktiviert werden können.*

- **Power Objects** ist eine weitere Umgebung für die Anwendungsentwicklung, die von Windows- und Mac-Umgebungen unterstützt wird. Die Benutzeroberfläche von Power Objects erinnert eher an Visual Basic als an Oracle Forms. Power Objects bietet auch objektorientierte Funktionen, wie z.B. die Vererbung.



*Unter Vererbung versteht man das Verfahren, mit dem die Instances einer Klasse alle Eigenschaften und Methoden der Klasse ableiten können.*

- **Designer/2000** ist eine Familie von Werkzeugen für die Entwicklung komplexer Anwendungen, die sich über viele Organisationen innerhalb eines Unternehmens erstrecken können. Jedes Werkzeug unterstützt unterschiedliche Aufgaben innerhalb der Entwicklungsorganisation - Analytiker, Datenbankentwickler, Anwendungsentwickler und Programmierer. Die Komponenten von Designer/2000 unterstützen die Entwicklung logischer Modelle, die automatische Erstellung von Datenbanken, die Entwicklung unternehmensspezifischer Funktionen und die Generierung von Anwendungsobjekten, wie z.B. Oracle-Forms-Anwendungen, Visual-Basic-Programme und Web-Anwendungen.
- **Database Designer** ist ein Werkzeug für die grafische Darstellung von Entitätsbeziehungen, mit dessen Hilfe eine Oracle-Datenbank aus einem Modell erzeugt wird. Bei Database Designer handelt es sich um ein Entwicklungswerkzeug für Einzelplatzsysteme, das eng mit einer der Komponenten von Designer/2000 verwandt ist.
- **Precompiler wie Pro\*C** sind Werkzeuge für die Software-Entwicklung, mit denen Sie SQL-Anweisungen in ein Programm einbetten können, das in einer Sprache der dritten Generation (3GL), wie z.B. C, Fortran oder Cobol, geschrieben wurde. Der entsprechende Oracle-Precompiler übersetzt die eingebettete SQL-Anweisung in einen Aufruf der Oracle-Bibliotheksfunktion. Das übersetzte Programm kann in Übereinstimmung mit den Anforderungen durch Sprache und Betriebssystem erstellt werden.

## Was bedeutet Client-Server-Datenverarbeitung?

In dieser Lektion erkunden Sie sowohl die Entwicklung der Architektur einer Client-Server-Datenverarbeitung als auch deren Zukunftsaussichten. Die meisten Werkzeuge für die Anwendungsentwicklung, über die wir in diesem Buch sprechen, unterstützen die Architektur einer Client-Server-Datenverarbeitung. Lassen Sie uns also gleich klären, worum es sich bei der

Architektur einer Client-Server-Datenverarbeitung handelt.

---



*Die Architektur einer Client-Server-Datenverarbeitung besteht aus einem oder mehreren Computern, die als Client-Computer bezeichnet werden, auf denen ein Anwendungsprogramm läuft, welches mit einem entfernten Rechner kommuniziert, der als Server-Computer bezeichnet wird und Anfragen der Client-Systeme bearbeitet.*

---



*Beim Grundmodell einer Client-Server-Architektur befindet sich ein Management-System für relationale Datenbanken (RDBMS) auf dem Server-Computer. Das auf dem Client-System befindliche Anwendungsprogramm ist über eine Schnittstelle mit einer weiteren Softwareschicht verbunden, die als Middleware bezeichnet wird und für die Übermittlung von Anfragen und deren Ergebnissen zwischen Anwendungsprogramm und RDBMS zuständig ist.*

---

Die Client-Server-Architektur bietet mehrere Vorteile gegenüber früheren Datenverarbeitungsarchitekturen:

- Sie unterstützt eine heterogene Mischung von Client-Systemen. Heutzutage können die Benutzer Windows 3.11, Windows 95, Windows NT, einen Mac oder eine Unix-Workstation verwenden, welche alle vom selben Server unterstützt werden können.
- Die Aufgaben der Datenverarbeitung werden sinnvoll auf Client und Server verteilt. Der Client-Computer ist für die Steuerung der Benutzeroberfläche zuständig - die Anzeige von Daten, die Validierung von Eingaben und die Bereitstellung aussagefähiger Rückmeldungen -, während die Server-Plattform normalerweise als Datenbank-Server dient.
- Sie ist unabhängig vom Netzwerkprotokoll, vom Betriebssystem der Server-Plattform und vom Betriebssystem der Client-Plattform. Durch diese Unabhängigkeit ergibt sich eine enorme Flexibilität bei der Auswahl der Hard- und Softwarekomponenten für ein neues System oder bei der Implementierung einer Client-Server-Anwendung in vorhandene Hardware.

Bei einer Client-Server-Architektur lastet die Unterstützung verschiedener Netzwerkprotokolle (wie z.B. TCP/IP, SPX/IPX und Named Pipes) und Betriebssysteme (wie z.B. Windows, Mac, Unix u.a.) auf den Schultern der einzelnen Software-Hersteller, d.h. der RDBMS-Hersteller, wie z.B. Oracle, Sybase, Microsoft und Informix; der Werkzeughersteller, wie z.B. Oracle, Powersoft und Microsoft und anderer Hersteller, die Komponenten wie ODBC-Treiber für besondere Plattformen und RDBMS liefern.

## Die Ursprünge der Client-Server-Datenverarbeitung

Um die Vorteile der Client-Server-Architektur einschätzen zu können, ist es notwendig, sich die frühere Datenverarbeitungsarchitektur vor Augen zu führen. Vor der Einführung des PC setzte sich ein Computersystem normalerweise aus einem Großrechner und vielen darauf zugreifenden »nichtprogrammierbaren« Terminals zusammen. Ein nichtprogrammierbares Terminal verfügte über keinen eigenen Speicher und keine Verarbeitungsmöglichkeiten; jedes auf dem Bildschirm erscheinende Zeichen wurde von einem Programm auf dem Großrechner gesteuert. Diese Architektur brachte einige Begleiterscheinungen mit sich:

- Der Großrechner war für alles zuständig: das Speichern und Abfragen von Daten, den Aufbau der Benutzeroberfläche

auf allen Terminals und die Validierung der Benutzereingaben. Als sich die Anzahl der aktiven Terminals erhöhte, wuchs auch die Belastung des Großrechners durch Ein- und Ausgaben. Um mit der Arbeitsbelastung fertigzuwerden, waren normalerweise Hardware-Lösungen wie Terminalkonzentratoren und Multiplexer erforderlich.

- Die Benutzer waren vom Großrechner vollkommen abhängig. War der Großrechner abgeschaltet, fand keine Datenverarbeitung statt; ein nichtprogrammierbares Terminal war ohne den Computer, an den es angeschlossen war, wertlos. In den siebziger Jahren wurden jedoch »programmierbare« Terminals eingeführt, welche über eigenen Speicher und begrenzte Verarbeitungsmöglichkeiten verfügten, die von einem klugen Software-Entwickler nutzbar gemacht werden konnten.
- Ein Anwendungsprogramm brachte Farbe auf den Bildschirm, nahm Anfragen von den Benutzern entgegen, z.B. über die Menüauswahl, validierte Benutzereingaben und verwaltete die Datenspeicherung. Angesichts dieser Aufgabenmenge war ein Anwendungsprogramm normalerweise recht komplex gestaltet.

Aufgrund der Hardware-Kosten und der Kenntnisse, die erforderlich waren, um einen Computer zum Laufen zu bringen, wurden die Rechnerressourcen der meisten großen Organisationen zentral durch eine einzige Gruppe gesteuert, die Abteilung für elektronische Datenverarbeitung (EDV). Angesichts der Komplexität der Anwendungsprogramme bestand ein beträchtlicher Rückstand bei der Entwicklung neuer Anwendungen, den man normalerweise in Jahren messen konnte. Mit der Einführung des PC änderte sich das alles schnell.

## Die Geburtsstunde des PC

Im August 1981 stellte IBM den Personalcomputer IBM 5150 PC vor. Natürlich war der IBM-PC nicht der erste Personalcomputer der Welt, es besteht aber kein Zweifel, daß IBM dem Wert eines Personalcomputers seine Rechtmäßigkeit verliehen hat. Verglichen mit dem heutigen Standard war der erste PC recht bescheiden mit einem 8088-Prozessor von Intel mit 4,77 MHz, 64 Kbyte RAM, einem Schwarzweißbildschirm und einem 5,25"-Diskettenlaufwerk zur permanenten Speicherung ausgerüstet.

Eines der ersten für den PC geschriebenen Programme war ein Terminalemulator, mit dem der PC ein Terminal des Types IBM 3270 oder DEC VT100 simulieren konnte. Im Vergleich zu einem nichtprogrammierbaren Terminal war der erste PC recht teuer. Einige Büroanwendungen, wie z.B. WordStar (Textverarbeitung) und VisiCalc (das erste Tabellenkalkulationsprogramm), gewannen jedoch zunehmend an Beliebtheit. Daher konnte ein PC einen doppelten Zweck erfüllen:

- Er konnte als Terminal verwendet werden und bot damit den Vorteil, daß von der Terminalsitzung ein Protokoll aufgezeichnet, gedruckt oder in ein Dokument eingebunden werden konnte.
- Mit Hilfe eines Standardprogramms, durch das die Produktivität des Benutzers beträchtlich erhöht wurde, konnte er selbständig eingesetzt werden.

## PC-Datenbanken

Ebenfalls 1981 stellte Ashton-Tate dBASE II vor, ein Datenbankprogramm für Einzelplatz-PCs. Zu der Zeit waren die meisten EDV-Abteilungen mit der Entwicklung neuer Anwendungen stark im Rückstand. Außerdem war die Entwicklung - und der Einsatz - einer neuen Anwendung für Großrechner eine teure und komplexe Angelegenheit. Die Entwicklungs- und Betriebsaufwendungen enthielten beträchtliche Kosten für die Betriebsstunden und permanente Datenspeicherung. Der PC stellte für den Benutzer eine attraktive Alternative dar - den Kauf eines PC und des dBASE-Programms sowie die Entwicklung eines kundenspezifischen dBASE-Programms, über das er die Kontrolle hatte.

## LAN-Datenbanken

Als der PC zunehmend an Popularität gewann, suchten Organisationen mit mehreren PCs eine bequeme, kostengünstige Lösung, mit der die Geräte untereinander kommunizieren und teure Ressourcen, wie z.B. Massenspeicher, Modems und



Drucker, gemeinsam nutzen konnten. Das Local Area Network (LAN, lokales Netzwerk) - eine Kombination von Netzwerkkarten, Kabeln und Software-Treibern - erfüllte diese Anforderungen. Mit wachsender Akzeptanz des LAN auf dem Markt wünschten sich die Benutzer eine Datenbank, auf die mehrere PCs zugreifen konnten. Bald kamen Mehrbenutzerversionen von dBASE und anderen Dateiverwaltungssystemen, wie z.B. Paradox und FoxPro, auf den Markt.

Zu jener Zeit wurden sowohl von Software-Herstellern als auch von internen Programmierern viele Einzel- und Mehrbenutzeranwendungen entwickelt. Auf diesen Werkzeugen basierende Anwendungen brachten enorme Vorteile mit sich, darunter auch einen ausgezeichneten Investitionsertrag im Vergleich zu ähnlichen, für Großrechner bestimmten Anwendungen. Es gab allerdings auch einige negative Aspekte dabei:

- **Ungleichmäßige Qualität der entwickelten Software:** Da diese Werkzeuge relativ leicht zu bedienen waren, konnte ein Neuling eine Anwendung erstellen, die mangelhaft entwickelt war. Obwohl sie vielleicht den ursprünglichen Anforderungen gerecht wurde, war die Anwendung so aufgebaut, daß künftige Änderungen einen beträchtlichen Aufwand erforderten. Außerdem war für viele dieser Organisationen das Konzept der Software-Entwicklung neu; da sie wenig Erfahrungen mit dem Verfahren und der Phase der Software-Entwicklung hatten, wurde viele wichtige Schritte, wie z.B. Dokumentationsbedarf und -entwurf, nicht berücksichtigt; das erschwerte die Pflege des Programms, wenn dessen eigentliche Entwickler nicht mehr zur Verfügung standen.
- **Spürbarer Verlust an Einfluß durch die EDV-Abteilung:** Die meisten werden sagen, daß der Verlust von Einfluß und Befugnissen auf seiten der EDV-Abteilung ein positiver Aspekt bei der Ausbreitung von PC-Anwendungen war. Einige EDV-Abteilungen waren proaktiv; sie erkannten den Wert der PC-Revolution und leiteten ihre Benutzergemeinde in diese Richtung. Viele EDV-Abteilungen betrachteten diese Entwicklung jedoch als ernsthafte politische und wirtschaftliche Bedrohung und reagierten mit der Aufstellung von Regeln, die, sofern sie angemessen waren, als taktisches Mittel eingesetzt wurden, um die Kontrolle zurückzuerlangen.
- **»Inseln der Automatisierung«:** Ein berechtigter Einwand von seiten der EDV-Abteilung war, daß wertvolle Unternehmensdaten in separaten PC-Datenbanken stillgelegt würden und die darin enthaltenen Informationen nicht gemeinsam genutzt werden könnten.

## Die Entstehung relationaler Datenbanken

Als Mitte bis Ende der achtziger Jahre die PC-Datenbanken anwuchsen, verzeichnete sich in der EDV-Abteilung eine weitere dramatische Veränderung. In den ersten Jahren wurden relationale Datenbanken hauptsächlich für den Aufbau einer Entscheidungshilfe (DSS) verwendet. Die Verkaufsabteilung konnte zum Beispiel eine relationale Datenbank mit Informationen über den Verkauf bestimmter Produktreihen in den einzelnen Verkaufsgebieten füttern. Mit Hilfe von SQL konnte ein Vertriebsanalytiker Anfragen an die Datenbank richten, bei denen nur wenige oder überhaupt keine Programmierungsarbeiten notwendig waren.

## Middleware



*Mitte der achtziger Jahre erkannten die Hersteller relationaler Datenbanken, wie sinnvoll es ist, die Verarbeitung einer Anwendung auf zwei Geräte aufzuteilen: einen Client-Computer, der für die Steuerung der Benutzeroberfläche zuständig sein sollte, und einen Server-Computer, der das Management-System für relationale Datenbanken verarbeiten sollte. Um dies zu erreichen, wurde eine Software-Kategorie namens Middleware erfunden.*

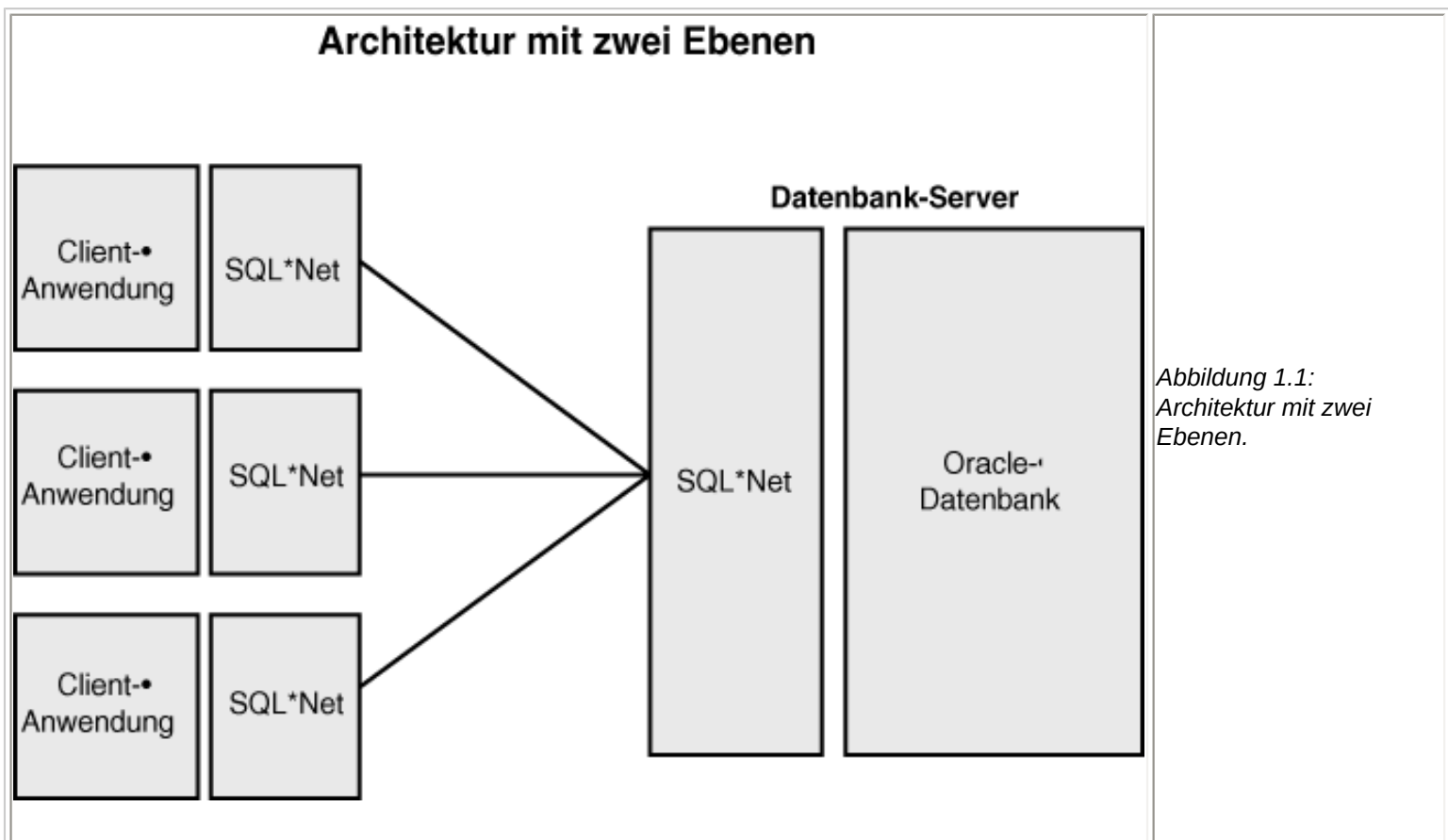
---

Jeder RDBMS-Hersteller erstellte zwei Middleware-Komponenten: einen proprietären Client-Treiber und einen entsprechenden proprietären Server-Treiber. Jedes Netzwerkprotokoll und jedes Client- oder Server-Betriebssystem

erforderte eine besondere Implementierung durch den RDBMS-Hersteller. Allerdings bot die Entwicklung von Middleware einige ernstzunehmende Vorteile für Anwendungsentwickler: Mit Hilfe der Middleware konnte ein Anwendungsprogramm auf mehreren Client-Computern eingesetzt werden, von denen jeder über ein anderes Betriebssystem verfügte, um mit einer Datenbank auf einem Server-System zu kommunizieren, auf dem wiederum ein anderes Betriebssystem installiert war. Das Middleware-Produkt von Oracle heißt SQL\*Net.



Bei einer Architektur mit zwei Ebenen handelt es sich um eine Architektur für die Client-Server-Datenverarbeitung, die aus Client-Systemen besteht, welche direkt mit dem Datenbank-Server kommunizieren (siehe Abbildung 1.1).



Bei einer Architektur mit drei Ebenen handelt es sich um eine Architektur für die Client-Server-Datenverarbeitung, die aus Client-Systemen besteht, welche mit einem Anwendungs-Server kommunizieren. Der Anwendungs-Server könnte eine Oracle-Datenbank enthalten, in der in PL/SQL geschriebene Programmblöcke gespeichert sind und vom Anwendungsprogramm des Client aufgerufen werden können. Diese gespeicherten Programmblöcke kommunizieren mit dem Datenbank-Server, der sich auf einem separaten System befindet. Die Architektur mit drei Ebenen wird im allgemeinen verwendet, um die Verarbeitungsbelastung auf den Server-Systemen auszugleichen (siehe Abbildung 1.2).



## Architektur mit drei Ebenen

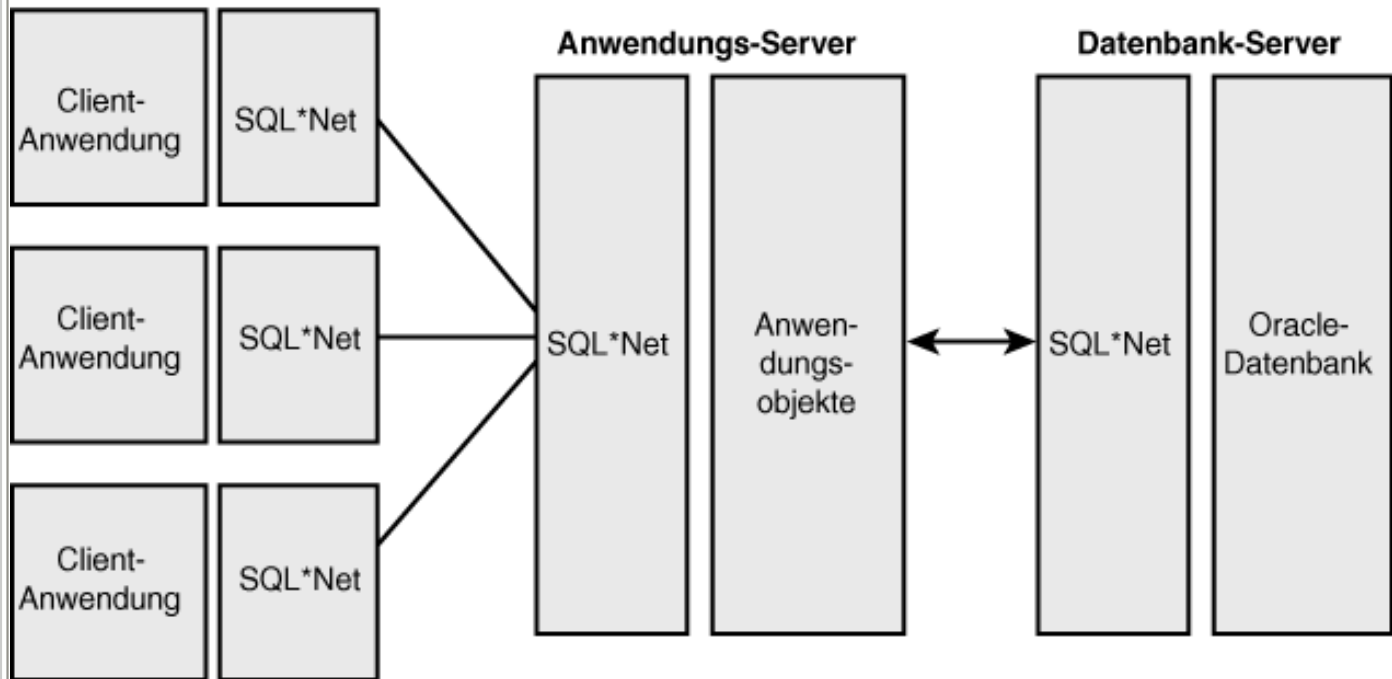


Abbildung 1.2: Architektur mit drei Ebenen.

## Werkzeuge für die Anwendungsentwicklung

Anfangs wurden die meisten Client-Server-Anwendungen in Sprachen der dritten Generation (3GL) erstellt, wie z.B. Fortran, C oder Cobol, mit denen Bibliotheksroutinen der RDBMS aufgerufen wurden. Einige RDBMS-Hersteller, wie z.B. Oracle, sahen allerdings die Notwendigkeit für ein Werkzeug, das die Entwicklung von Datenbank Anwendungen reibungsloser gestalten könnte.



Als Ergebnis wurde eine neue Kategorie von Entwicklungssoftware geschaffen: die Sprachen der vierten Generation (4GL), deren Name richtigerweise zu erkennen gibt, daß es sich bei diesen Werkzeugen um eine höhere Abstraktionsebene handelt als bei 3GL. Heute werden 4GLs allgemein als Umgebungen für die Anwendungsentwicklung bezeichnet.

Anfangs erzeugten 4GLs eine zeichenorientierte Benutzeroberfläche. Mit wachsender Akzeptanz der Windows-Plattform und der zunehmenden Verbreitung von VGA- und SVGA-Grafikkarten begannen die 4GL-Hersteller, grafische Benutzeroberflächen (GUIs), wie z.B. Windows, Mac und Motif, zu unterstützen. Zu Beginn der neunziger Jahre wurden mit diesen Werkzeugen Anwendungen erstellt, die recht langsam und fehlerhaft waren. Seitdem wurden sowohl in der Leistung des Betriebssystems (z.B. Upgrade von Windows 3.11 auf Windows 95) als auch der Hardware viele Verbesserungen erzielt, so daß es möglich ist, zuverlässige und gut funktionierende Anwendungen zu erstellen.

## »Thick«-Client und »Thin«-Client

1996 war die Entwicklung von Client-Server-Anwendungen ausgereift. Organisationen verstanden, wie Umgebungen für die Anwendungsentwicklung, z.B. Oracle Forms und PowerBuilder, bei der Erstellung einer Anwendung einzusetzen waren. Für große Organisationen war die Verwaltung einer Client-Server-Anwendung jedoch keine leichte Übung:

- Für jede Client-Plattform mußten unterschiedliche Anwendungsversionen erstellt werden.
- Wenn eine neue Version der Anwendung herauskam, mußte die Software an alle Client-Systeme verteilt werden.
- Wenn ein neuer Benutzer ein Anwendungsprogramm auf seinen Computer aufspielen wollte, mußte die richtige Middleware-Version für dieses System installiert und konfiguriert werden.

Eine umfangreiche Studie der Gartner Group stellte fest, daß die jährlichen Durchschnittskosten für die *Verwaltung* eines PC ca. DM 22.000 betragen. Viele Führungskräfte betrachteten diese Probleme besonders in finanzieller und administrativer Hinsicht als Alptraum - und sie suchten nach einem Ausweg.

Gleichzeitig fanden zwei Ereignisse statt, die man nicht außer acht lassen darf:

- Die außerordentliche Akzeptanz des Internet und des World Wide Web
- Die Einführung und Übernahme der Programmiersprache Java durch Hardware- und Software-Hersteller

Die explosionsartige Verbreitung des World Wide Web und der Web-Browser veranlaßte die RDBMS-Hersteller, wie z.B. Oracle, zur Entwicklung von Werkzeugen, welche die dynamische Darstellung von Datenbankinhalten im Web unterstützen sollten. Gleichzeitig zog Java - als plattformunabhängige, objektorientierte, verteilte und sichere Sprache - die Software-Hersteller an, die seit langem daran interessiert waren, Software einmal zu schreiben und diese dann auf einer Vielzahl von Hardware-Plattformen und unterschiedlichen Betriebssystemen einzusetzen. Angesichts dieser beiden Bewegungen begannen viele Software-Hersteller, das Modell des »dünnen« Client zu befürworten: eine Benutzeroberfläche auf der Basis eines Web-Browsers, auf die ein Java-Zusatzprogramm von einem Server heruntergeladen wird, um dann auf dem Client-Computer ausgeführt zu werden. Die traditionelle Client-Server-Architektur wurde als »dickes« Client-Modell bezeichnet, da der Client normalerweise eine große ausführbare Datei für das Anwendungsprogramm (z.B. mybigapp.exe), mehrere Bibliotheken (z.B. DLLs) sowie die Middleware-Treiber (z.B. SQL\*Net) benötigte.

Die Argumente für die Unterstützung des »dünnen« Client-Modells lauten wie folgt:

- Die Verwaltungskosten für Client-Systeme werden beträchtlich gesenkt.
- Anwendungen werden einmal geschrieben und können überall eingesetzt werden.
- Die Aktualisierung von Anwendungen erfolgt auf nur einem System: dem Server.
- Übertragbarkeit: Statt an einen bestimmten Client-Computer gebunden zu sein, »folgt« eine Anwendung dem berechtigten Benutzer.

Allerdings gibt es auch Bedenken hinsichtlich der Wirtschaftlichkeit des dünnen Client-Modells:

- Da die gesamte Anwendungssoftware von einem Server heruntergeladen wird, erhöht das dünne Client-Modell die Netzauslastung.
- Da es sich bei Java um eine im Byte-Code vorliegende Sprache handelt, ist sie nicht so schnell wie andere Sprachen; Leistung könnte ein wichtiger Punkt sein.

## CORBA

1989 wurde die Object Management Group (OMG) von Vertretern der Hard- und Software-Hersteller, Entwicklern und Software-Praktikern gegründet. Das Ziel der OMG war es, die Verwendung objektorientierter Software zu fördern. Zu diesem Zweck entwickelte die OMG eine begriffliche Grundlage namens Object Management Architecture. Auf dieser Grundlage wurde eine Spezifikation für die Common Object Request Broker Architecture, allgemein als CORBA bezeichnet, entwickelt.

Die Spezifikation von CORBA beschreibt mehrere Komponenten:

- Einen Object Request Broker (ORB), der Anfragen und Antworten zwischen Objekten in einer heterogenen, verteilten Umgebung unterstützt.
- Object Services, bei denen es sich um eine Reihe von Grunddiensten handelt, welche die Verwendung und Implementierung von Objekten unterstützen.
- Common Facilities, bei denen es sich um eine Reihe von Diensten handelt, die von Anwendungen gemeinsam benutzt werden können.
- Application Objects, bei denen es sich um vom Hersteller oder Entwickler bereitgestellte Objekte mit einer definierten Oberfläche handelt.



*Zur Unterstützung der Spezifikation von CORBA 2.0 hat Oracle eine Architektur namens Network Computing Architecture (NCA) vorgestellt. Die neueste Version von Oracle WebServer, Version 3.0, wurde den Vorgaben der NCA angepaßt. Mehr über NCA erfahren Sie am Tag 21, »Oracle: Die nächste Generation«.*

---

## Die Herausforderung an die Client-Server-Datenverarbeitung: Der Network Computer

Eine Zeitlang hat sich Oracle-Chef Larry Ellison beklagt, daß PCs zu teuer und komplex für den Durchschnittsanwender seien. Sein Wunsch wäre ein Gerät namens *Network Computer*, das weniger kosten würde als ein typischer PC und weit einfacher zu verwalten wäre, da die gesamte Software über einen Web-Browser heruntergeladen würde.

Im Jahr 1996 gründete die Oracle Corporation eine Tochtergesellschaft, Network Computer, Incorporated. Die Aufgabe dieser Tochtergesellschaft ist es, für den Entwurf ihres Network Computers Lizenzen an Hardware-Hersteller zu vergeben und die Entwicklung von Anwendungen für dieses Gerät zu fördern.

## Der Zweck dieses Buchs

Es richtet sich an Leser aus einer der folgenden Kategorien:

- Sie sind mit Datenbanken wie dBASE oder FoxPro vertraut, haben aber noch nicht mit Oracle oder anderen SQL-basierten Datenbanken gearbeitet.
- Sie haben einige Erfahrungen mit SQL, möchten aber noch mehr über ihre Verwendung in einer Oracle-Datenbank wissen.
- Sie sind mit 3GLs wie Cobol vertraut, haben aber keine Erfahrungen mit einer Client-Server-Umgebung und möchten mehr über Umgebungen für die Anwendungsentwicklung, wie zum Beispiel Developer/2000 oder Power Objects, erfahren.



*Um die in diesem Buch genannten Begriffe zu verdeutlichen, zeigen wir Ihnen viele Beispiele anhand eines Datenbankentwurfs für ein kleines College. Die Beispiele aus dieser Datenbank werden durch andere spezielle Beispiele zum jeweiligen Thema ergänzt. Die CD-ROM enthält diese Beispieldatenbank sowie entsprechende Dateien aus den einzelnen Kapiteln des Buchs. Weitere Informationen über die CD-ROM entnehmen Sie bitte der Einleitung.*

Wenn Sie die Lektüre dieses Buchs beendet haben, werden Sie mit folgenden Sprachen, Werkzeugen und Themen vertraut sein:

- **SQL:** Nach Industrienorm und ANSI die eigentliche Standardsprache, um mit relationalen Datenbanken zu kommunizieren. Mehr über SQL erfahren Sie an den Tagen 4 bis 8.
- **PL/SQL:** Die prozeduralen Spracherweiterungen zu SQL für das Schreiben von Datenbanktriggern, gespeicherten Prozeduren und Funktionen. An den Tagen 9 bis 11 erhalten Sie eine solide Einführung in PL/SQL.
- **Developer/2000:** Sie sollten fähig sein, mit Oracle Forms eine Grundanwendung zu entwickeln, mit Oracle Reports einen Master-/Detail-Bericht zu erstellen und mit Hilfe von Procedure Builder einen Datenbanktrigger oder ein Paket zu erstellen. Developer/2000 lernen Sie in fünf Lektionen an den Tagen 12 bis 16 kennen.
- **Power Objects:** Sie werden in der Lage sein, mit Power Objects eine Anwendung mit der Möglichkeit zur Dateneingabe, -abfrage und Berichtserstellung zu entwickeln; mit der Verwendung von Power Objects verbringen Sie zwei Lektionen an den Tagen 17 und 18.

Um aus diesem Buch größtmöglichen Nutzen zu ziehen, sollten Sie zumindest eine Oracle-Datenbank - entweder Personal Oracle7, einen Oracle Workgroup Server oder einen Oracle Universal Server - und SQL\*Plus zur Verfügung haben. Wahrscheinlich brauchen Sie aber auch Developer/2000 und Power Objects, um den Lektionen folgen zu können, die sich mit diesen Werkzeugen befassen.

## Zusammenfassung

Die wichtigsten Gedanken aus dem Bereich der in dieser Lektion vorgestellten Client-Server-Datenverarbeitung lauten wie folgt:

- Die theoretische Grundlage für relationale Datenbanken wurde im Jahr 1970 geschaffen.
- Die Structured Query Language (SQL) ist eine offizielle Standardsprache für die Kommunikation mit relationalen Datenbanken.
- Es gibt drei Konfigurationen des Management-Systems für relationale Datenbanken (RDBMS) von Oracle: Oracle Universal Server, Oracle Workgroup Server und Personal Oracle.
- Die Client-Server-Architektur unterstützt eine große Anzahl von Client-Plattformen, Server-Plattformen und Netzwerkprotokollen.
- Die Client-Server-Architektur erfordert die Verwendung einer Softwareschicht namens Middleware, deren Aufgabe es ist, die protokollunabhängige Kommunikation zwischen dem Client- und dem Server-System zu unterstützen.
- Eine Architektur mit zwei Ebenen besteht aus einem Client-Computer, der direkt mit einem Datenbank-Server kommuniziert.
- Eine Architektur mit drei Ebenen besteht aus einem Client-Computer, der mit einem Anwendungs-Server kommuniziert. Der Anwendungs-Server kommuniziert wiederum mit dem Datenbank-Server.
- Die Popularität des World Wide Web und der Programmiersprache Java wecken das Interesse vieler Software-Entwickler innerhalb eines dünnen Client-Modells: Ein Web-Browser lädt die Java-Zusatzprogramme entsprechend ihrer Notwendigkeit herunter.

## Wie geht es weiter?

Am Tag 2, »Richtlinien für die Entwicklung einer Oracle-Anwendung«, lernen Sie die für die Erstellung einer Oracle-Anwendung erforderlichen grundlegenden Schritte kennen, wie z.B. das Sammeln der von den Benutzern gestellten Anforderungen.

## Fragen und Antworten

Frage:

Oracle Corporation hat Oracle8 auf den Markt gebracht. Werden SQL, PL/SQL und die Oracle-Entwicklungswerkzeuge in Oracle8 weiterhin unterstützt?

Antwort:

*Natürlich. Oracle8 unterstützt SQL, PL/SQL und die Oracle-Entwicklungswerkzeuge. Statt die derzeit in der Oracle-RDBMS verwendeten Funktionen zu ersetzen, erweitert Oracle8 schließlich die vorhandenen Fähigkeiten durch benutzerdefinierte Datentypen, Vererbung, eine integrierte Java Virtual Machine und viele andere Funktionen.*

Frage:

Welche Art der Java-Unterstützung ist von Oracle geplant?

Antwort:

*Oracle hat die Verwendung von Java sowohl intern als auch kundenseitig vorgesehen. Das Oracle8-RDBMS wird endlich eine Java Virtual Machine enthalten. Oracle WebServer 3.0 ermöglicht, daß in PL/SQL gespeicherte Programmblöcke von Java-Zusatzprogrammen eingebunden werden können. Außerdem unterstützt die aktuelle Version von Developer/2000 - 1.4W - die Generation der Java-Zusatzprogramme aus Oracle Forms und Reports.*

Frage:

Welche Umgebungen für die Anwendungsentwicklung werden in diesem Buch besprochen?

Antwort:

*Sie werden drei Umgebungen für die Anwendungsentwicklung kennenlernen: Developer/2000, einschließlich Oracle Forms, Oracle Reports und Oracle Graphics; Oracle Power Objects und PowerBuilder.*

## Workshop

Der Zweck dieses Workshops ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Nennen Sie zwei Vorteile relationaler Datenbanken gegenüber Dateiverwaltungssystemen.
2. Nennen Sie drei Vorteile der Architektur für die Client-Server-Datenverarbeitung im Vergleich zu der Datenverarbeitungsarchitektur von Großrechnern.
3. Wie heißt das Middleware-Produkt von Oracle?
4. Was ist ein »dicker« Client? Was ist ein »dünner« Client?

## Übungen

Wie bereits erwähnt, wird in diesem Buch eine Beispieldatenbank eines kleinen Colleges verwendet. Denken Sie darüber nach, welche Art von Informationen Ihrer Meinung nach ein College speichern und abfragen möchte. Fragen Sie sich, welche Fragen ein Student, ein Professor oder ein Administrator an die Datenbank stellen könnte. Notieren Sie, wie Sie diese Informationen anordnen könnten. Sie werden am Tag 3, »Logischer Datenbankentwurf«, auf diese Informationen zurückkommen.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖·Kapitel ·❖ Top Inhalt Index **SAMS** Kapitel·❖

# 2

# Richtlinien für die Entwicklung einer Oracle-Anwendung

[Anwendungsentwicklung Schritt für Schritt](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

Der Aufwand für die Entwicklung einer neuen Oracle-Anwendung hängt von vielen Faktoren ab. Zum Thema Anwendungsentwicklung wurden bereits viele ausgezeichnete Bücher veröffentlicht. In dieser Lektion werden Sie einige der Überlegungen prüfen, die bei der Entwicklung einer Oracle-Datenbankanwendung angestellt werden.

---



*In dieser Lektion umfaßt der Begriff Benutzer alle an der Anwendung Beteiligten - Endbenutzer, Führungskräfte, Lieferanten, Kunden, Betriebspersonal und andere Entwickler.*

---

## Anwendungsentwicklung Schritt für Schritt

Traurige Wahrheit bei der Anwendungsentwicklung ist, daß die Fehlerquote erstaunlich hoch liegt; die Mehrheit der neuen Anwendungen geht nicht in Produktion. Für Sie als Anwendungsentwickler ist es wichtig zu verstehen, warum dies der Fall



ist und wie die Auswirkungen auf den Entwicklungsprozeß aussehen. Selbst wenn Sie die Vorschläge aus dieser Lektion - oder einer anderen Quelle - peinlich genau befolgen, gibt es keine Garantie, daß die Implementierung der neuen Anwendung erfolgreich verlaufen wird. Denn falls Sie es noch nicht bemerkt haben: Es gibt kein Patentrezept für die Softwareentwicklung.

Im Idealfall sind folgende Schritte für die Entwicklung einer neuen Anwendung erforderlich:

- Sammeln von Anforderungen
- Entwerfen der Anwendung
- Implementieren der Anwendung
- Testen der Anwendung
- Einsetzen der Anwendung
- Pflege der Anwendung

In der Realität folgen diese Schritte selten so nahtlos aufeinander. Statt dessen werden die Arbeitsschritte im allgemeinen mehrfach wiederholt, wobei sich die Anwendung der Implementierung der »tatsächlichen Anforderungen« immer weiter nähert (siehe Abbildung 2.1).

Lassen Sie uns etwas tiefer in die einzelnen Schritte einsteigen.

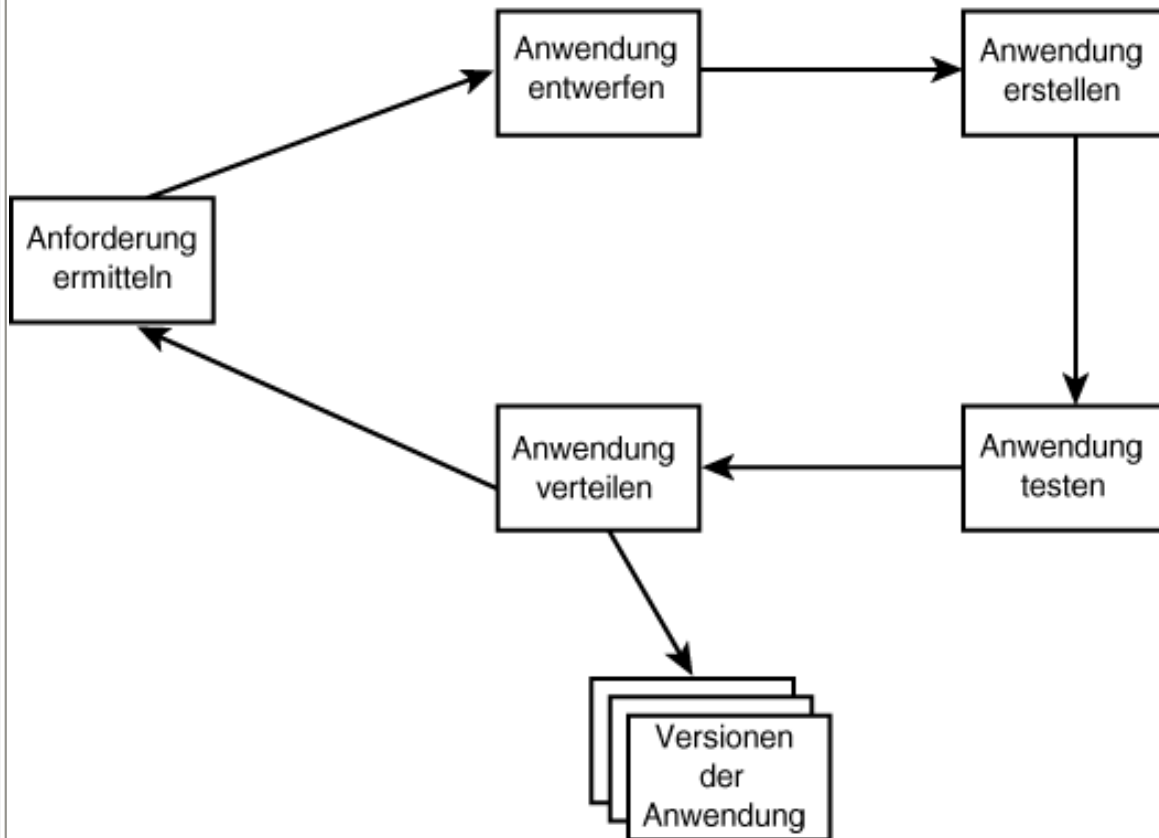


*Bei jedem Projekt gibt es mehrere Beteiligte, wie z.B.: Projektleiter, Unternehmensanalytiker, Datenbankentwickler, Anwendungsentwickler, Programmierer, Fachautor, Datenbankadministrator, Systemadministrator und Schulende. Bei einem sehr kleinen Projekt ruhen alle diese Aufgaben möglicherweise auf Ihren Schultern. Bei einem großen Projekt könnte ein ganzes Team für einen bestimmten Bereich, z.B. Schulung, zuständig sein. Man sollte immer daran denken, daß für jeden dieser Bereiche jemand zuständig sein muß, damit das Projekt zum Erfolg führt.*

---

## Entwicklungszyklen von Anwendungen

Abbildung 2.1:  
Zyklus der  
Anwendungsentwicklung.



### Sammeln von Anforderungen

Der Ausgangspunkt bei der Entwicklung einer neuen Anwendung liegt in der Aufgabe, Anforderungen zu sammeln. Die Aufgabe, die Anforderungen an eine neue Anwendung zu definieren, wird von allen, die einen Einfluß auf die neue Anwendung haben, als notwendig, unkompliziert und einleuchtend angesehen. Das Sammeln von Anforderungen ist eine Kunst, keine Wissenschaft, und bei jedem Schritt auf diesem Weg lauern Fallen:

- **Zwischenmenschliche Kommunikation:** Das Sammeln von Anforderungen erfordert eine gewisse zwischenmenschliche Kommunikation - entweder durch Gespräche, Zuhören, Lesen oder Schreiben. Da die menschliche Sprache ungenau ist, bedarf es der Interpretation. Diese Interpretation gilt sowohl für die anderen als auch für Sie, den Anwendungsentwickler.
- **Politik:** Die Entwicklung einer neuen Anwendung steckt voller politischer Gefahren. Eine Benutzergruppe formuliert möglicherweise eine Anforderung, während eine andere Gruppe eventuell eine dazu im Widerspruch stehende Forderung hat. Nun liegt es bei Ihnen, mit diesen Personen zusammenzuarbeiten und herauszufinden, wer recht hat, wessen Stimme zählt und wer eigentlich letztendlich bei diesem Projekt zufriedengestellt werden soll. Das Risiko eines Projekts im Bereich der Anwendungsentwicklung verhält sich proportional zu der Anzahl der an der Entwicklung beteiligten Organisationen (siehe Abbildung 2.2).
- **Ständige Veränderungen:** Wahrscheinlich entwickeln Sie neue Anwendungen häufig in einer Umgebung, in der sich die Anforderungen ändern. Wenn Sie Glück haben, denkt irgend jemand daran, Sie über die Art der Änderungen, deren Gründe und die Auswirkungen auf die bestehenden Anforderungen zu informieren. Vielleicht sehen Sie es als eine Herausforderung an, eine Architektur zu entwickeln, die diesen sich ändernden Anforderungen gerecht wird.

## Verhältnis zwischen Projektrisiko und organisatorischer Komplexität

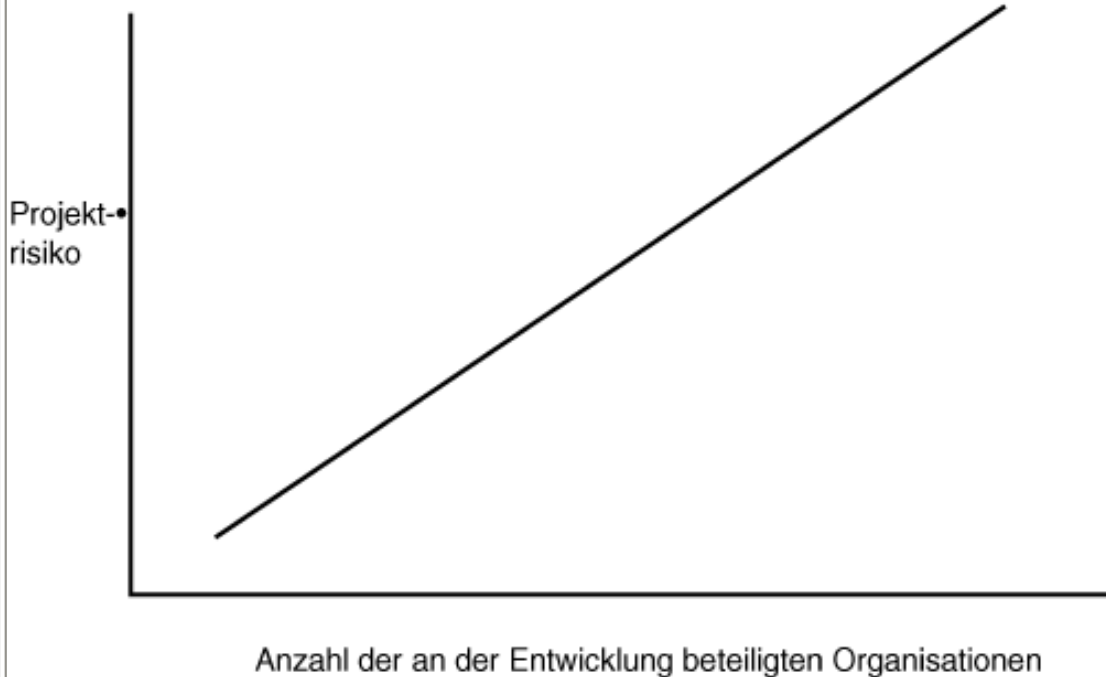


Abbildung 2.2:  
Das Verhältnis  
zwischen Risiko und  
organisatorischer  
Komplexität.

Sie werden sicherlich mehrere Arten von Anforderungen für Ihre Anwendung sammeln wollen:

- **Anforderungen an die Funktionen:** Hierzu gehören sowohl die von der Anwendung zu erfüllenden Anforderungen der höheren als auch der niederen Ebene. Ein Beispiel für eine Anforderung der höheren Ebene wäre: »Das Informationssystem des Flugle-College soll es den Lehrkräften ermöglichen, ihren Studenten Noten zuzuweisen.«
- **Anforderungen an die Daten:** Zu diesen Anforderungen zählen die von der Anwendung zu verwaltenden Informationen. Diese Anforderungen sind entscheidend für die Entwicklung eines logischen Datenmodells, das bei der Implementierung der funktionalen Anforderungen hilfreich ist. Ein Beispiel für Anforderungen an die Daten wäre: »Das Informationssystem des Flugle-College soll Daten über die von den einzelnen Fachabteilungen angebotenen Kurse enthalten.«
- **Anforderungen an die Leistung:** Diese Anforderungen beschreiben die Leistung, welche das gesamte System, einschließlich der Anwendung, erbringen muß. Eine Anforderung an die Leistung wäre zum Beispiel: »Das Informationssystem des Flugle-College soll in der Lage sein, gleichzeitig 120 Benutzer zu unterstützen, und dabei eine Antwortzeit von weniger als zwei Sekunden für die Registrierung eines Studenten für einen Jahrgang aufweisen.«

Es gibt viele Wege, Anforderungen zu verfolgen. Sie können sich natürlich ausführliche Notizen machen. Sie können auch eine Anforderungsdatenbank erstellen und mit deren Hilfe eine Dokumentation anfertigen. Sie können aber auch Komponenten aus Designer/2000 oder andere Werkzeuge für die computergestützte Software-Entwicklung (Computer-Aided Software Engineering, CASE) benutzen. Welche Methode Sie verwenden, um Anforderungen zu sammeln und zu dokumentieren, hängt jedoch von mehreren Punkten ab:

- **Komplexität der Anwendung:** Wird die Anwendung für eine kleine Benutzergruppe oder eine Hauptabteilung einer großen Organisation entwickelt?
- **Budget:** Soll die Anwendung mit geringen Mitteln finanziert werden, oder hat die Geschäftsführung die Finanzierung für dieses Großprojekt genehmigt?
- **Anzahl der mit der Anwendungsentwicklung betrauten Personen:** Sind Sie die einzige Person, die sich vollzeitlich mit der Anwendungsentwicklung beschäftigt (wobei andere nur bei Bedarf mitwirken), oder gibt es einige Dutzend Mitarbeiter, die an der Entwicklung beteiligt sind?
- **Sichtbarkeit:** Ist die zu entwickelnde Anwendung außerhalb eines kleinen Teams unbekannt, oder gibt es

Führungskräfte, die je nach Erfolg der implementierten Anwendung befördert werden oder zurücktreten müssen?

Die Antworten auf diese Fragen entscheiden, wie strukturiert Sie bei der Sammlung von Anforderungen und der Überprüfung Ihrer Vorstellung von diesen Anforderungen mit den Beteiligten - Endbenutzern, Führungskräften und anderen Entwicklern - vorgehen sollten. Stehen für Ihr Projekt klare Forderungen, bescheidene Mittel und wenig Personal zur Verfügung, könnte es sein, daß Sie eine informelle Methode bei der Sammlung und Überprüfung von Anforderungen anwenden möchten: Sie wollen zwar noch alles dokumentieren, allerdings ohne formelle Prüfung und Genehmigungsprozeß. Gibt es für Ihr Projekt jedoch komplexe Anforderungen, beträchtliche Mittel und viel Personal, sollten Sie dem Sammeln von Anforderungen, der Analyse, Validierung und Genehmigung einen formellen Prozeß vorausschicken; wenn Sie bei sich keinen solchen Prozeß erkennen, könnte dies von einem Fehler in der Anfangsphase zeugen.

## Design und Aufbau User Applikation

Irgend jemand hat einmal gesagt, daß bei der Entwicklung eines Systems zehntausend Entscheidungen getroffen und selten aufgezeichnet würden. Es gibt viele wichtige Aspekte bei der Anwendungsentwicklung:

- **Entwickeln Sie ein logisches Datenmodell:** Falls für die Anwendung keine Datenbank existiert, ist ein logisches Datenmodell erforderlich; darüber werden Sie am Tag 3, »Logischer Datenbankentwurf«, noch mehr erfahren. Hierbei handelt es sich um den wahrscheinlich wichtigsten Schritt im Entwicklungsprozeß. Ist das Projekt auch noch so klein, Sie sollten wirklich versuchen, ein Werkzeug zur Herstellung einer Entitätsbeziehung, wie z.B. den Entity-Relationship Diagrammer in Designer/2000, den Oracle Database Designer oder ERwin von Logic Works, zu verwenden; am Tag 19, »Eine Übersicht über die Datenbanksicherheit und die Datenbankoptimierung unter Oracle«, werden Sie lernen, wie man Database Designer einsetzen kann, um ein logisches Datenmodell zu erstellen. Solche Werkzeuge erlauben es Ihnen, ein grafisches Modell der von der Anwendung zu verwaltenden Daten zu konstruieren. Der große Vorteil dieser Werkzeuge liegt darin, daß sie aus dem Modell eine Oracle-Datenbank erstellen. Außerdem stellen sie eine Quelle für die Dokumentierung der von der Anwendung zu bearbeitenden Daten bereit.
- **Wählen Sie Entwicklungswerkzeuge aus:** Sofern nicht andere die Entscheidung für Sie treffen, sollten Sie sich die zu unterstützenden Client-Systeme (z.B. Windows, Mac oder Motif) und die Vorlieben anderer Entwickler anschauen. Natürlich sollten Sie auch Ihre Erfahrungen mit verschiedenen Entwicklungswerkzeugen miteinbeziehen. Bei der Auswahl von Entwicklungswerkzeugen sollten Sie außerdem die vorhandene Benutzerbasis nicht außer acht lassen; die Wahl eines hochwertigen Produkts bei einer kleineren Benutzerbasis ist weniger attraktiv als ein gutes Produkt bei einer sehr großen Benutzerbasis. Falls Sie sich entschlossen haben, ein Werkzeug zu verwenden, mit dem Sie noch nie zuvor gearbeitet haben, sollten Sie sich genügend Zeit lassen, um sich damit vertraut machen zu können; Sie möchten seine Mängel sicher nicht erst dann kennenlernen, wenn sie bereits die halbe Strecke zurückgelegt haben.
- **Implementieren Sie die von den Benutzern geforderten Funktionen:** Im allgemeinen ist es besser, mehrere Versionen der Anwendung für die Benutzer einzuplanen als zu wenige. Mit jeder Version erhöht sich die von den Benutzern erwartete Funktionalität. Jede Version bietet den Benutzern die Gelegenheit, Ihnen entsprechende Rückmeldungen zu geben; darüber hinaus erhalten Sie die Möglichkeit, diese Rückmeldungen in die nächste Version einzuarbeiten. Durch diesen Prozeß »bringen« sich die Benutzer in das Projekt ein; mit der Zeit werden Sie zu Beteiligten. Dies ist entscheidend; wenn die Benutzer die endgültige Version der Anwendung nicht akzeptieren, ist sie zum Scheitern verurteilt.
- **Nutzen Sie wiederverwendbare Komponenten:** Versuchen Sie möglichst, eine Reihe wiederverwendbarer Komponenten einzusetzen. Man braucht nicht zu erwähnen, daß der Grad der Wiederverwendbarkeit von den benutzten Entwicklungswerkzeugen abhängt. Handelt es sich nicht um objektorientierte Werkzeuge, bestehen die wiederverwendbaren Komponenten aus Formular- und Berichtsvorlagen, Bibliotheken und Header-Dateien. Unterstützen die Werkzeuge eine objektorientierte Entwicklung, bestehen die wiederverwendbaren Komponenten aus Vorgängerklassen, aus denen eine funktionierende Anwendung erstellt werden kann. Am Tag 18, »Eine Anwendung mit Oracle Power Objects entwickeln«, werden Sie erfahren, wie Sie mit Power Objects Klassen erzeugen können, um die Pflege einer Anwendung zu erleichtern. Obwohl die Wiederverwendbarkeit ein hochgestecktes Ziel ist, sollten Sie gut überlegen, wieviel Zeit und Energie Sie investieren.
- **Verwenden Sie ein Werkzeug für die Konfigurationsverwaltung:** Selbst wenn der Aufwand für die Entwicklung Ihrer Anwendung nur gering ist, sollten Sie die Verwendung einer Art von Konfigurationsverwaltung in Erwägung ziehen. Zumindest sollten Sie eine Sicherungskopie aller Versionen Ihrer Anwendung entweder auf Band oder

Diskette aufbewahren. Für mittlere oder große Projekte ist dies nicht ausreichend; planen Sie die Verwendung eines Mehrbenutzer-Werkzeugs für die Konfigurationsverwaltung ein, welches die Versionssteuerung unterstützt. Die Konfigurationsverwaltung sollte folgendes umfassen: Quellcodedateien, SQL-Skripts, um die Oracle-Datenbank zu erstellen und mit Daten zu füllen, PL/SQL-Quellcode, Entwürfe, Testdatenreihen, Schulungsmaterial und Textdateien für die Erstellung einer Online-Hilfe.

- **Identifizieren Sie Benutzerrollen und -berechtigungen:** Identifizieren Sie bei der Entwicklung Ihrer Anwendung die wichtigsten Rollen, die von den Benutzern beim Einsatz des Systems eingenommen werden. Sie sollten diese Rollen auf zwei Ebenen implementieren: auf Datenbankebene und auf Anwendungsebene - beides ist wichtig. Sie sollten die Berechtigungen - Erstellen, Lesen, Ändern und Löschen (create, read, update, delete - häufig als *CRUD* bezeichnet) - festlegen, die den einzelnen Rollen für alle Datenbanktabellen, -ansichten oder andere Objekte zugeteilt werden. In Abhängigkeit von dem von Ihnen für die Anwendungsentwicklung verwendeten Werkzeug sollten Sie in der Lage sein, die Anmeldung eines Benutzers einer Rolle und diese Rolle wiederum den dafür zur Verfügung stehenden Menüeinträgen zuzuordnen.
- **Bleiben Sie so weit wie möglich plattformunabhängig:** Es gibt zwei Plattformen, die Sie während der Entwicklungsphase des Projekts berücksichtigen müssen: den Server und den Client. Was den Server anbelangt, sollten Sie für die Implementierung der Anwendungslogik so oft wie möglich PL/SQL verwenden; dadurch verringert sich der erforderliche Aufwand, um die Anwendung von einer Server-Plattform (z.B. Windows NT) auf eine andere Server-Plattform (z.B. Solaris) zu übertragen. Für den Client sollten Sie herausfinden, welche für das Betriebssystem (BS) spezifischen Merkmale Sie bereitstellen müssen. Wenn Sie BS-spezifische Merkmale in die Anwendung einbinden (z.B. OLE), kann die Anwendung diese Eigenschaften nicht auf allen Client-Plattformen (wie z.B. Mac und Motif) zur Verfügung stellen. Das bedeutet, Sie würden einige Benutzer mehr erfreuen als andere.
- **Sorgen Sie für eine konsistente Benutzeroberfläche:** Alle Bildschirm- und Berichtskategorien sollten ein ähnliches Erscheinungsbild aufweisen. Beispielsweise sollten alle Eingabebildschirme über dieselben Schaltflächen verfügen, die etwa an derselben Stelle liegen und dieselben Funktionen haben. Für Ihre Formulare sollten Sie dieselben Schriftarten, Schriftgrößen und Farben verwenden. Wenn ein grauer Hintergrund auf dem einem Bildschirm bedeutet, daß das betreffende Feld schreibgeschützt ist, sollten alle schreibgeschützten Felder einen grauen Hintergrund aufweisen.
- **Binden Sie die Diagnosefähigkeit in die Anwendung ein:** Wenn möglich, sollten Sie die Diagnosefähigkeit in Ihre Anwendung integrieren. Diese Fähigkeit sollte ermöglichen, die Ausführung der Anwendung anzuzeigen oder zu protokollieren (oder beides). Dieses Protokoll kann den Namen des ausführenden Moduls oder der Funktion, einen Zeitstempel, signifikante Ereignisse, Fehlercodes und andere Punkte enthalten, die Ihnen in der Entwicklungs- oder Einsatzphase des Projekts beim Aufspüren von Problemen helfen können. Um größtmögliche Flexibilität zu erreichen, sollte die Diagnosefunktion zur Laufzeit konfigurierbar sein (z.B. in einer INI-Datei oder der Windows-Registrierung).

## Testen der Anwendung

Auf jeder Entwicklungsstufe Ihrer Anwendung sollten Sie eine Testphase einplanen. Wichtig ist zu definieren, was unter Testen zu verstehen ist. Wenn vielleicht auch im weitesten Sinne des Worts, kann Testen folgendes umfassen:

- Prüfen, ob das logische Datenmodell vollständig, korrekt und konsistent ist.
- Die Zustimmung der Benutzer zur allgemeinen Benutzeroberfläche, Menüstruktur und Flußkontrolle einholen.
- Repräsentative Daten, entweder bekannte oder neue Daten, in die Oracle-Datenbank eingeben, um das Datenmodell, Regeln und Größenvoraussetzungen zu validieren.
- Prüfen, ob die PL/SQL-Programmblöcke wie geplant funktionieren (PL/SQL ist eine Programmerweiterung der Programmiersprache SQL, die Sie in späteren Lektionen noch näher kennenlernen werden).

Zusätzlich zu den oben genannten Punkten müssen Sie spezielle Tests einplanen:

- Typische (und atypische) Testläufe, um festzustellen, ob die Anwendung richtige Eingaben akzeptiert, ungültige Eingaben und Auswahlvarianten ablehnt und richtige Ergebnisse liefert.
- Leistungstests, bei denen unterschiedliche Auslastungsstufen simuliert werden, wie zum Beispiel die Anzahl der Benutzer, die gleichzeitig Datensätze der Studenten ändern.

Wenn Sie eine Anwendung erstellen, die auf unterschiedlichen Client-Systemen (wie z.B. Windows 3.11, Windows 95 und Mac) eingesetzt werden soll, müssen Sie sicherstellen, daß Ihnen diese Computer für Tests, Einsatz und Pflege zur Verfügung stehen. Andernfalls werden Sie viel Zeit dafür aufbringen müssen, festgestellte Fehler zu reproduzieren, zu diagnostizieren und zu beheben. Außerdem sollten Sie sich vergewissern, daß diese Geräte für die Computer der Benutzer im Hinblick auf die Prozessorgeschwindigkeit, den Speicher und den verfügbaren Plattenplatz repräsentativ sind.

## Einsetzen der Anwendung

Während des Einsatzes sind mehrere Punkte zu berücksichtigen:

- **Installieren der Anwendung:** Wenn Sie die Anwendung für eine kleine Benutzergruppe installieren, deren Standorte dicht beieinander liegen, können Sie sich vielleicht den Luxus leisten, die Software selbst zu installieren. Ist die Anzahl der Benutzer jedoch groß oder sind deren Standorte weit verteilt, müssen Sie die Installation der Software weiter automatisieren. Es gibt verschiedene Möglichkeiten. Sie können eine Reihe von Installationsdisketten (oder CD-ROMs) herstellen, welche die Installation automatisieren. Sie können einen Datei-Server verwenden, auf den alle Benutzer bei der Installation zugreifen können. Oder Sie können einen Web Site einrichten, von der die Benutzer die Software herunterladen können und dann einer Reihe von Anweisungen folgen müssen. In allen Fällen umfaßt die Installation zwei Hauptaufgaben. Die erste ist die Erstellung eines Verzeichnisses auf dem Computer des Benutzers, das Kopieren aller notwendigen Dateien in dieses Verzeichnis und das Konfigurieren der Umgebung. Die zweite Aufgabe besteht aus der Installation und Konfiguration der SQL\*Net-Produkts, falls dieses noch nicht installiert ist.
- **Benutzerschulung:** Es muß in jedem Fall Zeit eingeräumt werden, um die Endbenutzer für den Einsatz der Anwendung zu schulen. Diese Schulung kann durch einen Lehrer oder ein Lernprogramm erfolgen. Manche Führungskräfte könnten einwenden, daß die Endbenutzer nicht genügend Zeit für eine regelmäßige Schulung haben. Aber wie man es auch nimmt, die Benutzer brauchen Zeit, um die Anwendung der Software zu erlernen.
- **Online-Hilfe bereitstellen:** Nach erfolgter Schulung werden die Benutzer sicherlich immer noch Fragen zur Verwendung der Software haben. Eine wirksame Möglichkeit, Antworten auf diese Fragen zu erteilen, ist die Bereitstellung einer Hilfedatei im Anwendungsprogramm, um kontextbezogene Hilfe zu leisten.

## Pflege der Anwendung

Sie haben einen langen Weg zurückgelegt. Sie haben Anforderungen gesammelt, die Anwendung entwickelt, implementiert, getestet und eingesetzt - was nun? Warten Sie auf die Rückmeldungen der Benutzer - sowohl (hoffentlich) positive als auch negative (falls es Ihnen hilft, betrachten Sie diese als konstruktive Kritik). Unter den Rückmeldungen der Benutzer werden sich wahrscheinlich auch Anfragen nach zusätzlichen Funktionen befinden (z.B. »Wie lange würden Sie brauchen, um einen Bericht zu erstellen, der so aussieht?«) sowie Fehlerberichte (z.B. »Wenn ich F5 drücke, ist der Bildschirm plötzlich leer.«). Wenn Sie keine Rückmeldungen erhalten, ist das ein schlechtes Zeichen - es könnte bedeuten, daß nur wenige die Software tatsächlich einsetzen.

Die Rückmeldungen sollten nach Schwere und Umfang geordnet werden. Die Behebung kritischer Fehler kann mit einer Notversion erfolgen, in der vielleicht nur die zu ersetzenden Dateien enthalten sind. Die Beseitigung weniger kritischer Fehler oder die Einbindung neuer Funktionen sollte mit einer regulären Version oder einem Upgrade erfolgen.

## Zusammenfassung

In dieser Lektion haben Sie die wichtigsten Phasen der Entwicklung einer Oracle-Anwendung kennengelernt:

- Sammeln von Anforderungen
- Entwerfen der Anwendung



- Implementieren der Anwendung
- Testen der Anwendung
- Einsetzen der Anwendung
- Pflege der Anwendung

## Wie geht es weiter?

Am Tag 3 werden Sie in die begriffliche Theorie von relationalen Datenbanken einsteigen: Entitäten, Attribute, Domänen und Beziehungen. Außerdem erfolgt eine Einführung in die Begriffe der Datenintegrität und der referentiellen Integrität sowie einige der normalen Formen.

## Fragen und Antworten

Frage:

Mit welchen Punkten sollte ich mich beschäftigen, wenn eine neue Anwendung ein vorhandenes System ersetzen soll?

Antwort:

*Es gibt eine Reihe von politischen und technischen Fragen, die Sie berücksichtigen müssen. Jede vorhandene Anwendung hat ihre Befürworter und Kritiker. Sicherlich sollten Sie mit beiden Gruppen viele Gespräche führen, um die Vor- und Nachteile der vorhandenen Anwendung verstehen zu können.*

Versuchen Sie, sich bei der Entwicklung der neuen Anwendung nicht von den Funktionen des vorhandenen Programms einengen zu lassen; Sie wollen die Anwendung bestimmt nicht einfach mit Hilfe eines gängigen Software-Entwicklungswerkzeugs »umschreiben« (im Jargon der Umkonstruierung nennt man das im allgemeinen »auf ausgetretenen Pfaden wandeln«). Versuchen Sie, das Entwicklungs-/Wartungspersonal der alten Anwendung so gut wie möglich in die Neuentwicklung einzubeziehen; diese haben sicherlich viele Ideen für eine Verbesserung des Systems und fühlen sich wahrscheinlich durch die Entwicklung einer neuen Anwendung bedroht. Stehen Sie der alten Anwendung nicht zu kritisch gegenüber; deren Entwickler haben wahrscheinlich mit den zur Verfügung stehenden Ressourcen ihr Bestes getan.

Wenn Sie den Übergang von der alten zur neuen Anwendung planen, sollten Sie schrittweise vorgehen und einen Zeitraum vorsehen, in dem das alte System weiterhin läuft, aber bereits Funktionselemente des neuen Systems eingesetzt werden. Unterschätzen Sie aber nicht den für die Migration der alten Daten in die neue Oracle-Datenbank erforderlichen Aufwand.

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Richtig oder falsch? Wenn Sie ein Werkzeug für die Anwendungsentwicklung benutzen, welches eine objektorientierte Entwicklung unterstützt, ist die Chance größer, eine erfolgreiche Anwendung zu implementieren.



2. Nennen Sie drei Arten von Anforderungen, die bei der Entwicklung eines Systems erforderlich sind.
3. Richtig oder falsch? Sie sollten erst mit der Softwareentwicklung beginnen, wenn Ihnen die Anforderungen vollständig bekannt sind.

## Übungen

1. Welche Arten von Risiken könnten bei der Entwicklung einer Oracle-Anwendung auftreten?
2. Welche Faktoren könnten zu diesen Risikokategorien gehören?
3. Welche Schritte könnten Sie unternehmen, um diese Risiken zu mildern?

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖Kapitel❖ [Top](#) | [Inhalt](#) | [Index](#) **SAMS** Kapitel❖

# 3

# Logischer Datenbankentwurf

[Theorie relationaler Datenbanken](#)

[Datenintegrität](#)

[Primärschlüssel](#)

[Referentielle Integrität](#)

[Beziehungen](#)

[Das Konzept des Nullwerts](#)

[Normalisierung](#)

[Werkzeuge für die Erstellung von Entitätsbeziehungsdigrammen](#)

[Die Beispieldatenbank](#)

[Terminologie zu Oracle: Verbindung, Benutzer und Arbeitssitzung](#)

[Verbindung zu einer Oracle-Datenbank](#)

[Erzeugen eines neuen Benutzers](#)

[Oracle-Datentypen](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

Die Grundlage aller Datenbank Anwendungen ist ein logisches Datenmodell. Wie bei allen Modellen handelt es sich bei einem logischen Datenmodell um die Idealform eines echten Systems. Ein Modell ist nur nützlich, wenn es auch genau ist. Wie ein

reales Unternehmen verhält sich ein logisches Datenmodell eher dynamisch als statisch. Es muß sich in dem Maße entwickeln, wie sich das ihm zugrundeliegende Unternehmen verändert.



Ein logisches Datenmodell stellt sowohl die von einem Unternehmen verwendeten Datenelemente als auch die Beziehung zwischen diesen Datenelementen dar.

Eine der gebräuchlichsten Methoden für die Entwicklung eines logischen Datenmodells ist die Erstellung eines Entitätsbeziehungsmodells. *Entitäten* sind Personen, Orte, Objekte oder Begriffe. Jede Entität wird anhand einer Reihe von Attributen beschrieben. In der Beispieldatenbank sieht der Registrator einen Kursleiter als eine Reihe von Attributen, wie z.B. Instructor ID (Kursleiter-Bezeichnung), last name (Nachname), first name (Vorname) und department (Fachbereich). Der Administrator eines Fachbereichs betrachtet eine Reihe anderer Attribute für denselben Kursleiter, z.B. seine derzeitige Position. Wie Sie sehen, stellen die Anforderungen des Unternehmens den Motor für das Datenmodell dar. Entitäten existieren nicht im leeren Raum - Sie definieren Beziehungen zwischen diesen Entitäten. Diese Beziehungen werden verwendet, um Regeln durchzusetzen. In der Beispielanwendung - einem Informationssystem für ein kleines College - muß eine Klasse von *einem einzigen* Kursleiter unterrichtet werden.

Warum soll man ein logisches Datenmodell entwickeln - warum wendet man sich nicht gleich dem Datenbankentwurf zu? Durch die Entwicklung eines logischen Datenmodells werden Sie gezwungen, sich mit den Daten einer Organisation und deren internen Beziehungen zueinander zu befassen, ohne sich gleich mit den Einzelheiten der Implementierung, wie z.B. dem Datentyp einer Spalte, auseinandersetzen zu müssen. Sie können sich das logische Datenmodell auf einer höheren Abstraktionsebene vorstellen als den Datenbankentwurf.

## Theorie relationaler Datenbanken

Kein Buch, das sich mit relationalen Datenbanken befaßt, kann als vollständig betrachtet werden, wenn es nicht die Grundbegriffe der Theorie relationaler Datenbanken behandelt. Es folgt eine kurze Zusammenfassung dieser Begriffe:

- Jede Entität verfügt über eine Reihe von Attributen, welche die Entität beschreiben. Eine Entität namens course (Kurs) würde die in einem College angebotenen Kurse beschreiben.
- Eine Zeile ist eine Einzelinstanz von Attributwerten. Eine Zeile in der Entität course beschreibt z.B. einen einzelnen vom College angebotenen Kurs.
- Einige Entitätsattribute identifizieren die einzelnen Reihen dieser Entität eindeutig. Diese Attribute bezeichnet man als *Primärschlüssel*. Für die Entität, welche die Studenten beschreibt, ist das Attribut student ID (Studenten-Bezeichnung) der Primärschlüssel, da es die einzelnen Studenten eindeutig identifiziert.
- Keines der Attribute, die den Primärschlüssel bilden, kann einen Nullwert haben.
- Ein Fremdschlüssel ist ein Attribut in einer Entität, dessen Werte als Primärschlüssel in einer anderen Entität vorkommen müssen.
- Entitäten werden zueinander in Bezug gesetzt.
- Die Reihenfolge der Zeilen innerhalb einer Entität ist beliebig.
- Die Reihenfolge der Attribute innerhalb einer Entität ist beliebig.

## Eine Tabelle besteht aus Zeilen und Spalten

Aus der Sicht eines Praktikers wird eine Entität als Datenbanktabelle implementiert. Die Attribute einer Entität werden als Tabellenspalten implementiert. Ein einzelner Satz Attribute oder Spaltenwerte wird als *Zeile* bezeichnet. Die Begriffe *Zeile* und *Datensatz* haben dabei dieselbe Bedeutung.



*Eine Spalte in einer Datenbanktabelle stellt die Attribute einer Entität dar.*

---

---



*Eine Zeile ist eine einzelne Gruppe von Entitätsattributen oder Spaltenwerten in einer Datenbanktabelle.*

---

---



*Ein Datensatz ist dasselbe wie eine Zeile - es handelt sich um eine einzelne Gruppe von Entitätsattributen oder Spaltenwerten.*

---

---



*Sie können nicht anhand eines Verzeichnisses im Dateisystem feststellen, welche Tabellen in einer Oracle-Datenbank gespeichert sind, da das Oracle-RDBMS die interne Struktur seiner Dateien verwaltet. Bei einem Dateiverwaltungssystem wie dBASE wird jede »Tabelle« als separate Datei in einem Verzeichnis gespeichert.*

---

### Die Reihenfolge der Zeilen ist beliebig

Ein Grundsatz der Theorie relationaler Datenbanken ist, daß eine Tabelle keine implizite Reihenfolge hat. Die einzige Möglichkeit, die Reihenfolge zu erfahren, in der die Zeilen einer Tabelle abgerufen werden, ist es, diese Reihenfolge vorzugeben. Das Konzept der nichtimpliziten Reihenfolge ist wirksam, da Sie auf diese Weise die Tabellen abstrakt betrachten und in den meisten Fällen die physische Implementierung von Datenbankstrukturen ignorieren können.

### Die Reihenfolge der Spalten ist beliebig

Wie die Zeilen einer Tabelle haben auch die Spalten keine implizite Reihenfolge. Wenn Sie für die Ausgabe einer Tabellenbeschreibung SQL\*Plus verwenden, gibt SQL\*Plus die Spalten in der Reihenfolge zurück, in der sie erstellt wurden. Sie können allerdings eine beliebige Reihenfolge vorgeben, in der die Spalten abgerufen werden sollen. Darüber hinaus können Sie die Definition einer Spalte ändern, ohne dadurch andere Spalten zu beeinträchtigen. So haben Sie zum Beispiel die Möglichkeit, eine Spalte zu vergrößern, ohne auch nur eine der bestehenden Tabellendefinitionen oder SQL-Anweisungen ändern zu müssen. Die Beachtung der mit der Änderung verbundenen physischen Details ist Aufgabe des Management-Systems für relationale Datenbanken (RDBMS) von Oracle. Eine relationale Datenbank soll für *logische Datenunabhängigkeit* sorgen, da die Definitionen der einzelnen Spalten voneinander unabhängig sind.

## Datenintegrität

Nach der relationalen Theorie verfügt jede Entität über eine Reihe von Attributen, welche die einzelnen Spalten dieser Entität eindeutig identifizieren. Die relationale Theorie sagt auch aus, daß in einer Tabelle keine Zeilen doppelt vorhanden sein können, was ganz einfach bedeutet, daß jede Tabelle einen Primärschlüssel haben muß. Dieses Konzept wird als *Datenintegrität* bezeichnet. Eindeutig ist zum Beispiel die Sozialversicherungsnummer der einzelnen Mitarbeiter.

## Primärschlüssel

Jede Entität verfügt über eine Gruppe von Attributen, welche eine Instanz dieser Entität eindeutig definieren. Diese Attributgruppe wird als Primärschlüssel bezeichnet. Der Primärschlüssel kann sich aus einem einzelnen Attribut - ein Student wird durch Student ID (Studenten-Bezeichnung) eindeutig identifiziert - oder mehreren Attributen - ein Kurs wird sowohl durch Department ID (Fachbereichs-Bezeichnung) als auch Course ID (Kurs-Bezeichnung) eindeutig identifiziert. Manchmal sind die den Primärschlüssel bildenden Attribute offensichtlich, manchmal sind sie es nicht. Um festzustellen, ob Ihre Vorstellungen von einem Primärschlüssel richtig sind, müssen Sie sich vorhandene Daten anschauen. Sie müssen aber auch mit Personen sprechen, welche die Art und Weise kennen, in der die Organisation arbeitet. Verlassen Sie sich bei der Validierung Ihrer Vorstellungen von einem Primärschlüssel nicht nur auf vorhandene Daten.



*Ein Primärschlüssel ist eine Gruppe von Attributen, welche eine Zeile eindeutig identifizieren.*

## Kein Teil des Primärschlüssels hat einen Nullwert

Ein Grundsatz der relationalen Theorie ist, daß kein Teil des Primärschlüssels einen Nullwert haben kann. Wenn Sie eine Weile darüber nachdenken, erscheint diese Tatsache intuitiv klar. Der Primärschlüssel muß die einzelnen Zeilen einer Entität eindeutig bezeichnen; wenn der Primärschlüssel (oder ein Teil davon) einen Nullwert hätte, könnte der Primärschlüssel nichts identifizieren. Ein Kurs, der über keine Course ID verfügt, kann weder identifiziert noch in irgendeiner Weise verarbeitet werden.

## Referentielle Integrität

Tabellen werden mit Hilfe von Fremdschlüsseln zueinander in Relation gesetzt. Ein *Fremdschlüssel* besteht aus einer oder mehreren Spalten, für die eine Gruppe möglicher Werte im Primärschlüssel einer zweiten Tabelle zu finden ist. Referentielle Integrität wird erreicht, wenn die Werte in der Spalte eines Fremdschlüssels auf den referenzierten Primärschlüssel oder auf den Nullwert eingeschränkt werden. Hat der Datenbankentwickler Primär- und Fremdschlüssel erst einmal deklariert, ist die Durchsetzung der Datenintegrität und der referentiellen Integrität Aufgabe des RDBMS.



*Ein Fremdschlüssel besteht aus einer oder mehreren Spalten, deren Werte im Primärschlüssel einer anderen Tabelle wiederzufinden sein müssen.*

## Beziehungen

Die Verbindung zwischen zwei Entitäten wird als Beziehung bezeichnet. In einer Beziehung wird eine Entität als übergeordnet und die andere Entität als untergeordnet identifiziert. Eine Beziehung wird durch folgende Merkmale definiert:

- **Identifizierend oder nichtidentifizierend:** Bei einer nichtidentifizierenden Beziehung bildet der Primärschlüssel der übergeordneten Entität einen Teil des Primärschlüssels der untergeordneten Entität. In einer nichtidentifizierenden Beziehung ist der Primärschlüssel der übergeordneten Entität nicht Teil des Primärschlüssels der untergeordneten Entität.
- **Kardinalität:** Die Kardinalität einer Beziehung wird durch die Anzahl der Zeilen innerhalb der untergeordneten Entität definiert, über welche die übergeordnete Entität verfügen muß. Beispiele für Kardinalitäten sind null, eins oder mehr. Zum Beispiel kann ein Kursleiter, der sich mit Grundlagenforschung beschäftigt, null oder mehrere Klassen unterrichten. Eine Fachabteilung muß jedoch einen oder mehrere Kurse anbieten, andernfalls wäre sie keine Fachabteilung.



*Die Kardinalität bezeichnet die Anzahl der Zeilen einer untergeordneten Entität, über die eine einzelne Zeile einer untergeordneten Entität verfügen muß. Die Kardinalität der Beziehung zwischen einer übergeordneten und einer untergeordneten Entität kann erforderlich sein - zum Beispiel darf eine Zeile einer übergeordneten Entität nur eine einzige Zeile in einer untergeordneten Entität haben. Die Kardinalität kann allerdings auch weniger restriktiv sein - eine Zeile in einer übergeordneten Entität kann zum Beispiel null oder mehrere Zeilen in einer untergeordneten Entität besitzen.*

- **Erforderlich oder optional:** Eine Beziehung ist erforderlich, wenn eine Zeile in einer übergeordneten Entität über eine gewisse Anzahl von Zeilen in einer untergeordneten Entität verfügen muß. Die Beziehung ist optional, wenn eine Zeile in der übergeordneten Entität ohne damit zusammenhängende Zeilen in der untergeordneten Entität auskommen kann.
- **Integritätsregeln:** Diese Regeln definieren, welche Maßnahmen ergriffen werden müssen, wenn eine Zeile in der übergeordneten oder untergeordneten Entität auf irgendeine Weise geändert wird. Es gibt sechs Möglichkeiten: Eine Zeile der übergeordneten Entität wird hinzugefügt, geändert oder entfernt, oder eine Zeile der untergeordneten Entität wird hinzugefügt, geändert oder entfernt. Für jede dieser Möglichkeiten gibt es mehrere anwendbare Maßnahmen - die Aktion einzuschränken, den Schlüsselwert in der über- oder untergeordneten Entität auf einen Nullwert zu setzen, die Operation in bezug auf die untergeordnete Entität zu kaskadieren oder den Schlüsselwert der über- oder untergeordneten Entität auf einen Standardwert zu setzen.

## Das Konzept des Nullwerts

Ein großer Unterschied zwischen einer relationalen Datenbank und der Technologie älterer Datenbanken liegt im Konzept des Nullwerts. In einer nichtrelationalen Datenbank gibt ein bestimmter Wert das Fehlen eines Werts in einem Zeichenfeld oder einem numerischen Feld an.



*Das Wort Null gibt an, daß der Wert einer Spalte oder eines Ausdrucks nicht verfügbar ist oder nicht zugewiesen wurde.*

In einer relationalen Datenbank stellt der Nullwert einer Spalte verschiedene Konzepte dar:

- Ein Wert für diese Spalte ist für die betreffende Zeile nicht verfügbar.

- Der Spalte wurde noch kein Wert zugewiesen.

In einer relationalen Datenbank können Sie den Wert einer Spalte auf null setzen oder eine Spalte daraufhin prüfen, ob ihr Wert null ist.

## Normalisierung

Es ist sehr einfach, eine Tabelle in einer Oracle-Datenbank zu erstellen. Sie können dazu aus einer Vielzahl von Werkzeugen auswählen - SQL\*Plus, SQL Worksheet u.a.; zum Beginn von Tag 4, »Implementierung Ihres logischen Modells: Physischer Datenbankentwurf«, werden Sie mehr über diese Werkzeuge erfahren. Es ist jedoch wichtig, daß Sie sich die Zeit nehmen, den optimalen Entwurf für die Datenbank Ihrer Anwendung zu prüfen.

An dieser Stelle kommt ein Aspekt aus der Theorie relationaler Datenbanken ins Spiel. Bei der Normalisierungstheorie handelt es sich um die Untersuchung von Relationen (Tabellen), Attributen (Spalten) und der Abhängigkeit der Attribute untereinander. Die Ziele der Normalisierung lauten wie folgt:

- Minimieren redundanter Daten
- Vermeiden von Änderungsanomalien
- Reduzieren inkonsistenter Daten
- Entwerfen von Datenstrukturen, die eine einfache Pflege erlauben

Da es sich hierbei um die Theorie handelt, müssen Sie einige wichtige Begriffe verstehen. In Tabelle 3.1 sind diese Begriffe in drei Kategorien eingeteilt - Theoretiker, Analytiker und Entwickler. Bei der Lektüre von Unterlagen zum Thema Datenbanken, sowohl wissenschaftlicher als auch kommerzieller Art, begegnen Ihnen Begriffe, die leicht untereinander ausgetauscht werden können. Wenn Sie diese Lektion lesen, werden Sie feststellen, daß diese Begriffe austauschbar verwendet werden. Sie können die Begriffe verwenden, die Sie bevorzugen, solange Sie diese richtig anwenden und verstehen, was sie bedeuten. Ein Professor der Informatik schreibt zum Beispiel über die Relation XYZ; ein Anwendungsentwickler könnte dieselbe *Sache* als Tabelle XYZ bezeichnen.

*Tabelle 3.1: Vergleich der Datenbankterminologie*

| Theoretiker | Analytiker | Entwickler      |
|-------------|------------|-----------------|
| Relation    | Entität    | Tabelle         |
| Attribut    | Attribut   | Spalte          |
| Tupel       | Zeile      | Zeile/Datensatz |

Die Normalisierungstheorie bezeichnet die gewünschte Anordnung von Tabellen und Spalten als Normalformen. Dieses Kapitel befaßt sich mit der ersten, zweiten und dritten Normalform, welche auch häufig 1NF, 2NF und 3NF genannt werden. Obwohl diese Begriffe theoretisch und abstrakt klingen, sind sie eigentlich recht intuitiv. Weitere Normalformen - Boyce-Codd, 4. und 5. Normalform - beziehen sich auf komplexere Normalisierungsaspekte. Diese Themen gehen aber über den Rahmen dieses Buches hinaus.

### Normalisierungsregel 1: Jede Spalte sollte genau einen Teil der Informationen enthalten

Eine Entität (Tabelle) befindet sich in der ersten Normalform (1NF), wenn alle ihre Attribute atomarer Form sind. Der Begriff *atomar* sagt aus, daß jedes Attribut aus einer einzigen Information über diese Entität besteht. Wenn zum Beispiel eine Entität zum Speichern von Informationen über Mitarbeiter bestimmt ist, würden Sie kein Einzelattribut, Dependents (Untergebene), um die



Namen der Untergebenen eines Mitarbeiters zu speichern. Einige Mitarbeiter könnten keine, andere dagegen viele Untergebene haben.

## Normalisierungsregel 2: Alle Spalten hängen nur vom Primärschlüssel ab

Bei einer Entität in der zweiten Normalform (2NF) dürfen alle Spalten nur vom Primärschlüssel abhängen. In einfachen Worten bedeutet diese Regel, daß eine Tabelle keine *Fremdinformationen* enthalten darf. Zusätzlich zu anderen Informationen bezeichnet die Class-Tabelle zum Beispiel alle Kurse und deren entsprechende Kursleiter. Der Kursleiter wird durch die Spalte Instructor ID (Kursleiter-Bezeichnung) identifiziert. Der Primärschlüssel für die Tabelle ist Class ID (Klassen-Bezeichnung). Wenn die Class-Tabelle auch die Position des Kursleiters enthalten würde, hätte die Tabelle nicht die zweite Normalform; die Position des Kursleiters hängt nur von der Instructor ID und nicht vom Primärschlüssel ab.

## Normalisierungsregel 3: Alle Spalten hängen einzig und allein vom Primärschlüssel ab

Um die dritte Normalform (3NF) einzunehmen, müssen die Spalten einer Tabelle vollständig vom Primärschlüssel abhängen. Das Schlüsselwort in diesem letzten Satz lautet *vollständig*. Jede Spalte der Tabelle muß vom gesamten Primärschlüssel abhängen, nicht nur von einem Teil.

Die Course-Tabelle identifiziert zum Beispiel jeden einzelnen Kurs sowie die Fachabteilung, welche diesen Kursus anbietet. Der Primärschlüssel besteht aus der Department ID und der Course ID; beide Spalten sind notwendig, um eine Zeile in der Tabelle eindeutig zu definieren. Wenn die Course-Tabelle auch eine Spalte zur Ablage des Fachbereichsleiters enthielte, würde sich die Tabelle nicht in der dritten Normalform befinden; der Fachbereichsleiter hängt nur von der Department ID und nicht vom gesamten Primärschlüssel ab. Somit kann man sagen, daß Department Chairperson (Fachbereichsleiter) von einem Teil des Primärschlüssels, jedoch nicht vollständig vom Primärschlüssel abhängt.

Manchmal wird dieses Konzept auch als *abgeleitete Spalte* bezeichnet. Im obigen Beispiel kann Department Chairperson von Department ID abgeleitet werden. Nach der relationalen Theorie sollte eine Tabelle keine abgeleiteten Spalten enthalten. In der Praxis enthalten Tabellen allerdings häufig abgeleitete Spalten.

## Anwenden der Normalisierung auf den Datenbankentwurf

Die Normalisierungstheorie nennt auch Normalformen, die über die dritte Normalform (3NF) hinausgehen, ich möchte dieses Thema an dieser Stelle aber nicht weiter vertiefen. Dagegen würde ich gern über die Anwendung der Normalisierungstheorie sprechen. Wenn Sie Artikel über relationale Datenbanken lesen, treffen Sie auf eine »große Debatte«. Puristen aus dem Bereich der relationalen Theorie meinen, daß alle Tabellen zumindest in der 3NF vorliegen müßten, obwohl Praktiker argumentieren, daß man die Normalform einer Datenbank aufheben - mit anderen Worten, den Datenbankentwurf von der 3NF auf die 2NF reduzieren - muß, um eine annehmbare Leistung zu erzielen. Meine Meinung liegt irgendwo dazwischen, und ich möchte folgendes empfehlen.

### Was Sie tun sollen

Stellen Sie sicher, daß Ihr Entwurf in der 3NF vorliegt.

Stellen Sie keine Vermutungen über die Leistungsfähigkeit der Datenbank an.  
Erzeugen Sie realistische Testdaten, und stufen Sie dann die Leistung Ihrer Datenbank ein.

### Was nicht

Versuchen Sie, Probleme hinsichtlich der Leistung durch Verbesserungen an der Hardware anstatt durch Kompromisse bei Ihrem Datenbankentwurf zu lösen. Denken Sie daran, daß langfristig eine bessere Hardware günstiger ist als ein Datenbankentwurf, dessen Normalform aufgehoben wurde.

Heben Sie die Normalform Ihrer Tabellen nicht auf, bevor Sie genau verstehen, was Sie dagegen eintauschen - bessere Abfrageleistung gegen höhere Datenredundanz, komplizierte Aktualisierungslogik und größere Schwierigkeiten bei der Vergrößerung des Datenbankentwurfs während des Einsatzzeitraums der Anwendung.



*Unter Schema versteht man eine Gruppe von Datenbankobjekten - Tabellen, Spalten, Primärschlüssel, Fremdschlüssel und andere Objekte -, welche ein logisches Datenmodell implementieren.*

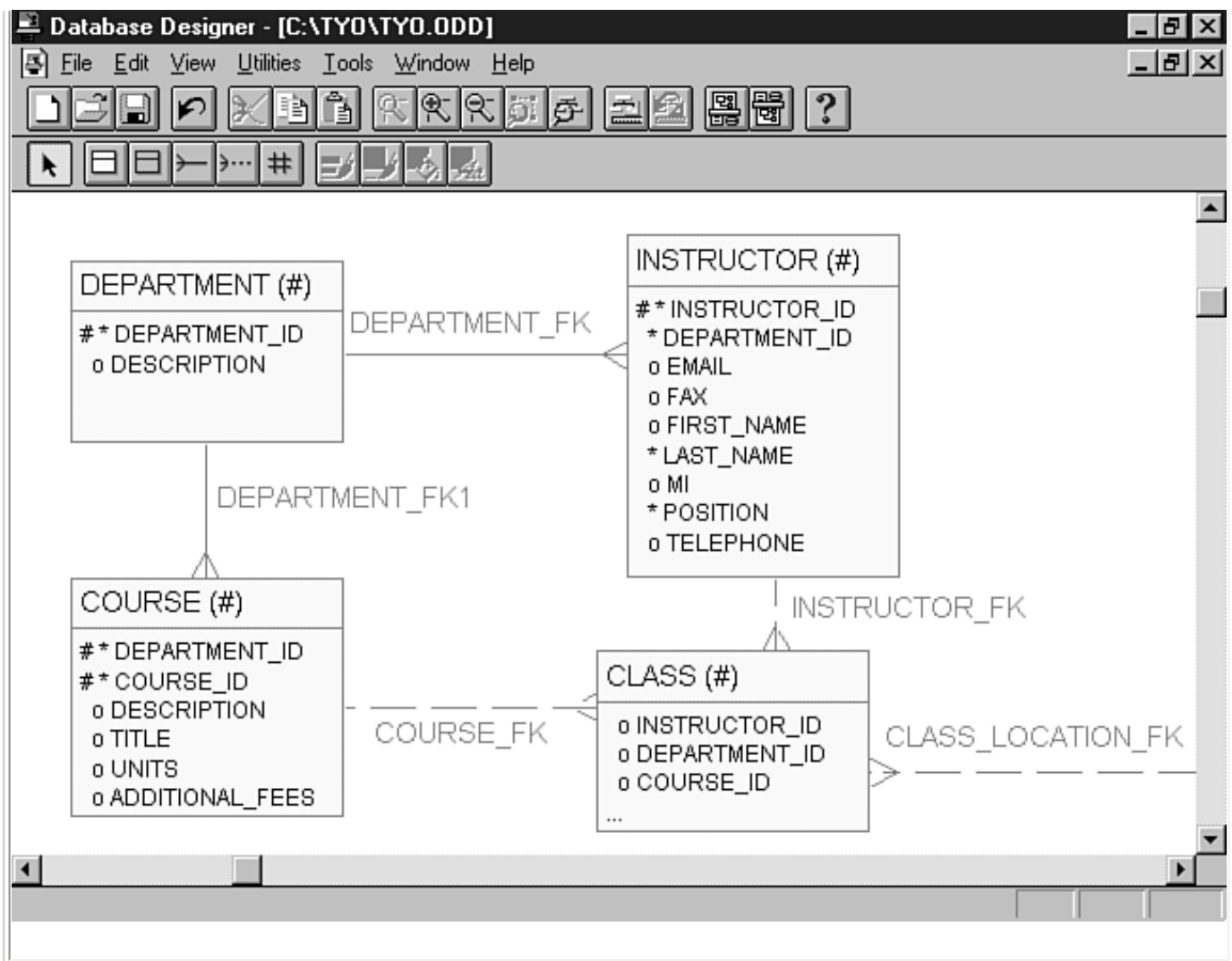
## Werkzeuge für die Erstellung von Entitätsbeziehungsdiagrammen

Um Ihnen bei der Entwicklung eines logischen Datenmodells behilflich zu sein, sollten Sie ernsthaft den Einsatz eines Werkzeugs für die Erstellung von Entitätsbeziehungsdiagrammen, wie z.B. Designer/2000, Oracle Data Designer, LogicWorks ERwin oder Sybase S-Designer, in Erwägung ziehen. Alle diese Werkzeuge sind für Windows NT oder Windows 95 erhältlich. Kurz gesagt ermöglichen Ihnen diese Werkzeuge, ein Entitätsbeziehungsdiagramm zu erstellen, indem Sie Objekte aus einer Werkzeugleiste auswählen und die Beziehungen zwischen diesen Objekten zeichnen.

Die Vorteile dieser Werkzeuge lauten wie folgt:

- Sie vereinfachen den Prozeß der Diagrammerstellung.
- Sie erzeugen automatisch die SQL-Anweisungen, mit denen die Tabellen eigentlich erstellt werden, sowie Integritätsregeln, Indizes und andere Objekte, welche die gewünschte Datenbank bilden.
- Sie erlauben Ihnen, die einzelnen Entitäten, Attribute, Beziehungen und Regeln zu dokumentieren. Diese Werkzeuge erzeugen außerdem Berichte, die auch von anderen Entwicklern, Benutzern und Analytikern verwendet werden können, damit Sie Rückmeldungen über die Genauigkeit des logischen Datenmodells erhalten.

*Abbildung  
3.1:  
Entwicklung  
eines  
Datenmodells  
mit Hilfe des  
Oracle  
Database  
Designers.*



## Die Beispieldatenbank

Bei dem in diesem Buch verwendeten Beispiel handelt es sich um eine Datenbank für ein kleines, fiktives College. Im Jahr 1884 stiftete Reginald Flugle, ein wohlhabender Manschettenknopf- und Krawattennadelfabrikant im Norden der südkalifornischen Wüste, einem kleinen, von der Dürre geplagten College einen beträchtlichen Geldbetrag. Bisher erfolgte die Verwaltung des Flugle-College auf dem Papier. Der Dekan des Colleges hat Ihnen den Auftrag für die Entwicklung eines computergestützten Informationssystems auf der Basis einer Client-Server-Architektur erteilt.

## Anforderungen

Die übergeordneten Anforderungen für das Informationssystem des Flugle-College (später FCIS genannt), umfassen folgende Punkte:

- Informationen über die Studenten speichern, pflegen und abrufen
- Informationen über die Kursleiter speichern, pflegen und abrufen
- Informationen über die einzelnen Fachbereiche speichern, pflegen und abrufen
- Informationen über angebotene Kurse speichern, pflegen und abrufen
- Informationen über die Kurstermine speichern, pflegen und abrufen
- den Studenten ermöglichen, Stundenpläne elektronisch zusammenzustellen
- pro Semester einen Kurskatalog erstellen

Unter Berücksichtigung dieser übergeordneten Anforderungen werden folgende Entitäten benötigt, um das Informationsbedürfnis des Flugle-College zu befriedigen:

- Die Student-Tabelle enthält Informationen über die einzelnen Studenten dieses Colleges.
- Die Department-Tabelle umfaßt Informationen über die einzelnen Fachbereiche des Colleges.
- Die Instructor-Tabelle enthält Informationen über die einzelnen Kursleiter am College.
- Die Course-Tabelle nennt die einzelnen vom College angebotenen Kurse.
- Die Class-Location-Tabelle bezeichnet die einzelnen Klassenräume, die für die Stunden zur Verfügung stehen.
- Die Schedule-Type-Tabelle enthält die Art des Stundenplans - ein Stundenplan kann zum Beispiel zwei Veranstaltungen pro Woche enthalten.
- Die Schedule-Type-Details-Tabelle enthält Einzelheiten über eine bestimmte Art von Stundenplänen, z.B. an welchen Tagen und um wieviel Uhr die Veranstaltungen stattfinden.
- Die Class-Tabelle enthält Informationen über alle geplanten Stunden, wie z.B. Kurs, Kursleiter und Stundenplan. Unter Klasse versteht man das Kursangebot während eines bestimmten Semesters.
- Die Student-Schedule-Tabelle identifiziert die Klassen, für die sich ein Student eingeschrieben hat.

Im nächsten Abschnitt wenden wir uns den Tabellen im einzelnen zu.

## Die Student-Tabelle

Der Zweck der Student-Tabelle liegt in der Pflege von Informationen über die einzelnen Studenten am Flugle-College. Jeder Student wird durch eine Student ID identifiziert - die Tabelle ist so definiert, daß zwei Studenten niemals dieselbe Student ID haben können. Zusätzlich zum Namen des Studenten, der Adresse und anderen wichtigen Informationen enthält die Student-Tabelle auch ein Attribut namens Year (Jahr), das den Jahrgang eines Studenten angibt - zum Beispiel Erstsemester. In dieser Hinsicht könnte diese Beispieldatenbank etwas unrealistisch sein; es ist wahrscheinlich wichtiger, die absolvierten Lerneinheiten als den Jahrgang zu verfolgen. Eine weitere Anmerkung: In dieser Tabelle ist der Status der Studenten nicht angegeben - z.B. aktiv, in Urlaub, inaktiv. Es gibt kein Attribut, welches angibt, ob ein Student einen Abschluß erlangt hat. Trotzdem sind die in der folgenden Student-Tabelle angegebenen Attribute für unsere Zwecke realistisch genug.

| <b>Student</b>                                          |
|---------------------------------------------------------|
| Student ID                                              |
| Last Name                                               |
| First Name                                              |
| Middle Initial (Anfangsbuchstabe des zweiten Vornamens) |
| Street Adress (Straße)                                  |
| City (Ort)                                              |
| State (Bundesland)                                      |
| Zipcode (PLZ)                                           |
| Telephone                                               |
| Fax                                                     |
| Email                                                   |

|                                    |
|------------------------------------|
| Year in School (Semester)          |
| <b>Primärschlüssel:</b> Student ID |

## Die Department-Tabelle

Sorgen Sie beim Erstellen der Department-Tabelle dafür, daß andere Tabellen sich in konsistenter Weise auf einen Fachbereich beziehen. Die Liste für die Department-Tabelle zeigt deren einfache Struktur. Der Primärschlüssel für die Department-Tabelle lautet Department ID.

|                                       |
|---------------------------------------|
| <b>Department</b>                     |
| Department ID                         |
| Description (Beschreibung)            |
| <b>Primärschlüssel:</b> Department ID |

## Die Instructor-Tabelle

Informationen über die einzelnen Kursleiter werden in der Instructor-Tabelle gespeichert, wie in der folgenden Liste dargestellt. Der Primärschlüssel für diese Tabelle lautet Instructor ID. Außerdem gibt es einen Fremdschlüssel - Department ID - der auf die Department-Tabelle verweist.

|                                                       |
|-------------------------------------------------------|
| <b>Instructor</b>                                     |
| Instructor ID                                         |
| Department ID (Fremdschlüssel zur Department-Tabelle) |
| Last Name                                             |
| First Name                                            |
| Middle Initial                                        |
| Position                                              |
| Telephone                                             |
| Fax                                                   |
| Email                                                 |

|                                       |
|---------------------------------------|
| <b>Primärschlüssel:</b> Instructor ID |
|---------------------------------------|

In der Flugle-Datenbank wird jeder Kursleiter einem einzigen Fachbereich zugeordnet, der durch die Department ID gekennzeichnet ist. Für ein echtes College oder Universität würden Sie eine derart künstliche Einschränkung wohl nicht vornehmen.

## Die Course-Tabelle

Jeder Fachbereich bietet eine Reihe von Kursen an. Da es möglich ist, daß zwei Fachbereiche einen Kurs mit derselben Nummer anbieten, enthält der Primärschlüssel der Course-Tabelle sowohl die Course ID als auch die Department ID. Zusätzlich zu den Attributen des Primärschlüssels enthält die folgende Liste die Attribute für einen Kurs - einen Titel, eine Beschreibung, die Anzahl der Lerneinheiten und eventuell erforderliche Zusatzgebühren.

|                                                       |
|-------------------------------------------------------|
| <b>Course</b>                                         |
| Department ID (Fremdschlüssel zur Department-Tabelle) |
| Course ID                                             |
| Title (Titel)                                         |
| Description                                           |
| Units (Lerneinheiten)                                 |
| Additional Fees (zusätzliche Gebühren)                |
| <b>Primärschlüssel:</b> Course ID                     |

Als Alternative könnte der Primärschlüssel der Course-Tabelle nur aus dem Attribut Course ID bestehen. Das würde aber voraussetzen, daß die Course ID für alle Fachbereiche eindeutig wäre. Diese Forderung könnte erfüllt sein, wenn eine Course ID die Department ID mit der Course ID verketteten würde - z.B. BIO101 für Biology 101. Der Nachteil dieses Ansatzes ist allerdings, daß es sich hierbei um eine Verletzung der ersten Normalform handelt - d.h. Course ID enthält nun zwei Teile der Informationen, Fachbereich und Kurs. Obwohl dies wirklich ein Nachteil ist, werden Sie in der Praxis viele Beispiele für Werte von Primärschlüsseln finden, welche auf diese Weise zusammengesetzt sind.

## Die Class-Location-Tabelle

Anhand dieser Tabelle kann ein College-Administrator den geeigneten Ort für eine Klasse anhand der vorhandenen Sitzplätze ermitteln. Der Primärschlüssel für die Class-Location-Tabelle besteht aus dem Gebäude und dem Klassenraum, wie in der folgenden Liste angegeben. Bei einer realen Anwendung würden Sie vielleicht mehr über den Standort der Klasse erfahren wollen - z.B. andere Ausstattungsmerkmale, wie Projektionswände oder Mikrofone.

|                                    |
|------------------------------------|
| <b>Class Location (Klassenort)</b> |
|------------------------------------|

|                          |
|--------------------------|
| Class Building (Gebäude) |
|--------------------------|

|                          |
|--------------------------|
| Class Room (Klassenraum) |
|--------------------------|

|                                          |
|------------------------------------------|
| Seating Capacity (Anzahl der Sitzplätze) |
|------------------------------------------|

|                                                    |
|----------------------------------------------------|
| <b>Primärschlüssel:</b> Class Building, Class Room |
|----------------------------------------------------|

## Die Schedule-Type-Tabelle

Der Zweck dieser Tabelle ist die Beschreibung verschiedener Arten von Stundenplantypen. Ein Beispiel für einen Stundenplantyp sind drei Veranstaltungen pro Woche. Der Primärschlüssel dieser Tabelle lautet Schedule ID (Stundenplan-Bezeichnung). Die folgende Liste enthält eine Beschreibung der Schedule Type-Tabelle.

|                                       |
|---------------------------------------|
| <b>Schedule Type (Stundenplantyp)</b> |
|---------------------------------------|

|             |
|-------------|
| Schedule ID |
|-------------|

|             |
|-------------|
| Description |
|-------------|

|                                     |
|-------------------------------------|
| <b>Primärschlüssel:</b> Schedule ID |
|-------------------------------------|

## Die Schedule-Type-Details-Tabelle

Die Schedule-Type-Details-Tabelle enthält Einzelheiten über einen bestimmten Stundenplantyp, wie z.B. Tag, Anfangszeit und Dauer der einzelnen Veranstaltungen einer Klasse für diesen Stundenplantyp. Wenn zum Beispiel die Schedule ID T10 drei Veranstaltungen pro Woche um 10 Uhr morgens umfaßt, gibt es in der Schedule-Type-Details-Tabelle drei Zeilen. Eine Zeile enthält einen Eintrag für 10 Uhr montags, eine andere einen Eintrag für 10 Uhr mittwochs und die letzte Zeile einen Eintrag für 10 Uhr freitags. Da die Dauer für jede Zeile angegeben wird, kann die Tabelle auch einen Stundenplantyp enthalten, bei der sich die Dauer der Veranstaltung an den einzelnen Tagen unterscheidet. Im folgenden wird die Struktur der Schedule-Type-Details-Tabelle beschrieben.

|                                                                |
|----------------------------------------------------------------|
| <b>Schedule Type Details (Einzelheiten zum Stundenplantyp)</b> |
|----------------------------------------------------------------|

|                                                        |
|--------------------------------------------------------|
| Schedule ID (Fremdschlüssel zur Schedule-Type-Tabelle) |
|--------------------------------------------------------|

|                 |
|-----------------|
| Day (Wochentag) |
|-----------------|

|                             |
|-----------------------------|
| Starting Time (Anfangszeit) |
|-----------------------------|

|                  |
|------------------|
| Duration (Dauer) |
|------------------|

|                                          |
|------------------------------------------|
| <b>Primärschlüssel:</b> Schedule ID, Day |
|------------------------------------------|

## Die Class-Tabelle



Die Class-Tabelle enthält die Besonderheiten der einzelnen vom College angebotenen Kurse für ein bestimmtes Semester. Sie können sich eine Klasse als *Beispiel* für einen Kurs vorstellen - daher muß die Class-Tabelle die Course ID und die Department ID enthalten. In der folgenden Liste ist die Struktur der Class-Tabelle dargestellt. Für jede Klasse muß ein Ort und ein Stundenplan angegeben sein. Und natürlich muß jede Klasse von einem Kursleiter unterrichtet werden.

|                                                            |
|------------------------------------------------------------|
| <b>Class</b>                                               |
| Class ID                                                   |
| Department ID (Fremdschlüssel zur Course-Tabelle)          |
| Course ID (Fremdschlüssel zur Course-Tabelle)              |
| Instructor ID (Fremdschlüssel zur Instructor-Tabelle)      |
| Schedule ID (Fremdschlüssel zur Schedule-Type-Tabelle)     |
| Class Building (Fremdschlüssel zur Class-Location-Tabelle) |
| Class Room (Fremdschlüssel zur Class-Location-Tabelle)     |
| Semester                                                   |
| School Year (Schuljahr)                                    |
| <b>Primärschlüssel:</b> Class ID                           |

## Die Student-Schedule-Tabelle

Die Student-Schedule-Tabelle dient zwei Zwecken. Erstens verzeichnet sie die Zensur, die ein Student in den einzelnen Klassen erreicht hat. Zweitens identifiziert Sie für die noch nicht abgeschlossenen Klassen, welcher Kurs Teil des aktuellen Stundenplans eines Studenten ist. In realistischeren Datenbanken würden Sie wahrscheinlich die Klassen, die ein Student bereits abgeschlossen hat, von seinem aktuellen Stundenplan unterscheiden. Der Primärschlüssel der Tabelle setzt sich aus Student ID und Class ID zusammen. Im folgenden wird die Struktur der Student-Schedule-Tabelle beschrieben.

|                                                     |
|-----------------------------------------------------|
| <b>Student Schedule (Stundenplan des Studenten)</b> |
| Student ID (Fremdschlüssel zur Student-Tabelle)     |
| Class ID (Fremdschlüssel zur Class-Tabelle)         |
| Grade (Zensur)                                      |
| Date Grade Assigned (Datum der Zensurvergabe)       |

|                                              |
|----------------------------------------------|
| <b>Primärschlüssel:</b> Student ID, Class ID |
|----------------------------------------------|

## Terminologie zu Oracle: Verbindung, Benutzer und Arbeitssitzung

Bevor Sie in der Lektion vom vierten Tag lernen, wie man eine Tabelle erstellt, müssen Sie sich mit der in diesem Buch, in verschiedenen Oracle-Werkzeugen und in der Dokumentation zu Oracle verwendeten Terminologie vertraut machen. Zwei Begriffe - *Datenbankverbindung* und *Datenbankbenutzer* - verlangen erhöhte Aufmerksamkeit.



*Eine Datenbankverbindung ist ein Oracle-Benutzer, der einen Benutzernamen und ein Kennwort an eine Oracle-Datenbank gesendet hat und in der Lage ist, der Oracle-Datenbank SQL-Anweisungen zu erteilen.*



*Unter einem Datenbankbenutzer versteht man einen einzelnen oder automatischen Prozeß, der über einen eindeutigen Namen und ein Kennwort verfügt, die von einer bestimmten Oracle-Datenbank erkannt werden.*

### Datenbankverbindung

Eine *Verbindung* mit Oracle herzustellen bedeutet, einen gültigen Benutzernamen und ein Kennwort anzugeben, welche von der Oracle-Datenbank akzeptiert werden. Mit Hilfe eines beliebigen Oracle-Werkzeugs kann ein Benutzer eine Verbindung zu einer Oracle-Datenbank herstellen. Auch mit einem Drittwerkzeug wie PowerBuilder kann sich ein Benutzer bei einer Oracle-Datenbank anmelden. Die Begriffe *Oracle-Sitzung*, *Oracle-Verbindung* und *Datenbankverbindung* werden oft gleichbedeutend verwendet. Sie meinen alle dasselbe - die Aufgaben, die ein Datenbankbenutzer vom Zeitpunkt der erfolgreichen Anmeldung bei einer Oracle-Datenbank bis zum Zeitpunkt der Abmeldung übernimmt.

Ein wichtiges Merkmal einer Datenbankverbindung ist die Tatsache, ob es sich um eine lokale oder um eine Ferndatenbank handelt. Der Ausdruck *lokale Verbindung* gibt an, daß sich das vom Benutzer für die Verbindung mit einer Oracle-Datenbank verwendete Programm auf demselben Computer befindet wie die Oracle-Datenbank selbst. Der Begriff *Fernverbindung* bedeutet, daß sich das vom Benutzer für die Verbindung mit einer Oracle-Datenbank verwendete Programm auf einem Rechner und die Oracle-Datenbank auf einem anderen Computer befindet.

### Datenbankbenutzer

Jede Verbindung zu einer Datenbank wird für einen Datenbankbenutzer hergestellt. Die Begriffe *Tabelleneigentümer*, *Oracle-Benutzer* und *Oracle-Account* werden gleichbedeutend verwendet. Ein *Tabelleneigentümer* ist der Datenbankbenutzer, dem eine Tabelle gehört. Ein Tabelleneigentümer ist immer ein Oracle-Benutzer, aber ein Oracle-Benutzer muß nicht immer eigene Tabellen besitzen. Ein Beispiel soll bei der Erläuterung dieses Konzepts helfen.

Betrachten wir die Beispieldatenbank. Ein Oracle-Account namens *flugle* besitzt alle Tabellen des College-Informationssystems. Sally Jensen ist die Verwalterin des Fachbereichs Biologie und verfügt über einen Oracle-Account namens *jensens*. Obwohl Sally Jensen Zugang zu allen Tabellen des *flugle*-Account hat, besitzt sie keine eigenen Tabellen.

Jede in einer Oracle-Datenbank befindliche Tabelle muß einem Eigentümer gehören. Ein sinnvoller Ansatz wäre es, einen Eigentümer zu erzeugen, bei dem es sich nicht um eine Einzelperson innerhalb der Organisation, sondern um die gesamte Organisation handelt. Wenn Sie darüber nachdenken, werden Sie feststellen, daß die Daten einer Organisation - Informationen, die von den Mitgliedern der Organisation erstellt, geändert und verwendet werden - nicht nur einer Person gehören. Außerdem ist die Dynamik einer Organisation zu berücksichtigen: Die Mitarbeiter kommen und gehen, aber die Organisation bleibt erhalten.

Anstatt nun eine Person zum Eigentümer der Oracle-Tabellen zu ernennen, werden Sie in der Beispielanwendung einen neuen Benutzer namens `flugle` anlegen und diesen Oracle-Account als Eigentümer aller Tabellen innerhalb der Anwendung angeben.

## Verbindung zu einer Oracle-Datenbank

Bevor Sie einen neuen Oracle-Benutzer erzeugen, müssen Sie wissen, wie Sie eine Verbindung zu einer Oracle-Datenbank herstellen. Die meisten Oracle-Werkzeuge stellen ein Dialogfenster bereit, um die Verbindung zu einer Oracle-Datenbank herzustellen. Im Dialogfenster werden Sie aufgefordert, drei Eingaben vorzunehmen:

- den Oracle-Datenbankbenutzer, den Sie für Ihre Verbindung verwenden werden
- ein Benutzerkennwort
- einen Datenbanknamen oder eine DB-Adresse

Falls Sie Personal Oracle verwenden wollen, brauchen Sie keinen Datenbanknamen einzugeben. Wenn Sie versuchen, eine Verbindung zu einer Oracle-Datenbank auf einem Server herzustellen, müssen Sie im allgemeinen den Namen der Datenbank angeben.

## Erzeugen eines neuen Benutzers

Als ersten Schritt bei der Erstellung der Beispieldatenbank werden Sie einen Oracle-Benutzer namens `flugle` erzeugen, dem alle Datenbankobjekte in der Anwendungsdatenbank gehören werden. In diesem Abschnitt werden zwei unterschiedliche Vorgehensweisen beschrieben. Wenn Sie Personal Oracle für Windows 95 verwenden, werden Sie lernen, wie Sie den Navigator einsetzen können, um den neuen Benutzer zu erzeugen. Bei Verwendung eines Oracle-Servers - entweder Workgroup Server oder Universal Server - werden Sie sehen, wie dieselbe Aufgabe mit Hilfe des Security Managers, einer Komponente des Oracle Enterprise Managers, ausgeführt werden kann.

### Erzeugung eines neuen Benutzers mit dem Personal Oracle Navigator

Rufen Sie zunächst den Navigator in der Programmgruppe Personal Oracle unter Windows 95 auf. Um die vorhandenen Benutzer der lokalen Datenbank zu prüfen, doppelklicken Sie auf den Ordner User. Eine Liste der vorhandenen Benutzer wird auf der rechten Seite des Hauptfensters angezeigt. Um den Benutzer `flugle` zu erzeugen, wählen Sie den Ordner User und klicken mit der rechten Maustaste auf New. Der Navigator öffnet ein Fenster, das Sie auffordert, den Benutzernamen, das Kennwort und die Kennwortbestätigung einzugeben. (Siehe Abbildung 3.2.)

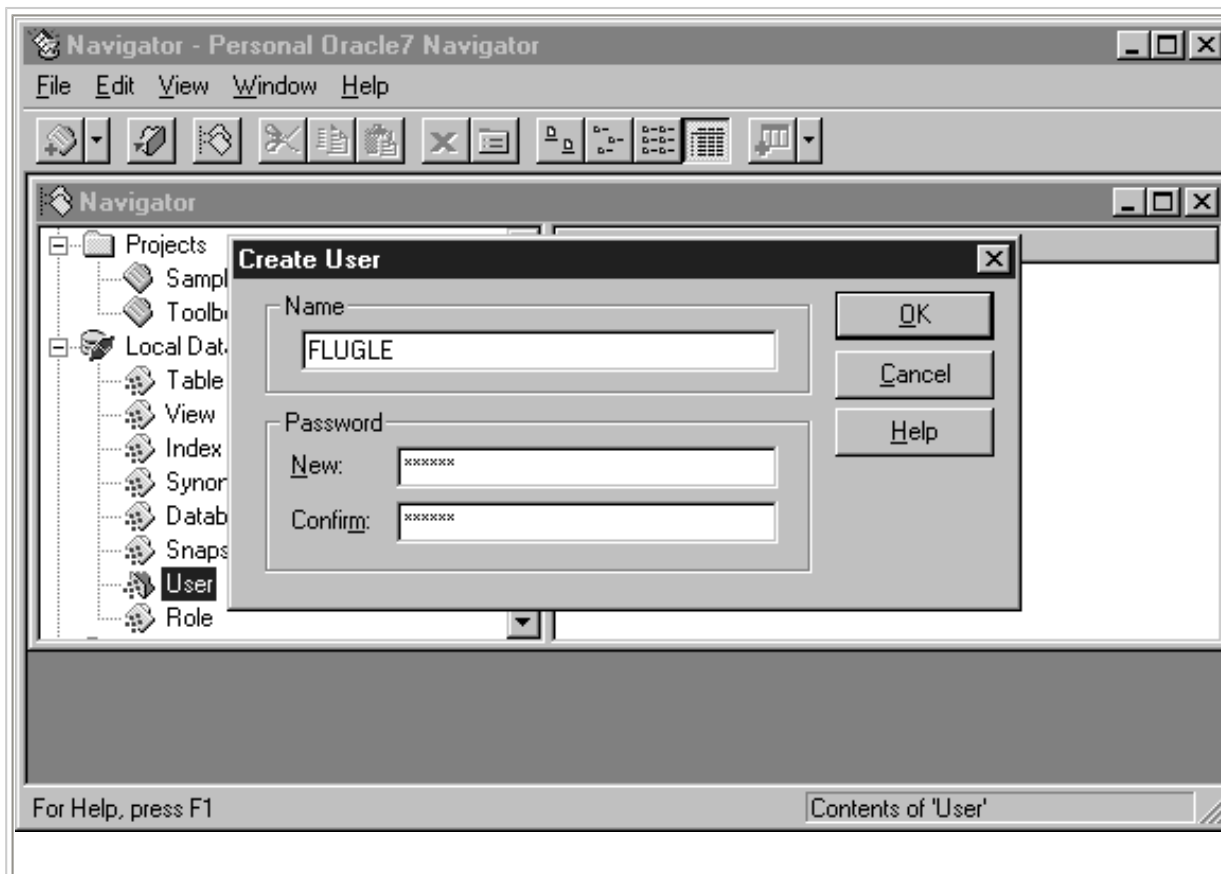


Abbildung 3.2:  
Erzeugung des  
Benutzers flugle mit  
dem Navigator.

Klicken Sie auf OK. Der Benutzer flugle erscheint in der Liste der vorhandenen Benutzer im Ordner Local Database (siehe Abbildung 3.3).

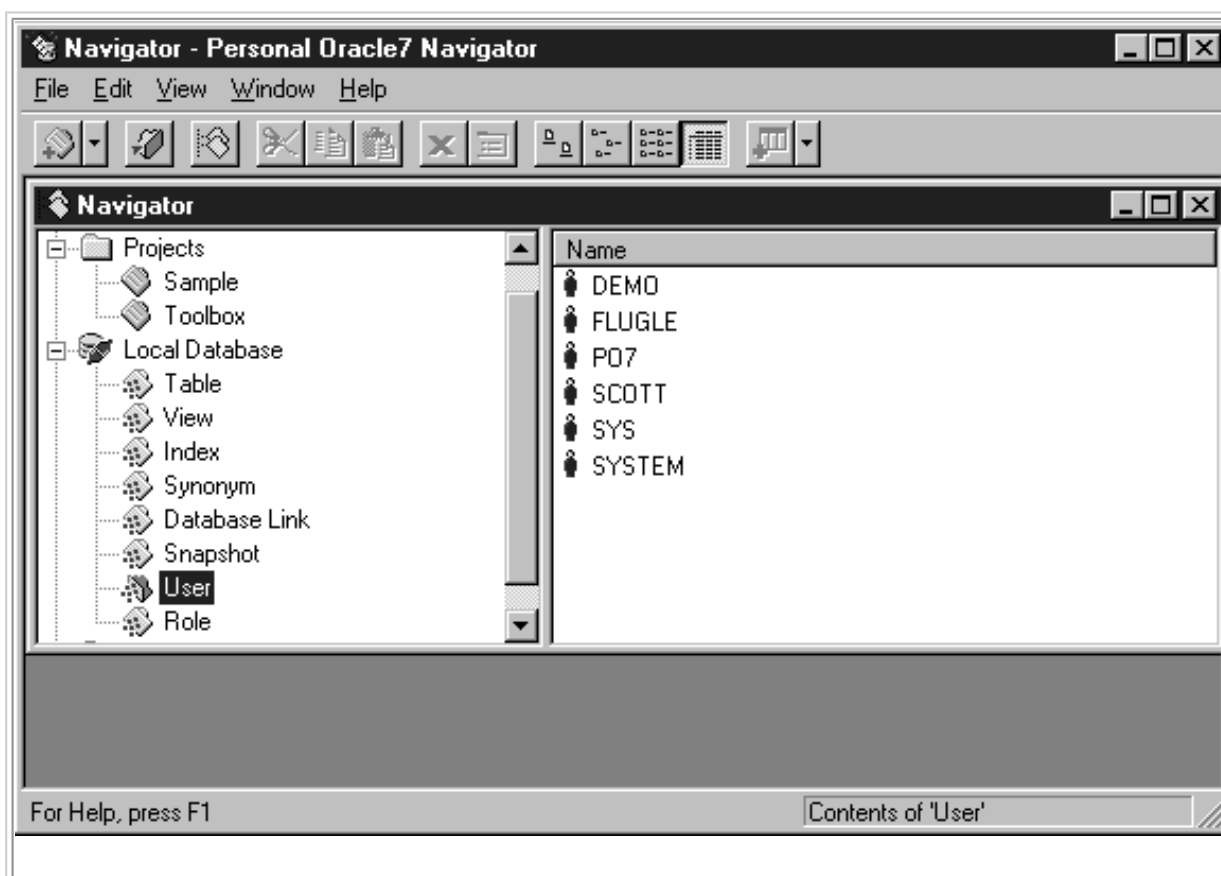


Abbildung 3.3:  
Der Benutzer flugle  
wird in der Liste der  
Benutzer in der  
lokalen Datenbank  
angezeigt.

Vielleicht möchten Sie dem Benutzer flugle eine vorhandene Datenbankrolle zuordnen - resource. Diese Rolle erlaubt es dem Oracle-Account flugle, neue Tabellen und andere Datenbankobjekte zu erstellen. Um diese Rolle zuzuweisen, wählen Sie den Benutzer flugle, und klicken Sie mit der rechten Maustaste auf Properties. Ein Fenster mit zwei Registerkarten erscheint; wählen Sie die Registerkarte Role/Privilege. Aktivieren Sie das Optionsfeld Roles. Wählen Sie die Rolle namens resource und klicken

Sie auf <. Die Rolle wird dem Benutzer flugle zugeordnet (siehe Abbildung 3.4).

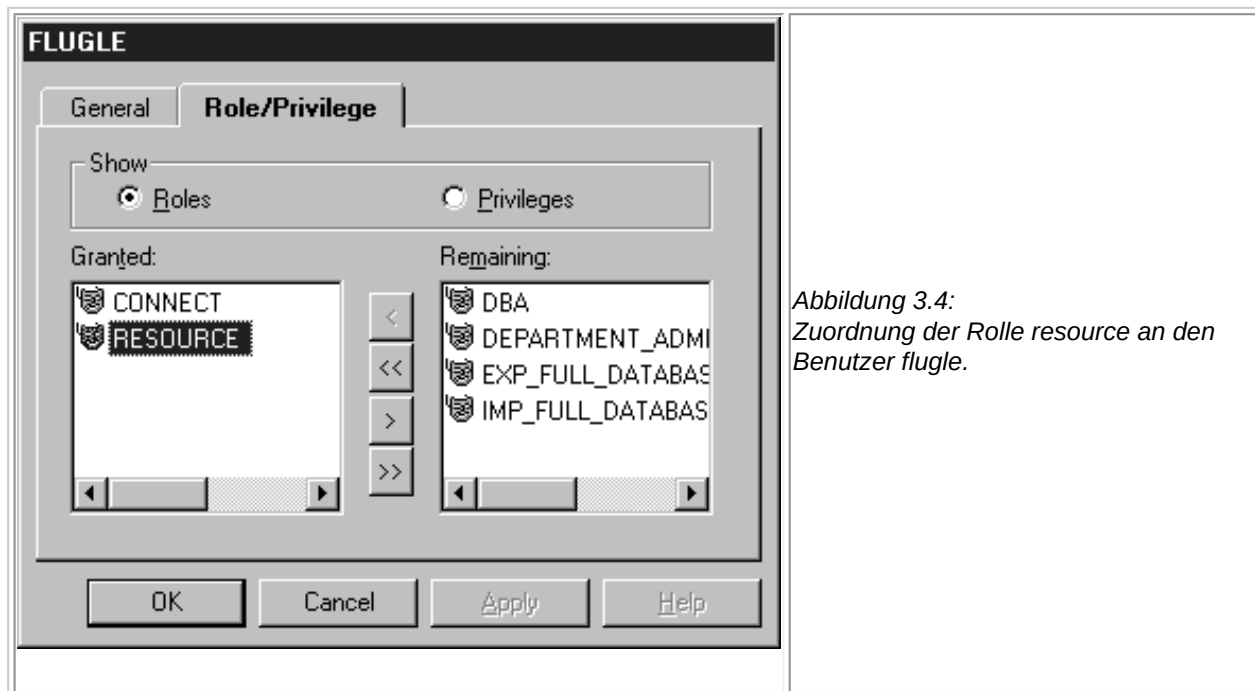


Abbildung 3.4:  
Zuordnung der Rolle resource an den Benutzer flugle.

## Erzeugung eines neuen Benutzers mit dem Security Manager

Als Alternative zum Personal Oracle Navigator zeigt Ihnen dieser Abschnitt, wie Sie einen Benutzer mit dem Security Manager, einer Komponente des Oracle Enterprise Managers, erzeugen können. Wenn Sie den Security Manager zum ersten Mal benutzen, müssen Sie unter Verwendung eines Accounts mit DBA-Berechtigungen eine Verbindung zur Oracle-Datenbank herstellen. Verwenden Sie zu diesem Zweck den system-Account, um sich erstmalig bei der Oracle-Datenbank anzumelden.

1. Um den Security Manager aufzurufen, wählen Sie diesen aus der Programmgruppe Oracle Enterprise Manager. Ein Dialogfenster erscheint, in dem Sie aufgefordert werden, den Benutzernamen, das Kennwort und den Dienst einzugeben. Geben Sie system, manager und fcis in die betreffenden Felder ein, wie in Abbildung 3.5 dargestellt.

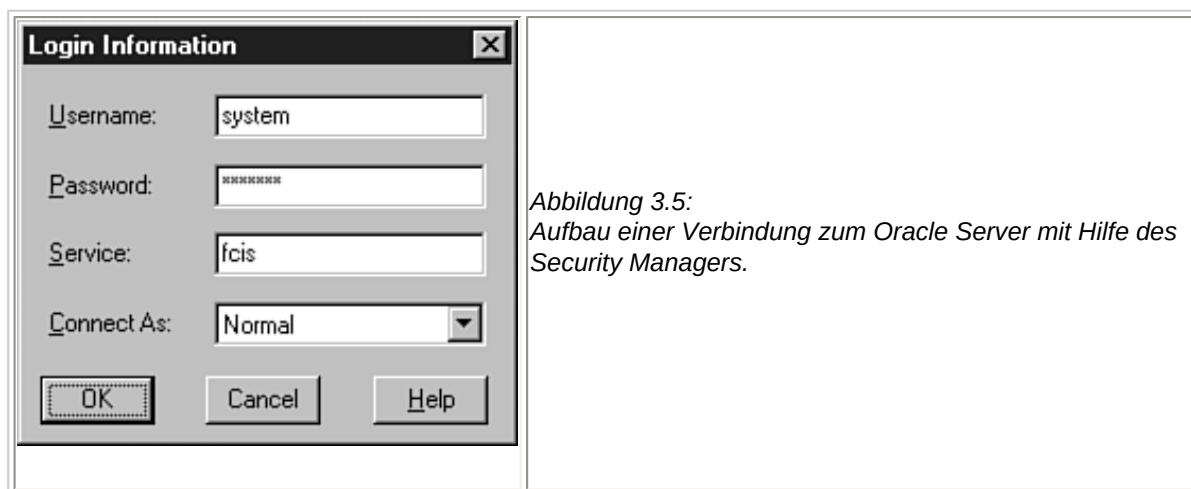
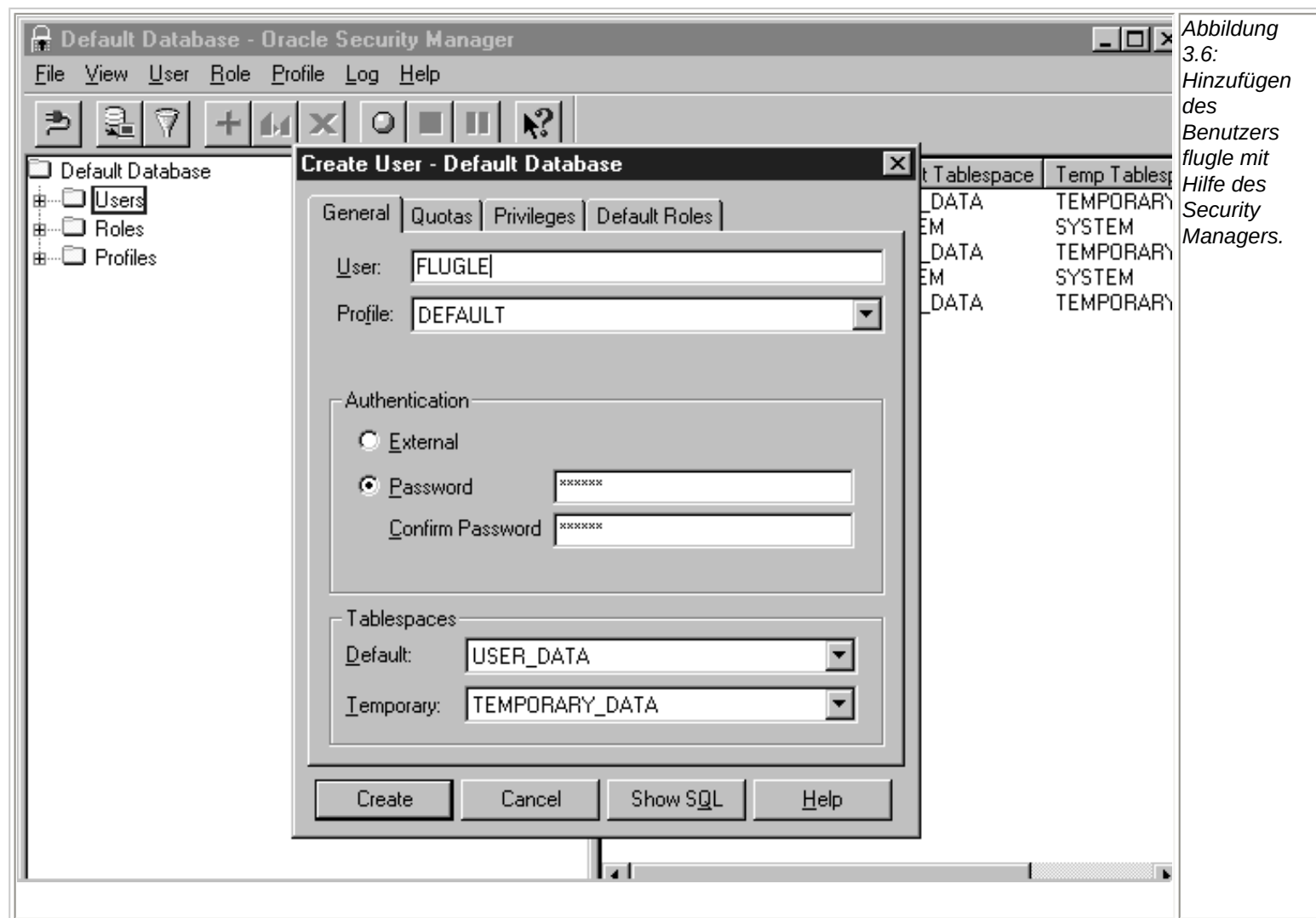


Abbildung 3.5:  
Aufbau einer Verbindung zum Oracle Server mit Hilfe des Security Managers.



In diesem Beispiel wird fcis als Dienst oder Datenbank verwendet. Sie können diesen Datenbank-Alias mit dem Dienstprogramm SQL\*Net Easy Configuration erzeugen, wie an Tag 1, »Die Welt relationaler Datenbanken erforschen«, beschrieben wird. Wenn Sie einen Oracle-Server verwenden, fragen Sie bitten Ihren DBA nach dem Knoten- und Instanznamen, den Sie beim Erzeugen eines neuen Datenbankalias mit Hilfe von SQL\*Net Easy Configuration benutzen sollen.

2. Klicken Sie zum Erzeugen des neuen Benutzers auf den Ordner Users, anschließend klicken Sie entweder auf die Schaltfläche + oder wählen User | Create.
3. Geben Sie, wie in Abbildung 3.6 dargestellt, in die Felder User, Password und Confirm Password jeweils den Namen flugle ein.
4. Falls Sie Personal Oracle verwenden, geben Sie unter Default Tablespace user\_data und unter Temporary Tablespace temporary\_data ein.



Bevor Sie den Benutzer flugle tatsächlich erzeugen können, ordnen Sie die Rolle resource dem Benutzer flugle zu, damit der Benutzer Tabellen und andere Datenbankobjekte erstellen kann.

1. Klicken Sie auf den Ordner Privileges.
2. Wählen Sie resource aus der Liste der Rollen, die im Dialogfenster unten angezeigt wird; klicken Sie dann auf die Schaltfläche Add (siehe Abbildung 3.7). Der Benutzer flugle wird erzeugt - er sollte nun auf der rechten Seite des Fensters im Security Manager angezeigt werden.
3. Wählen Sie den Menüeintrag File | Exit, um den Security Manager zu schließen.

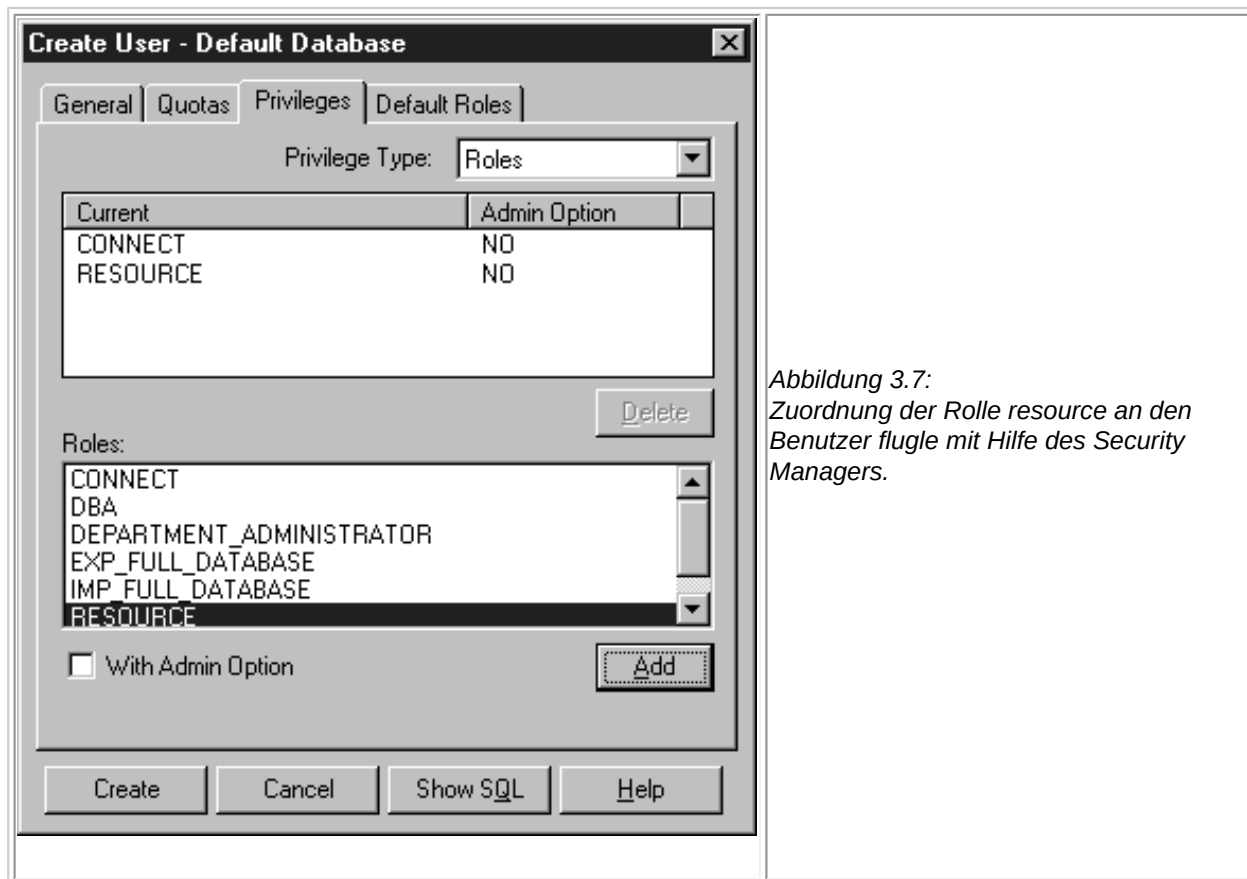


Abbildung 3.7:  
Zuordnung der Rolle resource an den Benutzer flugle mit Hilfe des Security Managers.

## Oracle-Datentypen

Als Überleitung zu der Lektion des vierten Tags sollten wir nun einige Grundlagen in bezug auf die für die Verwendung in einer Oracle-Datenbank verfügbaren Datentypen betrachten. Jede Spalte in einer Oracle-Datenbank muß durch einen dieser Datentypen definiert werden. Bei Oracle8 können Sie zusätzlich zu den vordefinierten Datentypen auch Ihre eigenen Datentypen verwenden. Die vordefinierten Datentypen für vier Datenkategorien werden in den folgenden Abschnitten erläutert:

- Zahlen
- Zeichenfolgen
- Datum und Uhrzeit
- Lange Zeichenfolgen und BLOBs (große Binärobjekte)

### Zahlen

Oracle bietet verschiedene Datentypen zum Speichern von Zahlen an; jeder Typ dient einem anderen Zweck.

**number** Speichert allgemeine Zahlen.

**decimal** Speichert Festkommazahlen und ermöglicht die Kompatibilität von Oracle zu anderen relationalen Datenbanken - besonders SQL/DS und DB2.

**float** Speichert Gleitkommazahlen und gewährleistet die Kompatibilität von Oracle zum ANSI-Datentyp float.

Der Datentyp **number** bietet die größte Flexibilität beim Speichern numerischer Werte. Er akzeptiert positive und negative ganze und reelle Zahlen und verfügt über eine Genauigkeit von 1 bis zu 38 Stellen.





Die Syntax für die Angabe des Datentyps `number` bei der Definition einer Spalte lautet

`number (Gesamtstellenzahl, Anzahl Nachkommastellen)`

Unter Gesamtstellenzahl versteht man die maximale Anzahl der zu speichernden Stellen.

Anzahl Nachkommastellen wird verwendet, um die Position der Nachkommastellen rechts (positiv) oder links (negativ) vom Dezimalpunkt anzugeben. Diese Anzahl reicht von -84 bis 127.

Oracle kann zwischen 1 und 38 Stellen für eine Zahl reservieren. Die Anzahl der zum Speichern einer Zahl in einer Oracle-Datenbank erforderlichen Bytes hängt von der Anzahl der Stellen ab, die verwendet werden, um diese Zahl auszudrücken.

Wenn Sie die Gesamtstellenzahl begrenzen, begrenzt Oracle die in der Spalte zu speichernden Werte auf die vorgegebene Gesamtstellenzahl. Nehmen wir zum Beispiel an, daß Sie eine Tabelle namens `Number_Digits_Demo` speichern, die zwei Spalten enthält:

```
Record_No Definiert als int
Value Definiert als number(4)
```

Wenn Sie in der Spalte `Value` einen Wert speichern, der die vorgegebene Gesamtstellenzahl überschreitet, gibt Oracle eine Fehlermeldung aus, wie in Listing 3.1 dargestellt.

***Listing 3.1: Oracle erzwingt numerische Genauigkeit.***

```
SQL> insert into number_digits_demo
2 (record_no, value)
3 values
4 (101, 9999);
1 row created.
```

```
SQL> insert into number_digits_demo
2 (record_no, value)
3 values
4 (101, 10000);
(101, 10000)
*
```

```
ERROR at line 4:
ORA-01438: value larger than specified precision allows for this column.
```

Betrachten wir ein anderes Beispiel für die Gesamtstellenzahl und die Anzahl der Nachkommastellen. Nehmen Sie an, Sie haben eine Tabelle namens `Scale_Precision_Demo` mit zwei Spalten:

```
Record_No Definiert als int
Value Definiert als number(6,2)
```

In der Spalte `number` lautet die Gesamtstellenzahl 6 und die Anzahl der Nachkommastellen 2. Mit anderen Worten, von insgesamt sechs Stellen reserviert Oracle zwei Stellen rechts vom Dezimalpunkt und läßt maximal vier Stellen links vom Dezimalpunkt übrig. Außerdem können in der Spalte für alle Zahlen nicht mehr als sechs Stellen gespeichert werden.

Wie in Listing 3.2 dargestellt, kann die Zahl 1234.5 in der Spalte `number` gespeichert werden. Die Zahl 12345.1 kann jedoch nicht gespeichert werden, da sie links vom Dezimalpunkt über fünf Stellen verfügt. Ähnlich verhält es sich mit der Zahl 12345, obwohl sie insgesamt nur fünf Stellen hat, kann sie in der Spalte nicht gespeichert werden, da sich diese fünf Stellen links vom Dezimalpunkt befinden.

**Listing 3.2: Oracle erzwingt die Gesamtstellenzahl und die Anzahl der Nachkommastellen.**

```
SQL> insert into scale_precision_demo
2 (record_no, value)
3 values
4 (901, 1234.5);
1 row created.
```

```
SQL> insert into scale_precision_demo
2 (record_no, value)
3 values
4 (901, 12345.1);
(901, 12345.1)
*
```

ERROR at line 4:  
ORA-01438: value larger than specified precision allows for this column

```
SQL> insert into scale_precision_demo
2 (record_no, value)
3 values
4 (901, 12345);
(901, 12345)
*
```

ERROR at line 4:  
ORA-01438: value larger than specified precision allows for this column

**Zeichenfolgen**

Zum Speichern von Zeichenfolgen können Sie aus mehreren Datentypen auswählen:

```
char
varchar
varchar2
long
```

In vielen Datenbanken handelt es sich bei dem Großteil der Daten um Zeichen; die Vor- und Nachteile eines jeden Datentyps werden in den folgenden Absätzen erläutert.

Der Datentyp char speichert Zeichenketten fester Länge von bis zu 255 Zeichen. Wenn Sie keine Länge vorgeben, wird in einer char-Spalte ein Zeichen gespeichert.

Den Datentyp varchar sollten Sie nur verwenden, wenn Sie eine ältere Oracle-Datenbank unterstützen. (Oracle gibt eine Warnung aus, daß der Datentyp varchar von künftigen Versionen vielleicht nicht mehr unterstützt wird.) Der Datentyp varchar2 wurde von Oracle eingeführt und soll varchar ersetzen. Wenn Sie eine neue Oracle-Anwendung erstellen, gibt es keinen Grund, den Datentyp varchar zu benutzen.

varchar2 speichert bis zu 2000 Zeichen in einer einzigen Spalte. Wenn Sie unbedingt eine größere Zeichenmenge speichern müssen, sollten Sie die Verwendung des Datentyps long in Erwägung ziehen.

Sie können 2 Gbyte Zeichen in einer long-Spalte speichern. Allerdings können Sie in long-Spalten keine der in Oracle integrierten Funktionen oder Operatoren verwenden. Am besten stellen Sie sich eine long-Spalte wie eine große schwarze Kiste vor. Sie können den Inhalt dieser Kiste erst sehen, wenn die Kiste geleert wurde.

Folgen Sie diesen Richtlinien, wenn Sie den geeigneten Datentyp für eine Spalte auswählen, in der Daten in Zeichenform

gespeichert werden sollen.

- Sie können char verwenden, um Spalten zu definieren, in denen ein einziges Zeichen gespeichert wird.
- Verwenden Sie varchar2 zum Speichern variabler Zeichenfolgen, die bis zu 2000 Zeichen enthalten.
- Verwenden Sie long, um mehr als 2000 Zeichen in einer Spalte zu speichern.

Wenn Sie versuchen, mehr als 2000 Zeichen zu speichern, sollten Sie dafür nicht mehrere varchar2-Spalten erstellen. Die Umwandlung, das Suchen oder Bearbeiten des Inhalts mehrerer Spalten ist unpraktisch.

## Den Datentyp char verwenden

Weil der Datentyp char Spalten fester Länge speichert, sollten Sie diesen Typ verwenden, wenn Sie Spalten definieren, die ein einziges Zeichen enthalten. Die Verwendung des Datentyps char zum Speichern längerer Zeichenketten ist nicht sinnvoll, da Sie damit Speicherplatz verschwenden.

## Den Datentyp varchar verwenden

Da hiermit Zeichenfolgen variabler Länge gespeichert werden, sollte der Datentyp varchar zum Speichern von Zeichenfolgen bevorzugt eingesetzt werden. Dieser Datentyp kann bis zu 2000 Zeichen speichern. Das Oracle-RDBMS ist aber sehr leistungsfähig; es weist nur den für die Speicherung der einzelnen Spaltenwerte benötigten Speicherplatz zu. Am Tag 7, »Die in SQL integrierten Funktionen ausnutzen«, werden Sie die Vielzahl der Möglichkeiten analysieren, wie Sie die in Oracle integrierten Zeichenfolgefunktionen und Operatoren anwenden können.

## Datums- und Zeitinformationen

Eine der Stärken von Oracle ist der Datentyp date. Der Datentyp date sollte eigentlich datetime heißen, da er sowohl für die Speicherung des Datums als auch der Zeit verwendet wird. Oracle weist einer date-Spalte einen festen Platz von 7 Byte zu, selbst wenn Sie eine date-Spalte lediglich zum Speichern von Datumsinformationen oder Zeitinformationen verwenden.

Oracle verfügt über einige integrierte Funktionen, die speziell zum Bearbeiten von date-Werten und -Ausdrücken dienen. Am siebten Tag werden wir Ihnen einige Beispiele für diese Funktionen zeigen. Der Datentyp date ermöglicht es Ihnen, Daten im Zeitraum vom 1. Januar 4712 v. Chr. bis zum 31. Dezember 4712 n. Chr. zu speichern. Oracle verwendet zur Datumseingabe und -anzeige das Format TT-MM-JJ.

## Lange Zeichenfolgen

Wie bereits zuvor erwähnt, müssen Sie den Oracle-Datentyp long verwenden, um mehr als 2000 Zeichen in einer einzelnen Spalte zu speichern. Eine long-Spalte faßt eine Zeichenmenge von bis zu 2 Gbyte. Wie beim Datentyp varchar2 ist das Oracle-RDBMS auch bei der Verwendung des Datentyps long sehr leistungsfähig; es reserviert lediglich den für die Speicherung der einzelnen Spaltenwerte erforderlichen Speicherplatz. Sie werden allerdings bei der Verwendung von long-Spalten in SQL auf eine Reihe von Einschränkungen stoßen. Sie können keine Oracle-Funktionen oder Operatoren verwenden, um den Inhalt einer long-Spalte zu suchen oder zu ändern. Wenn Sie eine long-Spalte suchen wollen, gibt Oracle eine Fehlermeldung zurück. In gewissem Sinn können Sie sich eine long-Spalte als großen Behälter vorstellen, in dem Sie Daten speichern oder abrufen, aber nicht bearbeiten können.



Denken Sie daran: Es ist nur eine long-Spalte pro Tabelle verfügbar.

---

## Große Binärobjecte (BLOBs)

Wie Sie wahrscheinlich wissen, erlauben die meisten Datenbanken das Speichern großer Binärobjecte (BLOBs). BLOBs umfassen Dokumente, Grafiken, Klänge, Videos - eigentlich alle Arten von Binärdateien, die Sie sich vorstellen können. Der Oracle-Datentyp long raw dient zur Speicherung von BLOBs.

Wenn Sie ein BLOB mit einer »normalen« Zeile verbinden möchten, stehen Ihnen dafür zwei Möglichkeiten zur Verfügung:

- Speichern Sie das BLOB in einer Datei des Betriebssystems (wie z.B. einer MS-DOS-Datei), und legen Sie das Verzeichnis und den Dateinamen in der entsprechenden Tabelle ab.
- Speichern Sie das BLOB selbst in einer long raw-Spalte.

Einige Entwickler bevorzugen die erste Methode. Sie meinen, daß das BLOB leichter verfügbar ist, wenn es in einem Dateisystem anstatt in der Oracle-Datenbank gespeichert ist. Sie argumentieren, daß ihnen das Speichern eines BLOBs in einer Tabellenspalte nur wenig Nutzen bringe, wenn die Oracle-Datenbank das BLOB nicht bearbeiten kann (oder sollte).

Andere Entwickler sehen dagegen einen Vorteil darin, alle Daten zentral in der Oracle-Datenbank zu speichern. Sie argumentieren, daß dieser Ansatz für eine bessere Übertragbarkeit Sorge; wenn der Verweis auf ein Verzeichnis und einen Dateinamen entfällt, bleiben weniger BS-spezifische Punkte, mit denen man sich befassen muß.

Eine als long raw definierte Spalte faßt bis zu 2 Gbyte pro Zeile. Wie die long-Spalten so weisen auch long raw-Spalten eine Reihe von Einschränkungen auf. Sie können zum Beispiel bei long raw-Spalten keine der integrierten Funktionen verwenden.

## Der Datentyp raw

Oracle stellt auch den Datentyp raw bereit, der bis zu 255 Byte an Binärdaten fassen kann. Aufgrund der Speicherbegrenzung ist eine raw-Spalte weniger nützlich als eine long raw-Spalte.

## Zusammenfassung

Versuchen Sie, wie in dieser Lektion erläutert, beim Entwerfen eines logischen Datenmodells folgende Schritte zu beachten:

1. Beginnen Sie mit einem logischen Datenmodell.
2. Identifizieren Sie jede Entität im zu modellierenden System.
3. Benennen Sie einen Primärschlüssel für jede Entität.
4. Bestimmen Sie die zu erstellenden Fremdschlüssel.

Beachten Sie bei der Arbeit mit den Oracle-Datentypen folgende Grundlagen:

- Jede Spalte in einer Tabelle muß einen Oracle-Datentyp enthalten.
- Verwenden Sie number zum Speichern numerischer Daten.

- Verwenden Sie char zum Speichern einzelner Zeichen.
- Verwenden Sie varchar2 zum Speichern von Zeichenketten, die weniger als 2000 Zeichen enthalten. varchar2 unterstützt die in Oracle integrierten Funktionen und Operatoren.
- Verwenden Sie long, wenn Sie sehr lange Zeichenfolgen speichern müssen.
- Verwenden Sie date zum Speichern aller Datums- und Zeitinformationen.
- Verwenden Sie long raw zum Speichern von bis zu 2 Gbyte großen BLOBs.

## Wie geht es weiter?

Am Tag 4 lernen Sie, wie Sie die Tabellen und andere Strukturen, welche ein logisches Datenmodell implementieren, tatsächlich aufbauen.

## Fragen und Antworten

Frage:

Stellt die Leistung nicht ein Problem dar, wenn alle Tabellen einer Anwendung in der dritten Normalform vorliegen?

Antwort:

*Es ist falsch, anzunehmen, daß die Leistung einer Datenbankanwendung unter dem gültigen Standard liegt, wenn die Tabellen in der dritten Normalform vorliegen. Es ist besser, mit einem Entwurf in der dritten Normalform - oder höher - zu beginnen und dann Bereiche zu ermitteln, in denen die Leistung ein Problem darstellen könnte.*

Frage:

Wie viele Spalten kann eine Tabelle in einer Oracle-Datenbank enthalten?

Antwort:

*Eine Tabelle kann bis zu 254 Spalten enthalten.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Richtig oder falsch? Wenn ein Attribut, das nicht Teil des Primärschlüssels ist, einen Fremdschlüssel darstellt, ist es unbedingt erforderlich; es läßt keinen Nullwert zu.
2. Kann die Student-Schedule-Tabelle verwendet werden, wenn ein Student einen Kurs wiederholen muß? Warum bzw. warum nicht?
3. Welche Eigenschaft einer relationalen Datenbank verhindert, daß eine Klasse aus der Class-Tabelle gelöscht wird, wenn in der Student-Schedule-Tabelle Zeilen existieren, welche die zu löschende Class ID enthalten?

4. Aus der Sicht der Entitätsbeziehung gibt es eine Beziehung zwischen der Class- und der Instructor-Tabelle. Handelt es sich hierbei um eine identifizierende oder nichtidentifizierende Beziehung?
5. Richtig oder falsch? Wenn Sie den entsprechenden Index für eine Tabelle erstellen, werden die Zeilen stets in aufsteigender Reihenfolge der indizierten Spalten abgerufen.

## Übungen

1. Angenommen, Sie möchten den Kursleiter identifizieren, der auch Bereichsleiter ist. Können Sie mindestens zwei Möglichkeiten nennen, dies herauszufinden? Wie lauten die Stärken und Schwächen der jeweiligen Ansätze?
2. Die Student-Schedule-Tabelle enthält zur Zeit sowohl die derzeitigen Klassen als auch die früheren Klassen, die ein Student besucht hat. Schlagen Sie einen alternativen Entwurf mit zwei Tabellen vor, wobei eine Tabelle den aktuellen Stundenplan und die zweite Tabelle die Klassen enthält, die der Student früher besucht hat. Nennen Sie alle Attribute in den zwei Tabellen. Welche Vorteile hat dieser Entwurf?

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖ Kapitel ❖ Top Inhalt Index **SAMS** Kapitel ❖

# 4

## Implementierung Ihres logischen Modells: Physischer Datenbankentwurf

[Grundlagen der create table-Anweisung](#)

[Verwendung von alter table zur Änderung der Tabellendefinition](#)

[Verwendung eines Primärschlüssels](#)

[Der Fremdschlüssel und referentielle Integrität](#)

[Tabellenindizes](#)

[Integritätsregeln ansehen](#)

[Einschränken von Werten mit einer Check-Regel für Spalten](#)

[Allgemeine Beschränkungen für die Änderung einer Tabelle](#)

[Skript für die Erstellung der Beispieldatenbank](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

# 4

# Implementierung Ihres logischen Modells: Physischer Datenbankentwurf

Die vorhergehende Lektion, Tag 3, »Logischer Datenbankentwurf«, befaßte sich mit den Elementen eines Datenmodells, wobei die Implementierung dieses Modells in eine »reale« Datenbank nicht berücksichtigt wurde. In dieser Lektion werden Sie sehen, wie ein logisches Datenmodell tatsächlich in eine Oracle-Datenbank implementiert wird. Diese Lektion behandelt einen weiteren Teil von SQL: die Data Definition Language (DDL, Datendefinitionssprache). DDL besteht aus SQL-Anweisungen zum Erzeugen, Ändern und Verwerfen von Datenbankobjekten.



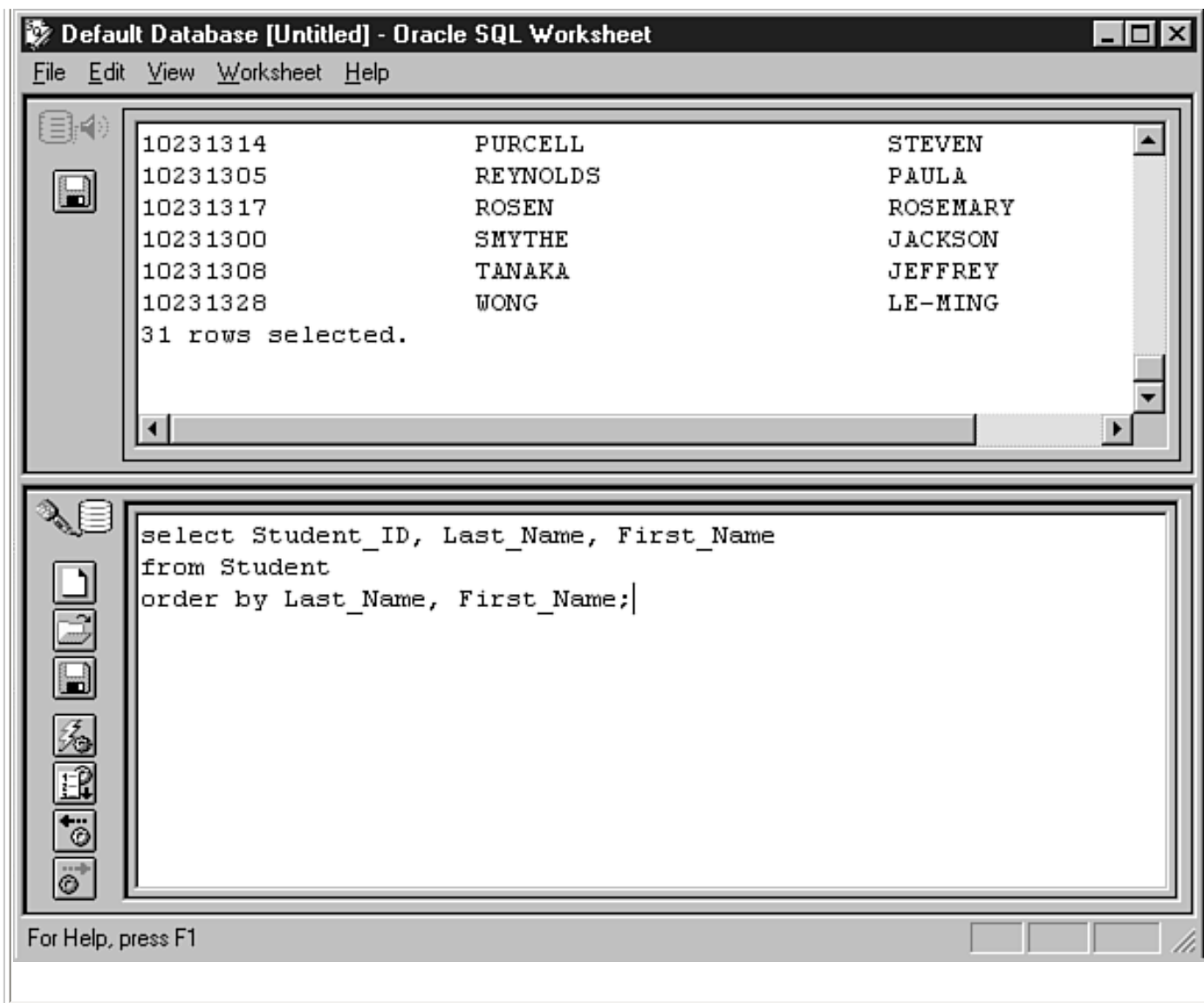
*DDL steht für Data Definition Language, die aus jenen SQL-Anweisungen besteht, welche zum Erzeugen oder Ändern von Datenbankstrukturen, wie z.B. Tabellen, Datensichten und Indizes, eingesetzt werden.*

Wenn Sie häufig mit der Entwicklung von Datenbankanwendungen betraut sind, sollten Sie unbedingt den Einsatz eines Werkzeugs für den Datenbankentwurf unter Windows erwägen, wie z.B. Designer/2000 oder Database Designer von Oracle, ERwin von Logic Works oder S-Designer von Sybase. Diese Produkte erlauben Ihnen, ein logisches Modell grafisch zu definieren und die richtigen SQL-Anweisungen zur Erstellung einer Datenbank zu erzeugen.

Selbst wenn Sie ein Werkzeug für den Datenbankentwurf verwenden, sollten Sie sich ausreichende Kenntnisse über die Werkzeuge von Oracle aneignen. Es gibt viele Werkzeuge, die eine Oracle-Datenbank mit einer SQL-Benutzeroberfläche ausstatten. Zwei Beispiele hierfür sind SQL\*Plus und SQL Worksheet, eine Komponente des Oracle Enterprise Managers. Diese Werkzeuge ermöglichen Ihnen die Eingabe von SQL-Anweisungen zum Erzeugen von Tabellen, Indizes und anderen Datenbankobjekten. Bei den meisten Beispielen in dieser Lektion wird SQL\*Plus verwendet. Abbildung 4.1 zeigt, wie Sie eine SQL-Anweisung mit Hilfe von SQL Worksheet ausführen.

*Abbildung 4.1:  
Ausführung einer SQL-Anweisung mit Hilfe von SQL Worksheet.*





Eine weitere Alternative ist der Einsatz eines Werkzeugs wie des Personal Oracle8 Navigators oder des Oracle Schema Managers, die eine grafische Benutzeroberfläche für die Bearbeitung einer Oracle-Datenbank bereitstellen. Wenn Sie Personal Oracle8 für Windows 95 verwenden, sollten Sie sich bei der Erstellung und Änderung von Tabellen die intuitive Oberfläche des Navigators zunutze machen (siehe Abbildung 4.2). Im Gegensatz zu SQL\*Plus oder SQL Worksheet erfordert der Navigator keine SQL-Kenntnisse. Der Navigator »packt« Ihre Eingaben und übergibt sie an den Kern der Oracle-Datenbank. Diese Werkzeuge sind ideal für Benutzer, die nicht mit SQL vertraut sind und die Sprache auch nicht lernen wollen. Der Navigator kann sowohl mit Personal Oracle als auch Oracle-Ferndatenbankkernen eingesetzt werden.

Abbildung 4.2:  
Erstellen und Ändern  
von Tabellen mit  
Hilfe des Personal  
Oracle Navigators.

**Design Table**

Name: INSTRUCTOR

Owner: FLUGLE

Columns:

|   | Column Name | Type     | Size | Scale | Default | Can't Null |
|---|-------------|----------|------|-------|---------|------------|
| 1 | INSTRUCTOR  | VARCHAR2 | 20   | 0     |         | ✓          |
| 2 | DEPARTMENT  | VARCHAR2 | 20   | 0     |         |            |
| 3 | LAST_NAME   | VARCHAR2 | 25   | 0     |         |            |
| 4 | FIRST_NAME  | VARCHAR2 | 25   | 0     |         | ✓          |
| 5 | MI          | VARCHAR2 | 1    | 0     |         | ✓          |
| 6 | POSITION    | VARCHAR2 | 25   | 0     |         | ✓          |
| 7 | TELEPHONE   | VARCHAR2 | 10   | 0     |         | ✓          |
| 8 | FAX         | VARCHAR2 | 10   | 0     |         |            |

Buttons: OK, Cancel, Help, Remove, Up, Down

## Grundlagen der create table-Anweisung

Aufgrund ihrer vielen Optionen und Klauseln kann die SQL-Anweisung create table recht komplex ausfallen.



Es folgt eine vereinfachte Form ihrer Syntax:

```
CREATE TABLE table_name (  
  column_name1 datatype [NOT NULL],  
  ...  
  column_nameN datatype [NOT NULL]);
```

Die Definition der Variablen lautet wie folgt:

*table\_name* steht für den Namen der Tabelle.

*column\_name1* bis *column\_nameN* sind gültige Spaltennamen.

*datatype* ist die gültige Angabe des Oracle-Datentyps.

Die create table-Anweisung kann direkt aus SQL\*Plus oder SQL Worksheet aufgerufen werden. Der Personal Oracle8 Navigator oder der Schema Manager, eine Komponente des Oracle Enterprise Managers, integrieren die Benutzereingaben in ein Paket und aktivieren indirekt die create table-Anweisung. Die Beispiele in dieser Lektion wurden mit SQL\*Plus erstellt.

## Benennen einer Tabelle

Oracle schränkt Tabellennamen in gewisser Weise ein:

- Jede Tabelle, die einem Oracle-Account gehört, muß einen eindeutigen Namen haben.
- Ein Tabellename darf nicht länger als 30 Zeichen sein.
- Ein Tabellename muß mit einem Buchstaben beginnen.
- Ein Tabellename kann die Buchstaben A bis Z, die Zahlen 0 bis 9 sowie die Zeichen \$, # und \_ (Unterstrich) enthalten.
- Ein Tabellename darf kein von SQL reserviertes Wort enthalten. (Eine vollständige Liste der reservierten Wörter finden Sie im *SQL-Referenzhandbuch* von Oracle Server. Falls Sie die Dokumentation zu Oracle auf Ihrem PC installiert haben, finden Sie das Dokument unter C:\Orawin95\Doc\A32538\_1.pdf).
- Bei der Benennung von Tabellen können Sie Groß- und Kleinbuchstaben verwenden; Oracle ignoriert die Groß- und Kleinschreibung, sofern die Tabellen- oder Spaltennamen nicht in doppelten Anführungszeichen stehen. Bei der Verwendung von doppelten Anführungszeichen handelt es sich um eine von Oracle unterstützte Syntax. Sie werden allerdings nie einen Grund haben, doppelte Anführungszeichen direkt verwenden zu müssen. Einige Fremdwerkzeuge (wie z.B. PowerBuilder) machen sich diese Syntax zunutze, dies ist aber für den Entwickler erkennbar.
- Der Name einer Tabelle sollte aussagekräftig sein. Benutzen Sie bei der Benennung von Tabellen und anderen Datenobjekten keine übertriebenen Abkürzungen. Viele Werkzeuge für die Anwendungsentwicklung, wie z.B. PowerBuilder oder Oracle Forms, stellen für die Auswahl einer Tabelle eine intuitive Oberfläche bereit. Da keine Eingaben erforderlich sind, ist ein langer Tabellename genauso einfach auszuwählen wie ein kurzer Tabellename.

#### Was Sie tun sollen

#### Was nicht

Verwenden Sie bei der Benennung von Tabellen aussagekräftige Namen.

Wählen Sie einen eindeutigen Begriff als Tabellennamen.

Gebrauchen Sie keine verschlüsselten Codes oder Zahlen in Ihrem Tabellennamen.

Benutzen Sie bei der Namensgebung für Ihre Tabelle nicht die Begriffe table oder tbl; sie sind überflüssig.

### Benennen einer Spalte

Es folgen einige Gedanken zur Benennung von Spalten:

- Innerhalb einer Tabelle muß der Spaltenname eindeutig sein. Sie können aber denselben Spaltennamen in unterschiedlichen Tabellen verwenden.
- Wie bei anderen Datenbankobjekten kann der Name einer Spalte aus bis zu 30 Zeichen bestehen.
- Der Spaltenname muß mit einem Buchstaben beginnen.
- Der Spaltenname kann die Buchstaben A bis Z, die Ziffern 0 bis 9 sowie die Zeichen \$, # und \_ (Unterstrich) enthalten. Leerzeichen sind nicht erlaubt.
- Bei dem Spaltennamen darf es sich nicht um ein von SQL reserviertes Wort handeln.
- Wie bei Tabellen sollten Sie auch für Spalten einen aussagekräftigen Namen wählen. Aussagekräftige Namen lassen die Definition der einzelnen Spalten für den Benutzer deutlicher erscheinen.
- Eine Oracle-Tabelle kann aus bis zu 254 Spalten bestehen.

## Beispiele für die Erstellung von Tabellen

Dieser Abschnitt beginnt mit einem einfachen Beispiel. Sie werden eine create table-Anweisung schreiben, um eine Tabelle mit den am Flugle-College vorhandenen Fachbereichen (Department) zu erstellen, wie in Listing 4.1 dargestellt.

### *Listing 4.1: Erstellung der Department-Tabelle.*

```
SQL> Create table Department
2 (Department_ID Varchar2(20) NOT NULL,
3 Department_Name Varchar2(25));
```

Table created.

Einige Aspekte hinsichtlich der Syntax von Anweisungen sind besonders zu erwähnen:

- Klammern schließen die Liste der Spalten ein.
- In erforderlichen Spalten folgt der Ausdruck NOT NULL auf die Nennung des Datentyps der Spalte.
- Wenn nötig, wird die Spaltenbreite nach dem Datentyp in Klammern gesetzt.

Sie haben die Möglichkeit, die Eigenschaft einer Spalte ausdrücklich als optional festzulegen, indem Sie nach der Nennung des Datentyps NULL eingeben, wie in Listing 4.2 dargestellt.

### *Listing 4.2: Erstellung der Student-Tabelle.*

```
SQL> create table Student
2 (Student_ID Varchar2(20) NOT NULL,
3 Last_Name Varchar2(25) NOT NULL,
4 First_Name Varchar2(25) NOT NULL,
5 MI Varchar2(1) NULL,
6 Year Varchar2(25) NULL,
7 Street_Address Varchar2(25) NULL,
8 City Varchar2(25) NULL,
9 State Varchar2(2) NULL,
10 Zipcode Varchar2(9) NULL,
11 Telephone Varchar2(10) NULL,
12 Fax Varchar2(10) NULL,
13 Email Varchar2(100) NULL);
```

Table created.

Wie ich bereits erwähnt habe, ignoriert Oracle bei der Implementierung von SQL die Groß- und Kleinschreibung (was nicht bei allen Herstellern der Fall ist), solange die Tabellen- und Spaltennamen nicht in doppelten Anführungszeichen stehen. Listing 4.3 zeigt ein Beispiel für die Student-Tabelle, in dem die Tabellen- und Spaltennamen in doppelten Anführungszeichen stehen. Allerdings erkennt Oracle diese Namen nur, wenn sie immer in doppelte Anführungszeichen gesetzt werden.

### *Listing 4.3: Die Tabellen- und Spaltennamen werden in Anführungszeichen gesetzt.*

```
SQL> create table "student"
2 ("student_id" Varchar2(20) NOT NULL,
3 "last_name" Varchar2(25) NOT NULL,
```

```
4 "first_name" Varchar2(25) NOT NULL,  
5 "mi" Varchar2(1),  
6 "year" Varchar2(25),  
7 "street_address" Varchar2(25),  
8 "city" Varchar2(25),  
9 "state" Varchar2(2),  
10 "zipcode" Varchar2(10),  
11 "telephone" Varchar2(10),  
12 "fax" Varchar2(10),  
13 "email" Varchar2(100));
```

Table created.

```
SQL> describe student
```

Object does not exist.

```
SQL> describe "student"
```

Name Null? Type

```
-----  
student_id NOT NULL VARCHAR2(20)  
last_name NOT NULL VARCHAR2(25)  
first_name NOT NULL VARCHAR2(25)  
mi VARCHAR2(1)  
year VARCHAR2(25)  
street_address VARCHAR2(25)  
city VARCHAR2(25)  
state VARCHAR2(2)  
zipcode VARCHAR2(10)  
telephone VARCHAR2(10)  
fax VARCHAR2(10)  
email VARCHAR2(100)
```

## Festlegen des Primärschlüssels

Bei dem Primärschlüssel einer Tabelle handelt es sich um die Gruppe von Spalten, welche die einzelnen Zeilen der Tabelle eindeutig bezeichnen.



```
TABLE table_name (  
  column_name1 datatype [NOT NULL],  
  ...  
  column_nameN datatype [NOT NULL],  
  [Constraint constraint_name]  
  [Primary key (column_nameA, column_nameB, ... column_nameX)]);
```

Die Definition der Variablen lautet wie folgt:

*table\_name* steht für den Namen der Tabelle.

*column\_name1* bis *column\_nameN* sind gültige Spaltennamen.

*datatype* ist die Angabe eines gültigen Oracle-Datentyps.

*constraint\_name* ist ein optionaler Name, welcher die Integritätsregel des Primärschlüssels identifiziert.

*column\_nameA* bis *column\_nameX* bezeichnen die Tabellenspalten, die den Primärschlüssel bilden.

Obwohl die Klausel für den Primärschlüssel in der create table-Anweisung einen wichtigen Bestandteil der Theorie relationaler Datenbanken darstellt, erfolgt ihre Anwendung in der Praxis optional. Unter nochmaliger Verwendung der Department-Tabelle zeigt Listing 4.4, wie der Primärschlüssel bei der Erstellung der Tabelle deklariert wird.

**Listing 4.4: Deklaration des Primärschlüssels bei Erstellung der Tabelle.**

```
SQL> create table Department
2 (Department_ID Varchar2(20),
3 Department_Name Varchar2(25),
4 Constraint PK_Department
5 Primary Key (Department_ID));
```

Table created.



*Primärschlüssel unterliegen gewissen Einschränkungen. Erstens kann eine Spalte, die Teil des Primärschlüssels ist, keinen Nullwert haben. Zweitens kann eine als long oder long raw definierte Spalte nicht Teil des Primärschlüssels sein. Drittens beträgt die maximale Anzahl von Spalten im Primärschlüssel 16. Sie können letztere Einschränkung durch die Verwendung eines Ersatzschlüssels umgehen - eines künstlichen Werts, der garantiert alle Zeilen einer Tabelle eindeutig identifiziert.*

---

## Festlegen von Fremdschlüsseln

Nun, da Sie wissen, wie der Primärschlüssel in der create table-Anweisung angegeben wird, sind Sie bereit, Ihre Fremdschlüssel zu deklarieren.



```
TABLE table_name (
column_specification1,
...
column_specificationN,
[Constraint constraint_name Foreign key (column_nameF1,...column_nameFN)
references referenced_table (column_nameP1,...column_namePN), ]
(column_nameA, column_nameB, ... column_nameX));
```

Die Definition der Variablen lautet wie folgt:

*table\_name* steht für den Namen der Tabelle.

*column\_specification* bis *column\_specificationN* sind gültige Spaltenangaben (wie unten im einzelnen beschrieben).

*constraint\_name* ist der Name der Integritätsregel, die Sie dem Fremdschlüssel zuordnen möchten.

*referenced\_table* ist der Name der von der Deklaration des Fremdschlüssels referenzierten Tabelle.

*column\_nameF1* bis *column\_nameFN* sind die Spalten, die den Fremdschlüssel bilden.

*column\_nameP1* bis *column\_namePN* sind die Spalten, die den Primärschlüssel in *referenced\_table* bilden.

*column\_nameA* bis *column\_nameX* sind die Tabellenspalten, die den Primärschlüssel bilden.



*Es folgt die Syntax für column\_specification.*

---

```
column_name datatype [DEFAULT default value]  
[CONSTRAINT constraint_name] [NULL] | [NOT NULL] | [UNIQUE] | [CHECK (condition)
```

Die Definition der Variablen lautet wie folgt:

*column\_name* steht für einen gültigen Oracle-Spaltennamen.

*datatype* ist die gültige Angabe eines Oracle-Datentyps.

*default\_value* ist ein zulässiger Standardwert, welcher der Spalte durch eine Einfügung zugeordnet wurde.

*constraint\_name* ist ein zulässiger Name, welcher der Integritätsregel zugeordnet werden soll.

*condition* ist eine gültige boolesche Bedingung in Oracle, die auf einen einer Spalte zuzuordnenden Wert zutreffen muß.

---



*Eine Integritätsregel ist ein Mechanismus, der gewährleistet, daß die Werte einer Spalte oder einer Spaltengruppe einer deklarierten Bedingung entsprechen.*

---

Für die Course-Tabelle müssen Sie einen Primär- und einen Fremdschlüssel deklarieren, wie in Listing 4.5 gezeigt.

**Listing 4.5: Deklaration von Primär- und Fremdschlüsseln bei Erstellung der Tabelle.**

```
SQL> create table Course
```

```
2 (Course_ID Varchar2(5),
3 Department_ID Varchar2(20) NOT NULL,
4 Title Varchar2(60) NOT NULL,
5 Description Varchar2(2000),
6 Units Number,
7 Additional_Fees Number,
8 Constraint PK_Course
9 Primary key (Course_ID, Department_ID),
10 Constraint FK_Course_Department_ID
11 Foreign key (Department_ID) references Department (Department_ID));
```

Table created.



*Auch wenn die Verwendung von Namen für Integritätsregeln optional ist, empfehle ich, diese zu benutzen, wenn Sie einen Primär- oder Fremdschlüssel oder eine Check-Regel deklarieren. Wenn Sie für die Integritätsregel keinen Namen ausdrücklich festlegen, erzeugt Oracle automatisch eine Integritätsregel und weist dieser einen recht verschlüsselten Namen zu. Wollen Sie den Fremdschlüssel löschen, müssen Sie den von Oracle erzeugten Namen der Integritätsregel auffindig machen. Sie ersparen sich selbst viel Kummer, wenn Sie zuerst eine Integritätsregel deklarieren.*

Die Spalten der referenzierten Tabelle müssen den Primär- oder den eindeutigen Schlüssel der referenzierten Tabelle bilden. Ist das nicht der Fall, erstellt Oracle den Fremdschlüssel nicht. Im folgenden Beispiel gehen wir davon aus, daß Sie den Fremdschlüssel für die Course-Tabelle noch nicht erzeugt haben. Löschen wir zuerst den Primärschlüssel der Department-Tabelle. Wenn Sie versuchen, die Course-Tabelle zu erstellen, wird Oracle feststellen, daß Department\_ID nicht als Primärschlüssel für die Department-Tabelle definiert wurde; die create table-Anweisung führt nicht zum Erfolg, wie Listing 4.6 zeigt.

**Listing 4.6: Ein Fremdschlüssel kann nicht deklariert werden, weil es keinen zugehörigen Primärschlüssel gibt.**

```
SQL> alter table Department drop primary key;
```

Table altered.

```
SQL> create table Course
2 (Course_ID Varchar2(5),
3 Department_ID Varchar2(20) NOT NULL,
4 Title Varchar2(60) NOT NULL,
5 Description Varchar2(2000),
6 Units Number,
7 Additional_Fees Number,
8 Constraint PK_Course
9 Primary key (Course_ID, Department_ID),
10 Constraint FK_Course_Department_ID
11 Foreign key (Department_ID) references Department (Department_ID));
```

```
Foreign key (Department_ID) references Department (Department_ID))
```

\*

ERROR at line 11:



ORA-02270: no matching unique or primary key for this column-list

## Einschränken eines Spaltenwerts mit der check-Klausel

Ein leistungsfähiges Merkmal von SQL ist die Möglichkeit, bei der Erstellung einer Tabelle die Validierung der elementaren Daten einer Tabelle festzulegen. Diese Aufgabe erledigt SQL mit einer optionalen check-Klausel, die für jede Spalte angegeben werden kann. Bei der check-Klausel handelt es sich um eine boolesche Bedingung, die entweder true (wahr) oder false (falsch) lautet. Erweist sich die Bedingung als true, wird der Spaltenwert von Oracle akzeptiert; erweist sich die Bedingung als false, gibt Oracle einen Fehlercode aus.

Listing 4.7 zeigt ein Beispiel, in dem Sie die Definition der Instructor-Tabelle so verändern werden, daß die Position eines Kursleiters mit einem der drei Werte - assistant professor (Lehrbeauftragter), associate professor (planmäßiger außerordentlicher Professor) oder professor - wiedergegeben wird.

*Listing 4.7: Festlegung einer check-Regel für eine Spalte.*

```
SQL> create table Instructor
2 (Instructor_ID varchar2(20),
3 Department_ID varchar2(20) NOT NULL,
4 Last_Name varchar2(25) NOT NULL,
5 First_Name varchar2(25),
6 MI varchar2(1),
7 Position varchar2(25) constraint CK_Instructor
8 check (Position in ('ASSISTANT PROFESSOR', 'ASSOCIATE PROFESSOR',
9 'PROFESSOR')),
10 Telephone varchar2(10),
11 Fax varchar2(10),
12 Email varchar2(100),
13 constraint PK_Instructor Primary Key (Instructor_ID),
14 constraint FK_Instructor_Instructor_ID
15 Foreign Key (Department_ID) references Department (Department_ID));
```

Table created.

Wenn Sie versuchen, der Tabelle einen Kursleiter hinzuzufügen, dessen Position keinem der drei zulässigen Werte entspricht, gibt Oracle einen Fehlercode aus, welcher aussagt, daß eine check-Regel verletzt wurde, wie in Listing 4.8 dargestellt.

*Listing 4.8: Die check-Regel wird durchgesetzt.*

```
SQL> insert into Instructor
2 (Instructor_ID, Department_ID, Last_Name, First_Name, Position)
3 values
4 ('C2222', 'PSYCH', 'MENCKEN', 'H.', 'CURMUDGEON');
```

```
insert into Instructor
*
```

ERROR at line 1:

ORA-02290: check constraint (TY02.CK\_INSTRUCTOR) violated



Da es sich bei curmudgeon (Brummbär) um keine gültige Kursleiterposition handelt, schlägt der Versuch, die Zeile einzufügen, fehl. Dies ist ein Beispiel dafür, wie eine check-Regel die Datenintegrität durchsetzt.

---

## Festlegen eines Standardwerts für eine Spalte

Durch die Verwendung der default-Klausel bei der Definition einer Spalte können Sie einen Standardwert für diese Spalte festlegen. Dieser Standardwert wird immer dann auf eine Spalte angewendet, wenn eine Zeile in die Tabelle eingefügt wird, ohne die Spalte in der insert-Anweisung zu nennen.

Nehmen wir zum Beispiel an, daß die meisten Studenten in Springfield leben. Aus diesem Grund möchten Sie sicherstellen, daß beim Einfügen eines Studenten oder einer Studentin in die Student-Tabelle der Standardwert springfield eingesetzt wird, wenn kein Ort angegeben ist. Listing 4.9 verdeutlicht, wie Sie die create table-Anweisung ändern würden, um zu diesem Ergebnis zu gelangen. Nach Festlegung eines Standardwerts wird eine in die Student-Tabelle eingefügte Zeile auf den Standardwert springfield gesetzt, sofern sie keinen Wert für City (Ort) enthält.

### *Listing 4.9: Festlegung eines Standardwerts für eine Spalte.*

```
SQL> create table Student
2 (Student_ID Varchar2(20),
3 Last_Name Varchar2(25) NOT NULL,
4 First_Name Varchar2(25) NOT NULL,
5 MI Varchar2(1),
6 Year Varchar2(25),
7 Street_Address Varchar2(25),
8 City Varchar2(25) Default 'SPRINGFIELD',
9 State Varchar2(2),
10 Zipcode Varchar2(9),
11 Telephone Varchar2(10),
12 Fax Varchar2(10),
13 Email Varchar2(100),
14 Constraint PK_Student Primary Key (Student_ID));
```

Table created.

```
SQL> insert into Student
2 (Student_ID, Last_Name, First_Name, Year)
3 values
4 ('109850', 'CARSON', 'RICARDO', 'JUNIOR');
```

1 row created.

```
SQL> select Student_ID, Last_Name, City
2 from Student
3 where
4 Student_ID = '109850';
```

```
STUDENT_ID LAST_NAME CITY
-----
109850 CARSON SPRINGFIELD
```

## Verwendung von alter table zur Änderung der Tabellendefinition

Manchmal ist es notwendig, die Definition einer Tabelle zu ändern. Diesem Zweck dient die alter table-Anweisung. Diese Anweisung verändert zwar die Struktur einer Tabelle, nicht aber deren Inhalt. Mit alter table können Sie eine Tabelle wie folgt ändern:

- eine neue Spalte in eine vorhandene Tabelle einfügen
- eine vorhandene Spalte vergrößern oder verkleinern
- die Eigenschaft einer bestehenden Spalte von erforderlichlich in optional oder umgekehrt ändern
- einen Standardwert für eine vorhandene Spalte bestimmen
- andere Integritätsregeln für eine bestehende Spalte festlegen



*Es folgen die vier Grundformen einer alter table-Anweisung:*

---

```
ALTER TABLE table_name
ADD (column_specification | constraint , ...
column_specification | constraint);
```

```
ALTER TABLE table_name
MODIFY (column_specification | constraint , ...
column_specification | constraint);
```

```
ALTER TABLE table_name
DROP PRIMARY KEY;
```

```
ALTER TABLE table_name
DROP CONSTRAINT constraint;
```

Die Variablen sind wie folgt definiert:

*table\_name* ist der Name der Tabelle.

*column\_specification* ist die gültige Angabe für eine Spalte (Spaltenname und Datentyp).

*constraint* ist eine Integritätsregel für eine Spalte oder Tabelle.

Die erste Form der Anweisung wird verwendet, um eine Spalte, den Primärschlüssel oder einen Fremdschlüssel in eine Tabelle einzufügen.

Die zweite Form wird zum Ändern einer vorhandenen Spalte benutzt. Unter anderem können Sie eine Spalte vergrößern oder eine erforderliche in eine optionale Spalte umwandeln.

Die dritte und vierte Form der alter table-Anweisung werden zum Löschen des Primärschlüssels einer Tabelle oder anderer Integritätsregeln verwendet.

Auf den nächsten Seiten demonstriere ich Ihnen die Verwendung dieser Anweisung anhand mehrerer Beispiele.

## Änderung einer Spaltendefinition von not null in null

Obwohl Sie die Eigenschaft eine Spalte frei von erforderlich in optional ändern können, sollten Sie darüber nachdenken, warum diese Änderung notwendig ist. Ist dieses Attribut wirklich optional, oder arbeiten Sie mit einer Reihe von Testdaten, die für die realistischen Anwendungsdaten nicht repräsentativ sind?

Listing 4.10 zeigt eine Tabelle namens `demo_table`. Die `alter table`-Anweisung wird dazu verwendet, die Definition der Spalte `Record_No` in erforderlich zu ändern.

**Listing 4.10: Änderung der Eigenschaft einer Spalte von optional in erforderlich.**

```
SQL> describe demo_table
```

```
Name Null? Type
```

```
-----  
RECORD_NO NOT NULL NUMBER(38)  
DESCRIPTION VARCHAR2(40)  
CURRENT_VALUE NOT NULL NUMBER
```

```
SQL> alter table demo_table modify (current_value number null);
```

```
Table altered.
```

```
SQL> describe demo_table
```

```
Name Null? Type
```

```
-----  
RECORD_NO NOT NULL NUMBER(38)  
DESCRIPTION VARCHAR2(40)  
CURRENT_VALUE NUMBER
```

## Änderung einer Spaltendefinition von null in not null

Ist eine Tabelle leer, können Sie einer Spalte die Definition `not null` zuweisen. Wenn die Tabelle aber nicht leer ist, können Sie die Eigenschaft einer Spalte nicht in `not null` ändern, solange alle Zeilen in der Tabelle einen Wert für die betreffende Spalte aufweisen. Listing 4.11 zeigt, wie Oracle reagiert, wenn Sie versuchen, die Eigenschaft der Spalte `current_value` in erforderlich zu ändern. Wenn Sie jedoch sicherstellen, daß `current_value` keinen Nullwert in den einzelnen Zeilen der Tabelle aufweist, kann der Wert von `current_value` auf `not null` gesetzt werden.

**Listing 4.11: Versuch, die Eigenschaft einer Spalte in erforderlich zu ändern.**

```
SQL> alter table demo_table modify (current_value number not null);  
alter table demo_table modify (current_value number not null)
```

```
*
```

```
ERROR at line 1:
```

```
ORA-01449: column contains NULL values; cannot alter to NOT NULL
```

```
SQL> update demo_table
2 set current_value = record_no + 100;

4 rows updated.

SQL> select current_value from demo_table;
```

```
CURRENT_VALUE
```

```
-----
```

```
121
122
123
124
```

```
SQL> commit;
```

Commit complete.

```
SQL> alter table demo_table modify (current_value number not null);
```

Table altered.



*Anfangs verfügen alle Zeilen der Tabelle über einen Nullwert für die Spalte `current_value`; daher kann die Eigenschaft der Spalte nicht in erforderlich geändert werden. Um die Spalte in erforderlich zu ändern, müssen alle Zeilen einen Wert für `current_value` haben. Aktualisieren Sie alle Zeilen in der Tabelle, so daß der Spalte `current_value` der Wert der Spalte `record_no` plus 100 zugeordnet wird. Anschließend kann die Definition der Spalte in `NOT NULL` geändert werden.*

---

## Erweitern der Breite einer Spalte

Sie können die `alter table`-Anweisung verwenden, um die Breite einer Zeichenspalte zu verändern. Das in Listing 4.12 dargestellte Beispiel zeigt die Verwendung dieser Anweisung. Die ursprüngliche Breite der `Description`-Spalte beträgt 40 Zeichen. Um die Breite der `Description`-Spalte von 40 auf 50 Zeichen zu erhöhen, wird die Spaltendefinition mit der `alter table`-Anweisung geändert.

### *Listing 4.12: Vergrößerung der Breite einer Spalte.*

```
SQL> describe demo_table
```

```
Name Null? Type
```

```
-----
```

```
RECORD_NO NOT NULL NUMBER(38)
```

```
DESCRIPTION VARCHAR2(40)
```

```
CURRENT_VALUE NOT NULL NUMBER
```

```
SQL> alter table demo_table modify (description varchar2(50));
```

Table altered.

```
SQL> describe demo_table
```

```
Name Null? Type
```

```
-----  
RECORD_NO NOT NULL NUMBER(38)  
DESCRIPTION VARCHAR2(50)  
CURRENT_VALUE NOT NULL NUMBER
```

## Verringern der Breite einer Spalte

Während des Entwurfs einer Datenbankanwendung könnten Sie feststellen, daß Sie sich bei der Größe einer Spalte geirrt haben: Sie haben die Spalte größer angelegt als notwendig. Anfangs können Sie sich vielleicht nicht vorstellen, daß es problematisch sein könnte, wenn eine Spalte größer als nötig ist, da trotzdem alle Daten in dieser Spalte Platz finden. Dennoch stellt dies ein Problem dar, weil es darauf hindeutet, daß Ihr Datenmodell ungenau ist. Wenn Sie »das Richtige tun wollen«, müssen Sie die Spalte verkleinern, so daß die Spaltendefinition den für die Daten reservierten Platz widerspiegelt.

Sie können die alter table-Anweisung auch dazu verwenden, die Breite einer Spalte zu verringern. Nehmen wir zum Beispiel an, Sie wollen für die Angabe des aktuellen Jahrgangs des Studenten ein Zeichen verwenden - 1 für Erstsemester, 2 für das zweite Jahr usw. Allerdings ist die Spalte Year zur Zeit mit varchar2(25) definiert. Listing 4.13 zeigt, was geschieht, wenn Sie dieses ausprobieren wollen.

*Listing 4.13: Versuch, die Breite einer Spalte zu verringern.*

```
SQL> alter table Student  
2 modify  
3 (Year varchar2(1));  
(Year varchar2(1))  
*
```

```
ERROR at line 3:  
ORA-01441: column to be modified must be empty to decrease column length
```

Wie Sie vielleicht bereits vermuten, gestattet es Oracle nicht, die Breite einer Spalte zu ändern, wenn dieser Spalte Werte zugeordnet sind. Selbst wenn Sie die Student-Tabelle so aktualisieren, daß alle Werte für Year aus nicht mehr als fünf Zeichen bestehen, läßt Oracle es immer noch nicht zu, die Breite der Spalte zu verringern, wie Sie aus Listing 4.14 entnehmen können. Sie können die Spalte erst verkleinern, wenn deren Wert auf null gesetzt wurde.

*Listing 4.14: Ein weiterer Versuch, die Breite einer Spalte zu verringern.*

```
SQL> update Student  
2 set Year = substr(Year,1,5);
```

```
31 rows updated.
```

```
SQL> alter table Student modify  
2 (Year varchar2(5));  
(Year varchar2(5))  
*
```

```
ERROR at line 2:  
ORA-01441: column to be modified must be empty to decrease column length
```

```
SQL> update Student
```

```
2 set Year = NULL;
```

31 rows updated.

```
SQL> alter table Student modify  
2 (Year varchar2(5));
```

Table altered.

Wenn die Spalte erst einmal auf null gesetzt wurde, sind natürlich alle Werte verloren. In der Realität hätten Sie eine Kopie der Tabelle angefertigt, um die Spaltenwerte neu laden zu können; mehr darüber erfahren Sie am Tag 6, »Ändern von Daten mit Hilfe von SQL«.

## Verwendung eines Primärschlüssels

Aus folgenden Gründen sollten Sie sicherstellen, daß Sie einen Primärschlüssel für jede Tabelle in Ihrer Datenbank definieren.

- Ein Primärschlüssel verhindert, daß Zeilen doppelt in eine Tabelle eingefügt werden. Je nach den jeweiligen Gegebenheiten können doppelte Zeilen stören oder eine Anwendung zum Absturz bringen. Es würde Ihnen zum Beispiel nicht gefallen, wenn zwei Fachbereiche denselben Bezeichner hätten; wenn Sie die `department_id` als Primärschlüssel deklarieren, kann das nicht passieren.
- Eine Tabelle muß über einen Primärschlüssel verfügen, wenn ein Fremdschlüssel in einer anderen Tabelle auf sie verweist.

## Definieren eines Primärschlüssels bei Erstellen der Tabelle



*Sie können den Primärschlüssel einer Tabelle mit folgender Syntax in der create table-Anweisung definieren:*

---

```
CREATE TABLE table-name  
(column-definition1,  
...  
column-definitionN,  
[CONSTRAINT constraint-name] PRIMARY KEY (column1, ..., columnN))
```

Die Definitionen der Variablen lauten wie folgt:

*table-name* steht für einen gültigen Oracle-Tabellennamen.

*column-definition1* bis *column-definitionN* sind zulässige Spaltendeklarationen.

*constraint-name* ist der Name, welcher der Integritätsregel des Primärschlüssels zugeordnet wurde.

*column1* bis *columnN* sind Spalten, die den Primärschlüssel bilden.

Listing 4.15 zeigt, wie der Primärschlüssel das Einfügen von doppelten Zeilen in die Tabelle verhindert. Der Primärschlüssel wird durchgesetzt, selbst wenn die erste insert-Transaktion nicht festgeschrieben wurde.



*Als doppelte Zeile wird eine Zeile bezeichnet, deren Spalten mit Primärschlüssel über dieselben Werte verfügen wie die einer anderen Zeile.*

---

**Listing 4.15: Durchsetzen eines Primärschlüssels.**

```
SQL> insert into Department
2 (Department_ID, Department_Name)
3 values
4 ('PSYCH', 'PSYCHOLOGY');
```

1 row created.

```
SQL> insert into Department
2 (Department_ID, Department_Name)
3 values
4 ('PSYCH', 'PSYCHIATRY');
```

```
insert into Department
*
```

ERROR at line 1:

ORA-00001: unique constraint (TYO.PK\_DEPARTMENT) violated

Sie können den Primärschlüssel auch nach Erstellung der Tabelle definieren.

---



*Es gibt Situationen, in denen Sie den Namen einer Integritätsregel kennen müssen. Wenn Sie zum Beispiel einen Fremdschlüssel löschen möchten, müssen Sie die mit diesem Fremdschlüssel verbundene Integritätsregel löschen. Wenn Sie den Namen der Regel für den Fremdschlüssel nicht angegeben haben, müssen Sie ihn suchen. Daher ist es immer sinnvoll, Namen für die Integritätsregeln von Primär- und Fremdschlüsseln festzulegen.*

---



*Sie könnten zum Beispiel die Regeln für Primärschlüssel als PK\_tablename und die Regeln für Fremdschlüssel als FK\_tablename\_column bezeichnen und blieben damit innerhalb der für die Namen von Oracle-Objekten gezogenen Grenze von 30 Zeichen. Sie sollten außerdem Namen für Check- und Unique-Regeln vergeben (z.B. tablename\_column\_CK oder tablename\_column\_UN).*



## Definieren eines Primärschlüssels nach Erstellen der Tabelle

Sie haben auch die Möglichkeit, die Integritätsregel eines Primärschlüssels nach Erstellen einer Tabelle zu definieren. Natürlich darf die Tabelle noch nicht über einen Primärschlüssel verfügen. Um eine Integritätsregel für den Primärschlüssel einer vorhandenen Tabelle zu definieren, verwenden Sie die `alter table`-Anweisung, wie in Listing 4.16 dargestellt.

### *Listing 4.16: Hinzufügen eines Primärschlüssels.*

```
SQL> create table Department
2 (Department_ID Varchar2(20),
3 Department_Name Varchar2(25)
4 constraint NN_Department_Name NOT NULL);
```

Table created.

```
SQL> alter table Department add
2 constraint PK_Department primary key (Department_ID);
```

Table altered.

Die Integritätsregel für den Primärschlüssel wird unabhängig davon durchgesetzt, ob der Primärschlüssel bei der Erstellung der Tabelle oder nachträglich definiert wurde.

## Der Fremdschlüssel und referentielle Integrität

Primär- und Fremdschlüssel wirken bei der Durchsetzung der referentiellen Integrität zusammen. Ein Fremdschlüssel in einer Tabelle ist eine Spalte oder eine Gruppe von Spalten, deren Werte auf die des Primärschlüssels einer anderen Tabelle beschränkt sind. Sie sollten so oft wie möglich Fremdschlüssel definieren. Bei Client-Server-Anwendungen ist die Anwendungssoftware des Client die erste Verteidigungslinie. Die letzte Verteidigungslinie der referentiellen Integrität bilden die für die Datenbank definierten Primär- und Fremdschlüssel.

Wie ein Primärschlüssel kann auch ein Fremdschlüssel gleich beim Erstellen einer Tabelle definiert werden.

## Deklariieren eines Fremdschlüssels bei Erstellen der Tabelle

Um die Definition eines Fremdschlüssels zu verdeutlichen, wollen wir uns den Department- und Instructor-Tabellen der Beispieldatenbank des Flugle-Colleges zuwenden. Jeder Fachbereich verfügt über einen eindeutigen Bezeichner, welcher in der Spalte `Department_ID` abgelegt ist. Jeder Kursleiter darf nur einem einzigen Fachbereich angehören. Bei einer realen Anwendung würden Sie wahrscheinlich die Möglichkeit berücksichtigen, daß ein Kursleiter mehreren Fachbereichen angehört.

Oracle setzt die referentielle Integrität durch, indem jegliche Operationen untersagt werden, durch die ein Fremdschlüssel einen Wert bekommen könnte, der in der Tabelle mit dem referenzierten Primärschlüssel nicht existiert. Listing 4.17 zeigt zum Beispiel, daß Sie keinen Datensatz einfügen können, der einen Kursleiter beschreibt, welcher zu

einer Department ID namens 'physics' gehört, da dieser Bezeichner in der Department-Tabelle nicht existiert.

***Listing 4.17: Erzwingen referentieller Integrität.***

```
SQL> insert into Instructor
2 (Instructor_ID, Department_ID, Last_Name)
3 values
4 ('S1000', 'PHYSICS', 'MICHELSON');
insert into Instructor
*
ERROR at line 1:
ORA-02291: integrity constraint (TYO.FK_INSTRUCTOR_INSTRUCTOR_ID)
violated - parent key not found
```

Als Alternative können Sie sich auch dafür entscheiden, den Datentyp einer Spalte mit Fremdschlüssel nicht anzugeben. Listing 4.18 liefert ein Beispiel, in dem die Deklaration der Department\_ID in der Course-Tabelle keinen Datentyp enthält. Statt dessen sucht Oracle den Datentyp und die Breite der Spalte Department\_ID in der Department-Tabelle aus und verwendet diese Definitionen bei der Erstellung der Course-Tabelle.

***Listing 4.18: Der Datentyp einer Spalte mit Fremdschlüssel ist nicht festgelegt.***

```
SQL> create table Course
2 (Course_ID Varchar2(5),
3 Department_ID constraint NN_Course_Department_ID NOT NULL,
4 Title Varchar2(60)
5 constraint NN_Course_Title NOT NULL,
6 Description Varchar2(2000),
7 Units Number,
8 Additional_Fees Number,
9 Constraint PK_Course
10 Primary key (Course_ID, Department_ID),
11 Constraint FK_Course_Department_ID
12 Foreign key (Department_ID) references Department (Department_ID));
```

Table created.

```
SQL> describe Course
Name Null? Type
-----
COURSE_ID NOT NULL VARCHAR2(5)
DEPARTMENT_ID NOT NULL VARCHAR2(20)
TITLE NOT NULL VARCHAR2(60)
DESCRIPTION VARCHAR2(2000)
UNITS NUMBER
ADDITIONAL_FEES NUMBER
```

Der Vorteil dabei, den Datentyp einer Spalte mit Fremdschlüssel nicht anzugeben, liegt darin, daß die betreffende Spalte garantiert über dieselbe Datentypdefinition verfügen wird wie ihr Primärschlüssel. Der Nachteil ist, daß Sie den Datentyp der Spalte mit Fremdschlüssel nicht durch Überprüfen der create table-Anweisung bestimmen können.

## Vereinbaren eines Fremdschlüssels nach Erstellen der Tabelle

Alternativ zur Deklaration eines Fremdschlüssels beim Erstellen einer Tabelle haben Sie die Möglichkeit, mit der `alter table`-Anweisung einen Fremdschlüssel für eine bereits vorhandene Tabelle zu deklarieren. Listing 4.19 zeigt ein Beispiel für die Deklaration eines Fremdschlüssels mit Hilfe der `alter table`-Anweisung.

***Listing 4.19: Deklarieren eines Fremdschlüssels***

```
SQL> create table Course
2 (Course_ID Varchar2(5),
3 Department_ID Varchar2(20)
4 constraint NN_Course_Department_ID NOT NULL,
5 Title Varchar2(60)
6 constraint NN_Course_Title NOT NULL,
7 Description Varchar2(2000),
8 Units Number,
9 Additional_Fees Number,
10 Constraint PK_Course
11 Primary key (Course_ID, Department_ID));
```

Table created.

```
SQL> alter table Course add constraint FK_Course_Department_ID
2 foreign key (Department_ID) references Department (Department_ID);
```

Table altered.

## Spalten mit Primärschlüsseln und Spalten mit Fremdschlüsseln

Wenn Sie einen Fremdschlüssel definieren, prüft Oracle folgendes:

- Wurde ein Primärschlüssel für die vom Fremdschlüssel referenzierte Tabelle definiert?
- Entspricht die Anzahl der Spalten, die den Fremdschlüssel bilden, der Anzahl von Spalten mit Primärschlüsseln?
- Stimmen Datentyp und Breite der einzelnen Fremdschlüssel mit dem Datentyp und der Breite der einzelnen Spalten mit Primärschlüsseln überein?

Listing 4.20 zeigt, was passiert, wenn der Datentyp der Spalte mit einem Primärschlüssel nicht mit der als Fremdschlüssel geplanten Spalte übereinstimmt. Die `Employee`-Tabelle enthält zum Beispiel die Mitarbeiter einer Firma und `Employee_Dependent` die Untergebenen dieser Mitarbeiter. Der Primärschlüssel der `Employee`-Tabelle lautet `Employee_ID`. Natürlich enthält die `Employee_Dependent`-Tabelle ebenfalls `Employee_ID`. Bitte beachten Sie, daß die Definition von `Employee_ID` in der `Employee`-Tabelle `number(4)` lautet, während die Definition von `Employee_ID` in der `Employee_Dependent`-Tabelle `varchar2(4)` lautet. Daher läßt es Oracle nicht zu, den Fremdschlüssel für `Employee_ID` in der Tabelle `Employee_Dependent` zu definieren, der auf `Employee_ID` in der `Employee`-Tabelle verweist.

***Listing 4.20: Ein Fremdschlüssel kann nicht erzeugt werden, da die Datentypen der Spalten nicht übereinstimmen.***

```
SQL> describe Employee
```

```
Name Null? Type
```

```
-----
EMPLOYEE_ID NOT NULL NUMBER(4)
LAST_NAME NOT NULL VARCHAR2(30)
FIRST_NAME NOT NULL VARCHAR2(20)
MIDDLE_INITIAL CHAR(1)
```

```
HIRE_DATE NOT NULL DATE
TERMINATION_DATE DATE
DATE_OF_BIRTH DATE
MONTHLY_SALARY NUMBER(5)
MANAGER NOT NULL CHAR(1)
USERNAME VARCHAR2(31)
```

```
SQL> describe Employee_Dependent
```

```
Name Null? Type
```

```
-----
EMPLOYEE_ID NOT NULL VARCHAR2(4)
LAST_NAME VARCHAR2(30)
FIRST_NAME VARCHAR2(20)
MIDDLE_INITIAL CHAR(1)
RELATIONSHIP VARCHAR2(30)
```

```
SQL> alter table Employee_Dependent add Constraint Employee_Dependent_FK1
2 Foreign Key (Employee_ID) references Employee;
Foreign Key (Employee_ID) references Employee
*
```

```
ERROR at line 2:
```

```
ORA-02256: number, type and size of referencing columns must match
referenced columns
```

## Deaktivieren und Aktivieren der Integritätsregeln für Schlüsselwerte

Wie bereits zuvor erwähnt, setzen die Integritätsregeln der Primär- und Fremdschlüssel zwei entscheidende Aspekte des relationalen Modells durch: Datenintegrität und referentielle Integrität. Allerdings sollten Sie diese Integritätsregeln für mindestens zwei Aufgaben deaktivieren: für den Datenbankentwurf und die Migration der alten Organisationsdaten.

## Löschen eines Primärschlüssels

Während des Datenbankentwurfs kann es notwendig sein, den Primärschlüssel einer Tabelle zu löschen.



*Zu diesem Zweck verwenden Sie die alter table-Anweisung mit folgender Syntax:*

---

```
ALTER TABLE table-name DROP PRIMARY KEY;
```

Die Variable wird wie folgt definiert:

*table-name* ist die mit dem Primärschlüssel verbundene Tabelle.

Zur Veranschaulichung nehmen wir an, daß Sie eine Tabelle für die Speicherung der Verbindlichkeitsdaten (AP)

erstellen und als Primärschlüssel `Vendor_Invoice_Number` (Rechnungsnummer des Verkäufers) verwenden, wie in Listing 4.21 dargestellt. Nach Validierung einiger Daten und Überlegungen zu dieser Definition erkennen Sie, daß es leicht passieren könnte, daß zwei unterschiedliche Verkäufer dieselbe Rechnungsnummer verwenden. Aus diesem Grund löschen Sie den Primärschlüssel für diese Tabelle.

***Listing 4.21: Löschen eines Primärschlüssels.***

```
SQL> create table AP_Header (  
2 Bill_Number NUMBER(4) NOT NULL,  
3 Vendor_Invoice_Number VARCHAR2(10),  
4 Vendor_ID VARCHAR2(6) NOT NULL,  
5 Date_Received DATE NOT NULL,  
6 Bill_Status VARCHAR2(5),  
7 primary key (Vendor_Invoice_Number));  
Table created.
```

```
SQL> alter table AP_Header drop primary key;  
table altered.
```

Was geschieht, wenn Sie bereits einen Fremdschlüssel in der `AP_Detail`-Tabelle deklariert haben, der auf den Primärschlüssel von `AP_Header` verweist? In diesem Fall gestattet Oracle Ihnen nicht, den Primärschlüssel von `AP_Header` zu löschen. Listing 4.22 verdeutlicht dies.

***Listing 4.22: Das Löschen eines Primärschlüssels ist nicht möglich.***

```
SQL> create table AP_Detail (  
2 Bill_Number NUMBER(4) NOT NULL,  
3 Vendor_Invoice_Number VARCHAR2(10) NOT NULL,  
4 Item_Number NUMBER(3) NOT NULL,  
5 Billed_Amount NUMBER(8,2) NOT NULL,  
6 Approved_Amount NUMBER(8,2),  
7 Paid_Amount NUMBER(8,2),  
8 Constraint AP_Detail_FK Foreign Key (Vendor_Invoice_Number)  
9 References AP_Header);  
Table created.
```

```
SQL> alter table AP_Header drop primary key;  
alter table AP_Header drop primary key  
*  
ERROR at line 1:  
ORA-02273: this unique/primary key is referenced by some foreign keys
```

Es gibt eine Option zu der `drop primary key`-Klausel; Sie können das Schlüsselwort `cascade` verwenden. Wenden Sie diese Funktion mit Vorsicht an! `cascade` löscht den Primärschlüssel sowie alle sich darauf beziehenden Fremdschlüssel. Listing 4.23 zeigt, wie der Primärschlüssel mit Hilfe der `cascade`-Klausel erfolgreich gelöscht wird.

***Listing 4.23: Löschen eines Primärschlüssels mit der cascade-Klausel***

```
SQL> alter table AP_Header drop primary key cascade;  
Table altered.
```

## Löschen eines Fremdschlüssels

Während der Phase des Datenbankentwurfs könnten Sie feststellen, daß Sie irrtümlich eine Spalte als Fremdschlüssel definiert haben. Das Löschen eines Fremdschlüssels funktioniert etwas anders als das Löschen eines Primärschlüssels.



*Da eine Tabelle mehr als einen Fremdschlüssel enthalten kann, verlangt die `alter table`-Anweisung die Angabe des Namens der mit dem Fremdschlüssel verbundenen Integritätsregel, und zwar unter Verwendung der folgenden Syntax:*

```
ALTER TABLE table-name DROP CONSTRAINT constraint-name;
```

Die Definition für diese Variablen lautet wie folgt:

*table-name* steht für die mit dem Primärschlüssel verbundene Tabelle.

*constraint-name* ist die mit dem Fremdschlüssel verbundene Integritätsregel.

Nehmen wir zum Beispiel an, daß die `AP_Header`-Tabelle über eine zusätzliche Spalte namens `Vendor_Status` (Verkäufer\_Status) verfügt, welche zufällig denselben Datentyp und dieselbe Breite aufweist wie `Vendor_ID` (Verkäufer\_Bezeichnung). Wie in Listing 4.24 dargestellt, erstellen Sie irrtümlicherweise einen Fremdschlüssel für `Vendor_Status`, welcher auf den Primärschlüssel der `Vendor`-Tabelle verweist. Bei dem Versuch, einen Wert in die Spalte `Vendor_Status` einzufügen, merken Sie Ihren Fehler schnell und löschen den der Spalte zugeordneten Fremdschlüssel.

**Listing 4.24: Löschen eines Fremdschlüssels.**

```
SQL> alter table AP_Header add constraint AP_Header_Vendor_Status_FK  
2 foreign key (Vendor_Status) references Vendor;  
Table altered.
```

```
SQL> alter table AP_Header drop constraint AP_Header_Vendor_Status_FK;  
Table altered.
```

## Deklaration von Unique-Integritätsregeln

Zusätzlich zu Primär- und Fremdschlüsseln können Sie bei Oracle angeben, daß eine Spalte eindeutige Werte haben soll. Eine Integritätsregel für Eindeutigkeit ist kein Ersatz für die Integritätsregel eines Primärschlüssels. Zum Beispiel enthält die `Patient`-Tabelle eine Liste der Patienten eines Krankenhauses. Jedem Patienten wird eine `Patient ID` (Patientenbezeichnung) zugeordnet, die als Primärschlüssel verwendet wird. Jedoch verfügt jeder Patient auch über eine eindeutige Sozialversicherungsnummer (`Social_Security_Number`). Listing 4.25 zeigt ein Beispiel, wie eine eindeutige Sozialversicherungsnummer mit Hilfe einer Integritätsregel für Eindeutigkeit durchgesetzt wird.

**Listing 4.25: Erstellung einer Integritätsregel für Eindeutigkeit.**

```
SQL> create table Patient (  
2 Patient_ID varchar2(6) primary key,
```

```
3 Last_Name varchar2(30) not null,  
4 First_Name varchar2(20) not null,  
5 Middle_Name varchar2(20),  
6 Social_Security_Number varchar2(9) unique,  
7 Insurance_Carrier_Code varchar2(4));  
Table created.
```

```
SQL> insert into Patient  
2 (Patient_ID, Last_Name, First_Name)  
3 values  
4 ('A901', 'NORTON', 'ED');  
1 row created.
```

```
SQL> insert into Patient  
2 (Patient_ID, Last_Name, First_Name, Social_Security_Number)  
3 values  
4 ('A902', 'KRAMDEN', 'RALPH', '123456789');  
1 row created.
```

```
SQL> insert into Patient  
2 (Patient_ID, Last_Name, First_Name, Social_Security_Number)  
3 values  
4 ('A903', 'NORTON', 'TRIXIE', '123456789');  
insert into Patient  
*  
ERROR at line 1:  
ORA-00001: unique constraint (TYO.UQ_PATIENT_PATIENT_ID) violated
```

## Unterschiede zwischen Primary Key- und Unique-Integritätsregeln

Zwei Unterschiede zwischen den Integritätsregeln für Primärschlüssel und Eindeutigkeit sind besonders zu erwähnen. Erstens kann eine Tabelle nur einen Primärschlüssel, aber viele Integritätsregeln für Eindeutigkeit haben. Zweitens werden die den Primärschlüssel bildenden Spalten bei der Definition des Primärschlüssels automatisch erforderlich. Bei der Deklaration einer Integritätsregel für Eindeutigkeit wird die Eigenschaft der Spalten, die diese Regel bilden, nicht automatisch als erforderlich definiert; Sie müssen der Spalte außerdem den Wert not null zuweisen.

## Tabellenindizes

Bei der Betrachtung der Integritätsregeln für Primär- und Fremdschlüssel muß man auch Indizes erwähnen - diese Themen sind eng miteinander verwandt. Dieser Abschnitt erläutert, was Tabellenindizes sind und wie Sie sowohl vom Anwendungsentwickler als auch von Oracle eingesetzt werden.



*Bei einem Tabellenindex handelt es sich um ein Oracle-Objekt, das die in einer oder mehreren Tabellenspalten vorkommenden Werte enthält.*

Oracle stellt zwei Arten von Tabellenindizes bereit: eindeutige und nicht eindeutige. Eindeutige Indizes setzen den Primärschlüssel und die Integritätsregeln für Eindeutigkeit durch. Nicht eindeutige Indizes können die Abfrageleistung verbessern. Beide Indexarten werden intern über die B\*-Baumstruktur implementiert. Eine B\*-Baumstruktur wird grafisch als symmetrischer, invertierter Baum dargestellt, an dem jedes Blatt einen Indexwert darstellt. Für den Entwurf einer Anwendungsdatenbank ist es wichtig, folgende Begriffe zu verstehen:

- Eine order by-Klausel in einer select-Anweisung kann jede Spalte einer Tabelle referenzieren - unabhängig davon, ob die Tabelle über einen auf dieser Spalte basierenden Index verfügt oder nicht.
- Ein Index kann sich nicht aus mehr als 16 Spalten zusammensetzen.
- Ein Index speichert keine null-Werte.
- SQL verfügt über keine Anweisung, mit der Sie den Inhalt eines Indexes überprüfen können.
- Ein Index kann nur für eine Tabelle, jedoch nicht für eine Datensicht erstellt werden.

### Was Sie tun sollen

Erstellen Sie einen Index für solche Spalten, deren Werte weit verteilt sind, wie z.B. ZIP-Codes.

Erstellen Sie einen Spaltenindex so, daß die indizierten Spalten maximal 20 Prozent der Tabellendaten enthalten.

Erstellen Sie keinen eindeutigen Index statt der Deklaration eines Primärschlüssels.

Übertreiben Sie es mit der Indexerstellung nicht. Sie können zwar die Datenabfrage verbessern, aber die Gesamtleistung der Datenbank negativ beeinflussen.

Erstellen Sie keinen Index für eine Spalte, deren Werte wenig verteilt sind, wie z.B. Geschlecht (prüfen Sie mit Ihrem DBA die Möglichkeiten der Verwendung eines Bitmap-Indexes für solche Spalten).

Indizieren Sie keine kleinen Tabellen.

### Was nicht

## Erstellen eines Index



*Es folgt die bei der Erstellung eines Index zu verwendende Grundsyntax:*

---

```
CREATE [UNIQUE] INDEX Indexname  
ON table-name (column1, ... columnN);
```



Die Definition der Variablen lautet wie folgt:

*index-name* stellt den dem Index (entsprechend der Namensbeschränkungen für Oracle-Datenbankobjekte) zuzuweisenden Namen dar.

*table-name* ist die Tabelle, für die der Index erstellt wird.

*column1* bis *columnN* sind die bei der Indexerstellung zu verwendenden Spalten.

Bitte beachten Sie, daß das Schlüsselwort `unique` optional ist. Wenn Sie `unique` nicht eingeben, wird ein nicht eindeutiger Index erstellt. Mit anderen Worten, der nicht eindeutige Index schränkt die Werte einer Spaltengruppe in keiner Weise ein. Wenn Sie das Schlüsselwort `unique` angeben, verhindert der Index, daß eine Gruppe von Spaltenwerten doppelt in der Tabelle gespeichert wird.

Listing 4.26 zeigt ein Beispiel. Der Primärschlüssel der Student-Tabelle lautet `Student_ID`. Allerdings fragen Sie die Student-Tabelle häufig nach dem Nachnamen eines Studenten ab. Um die Leistung solcher Abfragen zu verbessern, erstellen Sie einen nicht eindeutigen Index.

**Listing 4.26: Erstellung eines nicht eindeutigen Index.**

```
SQL> create index Student_Last_Name on  
2 Student (Last_Name);
```

```
Index created.
```

## Warum man keine unique-Indizes erstellen sollte

Obwohl es sich bei `create unique index` um eine zulässige Oracle-Anweisung handelt, sollten Sie diese nicht verwenden; deklarieren Sie statt dessen `primary key`- und `unique`-Regeln. Dieses sind die beiden Hauptgründe für diesen Ratschlag:

- Wenn Sie `primary key`- und `unique`-Regeln deklarieren, erstellt Oracle automatisch den entsprechenden eindeutigen Index, um diese Integritätsregeln durchzusetzen.
- Sie wären nicht in der Lage, eine `foreign key`-Regel zu deklarieren, wenn Sie nicht auch eine entsprechende `primary key`-Regel deklarieren.

## Integritätsregeln ansehen

Wenn Sie eine Spalte als `not null` deklarieren, behandelt Oracle die Eigenschaft »erforderlich« als Integritätsregel. Diese Regel ist die einzige, die Sie mit dem `SQL*Plus`-Befehl `describe` betrachten können. Listing 4.27 zeigt ein Beispiel. Wenn Sie die Instructor-Tabelle in `SQL*Plus` beschreiben, gibt Oracle an, daß die Instructor-Tabelle über drei erforderliche Spalten verfügt. Diese Anzahl ist etwas irreführend, da `Instructor_ID` als Primärschlüssel der Tabelle definiert wurde - dessen Wert ist automatisch `not null`.

**Listing 4.27: Betrachtung von not null-Regeln mit der describe-Anweisung von SQL\*Plus.**

```
SQL> describe Instructor  
Name Null? Type
```

-----

```
INSTRUCTOR_ID NOT NULL VARCHAR2(20)
DEPARTMENT_ID NOT NULL VARCHAR2(20)
LAST_NAME NOT NULL VARCHAR2(25)
FIRST_NAME VARCHAR2(25)
MI VARCHAR2(1)
POSITION VARCHAR2(25)
TELEPHONE VARCHAR2(10)
FAX VARCHAR2(10)
EMAIL VARCHAR2(100)
```

Sie können das Oracle-Data Dictionary namens `user_constraints` verwenden, um alle mit einer Tabelle verbundenen Integritätsregeln zu betrachten. Die von `user_constraints` zurückgegebenen Spalten enthalten

- `constraint_name`
- `constraint_type`
- `search_condition`

Listing 4.28 zeigt die Ergebnisse einer Abfrage von `user_constraints` nach allen mit der Instructor-Tabelle verbundenen Integritätsregeln: zwei not null-Regeln, eine check-Regel in der Spalte `Position`, eine Regel für den Primärschlüssel, die durch einen Wert `p` für `Constraint_Type` angegeben wird, und einen Fremdschlüssel zu Department ID.



*Sie werden feststellen, daß ich bei Tabellen- und Spaltennamen nur den Anfangsbuchstaben groß schreibe, während ich im Code den gesamten Namen groß schreibe. In Listing 4.28 muß zum Beispiel das Wort `instructor` in Zeile 4 vollständig in Großbuchstaben eingegeben werden, andernfalls erfolgt keine Ausgabe durch das Data Dictionary. Das liegt daran, daß die Werte in doppelten Anführungszeichen stets in Großbuchstaben zu schreiben sind.*

---

**Listing 4.28: Betrachtung von Regeln für die Instructor-Tabelle.**

```
SQL> select constraint_name, constraint_type, search_condition
2 from user_constraints
3 where
4 table_name = 'INSTRUCTOR';
```

```
CONSTRAINT_NAME C SEARCH_CONDITION
```

```
-----
NN_INSTRUCTOR_DEPT_ID C DEPARTMENT_ID IS NOT NULL
NN_INSTRUCTOR_LAST_NAME C LAST_NAME IS NOT NULL
CK_INSTRUCTOR_POSITION C Position in ('ASSISTANT PROFES
SOR', 'ASSOCIATE PROFESSOR',
'PROFESSOR
')
```

```
PK_INSTRUCTOR P
FK_INSTRUCTOR_DEPARTMENT_ID R
```

## Einschränken von Werten mit einer Check-Regel für Spalten

Bei der check-Regel handelt es sich um eine Integritätsregel auf Spaltenebene, die mindestens zwei Zwecken dient.

- Einige Spalten - häufig numerischer Natur - verfügen über keine Einzelwerte, die mit einem Fremdschlüssel referenziert werden können. Dennoch müssen Sie den Wert einer solchen Spalte auf einen bestimmten Bereich beschränken.
- Der zulässige Bereich einer Spalte ist nicht konstant; er muß beim Einfügen oder bei der Aktualisierung des Spaltenwerts berechnet werden.



Die check-Regel wird in einer *create table*- oder *alter table*-Anweisung unter Verwendung der folgenden Syntax deklariert:

---

```
column-name datatype [CONSTRAINT constraint-name] [CHECK (condition)]
```

Die Definition der Variablen lautet wie folgt:

*column-name* steht für den Spaltennamen.

*datatype* ist der Datentyp, die Breite und Skalierung der Spalte.

*constraint-name* stellt den Namen der Integritätsregel dar und unterliegt den Einschränkungen für die Benennung von Oracle-Datenbankobjekten.

*condition* ist eine zulässige SQL-Bedingung in Oracle, die einen booleschen Wert zurückgibt.

Um dieses Konzept zu verdeutlichen, zeigt Listing 4.29, wie Sie eine Tabelle erstellen können, die in einer Krankenhausdatenbank für die Speicherung von Patientendaten eingesetzt wird. Eine der Spalten in dieser Tabelle enthält die Körpertemperatur des Patienten in Grad Fahrenheit (*Body\_Temp\_Deg\_F*). Sie sollten die möglichen Werte dieser Spalten einschränken, indem Sie diese als *number(4,1)* definieren. Allerdings akzeptiert diese Spalte nach wie vor Zahlen von 0,0 bis 999,9 - einschließlich einiger für die Körpertemperatur offensichtlich unsinnigen Werte. Sie können eine check-Regel verwenden, um den Wertebereich auf 60,0 (für Patienten, die an Unterkühlung leiden) bis 110,0 einzuschränken.

**Listing 4.29: Erstellung einer check-Regel.**

```
SQL> create table Patient (  
2 Patient_ID varchar2(6) primary key,  
3 Body_Temp_Deg_F number(4,1) constraint Patient_Body_Temp  
4 Check (Body_Temp_Deg_F >= 60.0 and  
5 Body_Temp_Deg_F <= 110.0));  
Table created.
```

```
SQL> insert into Patient  
2 (Patient_ID, Body_Temp_Deg_F)  
3 values
```

```
4 ('A1001', 98.6);
```

```
1 row created.
```

```
SQL> insert into Patient
```

```
2 (Patient_ID, Body_Temp_Deg_F)
```

```
3 values
```

```
4 ('Q7777', 111.2);
```

```
('Q7777', 111.2)
```

```
*
```

```
ERROR at line 4:
```

```
ORA-02290: check constraint (TYO.CK_PATIENT_BODY_TEMP) violated
```

Sie können die in Oracle integrierten SQL-Funktionen in einer check-Regel verwenden. Als Beispiel zeigt Listing 4.30 die Definition einer check-Regel, welche überprüft, ob der Versicherungsstatus (Insurance\_Status) eines Patienten Ja (y) oder Nein (n) lautet.

*Listing 4.30: Verwendung des in-Operators in einer check-Regel.*

```
E SQL> create table Patient (
```

```
2 Patient_ID varchar2(6) primary key,
```

```
3 Body_Temp_Deg_F number(4,1) constraint Patient_Body_Temp
```

```
4 Check (Body_Temp_Deg_F >= 60.0 and
```

```
5 Body_Temp_Deg_F <= 110.0),
```

```
6 Insurance_Status Char(1) constraint Patient_Insurance_Status
```

```
7 Check (Insurance_Status in ('Y','y','N','n')));
```

```
Table created.
```

```
SQL> insert into Patient
```

```
2 (Patient_ID, Insurance_Status)
```

```
3 values
```

```
4 ('R4321','Y');
```

```
1 row created.
```

```
SQL> insert into Patient
```

```
2 (Patient_ID, Insurance_Status)
```

```
3 values
```

```
4 ('U3030','U');
```

```
('U3030','U')
```

```
*
```

```
ERROR at line 4:
```

```
ORA-02290: check constraint (TYO.CK_PATIENT_INSURANCE_STATUS) violated
```



Wie Sie sehen, wird die check-Regel verletzt, wenn Sie versuchen, den Wert u unter Insurance\_Status (Versicherungsstatus) einzugeben.

---

**Zu einer Spalte kann es mehr als eine Check-Regel geben**

Oracle schränkt die Anzahl der für eine Spalte oder eine Tabelle zu definierenden check-Regeln nicht ein. Kommen wir auf das Beispiel des Flugle-Colleges zurück. Listing 4.31 zeigt, wie zu einer einzelnen Spalte - Amount\_Approved (genehmigter Betrag) - zwei check-Regeln definiert werden.

**Listing 4.31: Eine einzelne Spalte mit mehr als einer check-Regel.**

```
SQL> create table Loan_Application (  
2  Loan_Application_No number(6) primary key,  
3  Borrower_Last_Name varchar2(30) not null,  
4  Borrower_First_Name varchar2(20) not null,  
5  Borrower_Middle_Name varchar2(20),  
6  Amount_Requested number(9,2) not null,  
7  Amount_Approved number(9,2)  
8  constraint Amount_Approved_Limit  
9  check (Amount_Approved <= 1000000)  
10 constraint Amount_Approved_Interval  
11 check (mod(Amount_Approved,1000)=0)  
12 );  
Table created.  
  
SQL> insert into Loan_Application  
2 (Loan_Application_No, Borrower_Last_Name, Borrower_First_Name,  
3 Amount_Requested, Amount_Approved)  
4 values  
5 (2001, 'RUBRIK', 'STANLEY', 1000000, 999950);  
insert into Loan_Application  
*  
ERROR at line 1:  
ORA-02290: check constraint (TYO.CK_LOAN_APPLICATION1) violated  
  
SQL> insert into Loan_Application  
2 (Loan_Application_No, Borrower_Last_Name, Borrower_First_Name,  
3 Amount_Requested, Amount_Approved)  
4 values  
5 (2001, 'RUBRIK', 'STANLEY', 1000000, 999000);  
1 row created.  
  
SQL> insert into Loan_Application  
2 (Loan_Application_No, Borrower_Last_Name, Borrower_First_Name,  
3 Amount_Requested, Amount_Approved)  
4 values  
5 (2001, 'RUBRIK', 'STANLEY', 1000000, 1001000);  
insert into Loan_Application  
*  
ERROR at line 1:  
ORA-02290: check constraint (TYO.CK_LOAN_APPLICATION2) violated
```



In dem Beispiel aus Listing 4.31 werden für die Tabelle *Loan\_Application* (Darlehensantrag) zwei Integritätsregeln definiert. Die erste Regel (Zeilen 8 und 9) begrenzt den Bewilligungsbetrag auf eine Million Dollar, während die zweite Regel (Zeilen 10 und 11) sicherstellt, daß der Bewilligungsbetrag in Schritten von tausend Dollar

vorliegt. Sie können beide Regeln auf einfache Weise in einer einzigen Integritätsregel kombinieren. Sie sollten allerdings separate Integritätsregeln definieren, wenn Sie glauben, daß es erforderlich werden könnte, eine einzelne Regel zu deaktivieren, während andere Regeln weiterhin aktiviert bleiben sollen.

## In einer Check-Regel auf andere Spalten verweisen

Eine der Einschränkungen einer check-Regel für eine Spalte ist, daß sie nicht auf andere Spalten derselben Tabelle verweisen kann. Nehmen wir an, Sie sollen eine Tabelle definieren, in der Informationen über Darlehensanträge gespeichert werden sollen. In dieser Tabelle enthält Amount\_Requested den vom Darlehensnehmer beantragten Betrag; bei Amount\_Approved handelt es sich um den vom Darlehensausschuß genehmigten Betrag. Der Darlehensgeber genehmigt niemals einen Betrag, der die beantragte Summe übersteigt.

Listing 4.32 verdeutlicht diese Regel. Die check-Regel einer Spalte kann nicht auf andere Spalten verweisen. Sie können jedoch eine Tabellenregel verwenden, um eine andere Spalte in einer Tabelle zu referenzieren. Durch Einfügen eines Kommas nach der Definition von Amount\_Approved, wird aus der Spalten- eine Tabellenregel.

**Listing 4.32: Verwendung einer für Tabellen geltenden check-Regel statt einer für eine Spalte geltenden check-Regel.**

```
SQL> create table Loan_Application (  
2 Loan_Application_No number(6) primary key,  
3 Borrower_Last_Name varchar2(30) not null,  
4 Borrower_First_Name varchar2(20) not null,  
5 Borrower_Middle_Name varchar2(20),  
6 Amount_Requested number(9,2) not null,  
7 Amount_Approved number(9,2)  
8 constraint Amount_Approved_Limit  
9 check (Amount_Approved <= Amount_Requested)  
10 );  
)  
*
```

ERROR at line 10:

ORA-02438: Column check constraint cannot reference other columns

```
SQL> create table Loan_Application (  
2 Loan_Application_No number(6) primary key,  
3 Borrower_Last_Name varchar2(30) not null,  
4 Borrower_First_Name varchar2(20) not null,  
5 Borrower_Middle_Name varchar2(20),  
6 Amount_Requested number(9,2) not null,  
7 Amount_Approved number(9,2),  
8 constraint Amount_Approved_Limit  
9 check (Amount_Approved <= Amount_Requested)  
10 );  
Table created.
```

```
SQL> insert into Loan_Application  
2 (Loan_Application_No, Borrower_Last_Name, Borrower_First_Name,  
3 Amount_Requested, Amount_Approved)  
4 values  
5 (2001, 'CRANDALL', 'JULIE', 300000, 310000);
```

```
insert into Loan_Application
```

```
*
```

```
ERROR at line 1:
```

```
ORA-02290: check constraint (TYO.AMOUNT_APPROVED_LIMIT_) violated
```

```
SQL> insert into Loan_Application
```

```
2 (Loan_Application_No, Borrower_Last_Name, Borrower_First_Name,
```

```
3 Amount_Requested, Amount_Approved)
```

```
4 values
```

```
5 (2001, 'CRANDALL', 'JULIE', 300000, 300000);
```

```
1 row created.
```

## Pseudospalten in einer check-Regel verwenden

Eine check-Regel kann nicht auf Pseudospalten, wie z.B. sysdate, rownum und user verweisen. Wenn Sie eine Firmenregel definieren müssen, die sich auf diese Pseudospalten bezieht, sollten Sie sich bei der Einschränkung von Spaltenwerten eines Datenbanktriggers bedienen. Am Tag 11, »Weitere Programmieretechniken für PL/SQL«, werde ich auf dieses Thema näher eingehen.



*Unter einer Pseudospalte versteht man einen integrierten Wert (oder eine Funktion ohne Argumente), der durch Abfragen einer Tabelle einen bestimmten Teil von Informationen liefert - normalerweise dual. sysdate gibt zum Beispiel stets das aktuelle Datum und die Uhrzeit aus.*

## Allgemeine Beschränkungen für die Änderung einer Tabelle

Während des Datenbankentwurfs experimentiert der Entwickler im allgemeinen mit den Datenbankstrukturen einer Anwendung, um ihre Genauigkeit und Eignung zu überprüfen. Diese Aufgabe kann kaum erledigt werden, ohne die Tabellen mit Beispieldaten zu füllen. Wenn eine Tabelle geändert werden muß - zum Beispiel in Form einer zusätzlichen Spalte -, so ist dies vorteilhafter, als die Tabelle zu löschen und mit einer neuen Definition nochmals zu erstellen. Manchmal kann die erforderliche Änderung jedoch nicht erfolgen, ohne die Beispieldaten zu ändern.

## Ändern des Primärschlüssels

Es ist oft schwierig, den Primärschlüssel einer Tabelle zu ändern, ohne den restlichen Datenbankentwurf zu beeinflussen. Der Grund dafür ist einfach: Tabellen werden normalerweise durch die Deklaration von Fremdschlüsseln zueinander in Bezug gesetzt. Ein Fremdschlüssel hängt vom Vorhandensein eines Primärschlüssels in einer anderen Tabelle ab. Daher kann sich die Änderung des Primärschlüssels auf die gesamte Datenbank auswirken. Oracle verhindert dies jedoch.

Das Ändern eines Primärschlüssels erfolgt in zwei Schritten: Löschen und Neuerstellen des Primärschlüssels. Dies wird in Listing 4.33 verdeutlicht. Nehmen wir an, Sie hätten die Course-Tabelle mit einem Primärschlüssel erstellt, der lediglich aus Course ID besteht. Nach einigen Überlegungen merken Sie, daß verschiedene Fachabteilungen eventuell

dieselben Kursnummern verwenden könnten. Daher ändern Sie den Primärschlüssel so, daß er sowohl die Course ID als auch die Department ID enthält.

**Listing 4.33: Änderung eines Primärschlüssels.**

```
SQL> alter table Course add  
2 (primary key (Course_ID));
```

Table altered.

```
SQL> alter table Course drop primary key;
```

Table altered.

```
SQL> alter table Course add  
2 (primary key (Course_ID, Department_ID));
```

Table altered.

## **Skript für die Erstellung der Beispieldatenbank**

Da Sie nun die Grundlagen der Tabellenerstellung kennengelernt haben, ist es an der Zeit, sich mit dem Skript für die Erstellung unserer Beispieldatenbank zu befassen, das in Listing 4.34 dargestellt ist.

Im ersten Abschnitt des Skripts werden alle Tabellen gelöscht. Wenn Sie versuchen, eine von dem Fremdschlüssel einer anderen Tabelle referenzierte Tabelle zu löschen, gibt Oracle einen Fehlercode aus.

Die einzige Ausnahme ist die Verwendung der cascade-Klausel in der drop table-Anweisung. Wenn Sie die cascade-Klausel nicht benutzen, ist es daher wichtig, die Tabellen in der Reihenfolge zu löschen, daß die gelöschte Tabelle nicht von dem Fremdschlüssel einer anderen Tabelle referenziert wird.



Sie finden dieses Skript unter dem Namen `\tyo\flugle.sql` auf der CD-ROM. Zum Ausführen des Skripts können Sie `SQL*Plus` oder `SQL Worksheet` einsetzen. Falls Sie `SQL*Plus` verwenden, geben Sie `@C:\TYO\flugle.sql` ein (sofern Sie die Skripts von der CD-ROM auf Ihrem Laufwerk C im Verzeichnis TYO installiert haben). Falls Sie `SQL Worksheet` benutzen, geben Sie `@C:\TYO\flugle.sql` in die Befehlszeile des unteren Fensters ein.

---

**Listing 4.34: Erstellung der Datenbanktabellen für das Beispiel.**

```
drop table Student_Schedule;
```

```
drop table Student;
```

```
drop table Class;
```

```
drop table Class_Location;
```

```
drop table Course;
```



```
drop table Schedule_Type_Details;

drop table Schedule_Type;

drop table Instructor;

drop table Department;

create table Department
(Department_ID Varchar2(20),
Department_Name Varchar2(25)
constraint NN_Department_Name NOT NULL,
constraint PK_Department
primary key (Department_ID));

create table Student
(Student_ID Varchar2(20),
Last_Name Varchar2(25)
constraint NN_Student_Last_Name NOT NULL,
First_Name Varchar2(25)
constraint NN_Student_First_Name NOT NULL,
MI Varchar2(1),
Year Varchar2(25),
Street_Address Varchar2(25),
City Varchar2(25),
State Varchar2(2),
Zipcode Varchar2(9),
Telephone Varchar2(10),
Fax Varchar2(10),
Email Varchar2(100),
constraint PK_Student
primary key (Student_ID));

create table Course
(Course_ID Varchar2(5),
Department_ID Varchar2(20)
constraint NN_Course_Department_ID NOT NULL,
Title Varchar2(60)
constraint NN_Course_Title NOT NULL,
Description Varchar2(2000),
Units Number,
Additional_Fees Number,
Constraint PK_Course
Primary key (Course_ID, Department_ID),
Constraint FK_Course_Department_ID
Foreign key (Department_ID) references Department (Department_ID));

create table Instructor
(Instructor_ID varchar2(20),
Department_ID varchar2(20)
constraint NN_Instructor_Dept_ID NOT NULL,
Last_Name varchar2(25)
constraint NN_Instructor_Last_Name NOT NULL,
First_Name varchar2(25),
MI varchar2(1),
```

```
Position varchar2(25) constraint CK_Instructor_Position
check (Position in ('ASSISTANT PROFESSOR', 'ASSOCIATE PROFESSOR',
'PROFESSOR')),
Telephone varchar2(10),
Fax varchar2(10),
Email varchar2(100),
constraint PK_Instructor Primary Key (Instructor_ID),
constraint FK_Instructor_Department_ID
Foreign Key (Department_ID) references Department (Department_ID));

create table Schedule_Type
(Schedule_ID varchar2(20),
Schedule_Description varchar2(2000),
constraint PK_Schedule_Type primary key (Schedule_ID));

create table Schedule_Type_Details
(Schedule_ID varchar2(20),
Day number
constraint CK_Schedule_Det_Day check (Day between 1 and 7),
Starting_Time date,
Duration number,
constraint PK_Schedule_Type_Details primary key (Schedule_ID, Day),
constraint FK_Schedule_Det_ID
foreign key (Schedule_ID) references Schedule_Type (Schedule_ID));

create table Class_Location
(Class_Building varchar2(25),
Class_Room varchar2(25),
Seating_Capacity number,
constraint PK_Class_Location primary key (Class_Building, Class_Room));

create table Class
(Class_ID varchar2(20),
Schedule_ID varchar2(20),
Class_Building varchar2(25),
Class_Room varchar2(25),
Course_ID varchar2(5),
Department_ID varchar2(20),
Instructor_ID varchar2(20),
Semester varchar2(6),
School_Year date,
constraint PK_Class primary key (Class_ID),
constraint FK_Class_Location
foreign key (Class_Building, Class_Room) references
Class_Location (Class_Building, Class_Room),
constraint FK_Class_Schedule_ID
foreign key (Schedule_ID) references Schedule_Type (Schedule_ID),
constraint FK_Class_Course_ID
foreign key (Course_ID, Department_ID) references
Course (Course_ID, Department_ID),
constraint FK_Class_Dept_ID
foreign key (Department_ID) references Department (Department_ID),
constraint FK_Class_Instructor_ID
foreign key (Instructor_ID) references Instructor (Instructor_ID));
```

```
create table Student_Schedule
(Student_ID varchar2(20),
Class_ID varchar2(20),
Grade varchar2(2)
constraint CK_Grade check (Grade in ('A', 'A+', 'A-', 'B', 'B+', 'B-',
'C', 'C+', 'C-', 'D', 'D+', 'D-',
'F', 'F+', 'F-'))),
Date_Grade_Assigned date,
constraint PK_Student_Schedule
primary key (Student_ID, Class_ID),
constraint FK_Student_Schedule_Student_ID
foreign key (Student_ID) references Student (Student_ID),
constraint FK_Student_Schedule_Class
foreign key (Class_ID) references Class (Class_ID));
```

## Zusammenfassung

Bei der Implementierung Ihres logischen Entwurfs sollten Sie folgendes berücksichtigen:

- Verwenden Sie die create table-Anweisung mit Hilfe von SQL\*Plus oder SQL Worksheet, um die Anweisungsoptionen besser steuern oder ein Skript verarbeiten zu können.
- Benutzen Sie die alter table-Anweisung, um eine vorhandene Tabelle zu ändern.
- Definieren Sie gegebenenfalls Standardwerte für Spalten.
- Jede Tabelle sollte über einen Primärschlüssel verfügen.
- Verwenden Sie Fremdschlüssel, um die referentielle Integrität durchzusetzen.
- Sie können den Primärschlüssel für eine Tabelle mit Hilfe der create table- oder der alter table-Anweisung definieren.
- Sie können den Primärschlüssel einer Tabelle mit der alter table-Anweisung löschen.
- Sie können eine Integritätsregel mit der alter table-Anweisung deaktivieren oder aktivieren.
- Sie können eine Integritätsregel mit der alter table-Anweisung löschen.
- Ein Tabellenindex kann für bis zu 16 Spalten definiert werden.
- Ein Tabellenindex kann eindeutig oder nicht eindeutig sein.
- Erstellen Sie keine eindeutigen Indizes mit der create unique index-Anweisung; deklarieren Sie statt dessen primary key- und unique-Regeln, und lassen Sie die Indizes von Oracle automatisch erstellen.
- Verwenden Sie eine check-Regel, um den Wert einer Spalte auf einen festen oder berechneten Bereich zu beschränken.

## Wie geht es weiter?

Am Tag 5, »Einführung in SQL«, erfahren Sie mehr über vier sehr wichtige SQL-Anweisungen: select, insert, update und delete. Diese Anweisungen werden zur Abfrage und Bearbeitung von Daten in einer Oracle-Datenbank eingesetzt.

## Fragen und Antworten

Frage:

Wie hoch ist die ideale Anzahl von Indizes für eine Tabelle?

Antwort:

*Die Antwort lautet: Es kommt darauf an. Zumindest wird ein Index automatisch durch die Deklaration des Primärschlüssels erstellt. Darüber hinaus können Sie vielleicht einen nicht eindeutigen Index für die einzelnen Fremdschlüssel erstellen. Sie könnten auch einen nicht eindeutigen Index in alle Spalten einfügen, die häufig Teil einer Abfragebedingung ist und deren Werte weit verteilt sind. Zusätzliche Indizes für Tabellen mit geringer Zeilenanzahl - z.B. einhundert oder weniger - sollten Sie allerdings nur sehr sparsam einsetzen.*

Frage:

Welche Vorteile hat die Deklaration einer *check-Regel* für eine Spalte? Kann die Integritätsregel nicht in einer Anwendung wie Oracle Forms oder Visual Basic durchgesetzt werden?

Antwort:

*Ja, es ist möglich und manchmal auch wünschenswert, eine Integritätsregel auch auf Anwendungsebene durchzusetzen. Sie sollten allerdings die deklarativen Integritätsregeln für eine Tabelle und ihre Spalten als erste »Verteidigungslinie« für die Datenintegrität betrachten. Da in der Client-Server-Umgebung eine große Anzahl von Werkzeugen eingesetzt wird, müssen Sie sicherstellen, daß niemand einer Spalte ungültige Werte zuordnen kann, egal welches Werkzeug benutzt wird.*

Frage:

Wie kann man eine Spalte in einer Tabelle löschen oder umbenennen?

Antwort:

*Oracle gestattet es nicht, eine Spalte direkt umzubenennen oder zu löschen. Wenn Sie eine Spalte umbenennen oder löschen wollen, stehen Ihnen zwei Möglichkeiten zur Verfügung: Sie können die Tabelle löschen und neu erstellen, oder Sie können aus einer Teilmenge von Spalten der vorhandenen Tabelle eine neue Tabelle erstellen. Wenn Sie sich für den letzteren Ansatz entscheiden, denken Sie daran, daß die neue Tabelle außer den **not null-Regeln keine weiteren Integritätsregeln enthält**.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Was ist an dieser Anweisung falsch?

```
CREATE TABLE new_table (  
  first_col number,  
  second_col date  
  third_col number default sysdate);
```

2. Beschreiben Sie eine SQL-Anweisung, welche die folgende Oracle-Fehlermeldung verursachen könnte.

ORA-02266: unique/primary keys in table referenced by enabled foreign keys

3. Worin besteht der Unterschied zwischen den check-Regeln für eine Spalte und für eine Tabelle?

## Übungen

Die Instructor-Tabelle besitzt eine Spalte namens Position. In dem derzeitigen Tabellenentwurf gibt es eine check-Regel für die Spalte Position, welche den Wert auf assistant professor, associate professor und full professor begrenzt. Ändern Sie den Datenbankentwurf so, daß eine zusätzliche Tabelle, INSTRUCTOR\_POSITION, verwendet wird, um zulässige Werte für die Position des Kursleiters anzugeben.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖ Kapitel ❖ Top | Inhalt | Index **SAMS** Kapitel ❖

# 5

# Einführung in SQL

[Abrufen und Ändern von Daten](#)

[Die SQL-Grammatik](#)

[Syntax der select-Anweisung](#)

[Festlegen von Kriterien in der where-Klausel](#)

[Sortieren von Daten in der order by-Klausel](#)

[Zählen von Zeilen in einer Tabelle](#)

[Ermitteln von Zeilen mit Spaltenwert null](#)

[Suche nach Zeilen mit Hilfe des like-Operators](#)

[Suche nach Zeilen mit dem between-Operator](#)

[Der in-Operator](#)

[Mit einem Alias auf Spalten verweisen](#)

[Eine Unterabfrage verwenden](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

Für die Entwicklung einer Oracle-Datenbankanwendung ist es wichtig, daß Sie sich ausreichende Kenntnisse in der strukturierten Abfragesprache SQL aneignen. SQL ist eine leistungsfähige Sprache, die sich von traditionellen Sprachen der dritten Generation (3GLs), wie z.B. C und Pascal, unterscheidet:

- Bei SQL handelt es sich um eine nichtprozedurale Sprache. Sie können SQL einsetzen, um Oracle mitzuteilen, *welche* Daten abgerufen oder geändert werden sollen, ohne dabei anzugeben, *wie* das zu geschehen hat.
- SQL enthält keine Programmierkonstrukte für die Ablaufsteuerung, Funktionsdefinitionen, Do-Schleifen oder if-then-else-Anweisungen. Wie Sie am Tag 9, »Programmierung einer Oracle-Datenbank mit PL/SQL«, sehen werden, stellt Oracle allerdings mit einem Produkt namens PL/SQL prozedurale Spracherweiterungen für SQL bereit.
- SQL verfügt über eine feste Anzahl Datentypen; Sie können keine neuen Datentypen definieren. Im Gegensatz zu modernen Programmiersprachen, mit denen Sie für bestimmte Zwecke einen Datentyp definieren können, zwingt Oracle Sie, bei der Erstellung oder Änderung einer Spalte den Datentyp aus einer vordefinierten Liste auszuwählen. Sie können aber davon ausgehen, daß Oracle in der Zukunft eine objektorientierte Datenbank herausbringen wird, bei der Sie anwendungsspezifische Datentypen definieren können.

Eine Ähnlichkeit zwischen SQL und einer traditionellen Programmiersprache besteht darin, daß Sie im allgemeinen bei beiden mehrere Möglichkeiten haben, ein und dieselbe Aufgabe zu erledigen - besonders im Bereich des Datenabrufs.

Unterschiedliche SQL-Anweisungen können zu demselben Ergebnis führen (sich allerdings in bezug auf Leistungsfähigkeit und Verständlichkeit unterscheiden).

## Abrufen und Ändern von Daten

Auf höchster Ebene können SQL-Anweisungen in die drei folgenden großen Kategorien eingeteilt werden:

- Data Manipulation Language (DML, Datenmanipulationssprache), mit der Daten abgerufen oder geändert werden
- Data Definition Language (DDL, Datendefinitionssprache), mit der die Datenstruktur definiert wird
- Data Control Language (DCL, Datenkontrollsprache), mit der die den Datenbankbenutzern erteilten Berechtigungen definiert werden

Die Kategorie DML enthält vier Grundanweisungen:

- select, mit der Zeilen einer Tabelle abgerufen werden
- insert, mit der Zeilen in eine Tabelle eingefügt werden
- update, mit der vorhandene Zeilen einer Tabelle geändert werden
- delete, mit der Zeilen aus einer Tabelle entfernt werden

Diese Anweisungen werden meist von Anwendungsentwicklern verwendet. DDL- und DCL-Anweisungen werden hauptsächlich von Datenbankentwicklern und -administratoren bei der Erstellung der von einer Anwendung benutzten Datenbankstrukturen eingesetzt.

## Die SQL-Grammatik

Es folgen einige grammatikalische Erfordernisse, die Sie bei der Arbeit mit SQL berücksichtigen müssen.

- Jede SQL-Anweisung endet mit einem Semikolon.
- Eine SQL-Anweisung kann in eine Zeile eingegeben oder der Deutlichkeit halber auf mehrere Zeilen aufgeteilt werden. In den meisten Beispielen dieses Buchs sind die Anweisungen in gut lesbare Abschnitte unterteilt.
- SQL ignoriert Groß- und Kleinschreibung; Sie können Groß- und Kleinbuchstaben mischen, wenn Sie auf SQL-Schlüsselwörter (wie z.B. select und insert), Tabellen- und Spaltennamen verweisen.

Bei dem Inhalt einer Spalte spielt die Groß- und Kleinschreibung jedoch eine Rolle; wenn Sie alle Kunden abfragen, deren Nachname mit *a* beginnt, und alle Namen in Großbuchstaben gespeichert sind, werden überhaupt keine Zeilen abgerufen.



Die meisten Beispiele in diesem Kapitel wurden mit SQL\*Plus erstellt. Natürlich können Sie auch jedes andere von Ihnen bevorzugte Werkzeug für die Eingabe von SQL-Anweisungen verwenden - z.B. SQL\*Plus, SQL Worksheet oder Discoverer/2000. Der Übersichtlichkeit halber habe ich die sich wiederholenden, von SQL erzeugten Spaltenüberschriften weggelassen.

---

## Syntax der select-Anweisung

Von den vier DML-Anweisungen wird die select-Anweisung in einer realen Anwendung am häufigsten ausgeführt, da Datensätze im allgemeinen häufiger gelesen als geändert werden. Eine select-Anweisung kann auch als Unterabfrage in einer update-, insert- oder delete-Anweisung vorkommen, darauf werden wir aber am Tag 6, »Ändern von Daten mit Hilfe von SQL«, noch eingehen.

Die select-Anweisung ist ein äußerst leistungsfähiges Werkzeug, und ihre Syntax ist aufgrund der zahlreichen Möglichkeiten, Tabellen, Spalten, Funktionen und Operatoren in zulässigen Anweisungen zu kombinieren, kompliziert. Anstatt uns die vollständige Syntax der select-Anweisung anzusehen, beginnen wir in diesem Abschnitt mit einigen grundlegenden Beispielen für eine select-Anweisung.

Eine select-Anweisung enthält mindestens zwei Elemente:

- die select-Liste, d.h. eine Liste der abzurufenden Spalten oder Ausdrücke
- die from-Liste, d.h. die Tabelle, aus der Zeilen abgerufen werden sollen

### Eine einfache select-Anweisung

---



Wenn Sie die Beispiele dieser Lektion durcharbeiten möchten, befolgen Sie bitte die Anweisungen in der Einleitung zu diesem Buch; dort wird die Installation der Beispieldatenbank beschrieben.

---

Eine einfache select-Anweisung - eine Abfrage, mit der nur die Student ID aus der Student-Tabelle abgerufen wird - ist in Listing 5.1 dargestellt.

#### Listing 5.1: Eine erste select-Anweisung.

```
SQL> select Student_ID  
2 from Student;
```

```
STUDENT_ID  
-----  
10231300  
10231301  
10231302  
10231305
```



```
10231306
10231308
10231309
10231322
10231314
10231315
10231317
10231319
10231320
10231325
10231326
10231328
10231329
10231327
10231330
10231316
10231303
10231310
10231318
10231324
10231304
10231312
10231321
10231313
10231311
10231323
10231307
```

31 rows selected.

Wenn Sie sowohl den Bezeichner (Student ID) als auch den Nachnamen (Last\_Name) des Studenten abrufen wollen, führen Sie die Spalten einfach in der gewünschten Reihenfolge auf, wie in Listing 5.2 dargestellt.

***Listing 5.2: Abfragen mehrerer Spalten.***

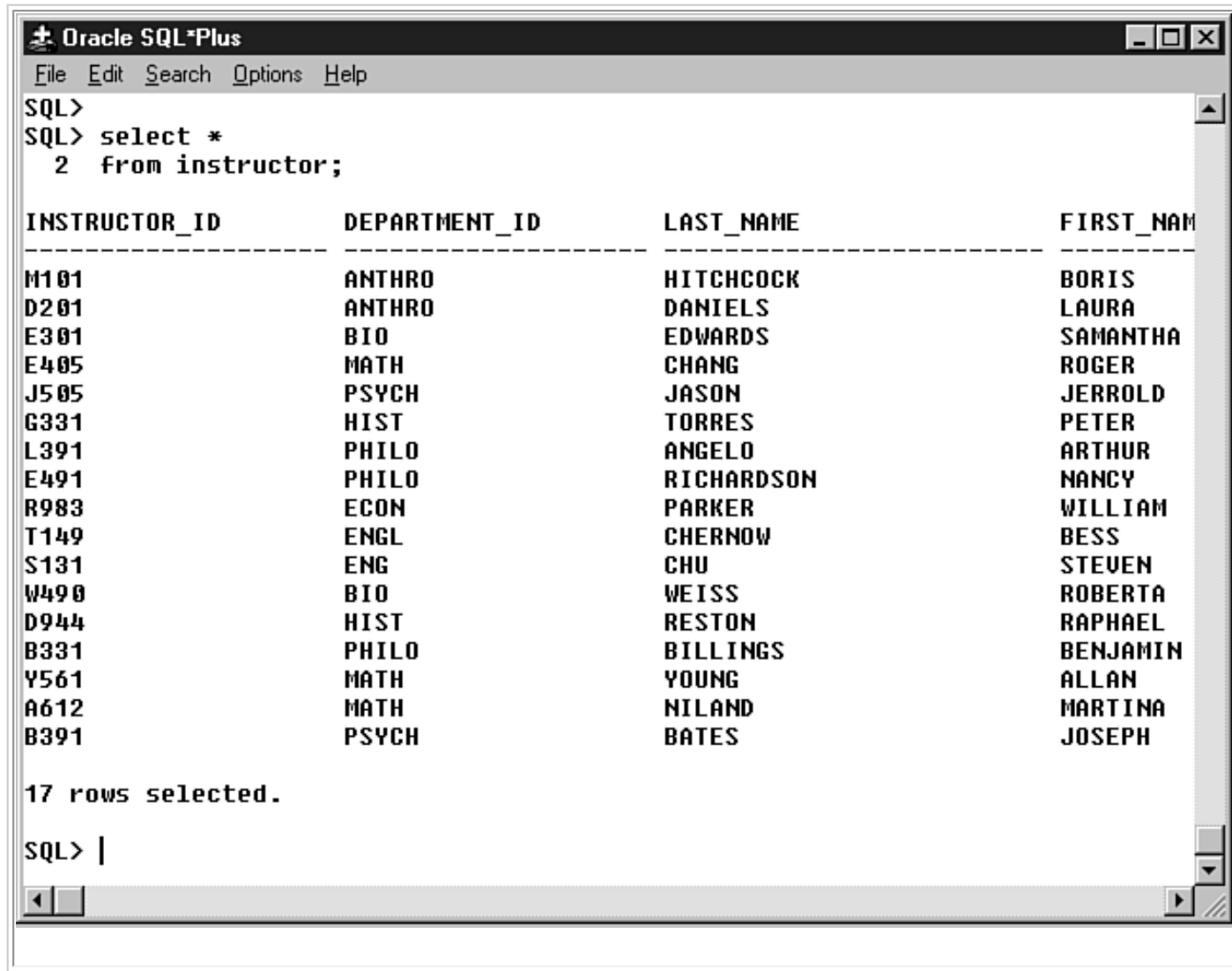
```
SQL> select Student_ID, Last_Name
2 from Student;
```

```
STUDENT_ID  LAST_NAME
-----
10231300    SMYTHE
10231301    HAN
10231302    GORDON
10231305    REYNOLDS
10231306    PARKER
10231308    TANAKA
10231309    COEN
10231322    NEWTON
10231314    PURCELL
10231315    JACKEL
10231317    ROSEN
10231319    MALLARD
10231329    GUSSEY
10231325    MICHAELS
10231326    DEURRE
10231328    WONG
10231329    ABBOT
```

10231327 POSEN  
10231330 CLAUSEN  
10231316 GOMEZ  
10231303 MASSEY  
10231310 FERGUSON  
10231318 BING  
10231324 ANASTATIA  
10231304 CHIN  
10231312 HOLMES  
10231321 PLOWCHARD  
10231313 JACKSON  
10231311 FERNANDEZ  
10231323 PINKWATER  
10231307 MABEISI

31 rows selected.

Wenn Sie alle Spalten einer Tabelle abrufen möchten, können Sie dazu eine SQL-Abkürzung (\*) verwenden, wie in Abbildung 5.1 dargestellt.



The screenshot shows the Oracle SQL\*Plus interface. The title bar reads 'Oracle SQL\*Plus'. The menu bar includes 'File', 'Edit', 'Search', 'Options', and 'Help'. The command prompt shows the following SQL query:

```
SQL> select *  
  2  from instructor;
```

The query results are displayed in a table with four columns: INSTRUCTOR\_ID, DEPARTMENT\_ID, LAST\_NAME, and FIRST\_NAME. There are 17 rows of data. Below the table, it says '17 rows selected.' and the prompt 'SQL> |' is visible.

| INSTRUCTOR_ID | DEPARTMENT_ID | LAST_NAME  | FIRST_NAME |
|---------------|---------------|------------|------------|
| M101          | ANTHRO        | HITCHCOCK  | BORIS      |
| D201          | ANTHRO        | DANIELS    | LAURA      |
| E301          | BIO           | EDWARDS    | SAMANTHA   |
| E405          | MATH          | CHANG      | ROGER      |
| J505          | PSYCH         | JASON      | JERROLD    |
| G331          | HIST          | TORRES     | PETER      |
| L391          | PHILO         | ANGELO     | ARTHUR     |
| E491          | PHILO         | RICHARDSON | NANCY      |
| R983          | ECON          | PARKER     | WILLIAM    |
| T149          | ENGL          | CHERNOW    | BESS       |
| S131          | ENG           | CHU        | STEVEN     |
| W490          | BIO           | WEISS      | ROBERTA    |
| D944          | HIST          | RESTON     | RAPHAEL    |
| B331          | PHILO         | BILLINGS   | BENJAMIN   |
| Y561          | MATH          | YOUNG      | ALLAN      |
| A612          | MATH          | NILAND     | MARTINA    |
| B391          | PSYCH         | BATES      | JOSEPH     |

Abbildung 5.1:  
Auswählen aller Spalten in einer Tabelle mit Hilfe von SQL\*Plus.

### Die select-Liste

Falls die select-Liste mehrere Spalten enthält, müssen diese Spalten durch Kommata getrennt werden. Die select-Liste kann außerdem zulässige Ausdrücke enthalten, die wiederum Spalten enthalten können, aber nicht müssen. Außerdem können Sie eine Spalte in einer select-Liste mehrfach verwenden. Die Abfrage in Listing 5.3 ist vollständig zulässig.

**Listing 5.3: Mehrfaches Abfragen einer Spalte.**

```
SQL> select Student_ID, Student_ID  
2 from Student;
```

```
STUDENT_ID STUDENT_ID  
-----  
10231300 10231300  
10231301 10231301  
10231302 10231302  
10231305 10231305  
10231306 10231306  
10231308 10231308  
10231309 10231309  
10231322 10231322  
10231314 10231314  
10231315 10231315  
10231317 10231317  
10231319 10231319  
10231320 10231320  
10231325 102313025  
10231326 10231326  
10231328 10231328  
10231329 10231329  
10231327 10231327  
10231330 10231330  
10231316 10231316  
10231303 10231303  
10231310 10231310  
10231318 10231318  
10231324 10231324  
10231304 10231304  
10231312 10231312  
10231321 10231321  
10231313 10231313  
10231311 10231311  
10231323 10231323  
10231307 10231307
```

31 rows selected.



*Sie können arithmetische Berechnungen durchführen, indem Sie die Aufgabe aus einer Hilfstabelle namens dual auswählen, wie z.B. in*



```
select 3.14159*20 from dual;
```

---



*Es erfolgt die Ausgabe:*



62.8318

## Von select zurückgelieferte Werte

Die von der select-Anweisung zurückgelieferten Ergebnisse bilden eine temporäre Tabelle. Jeder abgerufene Datensatz stellt eine Zeile in dieser temporären Tabelle dar, und alle Elemente der select-Liste sind Spalten. Wenn eine Abfrage keine Datensätze zurückgibt, kann man davon ausgehen, daß die temporäre Tabelle leer ist. Dieses Verhalten ist ein Grundprinzip des relationalen Modells.

## Verwendung von Ausdrücken in der select-Liste

Zusätzlich zu der Angabe von Spalten können Sie auch Ausdrücke in der select-Liste verwenden. Ausdrücke gehören zu denselben Datentypen wie Spalten - Zeichen, numerische Werte und Datumsinformationen. Durch die Verwendung von Operatoren, integrierten Funktionen und Konstanten können Sie den Bedürfnissen Ihrer Anwendung entsprechende komplexe Ausdrücke konstruieren.

Denken Sie daran, daß Oracle jedes Element in der select-Liste als separate Spalte betrachtet, selbst wenn der Ausdruck auf mehrere Spalten verweist.

## Arithmetische und logische Operatoren

Die arithmetischen und logischen Operatoren von SQL ähneln den Operatoren in C:

| Beschreibung   | Operator |
|----------------|----------|
| Addition       | +        |
| Subtraktion    | -        |
| Multiplikation | *        |
| Division       | /        |

|                     |    |
|---------------------|----|
| Kleiner als         | <  |
| Kleiner oder gleich | <= |
| Größer als          | >  |
| Größer oder gleich  | >= |
| Gleich              | =  |
| Ungleich            | != |

Die Course-Tabelle enthält zum Beispiel die Lerneinheiten (Units) aller Kurse. Wenn die aktuellen Kosten pro Einheit \$ 250 betragen, erhält man die Kosten für den Kurs mit der in Listing 5.4 gezeigten select-Anweisung.

***Listing 5.4: Verwendung eines arithmetischen Operators in einer select-Anweisung.***

```
SQL> select Title, Units*250
2 from Course;
```

```
TITLE UNITS*250
-----
INTRO TO ANTHROPOLOGY 750
PRE-CALCULUS 750
GENERAL CALCULUS 750
NUMBER THEORY 750
ADVANCED ARITHMETIC 750
INTRO TO PSYCHOLOGY 750
ABNORMAL PSYCHOLOGY 750
EARLY AMERICAN HISTORY 750
MODERN EUROPEAN HISTORY 750
INTRO TO BIOLOGY 750
INTRO TO ECONOMICS 750
MONETARY POLICY 750
INTRO TO PHILOSOPHY 750
INTRO TO ENGLISH LIT 750
MODERN ENGLISH LIT 750
INTRO TO STRUCTURES 750
MODERN PHILOSOPHY 750
ANCIENT GREEK PHILOSOPHERS 750
MAMMALIAN ANATOMY 750
INVERTEBRATE ANATOMY 750
WORKSHOP ON MARX 750
WORKSHOP ON JEFFERSON 750
INTRO TO CIRCUIT THEORY 750
INTRO TO DYNAMICS 750
SEMINAR ON CHAOS 500
SEMINAR ON NACIMERA 500
PHYS ANTHRO FIELD TECHNIQUES 750
EVOLUTIONARY GRAMMAR 500
SEMINAR ON THEME ANALYSIS 500
WORKSHOP ON NEUROSES 500
WORKSHOP ON NORMALITY 500
```

31 rows selected.



Anstatt sich die Prioritätsregeln für die in der SQL-Anweisung verwendeten arithmetischen Operatoren zu merken, sollten Sie stets Klammern einsetzen, wenn Sie sich hinsichtlich der richtigen Auswertung eines Ausdrucks unsicher sind.

## Zeichenfolgenoperatoren

Einer der wichtigsten Zeichenfolgenoperatoren in SQL ist der Verkettungsoperator, ||. Die Grammatik von SQL verlangt, daß Zeichenkonstanten in einfache Anführungszeichen zu setzen sind. Mit diesem Operator können Sie zwei oder mehrere Zeichenfolgen verketteten, wie in Listing 5.5 dargestellt.

### *Listing 5.5: Verwendung eines Operators für Zeichenfolgen.*

```
SQL> select Last_Name || ': ' || Position
2 from Instructor;
```

```
LAST_NAME||': '||POSITION
-----
HITCHCOCK: PROFESSOR
DANIELS: ASSOCIATE PROFESSOR
EDWARDS: PROFESSOR
CHANG: ASSISTANT PROFESSOR
JASON: ASSOCIATE PROFESSOR
TORRES: PROFESSOR
ANGELO: ASSOCIATE PROFESSOR
RICHARDSON: ASSISTANT PROFESSOR
PARKER: PROFESSOR
CHERNOW: ASSOCIATE PROFESSOR
PROFESSOR
WEISS: PROFESSOR
RESTON: ASSOCIATE PROFESSOR
BILLINGS: ASSISTANT PROFESSOR
YOUNG: ASSOCIATE PROFESSOR
NILAND: ASSOCIATE PROFESSOR
BATES: ASSISTANT PROFESSOR
```

17 rows selected.

Es folgen verschiedene Gründe für das Verketteten von Zeichenfolgen:

- **Einbetten von Zeichenfolgen in die von einer Abfrage zurückgegebenen Werte.** Sie möchten einen Formbrief an Ihre Kunden (Customer) erstellen und darin die Anrede (Salutation) mit einem Leerzeichen und dem Nachnamen des Kunden kombinieren.

```
select Salutation || ' ' || Last_Name
from Customer
order by Last_Name;
```

- **Kombinieren von Zeichenfolgen.** Zur Darstellung für den Benutzer muß Ihre Anwendung möglicherweise eine Teilzeichenkette aus einer Spalte mit einer Teilzeichenkette einer anderen Spalte kombinieren.
- **Erstellen neuer Werte, die einer Spalte zugeordnet werden können.** Bei der Implementierung Ihrer Anwendung müssen Sie eventuell vorhandene Daten von einem Format in ein anderes konvertieren.

Sie können den Verkettungsoperator für mehr als zwei Zeichenfolgen verwenden, wie in Listing 5.6 dargestellt.

**Listing 5.6: Mehrere Verkettungen.**

```
SQL> select Last_Name || ', ' || Position || ', can be contacted at ' ||  
2 Email  
3 from Instructor;  
  
LAST_NAME||', '||POSITION||', CANBECONTACTEDAT'| |EMAIL  
-----  
HITCHCOCK, PROFESSOR, can be contacted at BHITCHCOCK@FLUGLE.EDU  
LDANIELS, ASSOCIATE PROFESSOR, can be contacted at LDANIELS@FLUGLE.EDU  
EDWARDS, PROFESSOR, can be contacted at SEDWARDS@FLUGLE.EDU  
CHANG, ASSISTANT PROFESSOR, can be contacted at RCHANG@FLUGLE.EDU  
JASON, ASSOCIATE PROFESSOR, can be contacted at JJASON@FLUGLE.EDU  
TORRES, PROFESSOR, can be contacted at PTORRES@FLUGLE.EDU  
ANGELO, ASSOCIATE PROFESSOR, can be contacted at AANGELO@FLUGLE.EDU  
RICHARDSON, ASSISTANT PROFESSOR, can be contacted at NRICHARDSON@FLUGLE.EDU  
PARKER, PROFESSOR, can be contacted at WPARKER@FLUGLE.EDU  
CHERNOW, ASSOCIATE PROFESSOR, can be contacted at BCHERNOW@FLUGLE.EDU  
CHU, PROFESSOR, can be contacted at SCHU@FLUGLE.EDU  
WEISS, PROFESSOR, can be contacted at RWEISS@FLUGLE.EDU  
RESTON, ASSOCIATE PROFESSOR, can be contacted at RRESTON@FLUGLE.EDU  
BILLINGS, ASSISTANT PROFESSOR, can be contacted at BBILLINGS@FLUGLE.EDU  
YOUNG, ASSOCIATE PROFESSOR, can be contacted at AYOUNG@FLUGLE.EDU  
NILAND, ASSOCIATE PROFESSOR, can be contacted at MNILAND@FLUGLE.EDU  
BATES, ASSISTANT PROFESSOR, can be contacted at JBATES@FLUGLE.EDU  
  
17 rows selected.
```

## Integrierte Funktionen

Oracle stellt eine große Anzahl integrierter Funktionen bereit, die Sie bei der Bearbeitung und Konvertierung verschiedener Arten von Daten einsetzen können. Diese Funktionen können in folgende Kategorien eingeteilt werden:

- Zeichenfunktionen
- numerische Funktionen
- Datumsfunktionen
- Konvertierungsfunktionen
- Gruppenfunktionen
- sonstige Funktionen

Erläuterungen zu diesen Funktionen erhalten Sie am Tag 7, »Die in SQL integrierten Funktionen ausnutzen«.

## Festlegen von Kriterien in der where-Klausel

Normalerweise rufen Sie nicht alle Zeilen einer Tabelle ab, besonders dann nicht, wenn die Tabelle aus vielen Zeilen besteht. SQL verfügt über eine where-Klausel, in der Sie die zum Abrufen von Datensätzen zu verwendenden Kriterien festlegen können.

Eine where-Klausel besteht aus einer oder mehreren Bedingungen, die erfüllt werden müssen, bevor eine Zeile mit der Abfrage abgerufen werden kann. Wenn Sie zum Beispiel eine Liste der Erstsemester (FRESHMAN) des Flugle-Colleges haben möchten, würden Sie Year = 'FRESHMAN' in die where-Klausel einfügen, wie in Listing 5.7 dargestellt.

***Listing 5.7: Festlegen eines Kriteriums in einer where-Klausel.***

```
SQL> select Last_Name, First_Name, Street_Address
2 from Student
3 where
4 Year = 'FRESHMAN';
```

```
LAST_NAME FIRST_NAME STREET_ADDRESS
```

```
-----
SMYTHE JACKSON 144 OLIVE AVE.
REYNOLDS PAULA 7493 MAPLE ST.
TANAKA JEFFREY 838 PECAN RD.
MALLARD HENRY 123 WALNUT DR.
POSEN HUGO 9100 MAPLE ST.
CLAUSEN THOMAS 901 BIRCH RD.
GOMEZ LINDA 9391 MAPLE ST.
MASSEY RICHARD 431 PINE AVE.
ANASTATIA ANNA 831 BIRCH RD.
HOLMES IVAN 221 OLIVE AVE.
```

```
10 rows selected.
```

## Kombinieren von Bedingungen mit and und or

Sie können die Schlüsselwörter and und or verwenden, um mehrere von einer Abfrage zu erfüllende Bedingungen zu kombinieren. Um zum Beispiel eine Liste der Studenten im zweiten Jahr (SOPHOMORE) zu erhalten, deren Straße (Street\_Address) die Zeichenfolge ash enthält, würden Sie beide Bedingungen in der where-Klausel angeben, wie in Listing 5.8 gezeigt.

***Listing 5.8: Kombination von Abfragekriterien mit Hilfe von and.***

```
SQL> select Last_Name, First_Name, Street_Address
2 from Student
3 where
4 Year = 'SOPHOMORE' and
5 Street_Address like '%ASH%';
```

```
LAST_NAME FIRST_NAME STREET_ADDRESS
```

```
-----
GORDON IVY 544 ASH ST.
JACKEL LINDA 493 ASH ST.
DEURRE PIERRE 555 ASH ST.
PINKWATER PETER 533 ASH ST.
```

Bitte beachten Sie das Wort like in der vierten Zeile des obigen Beispiels. Dieser Operator ist eines der leistungsfähigsten Werkzeuge von SQL.





*Die Grundsyntax für die Verwendung des Operators like gestaltet sich folgendermaßen:*

```
column_name LIKE 'pattern'
```

Die Definition der Variablen lautet wie folgt:

*column\_name* ist eine gültige Spalte in der von der from-Klausel referenzierten Tabelle.

*pattern* ist das Zeichenfolgemuster, nach dem Sie suchen.

Das Zeichen % dient in diesem Zusammenhang als Platzhalter; es entspricht null oder mehreren Zeichen. Das Zeichen \_ (Unterstrich) wird als Platzhalter für ein einzelnes Zeichen verwendet.



*In den nachfolgenden Beispielen erscheint das Schlüsselwort null in Großbuchstaben. Ich habe die Großbuchstaben verwendet, um das Wort hervorzuheben, dies ist aber nicht unbedingt erforderlich; Sie können für alle von SQL reservierten Wörter in Oracle Groß- und Kleinschreibung verwenden.*

## Sortieren von Daten in der order by-Klausel

Die order by-Klausel legt fest, welche Spalten für die Sortierung der von der Abfrage zurückzugebenden Zeilen verwendet werden sollen. Die order by-Klausel ist optional, aber bitte bedenken Sie: Die Reihenfolge, in der Zeilen von einer Abfrage zurückgegeben werden, ist *immer* beliebig (ob es sich um eine indizierte Tabelle handelt oder nicht). Daher sollten Sie im allgemeinen eine order by-Klausel in der select-Anweisung angeben.

Sie brauchen zum Beispiel eine Liste der Kursleiter (Instructor) am Flugle-College, die zuerst nach dem Fachbereichsbezeichner (Department ID) und dann nach dem Nach- (Last\_Name) und Vornamen (First\_Name) des Kursleiters sortiert ist, wie in Listing 5.9 dargestellt.

**Listing 5.9: Sortierung abgefragter Zeilen nach mehreren Spalten.**

```
SQL> select Department_ID, Last_Name, First_Name
2  from Instructor
3  order by Department_ID, Last_Name, First_Name;
```

```
DEPARTMENT_ID LAST_NAME FIRST_NAME
-----
ANTHRO DANIELS LAURA
ANTHRO HITCHCOCK BORIS
BIO EDWARDS SAMANTHA
BIO WEISS ROBERTA
ECON PARKER WILLIAM
```

ENG CHU STEVEN  
ENGL CERNOW BESS  
HIST RESTON RAPHAEL  
HIST TORRES PETER  
MATH CHANG ROGER  
MATH NILAND MARTINA  
MATH YOUNG ALLAN  
PHILO ANGELO ARTHUR  
PHILO BILLINGS BENJAMIN  
PHILO RICHARDSON NANCY  
PSYCH BATES JOSEPH  
PSYCH JASON JERROLD

17 rows selected.



*Sie können Spalten in einer order by-Klausel angeben, selbst wenn diese nicht aus der Tabelle ausgewählt werden, wie in diesem Beispiel gezeigt:*

```
select Last_Name  
from Instructor  
order by Position
```



*Sie können in der order by-Klausel auch Spalten unabhängig davon angeben, ob sie Teil eines Indexes oder einer Tabelle sind.*

Standardmäßig sortiert Oracle die Zeilen in aufsteigender Reihenfolge. Um die Zeilen in absteigender Reihenfolge zu sortieren, müssen Sie das Schlüsselwort desc (für descending = absteigend) nach dem Spaltennamen eingeben. Sie können auf- und absteigende Spalten in derselben order by-Klausel nennen, wie in Listing 5.10 dargestellt.

***Listing 5.10: Sortierung in absteigender Reihenfolge.***

```
SQL> select Department_ID, Last_Name, First_Name  
2 from Instructor  
3 order by Department_ID desc, Last_Name, First_Name;
```

```
DEPARTMENT_ID LAST_NAME FIRST_NAME  
-----  
PSYCH BATES JOSEPH  
PSYCH JASON JERROLD  
PHILO ANGELO ARTHUR  
PHILO BILLINGS BENJAMIN  
PHILO RICHARDSON NANCY  
MATH CHANG ROGER  
MATH NILAND MARTINA
```

MATH YOUNG ALLAN  
HIST RESTON RAPHAEL  
HIST TORRES PETER  
ENGL CHERNOW BESS  
ENG CHU STEVEN  
ECON PARKER WILLIAM  
BIO EDWARDS SAMANTHA  
BIO WEISS ROBERTA  
ANTHRO DANIELS LAURA  
ANTHRO HITCHCOCK BORIS

17 rows selected.

## Zählen von Zeilen in einer Tabelle

Wenn Sie wissen möchten, wie viele Zeilen einer Tabelle dem angegebenen Kriterium entsprechen, aber die Zeilen selber nicht abrufen wollen, können Sie die count-Funktion verwenden. count gibt eine einzelne Zeile zurück, welche die Anzahl der dem genannten Kriterium entsprechenden Zeilen nennt. Listing 5.11 zeigt ein Beispiel.

*Listing 5.11: Zählen der Zeilen in einer Tabelle.*

```
SQL> select count(*)  
2 from Instructor  
3 where  
4 Position = 'ASSOCIATE PROFESSOR';
```

```
COUNT(*)  
-----  
7
```



Bei count handelt es sich um eine Gruppenfunktion (mehr darüber erfahren Sie am Tag 8, »Bilden fortgeschrittener Abfragen mit SQL«). Das Sternchen weist Oracle an, alle dem Kriterium entsprechenden Zeilen zurückzugeben. Statt des Sternchens können Sie auch einen Spaltennamen angeben; wenn Sie das tun, gibt Oracle allerdings nur jene Zeilen zurück, in denen dem angegebenen Spaltennamen ein Wert zugewiesen wurde (mit anderen Worten, Zeilen, deren Spaltenwert nicht null ist).

---

## Ermitteln von Zeilen mit Spaltenwert null

Ein großer Unterschied zwischen RDBMS (Management-Systemen für relationale Datenbanken) und älteren DBMS (Datenbankmanagementsystemen) ist das Konzept des Nullwerts. In nichtrelationalen Datenbanksystemen wird ein bestimmter Wert verwendet, um zu verdeutlichen, daß in einem Zeichenfeld oder einem numerischen Feld kein Wert genannt wird.

In einer relationalen Datenbank sagt der null-Wert einer Spalte verschiedene Dinge aus:

- Auf diese Spalte trifft in der betreffenden Zeile kein Wert zu.
- Der Spalte wurde noch kein Wert zugeordnet. (Es kann zum Beispiel vorkommen, daß ein Student noch kein Faxgerät

besitzt.)

Wenn Sie Datensätze aus einer Tabelle abrufen möchten, für die kein Spaltenwert definiert ist, können Sie die Kriterien in der where-Klausel nennen. Bei Listing 5.12 handelt es sich um eine Abfrage, welche die Namen der Studenten ohne Faxnummer zurückgibt.

***Listing 5.12: Abfrage der Zeilen, für die eine Spalte den Wert null enthält.***

```
SQL> select Last_Name, First_Name
2 from Student
3 where
4 Fax is null;
```

```
LAST_NAME FIRST_NAME
```

```
-----
```

```
SMYTHE JACKSON
HAN COREY
REYNOLDS PAULA
PARKER FREDERICK
TANAKA JEFFREY
COEN JOSEPH
NEWTON ELEANOR
PURCELL STEVEN
JACKEL LINDA
ROSEN ROSEMARY
MALLARD HENRY
GUSSEY LISA
MICHAELS MARCUS
DEURRE PIERRE
WONG LE-MING
ABBOT KENNETH
POSEN HUGO
GOMEZ LINDA
MASSEY RICHARD
FERGUSON RALPH
BING RUDOLPH
ANASTATIA ANNA
CHIN MICHAEL
HOLMES IVAN
PLOWCHARD CATHERINE
FERNANDEZ PAUL
PINKWATER PETER
MABEISI ANTON
```

```
28 rows selected.
```

Sie können auch den Operator not zum Abrufen von Zeilen verwenden, deren Spalten einen Wert ungleich null enthalten. Mit der Abfrage aus Listing 5.13 können Sie zum Beispiel die Anzahl der Studenten mit Faxnummer zählen.

***Listing 5.13: Abfrage der Zeilen, für die eine Spalte einen Wert ungleich null enthält.***

```
E SQL> select Last_Name, First_Name
2 from Student
3 where
4 Fax is not null;
```

```
LAST_NAME FIRST_NAME
```

-----  
GORDON IVY  
CLAUSEN THOMAS  
JACKSON ROBERT

Sie sollten sich darüber im klaren sein, wie null-Werte in arithmetischen Operationen verarbeitet werden. Damit Sie sehen können, wie ein null-Wert von dem Wert 0 abweicht, betrachten Sie bitte Listing 5.14. Nehmen wir an, Sie haben eine Tabelle mit dem Titel Intelligence (Intelligenz), die zwei Spalten enthält - Last\_Name und IQ.

**Listing 5.14: Abfrage aller Werte für Last\_Name und IQ.**

```
SQL> select Last_Name, IQ  
2 from Intelligence;
```

```
LAST_NAME IQ  
-----
```

```
SMITH 100  
GORDON 125  
JONES 150  
WILSON  
RICHARDS
```

In diesem Beispiel ist der IQ für Wilson und Richards gleich null. Wenn Sie den Durchschnitts-IQ [avg(IQ)] aus den Datensätzen der Intelligence-Tabelle wissen möchten, geben Sie die in Listing 5.15 gezeigte Abfrage ein.

**Listing 5.15: Bestimmung des durchschnittlichen IQs aller Zeilen, bei denen der Wert von IQ von null verschieden ist.**

```
SQL> select avg(IQ) from Intelligence;
```

```
AVG(IQ)  
-----
```

```
125
```

Wie Sie sehen können, wurden die Zeilen, bei denen der Wert von IQ gleich null ist, nicht für die Berechnung des Durchschnitts einbezogen. Oracle hat den durchschnittlichen IQ mit der Formel  $(100 + 125 + 150) / 3$  errechnet. Wenn Sie die null-Werte in 0 ändern, wird ein anderes Ergebnis erzielt, wie in Listing 5.16 dargestellt.



Um die Spaltenwerte in einer Tabelle zu ändern, verwenden Sie die update-Anweisung. Mehr darüber erfahren Sie am sechsten Tag.

---

**Listing 5.16: IQ wird überall dort auf 0 gesetzt, wo der aktuelle Wert null ist.**

```
SQL> update Intelligence
```

```
2 set IQ = 0
```

```
3 where
```

```
4 IQ is NULL;
```

```
2 rows updated.
```

```
SQL> select Last_Name, IQ
```

```
2 from Intelligence;
```

```
LAST_NAME IQ
-----
SMITH 100
GORDON 125
JONES 150
WILSON 0
RICHARDS 0

SQL> select avg(IQ) from Intelligence;

AVG(IQ)
-----
75
```

## Suche nach Zeilen mit Hilfe des like-Operators

Ein Beispiel für die Verwendung des like-Operators haben Sie bereits gesehen. Oracle-Benutzer setzen den like-Operator zum Durchsuchen einer Tabelle ein, wenn Sie sich hinsichtlich der Schreibweise der gesuchten Position nicht sicher sind.

Der Wohltäter des Flugle-Colleges, Reginald Flugle, hat einen Stipendienfonds für Studenten eingerichtet, deren Nachname die Buchstaben in enthält. Aus diesem Grund möchte die College-Registratorin, Jean Smith, alle Studenten, deren Nachname dieses Muster enthält, darüber informieren, daß sie für dieses Sonderstipendium in Frage kommen. Listing 5.17 zeigt, wie sie eine select-Anweisung konstruiert, um diese Studenten zu finden.

*Listing 5.17: Suche nach einem Muster mit Hilfe des like-Operators.*

```
SQL> select Student_ID, Last_Name, First_Name
2 from Student
3 where
4 Last_Name like '%IN%';

STUDENT_ID LAST_NAME FIRST_NAME
-----
10231318 BING RUDOLPH
10231304 CHIN MICHAEL
10231323 PINKWATER PETER
```

Jean wurde vom College-Präsidenten gebeten, eine Liste von Kursen zusammenzustellen, deren Beschreibung das Wort workshop enthält. Nachdem sie eine Weile über dieses Problem nachgedacht hat, startet Jean die in Listing 5.18 dargestellte Oracle-Abfrage.

*Listing 5.18: Bestimmung der Kurse, bei denen die Beschreibung das Muster workshop enthält.*

```
SQL> select Department_ID, Course_ID, Title
2 from Course
3 where
4 Description like '%workshop%';

DEPARTMENT_ID COURSE_ID TITLE
-----
ECON 189 MONETARY POLICY
PHILO 198 MODERN PHILOSOPHY
ECON 199 WORKSHOP ON MARX
HIST 199 WORKSHOP ON JEFFERSON
PSYCH 181 WORKSHOP ON NEUROSES
```

Allerdings erkannte Jean, daß diese Abfrage nicht die Kurse umfassen würde, deren Beschreibung (Description) mit dem Wort Workshop beginnt, da in diesem Fall der erste Buchstabe ein großes W ist. Aus diesem Grund hat sie die Abfrage, wie in Listing 5.19 dargestellt, geändert.

**Listing 5.19: Zeichenfolgenvergleich mit Hilfe der Funktion upper.**

```
SQL> select Department_ID, Course_ID, Title
2  from Course
3  where
4  upper(Description) like '%WORKSHOP%';
```

```
DEPARTMENT_ID  COURSE_ID  TITLE
-----
```

```
ECON 189 MONETARY POLICY
PHILO 198 MODERN PHILOSOPHY
ECON 199 WORKSHOP ON MARX
HIST 199 WORKSHOP ON JEFFERSON
ENGL 193 SEMINAR ON THEME ANALYSIS
PSYCH 181 WORKSHOP ON NEUROSES
```

```
6 rows selected.
```

Jean hatte recht. Anstatt nur fünf Datensätze, liefert die Abfrage nun sechs Datensätze zurück. Durch die Anwendung der Funktion upper auf die Spalte Description war sie in der Lage, den Inhalt der Spalte zu vergleichen, ohne sich darüber Gedanken machen zu müssen, ob das Muster nun in Groß- oder Kleinbuchstaben zu schreiben ist.

Sie können Zeichenfolgenmuster durch den Platzhalter % trennen, wie in Listing 5.20 gezeigt.

**Listing 5.20: Verwendung mehrerer Platzhalter.**

```
SQL> select Description
2  from Course
3  where
4  upper(Description) like '%STUDY%BEHAVIOR%';
```

```
DESCRIPTION
-----
```

```
Discussion and field study of the strange behavior of the Nacimera culture
```

Manchmal kommt es allerdings vor, daß Sie eine Zeichenfolge suchen, die tatsächlich das Zeichen % enthält. Normalerweise wird das %-Zeichen in einfachen Anführungszeichen vom Oracle-RDBMS als Platzhalter interpretiert. Wenn das Oracle-RDBMS ein Zeichen als Literal interpretieren soll, setzen Sie das Code-Umschaltzeichen vor das betreffende Zeichen. Wie in Listing 5.21 gezeigt, wollen Sie Kursbeschreibungen abrufen, die das Zeichen % enthalten; daher setzen Sie \% zwischen Platzhalter.

**Listing 5.21: Einleitung eines Literalzeichens mit Hilfe eines Code-Umschaltzeichens.**

```
SQL> select Description
2  from Course
3  where
4  Description like '%\%' escape '\';
```

```
DESCRIPTION
-----
```

```
Learn about advanced operations such as % and logarithms.
```

In dieser Abfrage wird der umgekehrte Schrägstrich (\) verwendet, um Oracle mitzuteilen, daß das %-Zeichen nach dem \ als Literal interpretiert werden soll. Die gleiche Methode können Sie auch einsetzen, wenn Sie nach einem Unterstrich suchen wollen, anstatt diesen als Platzhalter für ein einzelnes Zeichen einzusetzen.

Es folgen einige Vorschläge für die Verwendung des like-Operators bei Ihrer Suche:

- Wenden Sie die Funktionen upper und lower auf die zu suchende Spalte an, so daß alle Zeichen die gleiche Schreibweise haben.
- Wenn Sie nach Datensätzen suchen, bei denen eine Zeichenfolge am Ende einer Spalte steht, sollten Sie den Platzhalter nicht zu Beginn des Musters einsetzen:

```
select Department_ID, Title_ID, Description
from Course
where
upper(Description) like 'SEMINAR%;
```

- Auch bei der Suche nach Datensätzen, bei denen eine Zeichenfolge am Ende einer Spalte steht, sollten Sie den Platzhalter nicht am Ende des Musters einsetzen:

```
select Department_ID, Title_ID, Description
from Course
where
upper(Description) like '%TOPICS';
```

## Suche nach Zeilen mit dem between-Operator

In dieser Lektion habe ich bereits erläutert, daß eine select-Anweisung auf unterschiedliche Weise strukturiert sein kann, um zum selben Ergebnis zu führen. Der between-Operator ist ein gutes Beispiel für diese Flexibilität.

Der between-Operator ist ein inklusiver Operator und recht flexibel; er funktioniert mit numerischen Werten, Zeichenfolgen und Datumswerten. Nehmen wir zum Beispiel an, daß die aktuellen Kosten einer Lerneinheit am Flugle-College \$ 250 betragen. Da sich die Kosten für einen Kurs aus der Summe der Kosten pro Einheit und der anfallenden Zusatzkosten zusammensetzt, kann die Liste der Kurse, deren Kosten zwischen \$ 760 und \$ 800 liegen, aus der in Listing 5.22 gezeigten Abfrage abgeleitet werden.

*Listing 5.22: Verwendung des between-Operators.*

```
SQL> select Title, Units*250 + Additional_Fees
2 from Course
3 where
4 (Units*250 + Additional_Fees) between 760 and 800;
```

```
TITLE UNITS*250+ADDITIONAL_FEES
```

```
-----
ADVANCED ARITHMETIC 760
INTRO TO PSYCHOLOGY 775
ABNORMAL PSYCHOLOGY 770
INTRO TO ECONOMICS 775
INTRO TO CIRCUIT THEORY 795
INTRO TO DYNAMICS 785
```

6 rows selected.

Bei der vorangegangenen Abfrage handelt es sich wirklich um die gleiche wie in Listing 5.23.



**Listing 5.23: Eine Alternative zur Verwendung des between-Operators.**

```
SQL> select Title, Units*250 + Additional_Fees
2 from Course
3 where
4 (Units*250 + Additional_Fees) >= 760 and
5 (Units*250 + Additional_Fees) <= 800;
```

```
TITLE UNITS*250+ADDITIONAL_FEES
-----
```

```
ADVANCED ARITHMETIC 760
INTRO TO PSYCHOLOGY 775
ABNORMAL PSYCHOLOGY 770
INTRO TO ECONOMICS 775
INTRO TO CIRCUIT THEORY 795
INTRO TO DYNAMICS 785
```

```
6 rows selected.
```

Wie Sie sehen, handelt es sich bei dem des between-Operator um das Äquivalent zweier durch AND verbundener Bedingungen. Bei richtiger Anwendung vereinfacht der between-Operator eine Abfrage.

## Der in-Operator

Ein weiterer Operator, der den Wert einer Spalte oder eines Ausdrucks mit einer Liste möglicher Werte vergleicht, ist der in-Operator.



*Die Syntax für den in-Operator sieht folgendermaßen aus:*

---

*expression* IN (*expression1*, *expression2*, ... *expressionN*)

Die Definition der Variablen lautet wie folgt:

*expression* ist ein gültiger SQL-Ausdruck.

*expression1* bis *expressionN* stellen eine Liste gültiger SQL-Ausdrücke dar.

Der in-Operator gibt einen booleschen Wert zurück, der entweder true oder false ist:

true, wenn der Ausdruck einem der Werte in der Liste der Ausdrücke entspricht

false, wenn der Ausdruck keinem der Werte in der Liste der Ausdrücke entspricht

Als Beispiel für die Verwendung des in-Operators nehmen wir an, Sie wollen nur die Kursleiter der Fachabteilungen Mathematik, Anthropologie oder Psychologie abrufen. Falls die Liste der möglichen zu prüfenden Werte lang ist, erspart der in-Operator Ihnen eine Reihe von Eingaben sowie dem Oracle-Parser für SQL-Anweisungen einige Verarbeitungszeit. Listing

5.24 zeigt den Code.

***Listing 5.24: Verwendung des in-Operators.***

```
SQL> select Last_Name, First_Name, Department_ID
2 from Instructor
3 where
4 Department_ID in ('MATH', 'ANTHRO', 'PSYCH');
```

```
LAST_NAME FIRST_NAME DEPARTMENT_ID
```

```
-----
HITCHCOCK BORIS ANTHRO
DANIELS LAURA ANTHRO
CHANG ROGER MATH
JASON JERROLD PSYCH
YOUNG ALLAN MATH
NILAND MARTINA MATH
BATES JOSEPH PSYCH
```

```
7 rows selected.
```

Eine Alternative zum in-Operator ist in Listing 5.25 dargestellt.

***Listing 5.25: Eine Alternative zur Verwendung des in-Operators.***

```
SQL> select Last_Name, First_Name, Department_ID
2 from Instructor
3 where
4 Department_ID = 'MATH' or
5 Department_ID = 'ANTHRO' or
6 Department_ID = 'PSYCH';
```

```
LAST_NAME FIRST_NAME DEPARTMENT_ID
```

```
-----
HITCHCOCK BORIS ANTHRO
DANIELS LAURA ANTHRO
CHANG ROGER MATH
JASON JERROLD PSYCH
YOUNG ALLAN MATH
NILAND MARTINA MATH
BATES JOSEPH PSYCH
```

```
7 rows selected.
```

Sie können das Schlüsselwort not mit dem in-Operator kombinieren, so daß eine Bedingung als true angesehen wird, wenn ein Ausdruck keinem der in der Liste enthaltenen Ausdrücke entspricht. Siehe hierzu Listing 5.26.

***Listing 5.26: Verwendung des not in-Operators.***

```
SQL> select Last_Name, First_Name, Department_ID
2 from Instructor
3 where
4 Department_ID not in ('MATH', 'ANTHRO', 'PSYCH');
```

```
LAST_NAME FIRST_NAME DEPARTMENT_ID
```

```
-----
EDWARDS SAMANTHA BIO
```

```
TORRES PETER HIST
ANGELO ARTHUR PHILO
RICHARDSON NANCY PHILO
PARKER WILLIAM ECON
CHERNOW BESS ENGL
CHU STEVEN ENG
WEISS ROBERTA BIO
RESTON RAPHAEL HIST
BILLINGS BENJAMIN PHILO
```

10 rows selected.

Wenn Sie den not in-Operator nicht verwenden wollen, müssen Sie alle Werte, denen der Ausdruck oder die Spalte nicht entsprechen soll, mit dem logischen Operator and »verketteten«. Listing 5.27 zeigt ein Beispiel.

**Listing 5.27: Eine Alternative zur Verwendung des not in-Operators.**

```
SQL> select Last_Name, First_Name, Department_ID
2 from Instructor
3 where
4 Department_ID != 'MATH' and
5 Department_ID != 'ANTHRO' and
6 Department_ID != 'PSYCH';
```

```
LAST_NAME FIRST_NAME DEPARTMENT_ID
```

```
-----
EDWARDS SAMANTHA BIO
TORRES PETER HIST
ANGELO ARTHUR PHILO
RICHARDSON NANCY PHILO
PARKER WILLIAM ECON
CHERNOW BESS ENGL
CHU STEVEN ENG
WEISS ROBERTA BIO
RESTON RAPHAEL HIST
BILLINGS BENJAMIN PHILO
```

10 rows selected.

## Mit einem Alias auf Spalten verweisen

Bei der Angabe eines komplexen Ausdrucks in einer select-Liste können Sie durch Zuordnung eines Aliasnamens dokumentieren, was der Ausdruck darstellt.



*Die Syntax für eine select-Liste gestaltet sich folgendermaßen:*

---

```
expression_name1 [ [AS] alias_name1}, ..., expression_nameN
Â[ [AS] alias_nameN]
```

Die Definition der Variablen lautet wie folgt:

*expression\_name* ist ein Ausdruck, der auf null oder mehrere Spaltennamen verweist.

*alias\_name* ist ein für die Referenzierung von *expression\_name* in anderen Teilen der select-Anweisung verwendeter Aliasname.

Bitte beachten Sie, daß es sich bei as um ein optionales Schlüsselwort handelt.

Sie können das Beispiel in Listing 5.23 verwenden, um den Einsatz eines Aliasnamens zu verdeutlichen. Lassen Sie uns noch einmal wiederholen: Die aktuellen Kosten eines Kurses am Flugle-College betragen \$ 250 pro Einheit plus zusätzliche Kosten, was mit der Abfrage in Listing 5.28 ausgedrückt wird.

***Listing 5.28: Verwendung eines Alias in einer select-Anweisung.***

```
SQL> select Title, Units*250 + Additional_Fees as Course_Cost
2 from Course
3 where
4 Department_ID = 'ECON';
```

```
TITLE COURSE_COST
-----
INTRO TO ECONOMICS 775
MONETARY POLICY 1500
WORKSHOP ON MARX 750
```

Indem Sie dem Ausdruck Units\*250 + Additional\_Fees den Alias Course\_Cost zuordnen, gewinnen Sie zwei Vorteile:

- Sie haben nun einen Namen, der den Ausdruck genau beschreibt.
- Sie können in der order by-Klausel auf einen Alias verweisen.

## Eine Unterabfrage verwenden

Eine Unterabfrage wird als select-Anweisung definiert, die in einer anderen DML-Anweisung erscheint - einer anderen select-, update-, delete- oder insert-Anweisung. In einer select-Anweisung ist die Unterabfrage Teil einer Bedingung in der where-Klausel. Die folgende Abfrage wählt den Titel aller Kurse, deren zusätzliche Kosten niedriger sind oder den durchschnittlichen Zusatzkosten aller Kurse entsprechen:

```
select Title
from Course
where
Additional_Fees <=
(select avg(Additional_Fees) from Course)
```

Bei der Verwendung von Unterabfragen sollten Sie einige Dinge beachten:

- Die Unterabfrage muß in Klammern gesetzt werden.
- Die Anzahl der von der Unterabfrage zurückzuliefernden *Zeilen* muß der Anzahl der von der Funktion oder dem Operator erwarteten Werte entsprechen. Im vorangegangenen Beispiel erwartet der Operator <= einen zu vergleichenden Einzelwert, und die Funktion avg (bei der es sich um eine Gruppenfunktion handelt) gibt einen Einzelwert zurück.
- Die Anzahl der von der Unterabfrage zurückgegebenen *Spalten* muß mit der Anzahl der von der Funktion oder dem Operator erwarteten Spalten übereinstimmen.

- Die order by-Klausel wird nicht innerhalb einer Unterabfrage eingesetzt.

## Erstellen einer neuen Tabelle mit Hilfe der select-Anweisung

Betrachten wir eine Form der create table-Anweisung, bei der eine Unterabfrage eingesetzt wird, um die Struktur der zu erstellenden Tabelle anzugeben. Diese Anweisung wird Ihnen bei der Entwicklung und Prüfung Ihrer Anwendung sicherlich als sehr hilfreich erscheinen.

Wenn Sie mit dem Inhalt einer Tabelle experimentieren möchten - Hinzufügen, Löschen und Aktualisieren mehrerer Zeilen -, wären Sie gut beraten, eine Kopie der Tabelle anzufertigen, mit der Sie experimentieren möchten. Nehmen wir an, daß es sich bei der vorhandenen Tabelle um eine Liste der Kunden der letzten zehn Jahre handelt, die viele Datensätze - sagen wir 100.000 - enthält, Sie für Ihre Versuche aber nur eine Teilmenge dieser Zeilen verwenden möchten. Besonders interessiert sind Sie an den in Kalifornien ansässigen Kunden. Sie können eine Tabelle erstellen, die eine Teilmenge aller Kunden (Customer\_Subset) enthält, indem Sie die create table-Anweisung mit der select-Anweisung kombinieren, wie in Listing 5.29 dargestellt.

*Listing 5.29: Erstellung einer Tabelle aus einer Teilmenge von Zeilen einer anderen Tabelle.*

```
SQL> create table Customer_Subset
2 as
3 select *
4 from Customer
5 where
6 State = 'CA';
```

Table created.



*Betrachten wir nun die Syntax genau:*

---

```
CREATE TABLE new_table_name
AS
select_stmt
```

Die Definition der Variablen lautet wie folgt:

*new\_table\_name* steht für den Namen der zu erstellenden Tabelle.

*select\_stmt* ist eine gültige select-Anweisung.

Zusätzlich können Sie die create table- und die select-Anweisung noch für einen anderen Zweck gemeinsam nutzen. Wenn Sie eine weitere Tabelle erstellen wollen, welche dieselbe Struktur wie eine vorhandene Tabelle aufweisen soll - mit denselben Spaltendefinitionen, jedoch ohne die Daten -, können Sie folgende Anweisung einsetzen:

```
CREATE TABLE my_new_empty_table
AS
SELECT *
FROM existing_table
WHERE
```

1 = 2

Nun werden Sie wahrscheinlich sagen: »1 ist niemals gleich 2.« Das ist richtig. Und aus diesem Grund werden keine der Zeilen aus *existing\_table* in *my\_new\_empty\_table* kopiert. Die neue Tabelle verfügt über dieselben Spaltendefinitionen wie *existing\_table*, aber nicht über Daten. Sie könnte auch jede *false*-Anweisung verwenden, um das gleiche zu erreichen.

## Zusammenfassung

In der heutigen Lektion wurden einige Grundlagen von SQL vorgestellt:

- Bei der Datenmanipulationssprache (DML) handelt es sich um eine Kategorie von SQL, die aus der *select*-, *insert*-, *update*- und der *delete*-Anweisung besteht.
- Jede *select*-Anweisung enthält eine *select*-Liste und eine *from*-Klausel, welche die Tabelle bezeichnet, aus der die Datensätze abgerufen werden sollen.
- Normalerweise enthält die *select*-Anweisung auch eine *where*-Klausel, in der die Kriterien angegeben werden, denen die von der *select*-Anweisung zurückgegebenen Zeilen entsprechen müssen. Auch die *order by*-Klausel wird zur Angabe der Spalten eingesetzt, die verwendet werden, um die Reihenfolge festzulegen, in der die Zeilen von der Abfrage zurückgegeben werden sollen.
- Die von den *select*-Anweisungen zurückgegebenen Ergebnisse können Sie sich als temporäre Tabelle vorstellen.
- Geben Sie die abzurufenden Spalten stets ausdrücklich in der *select*-Anweisung an.

## Wie geht es weiter?

Am sechsten Tag lernen Sie, wie Sie andere DML-Anweisungen - *insert*, *update* und *delete* - einsetzen, um den Inhalt einer Tabelle zu ändern. Sie werden erfahren, wie eine *select*-Anweisung in den *update*- und *delete*-Anweisungen referenziert werden kann, um die betreffenden Zeilen zu bestimmen.

## Fragen und Antworten

Frage:

Kann der *between*-Operator sowohl für Ausdrücke mit numerischen Werten als auch Zeichen verwendet werden?

Antwort:

*Ja. Sie haben bereits ein Beispiel dafür gesehen, wie der **between-Operator mit einem numerischen Ausdruck verwendet wird. Listing 5.31 zeigt ein Beispiel für den Einsatz des between-Operators mit einer Zeichenfolge, um eine Liste der Kurstitel zu bekommen, die mit den Zeichen S und W beginnen. Bitte beachten Sie, daß die Abfrage in Listing 5.30 keine Titel zurückgeben würde, die mit dem Wort Workshop beginnen, weil dieses über W »hinausgeht«.***

*Listing 5.30: Anwendung des between-Operators auf Zeichenfolgen.*

```
SQL> select Title
2 from Course
3 where
4 Title between 'S' and 'W';
```

```
TITLE
-----
```

SEMINAR ON CHAOS  
SEMINAR ON NACIMERA  
SEMINAR ON THEME ANALYSIS

Frage:

Welchen internen Wert verwendet Oracle zur Speicherung eines *null*-Werts für einen Datensatz?

Antwort:

*Im allgemeinen können Sie den Wert nicht »sehen«. Außerdem brauchen Sie den Wert eigentlich nicht zu kennen, da Sie eine Spalte nur unter Verwendung des **is null/is not null-Vergleichs überprüfen können. Als Alternative können Sie eine integrierte Funktion, nvs, verwenden, die über zwei Argumente verfügt: Bei dem ersten Argument handelt es sich um die auszuwertende Spalte oder den Ausdruck, beim zweiten Argument um den Wert, der zurückgegeben werden soll, wenn das erste Argument gleich null ist.***

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

### Test

1. Richtig oder falsch? Sie müssen eine Spalte in die select-Liste aufnehmen, wenn Sie die von der select-Anweisung für diese Spalte zurückgegebenen Zeilen sortieren möchten.
2. Was ist an dieser Anweisung falsch?

```
select First_Name  
from Student  
order by Last_Name  
where  
Last_Name like '%IN%';
```

3. Richtig oder falsch? Eine Spalte muß zuerst indiziert werden, bevor sie in der order by-Klausel angegeben werden kann.

### Übungen

Konstruieren Sie unter Verwendung der in dieser Lektion behandelten Course-Tabelle eine select-Anweisung, welche Department ID, Course ID und Course Title sortiert nach Department ID und Course ID zurückgibt, und zwar für alle Kurse, deren Beschreibung das Wort introduc enthält, unabhängig von der Groß- und Kleinschreibung.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---





# 6

# Mit SQL Daten modifizieren

[SQL als Datenmanipulationssprache](#)

[Hinzufügen von Zeilen mit Hilfe der insert-Anweisung](#)

[Angabe von Werten in der insert-Anweisung](#)

[Eine Unterabfrage in einer insert-Anweisung verwenden](#)

[Ändern der Daten mit Hilfe der update-Anweisung](#)

[In Mengen denken, nicht in Datensätzen](#)

[Eine Unterabfrage in einer update-Anweisung verwenden](#)

[Löschen von Daten mit Hilfe der delete-Anweisung](#)

[Gleichzeitigkeit](#)

[Einen Spaltenwert mit Hilfe einer update-Anweisung auf null setzen](#)

[Standardwerte und null](#)

[Eine Transaktion ist eine logische Arbeitseinheit](#)

## [Sichern der Arbeit mit Hilfe der commit-Anweisung](#)

### [Savepoints](#)

### [Wie geht es weiter?](#)

### [Fragen und Antworten](#)

### [Workshop](#)

---

An Tag 5, »Einführung in SQL«, haben wir kurz über den DML (Datenmanipulationssprachen)-Aspekt von SQL gesprochen und uns ausführlich mit der Anweisung select beschäftigt. Die Anweisung select kann die Inhalte von Tabellen nur lesen, sie besitzt nicht die Fähigkeit, Daten zu erzeugen oder zu verändern. Diese Lektion untersucht die Verwendung der drei verbleibenden DML-Anweisungen - insert, update und delete.

Wenn Sie eine Anwendung für Oracle entwickeln, werden Sie vielleicht ein Front-End-Werkzeug wie Oracle Forms, Visual Basic oder PowerBuilder verwenden. Diese Umgebungen zur Applikationsentwicklung erzeugen intern viele insert, update und delete-Anweisungen. Trotzdem ist es bei fast jeder Anwendung erforderlich, Skripte mit SQL-Anweisungen zu entwickeln. Es kann auch nötig sein, Prozeduren und Funktionen in PL/SQL zu schreiben, die solche Anweisungen enthalten.

Ein weiterer Grund zu lernen, wie man Daten mittels SQL verändert, ist das Übertragen und Umwandeln von Daten aus anderen Datenbanken in eine Oracle-Datenbank. Um Fremddaten zu portieren, müssen Sie einen Satz von Skripten entwickeln, die SQL-Anweisungen enthalten, mit denen die Daten »gereinigt« werden - um beispielsweise falschen Code oder Werte zu korrigieren - bevor die Daten zu Ihrer Oracle-Datenbank hinzugefügt werden.

## **SQL als Datenmanipulationssprache**

An Tag 3, »Logischer Datenbankentwurf«, haben Sie die drei Perspektiven einer Datenbank kennengelernt: Die Benutzerperspektive, die konzeptionelle Perspektive und die physikalische Perspektive. Der beste Weg, vollständig zu verstehen, wie SQL Daten modifiziert, ist, sich auf die konzeptionelle Perspektive zu konzentrieren. Denken Sie nur an Tabellen, Spalten und Zeilen, und Sie werden SQL und Oracle schneller beherrschen. Sie sollten sich am Anfang nicht darum kümmern, wie Oracle eine SQL-Anweisung ausführt. Konzentrieren Sie sich statt dessen auf die Aufgabe der SQL-Anweisung.

Hier ist ein weiterer nützlicher Hinweis für die Verwendung von DML. Wenn Sie über die

Auswirkungen einer SQL-Anweisung (insert, update, oder delete) nachdenken, versuchen Sie sich vorzustellen, daß eine Gruppe von Zeilen und nicht nur eine einzelne Zeile betroffen ist.

Das Erste, was viele Programmierer über Oracle lernen wollen, sind die Interna der Dateiformate und spezielle Codes. Dieser Ansatz ist jedoch falsch! Merken Sie sich folgendes:

- Das Format der Datenbankdateien von Oracle ist ein proprietäres Format der Firma Oracle.
- Diese Formate und Codes ändern sich mit jeder neuen Version von Oracle.
- Ihre Anwendung ist leichter zu warten, wenn Sie sich auf einen Industriestandard wie SQL verlassen anstatt auf Ihre Kenntnisse über die Dateiformate von Oracle. Das erleichtert auch das Portieren einer Oracle-Anwendung zu einem anderen RDBMS.

## Hinzufügen von Zeilen mit Hilfe der insert-Anweisung

Die insert-Anweisung fügt Zeilen zu einer Tabelle hinzu. Sie können Literalwerte oder Ausdrücke für die Speicherung in einer Tabellenzeile angeben.



*Die Bezeichnung insert bringt SQL-Anfänger dazu, anzunehmen, daß sie kontrollieren können, wo eine Zeile in eine Tabelle eingefügt wird. Denken Sie daran, daß einer der wichtigen Gründe für die Verwendung relationaler Datenbanken ist, daß sie logische Unabhängigkeit der Daten bieten - mit anderen Worten: Eine Tabelle hat keine implizite Ordnung. Eine neu eingefügte Zeile wird an einer beliebigen Stelle in eine Tabelle eingefügt.*

---

## Die Syntax der insert-Anweisung



*Die insert-Anweisung hat zwei Formen. Die erste Form ist*

---

```
INSERT INTO table_name  
[(column_name[, column_name]...[, column_name])]
```

## VALUES

`(column_value[,column_value]...[column_value])`

Die Variablen sind wie folgt definiert:

- *table\_name* ist Tabelle, in die die Zeile eingefügt werden soll.
- *column\_name* ist eine Spalte, die zu *table\_name* gehört.
- *column\_value* ist ein Literalwert oder ein Ausdruck, der dem zugehörigen *column\_name* entspricht.

Stellen Sie sich beispielsweise vor, Sie wollten zur Kurstabelle einen neuen Kurs hinzufügen. Listing 6.1 enthält die insert-Anweisung, die das bewerkstelligt.

### *Listing 6.1: Beispiel für eine insert-Anweisung.*

```
SQL> insert into Course
2 (Course_ID, Department_ID, Title, Description, Units,
Additional_Fees)
3 values
4 ('501', 'PSYCH', 'PSYCH IN FILM',
5 'Seminar on the portrayal of psychologists and psychiatrists in
film',
6 3, 25);
```

1 row created.

Beachten Sie, daß die Anzahl der Spalten in der Liste der Spaltennamen der Anzahl der Literalwerte oder Ausdrücke in den Klammern nach dem Schlüsselwort values entsprechen muß. Listing 6.2 enthält ein Beispiel dafür, was passiert, wenn sechs Spalten, aber nur fünf Literalwerte angegeben werden. Wenn Sie mehr Spaltennamen als Werte angeben, gibt Oracle eine Fehlermeldung zurück. Oracle kann natürlich nicht wissen, welcher Spaltenwert fehlt.

### *Listing 6.2: Es sind mehr Spalten definiert, als in der insert-Anweisung Werte geliefert werden.*

```
SQL> insert into Course
2 (Course_ID, Department_ID, Title, Description, Units,
Additional_Fees)
3 values
4 ('501', 'PSYCH', 'PSYCH IN FILM',
5 3, 25);
values
*
```

ERROR at line 3:

ORA-00947: not enough values

Umgekehrt zeigt Listing 6.3, was geschieht, wenn Sie weniger Spaltennamen als Werte angeben.

**Listing 6.3: Es sind weniger Spalten definiert, als in der insert-Anweisung Werte geliefert werden.**

```
SQL> insert into Course
2 (Course_ID, Department_ID, Title, Description, Units)
3 value
4 ('501', 'PSYCH', 'PSYCH IN FILM',
5 'Seminar on the portrayal of psychologists and psychiatrists in
film',
6 3, 25);
values
*
ERROR at line 3:
ORA-00913: too many values
```

Wenn ein Spaltenname in der insert-Anweisung falsch geschrieben wird, gibt Oracle eine Fehlermeldung zurück. In Listing 6.4 wird die Spalte Units fälschlicherweise als Unit geschrieben.

**Listing 6.4: Ein in einer insert-Anweisung falsch geschriebener Spaltenname.**

```
SQL> insert into Course
2 (Course_ID, Department_ID, Title, Description, Unit)
3 values
4 ('501', 'PSYCH', 'PSYCH IN FILM',
5 'Seminar on the portrayal of psychologists and psychiatrists in
film',
6 3);
(Course_ID, Department_ID, Title, Description, Unit)
*
ERROR at line 2:
ORA-00904: invalid column name
```



Wenn Sie insert mit einer langen Liste von Spaltennamen ausführen und Oracle ora-00947 oder ora-00913 zurückgibt, ist es Ihre Aufgabe, die Zuordnung durchzuführen und die Liste der Spaltennamen mit der Liste der Wertausdrücke abzustimmen. Wenn Sie das Problem bei der Prüfung immer noch nicht finden können, versuchen Sie die Zahl der Spalten zu verringern und das Problem zu isolieren.

---

## Angabe von Werten in der insert-Anweisung

Jeder Spaltenwert, der in einer insert-Anweisung angegeben wird, muß einen der folgenden Werte aufweisen:

- eine null
- einen Literalwert wie 3.14159 oder »Rettich«
- einen Ausdruck, der aus Operatoren und Funktionen besteht, wie beispielsweise SUBSTR>Last\_Name, 1,4)

In der insert-Anweisung können Sie Literalwerte mit Ausdrücken mischen. Listing 6.5 demonstriert die Verwendung der insert-Anweisung, bei der eine zusätzliche Gebühr für einen Kurs in Höhe von 10 % der Anzahl der Einheiten multipliziert mit \$250 angegeben wird.

*Listing 6.5: Verwendung eines Ausdrucks in einer insert-Anweisung.*

```
SQL> insert into Course
2 (Course_ID, Department_ID, Title, Description, Units,
Additional_Fees)
3 values
4 ('501', 'PSYCH', 'PSYCH IN FILM',
5 'Seminar on the portrayal of psychologists and psychiatrists in
film',
6 3, 0.10 * 3 * 250);
```

1 row created.

```
SQL> select Additional_Fees
2 from Course
3 where
4 Course_ID = '501' and
5 Department_ID = 'PSYCH';
```

ADDITIONAL\_FEES

-----

75

## Der Datentyp einer Spalte und eines Werts müssen übereinstimmen

Mit wenigen Ausnahmen (die wir besprechen werden) müssen der Datentyp für eine Spalte und den dazugehörigen Wert identisch sein. Es ist beispielsweise nicht sinnvoll, eine Zeichenfolge in eine numerische Spalte einzufügen, wie in Listing 6.6 gezeigt wird.

***Listing 6.6: In einer insert-Anweisung stimmen die Datentypen von Spalten und Werten nicht überein.***

```
SQL> insert into Course
2 (Course_ID, Department_ID, Title, Description, Units,
Additional_Fees)
3 values
4 ('501', 'PSYCH', 'PSYCH IN FILM',
5 'Seminar on the portrayal of psychologists and psychiatrists in
film',
6 3, 'Seventy-five dollars');
3, 'Seventy-five dollars')
*
ERROR at line 6:
ORA-01722: invalid number
```

Im vorhergehenden Beispiel gibt Oracle den Fehlercode ORA-01722 zurück, da die Zeichenfolge »Seventy-five dollars« nicht in einer als Zahl definierten Spalte gespeichert werden kann. Wenn die Zeichenfolge jedoch eine zulässige Zahl ist, wird die insert-Anweisung erfolgreich ausgeführt, wie in Listing 6.7 gezeigt wird. Dies ist eine Ausnahme von der Regel, daß eine Spalte und der zugewiesene Wert denselben Datentyp haben müssen.

***Listing 6.7: Umwandlung eines numerischen Werts in eine Zeichenfolge innerhalb einer insert-Anweisung.***

```
SQL> insert into Course
2 (Course_ID, Department_ID, Title, Description, Units,
Additional_Fees)
3 values
4 ('501', 'PSYCH', 'PSYCH IN FILM',
5 'Seminar on the portrayal of psychologists and psychiatrists in
film',
6 3, '75');

1 row created.
```

Eine andere Ausnahme von dieser Regel betrifft Zeichenfolgen und Datumsangaben. Listing 6.8 zeigt, wie eine Zeichenkette, die dem voreingestellten Datumsformat von Oracle entspricht (*TT-MMM-JJ*), in eine Datumsspalte eingefügt werden kann.

***Listing 6.8: Einfügen einer Datumsangabe mit dem Standard-Datumsformat.***

```
SQL> insert into Student_Schedule
2 (Student_ID, Class_ID, Grade, Date_Grade_Assigned)
```

```
3 values
```

```
4 ('10231311', '104200', 'B', '02-JUN-97');
```

```
1 row created.
```

Obwohl 02-JUN-97 ein Zeichenfolgenliteral und kein Datumsliteral ist, kann es Date\_Grade\_Assigned zugewiesen werden, weil es das voreingestellte Datumsformat von Oracle verwendet: *TT-MMM-JJ*.

## Pseudospalten als Werte verwenden

Oracle bietet eine Gruppe von Funktionen, die *Pseudospalten* genannt werden. Oracle hat diese Funktionen »Pseudospalten« genannt, weil sie auf den ersten Blick wie Spalten aussehen. Zwei häufig verwendete Pseudospalten sind

- user Der Oracle-Benutzer, der momentan mit der Oracle-Datenbank verbunden ist
- sysdate Die aktuelle Zeit und das aktuelle Datum

Sie können Pseudospalten in einer insert-Anweisung verwenden, um einer Spalte einen Wert zuzuweisen. user und sysdate können beispielsweise den Namen des Oracle-Benutzers speichern, der eine Zeile erzeugt hat, sowie den Zeitpunkt, zu dem die Zeile eingefügt wurde, wie in Listing 6.9 gezeigt wird.

**Listing 6.9: Verwendung von user und sysdate in einer insert-Anweisung.**

```
SQL> insert into Student_Schedule
2 (Student_ID, Class_ID, Grade, Date_Grade_Assigned,
3 Created_Username, Created_Timestamp)
4 values
5 ('10231311', '104200', 'B', '02-JUN-97',
6 USER, SYSDATE);

1 row created.
```

Sie können die Pseudospalte user nur einer Zeichenfolgenspalte, die Pseudospalte sysdate nur einer Datumsspalte zuweisen. Sie können diese Pseudospalten verändern, indem Sie Funktionen und Operatoren anwenden. In Listing 6.10 wird ein Zeichenfolgenliteral mit dem Namen des momentanen Benutzers verbunden und in einer Zeichenfolgenspalte gespeichert - Created Username.

**Listing 6.10: Verkettung einer Pseudospalte mit einem Zeichenfolgenliteral.**

```
SQL> insert into Student_Schedule
2 (Student_ID, Class_ID, Grade, Date_Grade_Assigned,
```



```
3 Created_Username, Created_Timestamp)
4 values
5 ('10231311', '104500', 'B-', '03-JUN-97',
6 'User is: ' || USER, SYSDATE);
```

1 row created.

```
SQL> select Student_ID, Grade, Created_Username, Created_Timestamp
2 from Student_Schedule
3 where
4 Student_ID = '10231311' and
5 Class_ID = '104500';
```

```
STUDENT_ID GR CREATED_USERNAME CREATED_T
-----
10231311 B- User is: TYO 06-MAR97
```



*Die insert-Anweisung gibt zwei Pseudospalten an: user und sysdate. Die select-Anweisung zeigt, daß user und sysdate die erwarteten Werte aufweisen.*

## Warum Spalten in einer insert-Anweisung angegeben werden sollten

Wenn Sie das Syntaxdiagramm für die insert-Anweisung in der Online-Sprachreferenz für SQL von Oracle genau betrachten, sehen Sie, daß die Spaltenliste ein optionales Element ist. Darum verwendet Oracle per Voreinstellung alle Spalten, wenn Sie die Namen der Spalten, denen Werte zugewiesen werden sollen, nicht angeben. Außerdem ist die Reihenfolge der Spalten, die Oracle verwendet, die Reihenfolge, in der die Spalten beim Erzeugen der Tabelle angegeben wurden; diese Reihenfolge ist dieselbe, die Sie sehen, wenn Sie in SQL\*Plus den Befehl describe auf eine Tabelle anwenden.

Wie in Listing 6.11 gezeigt wird, erwartet Oracle, daß die erste Spalte das Gebäude mit dem Klassenraum enthält, wenn der Spaltenname nicht in der insert-Anweisung angegeben wird.

**Listing 6.11: Probleme, die auftreten, wenn Spaltennamen in einer insert-Anweisung ausgelassen werden.**

```
SQL> desc class_location
Name Null? Type
-----
```

```
CLASS_BUILDING NOT NULL VARCHAR2(25)
CLASS_ROOM NOT NULL VARCHAR2(25)
SEATING_CAPACITY NUMBER(38)
```

```
SQL> insert into Class_Location
2 values
3 ('250', 'MACLAREN COMPLEX', 500);
```

1 row created.

```
SQL> select Class_Building, Class_Room, Seating_Capacity
2 from Class_Location
3 where
4 Class_Building = '250';
```

```
CLASS_BUILDING CLASS_ROOM SEATING_CAPACITY
-----
250 MACLAREN COMPLEX 500
```

Weil die Spaltennamen in der insert-Anweisung nicht angegeben wurden, sieht der Wert, der in Class\_Room eingefügt werden sollte, 250, einwandfrei aus - die Zeile wurde erfolgreich in die Tabelle eingefügt. Diese Verwendung der Syntax kann jedoch gefährlich sein.

- Die Definition der Tabelle kann sich ändern; die Anzahl der Spalten kann ab- oder zunehmen; als Folge davon schlägt das insert fehl.
- Die insert-Anweisung kann schwierig zu verstehen sein. Die Werte können nicht voneinander unterscheidbar sein oder aus Versehen verschoben werden. Diese Probleme zurückzuverfolgen, wenn man nicht in der Lage ist, die Spalten visuell mit den entsprechenden Werten abzustimmen, ist schwierig.
- Die insert-Anweisung kann erfolgreich sein, es können jedoch die falschen Daten in die Tabelle eingefügt werden. Diese Situation kann auftreten, wenn zwei benachbarte Werte mit demselben Datentyp aus Versehen verschoben werden. Dieses Szenario ist die größte der potentiellen Gefahren, weil Ihnen Oracle eventuell keine Fehlermeldung zurückgibt, die darauf hinweisen könnte, daß etwas schiefgelaufen ist.

## Eine Unterabfrage in einer insert-Anweisung verwenden

Bis jetzt haben die Beispiele gezeigt, wie jede Ausführung der insert-Anweisung eine einzelne Zeile zu einer Tabelle hinzufügen kann. Die folgenden Beispiele zeigen, wie man ein insert durchführt, ohne Literalwerte anzugeben.

## Die Syntax der insert-Anweisung mit einer Unterabfrage



*Die alternative Form der insert-Anweisung ersetzt die Liste der Spaltenwerte auf folgende Weise mit Hilfe einer select-Anweisung:*

```
INSERT INTO table_name  
[(column_name, [,column_name] . . . [,column_name])]  
select_statement
```

Die Variablen sind folgendermaßen definiert:

- *table\_name* ist die Tabelle, in die die Zeile eingefügt werden soll.
- *column\_name* ist die zu *table\_name* gehörende Spalte.
- *select\_statement* ist eine zulässige select-Anweisung.

Nehmen wir an, daß eine Tabelle verwendet wird, um Informationen über Lehrer zu speichern, die nicht mehr am Flugle-College unterrichten. Im Rahmen der Feier zum hundertjährigen Bestehen des Flugle-College werden alle früheren Lehrer geehrt, indem man sie zu aktiven Lehrern macht.



*Beachten Sie bitte, daß die Tabelle Inactive\_Instructors in der Beispieldatenbank nicht vorhanden ist. Um die nichtaktiven Lehrer zur Instructor-Tabelle hinzuzufügen, verwenden Sie eine Unterabfrage, um die Reihen der Tabelle Inactive\_Instructor auszuwählen und sie in die Tabelle Instructor einzufügen, wie dies in Listing 6.12 gezeigt wird.*

**Listing 6.12: Einfügen von Zeilen mit einer Unterabfrage.**

```
SQL> insert into Instructor  
2 (Instructor_ID, Last_Name, First_Name, Department_ID)  
3 select Instructor_ID, Last_Name, First_Name, Department_ID  
4 from Inactive_Instructor;  
5 rows created.
```

Um eine Unterabfrage mit einer insert-Anweisung zu verwenden, muß die Zahl der in der insert-

Anweisung angegebenen Spalten der Anzahl der Elemente in der Auswahlliste entsprechen.

## Erzeugen von Testdaten

Wenn Sie eine Datenbankanwendung erzeugen, benötigen Sie Daten, um die Software zu testen. Eine größere Menge von Testdaten zu erzeugen, kann eine schwierige Aufgabe sein. Zum Glück können Sie die insert-Anweisung verwenden, um die Testdaten zu duplizieren und sie so zu vergrößern.

Listing 6.13 zeigt, wie Sie eine Unterabfrage verwenden können, um die in einer Tabelle vorhandenen Zeilen in dieselbe Tabelle zu kopieren. Nehmen Sie an, daß die Tabelle Instructor ursprünglich 17 Zeilen enthält. Wenn Sie ein insert mit einer Unterabfrage durchführen, die diese 17 Zeilen liefert, wächst Ihre Tabelle auf 34 Zeilen. Natürlich können Sie dieses Einfügen nur durchführen, wenn Sie den Primärschlüssel zeitweise löschen. Wenn Sie dasselbe insert ein weiteres Mal durchführen, hat die entstehende Tabelle 68 Zeilen. Wie Sie sehen können, verdoppelt sich die Zahl der Zeilen jedesmal, wenn Sie ein insert durchführen.

***Listing 6.13: Einfügen von Zeilen in eine Tabelle, indem die bereits vorhandenen Zeilen in dieselbe Tabelle kopiert werden.***

```
SQL> insert into Instructor
2 (Instructor_ID, Department_ID, Last_Name)
3 select Instructor_ID, Department_ID, Last_Name
4 from Instructor;

17 rows created.
```



Wenn Sie die Kopiertechnik verwenden, um Testdaten zu erzeugen, wächst die Zahl der Zeilen exponentiell. Wenn Sie mit 100 Zeilen beginnen, enthält die Tabelle 12.800 Zeilen, wenn insert siebenmal durchgeführt wurde. Wenn Sie nicht nach jedem insert ein commit durchführen, kann es sein, daß die Rollback-Segmente die nicht festgeschriebenen Transaktionen nicht speichern können und Oracle den Fehlercode ora-01653 zurückgibt; der Tablespace, in dem die Rollback-Segmente versucht haben, mehr Speicherplatz zu belegen, wird in der Fehlermeldung angegeben.

---

---



*Die commit-Anweisung wird verwendet, um alle Veränderungen dauerhaft zu speichern, die der Benutzer seit der letzten commit-Anweisung oder seit dem Beginn der Arbeitssitzung an der Datenbank vorgenommen hat, je nachdem, was weniger weit zurückliegt.*

---



*Die rollback-Anweisung wird verwendet, um alle Veränderungen rückgängig zu machen, die der Benutzer seit der letzten commit-Anweisung oder seit dem Beginn der Arbeitssitzung an der Datenbank vorgenommen hat, je nachdem, was weniger weit zurückliegt.*

---

## Ändern der Daten mit Hilfe der update-Anweisung

Wenn Sie vorhandene Daten in einer Oracle-Datenbank verändern wollen, müssen Sie die update-Anweisung verwenden. Mit dieser Anwendung können Sie eine oder mehrere Zeilen in einer Tabelle aktualisieren.

### Grundlegende Syntax der update-Anweisung

---



*Wie bei der insert-Anweisung ist die Syntax der update-Anweisung wesentlich einfacher, als die Syntax der select-Anweisung.*

---

Die update-Anweisung hat die folgende Syntax:

```
UPDATE table_name
SET column_name = expression [, column = expression]...
[, column = expression]
[WHERE condition]
```

Die Variablen sind folgendermaßen definiert:

- *table\_name* ist die Tabelle, die aktualisiert werden soll.
- *column\_name* ist eine Spalte in der zu aktualisierenden Tabelle.
- *expression* ist ein gültiger SQL-Ausdruck.
- *condition* ist eine gültige SQL-Bedingung.

Wie Sie sehen können, verweist die update-Anweisung auf eine einzelne Tabelle und weist mindestens einer Spalte einen Ausdruck zu. Die where-Klausel ist optional; wenn eine update-Anweisung keine where-Klausel enthält, wird die Zuweisung eines Werts an eine Spalte für alle Zeilen der Tabelle vorgenommen.

## Ändern der Werte von mehreren Spalten

Wie die Syntax der update-Anweisung zeigt, kann eine update-Anweisung den Wert für mehr als eine Spalte in einer Tabelle ändern. Listing 16.4 zeigt, wie eine update-Anweisung Werte an zwei Spalten zuweist: Position und Telephone.

**Listing 6.14: Änderung des Werts in mehreren Spalten durch eine update-Anweisung.**

```
SQL> update Instructor
2 set
3 Position = 'PROFESSOR',
4 Telephone = '8055551212';
```

17 rows updated.

## In Mengen denken, nicht in Datensätzen

Ein Weg, zu demonstrieren, daß SQL mengenorientiert ist, ist eine update-Anweisung, die einen Wert zwischen zwei Spalten vertauscht.

Listing 6.15 enthält eine Abfrage, die die aktuellen Werte für Instructor\_ID, Telephone und Fax aus der Tabelle Instructor auswählt. Sie können die Telefon- und Faxnummern in der Tabelle Instructor mit einer einzigen update-Anweisung vertauschen. Sie müssen die Telefon- und Faxnummern nicht in temporären Variablen speichern, wie Sie das tun müßten, wenn Sie eine konventionelle Programmiersprache verwenden würden, um diese Spalten zu vertauschen. Die zweite Abfrage der Tabelle Instructor zeigt, daß das Vertauschen der Spalten erfolgreich war.

**Listing 6.15: Veranschaulichung der Ausrichtung von SQL auf Mengen.**

```
SQL> select Instructor_ID, Telephone, Fax
2 from Instructor
3 order by Instructor_ID;
```

```
INSTRUCTOR_ID TELEPHONE FAX
-----
A612
B331
B391
D201 8055550131 8055559444
D944
E301
E405 8055554455
E491
G331
J505
L391
M101 8055550123 8055550321
R983
S131
T149
W490
Y561 8055550123
```

17 rows selected.

```
SQL> update Instructor
2 set
3 Telephone = Fax,
4 Fax = Telephone;
```

17 rows updated.

```
SQL> select Instructor_ID, Telephone, Fax
2 from Instructor
3 order by Instructor_ID;
```

```
INSTRUCTOR_ID TELEPHONE FAX
-----
A612
B331
B391
D201 8055559444 8055550131
D944
E301
E405 8055554455
E491
```

```
G331
J505
L391
M101 8055550321 8055550123
R983
S131
T149
W490
Y561 8055550123
```

17 rows selected.

## Eine Unterabfrage in einer update-Anweisung verwenden

Bis jetzt wurde in diesem Kapitel gezeigt, wie man Bedingungen in einer where-Klausel verwendet, um die Zahl der von einer update-Anweisung betroffenen Spalten einzuschränken. Nun sehen wir, wie eine Unterabfrage mit der update-Anweisung verwendet wird.

Stellen Sie sich beispielsweise vor, Sie wollten die zusätzlichen Gebühren für einen Kurs um 9.- DM vermindern, wenn die zusätzliche Gebühr den Durchschnitt der zusätzlichen Gebühren übersteigt. Dies kann, wie in Listing 6.16 gezeigt, mit einer einzigen update-Anweisung geschehen. Beachten Sie, daß die Unterabfrage in Klammern eingeschlossen werden muß.

*Listing 6.16: Verwendung einer Unterabfrage in einer update-Anweisung.*

```
SQL> select Department_ID, Course_ID, Additional_Fees
2 from Course
3 order by Department_ID, Course_ID;
```

```
DEPART  COURS  ADDITIONAL_FEES
-----
ANTHRO  101    0
ANTHRO  174   55
ANTHRO  189   7.5
BIO     101   55
BIO     177   65
BIO     178   70
ECON    101   25
ECON    189  750
ECON    199    0
ENG     101   75
ENG     102   45
ENG     103   35
```



```
ENG 199 45
ENGL 101 0
ENGL 189 0
ENGL 192 0
ENGL 193 0
HIST 115 0
HIST 184 0
HIST 199 0
MATH 101 0
MATH 189 0
MATH 50 0
MATH 51 10
PHILO 101 0
PHILO 174 0
PHILO 198 0
PSYCH 101 25
PSYCH 181 75
PSYCH 183 45
PSYCH 185 20
PSYCH 501 75
```

32 rows selected.

```
SQL> update Course
2 set Additional_Fees = Additional_Fees - 10
3 where
4 Additional_Fees >
5 (select avg(Additional_Fees) from Course);
```

8 rows updated.

```
SQL> select Department_ID, Course_ID, Additional_Fees
2 from Course
3 order by Department_ID, Course_ID;
```

```
DEPART COURS ADDITIONAL_FEES
-----
ANTHRO 101 0
ANTHRO 174 45
ANTHRO 189 7.5
BIO 101 45
BIO 177 55
BIO 178 60
ECON 101 25
ECON 189 740
ECON 199 0
ENG 101 65
```

```
ENG 102 45
ENG 103 35
ENG 199 45
ENGL 101 0
ENGL 189 0
ENGL 192 0
ENGL 193 0
HIST 115 0
HIST 184 0
HIST 199 0
MATH 101 0
MATH 189 0
MATH 50 0
MATH 51 10
PHILO 101 0
PHILO 174 0
PHILO 198 0
PSYCH 101 25
PSYCH 181 65
PSYCH 183 45
PSYCH 185 20
PSYCH 501 65
```

32 rows selected.

## Löschen von Daten mit Hilfe der delete-Anweisung

Die delete-Anweisung entfernt Zeilen aus einer Tabelle. Sie müssen die physikalische Ordnung der Zeilen in einer Tabelle nicht kennen, um ein delete auszuführen. Oracle verwendet die Kriterien in der where-Klausel, um zu bestimmen, welche Zeilen gelöscht werden sollen; der Datenbanker von Oracle bestimmt die interne Speicherstelle der Zeilen.

### Die Syntax der delete-Anweisung

---



*Die delete-Anweisung hat die einfachste Syntax der vier DML-Anweisungen:*

---

```
DELETE FROM table_name
[WHERE condition]
```

Die Variablen sind folgendermaßen definiert:

- *table\_name* ist die Tabelle, die aktualisiert werden soll.
- *condition* ist eine gültige SQL-Bedingung.

Wenn Sie denken, daß die Syntax von SQL inkonsistent ist, haben Sie recht. Beispielsweise unterscheidet sich die Syntax der update-Anweisung (update *table\_name*) von der Syntax der delete-Anweisung (delete from *table\_name*). SQL hat viele andere Eigenarten und sie werden so bald nicht verschwinden. Wenn Sie die Leistungsfähigkeit von SQL ausnutzen wollen, konzentrieren Sie sich darauf, die Syntax zu erlernen, und arbeiten Sie viele Beispiele durch. Listing 6.17 zeigt ein Beispiel für eine einfache delete-Anweisung.

***Listing 6.17: Beispiel für eine delete-Anweisung.***

```
SQL> delete from Class
2 where
3 Department_ID = 'BIO';

1 row deleted.
```



*Eine delete-Anweisung wird auch erfolgreich ausgeführt, wenn keine Zeilen aus der Tabelle gelöscht werden.*

---

```
SQL> delete from Class
2 where
3 Department_ID = 'ANTHRO';

0 rows deleted
```

## Löschen aller Zeilen mit Hilfe der truncate table-Anweisung

Wenn Sie eine Anwendung entwerfen, kann es sein, daß Sie alle Zeilen in einer Tabelle entfernen

müssen. Wenn die Tabelle viele Zeilen hat, kann es ziemlich ineffizient sein, diese Aufgabe mit der delete-Anweisung zu erledigen. Als Alternative sollten Sie die truncate table-Anweisung verwenden. Die Anweisung truncate table löscht Zeilen viel schneller als die delete-Anweisung

---



*Die truncate table-Anweisung wird typischerweise folgendermaßen verwendet:*

---

`TRUNCATE_TABLE table_name`

Die Variable ist folgendermaßen definiert

- *table\_name* ist die Tabelle, deren Inhalt gelöscht werden soll.

Vorsicht: Die truncate table-Anweisung ist keine DML-Anweisung. Wenn Sie eine truncate table-Anweisung abschicken, können Sie nicht Ihre Meinung ändern und einen Rollback ausführen, um die verlorenen Zeilen zurückzubekommen; die truncate table-Anweisung ist eine Einbahnstraße.

## Gleichzeitigkeit

Ein ideales Informationssystem bietet gleichzeitigen Zugang für mehrere Benutzer. Natürlich wollen Sie auch sicher sein, daß ein Benutzer nicht die Änderungen eines anderen Benutzers überschreiben kann. Wenn beispielsweise ein Administrator im Department für Anthropologie die Beschreibung eines Kurses ändert, sollte ein anderer Administrator nicht in der Lage sein, irgendeine Information über denselben Kurs zu ändern, bis der erste Administrator seine Änderungen festgeschrieben hat.

Andererseits sollte die Tatsache, daß ein Administrator in einer Abteilung die Informationen über die von dieser Abteilung angebotenen Kurse ändert, niemanden daran hindern, die Informationen über diese Kurse zu betrachten oder Informationen über einen anderen Kurs zu ändern.

## Lesekonsistenz und Nur-Lese-Transaktionen

In einer Oracle-Umgebung mit mehreren Benutzern bietet Oracle auf der Ebene der SQL-Anweisungen das, was man »Lesekonsistenz« nennt. *Lesekonsistenz* bedeutet, daß eine einzelne SQL-Anweisung keine Ergebnisse zurückliefern kann, die widersprüchlich oder inkonsistent sind. Listing 6.18 zeigt ein Beispiel für dieses Konzept.

Nehmen wir an, daß der oder die Vorsitzende der Abteilung für Biologie die Anzahl der Einheiten für jeden der von dieser Abteilung angebotenen Kurse wissen will.

***Listing 6.18: Beispiel für Lesekonsistenz.***

```
SQL> select Course_ID, Title, Units
2 from Course
3 where
4 Department_ID = 'BIO'
5 order by Course_ID;
```

```
COURSE_ID TITLE UNITS
-----
101 INTRO TO BIOLOGY 3
177 INVERTEBRATE ANATOMY 3
178 MAMMALIAN ANATOMY 3
```

Direkt nachdem die Abfrage des Vorsitzenden an die Datenbank abgeschickt wurde, änderte der Archivar des College alle Kurse mit drei Einheiten auf vier Einheiten. Konsistenz auf der Ebene der Anweisungen bedeutet, daß die Abfrage keine Zeile zurückgeben kann, in der die Zahl der Einheiten auf vier aktualisiert wurde. Je nachdem, wann der Archivar des College seine Änderungen festschreibt, sieht sie der Vorsitzende der Abteilung für Biologie oder er sieht sie nicht. Er wird jedoch in einer einzelnen SQL-Anweisung niemals eine teilweise durchgeführte Änderung sehen.

Obwohl Oracle Konsistenz innerhalb einer einzelnen SQL-Anweisung bietet, garantiert sein Standardverhalten keine Lesekonsistenz über mehr als eine Anweisung. Wenn Sie eine Tabelle zweimal abfragen, kann es sein, daß Sie beim zweiten Mal andere Resultate erhalten, wenn ein anderer Oracle-Benutzer die Tabelle zwischen der ersten und der zweiten Abfrage ändert.

Sie können in eine Situation kommen, in der Sie Lesekonsistenz für mehr als eine Anweisung benötigen. Es kann in der Tat sein, daß Sie Lesekonsistenz für eine bestimmte Transaktion brauchen. Zu diesem Zweck geben Sie die folgende Anweisung ein:

```
set transaction read only;
```

## Einen Spaltenwert auf null setzen

Hier folgen die verschiedenen Methoden, mit denen der Wert einer Spalte auf null gesetzt werden kann:

- Die Spalte bekommt während eines insert keinen Wert zugewiesen, und sie hat keinen Standardwert.

- Die Spalte bekommt während eines insert explizit den Wert null zugewiesen.
- Die Spalte wird bei einem update explizit auf null gesetzt.

## Zuweisen von null in einer insert-Anweisung

Sie können eine Spalte in einer insert-Anweisung explizit auf null setzen, wie in Listing 6.19 gezeigt wird.

*Listing 6.19: Ein Spaltenwert wird in einer insert-Anweisung auf null gesetzt.*

```
SQL> insert into Instructor
2 (Instructor_ID, Department_ID, Last_Name, Telephone)
3 values
4 ('P331', 'ANTHRO', 'POULSON', NULL);

1 row created.
```

## Einen Spaltenwert mit Hilfe einer update-Anweisung auf null setzen

Sie können die update-Anweisung verwenden, um den Wert einer Spalte auf null zu setzen. Listing 6.20 zeigt ein Beispiel. Nachdem Sie die Telefonnummer für den Lehrer mit der ID Y561 auf null gesetzt haben, können Sie überprüfen, daß Oracle die Änderung in der Tat ausführt.

*Listing 6.20: Ein Spaltenwert wird in einer update-Anweisung auf null gesetzt.*

```
SQL> select Last_Name, Telephone
2 from Instructor
3 where
4 Instructor_ID = 'Y561';
```

```
LAST_NAME TELEPHONE
-----
YOUNG 8055550123
```

```
SQL> update Instructor
2 set
3 Telephone = NULL
4 where
5 Instructor_ID = 'Y561';
```

1 row updated.

```
SQL> select Last_Name, Telephone
2 from Instructor
3 where
4 Instructor_ID = 'Y561';
```

```
LAST_NAME TELEPHONE
-----
YOUNG
```

Einer der Kritikpunkte an SQL ist seine inkonsistente Syntax. In der update-Anweisung wird der Spalte mit Hilfe eines Gleichheitszeichens (=) der Wert null zugewiesen. In der where-Klausel der select-Anweisung wird jedoch das Wort is (oder is not) anstatt eines Gleichheitszeichens verwendet, um eine Spalte auf einen null-Wert zu testen. Auf diese Weise können inkonsistente Anweisungen wie die in Listing 6.21 gezeigten entstehen.

*Listing 6.21: Veranschaulichung verwirrender SQL-Syntax.*

```
SQL> update Student
2 set Fax = NULL
3 where
4 Fax is not null;
```

3 rows updated.

## Standardwerte und null

Wenn Sie eine Tabelle erzeugen, können Sie eine Spalte als erforderlich angeben, indem Sie nach dem Datentyp der Spalte not null angeben. Eine erforderliche Spalte muß jedesmal, wenn eine Zeile in die Tabelle eingefügt wird, einen Wert zugewiesen bekommen. Wenn Sie eine Zeile einfügen, ohne einen Wert für eine erforderliche Spalte anzugeben, gibt Oracle eine Fehlermeldung zurück. Beispielsweise muß jeder Kurs am Flugle-College eine Kurs-ID haben, einer bestimmten Abteilung zugewiesen sein und einen Titel aufweisen. Listing 6.22 enthält eine insert-Anweisung, die versucht, eine Zeile in die Tabelle Course einzufügen, ohne einen Wert für die Spalte Title anzugeben, diese ist als not null definiert.

*Listing 6.22: In einer insert-Anweisung ist für eine erforderliche Spalte kein Wert angegeben.*

```
SQL> insert into Course
2 (Course_ID, Department_ID)
3 values
4 ('901', 'ANTHRO');
```

```
insert into Course
```

```
*
```

```
ERROR at line 1:
```

```
ORA-01400: mandatory (NOT NULL) column is missing or NULL during  
insert
```

Unglücklicherweise gibt Oracle die erforderliche Spalte (oder die erforderlichen Spalten) nicht an, auf die in der insert-Anweisung verwiesen werden muß. Wenn Sie den Fehlercode ora-01400 sehen, müssen Sie die Liste der Spalten in der insert-Anweisung mit der Definition der Tabelle vergleichen.

## Eine Transaktion ist eine logische Arbeitseinheit

Ein anderes leistungsfähiges Konzept, das Sie beherrschen müssen, ist die Transaktion.



*Eine Transaktion ist definiert als eine logische Arbeitseinheit - eine Menge von Veränderungen an einer oder mehreren Tabellen der Datenbank, mit denen eine bestimmte Aufgabe durchgeführt werden soll.*

Eine Transaktion beginnt nach einer commit-Anweisung, einer rollback-Anweisung oder mit der anfänglichen Verbindung zu Oracle. Eine Transaktion endet, wenn eines der folgenden Ereignisse auftritt:

- Eine commit-Anweisung wird ausgeführt.
- Eine rollback-Anweisung wird ausgeführt.
- Die Verbindung zu Oracle wird beendet.

Nehmen Sie beispielsweise an, daß das Flugle-College die Einheiten für alle Kurse um 50 Prozent anheben will. Dies kann mit einer einzigen update-Anweisung erreicht werden, diese betrifft alle Zeilen in der Tabelle Course.

## Sichern der Arbeit mit Hilfe der commit-Anweisung

Sie können sich eine Transaktion wie eine Veränderung an einem Dokument vorstellen, die Sie mit Ihrer bevorzugten Textverarbeitung vornehmen. Sie können mehrere Änderungen vornehmen und



diese dann rückgängig machen oder das Programm verlassen, ohne die Änderungen zu speichern. Wenn Sie die Textverarbeitung anweisen, die Datei zu speichern, verändern Sie die auf der Festplatte gespeicherte Datei dauerhaft.

Das Festschreiben einer Transaktion ähnelt dem Speichern einer Datei im Microsoft Word. Die commit-Anweisung schreibt eine Transaktion fest. Die commit-Anweisung macht alle Änderungen, die an der Datenbank seit dem letzten commit (oder rollback) erfolgt sind, dauerhaft. Sie können ein commit nur für die Änderungen an der Datenbank verwenden, die Sie selbst vorgenommen haben; die commit-Anweisung, die ein Benutzer eingibt, hat keine Auswirkungen auf Änderungen durch andere Benutzer.

## Rückgängig machen von Änderungen mit Hilfe der rollback-Anweisung

In ähnlicher Weise arbeitet eine rollback-Anweisung wie der Befehl »Rückgängig machen« bei einer Textverarbeitung, mit einer großen Ausnahme: Die rollback-Anweisung macht alle Änderungen an der Datenbank rückgängig, die der Benutzer seit der letzten festgeschriebenen Änderung oder seit dem Beginn der Sitzung vorgenommen hat.

Betrachten Sie das Wechselspiel zwischen commit und rollback. Listing 6.23 illustriert dieses Konzept. Eine einfache Abfrage aus table\_1 gibt vier Zeilen zurück. Wenn Sie die vier Zeilen aus table\_1 löschen und nach dem Löschen eine Abfrage durchführen, gibt diese Abfrage keine Zeilen zurück. Wenn Sie ein rollback durchführen und die Tabelle erneut abfragen, sehen Sie, daß die Tabelle zu dem Zustand wiederhergestellt wurde, den Sie vor der delete-Anweisung hatte.

### *Listing 6.23: Veranschaulichung von commit und rollback.*

```
SQL> select * from table_1;
TABLE_1_COL
-----
99
99
99
99
SQL> delete from table_1;
4 rows deleted.
SQL> select * from table_1;
no rows selected

SQL> rollback;
Rollback complete.
SQL> select * from table_1;
TABLE_1_COL
-----
99
```

99  
99  
99



Oracle führt für DDL-Anweisungen wie `create table` ein automatisches Festschreiben durch. Alle Änderungen, die Sie nach einer DDL-Anweisung an einer Datenbank vornehmen, werden ebenfalls automatisch festgeschrieben. Eine Tabelle, die mit einer `create table`-Anweisung erzeugt wurde, wird durch eine `rollback`-Anweisung nicht gelöscht. Wenn Sie eine Tabelle löschen wollen, müssen Sie die Anweisung `drop table` verwenden.

---

## Savepoints

Für Transaktionen, die die Ausführung mehrerer SQL-Anweisungen umfassen, können Sie *Savepoints* als Zwischenschritte für die Transaktion verwenden. Ein Savepoint ist eine Markierung innerhalb einer Transaktion, die eine Untermenge der bei der Transaktion vorgenommenen Änderungen enthält. Listing 6.24 zeigt, wie ein Savepoint deklariert wird; er bekommt den Namen `null_fax_numbers`.

---



Sie können sich einen Savepoint als eine Markierung innerhalb einer Transaktion vorstellen, die auf eine Untermenge der in der Transaktion vorgenommenen Änderungen verweist.

---

### *Listing 6.24: Vereinbarung eines Savepoints.*

```
SQL> savepoint null_fax_numbers;  
Savepoint created.
```

Sie können einen Savepoint mit einer `rollback`-Anweisung verwenden; der Savepoint gibt Ihnen die Option, eine Transaktion bis zu einem Zwischenpunkt (einem Savepoint) rückgängig zu machen.

---



*Die Syntax der rollback-Anweisung ist:*

---

```
ROLLBACK [TO savepoint];
```

Die Variable ist folgendermaßen definiert:

- *savepoint* ist ein vorher benannter Savepoint.

Betrachten Sie das in Listing 6.25 gezeigte Beispiel. Stellen Sie sich vor, daß Ihre Anwendung eine Transaktion enthält, die drei Tabellen aktualisiert: table\_1, table\_2 und table\_3. Wenn Oracle beim Aktualisieren von table\_2 einen Fehler zurückgibt, können Sie alles bis zum ersten Savepoint rückgängig machen, dieser ist table\_1\_update.

***Listing 6.25: Beispiel für die Verwendung eines Savepoints.***

```
SQL> update table_1
2 set table_1_col = 11;
4 rows updated.

SQL> savepoint table_1_update;
Savepoint created.

SQL> delete from table_2;
3 rows deleted.

SQL> rollback to table_1_update;
Rollback complete.

SQL> select * from table_2;
TABLE_2_COL
-----
99
99
99
```

Sie sollten Savepoints jedoch mit Vorsicht verwenden, weil eine Anwendung um eine weitere Ebene komplexer machen. Stellen Sie sicher, daß Ihre Transaktionen wohldefiniert sind, bevor Sie sich dafür entscheiden, Savepoints zu implementieren.

## Zusammenfassung

Die heutige Lektion hat die Verwendung der folgenden grundlegenden SQL-Anweisungen vorgestellt:

- insert fügt eine Zeile zu einer Tabelle hinzu.
- update modifiziert bestehende Zeilen.
- delete löscht Zeilen aus einer Tabelle.

In Verbindung mit diesen Anweisungen können Sie auch Unterabfragen verwenden.

Die truncate-Anweisung löscht alle Zeilen einer Tabelle.

Eine Datenbanktransaktion ist eine Menge von Veränderungen an einer oder mehreren Tabellen einer Datenbank, die eine logische Arbeitseinheit darstellen. Mit Hilfe der commit-Anweisung wird eine Transaktion dauerhaft gemacht. Alternativ können Sie eine rollback-Anweisung verwenden, um eine Transaktion rückgängig zu machen.

## Wie geht es weiter?

An Tag 7, »Die in SQL integrierten Funktionen ausnutzen«, lernen Sie, wie man eine Vielzahl von Funktionen in den vier SQL-Anweisungen - select, insert, update und delete - verwendet.

## Fragen und Antworten

Frage:

Welche Syntax kennzeichnet den Anfang einer Transaktion?

Antwort:

*Normalerweise keine. Normalerweise beginnt eine Transaktion mit einer beliebigen Änderung an einer Tabelle seit der letzten bestätigten Transaktion oder seit dem Beginn der Datenbanksitzung. Sie müssen auf jeden Fall mit **SET TRANSACTION READ ONLY** den Anfang einer Read-Only-Transaktion kennzeichnen.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser

Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Konstruieren Sie eine SQL-Anweisung, die einen Kurs mit den folgenden Eigenschaften hinzufügt: Department\_ID = BIO, Course\_ID = 137, Title = INSECT BEHAVIOR, Description = In-depth study of insect societies and their behavior patterns, Units = 3, keine zusätzlichen Gebühren.
2. Konstruieren Sie eine SQL-Anweisung, die zusätzliche Gebühren in Höhe von \$50 für alle Kurse in der Abteilung Philosophie einträgt.
3. Konstruieren Sie eine SQL-Anweisung, die eine geplante Klasse löscht, wenn diese von der Abteilung für Englisch angeboten wird oder in Flugle-Hall abgehalten werden soll.

## Übungen

Einige der Lehrer in Flugle-College haben sich entschieden, eine neue Abteilung namens Integrated Studies einzurichten. Als Folge davon werden die Abteilungen für Englisch, Geschichte und Philosophie zusammengelegt, um die Abteilung für Integrated Studies zu bilden. Die Department ID für diese neue Abteilung wird intstd sein. Erzeugen Sie in der Datenbank eine Abteilung für Integrated Studies (ohne die existierenden Abteilungen zu löschen). Verändern Sie zudem die Tabelle Instructor, so daß die Lehrer in den Abteilungen für Englisch, Geschichte und Philosophie der Abteilung für Integrated Studies zugewiesen werden.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

# 7

# In SQL integrierte Funktionen nutzen

[Bearbeiten von Zeichenfolgen](#)

[Bearbeiten von Datumsangaben](#)

[Festlegen einer numerischen Spalte](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

Diese Lektion untersucht die Techniken, die Oracle für die Bearbeitung von Zeichenfolgen, Datumsangaben und Zahlen zur Verfügung stellt. Diese Techniken sind in den folgenden Situationen nützlich:

- Beim Übertragen von Zeichenfolgen innerhalb einer Datenbank Anwendung von einer Form in eine andere
- Beim Erzeugen von Berichten
- Beim Durchführen von Ad-hoc-Abfragen
- Beim Konvertieren von Daten von einer Datenbank in eine andere

Oracle bietet eine große Menge integrierter Funktionen und Operatoren für die Umwandlung und Bearbeitung von Zeichenfolgen, Datumsangaben und Zahlen. Der volle Leistungsumfang dieser Funktionen und Operatoren wird wirksam, wenn man sie ineinander verschachtelt. Diese Lektion bietet mehrere Beispiele für diese Technik.

## Bearbeiten von Zeichenfolgen

Oracle bietet mehrere nützliche integrierte Funktionen, mit denen Zeichenfolgen bearbeitet werden können. Sie sehen auf den folgenden Seiten eine Reihe von Beispielen.

### Die Länge einer Zeichenfolge ermitteln

Sie können die `length`-Funktion verwenden, um die Länge einer Zeichenfolgenspalte zu ermitteln. `length` gibt die Zahl der Zeichen in ihrem Argument zurück, wie in Listing 7.1 gezeigt wird.

*Listing 7.1: Verwendung der Funktion `length`.*

```
SQL> select Last_Name, length(Last_Name)
3 order by Last_Name;
```

```
LAST_NAME  LENGTH(LAST_NAME)
-----
ANGELO      6
BATES       5
BILLINGS    8
CHANG       5
CHERNOW     7
CHU         3
DANIELS     7
EDWARDS     7
HITCHCOCK   9
JASON       5
NILAND      6
PARKER      6
POULSON     7
RESTON      6
RICHARDSON  10
TORRES      6
WEISS       5
YOUNG       5
```

18 rows selected.

## Eine Teilzeichenfolge aus einer Zeichenfolge gewinnen

Die Funktion substr extrahiert eine Teilzeichenfolge aus einer Zeichenfolge.



*Die Funktion substr wird folgendermaßen verwendet:*

---

SUBSTR (*string*, *starting character*, *number of characters*)

Die Variablen sind folgendermaßen definiert:

- *string* ist eine Zeichenfolgenspalte oder eine Zeichenfolge.
- *starting character* ist die Startposition der Teilzeichenfolge.
- *number of characters* ist die Anzahl der zu liefernden Zeichen.

Listing 7.2 zeigt, wie die Funktion substr verwendet wird, um die ersten vier Zeichen des Nachnamens eines Lehrers zurückzugeben.

### ***Listing 7.2: Verwendung der Funktion substr.***

```
SQL> select Last_Name, substr(Last_Name,1,4)
2 from Instructor
3 order by Last_Name;
```

```
LAST_NAME SUBS
```

```
-----
```

```
ANGELO ANGE
```

```
BATES BATE
```

```
BILLINGS BILL
```

```
CHANG CHAN
```

```
CHERNOW CHER
```

```
CHU CHU
```

```
DANIELS DAN
```

```
EDWARDS EDWA
```



HITCHCOCK HITC  
JASON JASO  
NILAND NILA  
PARKER PARK  
POULSON POUL  
RESTON REST  
RICHARDSON RICH  
TORRES TORR  
WEISS WEIS  
YOUNG YOUN

18 rows selected.

Zusätzlich zur Verwendung von Literalwerten können Sie als Argument der Funktion substr auch eine Funktion als Argument angeben. Listing 7.3 zeigt ein gutes, wenn auch nicht sehr nützliches Beispiel. Nehmen Sie an, daß Sie die drei letzten Zeichen des Nachnamens eines Lehrers erhalten wollen. Sie würden die Funktion length verwenden, um die Position des letzten Zeichens zu finden. Um das richtige Startzeichen für die Funktion substr zu finden, ziehen Sie  $n - 1$  von substr ab, wobei  $n$  die Zahl der Zeichen ist, die Sie erhalten wollen - in Listing 7.3 sind das  $3 - 1 = 2$ .

*Listing 7.3: Verwendung der Funktion length in einem Aufruf der Funktion substr.*

```
SQL> select substr>Last_Name,length>Last_Name)-2,3)
2 from Instructor
3 order by Last_Name;
```

SUB  
---  
ELO  
TES  
NGS  
ANG  
NOW  
CHU  
ELS  
RDS  
OCK  
SON  
AND  
KER  
SON  
TON  
SON  
RES  
ISS  
UNG

18 rows selected.

## Suche nach Mustern in einer Zeichenfolge

Sie haben an Tag 5, »Einführung in SQL« gelernt, wie man den like-Operator verwendet. Als kurzer Rückblick: Sie können den like-Operator verwenden, um Muster in einer Zeichenfolge zu suchen. Mit geschickt entworfenen Mustern können Sie sehr spezielle Suchvorgänge durchführen

Listing 7.4 zeigt, wie man den like-Operator verwendet, um alle Zeilen in einer Kursbeschreibung zu erhalten, die eines der Muster theory oder theories enthalten.

### *Listing 7.4: Verwendung des like-Operators.*

```
SQL> column description format a40 word_wrapped
SQL> column title format a35
SQL> select Title, Description
2 from Course
3 where
4 Description like '%theory%' or
5 Description like '%theories%';
```

TITLE DESCRIPTION

-----

NUMBER THEORY Introduction to number theory;  
characteristics of natural, rational,  
and irrational numbers; survey of  
lucky numbers

INTRO TO PSYCHOLOGY A general introduction to the theory of  
human psychology. Topics include  
Freudian theory, theory of behavior,  
psychological development of the child,  
and other topics.

INTRO TO ECONOMICS A general introduction to the "dismal"  
science of economics. Topics include  
supply and demand, the theory of money,  
theories of production, and other  
relevant topics.

INTRO TO STRUCTURES A general introduction to the theory of  
mechanical structures including levers,  
gears, beams, and others.

MODERN PHILOSOPHY A workshop on recent philosophical debates including bioethics, communication theory, social net theory, and philosophical relativism.

INTRO TO CIRCUIT THEORY A general introduction to the theory of circuits, logical design, boolean theory, and other topics.

INTRO TO DYNAMICS Introduction to the theory of dynamic systems, both linear and nonlinear.

SEMINAR ON CHAOS Seminar will explore examples of chaos theory as applied to economic, biological, and physical systems. Class schedule TBD.

EVOLUTIONARY GRAMMAR Seminar on current trends in English grammar. Topics include apostrophe theory and plurality.

WORKSHOP ON NEUROSES Intense workshop on the development and perfection of various neuroses. Exploration of neuroses-psychoses transition theories.

10 rows selected.

Beachten Sie bitte, daß die ersten beiden Zeilen in Listing 7.4 SQL\*Plus-Befehle sind. Sie wurden verwendet, um die Ausgabe der Abfrage für dieses Buch zu formatieren.

## Einen Teil einer Zeichenfolge ersetzen

Eine häufige Aufgabe bei der Bearbeitung von Daten ist, innerhalb einer Spalte ein Muster in ein anderes umzuwandeln. Stellen Sie sich vor, Sie wollten die Beschreibung der Kurse in der Tabelle Course so ändern, daß das Wort seminar durch das Wort discussion ersetzt wird.

Glücklicherweise bietet Oracle die Funktion replace, mit der der Inhalt einer Spalte verändert wird, indem man eine Zeichenfolge durch eine andere ersetzt.



*Die Syntax der Funktion replace ist*

---

`REPLACE (string, existing_string, [replacement_string])`

Die Variablen sind folgendermaßen definiert:

- *string* ist eine Zeichenfolge.
- *existing\_string* ist eine Zeichenfolge, die in *string* auftreten könnte.
- *replacement\_string* ist eine optionale Zeichenfolge, durch die *existing\_string* ersetzt werden soll.

Listing 7.5 zeigt, wie `replace` verwendet werden kann, um den Namen eines Kurses in der Tabelle `Course` zu ändern. Zuerst zeigt eine Abfrage die aktuellen Namen. Dann wird eine `update`-Anweisung mit der `replace`-Funktion verwendet, um jedes Auftreten von `seminar` in `discussion` zu ändern. Eine weitere Abfrage zeigt die Auswirkungen der `update`-Anweisung.

***Listing 7.5: Verwendung der Funktion replace.***

```
SQL> select Title
2 from Course
3 order by Title;
```

```
TITLE
-----
ABNORMAL PSYCHOLOGY
ADVANCED ARITHMETIC
ANCIENT GREEK PHILOSOPHERS
EARLY AMERICAN HISTORY
EVOLUTIONARY GRAMMAR
GENERAL CALCULUS
INTRO TO ANTHROPOLOGY
INTRO TO BIOLOGY
INTRO TO CIRCUIT THEORY
INTRO TO DYNAMICS
INTRO TO ECONOMICS
INTRO TO ENGLISH LIT
INTRO TO PHILOSOPHY
INTRO TO PSYCHOLOGY
INTRO TO STRUCTURES
INVERTEBRATE ANATOMY
MAMMALIAN ANATOMY
```

MODERN ENGLISH LIT  
MODERN EUROPEAN HISTORY  
MODERN PHILOSOPHY  
MONETARY POLICY  
NUMBER THEORY  
PHYS ANTHRO FIELD TECHNIQUES  
PRE-CALCULUS  
PSYCH IN FILM  
SEMINAR ON CHAOS  
SEMINAR ON NACIMERA  
SEMINAR ON THEME ANALYSIS  
WORKSHOP ON JEFFERSON  
WORKSHOP ON MARX  
WORKSHOP ON NEUROSES  
WORKSHOP ON NORMALITY

32 rows selected.

```
SQL> update Course
2 set Title = replace(Title, 'SEMINAR', 'DISCUSSION');
```

32 rows updated.

```
SQL> select Title
2 from Course
3 order by Title;
```

TITLE  
-----  
ABNORMAL PSYCHOLOGY  
ADVANCED ARITHMETIC  
ANCIENT GREEK PHILOSOPHERS  
DISCUSSION ON CHAOS  
DISCUSSION ON NACIMERA  
DISCUSSION ON THEME ANALYSIS  
EARLY AMERICAN HISTORY  
EVOLUTIONARY GRAMMAR  
GENERAL CALCULUS  
INTRO TO ANTHROPOLOGY  
INTRO TO BIOLOGY  
INTRO TO CIRCUIT THEORY  
INTRO TO DYNAMICS  
INTRO TO ECONOMICS  
INTRO TO ENGLISH LIT  
INTRO TO PHILOSOPHY  
INTRO TO PSYCHOLOGY  
INTRO TO STRUCTURES

INVERTEBRATE ANATOMY  
MAMMALIAN ANATOMY  
MODERN ENGLISH LIT  
MODERN EUROPEAN HISTORY  
MODERN PHILOSOPHY  
MONETARY POLICY  
NUMBER THEORY  
PHYS ANTHRO FIELD TECHNIQUES  
PRE-CALCULUS  
PSYCH IN FILM  
WORKSHOP ON JEFFERSON  
WORKSHOP ON MARX  
WORKSHOP ON NEUROSES  
WORKSHOP ON NORMALITY

32 rows selected.



*Wenn Sie in der Funktion replace keine Ersatzzeichenfolge angeben, wird die vorhandene Zeichenfolge aus der Spalte entfernt.*

---

## Entfernen führender und nachfolgender Leerzeichen in einer Zeichenfolge

Wenn eine Spalte mit Zeichen führende oder nachfolgende Leerzeichen enthält, kann eine Abfrage, die auf einem bestimmten Spaltenwert basiert, irreführende Resultate zurückgeben. Um dies zu verdeutlichen, werfen Sie einen Blick auf Listing 7.6. Zuerst wird eine update-Anweisung verwendet, um ein Leerzeichen an First\_Name anzufügen. Eine Abfrage der Tabelle Instructor nach dem Vornamen boris liefert keinen Datensatz, aber eine zweite Abfrage, die nach einem Vornamen sucht, der gleich 'boris' ist, zeigt, daß die Spalte geändert wurde.

### *Listing 7.6: Der Umgang mit nachfolgenden Leerzeichen.*

```
SQL> update Instructor
2 set First_Name = First_Name || ' ';
```

18 rows updated.

```
SQL> select Instructor_ID, Last_Name, First_Name
2 from Instructor
3 where
```

```
4 First_Name = 'BORIS';
```

no rows selected

```
SQL> select Instructor_ID, Last_Name, First_Name
2 from Instructor
3 where
4 First_Name = 'BORIS ';
```

```
INSTRUCTOR_ID LAST_NAME FIRST_NAME
-----
M101 HITCHCOCK BORIS
```

Wenn Sie nichts von den nachfolgenden Leerzeichen gewußt hätten, hätten Sie sich natürlich über die Tatsache gewundert, daß die Abfrage keine Zeile liefert.



*Um irreführende Ergebnisse bei einer Abfrage zu vermeiden, entfernen Sie führende oder nachfolgende Leerzeichen, bevor eine Zeile zu einer Tabelle hinzugefügt oder bearbeitet wird. Nachfolgende Leerzeichen sind hier häufiger ein Problem als führende Leerzeichen, weil sie weniger gut sichtbar sind.*

---

Oracle bietet zwei Funktionen, um Leerzeichen zu entfernen: [ltrim](#) und [rtrim](#). [ltrim](#) entfernt führende Leerzeichen einer Zeichenfolge, [rtrim](#) entfernt nachfolgende Leerzeichen.

Um führende und nachfolgende Leerzeichen aus einer Zeichenfolge zu entfernen, betten Sie einfach die Funktion [rtrim](#) in die Funktion [ltrim](#) ein. Wenn es keine führenden oder nachfolgenden Leerzeichen gibt, verändern [ltrim](#) und [rtrim](#) [die vorhandene Zeichenfolge nicht](#). Sehen Sie in [Listing 7.7](#), wie die Funktion [rtrim](#) verwendet wird.

***Listing 7.7: Abschneiden nachfolgender Leerzeichen mit Hilfe der Funktion rtrim.***

```
SQL> update Instructor
2 set First_Name = rtrim(First_Name)
3 where
4 First_Name like '% ';
```

18 rows updated.

Beachten Sie, wie in Listing 7.7 die where-Klausel verwendet wird, um nur die Zeilen zu aktualisieren,

in denen First\_Name ein nachfolgendes Leerzeichen enthält. rtrim entfernt nachfolgende Leerzeichen auch ohne die where-Klausel. Wenn Sie jedoch eine große Tabelle haben, ist Performance der update-Anweisung besser, wenn Sie die where-Klausel verwenden, um die Anzahl der Zeilen zu verringern, auf die rtrim angewandt wird.

Wie Sie sehen können, haben ltrim und rtrim ein einziges Argument: Die Zeichenfolge, von der die Leerzeichen entfernt werden sollen. Sie können bei beiden Funktionen auch ein optionales zweites Argument angeben: Eine alternative Menge von Zeichen, die von der Zeichenkette entfernt werden sollen. Als Beispiel betrachten Sie bitte Listing 7.8. Stellen Sie sich vor, daß Sie eine Tabelle namens Test\_Trim haben, mit einer einzigen Spalte namens MY\_COL vom Typ varchar2 . Sie wissen, daß einige Zeilen führende Zeichen aufweisen, die Sie entfernen wollen: x, y und z. Die Funktion ltrim wird in einer update-Anweisung verwendet, um die störenden Zeichen zu entfernen.

***Listing 7.8: Löschen führender Leerzeichen mit Hilfe der Funktion ltrim.***

```
SQL> select my_col
2 from test_trim;
MY_COL
-----
yzzxHello, world
zyxGoodbye, cruel world

SQL> update test_trim
2 set my_col = ltrim(my_col, 'xyz');
2 rows updated.
SQL> select my_col from test_trim;
MY_COL
-----
Hello, world
Goodbye, cruel world
```

Sie können nachfolgende Leerzeichen entfernen, indem Sie dieselbe Technik mit rtrim verwenden.

## Auffüllen einer Zeichenfolge

An einem gewissen Punkt kommen Sie in eine Situation, in der Sie eine Zeichenfolge mit führenden oder nachfolgenden Zeichen auffüllen müssen. Oracle bietet zu diesem Zweck zwei Funktionen: lpad und rpad.

## lpad

Um eine Zeichenfolge von links mit Zeichen aufzufüllen, verwenden Sie die Funktion lpad.





*Die Syntax ist*

**LPAD** (*string*, *n*, *pad\_string*)

Die Variablen sind folgendermaßen definiert:

- *string* ist die Literalzeichenfolge oder die Zeichenfolgenspalte, die von links aufgefüllt werden soll.
- *n* ist die Gesamtlänge der von `lpad` zurückgegebenen Zeichenfolge.
- *pad\_string* ist die Zeichenfolge, mit der *string* von links aufgefüllt werden soll.

Lassen Sie uns sehen, wie das funktioniert. Listing 7.9 zeigt, wie man mit `lpad` [eine Zeichenfolge mit führenden Leerzeichen auffüllt](#).

***Listing 7.9: Hinzufügen führender Leerzeichen zu einer Spalte mit Hilfe der Funktion `lpad`.***

```
SQL> select lpad(my_col,20) from test_trim;
LPAD(MY_COL,20)
-----
Hello, world
Goodbye, cruel world
```

Wenn das Argument *pad\_string* nicht angegeben wird, verwendet `lpad` Leerzeichen, um die Zeichenfolge von links aufzufüllen. Sie können eine Literalzeichenfolge angeben, die `lpad` verwendet, um die Zeichenfolge von links aufzufüllen. Die Anzahl der Zeichen, mit denen die Zeichenfolge aufgefüllt wird, hängt jedoch vom Wert von *n* ab. Listing 7.10 zeigt, wie man einen festen Betrag zur Funktion `length` [hinzufügen kann, die hier als Argument für `lpad` dient](#). Indem die zu `length` [hinzugefügte Zahl erhöht wird](#), können Sie *string* mehr als einmal mit *pad\_string* auffüllen, wie in der zweiten Abfrage gezeigt wird.

***Listing 7.10: Verwendung der Funktion `length` als Argument für die Funktion `lpad`.***

```
SQL> select lpad(my_col,length(my_col)+8,'You say ') from test_trim;
LPAD(MY_COL,LENGTH(MY_COL)+8,'YOUSAY')
-----
You say Hello, world
```

```
You say Goodbye, cruel world
```

```
SQL> select lpad(my_col,length(my_col)+16,'You say ') from
test_trim;
```

```
LPAD(MY_COL,LENGTH(MY_COL)+16,'YOUSAY')
```

```
-----
```

```
You say You say Hello, world
```

```
You say You say Goodbye, cruel world
```

Sie können eine Zeichenfolge auch mit dem Inhalt einer anderen Spalte von links auffüllen. Wie Sie bei der zweiten Abfrage in Listing 7.11 sehen können, wird my\_col von links mit my\_col2 aufgefüllt. Wie oft dieser Schritt ausgeführt wird, hängt von der Länge der Werte in beiden Spalten ab. Wenn my\_col [den Wert California hat und my\\_col2 den Wert Los\\_Angeles](#), dann gibt lpad eine Zeichenfolge zurück, in der California (10 Zeichen) von links mit Los\_Angeles (11 Zeichen) aufgefüllt wird, so daß Los Angeles 40 Zeichen ausfüllt.

***Listing 7.11: Verkettung zweier Spalten mit Hilfe der Funktion lpad.***

```
SQL> select * from test_trim;
```

```
MY_COL MY_COL2
```

```
-----
```

```
California Los Angeles
```

```
Michigan Jackson
```

```
Washington Seattle
```

```
Oregon Portland
```

```
SQL> select lpad(my_col,50,my_col2) from test_trim;
```

```
LPAD(MY_COL,50,MY_COL2)
```

```
-----
```

```
Los AngelesLos AngelesLos AngelesLos AngCalifornia
```

```
JacksonJacksonJacksonJacksonJacksonJacksonMichigan
```

```
SeattleSeattleSeattleSeattleSeattleSeattWashington
```

```
PortlandPortlandPortlandPortlandPortlandPortOregon
```

Durch Kombinieren dieser integrierten Funktionen können Sie komplexe Ausdrücke erstellen. Listing 7.12 verwendet die Längen von my\_col und my\_col2 als Argumente, um zu garantieren, daß das Auffüllen von links nur einmal stattfindet. Lassen Sie uns die select-Anweisung zerlegen. Das erste Argument von lpad ist my\_col2. Für das zweite Argument addieren Sie die Länge von my\_col2 zur Länge von my\_col und addieren eine 2 für das ', ' , das zwischen my\_col und my\_col2 eingefügt wird. Schließlich verketteten Sie für das dritte Argument von lpad [my\\_col2 mit ,](#) .

***Listing 7.12: Verwendung der Funktion length als Argument für die Funktion lpad.***

```
SQL> select lpad (my_col,
```

```
length(my_col)+length(my_col2)+2, my_col2 || ', ')
```

```
2 from test_trim;
```

```
LPAD(MY_COL, LENGTH(MY_COL)+LENGTH(MY_COL2)+2, MY_COL2 || ', ')
```

```
-----
```

```
Los Angeles, California
```

```
Jackson, Michigan
```

```
Seattle, Washington
```

```
Portland, Oregon
```

## rpad

rpad funktioniert genauso wie lpad.



*Verwenden Sie die folgende Syntax:*

---

RPAD (*string*, *n*, *pad\_string*)

Die Variablen sind folgendermaßen definiert:

- *string* ist die Literalzeichenfolge oder die Zeichenfolgenspalte, die von rechts aufgefüllt werden soll.
- *n* gibt an, wie oft von rechts mit *pad\_string* aufgefüllt werden soll.
- *pad\_string* ist die Zeichenfolge, mit der *string* von rechts aufgefüllt werden soll.

## Ändern der Groß- und Kleinschreibung in einer Zeichenfolge

Oracle bietet drei Funktionen, mit denen Sie die Groß- und Kleinschreibung einer Zeichenfolge ändern können:

- `initcap` ändert das erste Zeichen jedes Worts in Großschreibung.
- `lower` ändert alle Zeichen einer Zeichenfolge in Kleinschreibung.
- `upper` **ändert alle Zeichen einer Zeichenfolge in Großschreibung.**

Alle drei Funktionen haben ein einziges Argument: die Zeichenfolge, die bearbeitet werden soll. Listing 7.13 zeigt die Verwendung der Funktionen `upper` und `lower`.

***Listing 7.13: Verwendung der Funktionen upper und lower.***

```
SQL> select lower(Title), upper(Title)
2 from Course
3 order by Title;
```

```
LOWER(TITLE) UPPER(TITLE)
```

```
-----
abnormal psychology ABNORMAL PSYCHOLOGY
advanced arithmetic ADVANCED ARITHMETIC
ancient greek philosophers ANCIENT GREEK PHILOSOPHERS
early american history EARLY AMERICAN HISTORY
evolutionary grammar EVOLUTIONARY GRAMMAR
general calculus GENERAL CALCULUS
intro to anthropology INTRO TO ANTHROPOLOGY
intro to biology INTRO TO BIOLOGY
intro to circuit theory INTRO TO CIRCUIT THEORY
intro to dynamics INTRO TO DYNAMICS
intro to economics INTRO TO ECONOMICS
intro to english lit INTRO TO ENGLISH LIT
intro to philosophy INTRO TO PHILOSOPHY
intro to psychology INTRO TO PSYCHOLOGY
intro to structures INTRO TO STRUCTURES
invertebrate anatomy INVERTEBRATE ANATOMY
mammalian anatomy MAMMALIAN ANATOMY
modern english lit MODERN ENGLISH LIT
modern european history MODERN EUROPEAN HISTORY
modern philosophy MODERN PHILOSOPHY
monetary policy MONETARY POLICY
number theory NUMBER THEORY
phys anthro field techniques PHYS ANTHRO FIELD TECHNIQUES
pre-calculus PRE-CALCULUS
psych in film PSYCH IN FILM
seminar on chaos SEMINAR ON CHAOS
seminar on nacimera SEMINAR ON NACIMERA
seminar on theme analysis SEMINAR ON THEME ANALYSIS
workshop on jefferson WORKSHOP ON JEFFERSON
workshop on marx WORKSHOP ON MARX
workshop on neuroses WORKSHOP ON NEUROSES
workshop on normality WORKSHOP ON NORMALITY
```

```
32 rows selected.
```

Listing 7.14 zeigt, wie die Funktion initcap alle Zeichen in Kleinschreibung ändert und den ersten Buchstaben jedes Worts groß schreibt.

***Listing 7.14: Verwendung der Funktion initcap.***

```
SQL> select initcap(Title)
2 from Course
3 order by Title;

INITCAP(TITLE)
-----
Abnormal Psychology
Advanced Arithmetic
Ancient Greek Philosophers
Early American History
Evolutionary Grammar
General Calculus
Intro To Anthropology
Intro To Biology
Intro To Circuit Theory
Intro To Dynamics
Intro To Economics
Intro To English Lit
Intro To Philosophy
Intro To Psychology
Intro To Structures
Invertebrate Anatomy
Mammalian Anatomy
Modern English Lit
Modern European History
Modern Philosophy
Monetary Policy
Number Theory
Phys Anthro Field Techniques
Pre-Calculus
Psych In Film
Seminar On Chaos
Seminar On Nacimera
Seminar On Theme Analysis
Workshop On Jefferson
Workshop On Marx
Workshop On Neuroses
Workshop On Normality

32 rows selected.
```

**Die Funktion decode für die Rückgabe einer Zeichenfolge verwenden**

Viele Datenbank Anwendungen verwenden Spalten, die verschlüsselte Informationen enthalten. Manchmal erzeugt ein Datenbankdesigner eine Tabelle, um einen Code und seine Beschreibung zu speichern, besonders, wenn der Designer erwartet, daß die Codes sich ändern. In anderen Situationen steht die Spalte mit dem Code für sich alleine, ohne daß weitere Informationen in der Datenbank verfügbar wären.

Tag 3, »Logischer Datenbankentwurf«, besprach die Notwendigkeit für eine Tabelle, die die Details eines Zeitplans enthält - an welchen Tagen sich die Klassen treffen und für wie lange. Als Ergebnis haben Sie die Tabelle `Schedule_Type_Details` erzeugt. Eine der Spalten in dieser Tabelle, `Day`, speichert den Wochentag als ganze Zahl, wobei der Sonntag durch eine 1 repräsentiert wird, der Montag durch eine 2 usw.

Die meisten Benutzer finden es umständlich, die Inhalte dieser Tabelle zu entziffern. Glücklicherweise kann die Funktion `decode` kryptische Codes in etwas transformieren, das Benutzer problemlos interpretieren können.



*Die Syntax ist*

---

```
DECODE (expression, value1, returned_value1, ...
valueN, returned_valueN,
[default_returned_value])
```

Die Variablen sind folgendermaßen definiert:

- *expression* ist ein gültiger Oracle-Ausdruck.
- *valueN* ist der Wert, der gleich *expression* sein könnte.
- *returned\_valueN* ist der Wert, der von `decode` zurückgegeben wird, wenn *expression* gleich *valueN* ist.
- *default\_returned\_value* ist ein optionaler Wert, der von `decode` zurückgegeben wird, wenn *expression* keinem der Werte *value1* bis *valueN* gleich ist.

Listing 7.15 zeigt, wie die Funktion `decode` die numerische Angabe der Wochentage in den Wochentag transformiert.

**Listing 7.15: Umwandlung eines numerischen Wertes in eine Zeichenfolge mit Hilfe der Funktion `decode`.**

```
SQL> select Schedule_ID, Day,
```

```
2 decode (Day, 1, 'SUN', 2, 'MON', 3, 'TUE', 4, 'WED',
3 5, 'THU', 6, 'FRI', 7, 'SAT')
4 from Schedule_Type_Details
5 order by Schedule_ID, Day;
```

```
SCHEDULE_ID DAY DEC
-----
```

```
S180 6 FRI
T10 2 MON
T10 4 WED
T10 6 FRI
T13 2 MON
T13 4 WED
T13 6 FRI
T15 2 MON
T15 4 WED
T15 6 FRI
TT9 3 TUE
TT9 5 THU
```

12 rows selected.

## Umwandlung eines Zeichens in den zugehörigen ASCII-Wert

An einem gewissen Punkt kann es sein, daß Sie den ASCII-Wert eines Zeichens in einer Spalte erhalten wollen. Dazu gibt es die Funktion `ascii`. Sie hat ein einziges Argument. `ascii` gibt den ASCII-Wert des ersten Zeichens seines Arguments zurück. Listing 7.16 zeigt ein Beispiel für die Verwendung der Funktion `ascii`.

### *Listing 7.16: Verwendung der Funktion `ascii`.*

```
SQL> select Last_Name, ASCII(Last_Name)
2 from Instructor
3 order by Last_Name;
```

```
LAST_NAME ASCII(LAST_NAME)
-----
```

```
ANGELO 65
BATES 66
BILLINGS 66
CHANG 67
CHERNOW 67
CHU 67
DANIELS 68
```

EDWARDS 69  
HITCHCOCK 72  
JASON 74  
NILAND 78  
PARKER 80  
POULSON 80  
RESTON 82  
RICHARDSON 82  
TORRES 84  
WEISS 87  
YOUNG 89

18 rows selected.

## Bearbeiten von Datumsangaben

Es gibt eine unglaubliche Vielfalt in der Art und Weise, in der Datenbanksysteme Datums- und Zeitangaben behandeln. Glücklicherweise bietet Oracle einen speziellen Datentyp - date -, um Datums- und Zeitangaben zu behandeln. Dieser Datentyp hat sein eigenes internes Format, um das Sie sich nicht zu kümmern brauchen. Sie müssen nur wissen, daß er für die Speicherung von Jahrhundert, Jahr, Monat, Tag, Stunde, Minute und Sekunde zuständig ist. Wie diese Lektion zeigt, hat es viele Vorteile, diesen Datentyp an den richtigen Stellen zu verwenden.

### Der Oracle-Datentyp date

Der Oracle-Datentyp date ist effizient, weil er zur Speicherung nur sieben Byte benötigt. Zudem können Sie, wenn Sie eine Spalte als date definieren, alle integrierten Funktionen von Oracle zur Bearbeitung von Datums- und Zeitangaben verwenden.

Der Oracle-Datentyp date ist zudem sehr praktisch für Anwendungsentwickler. Man kann darüber streiten, ob die von Oracle verwendeten Algorithmen optimal sind; Sie könnten der Meinung sein, daß andere Methoden zur Speicherung von Datums- und Zeitangaben effizienter sind. Ohne Frage kann Ihnen jedoch die Verwendung des Oracle-Datentyps date eine Menge Zeit und Aufwand bei der Anwendungsentwicklung sparen. In der Tat, da die Vorteile dieses Datentyps so klar sind, sollten Sie wirklich den Oracle-Datentyp date verwenden, wenn Sie Datums- oder Zeitinformationen speichern müssen.

### Verwenden des Datentyps date zur Speicherung von Zeitinformationen



Der Oracle-Datentyp `date` speichert auch Zeitangaben: Stunde, Minute und Sekunde. Sie können eine Spalte, die als `date` definiert ist, dazu verwenden, um Datumsinformationen, Zeitinformationen oder beides zu speichern.

Wenn Sie sich dafür entscheiden, nicht den Datentyp `date` für die Speicherung von Datums- und Zeitangaben zu verwenden, müssen Sie andere Algorithmen verwenden, um die von Ihnen definierten Formate zu bearbeiten. Sie können dann keine der von Oracle angebotenen integrierten Funktionen zur Bearbeitung von Datums- und Zeitangaben nutzen. Eine Aufgabe, die Sie mit einer einzigen `select`-Anweisung hätten erledigen können, erfordert dann zusätzliche Bearbeitung in einer Programmiersprache oder einer Entwicklungsumgebung. Das folgende Szenario zeigt die Konsequenzen, die daraus entstehen, daß man den Oracle-Datentyp `date` dort *nicht* verwendet, wo es angemessen ist.

Stellen Sie sich vor, daß Sie das Einstellungsdatum und ein eventuelles Kündigungsdatum jedes Lehrers speichern müssen. Wenn Sie eine falsche Entscheidung fällen und sich dafür entscheiden, das Einstellungsdatum des Lehrers als numerischen Wert der Form `yymmdd` zu speichern, würde das Einstellungsdatum für einen Angestellten, der am 9. Mai 1957 eingestellt wurde, als 570509 gespeichert werden. Listing 7.17 enthält eine Beschreibung der Tabelle `Instructor`, so wie sie aussehen würde, wenn sie eine Spalte zur Speicherung des Einstellungsdatums enthielte.

**Listing 7.17: *Instructor-Tabelle, in der das Datum der Anstellung als Zahl gespeichert ist.***

```
SQL> desc Instructor
Name Null? Type
-----
INSTRUCTOR_ID VARCHAR2(20)
DEPARTMENT_ID NOT NULL VARCHAR2(20)
LAST_NAME NOT NULL VARCHAR2(25)
FIRST_NAME VARCHAR2(25)
MI VARCHAR2(1)
POSITION VARCHAR2(25)
TELEPHONE VARCHAR2(10)
FAX VARCHAR2(10)
EMAIL VARCHAR2(100)
HIRE_DATE NUMBER(6)
```

Schlauerweise (das dachten Sie zumindest) haben Sie sich entschieden, das Format `yymmdd` zu verwenden, so daß das Einstellungsdatum entweder in aufsteigender oder in absteigender Reihenfolge geordnet werden könnte. Sie könnten eine `select`-Anweisung verwenden, um die Zeilen aus der Tabelle `Instructor` nach Einstellungsdatum geordnet zu erhalten. Dieser Ansatz hat jedoch einige Probleme, nämlich:

- **Eine irrige Annahme.** Wenn ein Lehrer nach 1999 eingestellt wird, gibt die `select`-Anweisung nicht das richtige Ergebnis zurück. Ein Lehrer, der im Jahr 2000 oder danach eingestellt wurde, taucht am Anfang der Liste auf, da das Format nur die zwei letzten Ziffern des Jahres speichert, in dem der Lehrer angestellt wurde.

- **Mehr Arbeit und geringere Zuverlässigkeit.** Weil Sie sich nicht auf die Gültigkeitsprüfung von Oracle für die Daten verlassen können, ist es Ihre Aufgabe, einen Algorithmus zur Validierung des Einstellungsdatums zu entwickeln.
- **Eingeschränkte Funktionalität.** Wenn Sie eine Tabelle auf der Grundlage von Datenarithmetik abfragen müssen, erfordern Dinge, die mit einer einzigen select-Anweisung hätten erreicht werden können, zusätzliche Bearbeitung. Sie können keine select-Anweisung schreiben, die alle Angestellten liefert, deren Einstellungsdatum nach dem durchschnittlichen Einstellungsdatum liegt.

Wenn Sie statt dessen den Oracle-Datentyp `date` verwenden, können Sie sicher sein, daß jede Anwendung, die Sie entwickeln, auf andere Plattformen portiert werden kann.

Datenformate werden auf jeder Plattform unterstützt, auf der Oracle läuft. Wenn Sie planen, eine Anwendung auf einer Vielzahl von Betriebssystemen laufen zu lassen, werden Sie feststellen, daß die Verwendung des Oracle-Datentyps `date` einfacher ist, als für jedes Betriebssystem eine Vielzahl von Datums- und Zeitformaten zu unterstützen.

## Das aktuelle Datum und die aktuelle Uhrzeit: `sysdate`

Oracle besitzt eine Reihe von Werten, die Pseudospalten genannt werden und auf die man in SQL-Anweisungen zugreifen kann. Einer dieser Werte ist `sysdate`. Ungeachtet seines Namens enthält `sysdate` auch Zeitinformationen. Wie der Datentyp `date` ist `sysdate` bis auf eine Sekunde genau. `sysdate` ist ein extrem nützliches Konstrukt, um Zeilen während einer Einfüge- oder Aktualisierungsoperation mit einem Zeitstempel zu versehen. Viele Beispiele in dieser Lektion verwenden `sysdate`.

## Das Oracle-Modell für das Datumsformat

Weil der Datumsdatentyp die Werte für Jahrhundert, Jahre, Monat, Stunde, Minute und Sekunde speichert, kann jeder dieser Werte unabhängig voneinander extrahiert und formatiert werden. Die Datums- und Zeitelemente können zudem abgekürzt oder voll ausgeschrieben werden, je nachdem, welches Format Sie bevorzugen.



*Weil der Oracle-Datumsdatentyp das Jahrhundert enthält, sollten Sie keine Probleme mit dem Jahr 2000 haben, wenn Sie den Datumsdatentyp von Oracle für Spalten verwenden, die Datumsinformationen enthalten.*

---

Sie haben auch die Option, ein Datums- oder Zeitelement in verschiedenen Formaten zu wiederholen. Wie Sie sich erinnern, enthält die Tabelle `Schedule_Type_Details` eine Startzeit für jeden Tag einer bestimmten `Schedule-ID`. Es kann beispielsweise sein, daß ein Zeitplan einen Montag/Mittwoch/Freitag-Zeitplan beschreibt, der um 10 Uhr beginnt und 50 Minuten dauert. Die Spalte `Starting Time` enthält nur Informationen über die Startstunde und -minute. Darum müssen Sie die Funktion `to_char` mit einem Datums/Zeitformat verwenden, um den internen Datumswert in einen externen Zeichenwert zu übersetzen. Listing 7.18 zeigt, wie Sie den Inhalt der Tabelle `Schedule_Type_Details` anzeigen können.

***Listing 7.18: Umwandlung einer Datums- und Zeitangabe in eine Zeichenfolge.***

```
SQL> select Schedule_ID, Day, Starting_Time,  
to_char(Starting_Time, 'HH:MI PM')  
2 from Schedule_Type_Details  
3 order by Schedule_ID, Day;
```

```
SCHED DAY STARTING_ TO_CHAR(STARTING_TIME, 'HH:MIPM' )  
-----  
S180 6 01-MAR-97 09:00 AM  
T10 2 01-MAR-97 10:00 AM  
T10 4 01-MAR-97 10:00 AM  
T10 6 01-MAR-97 10:00 AM  
T13 2 01-MAR-97 01:00 PM  
T13 4 01-MAR-97 01:00 PM  
T13 6 01-MAR-97 01:00 PM  
T15 2 01-MAR-97 03:00 PM  
T15 4 01-MAR-97 03:00 PM  
T15 6 01-MAR-97 03:00 PM  
TT9 3 01-MAR-97 09:00 AM  
TT9 5 01-MAR-97 09:00 AM
```

12 rows selected.



*Sie können `sysdate` und die Tabelle `DUAL` verwenden, um mit verschiedenen Datums- und Zeitformaten zu experimentieren. Sie können `sysdate` aus der Tabelle `DUAL` auswählen, jedoch keine Zeilen in die Tabelle `DUAL` einfügen - sie darf nur eine Zeile haben, damit einige Oracle-Werkzeuge richtig funktionieren.*

---

Tabelle 7.1 enthält die gültigen Elemente des Datumsformats.

*Tabelle 7.1: Liste möglicher Elemente für das Datumsformat.*

| Formatelement | Beschreibung                          | Bereich                    |
|---------------|---------------------------------------|----------------------------|
| ss            | Sekunde                               | 0-59                       |
| sssss         | Sekunden nach Mitternacht             | 0-86399                    |
| mi            | Minute                                | 0-59                       |
| hh            | Stunde                                | 0-12                       |
| hh24          | 24-Stunden-Angabe                     | 0-23                       |
| dd            | Tag des Monats                        | 1-31 (hängt vom Monat ab)  |
| day           | Wochentag, ausgeschrieben             | SUNDAY-SATURDAY            |
| d             | Wochentag                             | 1-7                        |
| ddd           | Tag des Jahres                        | 1-366 (hängt vom Jahr ab)  |
| mm            | Monatsnummer                          | 1-12                       |
| mon           | Abgekürzter Monat                     | JAN-DEC                    |
| month         | Ausgeschriebener Monat                | JANUARY-DECEMBER           |
| yy            | Die letzten beiden Ziffern des Jahres | z.B. 96                    |
| yyyy          | Vollständige Jahresangabe             | z.B. 1996                  |
| year          | ausgeschriebenes Jahr                 | z.B. NINETEEN NINETY-SEVEN |

|    |                                    |              |
|----|------------------------------------|--------------|
| cc | Jahrhundert                        | z.B. 19      |
| q  | Quartal                            | 1-4          |
| j  | Tag nach dem Julianischen Kalender | z.B. 2448000 |
| w  | Woche des Monats                   | 1-5          |
| ww | Woche des Jahres                   | 1-52         |

## Das Standard-Datumsformat in Oracle

Das Standard-Datumsformat von Oracle ist dd-mon-yy. Zum Beispiel ist 01-jan-98 eine Datumsangabe gemäß dem Standard-Datumsformat von Oracle. Sie können ein Datum in diesem Format angeben, ohne irgendwelche Funktionen oder Datentypumwandlungen zu verwenden. Wenn Sie Datumsangaben jedoch in einem anderen Format anzeigen oder angeben müssen, müssen Sie eine der integrierten Funktionen verwenden, um das Datumsformat anzugeben, das Sie verwenden wollen.

Wenn Sie versuchen, einer Datumsspalte eine Zeichenfolge zuzuweisen, die nicht dem Standardformat entspricht, gibt Oracle wahrscheinlich eine Fehlermeldung zurück. Wenn beispielsweise die beiden ersten Ziffern größer sind als 31, gibt Oracle immer den Fehlercode ora-01847 zurück. Wenn die Abkürzung für den Monat nicht jan, feb, mar, may, jun, jul, aug, sep, oct, nov oder dec ist, gibt Oracle den Fehlercode ora-01843 zurück. Wenn der Tag des Monats nicht innerhalb des gültigen Bereichs für den jeweiligen Monat ist, gibt Oracle den Fehlercode ora-01839 zurück. Tabelle 7.2 enthält eine Liste der Fehlercodes von Oracle, die im Zusammenhang mit der Bearbeitung von Datumswerten stehen.

*Tabelle 7.2: Oracle-Fehlercodes für Datumsangaben.*

| Oracle-Fehlercode | Beschreibung                                                         |
|-------------------|----------------------------------------------------------------------|
| ora-01847         | Tag des Monats muß zwischen 1 und dem letzten Tag des Monats liegen. |
| ora-01813         | Stunde kann nur einmal angegeben werden                              |
| ora-01839         | Datum ist für angegebenen Monat nicht gültig                         |



*Wir nähern uns schnell dem neuen Jahrtausend. Weil die ersten Jahre des 21. Jahrhunderts im selben Bereich wie die Tage eines Monats liegen, wird es ohne Zweifel Verwechslungen zwischen den Jahren und den Tagen eines Monats geben. Wenn Sie beispielsweise den 2. Januar 2003 angeben wollen, können Sie leicht die Ziffern vertauschen und 03-jan-02 anstatt 02-jan-03 angeben. In beiden Fällen akzeptiert Oracle das Datum als gültiges Datumsformat.*

---

## Umwandlung von Datumsangaben in Zeichenfolgen

Sie müssen sich merken, daß eine Datumsspalte ein Datumswert bleibt, bis Sie sie in einen anderen Datentyp umwandeln. Wenn Sie beispielsweise die ersten fünf Zeichen einer Datumsspalte extrahieren wollen, müssen Sie den Wert mit der integrierten Funktion `to_char` in eine Zeichenfolge umwandeln.

---



*Das Format dieser Funktion ist*

---

`TO_CHAR(date_value, date_format)`

Die Variablen sind folgendermaßen definiert:

- *date\_value* ist ein Datumsliteral, ein Datumswert aus einer Spalte oder ein von einer integrierten Funktion zurückgegebener Datumswert.
- *date\_format* ist ein gültiges Datumsformat von Oracle.

Listing 7.19 zeigt, wie eine Abfrage die Funktion `to_char` verwendet, um das Einstellungsdatum eines Angestellten im Format `month dd, yyyy` zurückzugeben.

**Listing 7.19: Umwandlung einer Datumsangabe in eine Zeichenfolge.**

```
SQL> select Last_Name, First_Name,  
2 to_char(Hire_Date, 'MONTH DD, YYYY') H_DATE  
3 from Employee  
4 order by Hire_Date;  
LAST_NAME FIRST_NAME H_DATE
```

```
-----  
SMITH JEAN APRIL 10, 1982  
HERNANDEZ RANDY NOVEMBER 18, 1983  
GLEASON PAUL APRIL 05, 1984  
BARRETT SARAH JANUARY 16, 1989  
HIGGINS BEN FEBRUARY 11, 1989  
YEN CINDY JUNE 09, 1991  
GILROY MAX SEPTEMBER 22, 1992  
CARSON BETH DECEMBER 12, 1992  
SWANSON HARRY MAY 18, 1993  
9 rows selected.
```

Wenn eine Datumsangabe mit der Funktion `to_char` in eine Zeichenfolge umgewandelt wurde, können Sie sie als Argument für andere Zeichenfolgenfunktionen verwenden. Sie können beispielsweise die Funktion `substr` verwenden, um eine Teilzeichenfolge aus der Zeichenfolge zu gewinnen. Listing 7.20 zeigt, wie man die Funktion `substr` verwendet, um den ersten Buchstaben des Einstellungsmonats eines Angestellten zu gewinnen.

*Listing 7.20: Verwendung der Funktion `to_char` in einem Aufruf der Funktion `substr`.*

```
SQL> select Last_Name, First_Name,  
2 substr(to_char(Hire_Date, 'MON'),1,1) the_first_letter_of_the_month  
3 from Employee  
4 order by the_first_letter_of_the_month;  
LAST_NAME FIRST_NAME T  
-----  
SMITH JEAN A  
GLEASON PAUL A  
CARSON BETH D  
HIGGINS BEN F  
BARRETT SARAH J  
YEN CINDY J  
SWANSON HARRY M  
HERNANDEZ RANDY N  
GILROY MAX S  
9 rows selected.
```

Im nächsten Abschnitt betrachten wir einige der vielen Möglichkeiten, wie Datums- und Zeitangaben dargestellt werden können. Beim Darstellen und Angeben dieser Werte verfügen Sie über eine enorme Flexibilität.

## Umwandlung von Zeichenfolgen in Datumsangaben

Es ist nicht überraschend, daß die Umwandlung von Zeichenfolgen in Datumsangaben der

Umwandlung von Datumsangaben in Zeichenfolgen ähnelt. Anstatt der integrierten Funktion `to_char` verwenden Sie die integrierte Funktion `to_date`, weil Sie einen Datumswert unter Verwendung eines angegebenen Formats angeben wollen.



*Die Argumente der Funktion `to_date` sind die Umkehrung der Argumente der Funktion `to_char`.*

`TO_DATE (string_value, date_format)`

Die Variablen sind folgendermaßen definiert:

- *string\_value* ist ein Zeichenfolgenliteral, eine Zeichenfolge aus einer Spalte oder eine von einer Zeichenfolge zurückgegebene Zeichenfolge.
- *date\_format* ist ein gültiges Oracle-Datumsformat.

Wenn Sie beispielsweise eine Zeichenfolge konvertieren wollen, die nicht das Standarddatumsformat von Oracle verwendet (dd-mon-yy), verwenden Sie die Funktion `to_date`. Listing 7.21 zeigt, wie man eine Abfrage verwendet, um die Zahl der Tage zu bestimmen, die seit der amerikanischen Zweihundertjahrfeier vergangen sind.

**Listing 7.21: Ein Beispiel für Datumsberechnungen.**

```
SQL> select SYSDATE - TO_DATE('07-04-1976', 'MM-DD-YYYY')
2 from DUAL;
SYSDATE-TO_DATE('07-04-1976', 'MM-DD-YYYY')
-----
6878.9465
```

## Datums- und Zeitangaben

Denken Sie daran, daß jede Spalte, die unter Verwendung des Datentyps `date` definiert wird, sowohl ein Datums- als auch eine Zeitangabe enthält. Wenn Sie in dieser Spalte lediglich eine Zeitangabe speichern wollen, wird das Datum auf einen Standardwert gesetzt. Listing 7.22 zeigt die Verwendung eines gebräuchlichen Formats für Zeitangaben - hh:mi:ss.

**Listing 7.22: Verwendung eines Datumsformats in einem Aufruf der Funktion `to_char`.**



```
SQL> select Employee_ID, to_char(Time_Clocked_In, 'HH:MI:SS')
Time_Clocked_In
2 from Time_Clock
3 order by Employee_ID;
```

```
EMPLOYEE_ID TIME_CLOCKED_IN
```

```
-----
```

```
1002 09:02:03
```

```
1003 08:51:12
```

```
1004 08:59:33
```

```
1005 09:22:12
```

Denken Sie daran, daß der Oracle-Datentyp date die Uhrzeit bis auf eine Sekunde genau speichern kann.



*Wenn Sie ein 24-Stunden-Format für die Zeit verwenden wollen, sollte das Zeitformat folgendermaßen angegeben werden:*

```
SELECT TO_CHAR(arrival_time, 'HH24:MI:SS') FROM DUAL;
```



*Es ist leicht, Monate und Minuten in Datums- und Zeitformaten durcheinanderzubringen. Oracle akzeptiert beispielsweise die folgende insert-Anweisung, obwohl es nicht das ist, was Sie im Sinn hatten (mm anstatt mi):*

```
INSERT INTO EMPLOYEE
(EMPLOYEE_ID, START_TIME)
VALUES
(1033, TO_CHAR('08:05', 'HH24:MM'));
```



*Oracle wird diese Anweisung folgendermaßen interpretieren: Die Startzeit für den Angestellten Nummer 1033 wird auf 8:00 morgens gesetzt, der Monat auf Mai. Da mm immer zwischen 1 und 12 liegt, akzeptiert Oracle den übergebenen Wert immer, auch wenn es nicht das ist, was Sie vorhatten.*

---

Oracle hat auch ein Modell für das Zeitformat, das es Ihnen ermöglicht, die Zeit als Sekunden nach Mitternacht anzugeben. Im folgenden Beispiel nehmen wir an, daß die Uhrzeit 2:00 nachts ist:

**SELECT TO\_CHAR(SYSDATE, 'SSSSS') FROM DUAL**  
7200 Dies entspricht zwei Stunden.

Weil Sie das Formatmodell sssss verwenden, gibt Oracle das Datum als Zeit in Sekunden nach Mitternacht zurück. Die Zeitangabe 2:00 steht für zwei Stunden nach Mitternacht, das ist gleich 7200 Sekunden.

## Berechnen des Zeitraums zwischen zwei Datumsangaben

Ein weiterer Vorteil der Verwendung des Oracle-Datentyps date ist, daß er Datumsarithmetik unterstützt. Sie können beispielsweise Tage zu einem existierenden Datum hinzuaddieren oder davon abziehen:

```
select sysdate + 7 from dual;
```

Indem Sie 7 zu sysdate hinzuaddieren, erhalten Sie ein Datum eine Woche nach dem aktuellen Datum. In ähnlicher Weise können Sie Tage von einem Datum abziehen, um ein früheres Datum zu berechnen.

## Festlegen einer numerischen Spalte

Wie Sie bereits gesehen haben, wird der Datentyp einer Spalte in den Anweisungen create table und alter table festgelegt.

---



*Die allgemeine Syntax für die Angabe eines numerischen Datentyps ist*

---

NUMBER ([*precision* [, *scale*]])

Die Variablen sind folgendermaßen definiert:

- *precision* ist ein optionales Argument, das die Gesamtstellenzahl für diesen Spaltenwert angibt.
- *scale* ist ein optionales Argument, das die Anzahl der Nachkommastellen für den Spaltenwert angibt.

Wenn Sie *precision* oder *scale* nicht angeben, akzeptiert Oracle eine Gesamtstellenzahl von bis zu 38 - die maximale Genauigkeit, die Oracle bietet. Wenn Sie eine Spalte angeben, sollten Sie in Betracht ziehen, die Breite der numerischen Werte durch die Angabe einer entsprechenden Präzision einzuschränken. Wenn beispielsweise eine Spalte die Körpertemperatur eines Patienten in Grad Fahrenheit speichern soll, definieren Sie die Spalte als

Body\_Temp\_F Number(4,1)

Auf diese Weise können Sie in Body\_Temp\_F insgesamt vier Ziffern speichern, davon eine Nachkommastelle. Listing 7.23 zeigt, wie eine Oracle-Datenbank Temperaturangaben mit der angegebenen Gesamtzahl an Stellen und Nachkommastellen akzeptiert.

***Listing 7.23: Oracle akzeptiert Zahlen, die die vorgegebene Anzahl von Gesamt- und Nachkommastellen besitzen.***

```
SQL> update Patient
2 set Body_Temp_F = 99.2
3 where
4 Patient_ID = 'A2002';
1 row updated.
SQL> update Patient
2 set Body_Temp_F = 103.8
3 where
4 Patient_ID = 'E3893';
1 row updated.
```

Listing 7.24 zeigt, wie die Oracle-Datenbank Werte mit falscher Gesamtstellenzahl und Nachkommastellenzahl zurückweist.

***Listing 7.24: Oracle weist Werte zurück, die nicht die vorgegebene Anzahl von Gesamt- und Nachkommastellen aufweisen.***

```
SQL> update Patient
2 set Body_Temp_F = 1003.8
3 where
4 Patient_ID = 'N3393';
set Body_Temp_F = 1003.8
*
```

```
ERROR at line 2:
ORA-01438: value larger than specified precision allows for this
column
```

Diese Definition erlaubt für Body\_Temp\_F Werte von bis zu 999.9 Grad Fahrenheit - ein für Menschen unmöglicher Wert. Zusätzlich zur Angabe von Gesamtstellen- und Nachkommastellenzahl müssen Sie eine check-Regel angeben, um die Werte für eine Spalte einzuschränken.

Wenn Sie einen Wert für precision, aber nicht für scale angeben, schneidet Oracle den Nachkommateil einer reellen Zahl ab, bevor der Wert in der Spalte gespeichert wird. Dieses Konzept wird in Listing 7.25 gezeigt.

***Listing 7.25: Die Nachkommastellen werden abgeschnitten.***

```
SQL> create table Number_Demo (
2 Int_Value number(3),
3 Real_Value number(3,1),
4 Num_Value number);
Table created.
```

```
SQL> insert into Number_Demo
2 (Int_Value)
3 values
4 (12.2);
1 row created.
```

```
SQL> select Int_Value
2 from Number_Demo;
INT_VALUE
-----
12
```

Wenn Werte für precision und scale angegeben wurden und Sie einen numerischen Wert speichern, dessen Nachkommastellenzahl die Nachkommastellenzahl der Spalte überschreitet, schneidet Oracle die Nachkommastellen auf die Anzahl der Nachkommastellen in der Spalte ab, wie in Listing 7.26 gezeigt wird.

***Listing 7.26: Alle über die für die Spalte vorgegebene Anzahl hinaus vorhandenen Nachkommastellen werden abgeschnitten.***

```
SQL> insert into Number_Demo
2 (Real_Value)
3 values
4 (3.144);
1 row created.
```

```
SQL> select Real_Value
2 from Number_Demo;
REAL_VALUE
-----
3.1
```

Zusätzlich zum Datentyp number akzeptiert Oracle die folgenden Schlüsselwörter für die Beschreibung einer numerischen Spalte:

- numeric, decimal und dec
- integer, int und smallint
- float, double precision und real

Oracle unterstützt diese Datentypen, um kompatibel zu ANSI SQL und anderen relationalen Datenbanksystemen wie IBM SQL/DS und DB2 zu sein. Die Datentypen numeric, decimal und dec sind identisch zum Datentyp number. [integer, int und smallint werden in number\(38\) umgesetzt.](#) [float, double precision und real werden alle in number umgesetzt.](#) Wenn Sie nicht gerade ein Datenbankschema von einer Nicht-Oracle-Datenbank konvertieren, sollten Sie beim Definieren von Spalten immer den Datentyp number angeben.

## Speicherung von Zahlen in Oracle

Oracle speichert Zahlen nicht in derselben Weise wie Programmiersprachen, etwa C oder Fortran. In C benötigt beispielsweise eine Fließkommazahl unabhängig von ihrem Wert immer denselben Speicherplatz, während in Oracle die Anzahl der Bytes, die für die Speicherung einer Zahl verwendet werden, von ihrer Gesamtstellenzahl abhängt. Um diese Tatsache zu verdeutlichen, können Sie die in Oracle integrierte Funktion vsize verwenden, welche die Anzahl der von ihrem Argument belegten Bytes zurückgibt.

Listing 7.27 enthält die Abfrage einer Tabelle, die eine numerische Spalte namens Num\_Value besitzt. Rechts vom Spaltenwert ist vsize(Num\_Value) angegeben - diese Funktion liefert die Anzahl der Bytes, die verwendet werden, um Num\_Value zu speichern. Oracle kann zwei Ziffern einer Zahl in einem Byte speichern. Ein weiteres Byte wird verwendet, um das Vorzeichen und den Exponenten zu speichern.

**Listing 7.27: Bestimmung des von einer Zahl benötigten Speicherplatzes.**

```
SQL> select Num_Value, vsize(Num_Value)
```

```
2 from Number_Demo;
NUM_VALUE VSIZE(NUM_VALUE)
-----
123 3
1234 3
12345 4
123456 4
1234567 5
12345678 5
123456789 6
12345.679 6
```

## Umwandlung einer Zahl in eine Zeichenfolge

Es gibt zwei typische Gründe, einen numerischen Wert in eine Zeichenfolge umzuwandeln:

- Um das Anzeigeformat einer Zahl in einem Formular oder einem Bericht zu ändern
- Um einen numerischen Ausdruck in einem Formular oder einem Bericht mit einer Zeichenfolge zu verketten

## Automatische Umwandlung einer Zahl in eine Zeichenfolge

In einigen Situationen konvertiert SQL eine Zahl automatisch in eine Zeichenfolge. Listing 7.28 zeigt, wie Oracle eine angegebene Zahl in einen Zeichenwert umwandelt, wenn diese in einer Spalte vom Typ varchar2 gespeichert wird.

### *Listing 7.28: Automatische Umwandlung einer Zahl in eine Zeichenfolge.*

```
SQL> insert into Course
2 (Course_ID, Department_ID, Title, Additional_Fees)
3 values
4 (782, 'BIO', 'INVERTEBRATE CLONING LAB', 275);
```

1 row created.

```
SQL> select Course_ID, Department_ID, Title
2 from Course
3 where
4 Course_ID = '782';
```

```
COURS DEPARTMENT_ID TITLE
-----
```

## 782 BIO INVERTEBRATE CLONING LAB

Wenn Sie die in Listing 7.28 eingefügte Zeile betrachten, können Sie sehen, daß die Zahl 782, die als Course\_ID verwendet wurde, in einen varchar2-Wert von 782 umgewandelt wurde, der in der Spalte Course\_ID gespeichert wird.

### Umwandlung einer Zahl in eine Zeichenfolge mit Hilfe der Funktion to\_char

Die Funktion to\_char wird verwendet, um eine Zahl *explizit* in eine Zeichenfolge umzuwandeln.



*Ihre Syntax ist*

---

TO\_CHAR (*number* [, *format*])

Die Variablen sind folgendermaßen definiert:

- *number* ist ein numerischer Ausdruck, der umgewandelt werden soll.
- *format* ist ein optionales Formatmodell, das von to\_char verwendet werden soll.

Listing 7.29 bietet ein Beispiel für die Verwendung der Funktion to\_char ohne Format. Beachten Sie, daß die erste Spalte - Real\_Value - in SQL\*Plus rechtsbündig ist, während die zweite Spalte - to\_char(Real\_Value) - in SQL\*Plus linksbündig ist, da sie eine Zeichenspalte ist.

**Listing 7.29: Umwandlung einer Zahl in eine Zeichenfolge mit Hilfe der Funktion to\_char ohne Formatangabe.**

```
SQL> select Real_Value, to_char(Real_Value)
2 from Number_Demo;
REAL_VALUE TO_CHAR(REAL_VALUE)
-----
3.1 3.1
```

Jetzt ist der richtige Zeitpunkt, das Formatmodell von Oracle für Zahlen zu besprechen. Die in den Listings 7.30 bis 7.33 gezeigten Codes zeigen die wichtigsten Elemente des Formatmodells - diejenigen, die Sie am häufigsten verwenden werden.

Um die Zahl der anzuzeigenden Ziffern anzugeben, verwenden Sie 9 für jede Ziffer. Sie können auch ein Komma und einen Dezimalpunkt zum angegebenen Format hinzufügen. Beachten Sie als Beispiel bitte Listing 7.30.

***Listing 7.30: Verwendung einer numerischen Formatmaske.***

```
SQL> select Course_ID, Title, to_char(Additional_Fees, '9,999.99')
2 from Course
3 where
4 Department_ID = 'BIO'
5 order by Course_ID;
```

```
COURS TITLE TO_CHAR(A
-----
101 INTRO TO BIOLOGY 55.00
177 INVERTEBRATE ANATOMY 65.00
178 MAMMALIAN ANATOMY 70.00
```

Um eine Zahl mit führenden Nullen anzuzeigen, verwenden Sie 0 am Anfang des Formats, wie in Listing 7.31 gezeigt wird.

***Listing 7.31: Verwendung einer führenden Null in einer Formatmaske.***

```
SQL> select Course_ID, Title, to_char(Additional_Fees, '099.99')
2 from Course
3 where
4 Department_ID = 'ECON'
5 order by Course_ID;
```

```
COURS TITLE TO_CHAR
-----
101 INTRO TO ECONOMICS 025.00
189 MONETARY POLICY 750.00
199 WORKSHOP ON MARX 000.00
```

Um ein führendes Dollarzeichen anzuzeigen, beginnen Sie das Format mit \$ (siehe Listing 7.32).

***Listing 7.32: Angabe eines Dollarzeichens in einer Formatmaske.***

```
SQL> select Course_ID, Title, to_char(Additional_Fees, '$999.99')
2 from Course
3 where
4 Department_ID = 'ECON'
5 order by Course_ID;
```

```
COURS TITLE TO_CHAR(
```



---

```
101 INTRO TO ECONOMICS $25.00
189 MONETARY POLICY $750.00
199 WORKSHOP ON MARX $.00
```

Wenn Sie wollen, daß eine Zahl in wissenschaftlicher Schreibweise erscheint, geben Sie nach der Stellenzahl `eeee` an (siehe Listing 7.33).

**Listing 7.33: Festlegung der wissenschaftlichen Notation in einer Formatmaske.**

```
SQL> select Num_Value, to_char(Num_Value, '9.9999EEEE')
2 from Number_Demo
3 order by Num_Value;
NUM_VALUE TO_CHAR(NUM_
-----
123 1.2300E+02
1234 1.2340E+03
12345 1.2345E+04
12345.679 1.2346E+04
```

## Umwandlung einer Zeichenfolge in eine Zahl

Die Funktion `to_number` ist das Gegenstück zu `to_char`: Sie konvertiert einen Zeichenausdruck in eine Zahl mit dem angegebenen Format.



Die Syntax für `to_number` ist

---

```
TO_NUMBER (string [,format])
```

Die Variablen sind folgendermaßen definiert:

- *string* ist ein Zeichenausdruck, der konvertiert werden soll.
- *format* ist ein optionales Formatmodell, daß `to_number` verwenden soll

`to_number` verwendet dasselbe Formatmodell wie `to_char`. Listing 7.34 zeigt, wie man einen

Zeichenwert, der Dividenden repräsentiert, in eine Zahl umwandelt.

***Listing 7.34: Verwendung der Funktion to\_number.***

```
SQL> update Security_Price
2 set Last_Qtr_EPS = to_number('$2.81', '$999.99')
3 where
4 Symbol = 'ZGEGE';
1 row updated.
```

```
SQL> select Symbol, Last_Qtr_EPS
2 from Security_Price
3 where
4 Symbol = 'ZGEGE';
SYMBOL LAST_QTR_EPS
-----
ZGEGE 2.81
```

## Verwendung integrierter Statistikfunktionen

Oracle bietet die folgenden statistischen Funktionen, die in Wirklichkeit Gruppenfunktionen sind:

- avg(value) berechnet den Durchschnitt, oder Mittelwert, seines Arguments aus der Menge der Zeilen, auf die sie angewandt wird.
- stddev(value) gibt die Standardabweichung seines Arguments aus der Menge von Spalten, auf die sie angewandt wird, zurück.
- variance(value) gibt die Varianz seines Arguments aus der Menge der Spalten zurück, auf die sie angewandt wird.

Weil diese Gruppenfunktionen komplexer sind, wird ihre Verwendung im Detail an Tag 8, »Bilden fortgeschrittener Abfragen mit SQL«, beschrieben.

## Runden und Abschneiden von Zahlen

Oracle bietet vier integrierte Funktionen für das Runden und Abschneiden von Kommazahlen.

```
ROUND(value, [scale])
TRUNC(value, [scale])
FLOOR(value)
CEIL(value)
```

Bei diesen Funktionen ist *value* ein numerischer Ausdruck, *scale* ist ein optionales Argument, das die Anzahl der Ziffern angibt, die die Funktion beim Runden oder Abschneiden verwenden soll. (Die Voreinstellung ist 0.)

Die folgenden Beispiele zeigen Ihnen die Verwendung jeder Funktion.

## round

Die Funktion `round` hat zwei Argumente: einen numerischen Ausdruck und eine optionale Zahl von Ziffern, die beim Runden verwendet werden sollen. Wenn das zweite Argument nicht angegeben wird, gibt `round` den Wert seines Arguments auf die nächste ganze Zahl gerundet zurück. Wenn das zweite Argument angegeben wird, gibt `round` den Wert seines Arguments, auf die nächste Kommazahl mit der angegebenen Zahl von Nachkommastellen gerundet, zurück. `round` kann mit Literalwerten verwendet werden, wie in Listing 7.35 gezeigt wird.

### *Listing 7.35: Verwendung der Funktion round.*

```
SQL> select round(123.2) from dual;
ROUND(123.2)
-----
123

SQL> select round(123.27,1) from dual;
ROUND(123.27,1)
-----
123.3

SQL> select round(101.8) from dual;
ROUND(101.8)
-----
102
```

## trunc

Die Funktion `trunc` ähnelt der Funktion `round`. Anstatt auf die nächste ganze Zahl zu runden, schneidet `trunc` jedoch den Nachkommateil ihres numerischen Arguments ab. Sie können `trunc` mit einem Literalwert verwenden, wie in Listing 7.36 gezeigt wird.

### *Listing 7.36: Abschneiden einer Zahl.*

```
SQL> select trunc(123.33), trunc(123.567,2)
2 from dual;
```

```
TRUNC(123.33) TRUNC(123.567, 2)
-----
123 123.56
```

## floor

Die Funktion floor ist fast identisch mit der Funktion trunc, nur kann floor nicht auf eine Kommazahl abschneiden. Die Funktion floor gibt die ganze Zahl zurück, die kleiner oder gleich ihrem numerischen Argument ist, wie in Listing 7.37 gezeigt wird.

*Listing 7.37: Verwendung der Funktion floor.*

```
SQL> select floor(128.3), floor(129.8)
2 from dual;
FLOOR(128.3) FLOOR(129.8)
-----
128 129
```

## ceil

Die Funktion ceil gibt die *Aufrundungszahl* für ihr numerisches Argument zurück - die kleinste ganze Zahl, die größer oder gleich ihrem Argument ist. Betrachten Sie das Beispiel in Listing 7.38.

*Listing 7.38: Verwendung der Funktion ceil.*

```
SQL> select ceil(128.3), ceil(129.8)
2 from dual;
CEIL(128.3) CEIL(129.8)
-----
129 130
```

## Ermitteln des größten und des kleinsten Werts

Sie können die Funktionen max und min verwenden, um den größten und den kleinsten Wert für eine bestimmte Spalte einer Tabelle zu erhalten. Technisch gesehen sind max und min Gruppenfunktionen. Sie müssen jedoch nicht die group by-Klausel einer select-Anweisung verwenden, um diese Funktionen zu verwenden. Als Beispiel zeigt Listing 7.39, wie man den größten und den kleinsten Wert für Additional\_Fees aus der Tabelle Course erhält.

### ***Listing 7.39: Verwendung der Funktionen min und max.***

```
SQL> select min(Additional_Fees), max(Additional_Fees)
2 from Course;

MIN(ADDITIONAL_FEES) MAX(ADDITIONAL_FEES)
-----
0 750
```

### **Ermitteln, ob ein Wert null ist**

Wenn Sie eine Oracle-Anwendung entwickeln, kommen Sie in Situationen, in denen der Bildschirm oder ein Bericht Informationen über eine Spalte zeigt, die null sein kann. Wenn Sie anstatt der null einen bestimmten Wert zurückgeben wollen, können Sie die Funktion `nvl` von Oracle verwenden, um die Ersetzung vorzunehmen.



*Hier ist die Syntax:*

---

`NVL (column-value, substitute-value)`

Die Variablen sind folgendermaßen definiert.

- *column-value* ist der Spaltenwert, der ausgewertet werden soll.
- *substitute-value* ist der Wert, den die Funktion `nvl` zurückgeben soll, wenn *column-value* den Wert Null hat.

Listing 7.40 bietet ein Beispiel. Stellen Sie sich vor, wir hätten die Tabelle `Course` so modifiziert, das `Additional_Fees` auf null gesetzt wird, wenn es den Wert Null hat. Als nächstes fragen Sie die Tabelle `Course` ab und sehen, daß viele Zeilen keinen Wert für `Additional_Fees` haben. Danach fragen Sie die Tabelle ab, wobei Sie die Funktion `nvl` so verwenden, das sie eine Null zurückgibt, wenn `Additional_Fees` den Wert null hat.

### ***Listing 7.40: Verwendung der Funktion nvl.***

```
SQL> update course
2 set additional_fees = null
3 where additional_fees = 0;
```

15 rows updated.

```
SQL> select Department_ID, Course_ID, Additional_Fees
2 from Course
3 order by Department_ID, Course_ID;
```

```
DEPART  COURS  ADDITIONAL_FEES
```

```
-----
```

```
ANTHRO 101
ANTHRO 174 55
ANTHRO 189 7.5
BIO 101 55
BIO 177 65
BIO 178 70
ECON 101 25
ECON 189 750
ECON 199
ENG 101 75
ENG 102 45
ENG 103 35
ENG 199 45
ENGL 101
ENGL 189
ENGL 192
ENGL 193
HIST 115
HIST 184
HIST 199
MATH 101
MATH 189
MATH 50
MATH 51 10
PHILO 101
PHILO 174
PHILO 198
PSYCH 101 25
PSYCH 181 75
PSYCH 183 45
PSYCH 185 20
PSYCH 501 75
```

32 rows selected.

```
SQL> select Department_ID, Course_ID, nvl(Additional_Fees,0)
2 from Course
3 order by Department_ID, Course_ID;
```

```
DEPART  COURS  NVL(ADDITIONAL_FEES,0)
```

```
-----
```

```
ANTHRO 101 0
ANTHRO 174 55
ANTHRO 189 7.5
BIO 101 55
BIO 177 65
BIO 178 70
ECON 101 25
ECON 189 750
ECON 199 0
ENG 101 75
ENG 102 45
ENG 103 35
ENG 199 45
ENGL 101 0
ENGL 189 0
ENGL 192 0
ENGL 193 0
HIST 115 0
HIST 184 0
HIST 199 0
MATH 101 0
MATH 189 0
MATH 50 0
MATH 51 10
PHILO 101 0
PHILO 174 0
PHILO 198 0
PSYCH 101 25
PSYCH 181 75
PSYCH 183 45
PSYCH 185 20
PSYCH 501 75
```

32 rows selected.

## Zusammenfassung

Diese Lektion hat Ihnen gezeigt, wie man eine Reihe von Operatoren und integrierten Funktionen mit Zeichenfolgen, numerischen Werten und Datums-/Zeitwerten verwendet. Einige der integrierten Funktionen für Zeichenfolgen sind:

- substr gibt den angegebenen Teil einer Zeichenfolge zurück.
- length gibt die Länge einer Zeichenfolge zurück.
- lpad und rpad füllen Zeichenfolgen von links und von rechts mit Zeichenfolgen auf.
- ltrim und rtrim entfernen führende und nachfolgende Leerzeichen und andere Zeichen.
- replace ersetzt eine Zeichenfolge durch eine andere.
- lower, upper und initcap steuern die Groß- und Kleinschreibung in einer Zeichenfolge.
- decode übersetzt Spaltenwerte.
- ascii gibt das numerische ASCII-Äquivalent eines Zeichens zurück.
- instr liefert die Zeichenposition, an der ein Muster in einer Zeichenkette gefunden wird.

Zusätzlich gab es in dieser Lektion die folgenden Informationen über Datums- und Zeitwerte:

- Der Oracle-Datentyp date speichert Datums- und Zeitinformationen mit einer Genauigkeit von einer Sekunde.
- Der Oracle-Datentyp date sollte verwendet werden, wann immer es möglich ist, auch wenn nur Datums- oder nur Zeitinformationen gespeichert werden.
- sysdate ist eine Pseudospalte, die immer das aktuelle Datum und die aktuelle Uhrzeit bis auf eine Sekunde genau zurückgibt.
- Das Standarddatumsformat von Oracle ist dd-mon-yy, z.B. 01-jan-98 für den ersten Januar 1998.
- Die Funktion to\_char konvertiert Datumswerte in Zeichenfolgen.
- Die Funktion to\_date konvertiert Zeichenketten in Datumswerte.

Denken Sie an die folgenden Konzepte, wenn Sie SQL-Anweisungen für Ihre Anwendung erstellen.

- Verwenden Sie für numerische Spalten den Datentyp number.
- Sie können numerische Ausdrücke mit der Funktion to\_char in Zeichenfolgen umwandeln.
- Sie können Zeichenfolgen mit der Funktion to\_number in Zahlen umwandeln.
- max und min sind Gruppenfunktionen, die den Maximal- und Minimalwert für eine bestimmte Spalte oder einen bestimmten Ausdruck aus einer Menge von Zeilen liefern.
- Die Funktion nvl gibt einen anderen Wert - wie 0 oder n/a - für den Nullwert zurück.

## Wie geht es weiter?

An Tag 8 lernen Sie mehr über fortgeschrittene Merkmale von SQL, wie die Verwendung von Gruppenfunktionen, den Verbund zweier Tabellen und andere Themen.

## Fragen und Antworten

Frage:



Können die integrierten Funktionen von Oracle nur mit SQL\*Plus oder SQL Worksheet verwendet werden?

Antwort:

*Nein. Sie können die integrierten Funktionen mit jedem Werkzeug verwenden, das es Ihnen ermöglicht, SQL-Anweisungen einzugeben - z.B Oracle Forms oder PowerBuilder.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Erstellen Sie eine SQL-Anweisung, die die Zeilen der Tabelle Instructor wie im folgenden Beispiel gezeigt liefert:

Professor Parker

2. Erstellen Sie eine SQL-Anweisung, die den Instructor liefert, dessen Nachname der erste in der alphabetischen Reihenfolge ist.

## Übungen

Erzeugen Sie unter Verwendung der Anweisung create table <xyz> as [eine Tabelle namens NEW\\_CLASS](#), die auf einem Verbund der drei Tabellen Class, Schedule\_Type und Schedule\_Type\_Details basiert und die folgenden Spalten enthält:

- Class\_ID
- Department\_ID
- Course\_ID
- Den ausgeschriebenen Wochentag
- Die ausgeschriebene Zeit (z.B. 11:00 a.m.)

❖Kapitel❖ Top Inhalt Index **SAMS** Kapitel❖

# 8

# Fortgeschrittene Abfragen mit SQL

[Integrierte Gruppenfunktionen](#)

[Nach eindeutigen Zeilen suchen](#)

[Gruppieren von Zeilen](#)

[Der Umgang mit hierarchischen Informationen](#)

[Die Join-Operation](#)

[Verwendung von Mengenoperatoren in einer  
select-Anweisung](#)

[Erstellen einer Tabelle durch Auswahl aus einer anderen Tabelle](#)

[Datensichten verwenden](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

Sie haben bereits gesehen, wie die select-Anweisungen Datensätze aus einer einzelnen Tabelle abfragt, das Abfragen von Datensätzen ist jedoch nur eine der vielen Fähigkeiten dieser vielseitigen Anweisung. Diese Lektion untersucht die fortgeschritteneren Merkmale der select-Anweisung: die group by-Klausel, die having-Klausel und die Join-Operation. Sie werden sehen, wie leistungsfähig die select-Anweisung wirklich ist.

Die Syntax der select-Anweisung ist recht schwer zu verstehen und Sie können die Verwendung ihrer Klauseln nicht wirklich verstehen, indem Sie lediglich die in der Dokumentation zu Oracle gezeigten Syntaxdiagramme studieren. Darum sollten Sie sich die vielen Beispiele in dieser Lektion anschauen. Ich werde Ihnen zeigen, was funktioniert und was nicht.

## Integrierte Gruppenfunktionen

Der erste Punkt, den wir besprechen, sind die integrierten Funktionen von Oracle, die mit Gruppen von Zeilen arbeiten. Seien Sie gewarnt: Dieses Thema ist ein gutes Beispiel für SQLs merkwürdige Eigenschaften. Obwohl diese Funktionen Gruppenfunktionen sind, ist bei ihnen die Verwendung der group by-Klausel *nicht* nötig.

- count zählt die Anzahl der Zeilen, die Oracle aufgrund der angegebenen Kriterien liefert.
- max und min geben die Minimal- und Maximalwerte für die angegebene Spalte zurück.
- avg und sum berechnen den Durchschnitt und die Summe über eine Spalte.
- stddev und variance berechnen Standardabweichung und Varianz für eine Spalte.

Jede dieser Funktionen gibt einen einzelnen Wert zurück. Es folgen einige ausführliche Beispiele.

### Die Funktion count

Die Funktion count gibt es in zwei Versionen: count(\*), die alle Zeilen in einer Tabelle zählt, die ein gewisses Kriterium erfüllen, und count(column-name), die alle Zeilen in einer Tabelle zählt, die einen Wert ungleich Null für column-name haben und den angegebenen Kriterien entsprechen.

Um die Verwendung von count(\*) zu demonstrieren, zeigt Listing 8.1, wie man die Anzahl der von der Abteilung für Biologie angebotenen Kurse zählt.

**Listing 8.1: Eine select-Anweisung mit einer where-Klausel.**

```
SQL> select count(*)  
2 from Course
```

```

3 where
4 Department_ID = 'BIO';

COUNT(*)
-----
3

```

Wenn Sie die Anzahl der Studenten zählen wollen, die eine Faxnummer besitzen, geben Sie, wie in Listing 8.2 gezeigt, Fax als Argument für die Funktion count an.

**Listing 8.2: Ermittlung der Anzahl von Zeilen in einer Tabelle mit Hilfe der Funktion count.**

```

SQL> select count(*)
2 from Student;

COUNT(*)
-----
31

SQL> select count(Fax)
2 from Student;

COUNT(FAX)
-----
3

```

Sie sollten beachten, daß die Funktion count eine Zeile zurückgibt, auch wenn die Zählung den Wert Null ergab.

## Ermitteln maximaler und minimaler Werte

Sie können die Funktionen max und min in einer einzigen select-Anweisung kombinieren. Listing 8.3 zeigt die Verwendung dieser Funktionen. Die erste Abfrage wird auf eine numerische Spalte angewandt - Additional\_Fees. Die zweite Abfrage wendet die Funktionen max und min auf eine Zeichenspalte an - Instructor\_ID.

**Listing 8.3: Verwendung der Funktionen min und max mit unterschiedlichen Datentypen.**

```

SQL> select max(Additional_Fees), min(Additional_Fees)
2 from Course;
MAX(ADDITIONAL_FEES) MIN(ADDITIONAL_FEES)
-----
750 0

```

```
SQL> select max(Instructor_ID), min(Instructor_ID)
2 from Instructor;
```

```
MAX(INSTRUCTOR_ID) MIN(INSTRUCTOR_ID)
-----
Y561 A612
```

## Verwendung der Funktionen avg und sum

avg und sum arbeiten auf dieselbe Weise wie min und max. Listing 8.4 zeigt ein Beispiel. Eine Abfrage der Tabelle Course liefert den Durchschnitt und die Summe der für Kurse berechneten Gebühren.

*Listing 8.4: Verwendung der Funktionen avg und sum.*

```
SQL> select avg(Additional_Fees), sum(Additional_Fees)
2 from Course;

AVG(ADDITIONAL_FEES) SUM(ADDITIONAL_FEES)
-----
46.171875 1477.5
```

Beachten Sie, daß die Funktion avg bei ihren Berechnungen keine Nullwerte verwendet.

## Kombinieren von Gruppenfunktionen mit anderen Spalten

Sie können Gruppenfunktionen und Spalten in der Auswahlliste einer select-Anweisung nicht ohne eine group by-Klausel kombinieren. Listing 8.5 bietet ein Beispiel einer Abfrage, in der die Auswahlliste aus zwei Teilen zusammengesetzt wird - Department\_ID und sum(Additional\_Fees). Oracle gibt hier jedoch einen Fehler zurück. Um zu verstehen warum, hilft es, in folgenden Begriffen zu denken: Eine Gruppenfunktion gibt eine einzelne Zeile zurück, Department\_ID gibt dagegen alle Zeilen der Tabelle zurück. Es ist einfach nicht sinnvoll, diese beiden Werte gleichzeitig zurückzugeben; Oracle gibt statt dessen einen Laufzeitfehler zurück. Indem Sie jedoch eine group by-Klausel verwenden, können Sie Gruppenfunktionen und Spalten in der Auswahlliste kombinieren. Die nächsten Abschnitte betrachten die Verwendung der group by-Klausel genauer.

*Listing 8.5: Fehlerhafte Syntax in einer select-Anweisung.*

```
SQL> select Department_ID, sum(Additional_Fees)
2 from Course;
select Department_ID, sum(Additional_Fees)
*
```

ERROR at line 1:  
ORA-00937: not a single-group group function

## Nach eindeutigen Zeilen suchen

Die select-Anweisung besitzt ein optionales Schlüsselwort, das wir bis jetzt noch nicht besprochen haben: **distinct**. Dieses Schlüsselwort folgt auf select und weist Oracle an, nur Zeilen zurückzugeben, die eindeutige Werte für die angegebenen Spalten haben. Als Beispiel verwendet Listing 8.6 eine Abfrage, um alle Städte zu erhalten, in denen Studenten leben.

*Listing 8.6: Abrufen einer Menge voneinander verschiedener Werte mit einer Abfrage.*

```
SQL> select distinct City
2 from Student;
```

```
CITY
-----
DOVER
SPRINGFIELD
```

Die Option **distinct** ist sehr nützlich, um die Wertemenge einer Spalte zu bestimmen. Sie bietet eine Methode, mit der man schnell feststellen kann, welche Cluster die Werte in einer Tabelle bilden. Wenn Sie die Option **distinct** nicht angeben, liefert Oracle alle Zeilen, die den in der where-Klausel angegebenen Kriterien entsprechen. Als Standard verwendet Oracle die Option **all**, um alle Spalten zurückzugeben. Die beiden folgenden select-Anweisungen sind deswegen äquivalent:

```
select City from Student;
```

und

```
select all City from Student;
```

## Gruppieren von Zeilen

Die **group by**-Klausel ist ein weiterer Abschnitt der select-Anweisung. Diese optionale Klausel teilt Oracle mit, daß es die Zeilen nach den unterschiedlichen Werten, die für die angegebenen Spalten existieren, gruppieren soll. Zusätzlich kann die Klausel **having** in Verbindung mit der **group by**-Klausel verwendet werden, um die Menge der gelieferten Zeilen weiter einzuschränken.



Obwohl dieses Thema am besten durch ein Beispiel erklärt wird, werfen wir zuerst einen kurzen Blick auf die Syntax.

---

```
SELECT select-list
FROM table-list
[WHERE
condition [AND | OR]
...
condition]
[GROUP BY column1, column2, ..., columnN]
[HAVING condition]
[ORDER BY column1, column2, ..., columnN]
```

Die Variablen sind folgendermaßen definiert:

- *select-list* ist eine Menge von Spalten und Ausdrücken aus den in *table-list* aufgelisteten Tabellen.
  - *table-list* sind die Tabellen, aus denen die Zeilen abgefragt werden.
  - *condition* ist eine zulässige Bedingung in Oracle-SQL.
  - *column1* bis *columnN* sind Spalten der in *table-list* enthaltenen Tabellen.
- 



*Es ist einfach, diese Elemente - Spalten, Ausdrücke, Tabellen und Bedingungen - in einer SQL-Anweisung so zu mischen, daß Oracle diese zurückweist. Außerdem können Sie Anweisungen konstruieren, die Oracle ohne Fehler bearbeitet, deren Resultate aber schwierig zu interpretieren sind. Wenn diese Dinge passieren, ist es am besten, zu den Grundlagen zurückzugehen. Studieren Sie Ihre select-Anweisung. Wenn sie keinen Sinn mehr ergibt, verwenden Sie SQL\*Plus oder ein ähnliches Werkzeug, um sie Element für Element auseinanderzunehmen, bis wieder ein Sinn sichtbar wird. Analysieren Sie Ihre Verwendung von Gruppenfunktionen, die group by- und having-Klauseln sowie alle Join-Bedingungen.*

---

Listing 8.7 gibt ein Beispiel einer Abfrage, die eine Liste von Department\_IDs aus der Tabelle Course liefert.

**Listing 8.7: Beispiel für eine group by-Klausel.**



```
SQL> select Department_ID
2 from Course
3 group by Department_ID;
```

```
DEPARTMENT_ID
```

```
-----
```

```
ANTHRO
```

```
BIO
```

```
ECON
```

```
ENG
```

```
ENGL
```

```
HIST
```

```
MATH
```

```
PHILO
```

```
PSYCH
```

```
9 rows selected.
```

Listing 8.8 zeigt, wie man die Anzahl der Kurse zählt, die mit jeder Department\_ID assoziiert sind.

***Listing 8.8: Verwendung der Funktion count zusammen mit einer group by-Klausel.***

```
SQL> select Department_ID, count(*)
2 from Course
3 group by Department_ID;
```

```
DEPARTMENT_ID COUNT(*)
```

```
-----
```

```
ANTHRO 3
```

```
BIO 3
```

```
ECON 3
```

```
ENG 4
```

```
ENGL 4
```

```
HIST 3
```

```
MATH 4
```

```
PHILO 3
```

```
PSYCH 5
```

```
9 rows selected.
```

Als nächstes betrachten wir die gleichzeitige Verwendung der Klauseln group by und having. Sie können die having-Klausel verwenden, um die Abteilungen zu bekommen, die genau vier Kurse haben, wie in Listing 8.9 gezeigt wird.

***Listing 8.9: Verwendung der group by- und having-Klauseln.***

```
SQL> select Department_ID, count(*)
2 from Course
3 group by Department_ID
4 having count(*) = 4;
```

```
DEPARTMENT_ID COUNT(*)
```

```
-----
```

```
ENG 4
ENGL 4
MATH 4
```

Gruppenfunktionen können in der Auswahlliste ohne die group by-Klausel nicht mit Spalten kombiniert werden. Sie können beispielsweise den Durchschnittswert für zusätzliche Gebühren pro Abteilung erhalten, wie es in Listing 8.10 gezeigt wird.

*Listing 8.10: Kombination der Funktion avg mit der group by-Klausel.*

```
SQL> select Department_ID, avg(Additional_Fees)
2 from Course
3 group by Department_ID;
```

```
DEPARTMENT_ID AVG(ADDITIONAL_FEES)
```

```
-----
```

```
ANTHRO 20.833333
BIO 63.333333
ECON 258.33333
ENG 50
ENGL 0
HIST 0
MATH 2.5
PHILO 0
PSYCH 48
```

```
9 rows selected.
```

Im nächsten Beispiel werden Sie eine weitere Anwendung für Datenbanken kennenlernen: ein Informationssystem für die Patienten eines Krankenhauses. Eine weitere Verwendung der group by- und having-Klauseln ist die Identifikation doppelt vorhandener Zeilen. Es kann beispielsweise vorkommen, daß Sie keinen Primärschlüssel zu einer Tabelle hinzufügen können, weil die Spalte für den primären Schlüssel doppelte Werte enthält, wie in Listing 8.11 gezeigt wird. Eine Abfrage, die die Klauseln group by und having verwendet, stellt fest, welche Patient\_IDs mehr als einmal auftreten. Eine zweite Abfrage bestimmt, welche Zeile gelöscht werden soll. Nachdem sie gelöscht wurde, wird der Primärschlüssel erfolgreich zur Tabelle hinzugefügt.

*Listing 8.11: Ermittlung doppelter Zeilen.*

```
SQL> alter table Patient add
2 constraint Patient_PK
3 primary key (Patient_ID);
alter table Patient add
*
```

```
ERROR at line 1:
ORA-02299: cannot add or enable constraint (TYO.PATIENT_PK)-
duplicate keys found
```

```
SQL> select Patient_ID
2 from Patient
3 having count(*) > 1
4 group by Patient_ID;
PATIEN
-----
GG9999
```

```
SQL> select *
2 from Patient
3 where
4 Patient_ID = 'GG9999';
PATIEN BODY_TEMP_DEG_F FEVER_CLASS
-----
GG9999 107.6 LETHAL FEVER
GG9999 107
```

```
SQL> delete from Patient
2 where Patient_ID = 'GG9999' and Body_Temp_Deg_F = 107;
1 row deleted.
```

```
SQL> alter table Patient add
2 constraint Patient_PK
3 primary key (Patient_ID);
Table altered.
```

## Der Umgang mit hierarchischen Informationen

Um Verwirrung zu vermeiden, habe ich das Thema der hierarchischen Informationen in diese und nicht in eine der früheren Lektionen über SQL aufgenommen. Ein Beispiel für hierarchische Daten ist eine Bill of Materials (BOM), die alle Teile, Teilmontagen und Montagen zusammenfaßt, aus denen ein fertiges Produkt besteht; ein Flugzeughersteller kann eine BOM haben, die Tausende von Ebenen und Millionen von Teilen umfaßt.



*Hierarchische Daten sind Informationen in einer Tabelle, die mindestens eine Spalte und einen übergeordneten Schlüssel besitzt und eine weitere Spalte, die den Primärschlüssel der jeweiligen Zeile enthält.*

Können relationale Datenbanken im allgemeinen und SQL im besonderen diese Arten von Hierarchien unterstützen? Die Antwort ist: Jein. Die gute Nachricht ist, daß SQL durch die connect by-Klausel einige Unterstützung für hierarchische Daten bietet. Die schlechte Nachricht ist, daß diese Unterstützung recht begrenzt ist und die Syntax nicht intuitiv. Ich zeige Ihnen trotzdem an einem Beispiel, wie man SQL verwendet, um durch hierarchische Daten zu navigieren.

Eine dreizeilige Tabelle zeigt die Verwendung der connect by-Klausel:

|             |                                                        |
|-------------|--------------------------------------------------------|
| Product_ID  | Das jeweilige Produkt                                  |
| Assembly_ID | Die Montage oder Teilmontage, zu der Product_ID gehört |
| Description | Die Produktbeschreibung                                |

Das Design der Tabelle basiert auf dem Konzept, daß jedes Produkt zum Endprodukt oder einer Teilmontage gehört - außer dem Endprodukt selbst. Die Zeilen in der Tabelle Product\_Assembly zeigen dieses Konzept. Eine Kamera, die X1000, wird aus mehreren Teilen und einer Teilmontage, B200, zusammengesetzt. Listing 8.12 zeigt den Inhalt der Tabelle Product\_Assembly.

***Listing 8.12: Inhalt der Product\_Assembly-Tabelle.***

```
SQL> select Assembly_ID, Product_ID, Description
2 from Product_Assembly
3 order by Assembly_ID, Product_ID;
ASSEMBLY_ID PRODUCT_ID DESCRIPTION
```

```
-----
B200 I101 Titanium alloy, teflon-coated iris
B200 S42 Variable-speed shutter, standard
B200 W123 Manual film advance, winder
X1000 B200 Black body, stainless steel frame camera body
X1000 F100 Blue filter - standard
X1000 F55 Xenon flash unit
X1000 L100 100mm lens - standard
X1000 S04 4 foot camera strap, leather
X1000 Complete camera
```

9 rows selected.

Aufgrund der geringen Menge der hier gezeigten Daten können Sie sich die Organisation der Teile leicht klarmachen. Der Sinn dieses Beispiels ist, Ihnen zu zeigen, wie man den Aufbau einer Hierarchie abfragt.

Listing 8.13 bietet ein Beispiel dafür, wie man hierarchische Daten mit der select-Anweisung abfragt. Die select-Anweisung enthält zwei neue Klauseln: start with und connect by. Die start with-Klausel identifiziert die Zeile der obersten Ebene, für die Oracle untergeordnete Zeilen abfragen soll - in diesem Beispiel wollen Sie die untergeordneten Zeilen für die X1000 abfragen. Die connect by-Klausel teilt Oracle mit, wie die Zeilen präsentiert werden sollen, und der Operator prior weist Oracle an, die Zeilen so darzustellen, daß die Assembly\_ID jeder untergeordneten Zeile gleich der Product\_ID ihrer übergeordneten Zeile ist.

***Listing 8.13: Abrufen von hierarchischen Informationen aus einer Tabelle.***

```
SQL> select lpad(' ',2*(level-1)) || Assembly_ID Assembly_ID,
2 Product_ID, Description
3 from Product_Assembly
4 start with Product_ID = 'X1000'
5 connect by prior Product_ID = Assembly_ID;
ASSEMBLY_ID PRODUCT_ID DESCRIPTION
-----
X1000 Complete camera
X1000 L100 100mm lens - standard
X1000 B200 Black body, stainless steel frame camera body
B200 S42 Variable-speed shutter, standard
B200 I101 Titanium alloy, teflon-coated iris
B200 W123 Manual film advance, winder
X1000 S04 4 foot camera strap, leather
X1000 F100 Blue filter - standard
X1000 F55 Xenon flash unit
9 rows selected.
```



Um jede Zeile auf ihre hierarchische Ebene einzurücken, biete ich level in die Funktion lpad ein, so daß die Anzahl der von lpad zurückgegebenen Leerezeichen der hierarchischen Ebene entspricht. (Level ist eine Pseudospalte, welche die hierarchische Ebene für jede Zeile zurückgibt - von 1 für die oberste bis N für die detaillierteste Ebene.). lpad wird mit Assembly\_ID verkettet.



*Seien Sie bei der Konstruktion Ihres Datenmodells vorsichtig. Seien Sie sich wirklich sicher, daß Sie hierarchische Informationen unterstützen müssen, bevor Sie connect by und start with verwenden. Für hierarchische Abfragen gibt es Einschränkungen. Eine hierarchische Abfrage kann beispielsweise keinen Join durchführen.*

---

Um eine bestimmte Teilmontage zu betrachten, geben Sie in der start with-Klausel die Assembly\_ID an. Die select-Anweisung in Listing 8.14 zeigt diese Prozedur; sie zeigt auch den Wert von level für jede Zeile.

**Listing 8.14: Einrücken der Ausgabe einer Hierarchieabfrage mit Hilfe von level.**

```
SQL> select lpad(' ',2*(level-1)) || Assembly_ID Assembly_ID,
2 Product_ID, Level, Description
3 from Product_Assembly
4 start with Product_ID = 'B200'
5 connect by prior Product_ID = Assembly_ID;
ASSEMBLY_ID PRODUCT_ID LEVEL DESCRIPTION
```

```
-----
X1000 B200 1 Black body, stainless steel frame camera body
B200 S42 2 Variable-speed shutter, standard
B200 I101 2 Titanium alloy, teflon-coated iris
B200 W123 2 Manual film advance, winder
```

Sie sollten sich über eine Reihe von Dingen im klaren sein, wenn Sie hierarchische select-Anweisungen verwenden. Als erstes kann die hierarchische select-Anweisung auch für einen Join von zwei oder mehr Tabellen verwendet werden (das wird später in dieser Lektion besprochen). Zweitens, wenn Sie eine order by-Klausel angeben, zerstören Sie die hierarchische Ordnung der von der Abfrage zurückgegebenen Zeilen.

### Was Sie tun sollen

Verwenden Sie eine hierarchische Struktur für eine Tabelle, wenn diese hierarchische Informationen auf mehreren Ebenen repräsentiert. Die Struktur sollte eine Spalte mit einem übergeordneten Schlüssel enthalten.

### Was nicht

Verwenden Sie keine hierarchischen Schlüssel, wenn die von Ihnen modellierte Information keine verschiedenen hierarchischen Ebenen enthält.

## Verwenden des exists-Operators

Zusätzlich zu den anderen SQL-Operatoren sollten Sie auch mit dem exists-Operator vertraut sein. Dieser Operator arbeitet mit einer Unterabfrage und gibt einen boolschen Wert zurück:

- true, wenn die Unterabfrage mindestens eine Zeile zurückgibt
- false, wenn die Unterabfrage keine Zeilen zurückgibt

Beachten Sie das in Listing 8.15 gezeigte Beispiel, das auf den Tabellen Class und Instructor basiert. Stellen Sie sich vor, daß Sie die Namen der Lehrer feststellen wollen, die für einen Kurs eingeteilt sind. Dafür verwenden Sie eine Abfrage mit dem exists-Operator.

### *Listing 8.15: Verwendung des exists-Operators.*

```
SQL> select Last_Name, First_Name, Position
2 from Instructor I
3 where
4 exists
5 (select * from Class C
6 where
7 I.Instructor_ID = C.Instructor_ID);
```

```
LAST_NAME FIRST_NAME POSITION
```

```
-----
CHANG ROGER ASSISTANT PROFESSOR
JASON JERROLD ASSOCIATE PROFESSOR
TORRES PETER PROFESSOR
RICHARDSON NANCY ASSISTANT PROFESSOR
PARKER WILLIAM PROFESSOR
CHERNOW BESS ASSOCIATE PROFESSOR
CHU STEVEN PROFESSOR
WEISS ROBERTA PROFESSOR
RESTON RAPHAEL ASSOCIATE PROFESSOR
```

```
9 rows selected.
```

Der exists-Operator ist in Situationen nützlich, in denen Sie sich nicht für die von der Unterabfrage zurückgegebenen Spaltenwerte interessieren. Beachten Sie, wie Aliasnamen einer Tabelle - C für Class und I für Instructor - verwendet wurden, um den Tippaufwand zu verringern und die Lesbarkeit der Abfrage zu verbessern. Sie können dem exists-Operator auch den logischen Operator not voranstellen.

Als Beispiel zeigt Listing 8.16, wie Sie eine Liste von Lehrern erhalten, die *nicht* für eine Klasse eingeteilt sind.



Ein Alias ist ein in einer Abfrage verwendeter alternativer Name für eine Tabelle, eine Spalte oder einen Ausdruck. Ein Alias existiert nur während der Abfrage. Sie können ein Alias verwenden, um eine Abfrage zu vereinfachen oder klarer auszudrücken.

**Listing 8.16: Ein weiteres Beispiel für die Verwendung des exists-Operators.**

```
SQL> select Last_Name, First_Name, Position
2 from Instructor I
3 where
4 not exists
5 (select * from Class C
6 where
7 I.Instructor_ID = C.Instructor_ID);
```

```
LAST_NAME FIRST_NAME POSITION
-----
HITCHCOCK BORIS PROFESSOR
DANIELS LAURA ASSOCIATE PROFESSOR
EDWARDS SAMANTHA PROFESSOR
ANGELO ARTHUR ASSOCIATE PROFESSOR
BILLINGS BENJAMIN ASSISTANT PROFESSOR
YOUNG ALLAN ASSOCIATE PROFESSOR
NILAND MARTINA ASSOCIATE PROFESSOR
BATES JOSEPH ASSISTANT PROFESSOR
POULSON RANIER PROFESSOR

9 rows selected.
```



Wie Sie in Zeile 5 sehen können, fragt eine Unterabfrage für jede Klasse die Spalten aus der Tabelle Class ab, die der durch Instructor\_ID identifizierte Lehrer unterrichtet. Der Operator not exists führt dazu, daß die Abfrage Last\_Name, First\_Name und Position genau dann nicht zurückgibt, wenn die Unterabfrage die entsprechenden Zeilen zurückgibt.



---

## Die Join-Operation

Nun, da Sie verwertbare Kenntnisse von SQL besitzen, sind Sie soweit, daß Sie in die Welt der Joins eintauchen können. Ein *Join* liefert Zeilen aus zwei oder mehr Tabellen, die eine gemeinsame Menge von Merkmalen teilen. Ein relationale Datenbank hätte ohne die Join-Operation wenig Wert.

---



*Ein Join ist eine Abfrage, die Werte aus zwei oder mehr Tabellen abfragt, in denen eine oder mehrere Spalten der einen Tabelle gleich sind wie eine oder mehrere Spalten der anderen Tabelle.*

---

Die *Join-Operation* ist der Mechanismus, der es möglich macht, daß Tabellen miteinander verknüpft sind. Eine Join-Operation fragt Spalten aus zwei oder mehr Tabellen ab. Um beispielsweise zwei Tabellen zu verknüpfen, gibt das Abfragekriterium typischerweise die Bedingung an, daß eine Spalte der ersten Tabelle - die als Fremdschlüssel definiert ist - gleich einer Spalte der zweiten Tabelle ist - die ein Primärschlüssel ist, auf den der Fremdschlüssel verweist. Die where-Klausel eines Joins kann zusätzliche Bedingungen erhalten. Dieser Typ von Join-Operation wird als *Equi-Join* bezeichnet. In der Beispieldatenbank können Sie z.B. die Tabellen Instructor und Department verknüpfen, so daß Sie zusammenhängende Informationen aus beiden Tabellen mit einer einzigen SQL-Anweisung erhalten können.



Die allgemeine Syntax für die select-Anweisung ermöglicht es Ihnen, mehr als zwei Tabellen zu verknüpfen.

```
SELECT select-list
FROM table1, table2, ... , tableN
WHERE table1.column1 = table2.column2 and
...
table2.column3 = tableN.columnN
...
additional-conditions
```



Die Variablen sind folgendermaßen definiert:

- *select-list* ist die Menge der Spalten und Ausdrücke von *table1* bis *tableN*.
- *table1* bis *tableN* sind die Tabellen, aus denen die Spaltenwerte abgefragt werden.
- *column1* bis *columnN* sind die in *table1* bis *tableN* miteinander verknüpften Spalten.
- *additional-conditions* sind die optionalen Abfragekriterien.

Beachten Sie bitte, daß Sie *column1* bis *columnN* nicht in der *select-list* angeben müssen.

## Ein einfacher Join mit zwei Tabellen

Diese Syntax kann durch einen einfachen Verbund von zwei Tabellen demonstriert werden. Die Tabelle *Class* kennzeichnet durch *Instructor\_ID* den Lehrer einer Klasse. Die Tabelle *Instructor* enthält ausführliche Informationen über jeden Lehrer. Durch einen Verbund der Tabellen *Class* und *Instructor* mittels der Spalte *Instructor\_ID* können Sie zusätzliche Daten über den Lehrer abfragen.

### *Listing 8.17: Ein einfacher Join mit zwei Tabellen.*

```
SQL> select Class_ID, Course_ID, Last_Name, First_Name, Position
2 from Class C, Instructor I
3 where
4 C.Instructor_ID = I.Instructor_ID
5 order by Class_ID;
```

```
CLASS_ID COURS LAST_NAME FIRST_NAME POSITION
-----
103400 183 JASON JERROLD ASSOCIATE PROFESSOR
103600 50 CHANG ROGER ASSISTANT PROFESSOR
104200 198 RICHARDSON NANCY ASSISTANT PROFESSOR
104500 184 RESTON RAPHAEL ASSOCIATE PROFESSOR
108300 101 CHERNOW BESS ASSOCIATE PROFESSOR
108400 115 TORRES PETER PROFESSOR
108600 183 JASON JERROLD ASSOCIATE PROFESSOR
109100 101 WEISS ROBERTA PROFESSOR
110300 199 CHU STEVEN PROFESSOR
120200 199 PARKER WILLIAM PROFESSOR
```

10 rows selected.

Sehen Sie sich die Elemente der select-Anweisung an. Die Auswahlliste besteht aus fünf Spalten: *Class\_ID*, *Course\_ID*, *Last\_Name*, *First\_Name* und *Position*. Die Tabellen in der from-Klausel sind

Class und Instructor. Die where-Klausel weist die Oracle-RDBMS an, nur Zeilen zurückzugeben, in denen die Instructor\_ID der Tabelle Class mit einer Zeile in der Tabelle Instructor übereinstimmt, die denselben Wert für Instructor\_ID aufweist.

Es gibt keine Einschränkung für die Reihenfolge der Spalten in der select-list. Es ist auch nicht nötig, daß überhaupt eine oder mehrere Zeilen aus jeder der verknüpften Tabellen ausgewählt werden. Als Beispiel zeigt Listing 18.8 eine zulässige Verknüpfung, in der die Auswahlliste keine Spalten aus der Tabelle Class enthält.

**Listing 8.18: Bildung eines Joins über eine Auswahlliste, die Spalten einer einzigen Tabelle enthält.**

```
SQL> select Last_Name, First_Name, Position
2 from Class C, Instructor I
3 where
4 C.Instructor_ID = I.Instructor_ID
5 order by Class_ID;
```

```
LAST_NAME FIRST_NAME POSITION
```

```
-----
JASON JERROLD ASSOCIATE PROFESSOR
CHANG ROGER ASSISTANT PROFESSOR
RICHARDSON NANCY ASSISTANT PROFESSOR
RESTON RAPHAEL ASSOCIATE PROFESSOR
CHERNOW BESS ASSOCIATE PROFESSOR
TORRES PETER PROFESSOR
JASON JERROLD ASSOCIATE PROFESSOR
WEISS ROBERTA PROFESSOR
CHU STEVEN PROFESSOR
PARKER WILLIAM PROFESSOR
```

```
10 rows selected.
```

## Mehrdeutige Spalten

Jeder Verweis auf eine Spalte in einer Verknüpfung muß eindeutig sein. In diesem Zusammenhang bedeutet *eindeutig*, daß der Spaltenname durch den Tabellennamen gekennzeichnet werden muß, wenn eine Spalte in mehr als einer Tabelle der Verknüpfung vorhanden ist. Oracle gibt eine Fehlermeldung zurück, wenn Sie mehrdeutig auf eine Spalte verweisen. Listing 8.19 zeigt eine Verknüpfung, die eine mehrdeutige Spalte enthält - Instructor\_ID.

**Listing 8.19: Mehrdeutige Spalte in einem Join.**

```
SQL> select Instructor_ID, Position
2 from Class C, Instructor I
```

```

3 where
4 C.Instructor_ID = I.Instructor_ID
5 order by Class_ID;
select Instructor_ID, Position
*
ERROR at line 1:
ORA-00918: column ambiguously defined

```

Oracle gibt die Fehlermeldung zurück, weil `Instructor_ID`, die Teil der Auswahlliste ist, in den Tabellen `Class` und `Instructor` auftritt. Um dieses Problem zu korrigieren, müssen Sie die Spalte `Instructor_ID` mit dem Namen oder dem Alias der Tabelle kennzeichnen. Listing 8.20 zeigt, wie eine mehrdeutige Spalte in einer Auswahlliste gekennzeichnet werden kann.

*Listing 8.20: Kennzeichnung einer mehrdeutigen Spalte in der Auswahlliste.*

```

SQL> select I.Instructor_ID, Position
2 from Class C, Instructor I
3 where
4 C.Instructor_ID = I.Instructor_ID
5 order by Class_ID;

```

```

INSTRUCTOR_ID POSITION
-----

```

```

J505 ASSOCIATE PROFESSOR
E405 ASSISTANT PROFESSOR
E491 ASSISTANT PROFESSOR
D944 ASSOCIATE PROFESSOR
T149 ASSOCIATE PROFESSOR
G331 PROFESSOR
J505 ASSOCIATE PROFESSOR
W490 PROFESSOR
S131 PROFESSOR
R983 PROFESSOR

```

```

10 rows selected.

```

## Vorsicht vor dem kartesischen Produkt

Wenn Sie das erste Mal lernen, wie man mehrere Tabellen verknüpft, ist ein häufiger Fehler, daß man vergißt, in der `where`-Klausel eine Verknüpfungsbedingung anzugeben. Wenn Sie die Verknüpfungsbedingung vergessen, werden Sie zwei Dinge feststellen: Die Abfrage braucht erheblich länger für ihre Ausführung, und die Anzahl der gelieferten Datensätze ist erheblich größer, als Sie erwartet haben.



*Der technische Ausdruck für eine Verknüpfung ohne Verknüpfungsbedingung ist kartesisches Produkt.*

Beachten Sie das in Listing 8.21 gezeigte Beispiel. Zwei Tabellen, Class und Instructor, werden in der from-Klausel ohne Verknüpfungsbedingung aufgelistet. Die Tabelle Class hat 10 Zeilen, die Tabelle Location hat 13 Zeilen. Die Abfrage hat keine Verknüpfungsbedingung, die Oracle anweisen würde, die Spalten jeder Tabelle aufgrund von Class\_Building und Class\_Room abzufragen. Als Ergebnis gibt Oracle alle möglichen Kombinationen der zwei Tabellen zurück, dies sind 10mal 13, oder 130 Zeilen.

***Listing 8.21: Ein von einem Join ohne Join-Bedingung erzeugtes kartesisches Produkt.***

```
SQL> select Class_ID, Seating_Capacity
2 from Class, Class_Location
3 order by Class_ID;
```

```
CLASS_ID SEATING_CAPACITY
-----
103400 200
103400 120
103400 80
103400 45
103400 35
103400 45
103400 45
103400 90
103400 50
...
108600 200
108600 80
108600 90
108600 50
109100 45
109100 40
110300 50
110300 45
110300 40
110300 45
110300 40
110300 35
120200 80
```

130 rows selected.

## Join-Bildung mit mehreren Tabellen

Wie Sie sehen, schränkt die Syntax der select-Anweisung die Zahl der Tabellen nicht ein, die miteinander verknüpft werden können. Als Beispiel sehen Sie Listing 8.22, in dem drei Tabellen verknüpft werden - Class, Course und Instructor - um detaillierte Informationen über jede Klasse abzufragen.

### *Listing 8.22: Ein Join mit drei Tabellen.*

```
SQL> select Class_ID, CL.Course_ID, CL.Department_ID, Title,
Last_Name
2 from Class CL, Course CO, Instructor I
3 where
4 CL.Course_ID = CO.Course_ID and
5 CL.Department_ID = CO.Department_ID and
6 CL.Instructor_ID = I.Instructor_ID
7 order by Class_ID;
```

```
CLASS_ID  COURS  DEPARTME  TITLE  LAST_NAME
-----
103400  183  PSYCH  WORKSHOP ON NORMALITY  JASON
103600  50   MATH  PRE-CALCULUS  CHANG
104200  198  PHILO  MODERN PHILOSOPHY  RICHARDSON
104500  184  HIST  MODERN EUROPEAN HISTORY  RESTON
108300  101  ENGL  INTRO TO ENGLISH LIT  CHERNOW
108400  115  HIST  EARLY AMERICAN HISTORY  TORRES
108600  183  PSYCH  WORKSHOP ON NORMALITY  JASON
109100  101  BIO  INTRO TO BIOLOGY  WEISS
110300  199  ENG  SEMINAR ON CHAOS  CHU
120200  199  ECON  WORKSHOP ON MARX  PARKER
```

10 rows selected.

Beachten Sie, daß für jede der drei Tabellen ein Alias angegeben wurde:

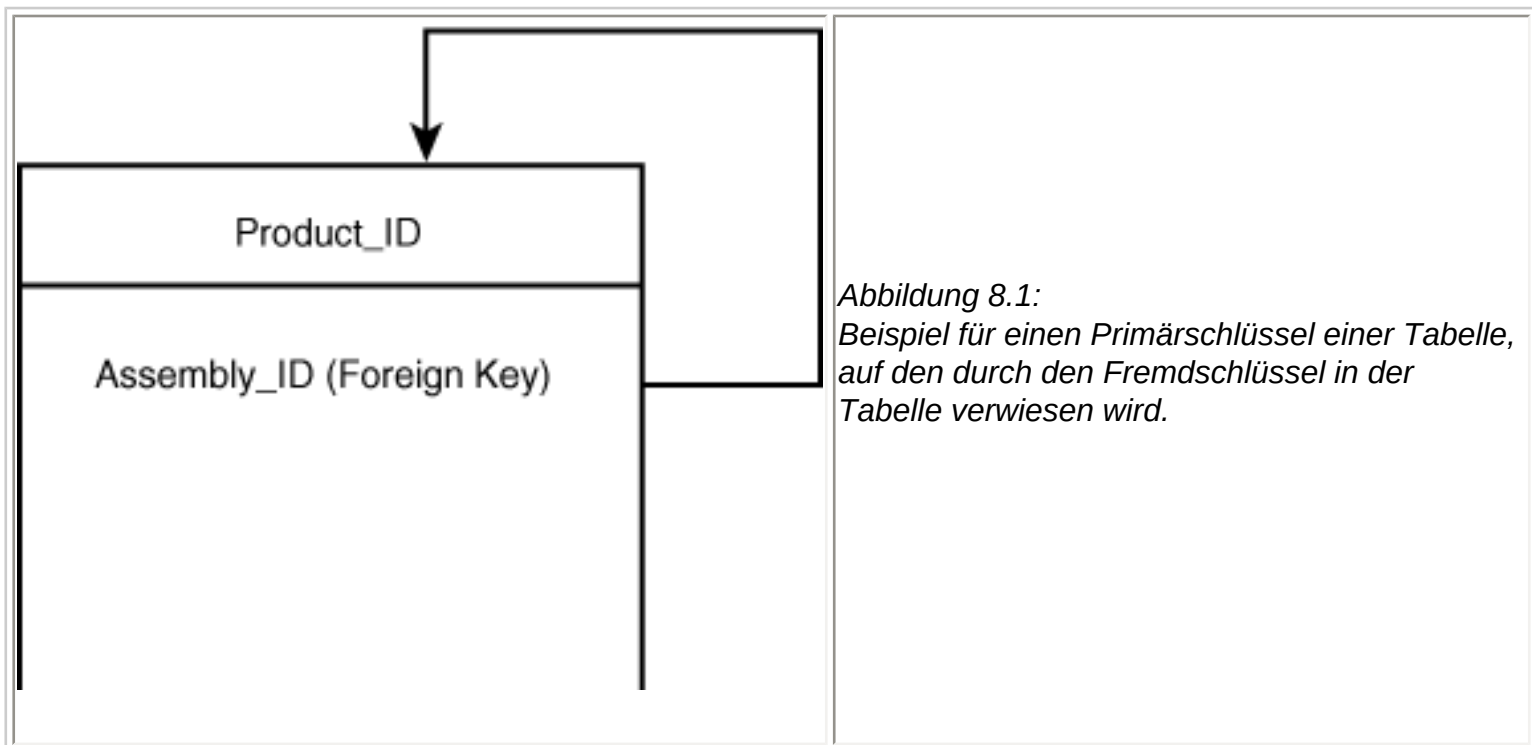
- cl für Class
- co für Course
- i für Instructor

Obwohl Aliase für Tabellen optional sind, sollten Sie sie bei der Verknüpfung mehrerer Tabellen verwenden, weil sie die Größe der select-Anweisung reduzieren und sie leichter lesbar machen.

Studieren Sie jede der Verknüpfungsbedingungen in Listing 8.22. Die ersten beiden Verknüpfungsbedingungen werden verwendet, um die Tabellen Class und Course über Course\_ID und Department\_ID zu verbinden. Diese Tabellen haben zwei Verknüpfungsbedingungen, da Course\_ID und Department\_ID der Primärschlüssel der Tabelle Course sind. Die dritte Verknüpfungsbedingung wird verwendet, um die Tabellen Class und Instructor über Instructor\_ID zu verknüpfen.

## Self-Joins

Eine andere Form der Verknüpfung von zwei Tabellen ist der *Self-Join*. Diese Art der Verknüpfung wird verwendet, wenn eine Tabelle einen Fremdschlüssel besitzt, der sich auf ihren eigenen Primärschlüssel bezieht. Ein gutes Beispiel für diesen Typ ist die Tabelle Product\_Assembly, die früher in dieser Lektion besprochen wurde. Der Primärschlüssel der Tabelle Product\_Assembly ist Product\_ID. Jedes Produkt - außer dem Endprodukt - gehört zu einer Montage. Deshalb ist Assembly\_ID ein Fremdschlüssel, der auf Product\_ID verweist (siehe Abbildung 8.1).



Als Beispiel für einen Self-Join stellen Sie sich vor, daß Sie die Beschreibung für jedes Produkt und die Beschreibung der Montage, zu der es gehört, abfragen wollen. Wie Listing 8.23 zeigt, muß die Join-Anweisung ein Alias für beide Kopien der Tabelle definieren. Die Verknüpfungsbedingung - P1.Assembly\_ID = P2.Assembly\_ID - veranlaßt Oracle, Produktinformation aus P1 und Montageinformation aus P2 abzufragen.

**Listing 8.23: Beispiel für einen Self-Join.**

```
SQL> select P1.Description || ' is part of ' || P2.Description
"Assembly Breakdown"
```

```

2 from Product_Assembly P1, Product_Assembly P2
3 where
4 P1.Assembly_ID = P2.Product_ID
5 order by P1.Assembly_ID, P1.Product_ID;
Assembly Breakdown

```

```

-----
Titanium alloy, teflon-coated iris is part of Black body, stainless
ste
Variable-speed shutter, standard is part of Black body, stainless
steel
Manual film advance, winder is part of Black body, stainless steel
fram
Black body, stainless steel frame camera body is part of Complete
camer
Blue filter - standard is part of Complete camera
Xenon flash unit is part of Complete camera
100mm lens - standard is part of Complete camera
4 foot camera strap, leather is part of Complete camera
8 rows selected.

```

## Outer-Join

In diesem Abschnitt werden Sie die Verwendung eines weiteren Typs von Join sehen - des Outer-Join. Um den Unterschied zwischen einem gewöhnlichen Join (also einem Equi-Join) und einem Outer-Join zu verstehen, lassen Sie uns einen weiteren Blick auf gewöhnliche Joins werfen. Ein Join über mehrere Tabellen gibt Tabellen aus allen Tabellen zurück, in denen die Verknüpfungsbedingungen zutreffen. Beachten Sie das in Listing 8.24 gezeigte Beispiel. Die Lehrer am Flugle-College können während eines Semesters für eine bestimmte Klasse eingeteilt sein oder nicht. Zusätzlich zu ihrer Lehrtätigkeit wird von den Mitarbeitern auch erwartet, daß sie sich an der Forschung auf ihrem Gebiet beteiligen. Listing 8.24 enthält einen Join der Tabellen Class und Instructor. Beachten Sie, daß die Abfrage nur Lehrer liefert, die eine Klasse unterrichten.

### *Listing 8.24: Equi-Join der Instructor- und Class-Tabellen.*

```

SQL> select I.Department_ID, Last_Name, First_Name, Class_ID
2 from Instructor I, Class C
3 where
4 I.Instructor_ID = C.Instructor_ID
5 order by I.Department_ID, Last_Name;

```

```

DEPARTME LAST_NAME FIRST_NAME CLASS_ID
-----
BIO WEISS ROBERTA 109100
ECON PARKER WILLIAM 120200

```



```

ENG CHU STEVEN 110300
ENGL CHERNOW BESS 108300
HIST RESTON RAPHAEL 104500
HIST TORRES PETER 108400
MATH CHANG ROGER 103600
PHILO RICHARDSON NANCY 104200
PSYCH JASON JERROLD 108600
PSYCH JASON JERROLD 103400

```

10 rows selected.

Wenn Sie Informationen über Lehrer abfragen wollen, ob ein Lehrer eine Klasse unterrichtet oder nicht, müssen Sie den Outer-Join-Operator verwenden, indem Sie ein (+) (ein Pluszeichen in Klammern) an die optionale Zeile der Verknüpfungsbedingung anhängen. Listing 8.25 zeigt die Verwendung des Outer-Join-Operators.

***Listing 8.25: Ein Outer-Join.***

```

SQL> select I.Department_ID, Last_Name, First_Name, Class_ID
2 from Instructor I, Class C
3 where
4 I.Instructor_ID = C.Instructor_ID (+)
5 order by I.Department_ID, Last_Name;

```

```

DEPARTMENT_ID LAST_NAME FIRST_NAME CLASS_ID
-----

```

```

ANTHRO DANIELS LAURA
ANTHRO HITCHCOCK BORIS
ANTHRO POULSON RANIER
BIO EDWARDS SAMANTHA
BIO WEISS ROBERTA 109100
ECON PARKER WILLIAM 120200
ENG CHU STEVEN 110300
ENGL CHERNOW BESS 108300
HIST RESTON RAPHAEL 104500
HIST TORRES PETER 108400
MATH CHANG ROGER 103600
MATH NILAND MARTINA
MATH YOUNG ALLAN
PHILO ANGELO ARTHUR
PHILO BILLINGS BENJAMIN
PHILO RICHARDSON NANCY 104200
PSYCH BATES JOSEPH
PSYCH JASON JERROLD 108600
PSYCH JASON JERROLD 103400

```

19 rows selected.

Wenn ein Lehrer eine Klasse nicht unterrichtet, gibt Oracle eine Null für Class\_ID - oder jede andere Spalte, die Sie aus der Tabelle Class auswählen - zurück. Listing 8.26 zeigt, wie Sie eine Liste von Lehrern erhalten können, die keine Klasse unterrichten, indem Sie eine weitere Bedingung zur where-Klausel hinzufügen - Class\_ID is null.

***Listing 8.26: Ein weiterer Outer-Join.***

```
SQL> select I.Department_ID, Last_Name, First_Name
2 from Instructor I, Class C
3 where
4 I.Instructor_ID = C.Instructor_ID (+) and
5 Class_ID is null
6 order by I.Department_ID, Last_Name;
```

```
DEPARTMENT_ID LAST_NAME FIRST_NAME
```

```
-----
ANTHRO DANIELS LAURA
ANTHRO HITCHCOCK BORIS
ANTHRO POULSON RANIER
BIO EDWARDS SAMANTHA
MATH NILAND MARTINA
MATH YOUNG ALLAN
PHILO ANGELO ARTHUR
PHILO BILLINGS BENJAMIN
PSYCH BATES JOSEPH
```

```
9 rows selected.
```

## Verwendung von Mengenoperatoren in einer select-Anweisung

Die Sprache SQL ist eine teilweise Implementation des relationalen Modells, wie es von Codd, dem Vater der relationalen Theorie, entworfen wurde: intersect, union und minus.

### Der intersect-Operator

Der intersect-Operator gibt die Zeilen zurück, die zwei Mengen von Zeilen gemeinsam haben.



Die Syntax für die Verwendung des intersect-Operators ist

```
select-stmt1
INTERSECT
select-stmt2
[order-by-clause]
```

Die Variablen sind folgendermaßen definiert:

- *select-stmt1* und *select-stmt2* sind zulässige select-Anweisungen.
- *order-by-clause* ist eine order by-Klausel, die durch Zahlen und nicht durch den Namen auf eine Spalte verweist.

Hier sind einige Anforderungen und Dinge, die Sie beim intersect-Operator berücksichtigen müssen.:

- Die zwei select-Anweisungen dürfen keine order by-Klausel enthalten, Sie können jedoch das Ergebnis der gesamten intersect-Operation ordnen.
- Die Anzahl der von *select-stmt1* abgefragten Spalten muß gleich den von *select-stmt2* abgefragten Spalten sein.
- Die Datentypen der von *select-stmt1* abgefragten Spalten müssen gleich den Datentypen der von *select-stmt2* abgefragten Spalten sein.
- Die optionale *order-by-clause* unterscheidet sich von der üblichen order by-Klausel einer select-Anweisung, weil die zum Ordnen verwendeten Spalten über Nummern statt Namen angesprochen werden *müssen*. Der Grund dafür, daß die Spalten in der *order-by-clause* über Zahlen statt über Namen angesprochen werden müssen, ist, daß es in SQL nicht notwendig ist, daß die in *select-stmt1* abgefragten Spalten mit den in *select-stmt2* abgefragten Spalten identisch sind. Deshalb müssen Sie die Spalten, die zum Ordnen verwendet werden sollen, über ihre Position in der Auswahlliste angeben.

Betrachten Sie das folgende typische Szenario für die Verwendung des intersect-Operators. Als Dekan des Flugle-College haben Sie den Verdacht, daß einige der Lehrer am Flugle-College auch am Hoover-College unterrichten, Flugles Erzrivalen. Eine Liste der Lehrer am Hoover-College, dem Erzrivalen des Flugle-College, ist in Ihren Besitz gelangt. Sie wird in Listing 8.27 gezeigt. Die erste Abfrage liefert die Lehrer am Flugle-College. Die zweite Abfrage liefert die Lehrer am Hoover-College. Die dritte Abfrage verwendet den intersect-Operator, um die Lehrer zu finden, die beiden Tabellen gemeinsam sind.

**Listing 8.27: Inhalt der Instructor-Tabelle für das Hoover-College und das Flugle-College.**

```
SQL> select Last_Name, First_Name, MI, Position
2 from Instructor;

LAST_NAME FIRST_NAME M POSITION
```

```
-----  
HITCHCOCK BORIS PROFESSOR  
DANIELS LAURA ASSOCIATE PROFESSOR  
EDWARDS SAMANTHA PROFESSOR  
CHANG ROGER ASSISTANT PROFESSOR  
JASON JERROLD ASSOCIATE PROFESSOR  
TORRES PETER PROFESSOR  
ANGELO ARTHUR ASSOCIATE PROFESSOR  
RICHARDSON NANCY ASSISTANT PROFESSOR  
PARKER WILLIAM PROFESSOR  
CHERNOW BESS ASSOCIATE PROFESSOR  
CHU STEVEN PROFESSOR  
WEISS ROBERTA PROFESSOR  
RESTON RAPHAEL ASSOCIATE PROFESSOR  
BILLINGS BENJAMIN ASSISTANT PROFESSOR  
YOUNG ALLAN ASSOCIATE PROFESSOR  
NILAND MARTINA ASSOCIATE PROFESSOR  
BATES JOSEPH ASSISTANT PROFESSOR  
POULSON RANIER PROFESSOR
```

18 rows selected.

```
SQL> select Last_Name, First_Name, MI, Position  
2 from Hoover_Instructor;
```

```
LAST_NAME FIRST_NAME M POSITION  
-----
```

```
CHANG ROGER ASSISTANT PROFESSOR  
JASON JERROLD ASSOCIATE PROFESSOR  
TORRES PETER PROFESSOR  
CHERNOW BESS ASSOCIATE PROFESSOR  
CHU STEVEN PROFESSOR  
RESTON RAPHAEL ASSOCIATE PROFESSOR  
YOUNG ALLAN ASSOCIATE PROFESSOR  
NILAND MARTINA ASSOCIATE PROFESSOR  
BATES JOSEPH ASSISTANT PROFESSOR  
HUSTON MYRON T PROFESSOR  
RABINOWITZ KERMIT A LECTURER  
CHRISTIANSON PAUL V PROFESSOR
```

12 rows selected.

```
SQL> select Last_Name, First_Name, MI, Position  
2 from Instructor  
3 intersect  
4 select Last_Name, First_Name, MI, Position  
5 from Hoover_Instructor;
```

```
LAST_NAME FIRST_NAME M POSITION
-----
```

```
BATES JOSEPH ASSISTANT PROFESSOR
CHANG ROGER ASSISTANT PROFESSOR
CHERNOW BESS ASSOCIATE PROFESSOR
CHU STEVEN PROFESSOR
JASON JERROLD ASSOCIATE PROFESSOR
NILAND MARTINA ASSOCIATE PROFESSOR
RESTON RAPHAEL ASSOCIATE PROFESSOR
TORRES PETER PROFESSOR
YOUNG ALLAN ASSOCIATE PROFESSOR
```

9 rows selected.

Wenn Sie die Zeilen jeder Tabelle untersuchen, stellen Sie fest, daß die beiden Tabellen in der Tat neun Zeilen gemeinsam haben.

## Der union-Operator

Früher oder später werden Sie sich in einer Situation befinden, in der Sie die Zeilen ähnlicher Tabellen kombinieren müssen, um einen Bericht oder eine Tabelle für die Analyse zu erzeugen. Obwohl die Tabellen ähnliche Informationen repräsentieren, können sie sich beträchtlich unterscheiden. Um die Aufgabe auszuführen, sollten Sie den union-Operator verwenden.



Die Syntax für diese Operation ist ziemlich einfach:

```
select-stmt1
UNION
select-stmt2
[order-by-clause]
```

Die Variablen sind folgendermaßen definiert:

- *select-stmt1* und *select-stmt2* sind zulässige select-Anweisungen.
- *order-by-clause* ist eine optionale order by-Klausel, die durch Zahlen und nicht durch den Namen auf eine Spalte verweist.

Der union-Operator kombiniert die von der ersten select-Anweisung gelieferten Zeilen mit den von der zweiten select-Anweisung gelieferten Zeilen. Denken Sie an folgende Punkte, wenn Sie den union-Operator verwenden:

- Die zwei select-Anweisungen dürfen keine order by-Klausel enthalten, Sie können jedoch das Ergebnis der gesamten union-Operation ordnen.
- Die Anzahl der von *select-stmt1* abgefragten Spalten muß gleich der von *select-stmt2* abgefragten Spalten sein.
- Die Datentypen der von *select-stmt1* abgefragten Spalten müssen gleich den Datentypen der von *select-stmt2* abgefragten Spalten sein.
- Die optionale *order-by-clause* unterscheidet sich von der üblichen order by-Klausel einer select-Anweisung, weil die zum Ordnen verwendeten Spalten über Nummern statt Namen angesprochen werden müssen. Der Grund dafür, daß die Spalten in der *order-by-clause* über Zahlen statt über Namen angesprochen werden müssen, ist, daß es in SQL nicht notwendig ist, daß die in *select-stmt1* abgefragten Spalten mit den in *select-stmt2* abgefragten Spalten identisch sind.

Um die Verwendung von union zu illustrieren, stellen Sie sich vor, daß Sie die Aufgabe hätten, Informationen aus zwei seismologischen Labors zu vereinen. Die erste Tabelle stammt aus der Abteilung für Geophysik der Universität von Northern South Dakota; die andere Tabelle stammt aus einem privaten Forschungsinstitut, RIND. Listing 8.28 beschreibt die Struktur der beiden Tabellen.

***Listing 8.28: Beschreibung zweier Tabellen für seismische Ereignisse.***

SQL> desc UNSD\_Event

Name Null? Type

```
-----
EVENT_NO NUMBER
EPICENTER_LATITUDE NUMBER
EPICENTER_LONGITUDE NUMBER
EVENT_MAGNITUDE NUMBER
EVENT_TIME DATE
EVENT_WAVE CHAR(1)
INSTRUMENT_NAME VARCHAR2(30)
```

SQL> desc RIND\_Event

Name Null? Type

```
-----
LOCATION_LAT NUMBER
LOCATION_LON NUMBER
RICHTER_NUMBER NUMBER
DATE_TIME VARCHAR2(30)
WAVE_TYPE CHAR(1)
```

Halten Sie für einen Moment inne, um die Ähnlichkeiten und Unterschiede zwischen UNSD\_Event und RIND\_Event zu untersuchen. Zuerst einmal speichern beide Tabellen Informationen über seismische Ereignisse. Die Tabelle UNSD\_Event hat jedoch zwei zusätzliche Spalten: Event\_No und Instrument\_Name. Beide Tabellen speichern Breiten- und Längengrad des Epizentrums, die Stärke und den Wellentyp (P oder S). UNSD\_Event definiert Event\_Time jedoch als date, während RIND\_Event varchar2 verwendet, um Datum und Uhrzeit in Date\_Time zu speichern. Listing 8.29 verdeutlicht, was

geschieht, wenn Sie ein union ausführen, ohne Event\_Time in UNSD\_Event und Date\_Time und RIND\_Event in einen gemeinsamen Datentyp zu konvertieren.

***Listing 8.29: Anwendung des union-Operators auf zwei Tabellen mit unterschiedlichen Datentypen.***

```
SQL> select Epicenter_Latitude, Epicenter_Longitude, Event_Magnitude,
Event_Time, Event_Wave
2 from UNSD_Event
3 UNION
4 select Location_Lat, Location_Lon, Richter_Number, Date_Time,
Wave_Type
5 from RIND_Event;
select Epicenter_Latitude, Epicenter_Longitude, Event_Magnitude,
Event_Time, Event_Wave
*
```

ERROR at line 1:  
ORA-01790: expression must have same datatype as corresponding expression

Sie können erzwingen, daß beide Spalten denselben Datentyp haben, indem Sie Date\_Time in der Tabelle RIND\_Event mit der Funktion to\_date in einen Datumswert umwandeln, wie dies in Listing 8.30 gezeigt wird.

***Listing 8.30: Erzwingen desselben Datentyps für beide Tabellen mit Hilfe einer Funktion.***

```
SQL> select Epicenter_Latitude, Epicenter_Longitude, Event_Magnitude,
Event_Time, Event_Wave
2 from UNSD_Event
3 UNION
4 select Location_Lat, Location_Lon, Richter_Number,
TO_DATE(Date_Time, 'MM-DD-YY HH:MI:SS'), Wave_Type
5 from RIND_Event;
EPICENTER_LATITUDE EPICENTER_LONGITUDE EVENT_MAGNITUDE EVENT_TIM E
-----
12.83 189.85 5.8 25-APR-95 P
22.33 233.31 5.9 03-FEB-95 P
23.33 179.11 5.3 10-JAN-95 P
29.84 238.41 6.2 22-MAR-95 S
31.17 208.33 6.6 19-APR-95 S
31.84 241.21 6.1 12-MAR-95 S
37.81 211.84 6.4 11-JAN-95 S
7 rows selected.
```

Wenn Sie das Ergebnis nach Event\_Time ordnen wollen, können Sie die order by-Klausel verwenden. Anstatt jedoch über den Namen auf die Spalte zu verweisen, müssen Sie diese über die Reihenfolge in der Auswahlliste ansprechen. Listing 8.31 zeigt, wie die von der union abgefragten Zeilen nach

Event\_Time geordnet werden, dem vierten Eintrag in der Auswahlliste.

### **Listing 8.31: Sortieren der durch union gebildeten Zeilen.**

```
SQL> select Epicenter_Latitude, Epicenter_Longitude, Event_Magnitude,
Event_Time, Event_Wave
2 from UNSD_Event
3 UNION
4 select Location_Lat, Location_Lon, Richter_Number,
TO_DATE(Date_Time, 'MM-DD-YY HH:MI:SS'), Wave_Type
5 from RIND_Event
6 order by 4;
```

| EPICENTER_LATITUDE | EPICENTER_LONGITUDE | EVENT_MAGNITUDE | EVENT_TIM | E |
|--------------------|---------------------|-----------------|-----------|---|
| 23.33              | 179.11              | 5.3             | 10-JAN-95 | P |
| 37.81              | 211.84              | 6.4             | 11-JAN-95 | S |
| 22.33              | 233.31              | 5.9             | 03-FEB-95 | P |
| 31.84              | 241.21              | 6.1             | 12-MAR-95 | S |
| 29.84              | 238.41              | 6.2             | 22-MAR-95 | S |
| 31.17              | 208.33              | 6.6             | 19-APR-95 | S |
| 12.83              | 189.85              | 5.8             | 25-APR-95 | P |

7 rows selected.

## Der minus-Operator

Zusätzlich zu den Operatoren `intersect` und `union` bietet Oracle für den Vergleich einer Menge von Zeilen mit einer anderen Menge auch den `minus`-Operator.



Die Syntax für die Verwendung des `minus`-Operators ähnelt der Syntax des `union`-Operators:

```
select-stmt1
MINUS
select-stmt2
[order-by-clause]
```

Die Variablen sind folgendermaßen definiert:

- `select-stmt1` und `select-stmt2` sind zulässige `select`-Anweisungen.
- `order-by-clause` ist eine optionale `order by`-Klausel, die durch Zahlen und nicht durch den Namen auf eine Spalte verweist.



Die Anforderungen und Besonderheiten bei der Verwendung des minus-Operators sind im wesentlichen dieselben wie bei den Operatoren intersect und union. Um die Verwendung des minus-Operators zu verdeutlichen, betrachten Sie erneut die Tabellen Instructor und Hoover\_Instructor. Sie wollen feststellen, welche Lehrer am Flugle-College nicht auch am Hoover-College unterrichten. Listing 8.32 zeigt ein Beispiel dafür, wie dies mit dem minus-Operator erreicht werden kann. Die Abfrage liefert neun Zeilen, diese enthalten die Lehrer am Flugle-College, die nicht am Hoover-College lehren.

***Listing 8.32: Verwendung des minus-Operators.***

```
SQL> select Last_Name, First_Name, MI, Position
2 from Instructor
3 minus
4 select Last_Name, First_Name, MI, Position
5 from Hoover_Instructor;
```

```
LAST_NAME FIRST_NAME M POSITION
-----
```

```
ANGELO ARTHUR ASSOCIATE PROFESSOR
BILLINGS BENJAMIN ASSISTANT PROFESSOR
DANIELS LAURA ASSOCIATE PROFESSOR
EDWARDS SAMANTHA PROFESSOR
HITCHCOCK BORIS PROFESSOR
PARKER WILLIAM PROFESSOR
POULSON RANIER PROFESSOR
RICHARDSON NANCY ASSISTANT PROFESSOR
WEISS ROBERTA PROFESSOR
```

```
9 rows selected.
```

Die Abfrage in Listing 8.32 sagt jedoch nichts über die Lehrer am Hoover-College aus, die nicht am Flugle-College unterrichten. Um diese Information zu sehen, müssen die zwei select-Anweisungen vertauscht werden, wie es in Listing 8.33 gezeigt wird.

***Listing 8.33: Vertauschen der über den minus-Operator miteinander verknüpften select-Anweisungen.***

```
SQL> select Last_Name, First_Name, MI, Position
2 from Hoover_Instructor
3 minus
4 select Last_Name, First_Name, MI, Position
5 from Instructor;
```

```
LAST_NAME FIRST_NAME M POSITION
-----
```

```
CHRISTIANSON PAUL V PROFESSOR
```

HUSTON MYRON T PROFESSOR  
RABINOWITZ KERMIT A LECTURER

Dieses Beispiel zeigt, daß der minus-Operator nicht alle Unterschiede zwischen den beiden Tabellen zeigt - er gibt nur die Zeilen der ersten Tabelle zurück, die in der zweiten Tabelle nicht gefunden werden. Sie können in den vom minus-Operator verwendeten select-Anweisungen auch eine where-Klausel angeben.

## Erstellen einer Tabelle durch Auswahl aus einer anderen Tabelle

Wenn Sie eine neue Datenbankanwendung entwickeln, gehen Sie normalerweise durch eine Phase des Experimentierens mit verschiedenen Tabellendesigns. Egal, ob Sie Fremddaten oder künstliche Daten verwenden, es ist immer schön, wenn man einige einfache Methoden zum Experimentieren hat. Eine DDL-Anweisung, die Sie verwenden können, ist create table - mit einer Unterabfrage.



Die Syntax ist

```
CREATE TABLE table-name
[(column-definition1, ... column-definitionN)]
AS
select-statement
```

Die Variablen sind folgendermaßen definiert:

- *table-name* ist der Name der neuen Oracle-Tabelle.
- *column-definition1* bis *column-definitionN* sind optionale Spaltendefinitionen, die verwendet werden, um verschiedene Namen von Spalten anzugeben, die mit den von der Unterabfrage zurückgegebenen Werten assoziiert werden.
- *select-statement* ist eine zulässige select-Anweisung, die beim Erzeugen der neuen Tabelle verwendet wird.

Als Beispiel enthält Listing 8.34 eine Anweisung, die eine neue Tabelle erzeugt, deren Zeilen eine Teilmenge einer existierenden Tabelle sind.

**Listing 8.34: Erzeugung einer auf einer Unterabfrage basierenden Tabelle.**

```
SQL> create table Anthro_Instructors
2 as
3 select * from Instructor
4 where
```

```
5 Department_ID = 'ANTHRO';
```

Table created.

```
SQL> select Department_ID, Last_Name, First_Name
2 from Anthro_Instructors;
```

```
DEPARTMENT_ID LAST_NAME FIRST_NAME
```

```
-----
ANTHRO HITCHCOCK BORIS
ANTHRO DANIELS LAURA
ANTHRO POULSON RANIER
```

Beachten Sie, daß Oracle Ihnen nicht mitteilt, wie viele Zeilen in die neue Tabelle eingefügt wurden. In der Tat können Sie eine neue leere Tabelle erzeugen, indem Sie eine unmögliche Bedingung angeben, wie es in Listing 8.35 gezeigt wird.

***Listing 8.35: Erzeugung einer leeren Tabelle.***

```
SQL> create table Empty_Instructor
2 as
3 select * from Instructor
4 where
5 1=2;
```

Table created.

```
SQL> select count(*) from Empty_Instructor;
```

```
COUNT(*)
-----
0
```



Oracle weist die create table-Anweisung zurück, wenn die Unterabfrage auf eine Spalte vom Typ long oder long raw verweist. Wenn Sie Zeilen aus einer long-Spalte von einer Tabelle in eine andere kopieren müssen, verwenden Sie den Befehl copy von SQL\*Plus. Er unterstützt das Kopieren von Spaltenwerten vom Typ long.



*Wenn sie mit einer Unterabfrage verwendet wird, kopiert die create table-Anweisung nur eine Sorte von Integritätsregeln für Tabellen oder Spalten, die Integritätsregel not null. Wenn Sie wollen, daß eine Tabelle alle existierenden Integritätsregeln aufweist, müssen Sie diese mit einer alter table-Anweisung angeben.*

---

## Datensichten verwenden

Eine *Datensicht* ist eine Abfrage einer oder mehrerer Tabellen, die einen weiteren Weg zur Darstellung von Informationen bietet. Eine Datensicht enthält oder speichert nicht wirklich eigene Daten; in der Tat können Sie sich eine Datensicht als virtuelle Tabelle vorstellen. Der einzige Speicherplatz, den eine Datensicht wirklich benötigt, ist der für die select-Anweisung, durch die sie definiert ist.

---



*Eine Datensicht ist eine gespeicherte Abfrage, die auf einer Abfrage einer oder mehrerer Tabellen basiert.*

---

Sie können eine Datensicht für die folgenden Aufgaben einsetzen:

- Genauere Kontrolle über die Sicherheit
- Komplexität vor einem Entwickler oder Benutzer verstecken
- Spalten umbenennen

## Die Syntax einer Datensicht

Die Syntax für das Erzeugen einer Datensicht ist wirklich ziemlich einfach.



Um eine Datensicht zu definieren, verwenden Sie diese Syntax:

```
CREATE VIEW view-name
```

```
(column1, ... ,columnN)
```

```
AS
```

```
select-statement
```

Die Variablen sind folgendermaßen definiert:

- *view-name* ist der Name der Datensicht (er unterliegt denselben Vorschriften wie andere Namen für Objekte in Oracle).
- *column1* bis *columnN* sind die Spaltennamen der Datensicht, die den in *select-statement* referenzierten Spalten entsprechen.
- *select-statement* ist eine in Oracle zulässige select-Anweisung.

Die folgende einfache Datensicht basiert auf einer einzelnen Tabelle. Nehmen Sie an, daß mehrere Verwalter am Flugle-College um eine sehr einfache Methode gebeten haben, um die aktuellen Kosten für jeden Kurs abzufragen. Die aktuellen Kosten für den Kurs sind die Anzahl der Einheiten multipliziert mit \$250 pro Einheit zuzüglich eventueller zusätzlicher Gebühren. Obwohl Sie den Verwaltern gezeigt haben, wie man eine einfache Abfrage erstellt, wollen diese, daß Sie den Prozeß verbessern.

Listing 8.36 zeigt die Datensicht, die Sie zu diesem Zweck entwerfen. Hier ist eine Beschreibung jeder Zeile der SQL-Anweisung, aus der die Datensicht besteht. Die erste Zeile gibt den Namen der Datensicht an: *Course\_Cost*. Die zweite Zeile listet die fünf Spalten auf, aus denen sich die Datensicht zusammensetzt: *Course\_ID*, *Department\_ID*, *Title*, *Description* und *Cost*. Die dritte Zeile gibt an, daß die Datensicht aus einer Abfrage erstellt werden soll. Die vierte Zeile gibt die ersten vier Zeilen an, die von der Datensicht geliefert werden sollen. *Course\_ID*, *Department\_ID*, *Title* und *Description*. Die fünfte Zeile gibt an, daß die fünfte Spalte der Datensicht - *Cost* - berechnet wird, indem die Einheiten mit \$250 pro Einheit multipliziert werden und die zusätzlichen Gebühren addiert werden.

***Listing 8.36: Erzeugung einer einfachen Datensicht.***

```
SQL> create view Course_Cost
2 (Course_ID, Department_ID, Title, Description, Cost)
3 as
4 select Course_ID, Department_ID, Title, Description,
5 Units*250 + Additional_Fees
6 from Course;
```

View created.

```
SQL> select Department_ID, Course_ID, Title, Cost
2 from Course_Cost
3 order by Department_ID, Course_ID;
```

```
DEPARTMENT COURS TITLE COST
```

```
-----
ANTHRO 101 INTRO TO ANTHROPOLOGY 750
```

```

ANTHRO 174 PHYS ANTHRO FIELD TECHNIQUES 805
ANTHRO 189 SEMINAR ON NACIMERA 507.5
BIO 101 INTRO TO BIOLOGY 805
BIO 177 INVERTEBRATE ANATOMY 815
BIO 178 MAMMALIAN ANATOMY 820
ECON 101 INTRO TO ECONOMICS 775
ECON 189 MONETARY POLICY 1500
ECON 199 WORKSHOP ON MARX 750
ENG 101 INTRO TO STRUCTURES 825
ENG 102 INTRO TO CIRCUIT THEORY 795
ENG 103 INTRO TO DYNAMICS 785
ENG 199 SEMINAR ON CHAOS 545
ENGL 101 INTRO TO ENGLISH LIT 750
ENGL 189 EVOLUTIONARY GRAMMAR 500
ENGL 192 MODERN ENGLISH LIT 750
ENGL 193 SEMINAR ON THEME ANALYSIS 500
HIST 115 EARLY AMERICAN HISTORY 750
HIST 184 MODERN EUROPEAN HISTORY 750
HIST 199 WORKSHOP ON JEFFERSON 750
MATH 101 GENERAL CALCULUS 750
MATH 189 NUMBER THEORY 750
MATH 50 PRE-CALCULUS 750
MATH 51 ADVANCED ARITHMETIC 760
PHILO 101 INTRO TO PHILOSOPHY 750
PHILO 174 ANCIENT GREEK PHILOSOPHERS 750
PHILO 198 MODERN PHILOSOPHY 750
PSYCH 101 INTRO TO PSYCHOLOGY 775
PSYCH 181 WORKSHOP ON NEUROSES 575
PSYCH 183 WORKSHOP ON NORMALITY 545
PSYCH 185 ABNORMAL PSYCHOLOGY 770
PSYCH 501 PSYCH IN FILM 825

```

32 rows selected.

## Den Zugriff auf Daten mit einer Datensicht einschränken

Oracle bietet mehrere Mechanismen für die Einschränkung des Zugriffs auf Daten: Datensichten, Datenbanktrigger sowie Zugriffsberechtigungen für Tabellen und Spalten. Bestimmte Privilegien für diese Objekte können an Datenbankrollen und einzelnen Benutzer vergeben werden. Um festzulegen, welche Methode am passendsten ist, müssen Sie die Sicherheitsanforderungen Ihrer Anwendung feststellen.

Wenn Sie eine Datensicht verwenden, um den Zugriff auf Daten einzuschränken, stehen Ihnen mehrere Methoden zur Verfügung. Jede Methode bietet einen anderen Grad der Kontrolle über den Zugriff auf

Daten, von grob bis sehr fein. Eine grobe Kontrolle ermöglicht es Ihnen, eine Datensicht zu definieren, die eine Teilmenge ihrer Basistabelle ist. Zum Beispiel enthält die Tabelle Student Adressinformationen, auf die nicht alle Benutzer zugreifen sollen, die Zugriff auf die Tabelle haben. Listing 8.37 zeigt, wie Sie eine Datensicht der Tabelle Student erzeugen, die diese persönlichen Informationen nicht enthält. Ein sehr feiner Grad der Kontrolle kann mit Datenbanktriggern erreicht werden; ein Beispiel wird an Tag 12, »Developer/2000: Einführung in Oracle Forms«, präsentiert.

***Listing 8.37: Erzeugung einer auf einer Teilmenge von Tabellenspalten basierenden Datensicht.***

```
SQL> create view Student_No_Personal
2 as
3 select Student_ID, Last_Name, First_Name, MI, Year, Email
4 from Student;
```

View created.

```
SQL> describe Student_No_Personal
Name Null? Type
```

```
-----
STUDENT_ID NOT NULL VARCHAR2(20)
LAST_NAME NOT NULL VARCHAR2(25)
FIRST_NAME NOT NULL VARCHAR2(25)
MI VARCHAR2(1)
YEAR VARCHAR2(25)
EMAIL VARCHAR2(100)
```

Wie Sie in Listing 8.37 sehen, gibt der describe-Befehl von SQL\*Plus nicht an, ob die Objektbeschreibung zu einer Tabelle oder einer Datensicht gehört. Auch wenn der Benutzer weiß, daß die Adresse eines Studenten in der Tabelle Student gespeichert wird, kann die Datensicht die Spalte nicht liefern, da diese nicht in der Definition der Datensicht enthalten ist.

## Die Komplexität mit einer Datensicht verbergen

Während der Anwendungsentwicklung haben Sie es oft mit Entwicklern oder Benutzern zu tun, die verschiedene organisatorische Perspektiven und einen unterschiedlichen Grad an technischer Erfahrung besitzen. Darum sollten Sie die Mechanismen verwenden, die Oracle zur Anpassung der Umgebung an Entwickler und Benutzer bietet.

Ich schließe Entwickler in diese Kategorie ein, weil ihre Kenntnisse von SQL im allgemeinen und Oracle im besonderen variieren. Beispielsweise verlangen viele Formulare und Berichte das Verknüpfen mehrerer Tabellen. Das Verwenden von Datensichten vereinfacht diesen Prozeß, weil Sie als Architekt der Anwendung die Datensichten festlegen können, mit denen sich die Entwickler befassen müssen.

Datensichten sind auch eine exzellente Möglichkeit, eine Datenbankumgebung für Endanwender anzupassen. Dies gilt besonders für große Organisationen, in denen für verschiedene Aufgaben auf dieselben Informationen zugegriffen wird. Typischerweise hat jede Gruppe einen eigenen Namen für dieselben Informationen. Aufgrund der weiten Verbreitung der Werkzeuge von Drittanbietern wie Business Objects, IQ Objects und Oracle Discoverer/2000, sollte der Name einer Spalte exakt beschreiben, welche Information diese enthält. So bekommt der Endanwender eine Hilfestellung bei der Frage, welche Spalten er abfragen soll. Indem Sie eine Datensicht erzeugen, können Sie die Spaltennamen anpassen, die eine Gruppe von Benutzern zu sehen bekommt.

Eine Datensicht kann die Komplexität einer Verknüpfung mehrerer Tabellen verbergen. Durch Definieren einer Datensicht werden die Benutzer davon befreit, die Eigenarten der select-Anweisung lernen zu müssen. Wenn Sie beispielsweise eine zusammenfassende Datensicht der Klassenpläne am Flugle-College zusammenstellen wollen, können Sie Informationen aus den Tabellen Class, Course und Instructor verknüpfen. Listing 8.38 enthält die create view-Anweisung, die für eine Datensicht auf der Basis des Joins von drei Tabellen erstellt wird.

***Listing 8.38: Eine auf einem Join mit mehreren Tabellen basierende Datensicht.***

```
SQL> create view Class_Summary as
2 select Class_ID, Class_Building, Class_Room,
3 CL.Department_ID, CL.Course_ID, CO.Title,
4 Last_Name, First_Name
5 from Class CL, Course CO, Instructor I
6 where
7 CL.Department_ID = CO.Department_ID and
8 CL.Course_ID = CO.Course_ID and
9 CL.Instructor_ID = I.Instructor_ID;
```

View created.

Durch Verwenden der Datensicht Class\_Summary kann der Benutzer die gewünschten Spalten einfach auswählen, ohne irgendwelche Verknüpfungsbedingungen anzugeben. Listing 8.39 zeigt, wie der Benutzer die Klassen auswählen kann, deren Title die Phrase intro enthält.

***Listing 8.39: Auswahl aus einer Datensicht.***

```
SQL> select Class_ID, Class_Building, Class_Room,
2 Department_ID, Course_ID, Title, Last_Name
3 from Class_Summary
4 where
5 Title like '%INTRO%';
```

```
CLASS_ CLASS_BUILDING CLAS DEPART COURS TITLE LAST_NAME
-----
109100 FLUGLE HALL 180 BIO 101 INTRO TO BIOLOGY WEISS
108300 FLUGLE HALL 150 ENGL 101 INTRO TO ENGLISH LIT CHERNOW
```



## Zusammenfassung

Diese Lektion hat sich auf die folgenden Merkmale von SQL konzentriert:

- Oracle bietet Gruppenfunktionen wie max und min, die einen einzelnen Wert für eine Menge von Zeilen zurückgeben.
- Die select-Anweisung hat zwei optionale Klauseln - group by und having - die Zeilen zurückgeben, welche nach einem gemeinsamen Wert für die angegebenen Spalten gruppiert sind.
- Hierarchische Daten können durch die connect by-Klausel organisiert werden. Sie müssen mit der start with-Klausel eine Startbedingung angeben.
- Eine Join-Operation ist eine select-Anweisung, deren from-Klausel eine oder mehrere Tabellen und eine oder mehrere Verknüpfungsbedingungen enthält, die typischerweise Fremd- und Primärschlüssel vergleichen.
- Der intersect-Operator gibt eine Menge von gemeinsamen Zeilen zurück, die in zwei Mengen von Zeilen gefunden wurden.
- Der union-Operator gibt alle eindeutigen Zeilen aus zwei Mengen von Zeilen zurück.
- Der minus-Operator gibt Zeilen zurück, die in einer Menge, aber nicht in der anderen gefunden wurden.

## Wie geht es weiter?

An Tag 9, »Programmierung einer Oracle-Datenbank mit PL/SQL«, lernen Sie etwas über die grundlegenden Elemente von PL/SQL, wie Anweisungen zur Steuerung des Programmablaufs, Variablendeklarationen sowie Prozedur- und Funktionsdeklarationen.

## Fragen und Antworten

Frage:

Können zwei Tabellen A und B durch Spalten verknüpft werden, wenn der Datentyp der einen Spalte in Tabelle A nicht der gleiche ist wie der Datentyp der anderen Spalte in Tabelle B?

Antwort:

*Ja, das ist möglich. Sie können beispielsweise eine Umwandlungsfunktion wie **to\_char**, **to\_date** oder **to\_number** verwenden, um die eine Spalte so umzuwandeln, daß der von der Funktion zurückgegebene Datentyp dem Datentyp der anderen Spalte entspricht.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Erstellen Sie eine SQL-Anweisung, die den Nachnamen eines Lehrers liefert, der einen Kurs mit zusätzlichen Gebühren über DM 50,- unterrichtet.
2. Erstellen Sie eine SQL-Anweisung, die eine Liste von Städten, in denen Studenten wohnen, sowie die Zahl der Studenten in jeder Stadt liefert.
3. Erstellen Sie eine Datensicht, die jede Klasse auflistet - ihre Class\_ID, Department\_ID und Kurs\_ID -, die am Montag zusammentrifft.

## Übungen

Die Anzahl der Lehrer ist 18, die Anzahl der angebotenen Klassen ist 10. Trotzdem ist die Zahl der eindeutigen Instructor\_IDs in der Tabelle Class gleich 9. Geben Sie unter Verwendung dieser Tabelle und von SQL eine vollständige Erklärung.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

# 9

# Programmierung einer Oracle-Datenbank mit PL/SQL

[PL/SQL ist eine blockstrukturierte Sprache](#)

[Variablendeklaration mit PL/SQL](#)

[Einige bekannte Kontrollstrukturen](#)

[SQL-Anweisungen in einem PL/SQL-Programm verwenden](#)

[PL/SQL-Unterprogramme](#)

[Verwenden von Unterblöcken](#)

[Deklarieren einer Prozedur](#)

[Deklarieren einer Funktion](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

## Workshop

---

Bis jetzt habe ich Ihnen SQL als eine Sprache ohne prozedurale Fähigkeiten präsentiert. Oracle bietet jedoch durch die Sprache PL/SQL Erweiterungen zu SQL. PL/SQL ist die Basis für die folgenden logischen Elemente von Anwendungen:

- **SQL\*Plus-Skript:** Ein SQL\*Plus-Skript kann PL/SQL-Unterprogrammblöcke enthalten.
- **Gespeicherte Prozeduren oder Funktionen:** Eine gespeicherte Prozedur oder Funktion ist ein PL/SQL-Unterprogramm, das durch eine Client-Anwendung, einen Datenbanktrigger oder einen Anwendungstrigger eines Oracle-Werkzeugs aufgerufen werden kann.
- **Datenbanktrigger:** Ein Datenbanktrigger ist ein PL/SQL-Unterprogramm, das eine Aktion aufgrund der Ausführung einer DML-Anweisung - wie insert, update oder delete - an einer Datenbanktabelle durchführt.
- **Package:** Eine Menge von PL/SQL-Prozeduren, -Funktionen, -Cursors und anderen PL/SQL-Variablen, die in einem Paket gebündelt werden.
- **Anwendungstrigger:** Entwicklungswerkzeuge für Oracle-Anwendungen wie Oracle Forms und Oracle Reports sind mit einer PL/SQL-Engine ausgestattet, so daß Entwickler mit PL/SQL Anwendungstrigger konstruieren können.

Der Zweck der heutigen Lektion ist es, die fundamentalen Elemente von PL/SQL einzuführen. Sie sollten mit der Syntax und dem Gebrauch von PL/SQL vertraut sein, bevor Sie am Tag 10, »Programmentwicklung mit PL/SQL«, oder am Tag 11, »Weitere Programmieretechniken für PL/SQL«, versuchen, gespeicherte Prozeduren zu entwerfen. Diese Lektion behandelt Merkmale, die in PL/SQL bis Version 2.3 enthalten sind.

---



*Verwenden Sie nicht SQL\*Plus, um die einzelnen Zeilen eines PL/SQL-Skripts einzugeben. Wenn Ihnen ein Tippfehler unterläuft, gibt Ihnen SQL\*Plus keine Rückmeldung, bis es das / liest, das den PL/SQL-Block abschließt - SQL\*Plus übergibt den gesamten Block erst an die PL/SQL-Engine, wenn der Block vollständig ist. Der PL/SQL-Block ist Teil des Oracle-RDBMS. Verwenden Sie statt dessen einen Texteditor wie Notepad, um Ihre PL/SQL-Skripte zu entwickeln. Sie können das Skript direkt in SQL\*Plus einfügen oder es mit den Befehlen start oder @ aufrufen. Am Tag 16 werden Sie etwas über Procedure Builder lernen, der SQL\*Plus bei der Entwicklung von Programmen weit überlegen ist. Es gibt Produkte von Drittherstellern mit ähnlichen Fähigkeiten; Sie können z.B. den SQL Station Coder von Platinum testen ([www.platinum.com](http://www.platinum.com)) oder das auf dem WEB\*PL-Browser basierende Entwicklungswerkzeug von DBCorp ([www.dbcorp.ab.ca](http://www.dbcorp.ab.ca)).*

---

## PL/SQL ist eine blockstrukturierte Sprache

PL/SQL ist eine blockstrukturierte Sprache mit einer ähnlichen Syntax wie die Programmiersprache C. Zusätzlich zu eingebetteten SQL-Anweisungen bietet PL/SQL Standardkonstrukte für das Programmieren wie Deklarationen von Prozeduren und Funktionen, Kontrollanweisungen wie if-then-else und loop und **deklarierte Variablen**. Ein PL/SQL-Programm besteht aus **Prozeduren, Funktionen oder anonymen Blöcken**. Ein *anonymer Block* ist ein unbenannter PL/SQL-Block, der keine Argumente besitzt und keine Werte zurückgibt. Anonyme Blöcke kommen häufig in Skripten vor, die in einer SQL\*Plus-Sitzung ausgeführt werden.

Abbildung 19.1 zeigt die oberste Ebene der Struktur eines SQL-Blocks. Dieser enthält einen optionalen Abschnitt mit Deklarationen, einen ausführbaren Abschnitt und einen optionalen Abschnitt für die Behandlung von Exceptions und Fehlern von PL/SQL und SQL

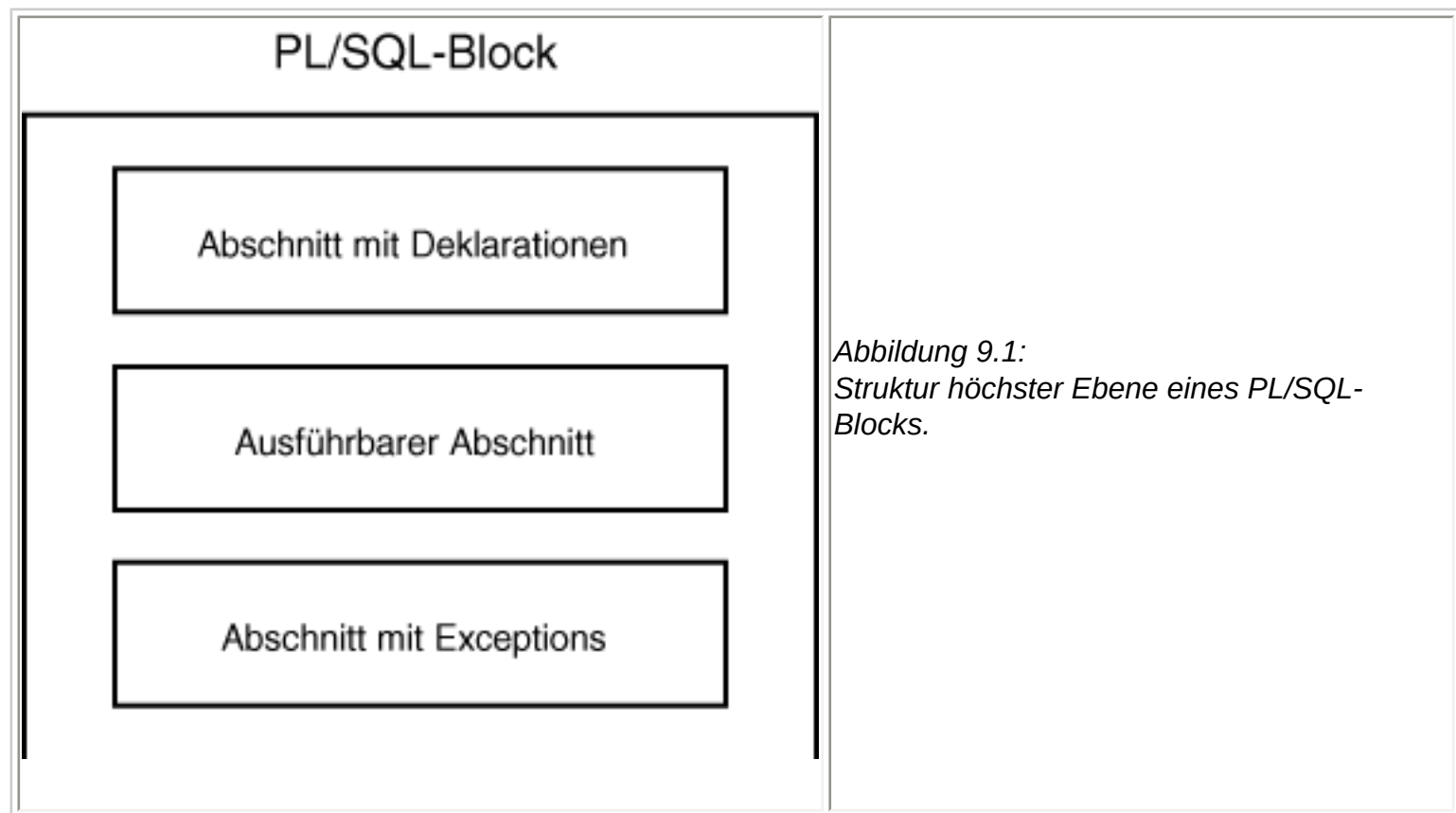


Abbildung 9.1:  
Struktur höchster Ebene eines PL/SQL-Blocks.

Lassen Sie uns einen einfachen anonymen SQL-Block betrachten, der einige Testdaten erzeugt. Listing 9.1 enthält einen anonymen Block, der durch ein SQL\*Plus-Skript ausgeführt wird und 100 Zeilen in TEST\_TABLE einfügt.

**Listing 9.1: Ausführung eines anonymen PL/SQL-Blocks in SQL\*Plus.**

```
SQL> @c:\tyo\day10_1
SQL> drop table test_table;
```

```

Table dropped.
SQL>
SQL> create table test_table (
2 record_number int,
3 current_date date);
Table created.
SQL>
SQL> DECLARE
2
2 max_records CONSTANT int := 100;
3 i int := 1;
4
4 BEGIN
5
5 FOR i IN 1..max_records LOOP
6
6 INSERT INTO test_table
7 (record_number, current_date)
8 VALUES
9 (i, SYSDATE);
10
10 END LOOP;
11
11 COMMIT;
12 END;
/
PL/SQL procedure successfully completed.

```



Werfen Sie einen Blick auf einige Elemente dieses PL/SQL-Skripts. Dieses Skript ist ein *anonymer PL/SQL-Block*, da es keinen Namen hat - es wird nicht als Prozedur, Funktion oder Paket deklariert. Alle Zeilen dieses Skripts sind in einem einzigen SQL\*Plus-Skript enthalten. Die ersten zwei Fenster löschen TEST\_TABLE und erzeugen sie dann neu. Der eigentliche PL/SQL-Block beginnt mit dem Wort *declare*. Der Deklarationsabschnitt enthält eine Konstante *max\_records* (in Zeile 2) und eine Variable *i* (in Zeile 3), diese dient als Zähler. Der Anfang des ausführbaren Teils des Blocks wird durch *begin* gekennzeichnet (in Zeile 4). Der Block enthält eine einzelne *for loop* (in Zeile 5), die eine Zeile in TEST\_TABLE einfügt, solange *i* kleiner oder gleich *max\_records* ist. Wenn die *for loop* beendet ist, wird die Transaktion festgeschrieben (in Zeile 11). Die letzte Zeile des Skripts ist ein */*, dies veranlaßt SQL\*Plus dazu, den PL/SQL-Block an die PL/SQL-Engine zu schicken. Wenn kein PL/SQL-Übersetzungsfehler auftritt, ist die einzige Rückmeldung von SQL\*Plus eine Meldung: *PL/SQL procedure successfully completed*. In der nächsten Lektion werden Sie sehen, wie man in PL/SQL Diagnosemeldungen erzeugt, so daß man die Fortschritte bei der Ausführung eines PL/SQL-Unterprogramms beobachten kann.

## Der Deklarationsteil

Der Deklarationsteil eines PL/SQL-Blocks ist optional. Sie müssen jedoch alle Variablen und Konstanten deklarieren, die in den PL/SQL-Anweisungen verwendet werden. Um einen Deklarationsteil in einen PL/SQL-Block aufzunehmen, beginnen Sie den PL/SQL-Block mit dem Wort `declare`. Jede Deklaration einer Variablen oder Konstanten besteht aus ihrem Namen, dem Datentyp und einem optionalen Standardwert. Wie bei allen PL/SQL-Anweisungen wird die Deklaration jeder Variablen und Konstanten mit einem Semikolon abgeschlossen. Listing 9.2 enthält einige Beispiele für Deklarationen von Variablen und Konstanten.

### *Listing 9.2: Beispiele deklarierter Variablen und Konstanten.*

```
Fax_Number VARCHAR2(10);  
Current_Used_Value NUMBER(6,2) := 100.00;  
Max_Current_Used_Value REAL := 9999.99;  
State VARCHAR2(2) := 'CA';
```

## Der Ausführungsteil

Der Ausführungsteil eines PL/SQL-Blocks folgt auf das Schlüsselwort `begin`. Jede PL/SQL-Anweisung wird durch ein Semikolon beendet. Diese Anweisungen können wie folgt aufgeteilt werden:

- Zuweisungsanweisungen
- Anweisungen, die den Programmablauf steuern
- SQL-Anweisungen
- Cursoranweisungen

## Der Exception-Teil

Eine Exception ist eine Fehlerbedingung, die während der Ausführung eines PL/SQL-Programms auftritt. Eine Exception kann vordefiniert sein - wenn beispielsweise eine `insert`-Anweisung versucht, eine doppelte Zeile in eine Tabelle einzufügen, wird eine `dup_val_on_index`-Exception ausgelöst. Sie können auch Ihre eigenen, für die Anwendung spezifischen Exceptions definieren. Der Exception-Teil definiert die Exception-Handler, die sowohl für vordefinierte, als auch für benutzerdefinierte Exceptions aufgerufen werden. Jeder Exception-Handler besteht aus einer oder mehreren PL/SQL-Anweisungen.

## Variablendeklaration mit PL/SQL

Eine weitere Fähigkeit von PL/SQL - jedoch nicht von SQL - sind zusätzliche Datentypen. Zusätzlich zu den normalen Datentypen von Oracle-SQL können Sie mit PL/SQL Variablen der folgenden Datentypen deklarieren:

|                |                                                                                                                                                                                                                                                                |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| boolean        | Eine boolesche Variable, die folgende Werte aufweisen kann: true, false oder null.                                                                                                                                                                             |
| binary_integer | Dieser Typ wird für vorzeichenbehaftete ganze Zahlen im Bereich von -2147483647 bis 2147483647 verwendet.                                                                                                                                                      |
| natural        | Eine Untermenge von binary_integer. <b>Dieser Datentyp umfaßt die ganzen Zahlen von 0 bis 2147483647.</b>                                                                                                                                                      |
| positive       | Eine weitere Untermenge von binary_integer, dieser Datentyp umfaßt die ganzen Zahlen von 1 bis 2147483647.                                                                                                                                                     |
| %type          | Diese Bezeichnung ermöglicht es, den Datentyp einer Variablen als äquivalent zum Datentyp der angegebenen Spalte zu deklarieren. Das Ergebnis ist PL/SQL-Code, der sich leichter warten läßt.                                                                  |
| %rowtype       | Mit diesem Datentyp können Sie einen zusammengesetzten Datentyp deklarieren, der äquivalent zu einer Zeile der angegebenen Tabelle ist. Die zusammengesetzte Variable wird aus den Spaltennamen und den Datentypen der referenzierten Tabelle zusammengesetzt. |

Zusätzlich bietet PL/SQL zwei zusammengesetzte Datentypen: table und record. Sie werden an Tag 10 mehr darüber lernen.

## Deklaration einer Variablen mit Hilfe von %type



Die Syntax für die Deklaration einer Variablen mit %type ist



*variable-name table-name.column-name*%TYPE

Die Variablen sind folgendermaßen definiert:

- *variable-name* ist die Variable, die Sie deklarieren.
- *table-name* ist eine Tabelle, die eine Spalte enthält, deren Typ die Variable annehmen soll.
- *column-name* ist eine Spalte, die in *table-name* definiert ist.

Eine Variable, die den Namen eines Servicetechnikers speichert, deklarieren Sie beispielsweise auf folgende Weise:

```
Tech_Name Depot_Estimate.Technician%TYPE;
```

Der Vorteil der Verwendung von %type in einer Variablendeklaration ist, daß der PL/SQL-Code von der Definition der Spalte Technician in der Tabelle Depot\_Estimate abhängt.

## Deklaration einer Variablen mit Hilfe von %rowtype



Die Syntax für die Deklaration einer Variablen mit %rowtype ist

*variable-name table-name*%ROWTYPE

Die Variablen sind folgendermaßen deklariert:

- *variable-name* ist die zusammengesetzte Variable, die Sie deklarieren.
- *table-name* ist die Tabelle, deren Struktur die zusammengesetzte Variable annehmen wird.

Eine Variable, die eine Zeile aus der Tabelle Depot\_Estimate speichert, wird beispielsweise folgendermaßen deklariert:

```
Depot_Est_Row Depot_Estimate%ROWTYPE;
```

Auf ein Element von Depot\_Est\_Row kann folgendermaßen zugegriffen werden:

```
Depot_Est_Row.Technician := 'RICHARDSON';
```

## Einige bekannte Kontrollstrukturen

Verschiedene PL/SQL-Kontrollanweisungen steuern den Programmablauf in einem PL/SQL-Unterprogramm.

- `if-then-elsif`
- `loop`
- `exit`
- `while loop`
- `for loop`
- `goto`
- `null`

Bevor Sie gespeicherte Prozeduren oder Trigger konstruieren, sollten Sie mit den Grundlagen der Programmierung in PL/SQL vertraut sein. Der nächste Teil dieser Lektion zeigt den Gebrauch dieser Anweisungen im Detail.

## Die if-Anweisung

Die Syntax der if-then-elsif-Anweisung von PL/SQL unterscheidet sich etwas von der vergleichbaren Anweisung in der Programmiersprache C.



Die PL/SQL-Syntax ist

```
IF condition THEN
statement; ... statement;
[ELSIF condition THEN
statement; ... statement;]
...
[ELSIF condition THEN
statement; ... statement;]
[ELSE condition THEN
statement; ... statement;]
END IF;
```

Die Variablen sind folgendermaßen definiert:

- *condition* ist eine gültige PL/SQL-Bedingung.
- *statement* ist eine gültige PL/SQL-Anweisung.

Was die if-then-elsif-Anweisung angeht, beachten Sie bitte folgende Tatsachen:

- Die elsif- und else-Klauseln sind optional.
- Eine if-Anweisung kann mehrere elsif-Klauseln, aber nur eine else-Klausel haben.
- Beachten Sie die Schreibweise: elsif, nicht elseif.

Listing 9.3 zeigt ein einfaches Beispiel für die if-then-elsif-Anweisung.

**Listing 9.3: Verwendung der if-then-elsif-Anweisung.**

```
if MOD(i,5) = 0 then
rec_number := 5;
elsif MOD(i,7) = 0 then
rec_number := 7;
else
rec_number := i;
end if;
```

## Die einfache loop-Anweisung



Der einfachste Schleifentyp ist die loop-Anweisung ohne zusätzliche Kennzeichner.

```
LOOP
statement; ... statement;
END LOOP
```

Die Variable ist folgendermaßen definiert:

- *statement* ist eine gültige PL/SQL-Anweisung, die auch eine weitere Schleife enthalten kann.

Diese Schleife ist offensichtlich eine Endlosschleife. Um diese Schleife zu verlassen, wenn eine bestimmte Bedingung erfüllt ist, verwenden Sie die exit-Anweisung.

## Die exit-Anweisung

Die exit-Anweisung hat zwei Formen:

- exit ohne andere Klauseln - ein unbedingtes Exit
- exit [*label\_name*] when *condition*

Die erste Form von `exit` sorgt dafür, daß der Programmablauf die Schleife verläßt, in der die `exit`-Anweisung eingeschlossen ist.

Die zweite Form von `exit` sorgt dafür, daß der Programmablauf die einschließende Schleife verläßt, wenn die angegebene Bedingung erfüllt ist, wie in Listing 9.4 gezeigt wird.

**Listing 9.4: Verwendung der `exit`-Anweisung.**

```
SQL> declare
2
2 i positive := 1;
3 max_loops constant positive := 100;
4
4 begin
5
5 loop
6
6 i := i + 1;
7 exit when i > max_loops;
8
8 end loop;
9
9 end;
10 /
PL/SQL procedure successfully completed.
```

## Die while-loop-Anweisung



Die while-loop-Anweisung fügt eine Bedingung zu einer Schleife hinzu.

```
WHILE condition LOOP
statement; ... statement;
END LOOP;
```

Die Variablen sind folgendermaßen definiert:

- *condition* ist eine gültige PL/SQL-Bedingung.
- *statement* ist eine gültige PL/SQL-Anweisung.

Listing 9.5 zeigt die Verwendung der while-loop-Anweisung.

### ***Listing 9.5: Verwendung der while-loop-Anweisung.***

```
WHILE I < 100 LOOP
I := I + 1;
insert into temp_table (rec_number) values (I);
END LOOP;
```

## **Die for-loop-Anweisung**

Die for-loop-Anweisung ist der while-loop-Anweisung recht ähnlich.



Hier ist die Syntax:

```
FOR loop-variable IN [REVERSE] lower-bound..upper-bound LOOP
statement; ... statement;
END LOOP;
```

Die Variablen sind folgendermaßen definiert:

- *loop\_variable* ist eine Integervariable, die als Zähler dient.
- *lower-bound* ist die untere Grenze des Zählbereichs.
- *upper-bound* ist die obere Grenze des Zählbereichs.

*reverse* ist ein optionales Schlüsselwort, das, wenn es angegeben wird, dafür sorgt, daß die Schleife von *upper-bound* bis *lower-bound* abwärts gezählt wird.

Listing 9.6 zeigt die Verwendung der for-loop-Anweisung.

### ***Listing 9.6: Verwendung einer for-loop-Anweisung.***

```
for i in 1..max_loops loop
j := j + j;
dbms_output.put_line('j: ' || to_char(j));
end loop;
```

## **Die goto-Anweisung**

Ja, PL/SQL erlaubt es Ihnen, die potentiell gefährliche goto-Anweisung zu verwenden. Um goto zu

verwenden, müssen Sie natürlich ein Label angeben, zu dem der Programmablauf springen kann. In PL/SQL wird ein Label folgendermaßen definiert:

```
<<my_label>>
```

Listing 9.7 zeigt ein Beispiel einer goto-Anweisung, ein Label kann als Alternative zu einer exit-Anweisung verwendet werden.

***Listing 9.7: Verwendung der goto-Anweisung.***

```
SQL> declare
2
2 i positive := 1;
3 max_loops constant positive := 100;
4
4 begin
5
5 i := 1;
6
6 loop
7
7 i := i + 1;
8 if i > max_loops then
9 goto more_processing;
10 end if;
11
11 end loop;
12
12 <<more_processing>>
13 i := 1;
14
14 end;
15 /
PL/SQL procedure successfully completed.
```

## Die null-Anweisung

In bestimmten Situationen sollten Sie PL/SQL anzeigen, daß keine Aktion durchgeführt werden soll. In einem Exception-Handler kann es beispielsweise vorkommen, daß Sie nichts tun wollen, wenn eine bestimmte Exception auftritt. Um Klarheit zu schaffen, verwenden Sie die null-Anweisung in einem if-then-elsif, [um anzuzeigen, daß für eine bestimmte elsif-Klausel keine Aktion durchgeführt werden soll](#).

Unglücklicherweise hat Oracle dieser Anweisung den Namen null [gegeben, obwohl sie nichts mit dem Nullwert zu tun hat](#). Listing 9.8 zeigt, wie die null-Anweisung eingesetzt werden kann.

***Listing 9.8: Verwendung der null-Anweisung.***

```
if (mod(i,10) = 0) then
i := i + 1;
else
NULL;
end if;
```

## Die Zuweisungsanweisung

Wie Sie bereits gesehen haben, verwendet PL/SQL ein :=, um einer PL/SQL-Variablen einen Wert zuzuweisen. Sie können den Standardwert einer Konstanten oder Variablen im Deklarationsteil angeben. Eine Bemerkung: Sie können einer Variablen, die mit der Notation %type deklariert wurde, nicht null [zuweisen, wenn die referenzierte Spalte als not null definiert ist](#).

## Einfügen von Kommentaren in ein PL/SQL-Unterprogramm

PL/SQL bietet Ihnen zwei Wege, Ihre Codes zu dokumentieren. Zum einen können Sie in jeder Zeile einen Kommentar angeben, indem Sie -- , gefolgt vom Kommentar, angeben, wie hier gezeigt wird:

```
Depot_Est_Row.Technician := Last_Tech_Name;
-- Assign the name of the last technician involved
```

Sie können auch Kommentare im *C-Stil* verwenden - indem Sie diese zwischen /\* und \*/ einschließen. Diese Methode ist am besten für mehrzeilige Kommentare geeignet, wie in Listing 9.9 gezeigt wird.

***Listing 9.9: Kommentieren des PL/SQL-Codes.***

```
j + 1;
/* The next section inserts a row into the Utility_Audit table
to record the name of the current Oracle user and the
current date and time (SYSDATE).
*/
insert into Utility_Audit
...
```

## SQL-Anweisungen in einem PL/SQL-Programm verwenden

Sie können SQL-Anweisungen in einem anonymen Block, in einer Prozedur oder in Funktionen, wie Sie in SQL\*Plus angewandt werden, mit nur wenigen Abweichungen verwenden. Wie andere PL/SQL-Anweisungen wird jede SQL-Anweisung mit einem Semikolon beendet. Sie können jedoch in PL/SQL in einer SQL-Anweisung auf deklarierte Variablen zugreifen. Listing 9.10 zeigt an einem Beispiel, wie man in einer SQL-Anweisung auf deklarierte Variablen zugreift.

### *Listing 9.10: Zugriff auf Variablen in einer SQL-Anweisung.*

```
DECLARE
max_records CONSTANT int := 100;
i int := 1;
BEGIN
FOR i IN 1..max_records LOOP
if (mod(i,10) = 0) then
INSERT INTO test_table
(record_number, current_date)
VALUES
(i, SYSDATE);
else
NULL;
end if;
END LOOP;
COMMIT;
END;
/
```

In diesem Beispiel verwendet die insert-Anweisung die numerische Variable i und die Pseudospalte sysdate, um die Werte in den Spalten Record\_Number und Current\_Date zu plazieren.

## PL/SQL und die select-Anweisung

In einem PL/SQL-Unterprogramm verwendet die select-Anweisung eine andere Klausel - into - um die PL/SQL-Variablen zu identifizieren, in die Spaltenwerte eingetragen werden sollen. Stellen Sie die into-Klausel zwischen die Auswahlliste und die from-Klausel. Listing 9.11 enthält ein Beispiel für einen anonymen PL/SQL-Block, der eine select-Anweisung enthält.

### *Listing 9.11: Verwendung einer select-Anweisung in einem PL/SQL-Block.*

```
SQL> set serveroutput on
SQL>
SQL> declare
```



```

2
2 Average_Body_Temp Patient.Body_Temp_Deg_F%type;
3
3 begin
4
4 dbms_output.enable;
5
5 select avg(Body_Temp_Deg_F)
6 into Average_Body_Temp
7 from Patient;
8
8 dbms_output.put_line('Average body temp in Deg. F: ' ||
to_char(Average_Body_Temp, '999.99'));
9
9 end;
10 /
Average body temp in Deg. F: 99.80
PL/SQL procedure successfully completed.

```

## PL/SQL-Unterprogramme

PL/SQL unterstützt auch die Verwendung von Unterprogrammen - benannte Prozeduren und Funktionen. Eine PL/SQL-Prozedur führt diverse Aktionen durch und kann optionale Parameter haben. Eine PL/SQL-Funktion gibt einen Wert mit einem bestimmten Datentyp zurück und kann ebenfalls optionale Parameter haben.

## Verwenden von Unterblöcken

PL/SQL ermöglicht es Ihnen, in einen Block Unterblöcke einzuschließen. Listing 9.12 zeigt beispielsweise einen anonymen Block, der einen anderen anonymen Unterblock enthält, welcher seinen eigenen Deklarationsteil hat.

*Listing 9.12: Beispiel für einen Unterblock.*

```

SQL> declare
2
2 max_i constant int := 100;
3 i int := 1;
4 rec_number int;
5
5 begin

```

```

6
6 for i in 1..max_i loop
7
7 if mod(i,5) = 0 then
8 rec_number := 5;
9 elsif mod(i,7) = 0 then
10 rec_number := 7;
11 else
12 rec_number := i;
13 end if;
14
14 insert into test_table
15 (record_number, current_date)
16 values
17 (rec_number, sysdate);
18
18 - Here is a sub-block:
19
19 declare
20 max_j constant int := 20;
21 j int := 1;
22
22 begin
23
23 for j in 1..max_j loop
24
24 rec_number := rec_number * j;
25
25 insert into test_table
26 (record_number, current_date)
27 values
28 (rec_number, sysdate);
29
29 end loop;
30
30 end;
31
31 commit;
32 end loop;
33
33 end;
34 /
PL/SQL procedure successfully completed.
SQL> select count(*) from test_table;
COUNT(*)
-----

```



*Die erste Zeile von Listing 9.12 startet den Deklarationsteil des anonymen Blocks. In Zeile 5 beginnt der Ausführungsteil des Hauptblocks. Der Deklarationsteil des Unterblocks beginnt in Zeile 19; Zeile 22 markiert den Beginn des Ausführungsteils des Unterblocks.*



*Obwohl PL/SQL die Möglichkeit unterstützt, Blöcke ineinander einzubetten, ist diese Praxis aus zwei Gründen nicht wünschenswert. Zum einen reduziert sie die Lesbarkeit - und damit die Wartbarkeit - Ihres Codes. Zum anderen können eingebettete Blöcke nicht von anderen PL/SQL-Unterprogrammen verwendet werden. Sie sollten darauf achten, Prozeduren und Funktionen so zu entwerfen, daß die Wiederverwendbarkeit des Codes und seine Wartbarkeit erhöht werden.*

## Deklarieren einer Prozedur

Zusätzlich zu anonymen Blöcken können Sie in PL/SQL auch Prozeduren und Funktionen deklarieren.



*Die Syntax für die Deklaration einer Prozedur ist*

```
PROCEDURE procedure-name [(argument1 ... [, argumentN) ] IS
[local-variable-declarations]
BEGIN
executable-section
[exception-section]
END [procedure-name];
```



Die Variablen sind folgendermaßen definiert:

- *procedure-name* ist der Name der Prozedur, er unterliegt den Einschränkungen von Oracle-Datenbanken bezüglich der Namensgebung von Objekten.
  - *argument1* bis *argumentN* sind optionale Deklarationen von Argumenten, diese sind folgendermaßen aufgebaut:
  - *argument-name* [IN | OUT] *datatype* [ {:= | DEFAULT} *value* ]
  - *local-variable-declarations* sind optionale Deklarationen von Variablen, Konstanten und anderen Prozeduren, die lokal für *procedure-name* sind.
  - *executable-section* sind die PL/SQL-Anweisungen, aus denen sich die Prozedur zusammensetzt.
  - *exception-section* ist der optionale Abschnitt zur Behandlung von Exceptions in der Prozedur.
- 



Die Unterscheidung zwischen gespeicherten Prozeduren und Prozeduren, die in anonymen Blöcken deklariert und verwendet werden, ist wichtig. Die Prozeduren, die in anonymen Blöcken deklariert und aufgerufen werden, sind temporär; wenn die Ausführung der anonymen Blöcke beendet ist, sind sie für Oracle nicht mehr vorhanden. Eine gespeicherte Prozedur, die mit *create procedure* erzeugt wurde oder in einem Paket enthalten ist, ist permanent in dem Sinn, daß sie von einem SQL\*Plus-Skript, einem PL/SQL-Unterprogramm oder einem Datenbanktrigger aufgerufen werden kann.

---

Um diese Syntax zu verdeutlichen, enthält Listing 9.13 ein Beispiel für einen anonymen Block, der eine Prozedur namens *Record\_Patient\_Temp\_Deg\_C* deklariert. Diese Prozedur hat zwei Argumente: die ID des Patienten und die Körpertemperatur des Patienten, gemessen in Grad Celsius (siehe Zeile 4). In Zeile 3 wird die Variable *High\_Fever* auf 42 initialisiert. In Zeile 17 wird die Prozedur mit zwei Argumenten aufgerufen: *New\_Patient\_ID* (mit dem Wert GG9999) und *High\_Fever*. Die *select*-Anweisung, die auf den anonymen Block folgt, demonstriert, daß die Prozedur ihre Aufgabe erfüllt hat - sie hat 42 Grad Celsius in 107.6 Grad Fahrenheit umgewandelt.

**Listing 9.13: Beispiel für eine Prozedur.**

```
SQL> declare
2
2 New_Patient_ID Patient.Patient_ID%type;
3 High_Fever constant real := 42.0;
4
4 procedure Record_Patient_Temp_Deg_C (Patient_ID varchar2,
```

```

5 Body_Temp_Deg_C real) is
6
6 Temp_Deg_F real;
7
7 begin
8
8 Temp_Deg_F := (9.0/5.0)*Body_Temp_Deg_C + 32.0;
9
9 insert into Patient
10 (Patient_ID, Body_Temp_Deg_F)
11 values
12 (Patient_ID, Temp_Deg_F);
13
13 commit;
14 end;
15
15 begin
16
16 New_Patient_ID := 'GG9999';
17
17 Record_Patient_Temp_Deg_C (New_Patient_ID, High_Fever);
18
18 end;
19 /
PL/SQL procedure successfully completed.

```

```

SQL> select Patient_ID, Body_Temp_Deg_F
2 from Patient
3 where
4 Patient_ID = 'GG9999';
PATIEN BODY_TEMP_DEG_F
-----
GG9999 107.6

```

Listing 9.14 zeigt, daß Variablen, die innerhalb einer Prozedur deklariert sind, außerhalb der Prozedur nicht zugänglich sind.

*Listing 9.14: Beispiel für den Gültigkeitsbereich von Variablen in PL/SQL.*

```

SQL> declare
2
2 procedure Delete_Patients is
3
3 Temp_Deg_F real;
4
4 begin

```

```

5
5 delete from Patient
6 where
7 Patient_ID = 'GG3333';
8
8 commit;
9
9 end;
10
10 begin
11
11 Temp_Deg_F := 100.0;
12
12 end;
13 /
Temp_Deg_F := 100.0;
*
ERROR at line 11:
ORA-06550: line 11, column 1:
PLS-00201: identifier 'TEMP_DEG_F' must be declared
ORA-06550: line 11, column 1:
PL/SQL: Statement ignored

```

---



*Listing 9.14 beginnt mit dem Deklarationsteil eines anonymen Blocks. In Zeile 2 wird die Prozedur Delete\_Patients deklariert; innerhalb dieser Prozedur wird die Variable Temp\_Deg\_F als real vereinbart. In Zeile 11 referenziert der anonyme Block jedoch Temp\_Deg\_F, das Ergebnis ist die PL/SQL-Fehlermeldung PLS-00201. Dieses Listing zeigt, daß der Gültigkeitsbereich einer lokalen Variablen in einer Prozedur oder Funktion nicht bis zu einem anderen Unterprogramm reicht, das diese Prozedur oder Funktion aufruft.*

---

Weitere Informationen über gespeicherte Prozeduren gibt die Lektion am Tag 11.

## Deklarieren einer Funktion

Die Deklaration einer Funktion in PL/SQL ähnelt der Deklaration einer Prozedur - außer, daß die Funktion einen Wert mit einem vordefinierten Datentyp zurückgibt.



Die Syntax für die Deklaration einer Funktion ist:

```
FUNCTION function-name [(argument1 ... [, argumentN) ]
RETURN function-datatype IS
[local-variable-declarations]
BEGIN
executable-section
[exception-section]
END [function-name];
```

Die Variablen sind folgendermaßen definiert:

- *function-name* ist der Name der Funktion, er unterliegt den Einschränkungen von Oracle-Datenbanken bezüglich der Namensgebung von Objekten.
- *argument1* bis *argumentN* sind optionale Deklarationen von Argumenten, diese sind folgendermaßen aufgebaut:
- *argument-name* [IN | OUT] *datatype* [ {:= | DEFAULT} *value*]
- *local-variable-declarations* sind optionale Deklarationen von Variablen, Konstanten und anderen Prozeduren, die lokal für *function-name* sind.
- *executable-section* sind die PL/SQL-Anweisungen, aus denen sich die Funktion zusammensetzt.
- *exception-section* ist der optionale Abschnitt zur Behandlung von Exceptions in der Funktion.

Listing 9.15 bietet ein Beispiel für die Deklaration und Verwendung einer Funktion. Die Funktion `Max_Additional_Fees` hat ein einziges Argument, `Dept_ID` (siehe Zeile 3). Die Funktion gibt die Course-ID des Kurses mit den höchsten zusätzlichen Gebühren aller Kurse in der angegebenen Abteilung zurück (die select-Anweisung beginnt in Zeile 9). Der anonyme Block ruft die Funktion mit dem Wert `econ` für die `Dept_ID` auf (siehe Zeile 24) und die Funktion gibt die `Course_ID` 189 zurück. Die Abfrage am Ende von Listing 9.15 zeigt, daß Nummer 189 tatsächlich die höchsten zusätzlichen Gebühren hat.

**Listing 9.15: Beispiel für eine Funktion.**

```
SQL> declare
2
2 Course_ID Course.Course_ID%type;
3
3 function Max_Additional_Fees (Dept_ID IN varchar2)
4 return varchar2 is
```

```

5
5 Additional_Fees Course.Additional_Fees%type;
6 Units Course.Units%type;
7 Course_ID Course.Course_ID%type;
8
8 begin
9
9 select Course_ID
10 into Course_ID
11 from Course
12 where
13 Department_ID = Dept_ID and
14 Additional_Fees =
15 (select max(Additional_Fees)
16 from Course
17 where
18 Department_ID = Dept_ID);
19
19 return Course_ID;
20
20 end;
21
21 - Beginning of executable section of anonymous block.
22
22 begin
23
23 dbms_output.enable;
24
24 Course_ID := Max_Additional_Fees ('ECON');
25
25 dbms_output.put_line('Course_ID: ' || Course_ID);
26
26 end;
27 /
Course_ID: 189

```

PL/SQL procedure successfully completed.

```

SQL> select Course_ID, Additional_Fees
2 from Course
3 where
4 Department_ID = 'ECON'
5 order by Course_ID;

```

```

COURS ADDITIONAL_FEES
-----
101 25

```



189 750

199 0

## Prozedur- und Funktionsargumente

Jedes Argument einer Prozedur oder Funktion kann optional aus einer der folgenden Möglichkeiten gewählt werden:

**in** Der Wert des Arguments wird an die Prozedur oder Funktion übergeben, es wird jedoch kein Wert an das PL/SQL-Unterprogramm zurückgegeben. Innerhalb einer Prozedur oder Funktion können Sie einer als **in** deklarierten Variablen keinen Wert zuweisen; Sie können nur auf den Wert des Arguments zugreifen.

**out** Die Prozedur oder Funktion verwendet den übergebenen Wert nicht, sie gibt jedoch einen Wert an das aufrufende PL/SQL-Unterprogramm zurück. Innerhalb einer Prozedur oder Funktion können Sie nicht auf den Wert einer als **out** **deklarierten Variablen zugreifen; Sie können dieser Art von Argument lediglich einen Wert zuweisen.**

**in out** Der Wert des Arguments wird an die Prozedur oder Funktion übergeben und wird auch an das aufrufende PL/SQL-Unterprogramm zurückgegeben. Auf ein Argument, das als **in out** deklariert ist, kann innerhalb seiner Prozedur oder Funktion zugegriffen werden, oder ihm kann ein Wert zugewiesen werden.

Listing 9.16 zeigt ein Beispiel dafür, wie alle drei Arten von Argumenten verwendet werden.

**Listing 9.16: Verwendung unterschiedlicher Arten von Argumenten - *in*, *out* und *in out*.**

```
SQL> declare
2
2 This_Arg1 number;
3 This_Arg2 number;
4 This_Arg3 number;
5
5 procedure Different_Arguments
6 (arg1 IN number,
7 arg2 OUT number,
8 arg3 IN OUT number) is
9
```

```
9 begin
10
10 arg2 := arg1;
11 arg3 := arg3 + 1;
12
12 end;
13
13 - Beginning of executable section of anonymous block.
14
14 begin
15
15 This_Arg1 := 3.14159;
16
16 Different_Arguments (This_Arg1, This_Arg2, This_Arg3);
17
17 end;
18 /
PL/SQL procedure successfully completed.
```

## Zusammenfassung

In dieser Lektion wurden die folgenden grundlegenden Merkmale von PL/SQL besprochen:

- PL/SQL ist eine blockstrukturierte Sprache, in der ein Block aus einem optionalen Deklarationsteil, einem Ausführungsteil und einem optionalen Teil zur Behandlung von Exceptions besteht.
- In den SQL-Erweiterungen von PL/SQL gibt es einige zusätzliche Datentypen: `binary_integer`, `natural`, `positive`, `%type`, `%rowtype`, `table` und `record`.
- Die ausführbaren PL/SQL-Anweisungen umfassen: `if-then-elsif`, `loop`, `exit`, die `while-loop`, die `for-loop` und `goto`.
- Ein PL/SQL-Unterprogramm kann SQL-Anweisungen enthalten, die auf PL/SQL-Variablen zugreifen.
- Ein PL/SQL-Unterprogramm kann seine eigenen Prozeduren und Funktionen deklarieren.

### Was Sie tun sollen

Verwenden Sie PL/SQL, um Vorgaben für die Daten in der Oracle-Datenbank durchzusetzen.

### Was nicht

Verwenden Sie PL/SQL, um Pakete zu konstruieren, die von Anwendungen aufgerufen werden können, welche mit Forms, Reports und Graphics von Oracle geschrieben wurden.

Konstruieren Sie keine PL/SQL-Unterprogramme, die Eingaben des Benutzers direkt behandeln; verwenden Sie statt dessen ein Entwicklungswerkzeug, das dem Benutzer eine grafische Benutzeroberfläche (Graphical User Interface, GUI) bietet.

## Wie geht es weiter?

Am Tag 10 lernen Sie viel mehr über die Deklaration und Verwendung von Prozeduren, Funktionen und Paketen.

## Fragen und Antworten

Frage:

Kann eine Funktion andere Prozeduren und Funktionen aufrufen und kann eine Prozedur andere Prozeduren und Funktionen aufrufen?

Antwort:

*Ja, das wird von PL/SQL unterstützt. PL/SQL unterstützt sogar Rekursion.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Nennen Sie die drei Teile eines PL/SQL-Unterprogramms.

2. Richtig oder falsch? Eine PL/SQL-Variable, die einen Spaltenwert speichert, muß denselben Namen wie die Spalte haben.
3. Warum ist es eine gute Angewohnheit, bei der Deklaration von Variablen %type zu verwenden?

## Übungen

Konstruieren Sie einen anonymen Block, der eine Prozedur deklariert, die für einen Kurs eine zusätzliche Gebühr von DM 50,- festlegt, wenn es keine laufenden Gebühren gibt. Die Prozedur sollte nur ein Argument haben - `Department_ID` - und sollte diese Aktion nur ausführen, wenn der Wert von `Department_ID` angegeben wird.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖ Kapitel ❖ Top Inhalt Index **SAMS** Kapitel ❖

# 10

# Programmentwicklung mit PL/SQL

[Erstellen einer gespeicherten Prozedur oder Funktion](#)

[Abrufen einer gespeicherten Prozedur](#)

[Vorwärtsdeklaration für Prozeduren und Funktionen](#)

[Verwenden gespeicherter Funktionen in einer SQL-Anweisung](#)

[Speichern von Rückgabewerten in einer Tabelle](#)

[Aufrufen einer gespeicherten Prozedur oder Funktion](#)

[Pakete](#)

[Zusätzliche PL/SQL-Datentypen](#)

[Festlegen von Standardwerten für Variablen](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

Während der letzten Lektion haben Sie etwas über die in PL/SQL verwendeten Programmkonstrukte gelernt. In dieser Lektion lernen Sie, wie man PL/SQL-Module erstellt - Prozeduren, Funktionen und

**Pakete** - die in einer Oracle-Datenbank gespeichert werden. Sie lernen auch etwas über PL/SQL-Datentypen, die in SQL nicht verfügbar sind.

Eine gespeicherte Prozedur oder Funktion ist ein PL/SQL-Programm, das in einer Oracle-Datenbank gespeichert ist und von einem Benutzer aufgerufen wird, entweder direkt oder indirekt. Die Vorteile der Verwendung gespeicherter Prozeduren und Funktionen sind:

- **Effizienz:** In einer Client-Server-Architektur schickt der Client SQL-Anforderungen an einen Datenbank-Server. Mit der Zahl der Benutzer nimmt auch die Zahl der Anforderungen zu und das Netzwerk kann schnell zum Flaschenhals für die Systemleistung werden. Die Verwendung gespeicherter Prozeduren bietet eine erhebliche Verbesserung der Systemleistung, da ein einziger Aufruf einer gespeicherten Prozedur mehrere SQL-Anweisungen aufrufen kann, die auf dem Server ausgeführt werden, so daß der Netzwerkverkehr reduziert wird.
- **Wiederverwendbarkeit:** Ein PL/SQL-Programm wird einmal geschrieben und in einer Vielzahl von Situationen verwendet - in SQL-Skripten, in Datenbank-Triggern und in der Anwendungslogik von Clients.
- **Portabilität:** Sie können eine gespeicherte Prozedur in jeder Oracle8-Datenbank verwenden, unabhängig von der Plattform. Als Folge müssen Sie sich nicht um Kompatibilitätsfragen wie das Betriebssystem oder Compiler-Versionen kümmern. Wenn die Plattform Oracle8 unterstützt, kann die gespeicherte Prozedur ohne zusätzliche Änderungen auf diese portiert werden. Wenn die gespeicherte Prozedur Verweise auf Datei- und Verzeichnisnamen enthält, müssen Sie natürlich eventuell zusätzliche Änderungen vornehmen.
- **Wartungsfreundlichkeit:** Eine gespeicherte Prozedur ist für eine bestimmte Tätigkeit entworfen. Diese Tätigkeit kann von einem Datenbank-Trigger, einem SQL\*Plus-Skript, einem Anwendungsprogramm oder einer anderen gespeicherten Prozedur benötigt werden. Indem aus allen diesen Quellen die gespeicherte Prozedur aufgerufen wird, können Sie die Kosten für die Wartung der Software reduzieren.



*Sie können verschiedene Werkzeuge verwenden, um gespeicherte Prozeduren, Funktionen und Pakete zu erzeugen und zu warten. In dieser Lektion lernen Sie, wie man dafür SQL\*Plus oder SQL Worksheet verwendet. Ein Werkzeug wie Oracle Procedure Builder, eine Komponente von Developer/2000, hat jedoch mehrere Vorteile. Zuerst einmal besitzt Procedure Builder einen Editor für gespeichertes PL/SQL, der es erleichtert, PL/SQL-Code zu vereinfachen. Zweitens können Sie Ihr gespeichertes Programm übersetzen, indem Sie einfach einen Knopf im Editor drücken. Drittens werden alle Übersetzungsfehler bequem in einem separaten Fenster angezeigt. Es gibt auch Editoren von Drittherstellern, die Sie testen können, bevor Sie sich in ein größeres PL/SQL-Projekt stürzen.*

---

## Erstellen einer gespeicherten Prozedur oder Funktion

Es empfiehlt sich, beim Erstellen einer gespeicherten Prozedur einen Texteditor wie Notepad oder WordPad zu verwenden. Besser ist die Verwendung einer der Komponenten von Oracle Developer/2000 - dem Procedure Builder - der in der Lektion von Tag 16, »Developer/2000: Oracle Graphics und den Procedure Builder verwenden«, vorgestellt wird. Wenn Notepad oder WordPad geöffnet ist, können Sie die gespeicherte Prozedur kopieren und Sie zum Entwickeln und Testen in SQL\*Plus oder SQL Worksheet einfügen.



*Die Syntax für das Erzeugen einer gespeicherten Prozedur ist folgendermaßen:*

---

```
CREATE [OR REPLACE] PROCEDURE procedure-name
[(argument1 ... [, argumentN) ] IS
[local-variable-declarations]
BEGIN
executable-section
[exception-section]
END [procedure-name];
```

Die Variablen sind folgendermaßen definiert.

- *procedure-name* ist der Name der Prozedur, entsprechend den Einschränkungen von Oracle für die Benennung von Objekten.
- *argument1* bis *argumentN* sind optionale Deklarationen von Argumenten, sie bestehen aus

*argument-name* [IN | OUT] *datatype* [ {:= | DEFAULT} *value*]

- *local-variable-declarations* sind optionale Deklarationen von Variablen, Konstanten und anderen Prozeduren und Funktionen, die für *procedure-name* lokal sind.
- *executable-section* sind die PL/SQL-Anweisungen, aus denen sich die Prozedur zusammensetzt.
- *exception-section* ist der optionale Teil für die Behandlung von Exceptions in der Prozedur.

Als Beispiel enthält Listing 10.1 eine gespeicherte Prozedur, die ein einzelnes Argument hat, das von der delete-Anweisung verwendet wird, um zu bestimmen, welche Klasse aus der Tabelle Course entfernt wird.

***Listing 10.1: Erstellen einer gespeicherten Prozedur mit einem einzigen Argument.***

```
SQL> create or replace procedure Delete_Specified_Course
2 (Description_Phrase varchar2) is
3
4 begin
5
6 delete from Course
7 where
8 upper(Description) like Description_Phrase;
9
10 end;
11 /
Procedure created.
```

---



*Die Syntax für das Erzeugen einer gespeicherten Funktion ist sehr ähnlich wie die Syntax für das Erzeugen einer gespeicherten Prozedur. Natürlich muß eine gespeicherte Funktion einen Wert zurückgeben.*

---

```
CREATE [OR REPLACE] FUNCTION function-name
[ (argument1 ... [, argumentN) ]
RETURN function-datatype IS
[ local-variable-declarations ]
BEGIN
executable-section
[ exception-section ]
RETURN function-value
END [ function-name ];
```

Die Variablen sind folgendermaßen definiert.

- *function-name* ist der Name der Funktion entsprechend den Einschränkungen von Oracle für die Benennung von Objekten.
- *argument1* bis *argumentN* sind optionale Deklarationen von Argumenten, sie bestehen aus

*argument-name* [IN | OUT] *datatype* [ {:= | DEFAULT} *value* ]

- *function-datatype* ist der Datentyp des von der Funktion zurückgegebenen Werts.
- *local-variable-declarations* sind optionale Deklarationen von Variablen, Konstanten und anderen Prozeduren und Funktionen, die für *function-name* lokal sind.
- *executable-section* sind die PL/SQL-Anweisungen, aus denen sich die Funktion



zusammensetzt.

- exception-section ist der optionale Teil für die Behandlung von Exceptions in der Funktion.
- function-value ist der Wert, den die Funktion an die aufrufende Instanz zurückgibt.



*Der Unterschied zwischen einer gespeicherten Prozedur und einer gespeicherten Funktion ist, daß die gespeicherte Prozedur keinen Wert zurückgibt, während eine gespeicherte Funktion einen Wert zurückliefert. Als Folge kann eine gespeicherte Funktion in einer SQL-Anweisung auf die gleiche Weise wie eine integrierte Funktion aufgerufen werden; bei einer gespeicherten Prozedur ist das nicht möglich. Gespeicherte Prozeduren und Funktionen können jedoch beide einen geänderten Argumentwert zurückliefern, wenn das Argument als out oder in out deklariert wurde.*

---

Listing 10.2 zeigt, wie man eine gespeicherte Funktion erzeugt, die den Punktedurchschnitt eines Studenten liefert.

***Listing 10.2: Erstellen einer gespeicherten Funktion mit einem einzigen Argument.***

```
create or replace function student_GPA (arg_student_ID IN varchar2)
return number is

GPA number;

begin

select avg(decode(grade, 'A+', 4.25, 'A', 4, 'A-', 3.75,
'B+', 3.25, 'B', 3, 'B-', 2.75,
'C+', 2.25, 'C', 2, 'C-', 1.75,
'D+', 1.25, 'D', 1, 'D-', 0.75, 'F', 0))
into GPA
from student_schedule
where
student_id = arg_student_id;

return GPA;

end;
```

---



*Viele der Beispiele in dieser Lektion rufen ein Paket namens `dbms_output` auf. Dieses Paket bietet eine Menge von Prozeduren und Funktionen, die nützlich sind, um Werte von einem PL/SQL-Block aus anzuzeigen. Um das Paket mit SQL\*Plus zu verwenden, geben Sie die folgende Anweisung ein, bevor die erste Prozedur oder Funktion aus `dbms-output` aufgerufen wird:*

---

```
set serveroutput on
```

## Ermitteln von Fehlermeldungen bei Erstellen gespeicherter Prozeduren

Wenn Oracle beim Erzeugen eines gespeicherten PL/SQL-Programms Fehler entdeckt, gibt es eine undefinierbare Meldung aus, die angibt, daß Fehler aufgetreten sind - ohne irgendwelche zusätzlichen Details anzugeben. Als Beispiel zeigt Listing 10.3, was geschieht, wenn Sie eine gespeicherte Prozedur mit einem Syntaxfehler erzeugen wollen. Um die Fehler zu sehen, die bei dem Versuch entstehen, diesen PL/SQL-Code zu kompilieren, können Sie den SQL\*Plus-Befehl `show errors` verwenden, dieser zeigt die speziellen PL/SQL-Übersetzungsfehler.

***Listing 10.3: Eine während der Kompilierung einer gespeicherten Prozedur von Oracle zurückgegebene Fehlermeldung.***

```
SQL> create or replace procedure show_inserts IS
2
2 max_records constant int := 100;
3 i int := 1;
4
4 begin
5
5 dbms_output.enable;
6
6 for i in 1..max_records loop
7
7 if (mod(i,10) = 0) then
8 insert into test_table
9 (record_number, current_date)
10 values
11 (i, SYSDATE)
12 dbms_output.put_line('The value of i is ' || to_char(i));
13
13 else
14 null;
```

```
15
15 end if;
16
16 end loop;
17
17 end;
18 /
```

Warning: Procedure created with compilation errors.

```
SQL> show errors
```

```
Errors for PROCEDURE SHOW_INSERTS:
```

```
LINE/COL ERROR
```

```
-----
12/5 PLS-00103: Encountered the symbol "DBMS_OUTPUT" when
expecting one of the following:
;
; was inserted before "DBMS_OUTPUT" to continue.
```



*Beim Versuch, die Prozedur `show_inserts` zu erstellen oder zu ersetzen, gibt Oracle direkt nach Zeile 18 eine Fehlermeldung zurück, die anzeigt, daß es Übersetzungsfehler gab. Um den genauen Fehler zu sehen, geben Sie den Befehl `show errors` ein. Wie Sie in Zeile 11 sehen können, endet die insert-Anweisung nicht mit einem Semikolon.*

---

## Abrufen einer gespeicherten Prozedur

Nachdem eine gespeicherte Prozedur erzeugt wurde, möchten Sie vielleicht einen Blick auf den Code des PL/SQL-Programms werfen. Auch wenn das SQL-Skript, das für das Erstellen der gespeicherten Prozedur verwendet wurde, nicht verfügbar ist, können Sie den Code einer gespeicherten Prozedur immer noch durch eine Abfrage einer Datensicht des Oracle Data Dictionary erhalten.

Das Data Dictionary von Oracle ist eine Gruppe von Tabellen, die Informationen über die Oracle-Datenbank selbst enthalten. Weil die Tabellen dieses Data Dictionary eine etwas kryptische Struktur haben, definiert Oracle eine Menge von Datensichten, die einen kohärenteren Blick auf das Data Dictionary bieten (für weitere Informationen über Datensichten beziehen Sie sich bitte auf Tag 8, »Bilden fortgeschrittener Abfragen mit SQL«). Eine dieser Datensichten hat den Namen `USER_SOURCE` und enthält die folgenden vier Spalten:

|      |                                                                                                   |
|------|---------------------------------------------------------------------------------------------------|
| NAME | Enthält den Namen der Prozedur, der Funktion, des Pakets oder Pakettrumpfs                        |
| TYPE | Gibt an, ob der Code zu einer Prozedur, einer Funktion, einem Paket oder einem Pakettrumpf gehört |
| TEXT | Enthält eine Zeile des Codes                                                                      |
| LINE | Enthält die Zeilennummer des Codes in TEXT                                                        |

Stellen Sie sich als Beispiel vor, daß Sie eine gespeicherte Prozedur namens `drop_class` erstellen. Wenn Sie den Code von `drop_class` sehen wollen, fragen Sie die Datensicht `user_source` des Data Dictionary ab, wie dies in Listing 10.4 gezeigt wird.

***Listing 10.4: Abrufen des Codes für eine gespeicherte Prozedur.***

```
SQL> select line, text
2 from user_source
3 where
4 name = 'DROP_CLASS'
5 order by line;
```

```
LINE TEXT
```

```
-----
1 procedure drop_class (arg_student_ID IN varchar2,
2 arg_class_ID IN varchar2,
3 status OUT number) is
4
5 counter number;
6
7 begin
8
9 status := 0;
10
11 - Verify that this class really is part of the student's
schedule.
12
13 select count(*) into counter
14 from student_schedule
15 where
16 student_id = arg_student_id and
17 class_id = arg_class_id;
18
```

```
19 if counter = 1 then
20 delete from student_schedule
21 where
22 student_id = arg_student_id and
23 class_id = arg_class_id;
24 status := -1;
25 end if;
26
27 end;

27 rows selected.
```

## Ermitteln einer Liste von Prozeduren, Funktionen, Paketen und Paketrümpfen

Sie können USER\_OBJECTS abfragen, um eine Liste gespeicherter Prozeduren, Funktionen, Pakete und Paketrümpfe zu erhalten, die zu dem Oracle-Account gehört, mit dem Sie gerade verbunden sind. Wenn Sie alle diese Objekte, unabhängig von ihrem Eigentümer, sehen wollen, müssen Sie DBA\_OBJECTS anstelle von USER\_OBJECTS abfragen. Die Spalte OBJECT\_TYPE von DBA\_OBJECTS gibt den Typ des Objekts an: Tabelle, Datensicht, Prozedur usw.

Um eine Liste der Typen von Datenbankobjekten zu erhalten, die einem Benutzer gehören, verwenden Sie die in Listing 10.5 gezeigte Abfrage.

*Listing 10.5: Bestimmung der im Besitz des aktuellen Benutzers befindlichen Objekttypen.*

```
SQL> select distinct object_type
2 from user_objects;
```

```
OBJECT_TYPE
-----
FUNCTION
INDEX
PACKAGE
PACKAGE BODY
PROCEDURE
TABLE
VIEW

7 rows selected.
```

## Vorwärtsdeklaration für Prozeduren und Funktionen

In PL/SQL muß jeder Bezeichner deklariert werden - Konstanten, Variablen, Cursor oder Funktionen - , der an irgendeiner Stelle eines PL/SQL-Unterprogramms verwendet wird. Diese Anforderung kann ein Problem verursachen, wenn zwei Unterprogramme aufeinander verweisen, wie es in Listing 10.6 gezeigt wird.

***Listing 10.6: Aufruf einer Funktion vor ihrer Deklaration.***

```
SQL> set serveroutput on
SQL>
SQL> declare
2
2 function Medicare_Patient (Patient_ID IN varchar2)
3 return number is
4
4 status number;
5 Pat_ID varchar2(6);
6
6 begin
7
7 if Insurable_Patient (Pat_ID) = 2 then
8 status := 1;
9 end if;
10
10 return status;
11
11 end Medicare_Patient;
12
12
12 function Insurable_Patient (Patient_ID IN varchar2)
13 return number is
14
14 status number;
15 Pat_ID varchar2(6);
16
16 begin
17
17 if Medicare_Patient (Pat_ID) = 2 then
18 status := 1;
19 end if;
20
20 return status;
21
21 end Insurable_Patient;
22
22 - Executable portion of anonymous block.
23
```

```
23 begin
24
24 dbms_output.enable;
25
25 end;
26 /
declare
*
```

ERROR at line 1:  
ORA-06550: line 7, column 4:  
PLS-00313: 'INSURABLE\_PATIENT' not declared in this scope  
ORA-06550: line 7, column 1:  
PL/SQL: Statement ignored

---



Wie Sie in Listing 10.6 sehen können, erkennt PL/SQL den Aufruf von *Insurable\_Patient* in der Funktion *Medicare\_Patient* nicht an, weil die Deklaration von *Insurable\_Patient* nach der Deklaration von *Medicare\_Patient* erfolgt (Zeile 2). Um dieses Dilemma zu umgehen, nehmen Sie eine Vorwärtsdeklaration des Unterprogramms in den Deklarationsteil auf. Eine Vorwärtsdeklaration ist eine Deklaration eines Unterprogramms, seiner Argumente und seines Rückgabewerts. Listing 10.7 zeigt, wie man für das vorhergehende Beispiel eine Vorwärtsdeklaration für *Insurable\_Patient* (Zeile 2) angibt.

---

**Listing 10.7: Bereitstellung einer Vorwärtsdeklaration für eine Funktion.**

```
SQL> set serveroutput on
SQL>
SQL> declare
2
2 function Insurable_Patient (Patient_ID IN varchar2) return number;
3
3 function Medicare_Patient (Patient_ID IN varchar2)
4 return number is
5
5 status number;
6 Pat_ID varchar2(6);
7
7 begin
8
8 if Insurable_Patient (Pat_ID) = 2 then
```

```
9 status := 1;
10 end if;
11
11 return status;
12
12 end Medicare_Patient;
13
13
13 function Insurable_Patient (Patient_ID IN varchar2)
14 return number is
15
15 status number;
16 Pat_ID varchar2(6);
17
17 begin
18
18 if Medicare_Patient (Pat_ID) = 2 then
19 status := 1;
20 end if;
21
21 return status;
22
22 end Insurable_Patient;
23
23 - Executable portion of anonymous block.
24
24 begin
25
25 dbms_output.enable;
26
26 end;
27 /
PL/SQL procedure successfully completed.
```

## Verwenden gespeicherter Funktionen in einer SQL-Anweisung

Mit der Version 7.1 der Oracle-RDBMS wurde es möglich, eine gespeicherte Funktion in einer SQL-Anweisung aufzurufen. Dieses Merkmal ist sehr leistungsfähig, weil es die Funktionalität einer einzelnen SQL-Anweisung so ausdehnt, daß sie die Logik einer gespeicherten Funktion einschließt. Lassen Sie uns ein elementares Beispiel dafür betrachten, wie man diese Funktionalität erhält.

Weil es in Oracle keine integrierte Funktion für die Umwandlung einer Temperatur von Fahrenheit nach Grad Celsius gibt, erstellen Sie eine gespeicherte Funktion, die diese Umwandlung durchführt, wie dies in Listing 10.8 gezeigt wird. Nachdem die gespeicherte Prozedur erfolgreich erstellt wurde,



können Sie sie in einer select-Anweisung verwenden.

***Listing 10.8: Verwendung einer gespeicherten Funktion in einer select-Anweisung.***

```
SQL> create or replace function DegF_to_DegC (Deg_F IN number)
2 return number is
3
3 Deg_C number;
4
4 begin
5
5 Deg_C := (5.0/9.0)*(Deg_F - 32);
6
6 return Deg_C;
7
7 end DegF_to_DegC;
8 /
Function created.
```

```
SQL> select body_temp, degf_to_degc(body_temp)
2 from patient;
BODY_TEMP  DEGF_TO_DEGC(BODY_TEMP)
-----
99.2  37.333333
100.2  37.888889
103.8  39.888889
```

Wenn Sie eine gespeicherte Funktion erzeugt haben, kann sie jederzeit verwendet werden.

## Speichern von Rückgabewerten in einer Tabelle

Obwohl PL/SQL keine integrierte Unterstützung für die Kommunikation mit dem Benutzer besitzt, können Sie dennoch PL/SQL verwenden, um einem Benutzer oder einem anderen Programm Ergebnisse zu liefern, indem Sie

- Informationen in eine Zwischentabelle schreiben, die ein Benutzer oder ein Programm abfragen kann;
- die Prozeduren und Funktionen des von Oracle zur Verfügung gestellten Pakets dbms\_output verwenden.

Sie haben bereits ein Beispiel dafür gesehen, wie man mit PL/SQL eine Zwischentabelle schreiben kann. Wenn gespeicherte Prozeduren und Funktionen in PL/SQL kompiliert werden, schreibt der PL/SQL-Kern selbst Fehlermeldungen in eine Tabelle des Data Dictionary, die vom Benutzer

abgefragt werden kann. Wenn Sie über SQL\*Plus Ausgaben zur Verfügung stellen wollen, ist die Verwendung von `dbms_output` eine gute Strategie. Wenn Sie viele Werte an einen Benutzer oder ein Programm übergeben wollen, ist eine Tabelle sinnvoller.

## Aufrufen einer gespeicherten Prozedur oder Funktion

Die Methode für den Aufruf einer gespeicherten Prozedur hängt vom Umfeld ab.

Für SQL\*Plus verwenden Sie den Befehl `execute` (die folgende Syntax gilt für eine gespeicherte Prozedur ohne Argumente) auf folgende Weise:

```
execute show_inserts;
```

Aus einem PL/SQL-Unterprogramm rufen Sie die gespeicherte Prozedur oder Funktion mit allen benötigten Argumenten auf. Während Tag 14, »Developer/2000: Anwendungsentwicklung mit Hilfe von Oracle Forms«, werden Sie lernen, wie man eine gespeicherte Prozedur oder Funktion aus einer Oracle-Forms-Anwendung aufruft.

## Pakete

Ein Paket ist eine Gruppe verwandter Prozeduren und Funktionen in PL/SQL. Wie ein Paket der Programmiersprache Ada besteht ein PL/SQL-Paket aus einer Paketspezifikation und einem Pakettrumpf. Sie können Pakete erstellen, die anwendungsabhängig sind - beispielsweise könnte ein Paket namens *patient\_data* Prozeduren und Funktionen enthalten, die mit der Behandlung und Beschaffung von Informationen über die Patienten eines Krankenhauses zu tun haben. Außerdem könnte ein Paket Prozeduren und Funktionen enthalten, die häufig vorkommende Aufgaben wie die Umwandlung einer Ortsangabe von einem Koordinatensystem in ein anderes ausführen.



*Ein Paket ist eine Sammlung verwandter PL/SQL-Prozeduren und Funktionen, die in einer Oracle-Datenbank gespeichert wird. Um ein Paket zu erzeugen, müssen Sie eine Paketspezifikation und einen Pakettrumpf angeben.*

---

## Deklarieren eines Pakets



*Die allgemeine Syntax für das Erstellen einer Paketspezifikation ist:*

```
CREATE [OR REPLACE] PACKAGE package-name IS  
declaration-section  
END package-name;
```

Die Variablen sind folgendermaßen definiert:

- *package-name* ist der Name des zu erstellenden Pakets entsprechend den Einschränkungen von Oracle für die Benennung von Objekten.
- *declaration-section* besteht aus Deklarationen von Typen, Variablen, Cursors, Prozeduren und Funktionen.

Als Beispiel enthält Listing 10.9 die Paketspezifikation für das Informationssystem des Flugle-College.

***Listing 10.9: Deklaration einer Paketspezifikation.***

```
create or replace package Flugle is  
  
function register_for_class (arg_student_ID IN varchar2,  
arg_class_ID IN varchar2)  
return number;  
  
function schedule_conflict (arg_student_ID IN varchar2,  
arg_class_ID IN varchar2)  
return number;  
  
procedure drop_class (arg_student_ID IN varchar2,  
arg_class_ID IN varchar2,  
status OUT number);  
  
procedure assign_instructor (arg_class_ID IN varchar2,  
arg_instructor_ID IN varchar2,  
status OUT number);  
  
procedure assign_grade (arg_student_ID IN varchar2,  
arg_class_ID IN varchar2,  
arg_grade IN varchar2,
```

```
status OUT number);
```

```
function student_GPA (arg_student_ID IN varchar2)  
return number;
```

```
END;
```



*Das Flugle-Paket enthält sechs Teile: drei Prozeduren und drei Funktionen:*

---

- Die Prozedur drop\_class
  - Die Prozedur assign\_instructor
  - Die Prozedur assign\_grade
  - Die Funktion register\_for\_class
  - Die Funktion schedule\_conflict
  - Die Funktion student\_GPA
- 



*Wenn Sie Paketspezifikationen oder Paketrümpfe mit einem Skript erzeugen, verwenden Sie die or replace-Klausel. Oracle bietet auch die Anweisungen drop package und drop package body, aber die or replace-Klausel erspart Ihnen den Ärger, sich zu merken, ob Sie ein Paket gelöscht haben, bevor Sie versuchen, es zu erzeugen.*

---

## **Deklarieren eines Paketrümpfs**

Ein Paketrumpf enthält die öffentlichen und privaten Elemente eines Pakets. Er versteckt die Details, wie Cursors, Prozeduren und Funktionen konkret implementiert sind - Details, die vor Entwicklern verborgen werden sollten.

---



Ein Pakettrumpf wird mit der folgenden Syntax vereinbart:

---

```
CREATE PACKAGE BODY package-name IS
declaration-section
procedure-bodies;
function-bodies;
initialization-section
END package-name;
```

Die Paketvariablen sind folgendermaßen definiert:

- *package-name* ist der Name des zu erstellenden Pakets, entsprechend den Einschränkungen von Oracle für die Benennung von Objekten.
- *declaration-section* besteht aus Deklarationen von Typen, Variablen und Cursors.
- *procedure-bodies* besteht aus den ausführbaren Teilen der in der Paketspezifikation vereinbarten Prozeduren.
- *function-bodies* besteht aus den ausführbaren Teilen der in der Paketspezifikation vereinbarten Funktionen.
- *initialization-section* ist ein optionaler Abschnitt, der einmal beim ersten Aufruf des Pakets ausgeführt wird.

Listing 10.10 listet den Inhalt des Flugle-Pakettrumpfs auf, der die Details jeder Prozedur und Funktion enthält, die Teil dieses Pakets ist.

***Listing 10.10: Deklaration eines Pakettrumpfs.***

```
create or replace package flugle is
```

```
- Declare some exceptions.
```

```
schedule_conflict_exists exception;
already_registered exception;
not_registered exception;
```

```
- Declare values for status.
```

```
conflicting_classes number := -2;
unsuccessful number := -1;
normal number := 0;
```

```
- *****
- Function register_for_class
```

-

```
function register_for_class (arg_student_ID IN varchar2,  
arg_class_ID IN varchar2)  
return number is
```

```
status number;  
counter number;
```

```
begin
```

```
- Determine if the student isn't already registered for this class.
```

```
select count(*) into counter  
from student_schedule  
where  
student_id = arg_student_id and  
class_id = arg_class_id;
```

```
if counter > 0 then
```

-

```
- The student is already registered for this class.
```

-

```
raise already_registered;
```

```
else
```

-

```
- The student isn't registered for this class.
```

```
- Determine if there is a schedule conflict.
```

-

```
if schedule_conflict(arg_student_id, arg_class_id) = 0 then
```

```
insert into student_schedule
```

```
(student_id, class_id)
```

```
values
```

```
(arg_student_id, arg_class_id);
```

```
else
```

```
raise schedule_conflict_exists;
```

```
end if;
```

```
end if;
```

```
status := normal;
```

```
return status;
```

```
exception
```

```
when schedule_conflict_exists then
```

```
raise_application_error (-20001, 'Schedule conflict exists');
```

```
when already_registered then
```

```
raise_application_error (-20002, 'Student is already registered for  
class');
```

```
when others then
null;
end;

_ *****
- Function schedule_conflict
-
function schedule_conflict (arg_student_ID IN varchar2,
arg_class_ID IN varchar2)
return number is

- Declare a cursor to look for other classes with the same schedule
- as this one.

cursor get_other_classes is
select SS.Class_ID
from Student_Schedule SS, Class C
where
SS.Class_ID = C.Class_ID and
(C.Semester, C.School_Year, C.Schedule_ID) =
(select Semester, School_Year, Schedule_ID
from Class
where
Class_ID = arg_class_ID);

Conflicting_Class_ID Class.Class_ID%type;
status number;

begin

- Need to look at the other classes in the student's schedule
- for the same semester and school year.

for get_other_classes_Rec in get_other_classes loop

fetch get_other_classes into Conflicting_Class_ID;
exit when get_other_classes%notfound;

end loop;

close get_other_classes;

if get_other_classes%rowcount > 0 then
status := conflicting_classes;
else
status := normal;
end if;
```

```
return status;

end;

_ *****
- Procedure drop_class
-
procedure drop_class (arg_student_ID IN varchar2,
arg_class_ID IN varchar2,
status OUT number) is

counter number;

begin

- Verify that this class really is part of the student's schedule.

select count(*) into counter
from student_schedule
where
student_id = arg_student_id and
class_id = arg_class_id;

if counter = 1 then
delete from student_schedule
where
student_id = arg_student_id and
class_id = arg_class_id;
end if;

end;

_ *****
- Procedure assign_instructor
-
procedure assign_instructor (arg_class_ID IN varchar2,
arg_instructor_ID IN varchar2,
status OUT number) is

counter number;

begin

- Determine if this instructor is associated with the department
- that offers this class.

select count(*)
into counter
```



```
from Instructor I, Class C
where
C.Class_ID = arg_class_ID and
I.Instructor_ID = arg_instructor_ID and
C.Department_ID = I.Department_ID;

if counter = 0 then
status := unsuccessful;
else
-
- Assign this instructor to this class.
-
update class
set
Instructor_ID = arg_instructor_ID
where
Class_ID = arg_class_ID;

status := normal;

end if;

end;

_ *****
- Procedure assign_grade
-
procedure assign_grade (arg_student_ID IN varchar2,
arg_class_ID IN varchar2,
arg_grade IN varchar2,
status OUT number) is
counter number;

begin

- Determine if the student is registered for this class.

select count(*) into counter
from student_schedule
where
student_id = arg_student_id and
class_id = arg_class_id;

if counter = 0 then
-
- The student is not taking this class.
-
raise not_registered;
```

```
else
-
- Assign the grade for this class.
-
update student_schedule
set
grade = arg_grade
where
student_id = arg_student_id and
class_id = arg_class_id;
end if;

exception
when not_registered then
raise_application_error (-21003, 'Student not registered for
class');
when others then
null;

end;

_ *****
- Function student_GPA
-
function student_GPA (arg_student_ID IN varchar2)
return number is

GPA number;

begin

-
- Calculate the average grade point for this student based on all
- classes for which a grade has been assigned.

select avg(decode(grade, 'A+', 4.25, 'A', 4, 'A-', 3.75,
'B+', 3.25, 'B', 3, 'B-', 2.75,
'C+', 2.25, 'C', 2, 'C-', 1.75,
'D+', 1.25, 'D', 1, 'D-', 0.75, 'F', 0))
into GPA
from student_schedule
where
student_id = arg_student_id;

return GPA;

end;
```

end;

Wenn der Flugle-Pakettrumpf erstellt ist, können Sie die Funktionen aus dem Paket aufrufen. Werfen Sie einen Blick auf Listing 10.11. Die update-Anweisung wird verwendet, um den Lehrer für Class 104200 auf null zu setzen. Die nachfolgende select-Anweisung beweist, daß dieser Klasse kein Lehrer zugewiesen ist. Als nächstes wird ein anonymer Block verwendet, in dem Assign\_Instructor aufgerufen wird, um Instructor E491 der Klasse 104200 zuzuweisen. Der Status ist 0, dies zeigt an, daß die Prozedur erfolgreich war. Schließlich zeigt die letzte select-Anweisung an, daß der Lehrer in der Tat der Klasse zugewiesen wurde.

*Listing 10.11: Aufruf einer Prozedur in einem Paket aus einem anonymen PL/SQL-Block heraus.*

```
SQL> set serveroutput on
```

```
SQL>
```

```
SQL> update class
```

```
2 set
```

```
3 Instructor_ID = null
```

```
4 where
```

```
5 class_ID = '104200';
```

```
1 row updated.
```

```
SQL> select Instructor_ID
```

```
2 from Class
```

```
3 where
```

```
4 Class_ID = '104200';
```

```
INSTRUCTOR_ID
```

```
-----
```

```
SQL> declare
```

```
2
```

```
2 status number;
```

```
3
```

```
3 begin
```

```
4
```

```
4 dbms_output.enable;
```

```
5
```

```
5
```

```
5 Flugle.Assign_Instructor ('104200', 'E491', status);
```

```
6
```

```
6 dbms_output.put_line('Status: ' || to_char(status));
```

```
7
```

```
7 end;
```

```
8 /
```

```
Status: 0
```

PL/SQL procedure successfully completed.

```
SQL> select Instructor_ID
2 from Class
3 where
4 Class_ID = '104200';
```

```
INSTRUCTOR_ID
-----
E491
```

## Entwerfen eines Pakets für Datenbank-Trigger

Die Prozeduren und Funktionen eines Pakets können aus SQL\*Plus-Skripten, aus PL/SQL-Unterprogrammen, aus Skripten von Client-Anwendungen (wie Oracle Forms oder Power Builder) aufgerufen werden - sowie aus Datenbank-Trigger. Ein Datenbank-Trigger kann jedoch keine gespeicherte Prozedur, Funktion und kein Unterprogramm eines Pakets aufrufen, das eine commit-, rollback- oder savepoint-Anweisung enthält. Wenn Sie die Flexibilität benötigen, daß die Unterprogramme eines Pakets von einem Datenbank-Trigger aufgerufen werden können, stellen Sie deshalb sicher, daß Sie keine der Prozeduren und Funktionen im Paket Transaktionen festschreiben oder rückgängig machen. Lesen Sie Tag 11, »Weitere Programmieretechniken für PL/SQL«, um weitere Informationen über Datenbank-Trigger zu erhalten.

## Zusätzliche PL/SQL-Datentypen

An Tag 9, »Programmierung einer Oracle-Datenbank mit PL/SQL«, habe ich Ihnen einige Grundzüge der Programmierung von PL/SQL präsentiert. Wie Sie gesehen haben, unterstützt PL/SQL alle Datentypen, die in SQL verfügbar sind. PL/SQL bietet jedoch die folgenden zusätzlichen Datentypen, die in gewöhnlichen SQL-Anweisungen nicht zur Verfügung stehen:

- boolean
- binary\_integer, natural und positive
- %type
- %rowtype
- die PL/SQL-Tabelle (oder den Array)
- den benutzerdefinierten Datensatz

## Der Datentyp boolean

Einer der zusätzlichen Datentypen, die PL/SQL unterstützt, ist boolean. Listing 10.12 zeigt, wie Sie eine boolean-Variable deklarieren. Sie können eine boolean-Variable auch mit true oder false initialisieren.

***Listing 10.12: Deklaration einer Variablen vom Datentyp boolean und Initialisierung dieser Variablen.***

```
SQL> set serveroutput on
SQL>
SQL> declare
2 Payment_Is_Late boolean := TRUE;
3
3 begin
4
4 dbms_output.enable;
5
5 if Payment_Is_Late then
6 dbms_output.put_line('The payment is late!');
7 end if;
8
8 end;
9 /
The payment is late!
PL/SQL procedure successfully completed.
```



*In Zeile 2 wird die boolean-Variable `Payment_Is_Late` mit `true` initialisiert. In Zeile 5 wird `Payment_Is_Late` ausgewertet; weil Zeile 5 `true` ergibt, wird Zeile 6 ausgeführt.*

---

Bis Sie ihr einen Wert zuweisen, hat eine boolean-Variable den Wert Null. In Listing 10.13 wird der boolean-Variablen `Payment_Is_Late` der boolean-Ausdruck `Day_of_Month > 5` zugewiesen.

***Listing 10.13: Zuweisung eines Ausdrucks vom Datentyp boolean an eine Variable desselben Datentyps.***

```
SQL> set serveroutput on
SQL>
SQL> declare
2 Payment_Is_Late boolean;
3 Day_of_Month integer;
```

```
4
4 begin
5
5 dbms_output.enable;
6
6 select to_number(to_char(sysdate, 'DD'))
7 into Day_of_Month
8 from dual;
9
9 Payment_Is_Late := Day_of_Month > 3;
10
10 if Payment_Is_Late then
11 dbms_output.put_line('The payment is late!');
12 end if;
13
13 end;
14 /
The payment is late!
PL/SQL procedure successfully completed.
```

## Der Datentyp binary integer

Der Datentyp binary integer speichert vorzeichenbehaftete ganze Zahlen im Bereich von -2147483647 bis 2147483647. PL/SQL bietet auch zwei andere Datentypen, die Untertypen von binary\_integer sind.

- natural kann ganze Zahlen im Bereich von 0 bis 2147483647 speichern.
- positive kann ganze Zahlen im Bereich von 1 bis 2147483647 speichern.

Zahlen, die niemals einen Nachkommateil haben, wie etwa Schleifenzähler, können Sie mit den Datentypen natural oder positive deklarieren.

Wenn Sie reelle Zahl einer als binary\_integer, natural oder positive deklarierten Zahl zuweisen, wird die Zahl abgeschnitten. Listing 10.14 zeigt ein Beispiel dafür.

***Listing 10.14: Eine reelle Zahl wird abgeschnitten, wenn sie einer Variablen eines ganzzahligen PL/SQL-Datentyps zugewiesen wird.***

```
SQL> declare
2 Counter natural;
3
3 begin
4
```

```
4 dbms_output.enable;
5
5 Counter := 103.2;
6
6 dbms_output.put_line('Counter: ' || to_char(Counter, '999.999'));
7
7 end;
8 /
Counter: 103.000
```

## %type verwenden

PL/SQL bietet zwei Notationen für Verweise auf Tabellen und Datentypen von Spalten in Oracle.

- Verwenden Sie %type, um eine Variable mit demselben Datentyp wie eine bestimmte Spalte (oder einen vorher deklarierte Variable) zu deklarieren.
- Verwenden Sie %rowtype, um eine zusammengesetzte Variable zu deklarieren, deren Struktur diejenige einer bestimmten Tabelle oder eines bestimmten Cursor widerspiegelt.

Diese beiden Datentypen helfen Ihnen, PL/SQL-Code mit den im Data Dictionary von Oracle existierenden Tabellen- und Spaltendefinitionen zu integrieren.



*Um eine Variable mit demselben Datentyp wie eine Spalte zu deklarieren, verwenden Sie die Bezeichnung %type mit der folgenden Syntax:*

---

*variable-name table-name.column-name%TYPE*

Die Variablen sind folgendermaßen definiert:

- variable-name ist die deklarierte Variable
- table-name.column-name gibt die Spalte an, deren Datentyp für variable-name verwendet werden soll.

Das Schöne bei der Verwendung von %type ist, daß der Wartungsaufwand für PL/SQL-Code verringert wird. Sie können beispielsweise die Breite einer Spalte ändern, ohne daß Sie die Deklarationen der auf dieser Spalte basierenden Variablen ändern müssen.

## %rowtype verwenden

Sie können die Bezeichnung %rowtype verwenden, um eine Variable - in Wirklichkeit einen Datensatz - zu deklarieren, dessen Struktur identisch mit der Struktur einer bestimmten Tabelle ist.

---



*%rowtype wird mit der folgenden Syntax verwendet:*

---

*variable-name table-name%ROWTYPE*

Die Variablen sind folgendermaßen definiert:

- variable-name ist der Name der deklarierten Variable.
- table-name gibt die Tabelle an, der variable-name entspricht.

Beispielsweise wird ein Datensatz namens Instructor\_Rec als Instructor%rowtype deklariert. Das Ergebnis ist, daß die Felder von Instructor\_Rec dieselben Namen und Datentypen wie die Spalten der Tabelle Instructor haben.

### *Listing 10.15: Verwendung von %rowtype in einer select-Anweisung*

```
SQLWKS> declare
2>
3> Instructor_Rec Instructor%ROWTYPE;
4>
5> begin
6>
7> dbms_output.enable;
8>
9> select *
10> into Instructor_Rec
11> from Instructor
12> where
13> Instructor_ID = 'P331';
14>
15> dbms_output.put_line('Instructor ID: ' ||
Instructor_Rec.Instructor_ID);
```



```
16> dbms_output.put_line('Last Name: ' || Instructor_Rec.Last_Name);
17> dbms_output.put_line('First Name: ' ||
Instructor_Rec.First_Name);
18>
19> end;
20> /
Statement processed.
Instructor ID: P331
Last Name: POULSON
First Name: RANIER
```

Wie Sie sehen können (in Zeile 15 bis 17), werden die Felder des %rowtype-Datensatzes über `variable-name.field-name` angesprochen.

`variable-name` ist der Name der deklarierten %rowtype-Variable.

`field-name` ist der Name einer Spalte, die in der Deklaration von `variable-name` angegeben wird.



*Obwohl Sie in einer select-Anweisung auf einen Datensatz verweisen können, der mit %rowtype deklariert wurde, können Sie nicht mit einer insert-Anweisung auf den gesamten Datensatz verweisen. PL/SQL weist beispielsweise die folgende insert-Anweisung zurück:*

---

```
SQL> declare
2 Patient_Rec Patient%rowtype;
3
3 begin
4
4 Patient_Rec.Patient_ID := 'HHH111';
5 Patient_Rec.Body_Temp_Deg_F := 102.7;
6
6 insert into Patient
7 (Patient_ID, Body_Temp_Deg_F)
8 values
9 Patient_Rec;
10
end;
```

```
/
Patient_Rec;
*
ERROR at line 9:
ORA-06550: line 9, column 1:
PLS-00103: Encountered the symbol "PATIENT_REC"
when expecting one of the follow
an aggregate
Resuming parse at line 9, column 12.
```

---



*Statt dessen müssen Sie jede Komponente des Datensatzes Patient\_Rec angeben, die den in der insert-Anweisung angegebenen Spalten entspricht.*

---

Sie können auch eine Variable einer anderen Variablen zuweisen, wenn beide mit der Bezeichnung %rowtype für dieselbe Tabelle deklariert wurden. Listing 10.16 zeigt dieses Konzept, indem New\_Patient an ER\_Patient zugewiesen wird.

***Listing 10.16: Zuweisungen an PL/SQL-Variablen, die auf %rowtype basieren.***

```
SQL> declare
2
2 New_Patient Patient%ROWTYPE;
3 ER_Patient Patient%ROWTYPE;
4
4 begin
5
5 dbms_output.enable;
6
6 select *
7 into New_Patient
8 from Patient
9 where
10 Patient_ID = 'ZZ0123';
11
11 ER_Patient := New_Patient;
12
12 dbms_output.put_line('ER_Patient.Body_Temp_Deg_F: ' ||
13 to_char(ER_Patient.Body_Temp_Deg_F));
K 14
```

```
14 end;
15 /
ER_Patient.Body_Temp_Deg_F: 98.6
```

Sie können jedoch eine %rowtype-Variable nicht einer anderen %rowtype-Variablen zuweisen, wenn beide nicht auf dieselbe Tabelle einer Datenbank verweisen, auch wenn die Tabellen identisch sind.

*Listing 10.17: %rowtype-Variablen, die auf unterschiedlichen Tabellen basieren, können einander nicht zugewiesen werden.*

```
SQL> create table Identical_Patient as
2 select * from Patient;
Table created.
SQL> set serveroutput on
SQL>
SQL> declare
2
2 New_Patient Patient%ROWTYPE;
3 ER_Patient Identical_Patient%ROWTYPE;
4
4 begin
5
5 dbms_output.enable;
6
6 select *
7 into New_Patient
8 from Patient
9 where
10 Patient_ID = 'ZZ0123';
11
11 ER_Patient := New_Patient;
12
12 dbms_output.put_line('ER_Patient.Body_Temp_Deg_F: ' ||
13 to_char(ER_Patient.Body_Temp_Deg_F));
14
14 end;
15 /
declare
*
```

ERROR at line 1:  
ORA-06550: line 11, column 15:  
PLS-00382: expression is of wrong type  
ORA-06550: line 11, column 1:  
PL/SQL: Statement ignored

## Weitere komplexe Datentypen: table und record

PL/SQL unterstützt zwei zusätzliche zusammengesetzte Datentypen. *Tabellen* und *Datensätze*. Jedes dieser Objekte wird zuerst als Datentyp deklariert, dann wird die eigentliche PL/SQL-Tabelle oder der eigentliche PL/SQL-Datensatz auf der Basis des angegebenen Datentyps deklariert.

Sie können sich eine PL/SQL-Tabelle als Array vorstellen: Sie besteht aus einem einzigen Feld. Sie deklarieren auch keine obere Grenze für die Anzahl der Elemente, die eine PL/SQL-Tabelle enthalten kann; ihre Größe ist dynamisch.



*Eine PL/SQL-Tabelle ist eine Sammlung von Elementen desselben Typs, die durch eine Indexnummer geordnet ist.*

---



*Es ist ungünstig, daß sich Oracle entschlossen hat, den Namen Tabelle für eine Struktur zu verwenden, die angemessener als Array beschrieben werden kann. Eine PL/SQL-Tabelle besteht im Gegensatz zu einer Datenbanktabelle aus einer einzigen Spalte. Wie bei einem Array wird auf die Werte einer PL/SQL-Tabelle über einen Index zugegriffen. Denken Sie daran, daß eine PL/SQL-Tabelle und eine Datenbanktabelle zwei verschiedene Dinge mit sehr speziellen Eigenschaften und Verwendungszwecken sind.*

---

Ein benutzerdefinierter Datensatz bietet mehr Flexibilität als die Bezeichnung %rowtype. Sie sollten die Verwendung eines benutzerdefinierten Datensatzes in Betracht ziehen, wenn eine der folgenden Bedingungen erfüllt ist:

- Sie benötigen nur eine Teilmenge der Spalten einer Tabelle.
- Sie wollen auch verknüpfte oder *abgeleitete* Informationen im Datensatz speichern.

Der folgende Abschnitt beschäftigt sich mit der Verwendung von PL/SQL-Tabellen.

---



*Wenn Sie einen benutzerdefinierten Datensatz deklarieren, der mit einer Datenbanktabelle verknüpft ist, sollten Sie die Bezeichnung %type für jedes Feld verwenden, das eine Spalte der Datenbanktabelle spiegelt. Das reduziert den Aufwand für die Wartung des PL/SQL-Codes, wenn die unvermeidlichen Änderungen an der Datenbank vorgenommen werden.*

---

## Deklarieren von PL/SQL-Tabellen

---



*Der Typ für eine PL/SQL-Tabelle wird mit der folgenden Syntax vereinbart:*

---

```
TYPE type-name IS TABLE OF  
table-name.column-name%TYPE  
INDEX BY BINARY_INTEGER;
```

Die Variablen sind folgendermaßen definiert:

- *type-name* ist der Name des deklarierten Typs.
- *table-name.column-name* gibt die Spalte an, deren Datentyp die Basis für *type-name* ist.

Nachdem Sie den PL/SQL-Tabellentyp deklariert haben, können Sie Variablen auf der Basis dieses Typs deklarieren. Beispielsweise wird in Listing 10.18, das einen anonymen PL/SQL-Block enthält, `Class_ID_Tab` als Tabelle der Spalte `Class_ID` in der Tabelle `Class` deklariert. Ein Cursor `for loop` wählt jede `Class_ID` aus der Tabelle `Class` aus und weist sie einem Element von `Class_ID_Tab` zu. Das in Listing 10.18 gezeigte Beispiel wurde mit SQL Worksheet ausgeführt.

### ***Listing 10.18: Verwendung einer PL/SQL-Tabelle.***

```
SQLWKS> set serveroutput on
```

```
Server Output ON
```

```
SQLWKS> declare
```

```
2>
```

```
3> type Class_ID_Type is table of Class.Class_ID%TYPE
```

```
4> index by binary_integer;
```

```
5>
6> Class_ID_Tab Class_ID_Type;
7> i binary_integer := 0;
8> final_count binary_integer;
9>
10> begin
11>
12> dbms_output.enable;
13>
14> for Class_ID_Rec in (select Class_ID from Class) loop
15>
16> i := i + 1;
17> Class_ID_Tab(i) := Class_ID_Rec.Class_ID;
18>
19> end loop;
20>
21> final_count := i;
22>
23> for i in 1..final_count loop
24>
25> dbms_output.put_line('Class_ID_Tab(' || to_char(i) || ') = ' ||
26> Class_ID_Tab(i));
27> end loop;
28>
29> end;
30> /
Statement processed.
Class_ID_Tab(1) = 104200
Class_ID_Tab(2) = 104500
Class_ID_Tab(3) = 109100
Class_ID_Tab(4) = 120200
Class_ID_Tab(5) = 110300
Class_ID_Tab(6) = 108300
Class_ID_Tab(7) = 108400
Class_ID_Tab(8) = 108600
Class_ID_Tab(9) = 103400
Class_ID_Tab(10) = 103600
```

Sie können eine PL/SQL-Tabelle als Argument an eine Prozedur oder Funktion übergeben. Zusammen mit der PL/SQL-Tabelle werden Sie wahrscheinlich auch eine `binary_integer`-Variable übergeben, die die Anzahl der Elemente in der PL/SQL-Tabelle angibt. Listing 10.19 zeigt ein Beispiel für eine Prozedur, die eine PL/SQL-Tabelle zurückgibt, die diejenigen `Course_IDs` enthält, für welche die zusätzlichen Gebühren mehr als \$ 50 betragen.

***Listing 10.19: Rückgabe einer PL/SQL-Tabelle.***

```
SQLWKS> declare
2>
3> type Course_ID_Type is table of Course.Course_ID%TYPE
4> index by binary_integer;
5> Course_ID_Tab Course_ID_Type;
6>
7> i binary_integer := 0;
8> Total_Number binary_integer;
9>
10> procedure Get_Course_IDs (Num_Rows out binary_integer,
11> Course_ID_Table out Course_ID_Type) is
12>
13> i binary_integer := 0;
14>
15> begin
16>
17> for Course_ID_Rec in (select Course_ID from Course
18> where Additional_Fees > 50) loop
19>
20> i := i + 1;
21> Course_ID_Table(i) := Course_ID_Rec.Course_ID;
22>
23> end loop;
24>
25> Num_Rows := i;
26>
27> end Get_Course_IDs;
28>
29> - Main block.
30>
31> begin
32>
33> dbms_output.enable;
34>
35> Get_Course_IDs (Total_Number, Course_ID_Tab);
36>
37> for i in 1..Total_Number loop
38> exit when Course_ID_Tab(i) = NULL;
39> dbms_output.put_line('Course_ID_Tab(' || to_char(i) ||
40> ') = ' || Course_ID_Tab(i));
41> end loop;
42>
43> end;
44> /
```

Statement processed.

```
Course_ID_Tab(1) = 101  
Course_ID_Tab(2) = 189  
Course_ID_Tab(3) = 101  
Course_ID_Tab(4) = 178  
Course_ID_Tab(5) = 177  
Course_ID_Tab(6) = 174  
Course_ID_Tab(7) = 181  
Course_ID_Tab(8) = 501
```

---



*Die erste Zeile in Listing 10.19 beginnt mit dem Deklarationsteil eines anonymen Blocks. Der Typ `Course_ID_Type` wird in Zeile 3 vereinbart. Die Deklaration der Prozedur `Get_Course_IDs` beginnt in Zeile 10. In Zeile 13 wird der Index der PL/SQL-Tabelle `Course_ID_Table` auf 0 initialisiert. Der Ausführungsteil des anonymen Blocks beginnt in Zeile 31. In Zeile 35 wird im Ausführungsteil des anonymen Blocks die Prozedur `Get_Course_IDs` aufgerufen.*

---

---



*PL/SQL schränkt den Bereich des Index einer PL/SQL-Tabelle nicht ein; Sie könnten bei -100, 0 oder 1 anfangen, je nachdem, wie es gerade nötig ist.*

---

## **Deklarieren von benutzerdefinierten Datensätzen**

Der Prozeß der Verwendung benutzerdefinierter Datensätze ist sehr ähnlich wie bei einer PL/SQL-Tabelle: Sie definieren einen Datentyp für den Datensatz und deklarieren dann Variablen auf der Basis des neuen Typs.

---



*Hier folgt die Syntax für die Deklaration eines Datensatztyps:*

---



```
TYPE type-name IS RECORD
(field-name field-datatype [NOT NULL] [initial-value],
...
(field-name field-datatype [NOT NULL] [initial-value];
```

Die Variablen für die Deklaration eines Datensatztyps sind folgendermaßen definiert:

- *type-name* ist der Name des deklarierten Datensatztyps.
- *field-name* ist der Name des Felds, er unterliegt den Einschränkungen von PL/SQL bezüglich der Namensgebung von Variablen.
- *field-datatype* ist der Datentyp des Felds, dies kann ein bestimmter PL/SQL-Datentyp (wie *number* oder *boolean*) oder ein Verweis auf den Datentyp einer Spalte sein, der den Bezeichner *%type* verwendet.
- *initial-value* ist ein Startwert, der *field-name* zugewiesen werden muß, wenn dieser als *not null* **deklariert ist**.



*Ein Vorteil eines benutzerdefinierten Datensatzes ist es, daß Sie Felder für die Speicherung abgeleiteter Daten in einem Datensatz definieren können, der nicht in der zugehörigen Datenbanktabelle gespeichert wird.*

---

Listing 10.20 zeigt die Deklaration eines benutzerdefinierten Datensatztyps. In diesem Fall vereinbaren wir einen Datensatztyp namens *Patient\_Rec\_Type*, der sich aus drei Feldern zusammensetzt: *Patient\_ID*, *Body\_Temp* und *Bed\_Number*. Die ersten zwei Felder sind in der Tabelle *Patient* vorhanden; *Body\_Temp* hat jedoch in der Tabelle einen anderen Namen - *Body\_Temp\_Deg\_F*. Das dritte Feld, *Bed\_Number*, existiert in der Tabelle *Patient* nicht.

***Listing 10.20: Verwendung eines benutzerdefinierten Verbunddatentyps.***

```
SQL> declare
2
2 type Patient_Rec_Type is record
3 (Patient_ID Patient.Patient_ID%TYPE,
4 Body_Temp Patient.Body_Temp_Deg_F%TYPE,
5 Bed_Number varchar2(4));
6
6 Patient_Rec Patient_Rec_Type;
7
7 begin
8
```

```
8 dbms_output.enable;
9
9 Patient_Rec.Patient_ID := 'ZZ0123';
10 Patient_Rec.Body_Temp := 98.6;
11 Patient_Rec.Bed_Number := 'A123';
12
12 dbms_output.put_line('Patient ID: ' || Patient_Rec.Patient_ID);
13 dbms_output.put_line('Body_Temp: ' ||
to_char(Patient_Rec.Body_Temp));
14 dbms_output.put_line('Bed Number: ' || Patient_Rec.Bed_Number);
15
15 insert into Patient
16 (Patient_ID, Body_Temp_Deg_F)
17 values
18 (Patient_Rec.Patient_ID, Patient_Rec.Body_Temp);
19
19 end;
20 /
```

```
Patient ID: ZZ0123
Body_Temp: 98.6
Bed Number: A123
```

## Festlegen von Standardwerten für Variablen

Als Standard werden beim Eintritt in eine Prozedur, eine Funktion oder einen anonymen Block alle Variablen mit null initialisiert. Sie können eine Variable im Deklarationsteil von PL/SQL auf zwei Weisen initialisieren:

```
variable-name data-type := initial-value;
```

oder

```
variable-name data-type DEFAULT initial-value;
```

Hier ist ein anonymes Block, der beide Methoden der Initialisierung einer PL/SQL-Variablen zeigt:

```
SQL> declare
2
2 i natural := 33;
3 mystring varchar2(30) default 'JACKSON';
4
4 begin
```

```
5
5 dbms_output.enable
6
6 end;
7 /
PL/SQL procedure sucessfully completed.
```

Ein Grund, einen Standardwert für eine Variable anzugeben, ist, daß der Code oft besser zu verstehen und zu warten ist. Indem Sie Standardwerte angeben, machen Sie weniger Annahmen darüber, wie der Code sich verhalten wird.

## Zusammenfassung

Diese Lektion hat sich auf die folgenden Konzepte konzentriert:

- Sie können die Oracle-DDL-Anweisungen `create procedure` und `create function` verwenden, um gespeicherte Prozeduren bzw. Funktionen zu erstellen.
- Eine gespeicherte Prozedur oder Funktion besteht aus Deklarationen und ausführbaren Anweisungen in PL/SQL, die übersetzt und in der Datenbank gespeichert werden.
- Sie können für eine gespeicherte Prozedur oder Funktion Argumente definieren, die nur übergeben werden, die nur zurückgegeben werden, oder die übergeben und zurückgegeben werden.
- Oracle stellt ein vordefiniertes Paket, `dbms_output`, zur Verfügung, das Routinen für das Anzeigen von Informationen aus einem PL/SQL-Unterprogramm heraus enthält.
- Eine gespeicherte Funktion kann in einer Oracle-SQL-Anweisung aufgerufen werden. Der Datentyp der übergebenen Werte muß dem zugehörigen Argument entsprechen.
- Ein Paket ist eine Gruppe verwandter Prozeduren und Funktionen. Ein Paket hat eine Spezifikation und einen Rumpf.
- Eine Paketspezifikation wird mit der Anweisung `create package` erzeugt. Ein Pakettrumpf wird mit Hilfe der Anweisung `create package body` erzeugt.
- PL/SQL unterstützt mehrere Datentypen, die in SQL-Anweisungen von Oracle nicht zur Verfügung stehen: `boolean`, `binary_integer`, `natural` und `positive`.
- Eine Variable vom Typ `boolean` kann die Werte `true`, `false` oder `null` haben.
- Indem Sie `%type` verwenden, können Sie Variablen deklarieren, die denselben Datentyp wie eine bestimmte Spalte einer Tabelle haben.
- Verwenden Sie den Bezeichner `%rowtype`, um einen Datensatz zu deklarieren, dessen Struktur einer bestimmten Tabelle entspricht. Sie können `%rowtype` auch verwenden, um einen Datensatz zu deklarieren, der dieselbe Struktur wie ein Cursor hat.
- Eine PL/SQL-Tabelle ist ein benutzerdefinierter Datentyp, den Sie sich als unbegrenztes Array aus einem Feld vorstellen können. Um eine PL/SQL-Tabelle zu deklarieren, müssen Sie zuerst einen Datentyp für die Tabelle deklarieren.
- Ein benutzerdefinierter Datensatz besteht aus einem oder mehreren Feldern. Der Datentyp jedes Felds kann unter Verwendung eines Datentyps von Oracle SQL oder PL/SQL explizit deklariert werden, oder es wird der Bezeichner `%type` verwendet, um auf den Datentyp einer

Datenbankspalte zu verweisen. Ein Feld kann als not null deklariert und bei seiner Deklaration initialisiert werden.

## Wie geht es weiter?

An Tag 11 werden Sie etwas über einige bedeutende Aspekte von PL/SQL lernen - wie man Fehler behandelt, wie man mehrere Zeilen mit einem Cursor abfragt, und wie man einen Trigger für eine Tabelle schreibt.

## Fragen und Antworten

Frage:

Ist es besser, anstatt einzelner Prozeduren und Funktionen ein Paket mit Prozeduren und Funktionen zu erstellen?

Antwort:

*Aus der Perspektive der Software-Entwicklung ist es besser, ein Paket zu erstellen. Indem Sie eine Paketspezifikation erstellen, wird die Schnittstelle - die Namen aller Prozeduren und Funktionen und ihrer Argumente - vom PL/SQL-Code getrennt, der die Schnittstelle implementiert.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Richtig oder falsch? Eine gespeicherte Prozedur kann eine gespeicherte Funktion aufrufen, eine gespeicherte Funktion jedoch keine gespeicherte Prozedur.
2. Nennen Sie drei Gründe für das Speichern von Prozeduren, Funktionen und Paketen in einer Anwendung.

3. Welche Werkzeuge können verwendet werden, um PL/SQL-Unterprogramme zu erstellen?
4. Wenn »x > 32« einer PL/SQL-Variablen zugewiesen wird, wie ist dann der Datentyp der Variablen?

## Übungen

1. Erzeugen Sie einen anonymen PL/SQL-Block, der die Prozedur Assign\_Grade aus dem Flugle-Paket aufruft und Anna Anastasia für Biologie 101 ein B zuweist.
2. Ein Student Jackson Smythe hat eine gespeicherte Funktion namens Change\_My\_Grade mit zwei Argumenten erstellt: Student ID und Class ID. Die Funktion ändert die Note für jeden beliebigen Studenten und eine beliebige Klasse auf A+. Schreiben Sie diese Funktion.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

# 11

# Weitere Programmiertechniken mit PL/SQL

[Fehlerbehandlung unter PL/SQL](#)

[Der Exception-Teil](#)

[Vordefinierte Exceptions](#)

[Deklarieren einer Exception](#)

[Erfolg oder Fehlschlag: Auswerten von sqlcode und sqlerrm](#)

[Rückgabe von Fehlern mit Hilfe der Prozedur raise\\_application\\_error](#)

[Abrufen von Daten mit Hilfe eines Cursor](#)

[Deklarieren eines Cursor](#)

[Mit Cursor-gesteuerten for-Schleifen arbeiten](#)

[%found oder %notfound](#)

[Ermitteln der Zeilenanzahl mit Hilfe von %rowcount](#)

[Erzwingen von Firmenregeln mit Hilfe von Datenbank-Triggern](#)

[Erstellen eines Trigger](#)

[Spaltenwerte mit Hilfe eines Trigger auf Gültigkeit prüfen](#)

[Sicherheit mit Hilfe eines Trigger erzwingen](#)

[Setzen von Spaltenwerten mit Hilfe eines Trigger](#)

[commit- und rollback-Anweisungen in Triggern nicht erlaubt](#)

[Aufruf von gespeicherten Prozeduren in einem Trigger](#)

[Löschen, Aktivieren und Deaktivieren von Triggern](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

Die heutige Lektion umfaßt die abschließende Diskussion von SQL und PL/SQL, ehe Sie sich an die Verwendung von Developer/2000 und Power-Objects zur Anwendungserstellung wagen. Die heutige Lektion läßt sich in drei Kategorien unterteilen:

- Die Behandlung von Fehlern in einem PL/SQL-Unterprogramm. Sie werden lernen, eine Fehlerbehandlungsroutine für Oracle-Fehler und benutzerdefinierte Exceptions zu definieren. Außerdem werden Sie einige Beispiele für vordefinierte Exceptions sehen.
- Die Deklaration und Verwendung von Cursors in einem PL/SQL-Unterprogramm, um mehrere Zeilen aus der Datenbank abzurufen.
- Die Verwendung von PL/SQL zur Erstellung eines Datenbank-Triggers. Sie werden den Einsatz eines Triggers zur Validierung von Daten, der Durchsetzung von Sicherheitsmaßnahmen und für Änderungen an Datensätzen einer Tabelle erlernen.

## Fehlerbehandlung unter PL/SQL

Das Oracle-Handbuch *Error Messages and Codes* führt alle Fehlercodes und -meldungen außer denen für betriebssystemspezifische Fehler auf. Zu irgendeinem Zeitpunkt wird Ihre Anwendung wahrscheinlich auf einige dieser Fehler treffen. Unter PL/SQL werden Oracle-Fehler als *Exceptions* bezeichnet. Einige dieser Exceptions besitzen vordefinierte Namen, auf die in PL/SQL-Unterprogrammen Bezug genommen werden

kann. Zusätzlich zu diesen vordefinierten Oracle-Exceptions können Sie anwendungsspezifische Exceptions in einem PL/SQL-Unterprogramm definieren.

---



*Eine Exception ist ein vordefinierter oder benutzerdefinierter Anwendungsfehler, der automatisch vom Oracle-RDBMS oder absichtlich in einem PL/SQL-Unterprogramm ausgelöst wird.*

---

Eine Methode zur Fehlerbehandlung in einem PL/SQL-Unterprogramm ist die Validierung auf einen Oracle-Fehlercode nach jeder SQL-Anweisung. Es ist bei diesem Ansatz jedoch problematisch, das entsprechende Unterprogramm nachzuvollziehen. Eine Alternative ist die Möglichkeit, daß Sie unter PL/SQL genau angeben, welche Verarbeitung bei einer bestimmten Exception stattfinden soll. Dieser Abschnitt eines PL/SQL-Unterprogramms wird der *Exception-Teil* eines PL/SQL-Unterprogramms genannt. Eine vordefinierte Exception wird sozusagen »ausgelöst«, wenn während der Ausführung eines PL/SQL-Unterprogramms ein Oracle-Fehler auftritt. Eine benutzerdefinierte Exception lösen Sie aus, in dem Sie in Ihrem PL/SQL-Code die RAISE-Anweisung an angebrachter Stelle aufrufen.

Nehmen wir z.B. an, Sie haben PL/SQL-Unterprogramm geschrieben, das Informationen aus einer *unstrukturierten Datei* in die Tabelle einer Datenbank lädt. Wenn diese eine ungültige Zahl enthält - 3.1A anstatt 3.14 - wird eine vordefinierte Exception ausgelöst, wenn das PL/SQL-Unterprogramm versucht, eine Zeile mit der ungültigen Zahl einzufügen.

## Der Exception-Teil

Der Exception-Teil ist ein optionaler Abschnitt eines PL/SQL-Unterprogramms, der PL/SQL anweist, wie bestimmte Exceptions behandelt werden sollen.

---



*Die Syntax eines Exception-Teils lautet folgendermaßen:*

---

```
EXCEPTION
WHEN exception-name1 THEN
PL/SQL-statements;
...
WHEN exception-nameN THEN
PL/SQL-statements;
```



```
...
{WHEN OTHERS THEN
PL/SQL-statements;}
END;
```

Die Variablen sind wie folgt definiert:

- *exception-name1* bis *exception-nameN* sind die Namen vor- und benutzerdefinierter Exceptions.
- *PL/SQL-statements* sind eine oder mehrere PL/SQL-Anweisungen, die beim Auslösen der Exception ausgeführt werden.

Zur Veranschaulichung enthält Listing 11.1 einen PL/SQL-Block mit einem Exception-Teil. Beachten Sie, daß der Exception-Teil zwei Exception-Handler enthält: einen für eine vordefinierte Exception - die Exception `TOO_MANY_ROWS` (in Zeile 11) - und einen für alle anderen Exceptions - gekennzeichnet durch das Wort `OTHERS` (in Zeile 13).

***Listing 11.1: Behandlung der too\_many\_rows-Exception.***

```
SQL> declare
2
2 Course_Rec Course%ROWTYPE;
3
3 begin
4
4 dbms_output.enable;
5
5 select *
6 into Course_Rec
7 from Course
8 where
9 Department_ID = 'BIO';
10
10 exception
11 when TOO_MANY_ROWS then
12 dbms_output.put_line('TOO_MANY_ROWS raised - use a cursor');
13 when OTHERS then
14 NULL;
15 end;
16 /
TOO_MANY_ROWS raised - use a cursor
```

Wenn Sie den Exception-Handler für `OTHERS` entfernen und eine Exception auslösen, für die kein Exception-Handler vorhanden ist, gibt PL/SQL eine Fehlermeldung zurück. Listing 11.2 zeigt ein Beispiel - eine Zeichenfolge mit 18 Zeichen wird einer Variablen zugewiesen, die bis zu fünf Zeichen speichern kann, was zu einem Oracle-Fehler führt.

***Listing 11.2: Der Exception-Teil behandelt keine others-Exceptions.***

```
SQL> declare
```

```
2
2 xyz varchar2(5);
3
3 begin
4
4 dbms_output.enable;
5
5 xyz := 'This will not fit!';
6
6 exception
7
7 when TOO_MANY_ROWS then
8 dbms_output.put_line('TOO_MANY_ROWS Exception Raised');
9 dbms_output.put_line('Occurred in anonymous block');
10
10 end;
11 /
declare
*
```

ERROR at line 1:  
ORA-06502: PL/SQL: numeric or value error  
ORA-06512: at line 5

## Vordefinierte Exceptions

Alle Exceptions können entweder als vordefinierte oder benutzerdefinierte Exception eingestuft werden. Vordefinierte Exceptions werden automatisch ausgelöst; z.B. resultiert der Bezug einer SQL-Anweisung auf eine nicht vorhandene Tabelle in einem Oracle-Fehler. Als Beispiel nehmen Sie an, Ihr PL/SQL-Unterprogramm enthalte eine SELECT-Anweisung, die unter bestimmten Umständen keine Zeilen zurückgibt, wodurch die Exception NO\_DATA\_FOUND ausgelöst wird. Vordefinierte Exceptions besitzen bezeichnende Namen. Es folgen einige vordefinierte Exceptions, die Ihnen bei der Entwicklung einer Oracle-Anwendung begegnen können:

- DUP\_VAL\_ON\_INDEX
- INVALID\_NUMBER
- NO\_DATA\_FOUND
- TOO\_MANY\_ROWS
- VALUE\_ERROR

Im folgenden Abschnitt werden die Bedingungen betrachtet, die jede dieser vordefinierte Exceptions auslösen.

### Die DUP\_VAL\_ON\_INDEX-Exception

Die DUP\_VAL\_ON\_INDEX-Exception wird ausgelöst, wenn eine SQL-Anweisung versucht, einen gleichen Wert in einer Spalte zu erzeugen, für die ein eindeutiger Index besteht. Zur Veranschaulichung enthält Listing 11.3 einen anonymen SQL-Block, der versucht, die Course-Tabelle in der Form zu aktualisieren, daß alle Spalten den gleichen Wert für Course\_ID haben und somit die Exception DUP\_VAL\_ON\_INDEX auslöst.

***Listing 11.3: Behandlung der dup\_val\_on\_index-Exception.***

```
SQL> declare
2
2 begin
3
3 dbms_output.enable;
4
4 update Course
5 set
6 Department_ID = 'BIO',
7 Course_ID = '101';
8
8 exception
9 when DUP_VAL_ON_INDEX then
10 dbms_output.put_line('DUP_VAL_ON_INDEX exception raised');
11 end;
12 /
DUP_VAL_ON_INDEX exception raised
PL/SQL procedure successfully completed.
```

## Die INVALID\_NUMBER-Exception

Die INVALID\_NUMBER-Exception wird ausgelöst, wenn eine SQL-Anweisung eine ungültige Zahl angibt. So zeigt Listing 11.4 z.B. einen anonymen SQL-Block, der versucht, die Spalte Additional-Fees der Course-Tabelle zu aktualisieren. Die Exception wird ausgelöst, weil die Funktion To\_Number versucht, die Zeichenfolge-variable Bogus\_Value in eine Zahl umzuwandeln.

***Listing 11.4: Behandlung der invalid\_number-Exception.***

```
SQL> declare
2
2 Bogus_Value varchar2(30) := 'NOT A NUMBER';
3
3 begin
4
4 dbms_output.enable;
5
5 update Course
6 set
7 Additional_Fees = to_number(Bogus_Value);
8
```

```
8 exception
9 when INVALID_NUMBER then
10 dbms_output.put_line('INVALID_NUMBER exception raised');
11 end;
12 /
INVALID_NUMBER exception raised
PL/SQL procedure successfully completed.
```

## Die no\_data\_found-Exception

Die no\_data\_found-Exception wird ausgelöst, wenn eine SELECT-Anweisung keine Zeilen zurückgibt, wie es in Listing 11.5 gezeigt wird.

*Listing 11.5: Die no\_data\_found-Exception wird ohne Exception-Handler ausgelöst.*

```
SQL> declare
2
2 Course_Rec Course%ROWTYPE;
3
3 begin
4
4 dbms_output.enable;
5
5 select *
6 into Course_Rec
7 from Course
8 where
9 Course_ID = '777';
10
10 end;
11 /
declare
*
```

ERROR at line 1:  
ORA-01403: no data found  
ORA-06512: at line 5

Nachdem Sie für NO\_DATA\_FOUND (in Zeile 11) einen Exception-Handler hinzugefügt haben, gibt PL/SQL nicht länger den Fehler - no data returned - an die aufrufende Umgebung zurück, wie es in Listing 11.6 gezeigt wird.

*Listing 11.6: Behandlung der no\_data\_found-Exception.*

```
SQL> declare
2
2 Course_Rec Course%ROWTYPE;
3
```

```
3 begin
4
4 dbms_output.enable;
5
5 select *
6 into Course_Rec
7 from Course
8 where
9 Course_ID = '777';
10
10 exception
11 when NO_DATA_FOUND then
12 dbms_output.put_line('No data returned');
13 when OTHERS then
14 NULL;
15 end;
16 /
No data returned
```

PL/SQL procedure successfully completed.

## Die too\_many\_rows-Exception

In der PL/SQL-Umgebung kann eine SELECT-Anweisung nicht mehr als eine Zeile abrufen, ohne die too\_many\_rows-Exception auszulösen. Um von einer Abfrage eine beliebige Anzahl Zeilen abzurufen, können Sie einen Cursor einsetzen, den Sie sich wie ein Fenster auf das Abfrageergebnis vorstellen können. Listing 11.7 zeigt an einem Beispiel, wie ein Exception-Handler für die too\_many\_rows-Exception benutzt wird.

*Listing 11.7: Behandlung der too\_many\_rows-Exception.*

```
SQL> declare
2
2 Course_Rec Course%ROWTYPE;
3
3 begin
4
4 dbms_output.enable;
5
5 select *
6 into Course_Rec
7 from Course
8 where
9 Department_ID = 'BIO';
10
10 exception
11 when TOO_MANY_ROWS then
```

```
12 dbms_output.put_line('TOO_MANY_ROWS raised - use a cursor');
13 when OTHERS then
14 NULL;
15 end;
16 /
TOO_MANY_ROWS raised - use a cursor
```

PL/SQL procedure successfully completed.

## Die value\_error-Exception

Die value\_error-Exception wird bei einer Reihe von Situationen ausgelöst, die in Beziehung zu Fehlern beim Abschneiden und Konvertieren stehen. In Listing 11.8 wird z.B. ein PL/SQL-Block (in der ersten Zeile beginnend) aufgezeigt, der versucht, die Zeichenfolge More than 5 characters einer Variablen zuzuweisen, die als VARCHAR2(5) deklariert wurde.

*Listing 11.8: Behandlung der value\_error-Exception.*

```
SQL> declare
2
2 xyz varchar2(5);
3
3 begin
4
4 dbms_output.enable;
5
5 xyz := 'More than 5 characters';
6
6 exception
7
7 when VALUE_ERROR then
8 dbms_output.put_line('VALUE_ERROR raised');
9
9 when OTHERS then
10 NULL;
11
11 end;
12 /
VALUE_ERROR raised
```

## Deklarieren einer Exception

Zusätzlich zur Verarbeitung vordefinierter Exceptions können Sie auch anwendungsspezifische Exceptions definieren und in der folgenden Form deklarieren:

*exception-name* EXCEPTION;

Die Variable *exception-name* ist die deklarierte Exception und unterliegt den Einschränkungen bei der Namensvergabe für Objekte unter PL/SQL.

Listing 11.9 zeigt an einem Beispiel, wie eine Exception namens *Life\_Threatening\_Fever* deklariert wird, die aufgerufen wird, wenn die Körpertemperatur eines Patienten über 106° Fahrenheit (41°C) steigt.

*Listing 11.9: Deklaration einer anwendungsspezifischen Exception.*

```
SQL> declare
2
2 Life_Threatening_Fever exception;
3 Patient_ID Patient.Patient_ID%TYPE;
4
4 begin
5
5 dbms_output.enable;
6
6 for Patient_Rec in
7 (select Patient_ID, Body_Temp_Deg_F from Patient) loop
8
8 if Patient_Rec.Body_Temp_Deg_F > 106.0 then
9
9 Patient_ID := Patient_Rec.Patient_ID;
10 raise Life_Threatening_Fever;
11
11 end if;
12 end loop;
13
13 exception
14
14 when Life_Threatening_Fever then
15 dbms_output.put_line(Patient_ID || ' has a life ' ||
16 'threatening fever!');
17
17 end;
18 /
GG9999 has a life threatening fever!
```

## Erfolg oder Fehlschlag: Auswerten von sqlcode und sqlerrm

sqlcode ist ein vordefiniertes Symbol, das den Oracle-Fehlerstatus der zuvor ausgeführten PL/SQL-Anweisung enthält. Wird eine SQL-Anweisung ohne Fehler ausgeführt ist sqlcode gleich 0.

sqlerrm ist ein PL/SQL-Symbol, das die mit sqlcode verbundene Fehlermeldung enthält. Wird eine SQL-

Anweisung erfolgreich ausgeführt, ist sqlcode gleich 0 und sqlerrm enthält die Zeichenfolge Oracle-0000: normal, successful completion, wie sie in Listing 11.10 gezeigt wird.

***Listing 11.10: Zugriff auf sqlcode und sqlerrm.***

```
SQL> declare
2
2 begin
3
3 dbms_output.enable;
4
4 dbms_output.put_line('SQLCODE: ' || to_char(SQLCODE));
5 dbms_output.put_line('SQLERRM: ' || SQLERRM);
6
6 end;
7 /
SQLCODE: 0
SQLERRM: ORA-0000: normal, successful completion
```

Tritt tatsächlich ein Fehler auf, enthalten sqlcode und sqlerrm den jeweils zutreffenden Code und die entsprechende Meldung - wie es in der Ausgabe von Listing 11.11 aufgeführt wird.

***Listing 11.11: Zugriff auf sqlcode und sqlerrm im Exception-Teil.***

```
SQL> declare
2 Class_Rec Class%ROWTYPE;
3
3 begin
4
4 dbms_output.enable;
5
5 select *
6 into Class_Rec
7 from Class;
8
8 exception
9 when OTHERS then
10 dbms_output.put_line('SQLCODE: ' || to_char(SQLCODE));
11 dbms_output.put_line(SQLERRM);
12 end;
13 /
SQLCODE: -1422
ORA-01422: exact fetch returns more than requested number of rows

PL/SQL procedure successfully completed.
```

## Rückgabe von Fehlern mit Hilfe der Prozedur raise\_application\_error



Im DBMS-STANDARD-Packet stellt Oracle eine Prozedur `raise_application_error` zur Verfügung. Diese Prozedur können Sie dazu einsetzen, anwendungsspezifische Fehlermeldungen an Aufrufende wie SQL\*Plus, ein SQL-Unterprogramm oder eine Client-Anwendung zurückzugeben. Oracle reserviert Fehlercodes im Bereich -20000 bis -20999 für diese benutzerdefinierten Fehler. Veranschaulicht wird dies in Listing 11.12 durch einen PL/SQL-Block, der eine Exception `Fever_Out_Of_Range` deklariert. Ein Cursor For Loop (in Zeile 6) liest jede Zeile der Patiententabelle. Übersteigt die Temperatur eines Patienten 115° Fahrenheit, wird die Exception `Fever_Out_Of_Range` (in Zeile 8) ausgelöst. Im Exception-Teil ruft der Exception-Handler für `Fever_Out_Of_Range` `RAISE_APPLICATION_ERROR` auf und übergibt ihr den Fehlercode -20000 und eine entsprechende Meldung (in Zeile 14).

***Listing 11.12: Auslösen einer anwendungsspezifischen Exception.***

```
SQL> declare
2
2 Fever_Out_of_Range exception;
3 Patient_ID Patient.Patient_ID%TYPE;
4
4 begin
5
5 dbms_output.enable;
6
6 for Patient_Rec in
7 (select Patient_ID, Body_Temp_Deg_F from Patient) loop
8
8 if Patient_Rec.Body_Temp_Deg_F > 115.0 then
9
9 raise Fever_Out_of_Range;
10
10 end if;
11
11 end loop;
12
12 exception
13
13 when Fever_Out_of_Range then
14 raise_application_error (-20000,
'Fever is out of the range 65 Deg. F to 115 Deg. F');
15
15 end;
16 /
declare
*
```

ERROR at line 1:  
ORA-20000: Fever is out of the range 65 Deg. F to 115 Deg. F  
ORA-06512: at line 14

## Abrufen von Daten mit Hilfe eines Cursor

Wenn Sie SQL\*Plus oder SQL Worksheet einsetzen, können Sie eine Abfrage stellen, ohne sich Gedanken über die Anzahl der zurückgegebenen Zeilen machen zu müssen. Es spielt keine Rolle, ob die Abfrage keine, eine oder tausend Zeilen zurückgibt. Jedoch stimmt dies so nicht für PL/SQL-Unterprogramme. Sie können keine gewöhnliche SELECT-Anweisung verwenden, um mehr als eine Zeile abzurufen. Wenn eine SELECT-Anweisung in einem PL/SQL-Unterprogramm - ob als anonymer Block, gespeicherte Prozedur oder Trigger - mehr als eine Zeile abrufen, gibt Oracle eine Fehlermeldung zurück. Offensichtlich ist die Fähigkeit, mehr als eine Zeile abzurufen, so wesentlich, daß es dafür einen Mechanismus geben muß. Die von Oracle zur Erledigung dieser Aufgabe zur Verfügung gestellte Ressource ist der Cursor. Cursors werden von Oracle-Dienstprogrammen wie SQL\*Plus automatisch erstellt und verwendet.



*Ein Cursor stellt einen Mechanismus dar, mit dem programmgesteuert eine beliebige Anzahl Zeilen mit einer SELECT-Anweisung abgerufen werden kann.*

Listing 11.13 zeigt, wie Oracle einen Fehler zurückgibt, weil die SELECT-Anweisung mehr als einen Instructor zurückgibt.

***Listing 11.13: Oracle-Fehler, der aus einer select-Anweisung resultiert, die mehr als eine Zeile zurückgibt.***

```
SQL> declare
2
2 Instructor_ID Instructor.Instructor_ID%type
3 Last_Name Instructor.Last_Name%type
4 First_Name Instructor.First_Name%type
5
5 begin
6
6 select Instructor_ID, Last_Name, First_Name
7 into Instructor_ID, Last_Name, First_Name
8 from Instructor
9 order by Instructor_ID
10
10 end
11 /
declare
*
```

ERROR at line 1:  
ORA-01422: exact fetch returns more than requested number of rows  
ORA-06512: at line 6

Sie können sich einen Cursor als ein Fenster auf die Ergebnismenge einer Abfrage vorstellen (siehe Abbildung 11.1). Im allgemeinen führen Sie beim Einsatz eines Cursor vier Schritte durch:

1. **Deklarieren des Cursor.** Dem Cursor wird ein Name zugewiesen, und er wird einer zu parsenden SELECT-Anweisung zugeordnet.
2. **Öffnen des Cursor.** Das Oracle-RDBMS führt die dem Cursor zugeordnete Abfrage aus und bestimmt die angegebenen Zeilen (die aktive Menge).
3. **Lesen der Zeilen des Cursor.** Die Werte einer jeden Zeile werden der Umgebung des PL/SQL-Unterprogramms übergeben. Die Zeilen werden einzeln übergeben.
4. **Schließen des Cursor.** Alle von Oracle in Verbindung mit dem Cursor beanspruchten Ressourcen werden wieder freigegeben.

| Active Set    |            |            |
|---------------|------------|------------|
| Instructor ID | Last Name  | First Name |
| A612          | NILAND     | MARTINA    |
| B331          | BILLINGS   | BENJAMIN   |
| B391          | BATES      | JOSEPH     |
| D201          | DANIELS    | LAURA      |
| D944          | BESTON     | RAPHAEL    |
| E301          | EDWARDS    | SAMANTHA   |
| E405          | CHANG      | ROGER      |
| E481          | RICHARDSON | NANCY      |
| G331          | TORRES     | PETER      |
| J505          | JASON      | JERROLD    |

Abbildung 11.1:  
Veranschaulichung  
eines Cursor.



Unter PL/SQL löst eine SELECT-Anweisung, die mehr als eine Zeile zurückgibt, die vordefinierte Exception `TOO_MANY_ROWS` aus.

Es folgt eine gespeicherte Funktion, die feststellt, ob ein bestimmter Kurs in den Unterrichtsplan des Studenten paßt. Die Funktion - `Schedule_Conflict` - hat zwei Argumente: `Student_ID` und `Clas-ID`. Sie stellt Konflikte fest, indem nach anderen Kursen im gegenwärtigen Unterrichtsplan des Studenten gesucht wird, die die gleichen Zeiten wie der vorgeschlagene Kurs haben. Mit einem Cursor wird diese Abfrage durchgeführt. Listing 11.14 enthält den kompletten Text der Funktion.

***Listing 11.14: Verwendung eines Cursor in einer gespeicherten Funktion.***

```
SQL> create or replace function schedule_conflict [ccc](arg_student_ID IN
varchar2,
2 arg_class_ID IN varchar2)
3 return number is
4
4 conflicting_classes number := -1;
5 normal number := 0;
6
6 cursor get_other_classes is
7 select SS.Class_ID
8 from Student_Schedule SS, Class C
9 where
10 SS.Class_ID = C.Class_ID and
11 (C.Semester, C.School_Year, C.Schedule_ID) =
12 (select Semester, School_Year, Schedule_ID
13 from Class
14 where
15 Class_ID = arg_class_ID);
16
16 Conflicting_Class_ID Class.Class_ID%type;
17 status number;
18
18 begin
19
19 - Need to look at the other classes in the student's schedule
20 - for the same semester and school year.
21
21 open get_other_classes;
22
22 loop
23
23 fetch get_other_classes into Conflicting_Class_ID;
24 exit when get_other_classes%notfound;
25
25 end loop;
26
26 if get_other_classes%rowcount > 0 then
27 status := conflicting_classes;
28 else
29 status := normal;
30 end if;
31
31 close get_other_classes;
32
32 return status;
33
33 end;
```

34 /

Function created.

```
SQL> select Student_ID, Class_ID
2 from Student_Schedule
3 where
4 Student_ID = '10231311';
```

```
STUDENT_ID CLASS_ID
-----
104200
104500
```

```
SQL> select schedule_conflict('10231311','104200') from dual;
```

```
SCHEDULE_CONFLICT('10231311','104200')
-----
-1
```



*Es folgt eine schrittweise Betrachtung der einzelnen Schritte dieses Cursor. Als erstes wird in Zeile 6 der Cursor - Set\_Other\_Classes - als Verbindung zwischen zwei Tabellen - Student\_Schedule und Class - deklariert. Zweitens wird der Cursor im ausführbaren Teil der gespeicherten Funktion (in Zeile 21) geöffnet. Drittens liest eine Schleifenanweisung Zeilen vom Cursor, bis keine Zeilen mehr empfangen werden (in Zeile 23).*

---

Wie Sie am Ende des Listing sehen, ergibt eine Abfrage der Tabelle Student\_-Schedule für Student\_ID 10231311, daß dieser für zwei Kurse eingeschrieben ist: 104200 und 104500. Die Funktion wird durch die abschließende SELECT-Anweisung aufgerufen; sie überprüft, ob der Kurs 104200 für den Studenten 10231311 zu einer Überschneidung führt. Dies ist natürlich der Fall, denn der Student ist bereits für diesen Kurs eingeschrieben. Die Funktion gibt -1 zurück, was den Zustand sich überschneidender Kurse anzeigt.

## Deklarieren eines Cursor

Jeder Cursor muß deklariert werden, ehe er genutzt werden kann. Einen Cursor zu deklarieren, bedeutet ihn zu benennen und die SELECT-Anweisung zu bestimmen, der der Cursor zugeordnet ist.

---



Die grundlegende Syntax zur Deklaration eines Cursor unter PL/SQL lautet folgendermaßen:

---

```
CURSOR cursor-name
{(parameter1 parameter1-datatype {:= default1},
...
parameterN parameterN-datatype {:= defaultN})}
IS select-stmt;
```



Die Variablen sind wie folgt definiert:

- *cursor-name* ist der Name des Cursor und unterliegt den Vorgaben von Oracle für die Benennung von Objekten.
- *parameter1* ist der Name des ersten an den Cursor übergebenen Parameters.
- *parameter1-datatype* ist der Datentyp von *parameter1*.
- *default1* ist ein optionaler Standardwert für *parameter1*.
- *parameterN* ist der Name des letzten an den Cursor übergebenen Parameters.
- *parameterN-datatype* ist der Datentyp von *parameterN*.
- *defaultN* ist ein optionaler Standardwert für *parameterN*.
- *select-stmt* ist eine gültige SELECT-Anweisung, die dem deklarierten Cursor zugeordnet ist.

Listing 11.15 zeigt zwei unterschiedliche Cursors - der erste Cursor hat drei Parameter ohne Standardwerte, und der zweite Cursor hat drei Parameter mit bestimmten Standardwerten.

**Listing 11.15: Cursor mit und ohne Standardwerte für Parameter.**

```
cursor patients_with_hypertension
(patient_age number,
normal_dyastolic,
normal_systolic) is
select patient_id, age, dyastolic, systolic
from patient
where
dyastolic > normal_dyastolic * (age+200)/200 and
systolic > normal_systolic * (age+200)/200;
```

```
cursor patients_with_hypertension
(patient_age number default 55,
normal_dyastolic number default 70,
normal_systolic number default 130) is
select patient_id, age, dyastolic, systolic
from patient
where
```

```
dyastolic > normal_dyastolic * (age+200)/200 and  
systolic > normal_systolic * (age+200)/200;
```

---



*Wenn Sie einen der Precompiler von Oracle verwenden, - z.B. Pro\*C - müssen Sie einen Cursor einsetzen, um über eine SELECT-Anweisung mehr als eine Zeile abzurufen.*

---

## Öffnen eines Cursor

Bevor Sie die Zeilen eines Cursor lesen können, müssen Sie den Cursor öffnen. Ist dieser geöffnet, wird dessen SELECT-Anweisung ausgeführt, und Oracle erstellt eine Liste der betreffenden Zeilen. Diese Zeilen werden als *aktive Menge* bezeichnet. Wurde der Cursor ohne Parameter deklariert, ist die Syntax recht einfach.

```
open my_cursor;
```

Wurde der Cursor mit Parameter deklariert, müssen Sie, wenn Sie den Cursor öffnen, für jeden Parameter eine PL/SQL-Variable oder einen Literalwert bereitstellen, wie es im Listing 11.6 gezeigt wird.

### *Listing 11.16: Deklaration eines Cursor mit Parametern.*

```
SQL> declare  
2  
2 cursor patients_with_hypertension  
3 (patient_age number,  
4 normal_dyastolic number) is  
5 select patient_id  
6 from patient  
7 where  
8 dyastolic > normal_dyastolic * (age+200)/200 and  
9 systolic > 180;  
10  
10 Patient_ID Patient.Patient_ID%type;  
11  
11 begin  
12  
12 open patients_with_hypertension (45, 80);  
13  
13 end;  
14 /  
PL/SQL procedure successfully completed
```

Wurde der Cursor mit Parametern deklariert - und für diese Parameter Standardwerte angegeben -, ist es nicht notwendig, daß Sie für jeden Parameter eine PL/SQL-Variable oder einen Literalwert bereitstellen, wie es im Listing 11.7 gezeigt wird.

***Listing 11.17: Standardwerte für verwendete Parameter eines Cursor.***

```
SQL> declare
2
2 cursor patients_with_hypertension
3 (patient_age number default 55,
4 normal_dyastolic number default 70,
5 normal_systolic number default 130) is
6 select patient_id
7 from patient
8 where
9 dyastolic > normal_dyastolic * (age+200)/200 and
10 systolic > normal_systolic * (age+200)/200;
11
11 Patient_ID Patient.Patient_ID%type;
12
12 begin
13
13 dbms_output.enable;
14
14 open patients_with_hypertension;
15
15 loop
16
16 fetch patients_with_hypertension
17 into Patient_ID;
18 exit when patients_with_hypertension%notfound;
19
19 dbms_output.put_line(patient_record.patient_id); 20
20 end loop;
21
21 end;
22 /
N3393
PL/SQL procedure successfully completed.
```

Wurde der Cursor mit Parametern deklariert - aber ohne angegebene Standardwerte für diese Parameter - müssen Sie für jeden Parameter eine PL/SQL-Variable oder einen Literalwert bereitstellen. Listing 11.8 zeigt, wie Oracle die Anweisung zum Öffnen des Cursor zurückweist, wenn die erforderlichen Argumente nicht bereitgestellt werden.

***Listing 11.18: Oracle meldet einen Fehler, wenn für den Cursor keine Argumente geliefert werden.***

```
SQL> declare
2
```



```
2 cursor patients_with_hypertension
3 (patient_age number,
4 normal_dyastolic number,
5 normal_systolic number) is
6 select patient_id
7 from patient
8 where
9 dyastolic > normal_dyastolic * (age+200)/200 and
10 systolic > 180;
11
11 Patient_ID Patient.Patient_ID%type;
12
12 begin
13
13 open patients_with_hypertension;
14
14 end;
15 /
declare
*
```

```
ERROR at line 1:
ORA-06550: line 13, column 1:
PLS-00306: wrong number or types of arguments
in call to 'PATIENTS_WITH_HYPERTENSION'
ORA-06550: line 13, column 1:
PL/SQL: SQL Statement ignored
```

## Lesen von Zeilen eines Cursor

Ist der Cursor einmal geöffnet worden, würde die Abfrage ausgeführt, und die entsprechenden Zeilen würden identifiziert. Um diese Zeilen zu lesen, muß die FETCH-Anweisung ausgeführt werden, die den Wert jeder in der SELECT-Anweisung des Cursor angegebenen Spalte abrufen und in einer PL/SQL-Variablen speichert. Üblicherweise werden Sie Zeilen innerhalb einer Schleife lesen wollen. Zur Veranschaulichung enthält Listing 11.19 einen anonymen SQL-Block (in der ersten Zeile beginnend), in dem Zeilen aus der Instructor-Tabelle gelesen werden.

*Listing 11.19: Lesen von Zeilen eines Cursor.*

```
SQL> declare
2
2 Instructor_ID Instructor.Instructor_ID%type;
3 Last_Name Instructor.Last_Name%type;
4 First_Name Instructor.First_Name%type;
5
5 cursor get_instructors is
6 select Instructor_ID, Last_Name, First_Name
7 from Instructor
```

```
8 order by Instructor_ID;
9
9 begin
10
10 dbms_output.enable;
11
11 open get_instructors;
12
12 loop
13
13 fetch get_instructors into
14 Instructor_ID, Last_Name, First_Name; 15 exit when
get_instructors%notfound;
16
16 dbms_output.put_line(Instructor_ID);
17
17 end loop;
18
18 end;
19 /
A612
B331
B391
D201
D944
E301
E405
E491
G331
J505
L391
M101
P331
R983
S131
T149
W490
Y561
PL/SQL procedure successfully completed.
```

Wenn alle Zeilen gelesen worden sind, sollten Sie die EXIT-Anweisung verwenden, um die Schleife zu beenden.



*Die Syntax lautet folgendermaßen:*

**EXIT** {*label*}{*WHEN condition*}

Die Variablen sind wie folgt deklariert:

- *label* ist der optionale Name des Labels, das die zu verlassende Schleife angibt.
- *condition* ist eine PL/SQL-Bedingung, die einen Boolean-Wert zurückgibt.

Vier bestimmte Attribute sind dem deklarierten Cursor zugeordnet: ROWCOUNT, %FOUND, %NOTFOUND und %ISOPEN. Auf diese Attribute wird verwiesen, indem sie hinter den Namen des Cursor gesetzt werden. Um eine Schleife mit der EXIT-Anweisung zu beenden, verweisen Sie auf das %NOTFOUND-Attribut eines Cursor wie folgt:

```
exit when get_instructors%notfound;
```

## Schließen eines Cursor

Aus zwei Gründen müssen Sie einen Cursor schließen:

- um ihn mit anderen Parametern wieder öffnen zu können
- um die vom Cursor belegten Ressourcen freizugeben.

Wenn ein PL/SQL-Programm einen Cursor nicht schließt, schließt Oracle diesen entweder durch Beenden oder ein DISCONNECT, wenn das Unterprogramm sich von der Oracle-Datenbank abmeldet. Einen Cursor zu schließen ist einfach:

```
close get_instructors;
```

Listing 11.20 zeigt, wie einem Cursor neue Parameter zugewiesen werden, indem der Cursor geschlossen, die Parameterwerte geändert und der Cursor wieder geöffnet wird. Der Cursor `patients_with_hypertension` wird zur Veranschaulichung dieses Vorgangs verwendet. Als erstes wird der Cursor mit `age set to 50` und `normal_dyastolic set to 80` (Zeile 14) geöffnet. Die Zeilen werden in einer Schleife (Zeile 16) gelesen und der Cursor geschlossen. Dann wird der Cursor mit `age equal to 40` und `normal_dyastolic set to 70` (Zeile 22) erneut geöffnet. Die Zeilen werden mit anderen Ergebnissen in einer Schleife gelesen.

**Listing 11.20: Änderung der Argumente eines Cursor durch Schließen und erneutes Öffnen des Cursor.**

```
SQL> declare
2
2 Patient_ID Patient.Patient_ID%type;
3 Age Patient.Age%type;
4 Dyastolic Patient.Dyastolic%type;
5
5 cursor patients_with_hypertension
6 (patient_age number,
```

```
7 normal_dyastolic number) is
8 select patient_id, age, dyastolic
9 from patient
10 where
11 dyastolic > normal_dyastolic * (age+200)/200;
12
12 begin
13
13 dbms_output.enable;
14
14 open patients_with_hypertension (50, 80);
15
15 loop
16
16 fetch patients_with_hypertension
17 into Patient_ID, Age, Dyastolic;
18 exit when patients_with_hypertension%notfound;
19
19 dbms_output.put_line('With age=50, dyas=80: ' || Patient_ID); 20
20 end loop;
21
21 close patients_with_hypertension;
22
22 open patients_with_hypertension (40, 70);
23
23 loop
24
24 fetch patients_with_hypertension
25 into Patient_ID, Age, Dyastolic;
26 exit when patients_with_hypertension%notfound;
27
27 dbms_output.put_line('With age=40, dyas=70: ' || Patient_ID);
28
28 end loop;
29
29 close patients_with_hypertension;
30
30 end;
31 /
With age=50, dyas=80: N3393
With age=40, dyas=70: A2002
With age=40, dyas=70: N3393
With age=40, dyas=70: E3893
PL/SQL procedure successfully completed
```

## Mit Cursor-gesteuerten for-Schleifen arbeiten

<http://www.informit.de/bookshelf/HTM3827220173/data/tag11.htm> (24 von 45) [19.11.2001 22:11:31]

```
SQL> declare
2
2 Instructor_ID Instructor.Instructor_ID%type;
3 Last_Name Instructor.Last_Name%type;
4 First_Name Instructor.First_Name%type;
5
5 cursor Get_Associate_Profs is
6 select Instructor_ID, Last_Name, First_Name
7 from Instructor
8 where Position = 'ASSOCIATE PROFESSOR'
9 order by Instructor_ID;
10
10 begin
11
11 dbms_output.enable;
12
12 for Get_Associate_Profs_Rec in Get_Associate_Profs loop
13
13 dbms_output.put_line('Last name: ' ||
13 Get_Associate_Profs_Rec.Last_Name);
14
14 end loop;
15
15 Last_Name := Get_Associate_Profs_Rec.Last_Name;
16
16 end;
17 /
declare
*
```

```
ERROR at line 1:
ORA-06550: line 15, column 14:
PLS-00201: identifier 'GET_ASSOCIATE_PROFS_REC.LAST_NAME'
[ccc]must be declared
ORA-06550: line 15, column 1:
PL/SQL: Statement ignored.
```

## %found oder %notfound

In den vorangegangenen Beispielen wurde das Attribut %NOTFOUND dazu verwendet, festzustellen, ob eine FETCH-Anweisung eine Zeile einlas oder nicht. Nachdem alle Zeilen der aktiven Gruppe eingelesen wurden und die letzte FETCH-Anweisung keine mehr einlesen kann, ergibt die Auswertung von %NOTFOUND TRUE.



Vor dem Aufruf der *FETCH*-Anweisung gibt *%NOTFOUND* *NULL* zurück. Enthält Ihr *SQL*-Programm eine Schleife, in der die *FETCH*-Anweisung nicht aufgerufen wird, sollten Sie überlegen, ob Sie nicht eine Validierung von *%NOTFOUND* auf den Wert *NULL* vornehmen.

---

## Ermitteln der Zeilenanzahl mit Hilfe von *%rowcount*

Sie benötigen keinen Zähler, um die Anzahl der aus einem Cursor eingelesenen Zeilen zu verfolgen. Statt dessen verweisen Sie auf das Attribut *%ROWCOUNT* des Cursor. Wie Sie in Zeile 13 des Listing 11.23 sehen können, gibt *%ROWCOUNT* die *laufende* Anzahl der eingelesenen Zeilen zurück.

*Listing 11.23: Ermittlung der Anzahl der aus einem Cursor gelesenen Zeilen mit Hilfe von %rowcount.*

```
SQL> declare
2
2 Instructor_ID Instructor.Instructor_ID%type;
3 Last_Name Instructor.Last_Name%type;
4 First_Name Instructor.First_Name%type;
5
5 cursor Get_Associate_Profs is
6 select Instructor_ID, Last_Name, First_Name
7 from Instructor
8 where Position = 'ASSOCIATE PROFESSOR'
9 order by Instructor_ID;
10
10 begin
11
11 dbms_output.enable;
12
12 for Get_Associate_Profs_Rec in Get_Associate_Profs loop
13
13 dbms_output.put_line ('Rowcount: ' ||
[ccc]Get_Associate_Profs%rowcount);
14
14 end loop;
15
15 end;
16 /
Rowcount: 1
Rowcount: 2
Rowcount: 3
Rowcount: 4
Rowcount: 5
Rowcount: 6
```

Rowcount: 7

PL/SQL procedure successfully completed.

Anstatt eine Schleife zu verlassen, wenn der Cursor keine Zeilen mehr enthält, können Sie, wie in Listing 11.24 gezeigt, für das Erreichen einer bestimmten Anzahl in %ROWCOUNT eine Abbruchbedingung festlegen (siehe Zeile 13).

***Listing 11.24: Verlassen einer Cursor-gesteuerten Schleife, nachdem eine bestimmte Anzahl von Zeilen gelesen wurde.***

```
SQL> declare
2
2 Instructor_ID Instructor.Instructor_ID%type;
3 Last_Name Instructor.Last_Name%type;
4 First_Name Instructor.First_Name%type;
5
5 cursor Get_Associate_Profs is
6 select Instructor_ID, Last_Name, First_Name
7 from Instructor
8 where Position = 'ASSOCIATE PROFESSOR'
9 order by Instructor_ID;
10
10 begin
11
11 dbms_output.enable;
12
12 for Get_Associate_Profs_Rec in Get_Associate_Profs loop
13
13 exit when Get_Associate_Profs%rowcount >= 5;
14 dbms_output.put_line ('Rowcount: ' ||
[ccc]Get_Associate_Profs%rowcount);
15
15 end loop;
16
16 end;
17 /
Rowcount: 1
Rowcount: 2
Rowcount: 3
Rowcount: 4
PL/SQL procedure successfully completed.
```

In Listings 11.23 und 11.24 sehen Sie, wie Sie %ROWCOUNT einsetzen können, um die Anzahl der vom Cursor eingelesenen Zeilen zu verfolgen.

## Erzwingen von Firmenregeln mit Hilfe von Datenbank-Triggern



Mit dem Einsatz des bisher über SQL und PL/SQL Gelernten ist es an der Zeit, die Welt der Datenbank-Trigger zu erkunden. In einem Informationssystem, das eine nicht-relationale Datenbank zur Grundlage hat, werden die Firmenregeln der Organisation durch Anwendungs-Software umgesetzt. So könnte z.B. eine Firmenregel lauten, daß eine Bestellung über eine benötigte Menge ins System eingegeben wird, sobald eine manuelle Inventur für einen Artikel einen Bestand ergibt, der unter der Lagermenge dieses Artikels liegt. Diese Regel würden Sie üblicherweise durchsetzen, indem Sie unter Cobol oder einer anderen Programmiersprache eine Routine schreiben würden, die an entsprechender Stelle im Programm aufgerufen würde. Diese Methode birgt jedoch einige Probleme:

- Die Regel wird nur dann durchgesetzt, wenn ein Anwendungs- oder Hilfsprogramm diese Routine aufruft.
- Eine Liste der Algorithmen zu erhalten, die zur Durchsetzung von Regeln verwendet werden, ist schwierig. Sie müssen sich auf Unterlagen stützen (die unzuverlässig sein könnten) oder vor der unangenehmen Aufgabe stehen, Anwendungs-Software zu überarbeiten.

Oracle, wie auch andere moderne RDBMS, stellen Mechanismen - Datenbank-Trigger - zur Verfügung, die die Aufgabe erleichtern, Firmenregeln einer Organisation durchzusetzen. Ein *Datenbank-Trigger* besteht aus einer Gruppe von PL/SQL-Anweisungen, die dann ausgeführt werden, wenn eine SQL-Anweisung - eine DELETE-, UPDATE- oder INSERT-Anweisung - auf eine Tabelle angewendet wird. Einen Datenbank-Trigger können Sie zur Erledigung folgender Aufgaben einsetzen:

- Durchsetzung ausgeklügelter Sicherheitsrichtlinien.
- Ändern eines Spaltenwerts auf der Grundlage von Werten in anderen Spalten derselben oder einer anderen Tabelle.
- Komplexe Validierung von Spaltenwerten - so könnte es z.B. notwendig sein, daß Sie einen Spaltenwert mit dem Aggregat eines Spaltenwerts einer anderen Tabelle vergleichen müssen.
- Dokumentieren der Änderungen an einem Datensatz durch Schreiben der geänderten Werte in eine andere Tabelle.



*Ein Datenbank-Trigger ist eine Gruppe von PL/SQL-Anweisungen, die ausgeführt werden, wenn der Inhalt einer Tabelle in irgendeiner Weise verändert wird - entweder durch eine INSERT-, UPDATE- oder DELETE-Anweisung.*

---

In dieser Lektion werden die Details zur Erstellung eines Trigger zu verschiedenen Zwecken untersucht.

## Erstellen eines Trigger

Um Ihre Trigger zu schreiben sollten Sie einen Texteditor verwenden. Die Oracle-Anweisung CREATE TRIGGER erstellt (oder ersetzt) einen Trigger, der ausgelöst wird, wenn in einer Tabelle ein bestimmtes

Ereignis auftritt.

---



Die Syntax für die *CREATE-TRIGGER*-Anweisung lautet wie folgt:

---

```
CREATE [OR REPLACE] TRIGGER trigger-name {BEFORE | AFTER}  
triggering-event ON table-name  
[FOR EACH ROW]  
[WHEN (condition)]  
PL/SQL-block
```

Die Variablen sind wie folgt deklariert:

- *trigger-name* ist der Name des zu erstellenden Trigger und unterliegt den Namenskonventionen von Oracle.
- *triggering-event* ist entweder INSERT, UPDATE oder DELETE entsprechend den drei DML-Anweisungen.
- *table-name* ist der Name der dem Trigger zugeordneten Tabelle.  
FOR EACH ROW ist eine optionale Klausel, die bei ihrer Verwendung den Trigger für jede betroffene Zeile auslöst.
- *condition* ist eine optionale Oracle-Bedingung vom Typ Boolean, die, wenn sie den Wert TRUE aufweist, den Trigger auslöst.
- *PL/SQL-block* ist der PL/SQL-Block, der ausgeführt wird, wenn der Trigger ausgelöst wird - er wird auch als *Trigger-Rumpf* bezeichnet.

Die folgenden Abschnitte erläutern die Verwendung dieser *CREATE-TRIGGER*-Anweisungselemente.

## Trigger auf Anweisungsebene und auf Zeilenebene

Ein Datenbank-Trigger gehört in eine der beiden folgenden Kategorien:

- Trigger auf *Anweisungsebene* enthalten nicht die Klausel FOR EACH ROW in der *CREATE-TRIGGER*-Anweisung.
- Trigger auf *Zeilenebene* enthalten die Klausel FOR EACH ROW in der *CREATE-TRIGGER*-Anweisung.

Ein Trigger auf Anweisungsebene wird durch das auslösende Ereignis nur einmal ausgelöst und hat keinen Zugang zu den Spaltenwerten der einzelnen vom Trigger betroffenen Zeilen. Ein Trigger auf Zeilenebene wird für jede vom Trigger betroffene Zeile ausgelöst und kann auf die Ausgangswerte und neuen Werte der durch eine SQL-Anweisung bearbeiteten Spalte zugreifen.

Im allgemeinen verwenden Sie Trigger auf Anweisungsebene zur Bearbeitung von Informationen über die SQL-Anweisung, die den Trigger ausgelöst hat - z.B. wer ihn ausgelöst hat und wann er ausgelöst wurde. Typischerweise verwenden Sie einen Trigger auf Zeilenebene, wenn Sie die Spaltenwerte einer Zeile kennen müssen, um eine Firmenregel umzusetzen.

## Zugriff auf Spaltenwerte im Trigger-Rumpf

Innerhalb eines Trigger-Rumpfs kann ein Trigger auf Zeilenebene auf den Spaltenwert einer Zeile verweisen, der zum Zeitpunkt der Auslösung des Trigger bestand. Diese Werte hängen davon ab, welche SQL-Anweisung den Trigger ausgelöst hat.

- Bei einer INSERT-Anweisung sind die Werte, die eingefügt werden, in *:new.column-name* enthalten, und *column-name* steht für eine Spalte der Tabelle.
- Bei einer UPDATE-Anweisung ist der ursprüngliche Spaltenwert in *:old.column-name*; und der neue Spaltenwert in *:new.column-name* enthalten.
- Bei einer DELETE-Anweisung sind die Spaltenwerte der zu entfernenden Zeile in *:old.column-name* enthalten.

## Auslösen von Ereignissen

Wenn Sie einen Trigger erstellen, legen Sie fest, welches Ereignis ihn auslösen wird. Die drei möglichen Ereignisse sind:

- Das Einfügen einer neuen Zeile in eine Tabelle mit der INSERT-Anweisung.
- Das Aktualisieren einer Reihe von Zeilen (oder keiner Zeile) durch eine UPDATE-Anweisung.
- Das Entfernen einer Reihe von Zeilen (oder keiner Zeile) mit einer DELETE-Anweisung.

Außerdem lassen sich diese drei Trigger-auslösenden Ereignisse derart kombinieren, daß ein Trigger immer dann ausgelöst wird, wenn eine INSERT-, UPDATE- oder DELETE-Anweisung ausgeführt wird, wie es in Listing 11.25 gezeigt wird.

***Listing 11.25: Deklaration eines Trigger, der bei jedem Trigger-auslösenden Ereignis ausgelöst wird.***

```
SQL> create or replace trigger Block_Trade_After_All After
2 insert or update or delete on Tab1
3 for each row
4
4 declare
5
5 begin
6
6 insert into Tab11
7 (col11)
8 values
```

```
9 (11);  
10  
10 end;  
11 /  
Trigger created.
```

## before- und after-Trigger

Ein BEFORE-Trigger auf Zeilenebene wird vor Ausführung des Trigger-auslösenden Ereignisses ausgelöst. Somit können Sie einen BEFORE-Trigger auf Zeilenebene zur Änderung von Spaltenwerten einer Zeile einsetzen. Ein AFTER-Trigger auf Zeilenebene wird nach Auftreten des Trigger-auslösenden Ereignisses ausgelöst. Mit einem AFTER-Trigger auf Zeilenebene können Sie keine Spaltenwerte ändern.

## Mögliche Trigger für eine Tabelle

Auf der Grundlage aller in einer CREATE-TRIGGER-Anweisung möglichen Permutationen, kann eine Tabelle bis zu 12 verschiedene Trigger aufweisen:

- Sechs Trigger auf Zeilenebene für BEFORE DELETE, BEFORE INSERT, BEFORE UPDATE, AFTER DELETE, AFTER INSERT und AFTER UPDATE
- Sechs Trigger auf Anweisungsebene für BEFORE DELETE, BEFORE INSERT, BEFORE UPDATE, AFTER DELETE, AFTER INSERT und AFTER UPDATE



*Wenn Sie den Einsatz eines Entity-Relationship-Modeling-Tools für den Datenbankentwurf planen - was ratsam ist - werden Sie feststellen, daß die meisten von ihnen automatisch Datenbank-Trigger auf Basis der von Ihnen definierten Primär- und Fremdschlüssel generieren. Einige Tools, wie ERwin von LogicWorks, erstellen Trigger entweder direkt durch eine Oracle-Verbindung oder speichern Trigger in einer Skriptdatei. Wenn Sie die letzte Methode wählen, können Sie das Erstellungsskript für den Trigger verändern, indem Sie jedem generierten Trigger anwendungsspezifische Firmenregeln hinzufügen.*

---

Nur weil Sie alle 12 Trigger-Arten für eine Tabelle erstellen können, heißt dies nicht, daß Sie sie auch alle verwenden müssen. Tatsächlich sollten Sie bei der Erstellung von Triggern für Ihre Tabellen ein gutes Urteilsvermögen beweisen.

---



*Ab Version 7.1 unterstützt Oracle mehrere gleichartige Trigger für ein und dieselbe Tabelle.*

*Installationen von Oracle 7.0 hingegen, die Snapshots verwendeten, konnten keine AFTER-ROW-Trigger für eine Mastertabelle erstellen, weil die Snapshot-Logs AFTER-ROW-Trigger für die gleiche Tabelle verwendeten. Wenn Sie nicht gerade vorhaben, einen AFTER-ROW-Trigger für eine Tabelle, auf die von Snapshot Bezug genommen wird, zu verwenden, sollten Sie es vermeiden, mehrere gleichartige Trigger für eine bestimmte Tabelle zu definieren - zu leicht entstehen Fehler beim Entwurf, und es leidet die Übersichtlichkeit.*

Der folgende Teil der Lektion untersucht einige Verwendungsmöglichkeiten von Triggern.

## Spaltenwerte mit Hilfe eines Trigger auf Gültigkeit prüfen

Als DBA (Datenbankadministrator) einer Kreditkartengesellschaft sind Sie für die Umsetzung der Kreditrichtlinien durch Datenbank-Trigger verantwortlich. Untersuchungen des Unternehmens haben ergeben, daß die Wahrscheinlichkeit eines Kreditkartenbetrugs mehr als 80 Prozent beträgt, wenn innerhalb von drei Tagen auf einem einzelnen Konto Belastungen von mehr als \$ 1000 zusammenkommen. Der Betriebsleiter möchte, daß alle Konten, auf die diese Kriterien zutreffen, in einer gesonderten Tabelle aufgeführt werden, in der sie dann genau untersucht werden können.

Zur Erledigung dieser Aufgabe erstellen Sie den Trigger für die Tabelle Credit\_Card\_Log, der vor dem Einfügen einer Zeile ausgelöst wird. Der Trigger schaut, ob der Gesamtbetrag der Belastungen der letzten drei Tage für eine bestimmte Kartenummer \$ 1000 übersteigt und führt in diesem Fall ein INSERT für die Tabelle Credit\_Charge\_Attempt\_Log aus, in der dieser Datensatz dann von Kreditsachbearbeitern untersucht werden kann. Listing 11.26 veranschaulicht die Erstellung dieses Trigger.

**Listing 11.26: Erzeugung eines before-insert-Trigger.**

```
SQL> create or replace trigger Credit_Charge_Log_Ins_Before before 2
insert on Credit_Charge_Log
3 for each row
4
5 declare
6
7 total_for_past_3_days number;
8
9 begin
10
11 - Check the credit charges for the past 3 days.
12 - If they total more than $1000.00, log this entry
13 - in the Credit_Charge_Attempt_Log for further handling.
14
15 select sum(amount)
```

```
11 into total_for_past_3_days
12 from Credit_Charge_Log
13 where
14 Card_Number = :new.Card_Number and
15 Transaction_Date >= sysdate-3;
16
16 if total_for_past_3_days > 1000.00 then
17
17 insert into Credit_Charge_Attempt_Log
18 (Card_Number, Amount, Vendor_ID, Transaction_Date)
19 values
20 (:new.Card_Number, :new.Amount,
: new.Vendor_ID, :new.Transaction_Date);
21
21 end if;
22
22 end;
23 /
Trigger created.
```

Um den Trigger so einzustellen, daß er ausgelöst wird, initialisieren Sie die Tabelle Credit\_Card\_Log mit einigen Zeilen, wie es in Listing 11.27 gezeigt wird.

***Listing 11.27: Inhalt der Credit\_Charge\_Log-Tabelle, bevor die nächste Zeile eingefügt wird.***

```
SQL> select * from credit_charge_log;
CARD_NUMBER AMOUNT VENDOR_I TRANSACTI
-----
8343124443239383 128.33 12345678 19-JUN-95
9453128834232243 83.12 98765432 18-JUN-95
4644732212887321 431.1 18181818 19-JUN-95
0944583312453477 211.94 09090909 18-JUN-95
0944583312453477 413.81 08080808 18-JUN-95
0944583312453477 455.31 91919191 19-JUN-95
0944583312453477 225 12341234 20-JUN-95
0944583312453477 512.22 12341234 20-JUN-95
rows selected.
```

Vor dem Einfügen einer Zeile in die Tabelle für die Kartennummer 0944583312453477 wird der Trigger ausgelöst. Die Tabelle wird abgefragt, ob die Belastungen für die Kartennummer innerhalb der letzten drei Tage \$ 1000 übersteigt. Ist dies der Fall, wird der Tabelle Credit\_Charge\_Attempt\_Log eine Zeile hinzugefügt, wie Sie es in Listing 11.28 (Zeile 1) sehen können. Weil die Kartennummer 0944583312453477 innerhalb der letzten drei Tage mit mehr als \$ 1000 belastet wurde, fügt der Trigger eine Zeile in die Tabelle Credit\_Charge\_Attempt\_Log ein.

***Listing 11.28: Der Trigger wird ausgelöst, wenn eine Zeile in die Credit\_Charge\_Log-Tabelle eingefügt wird.***

```
SQL> insert into Credit_Charge_Log
```

```
2 (Card_Number, Amount, Vendor_ID, Transaction_Date)
3 values
4 ('0944583312453477', 128.28, '43214321', '20-JUN-95');
row created.
SQL> select * from Credit_Charge_Attempt_Log;
CARD_NUMBER AMOUNT VENDOR_I TRANSACTI
-----
0944583312453477 128.28 43214321 20-JUN-95
```

## Sicherheit mit Hilfe eines Trigger erzwingen

Nachfolgend sehen Sie ein Beispiel, wie Sie mit Hilfe eines Trigger Sicherheitsrichtlinien erzwingen können. Die Datenbank der Acme Corporation ist so angelegt, daß bei einer erfolgten Auslieferung der Tabelle Shipment eine Zeile hinzugefügt werden muß. Diese Tabelle Shipment enthält eine Spalte Manual\_Check, in der angegeben ist, ob der Angestellte im Auslieferungslager telefonisch die Richtigkeit der Lieferungsanforderung bestätigen muß. Zur Reduzierung von Unterschlagungen lautet die Richtlinie der Firma, daß der Angestellte im Auslieferungslager jede Lieferungsanforderung bestätigen muß, die nach Ende der regulären Arbeitszeit eingeht - 17.00 Uhr.

Als DBA sind Sie für die Umsetzung dieser Richtlinie verantwortlich. Wie in Listing 11.29 zu sehen ist, können Sie einen Trigger Shipment\_Ins\_Before erstellen, der vor der Ausführung einer INSERT-Anweisung auf die Tabelle Shipment ausgelöst wird. Der Trigger-Rumpf besteht aus einer einzigen PL/SQL-Anweisung - der Zuweisung von Y der Spalte Manual\_Check. Zusätzlich entscheiden Sie sich für die Verwendung einer WHEN-Klausel, damit der Trigger nur nach 17.00 Uhr ausgelöst wird.

*Listing 11.29: Erzeugung eines Trigger, der eine Bedingung erfüllen muß, bevor er ausgelöst wird.*

```
SQL> create or replace trigger Shipment_Ins_Before before
2 insert on Shipment
3 for each row
4 when (to_number(to_char(sysdate, 'HH24')) > 17)
5
5 declare
6
6 begin
7
7 :new.Manual_Check := 'Y';
8
8 end;
9 /
Trigger created.
```

Nachdem der Trigger erstellt wurde, können Sie ihn testen. Wie aus Listing 11.30 ersichtlich, ist die aktuelle Zeit nach 17.00 Uhr - es ist 19.00 Uhr. Wird der Tabelle Shipment eine Zeile hinzugefügt, wird die Spalte Manual\_Check wie beabsichtigt auf Y gesetzt.

***Listing 11.30: Der Trigger prüft die Zeit und wird anschließend ausgelöst.***

```
SQL> select to_char(sysdate, 'HH24') from dual;
TO_CHAR(SYSDATE, 'HH24')
-----
SQL> insert into Shipment
2 (Shipment_ID, Product_Code, Quantity, Customer_ID)
3 values
4 ('SHIP1001', 'PROD123', 100, 'CUST999');
row created.
SQL> select * from Shipment;
SHIPMENT_ID PRODUCT_CODE QUANTITY CUSTOMER_ID M ENTERED_BY
-----
SHIP1001 PROD123 100 CUST999 Y
```

## Setzen von Spaltenwerten mit Hilfe eines Trigger

Eine weitere Einsatzmöglichkeit für einen Trigger ist das Setzen eines Spaltenwerts auf einen bestimmten Wert, ehe sich eine SQL-Anweisung auswirkt. Das folgende Szenario demonstriert diesen Vorgang. Nehmen Sie an, eine Tabelle namens `Block_Trade_Log` wird zur Auflistung des Handels mit Wertpapieren an der NASDAQ (National Association of Securities Dealers Automated Quotations) verwendet. Die Tabelle enthält: das Zeichen der Aktie, den Verkaufspreis, die Anzahl der gehandelten Wertpapiere, wann der Handel erfolgte, ob die Wertpapiere ge- oder verkauft wurden und den aktuellen Durchschnittswert des Wertpapiers für die letzten drei Tage. Beim Einfügen einer Zeile in diese Tabelle wird ein Trigger verwendet, um den Wert für `Running_Avg_3_Days` zu setzen. Listing 11.31 zeigt, wie dieser Trigger erzeugt wird.

***Listing 11.31: Setzen eines Spaltenwerts in einem Trigger.***

```
create or replace trigger Block_Trade_Log_BI before
insert on Block_Trade_Log
for each row
declare
Running_Avg number;
begin
select avg(price)
into Running_Avg
from Block_Trade_Log
where
Stock_Symbol = :new.Stock_Symbol and
Timestamp >= SYSDATE-3;
:new.Running_Avg_3_Days := Running_Avg;
end;
/
```

Beachten Sie, daß der Wert für `Running_Avg_3_Days` gesetzt wird, indem der Wert `:new.Running_Avg_3_Days := Running_Avg;` zugewiesen wird. Vergessen Sie nicht, daß auf die tatsächlich in einer Tabelle gespeicherten Werte mit `:new` Bezug genommen wird, wenn das Trigger-auslösende Ereignis



eine INSERT-Anweisung ist.

Listing 11.32 zeigt den Inhalt der Tabelle `Block_Trade_Log`. Beachten Sie die beiden Aktien mit dem Zeichen `QQQQQ`; eine zu \$ 102,125 und die andere zu \$ 103,5. Wird für diese Aktie eine weitere Zeile in die Tabelle `Block_Trade_Log` eingefügt, wird der Trigger ausgelöst und der aktuelle Durchschnittswert der letzten drei Tage - 102,8125 - dieser Aktie berechnet und der Spalte `Running_Avg_3_Days` zugewiesen.

**Listing 11.32: Auslösen des Trigger mit einer insert-Anweisung.**

```
SQL> select * from block_trade_log;
STOCK_ PRICE BLOCKS_TRADED B RUNNING_AVG_3_DAYS TIMESTAMP
-----
QQQQQ 102.125 100 B 19-JUN-95
QQQQQ 103.5 100 S 19-JUN-95
VVVVV 55.75 3000 S 19-JUN-95
VVVVV 55.5 1000 B 20-JUN-95
SQL> insert into block_trade_log
2 (Stock_Symbol, Price, Blocks_Traded, Bought_Sold, Timestamp)
3 values
4 ('&stock',&price,&numblocks, '&BS', '&date')
5 ;
Enter value for stock: QQQQQ
Enter value for price: 104.25
Enter value for numblocks: 300
Enter value for bs: B
Enter value for date: 20-JUN-95
old 4: ('&stock',&price,&numblocks, '&BS', '&date')
new 4: ('QQQQQ',104.25,300, 'B', '20-JUN-95')
row created.
SQL> select * from block_trade_log;
STOCK_ PRICE BLOCKS_TRADED B RUNNING_AVG_3_DAYS TIMESTAMP
-----
QQQQQ 102.125 100 B 19-JUN-95
QQQQQ 103.5 100 S 19-JUN-95
VVVVV 55.75 3000 S 19-JUN-95
VVVVV 55.5 1000 B 20-JUN-95
QQQQQ 104.25 300 B 102.8125 20-JUN-95
```

## Kaskadierende Trigger

Das Zusammenwirken von Triggern kann recht komplex sein. So können Sie z.B. einen Trigger erstellen, der bei seiner Auslösung einen weiteren Trigger auslöst. Derartige Trigger werden als *kaskadierende Trigger* bezeichnet. Um das Konzept kaskadierender Trigger zu erläutern, schauen Sie sich die folgenden drei einfachen Tabellen in Listing 11.33 an - tab1, tab2 und tab3. Zu Beginn hat jede Tabelle eine einzige Zeile.

**Listing 11.33: Auslösen des Trigger mit einer insert-Anweisung.**

```
create table tab1
(col1 number);
create table tab2
(col2 number);
create table tab3
(col3 number);
SQL> select * from tab1;
COL1
-----
7
SQL> select * from tab2;
COL2
-----
10
SQL> select * from tab3;
COL3
-----
13
```

Erstellen Sie für Tabelle tab1 einen BEFORE-UPDATE-Trigger auf Zeilenebene, der den alten Wert der Spalte col1 aus der Tabelle tab1 in die Tabelle tab2 einfügt, wie es in Listing 11.34 gezeigt wird. Für die Tabelle tab2 erstellen Sie einen BEFORE-INSERT-Trigger auf Zeilenebene, der die Tabelle tab3 aktualisiert und den Wert der Spalte col3 auf den neuen Wert der Spalte col2 setzt. Schließlich erstellen Sie für die Tabelle tab3 einen AFTER-UPDATE-Trigger auf Anweisungsebene, der in Tabelle tab3 eine Zeile mit dem Wert 27 der Spalte col3 setzt.

*Listing 11.34: Erzeugung eines Trigger, der einen anderen Trigger auslöst.*

```
SQL> create or replace trigger tab1_Update_Before before
2 update on tab1
3 for each row
4
4 declare
5
5 begin
6
6 insert into tab2
7 (col2)
8 values
9 (:old.col1);
10
10 end;
11 /
Trigger created.
SQL> create or replace trigger tab2_Insert_Before before
2 insert on tab2
3 for each row
4
4 declare
```

```
5
5 begin
6
6 update tab3
7 set
8 col3 = :new.col2;
9
9 end;
10 /
Trigger created.
SQL> create or replace trigger tab3_Update_After after
2 update on tab3
3
3 declare
4
4 begin
5
5 insert into tab3
6 (col3)
7 values
8 (27);
9
9 end;
10 /
Trigger created.
```



*Eine Tabelle ändert sich, wenn ihr Inhalt durch eine INSERT-, UPDATE- oder DELETE-Anweisung verändert wird und diese Änderungen noch nicht festgeschrieben sind. Ein Trigger auf Zeilenebene kann Inhalte einer sich verändernden Tabelle weder lesen noch verändern, weil eine sich ändernde Tabelle sich in einem nicht definierten Zustand befindet. Die einzige Ausnahme stellt ein BEFORE-INSERT-Trigger auf Zeilenebene dar, weil eine Tabelle mit einem Fremdschlüssel Spaltenwerte der Tabelle mit dem Primärschlüssel verändern kann. Um mehr über sich verändernde Tabellen zu erfahren, schauen Sie im online verfügbaren Kapitel 8, des Oracle7 Server Application Developer's Guide, nach. Wenn Sie die Oracle-Dokumentation auf Ihren Windows-95-PC installiert haben, finden Sie sie unter C:\Orawin95\Doc\AS2536\_1.PDF; sie liegt als PDF-Datei (Adobe Portable Document Format) vor und kann mit dem Adobe Acrobat Reader gelesen werden.*

---

Was geschieht nun, wenn in tab1 eine Zeile aktualisiert wird? Wie in Listing 11.35 zu sehen ist, finden die folgenden Veränderungen statt:

- In tab1 ist der Wert von col1 auf 8 aktualisiert worden.
- In tab2 hat der Trigger Tab1\_Update\_Before eine Zeile mit dem alten Wert 7 von col1 eingefügt.
- In tab3 wurde der Trigger Tab2\_Insert\_Before infolge der neuen Zeile in tab2 ausgelöst und der Wert

von col3 auf den in tab2 eingefügten Wert 7 gesetzt. Als tab3 aktualisiert wurde, hat der Trigger Tab\_Update\_Before eine weitere Zeile in tab3 mit dem Wert 27 eingefügt.

**Listing 11.35: Testen kaskadierender Trigger.**

```
SQL> update tab1
2 set col1 = 8;
row updated.
SQL> select * from tab1;
COL1
-----
8
SQL> select * from tab2;
COL2
-----
10
7
SQL> select * from tab3;
COL3
-----
7
27
```



*Standardmäßig ist die Anzahl kaskadierender Trigger auf 32 begrenzt. Denken Sie jedoch immer daran, daß Ihr Verständnis für die Auswirkungen der Anweisungen INSERT, UPDATE und DELETE in einem umgekehrt proportionalen Verhältnis zur Anzahl der dieser SQL-Anweisung zugeordneten kaskadierenden Trigger steht. Mit anderen Worten, machen Sie es nicht unnötig kompliziert.*

---

## **commit- und rollback-Anweisungen in Triggern nicht erlaubt**

In einem Datenbank-Trigger können Sie keine COMMIT- oder ROLLBACK-Anweisung ausführen.

---



*Als Rollback wird der Prozeß des Verwerfens oder Rückgängigmachens aller an der Datenbank von einem Benutzer seit Ausführung der letzten COMMIT-Anweisung oder dem Beginn der Datenbank-Arbeitssitzung durchgeführten Änderungen bezeichnet.*

---

Außerdem kann ein Trigger keine gespeicherte Prozedur, Funktion oder vordefiniertes Unterprogramm aufrufen, die eine ROLLBACK- oder COMMIT-Anweisung ausführen. Oracle besitzt diese Einschränkung aus einem wichtigen Grund. Trifft ein Trigger auf einen Fehler, sollten alle von diesem Trigger verursachten Änderungen an der Datenbank rückgängig gemacht werden. Hätte der Trigger aber einen Teil der Änderungen an der Datenbank festgeschrieben, wäre Oracle nicht in der Lage, die gesamte Transaktion rückgängig zu machen.

## Aufruf von gespeicherten Prozeduren in einem Trigger

Sie können eine gespeicherte Prozedur oder Funktion, ob eigenständig oder als Teil eines Pakets, aus dem PL/SQL-Block eines Datenbank-Trigger aufrufen. Als Beispiel folgt eine Version des Trigger `Block_Trade_Log_BI`, die ursprünglich aus Listing 11.31 stammt und die gespeicherte Funktion `Get_3_Day_Running_Avg` aufruft. Der Trigger basiert auf der Tabelle `Block_Trade_Log` aus der vorstehenden Lektion. Listing 11.36 enthält die gespeicherte Funktion, auf die der Trigger Bezug nehmen wird.

*Listing 11.36: Erstellen der gespeicherten Funktion, die von einem Trigger ausgelöst werden soll.*

```
SQL> create or replace function Get_3_Day_Running_Avg
2 (Stock_Symb in varchar2)
3 return number is
4
4 Running_Avg number;
5
5 begin
6
6 select avg(price)
7 into Running_Avg
8 from Block_Trade_Log
9 where
10 Stock_Symbol = Stock_Symb and
11 Timestamp >= SYSDATE-3;
12
12 return Running_Avg;
13
13 end;
14 /
Function created.
```

Listing 11.37 enthält eine modifizierte Version von `Block_Trade_Log_BI`, die die gespeicherte Funktion `Get_3_Day_Running_Avg` aufruft.

*Listing 11.37: Änderung der gespeicherten Funktion, die von einem Trigger ausgelöst werden soll.*

```
SQL> create or replace trigger Block_Trade_Log_BI before
```

```
2 insert on Block_Trade_log
3 for each row
4
4 declare
5
5 Running_Avg number;
6
6 begin
7
7 :new.Running_Avg_3_Days :=
[ccc]Get_3_Day_Running_Avg (:new.Stock_Symbol);
8
8 end;
9 /
Trigger created.
```

## Löschen, Aktivieren und Deaktivieren von Triggern

Haben Sie sich entschlossen, daß Sie einen bestimmten Trigger wirklich nicht einsetzen wollen, können Sie ihn mit der folgenden Anweisung löschen:

```
DROP TRIGGER trigger-name
```

Die Variable *trigger-name* ist der Name des zu löschenden Trigger.

Um z.B. den Trigger DELETE AFTER für die Tabelle Repair Header zu löschen, geben Sie folgende Anweisung in SQL\*Plus ein:

```
SQL> drop trigger Repair_Header_Delete_After;
Trigger dropped.
```

Manchmal ist das Löschen eines Trigger eine zu drastische Maßnahme. Statt dessen möchten Sie ihn vielleicht nur zeitweilig außer Kraft setzen. Sie können einen Trigger solange deaktivieren, bis dessen Einsatz wieder sinnvoll ist. Um einen Trigger vorübergehend zu deaktivieren, verwenden Sie die ALTER-TRIGGER-Anweisung

```
ALTER TRIGGER trigger-name DISABLE.
```

Die Variable *trigger-name* ist der Name des zu deaktivierenden Trigger.

Das folgende Beispiel deaktiviert den Trigger Repair\_Header\_Delete\_After;

```
SQL> alter trigger Repair_Header_Delete_After disable;
Trigger altered.
```

Um einen deaktivierten Trigger zu aktivieren, verwenden Sie die Anweisung

**ALTER TRIGGER *trigger-name* ENABLE.**

Die Variable *trigger-name* ist der Name des zu aktivierenden Trigger.

So können Sie z.B. den Trigger Repair\_Header\_Delete\_After durch folgenden Befehl aktivieren:

```
SQL> alter trigger Repair_Header_Delete_After enable;  
Trigger altered.
```

## Zusammenfassung

Meinen Glückwunsch! Sie haben in dieser Lektion eine Menge an Stoff bearbeitet. Die wichtigsten Erkenntnisse dieser Lektion waren:

- Unter PL/SQL wird ein Fehler oder eine Meldung Exception genannt.
- Eine Anzahl vordefinierter Exceptions sind Oracle-SQL-Fehlern zugeordnet.
- Ein Exception-Handler ist eine Reihe PL/SQL-Anweisungen, die als Antwort auf eine ausgelöste Exception ausgeführt werden.
- Sie können anwendungsspezifische Exceptions deklarieren und für diese Exception-Handler bestimmen.
- Ein PL/SQL-Block kann einen Exception-Teil mit einem oder mehreren Exception-Handlern enthalten.
- Sie verwenden die RAISE-Anweisung zum Auslösen einer benutzerdefinierten Exception.
- Oracle stellt eine Prozedur namens RAISE\_APPLICATION\_ERROR zur Verfügung, um einen anwendungsspezifischen Fehler einem aufrufenden SQL-Unterprogramm zu übergeben.
- Könnte eine Abfrage mehr als eine Zeile zurückgeben, müssen Sie einen Cursor verwenden.
- Ein Cursor ist ein Fenster auf die Menge der von einer Abfrage abgerufenen Zeilen.
- Sie müssen jeden expliziten Cursor im Deklarationsabschnitt eines PL/SQL-Unterprogramms deklarieren.
- Sie können einen Cursor mit Parametern mit optionalen Standardwerten deklarieren. Die Werte eines jeden Parameters können beim Öffnen des Cursor angegeben werden.
- Die einem Cursor zugeordnete Abfrage wird beim Öffnen des Cursor ausgeführt.
- Die Ergebnisse einer Abfrage erhalten Sie, indem Zeilen aus einem Cursor gelesen werden.
- Jeder Cursor besitzt vier Attribute: %NOTFOUND, %FOUND, %ISOPEN und %ROWCOUNT.
- Ein PL/SQL-Unterprogramm kann gleichzeitig mehrere Cursors geöffnet haben.
- Ein Datenbank-Trigger ist ein PL/SQL-Block, der beim Eintreten eines Trigger-auslösenden Ereignisses - INSERT, UPDATE oder DELETE - ausgeführt wird.
- Sie definieren einen Trigger mit der CREATE-TRIGGER-Anweisung, die aus SQL\*Plus eingegeben werden kann.
- Sie können einen Trigger zu ganz unterschiedlichen Zwecken einsetzen, die die Durchsetzung von Firmenregeln, komplexe Validierungen von Spaltenwerten und Berechnung der Standardwerte von Spalten einschließen.
- Ein Trigger auf Zeilenebene wird für jede vom Trigger betroffenen Zeile ausgelöst.
- Ein Trigger auf Anweisungsebene wird nur einmal vom Trigger-auslösenden Ereignis ausgelöst.
- Ein BEFORE-Trigger wird vor seinem auslösenden Ereignis ausgelöst. Ein AFTER-Trigger wird nach

seinem auslösenden Ereignis ausgelöst.

- Ein Trigger kann eine optionale Klausel besitzen, die die Bedingung bestimmt, die zu seiner Auslösung erfüllt sein muß.
- Sie können einen Trigger mit der DROP-TRIGGER-Anweisung löschen.
- Sie können einen Trigger mit der ALTER-TRIGGER-Anweisung aktivieren oder deaktivieren.
- Der Rumpf eines Trigger kann keine COMMIT- oder ROLLBACK-Anweisung enthalten.
- Sie können eine gespeicherte Prozedur oder Funktion aus dem Trigger-Rumpf aufrufen.

## Wie geht es weiter?

Nachdem Sie nun mit SQL und PL/SQL vertraut sind, wird es Zeit zu etwas gänzlich anderem zu wechseln. An Tag 12, »Developer/2000: Einführung in Oracle Forms«, lernen Sie ein Werkzeug kennen, das Sie für die Erstellung von Client-Server-Anwendungen einsetzen.

## Fragen und Antworten

Frage:

Ist es möglich, einen Trigger zu erstellen, der ausgelöst wird, wenn während einer Abfrage eine Zeile gelesen wird?

Antwort:

*Nein, ein Trigger wird nur von **INSERT, UPDATE oder DELETE** ausgelöst. Sie können die Auditing-Möglichkeiten unter Oracle-RDBMS verwenden, um jede von einem Benutzer abgesetzte Abfrage aufzuzeichnen.*

Frage:

Worin liegt der Vorteil bei der Verwendung eines Trigger, um eine Bedingung zu überprüfen, die bereits durch die referentielle Integrität erzwungen wird?

Antwort:

*Einige Entity-Relationship-Entwurfswerkzeuge generieren scheinbar überflüssige Trigger. Das heißt, die generierten Trigger suchen nach Verstößen gegen die referentielle Integrität und geben bestimmte Fehlermeldungen zurück. Ohne einen solchen Trigger würde Oracle-RDBMS eine Fehlermeldung zurückgeben, daß die referentielle Integrität verletzt wurde, ohne aber die entsprechenden Spalten oder Beziehung anzugeben. Diese generierten Trigger erleichtern es sowohl dem Entwickler als auch dem Benutzer, festzustellen, warum der Vorgang nicht zugelassen wurde.*

Frage:



Ist es immer erforderlich, einen Cursor zu schließen?

Antwort:

*Es ist erforderlich, wenn Sie den Wert einer vom Cursor verwendeten Variablen ändern und wieder öffnen wollen. Im allgemeinen ist es eine gute Angewohnheit, einen nicht mehr verwendeten Cursor zu schließen.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Wie werden SQLCODE und SQLERRM in PL/SQL-Unterprogrammen verwendet?
2. Richtig oder falsch? Ein PL/SQL-Unterprogramm kann gleichzeitig mehrere Cursors geöffnet haben.
3. Welches sind die Vorteile bei der Verwendung von Datenbank-Triggern?
4. Welchen Datenbank-Trigger würden Sie verwenden, um den in einer Spalte zu speichernden Wert zu verändern, wenn der Tabelle eine neue Zeile hinzugefügt wird?

## Übungen

1. Erstellen Sie eine gespeicherte Funktion Teaching\_Load mit einem einzigen Argument - einer Abteilungsnummer (Department\_ID). Teaching\_Load soll die durchschnittliche Anzahl an Stunden zurückgeben, die Ausbilder in den entsprechenden Abteilungen zur Zeit unterrichten.
2. Erstellen Sie eine gespeicherte Prozedur Suitable\_Locations, mit einem einzigen Argument - Anzahl der Sitzplätze. Suitable\_Locations soll die ersten drei Gebäude und Räume ausdrucken, die die angegebene Anzahl Sitzplätze überschreiten.
3. Schreiben Sie einen Before-Delete-Trigger für die Tabelle Instructor, der eine Exception auslöst, wenn ein Ausbilder laut Plan eine Klasse unterrichtet.

❖Kapitel❖ Top Inhalt Index **SAMS** Kapitel❖

# 12

## Developer/2000: Einführung in Oracle Forms

[Was ist Developer/2000?](#)

[Erstellen eines einfachen Formulars mit dem Forms-Designer](#)

[Elemente eines Formulars](#)

[Ändern von Oracle-Komponenten mit Hilfe des Registrierungseditors](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

Bisher haben Sie schon sehr viel über den Einsatz von SQL zur Erstellung einer Oracle-Datenbank, Tabellenabfragen und Veränderungen des Inhalts von Tabellen gelernt. Außerdem haben Sie gelernt, PL/SQL zur Entwicklung von Anwendungslogik in Form von gespeicherten Prozeduren, Funktionen und Paketen einzusetzen - und wie Datenbank-Trigger für verschiedene Zwecke erstellt werden.

Jetzt ist es an der Zeit, mit Developer/2000 die Entwicklung von Oracle-Anwendungen zu erkunden. In dieser Lektion und an den Tagen 14 bis 17 lernen Sie, welche Rolle jede Komponente von Developer/2000 - Oracle Forms, Oracle Reports, Oracle Graphics und Procedure Builder - bei der Entwicklung einer Client-Server-Anwendung spielt.

### Was ist Developer/2000?

Developer/2000 ist eine Produktfamilie für den Entwurf und die Erstellung von Client-Server-Anwendungen. Diese Anwendungen können für Windows-, Mac- und UNIX-Motif-Umgebungen entwickelt werden. Zusätzlich können Sie auch zeichenorientierte Versionen Ihrer Anwendungen entwickeln. Dies sind die vier Hauptbestandteile von Developer/2000:

- **Oracle Forms:** Einzusetzen für den Entwurf und die Erstellung Forms-basierender Anwendungen. Diese Forms umfassen Dateneingabe-, Abfrage- und Browser-Formulare.
- **Oracle Reports:** Einzusetzen für den Entwurf einer Vielzahl von einfachen bis hin zu komplexen Berichten. Ein Bericht kann vor dem Ausdruck am Bildschirm betrachtet oder direkt zum Drucker gesendet werden.
- **Oracle Graphics:** Erlaubt Ihnen, Abfragen zu definieren, die bei ihrer Ausführung zur Laufzeit Geschäftsgrafiken erzeugen.
- **Procedure Builder:** Einzusetzen für den Entwurf und die Verwaltung von PL/SQL-Code am Datenbank-Server - gespeicherte Prozeduren, Funktionen und Paketen.

Die folgende Lektion befaßt sich mit Oracle Forms.

## Erstellen eines einfachen Formulars mit dem Forms-Designer

Lassen Sie uns direkt in Oracle Forms eintauchen. Dieser Abschnitt führt Sie über die Erstellung eines Formulars bis hin zur Anzeige und Veränderung von Informationen über Studenten.

- Um den Forms-Designer über die Start-Schaltfläche aufzurufen, wählen Sie Programme | Oracle Developer/2000 für Win95 | Forms Designer.
- Der Forms-Designer zeigt daraufhin den Object-Navigator an (siehe Abbildung 12.1).

Ehe Sie fortfahren, sollten Sie eine Verbindung zur Oracle-Datenbank herstellen.

### Eine Verbindung zu einer Oracle-Datenbank aufbauen

Um Forms-Designer mit einer Oracle-Datenbank zu verbinden, wählen Sie den Menüeintrag File | Connect. Es wird ein Dialogfeld angezeigt, in dem Sie nach dem Benutzernamen, Kennwort und der zu benutzenden Oracle-Datenbank für die Verbindung gefragt werden. Wenn Sie zur Entwicklung Personal Oracle verwenden, geben Sie folgendes ein:

- Benutzername **flugle**
- Kennwort **flugle**
- Datenbank (leer lassen)

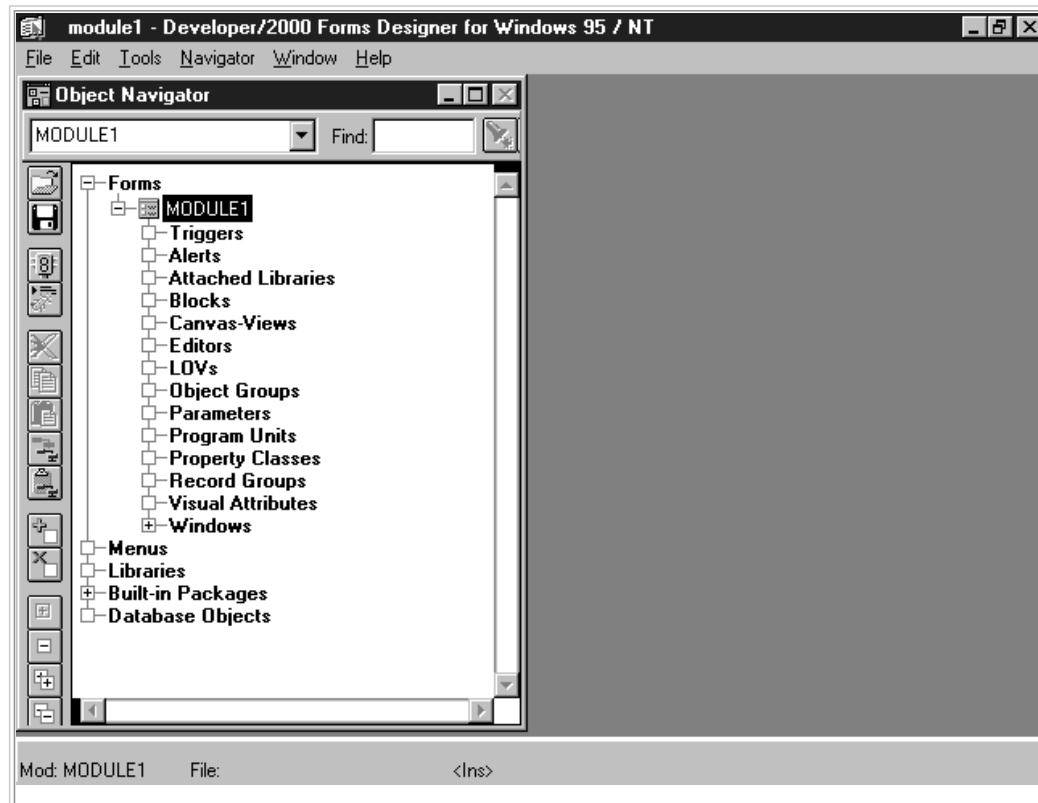


Abbildung 12.1:  
Der Forms-Designer zeigt  
zunächst den Object-  
Navigator an.

- 

Wenn Sie zur Entwicklung einen Oracle-Server einsetzen, machen Sie, wie in Abbildung 12.2 gezeigt, folgende Eingaben:

- Benutzername **flugle**
- Kennwort **flugle**
- Datenbank **flugle** (unter der Annahme, daß Sie bereits mit SQL\*Net Easy Configuration einen Datenbank-Alias namens flugle erstellt haben, welcher auf die Datenbank zeigt, in der Sie die Beispieltabellen geladen haben)

In beiden Fällen klicken Sie die Schaltfläche Connect an. Wenn keine Fehlermeldungen erscheinen, können Sie davon ausgehen, daß Sie sich erfolgreich mit der Datenbank verbunden haben.

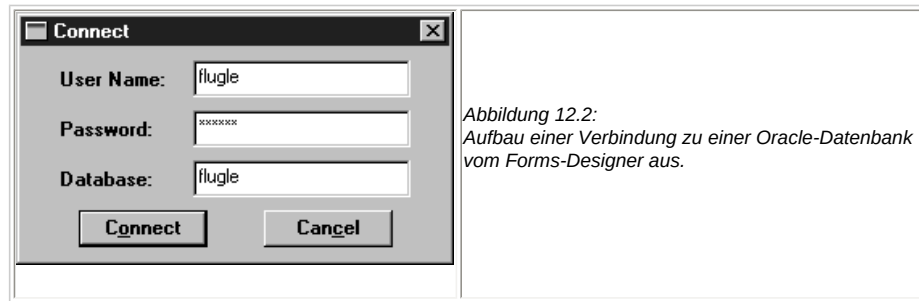


Abbildung 12.2:  
Aufbau einer Verbindung zu einer Oracle-Datenbank  
vom Forms-Designer aus.

## Erstellen eines neuen Blocks

Standardmäßig öffnet der Forms-Designer beim ersten Aufruf ein neues Formular mit dem Namen Module1. Es gibt drei Elemente, die in fast allen Formularen vorkommen:

- ein oder mehrere Blöcke
- eine Canvas-Ansicht
- ein Fenster

Fahren Sie fort, Ihr einfaches Formular durch Einrichten eines Blocks zu erstellen. Oracle Forms verwendet Blöcke, um in einem Formular Felder den Spalten einer Oracle-Tabelle zuzuordnen. Wählen Sie den Menüeintrag Tools | New Block. Ein Dialogfeld mit dem Titel New Block Options erscheint. In diesem Fenster stehen vier Register zur Verfügung: General, Items, Layout und Master/Detail. Beim Aufruf des Fensters wird das Register General angezeigt. Beachten Sie, daß das Feld Base Table <None> enthält (siehe Abbildung 12.3).

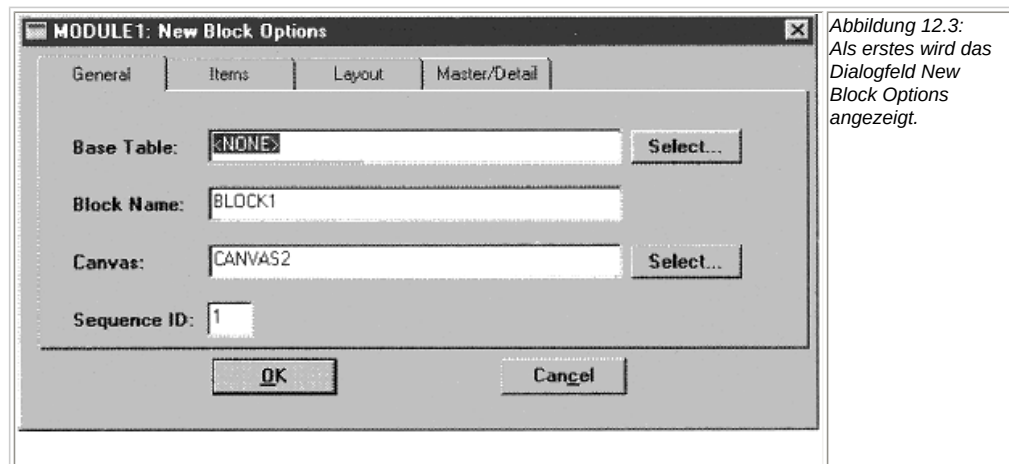


Abbildung 12.3:  
Als erstes wird das  
Dialogfeld New  
Block Options  
angezeigt.

Um die Student-Tabelle auszuwählen, betätigen Sie die Select-Schaltfläche rechts neben dem Feld Base Table. Der Forms-Designer zeigt ein weiteres Fenster an, das fünf Kontrollkästchen enthält, um die Ihnen zur Verfügung zu stellende Auswahl an Datenbankobjekten festzulegen. Standardmäßig sind zwei Kontrollkästchen ausgewählt - Current User und Tables (siehe Abbildung 12.4). Unter Beibehaltung dieser Standardwerte klicken Sie auf OK.

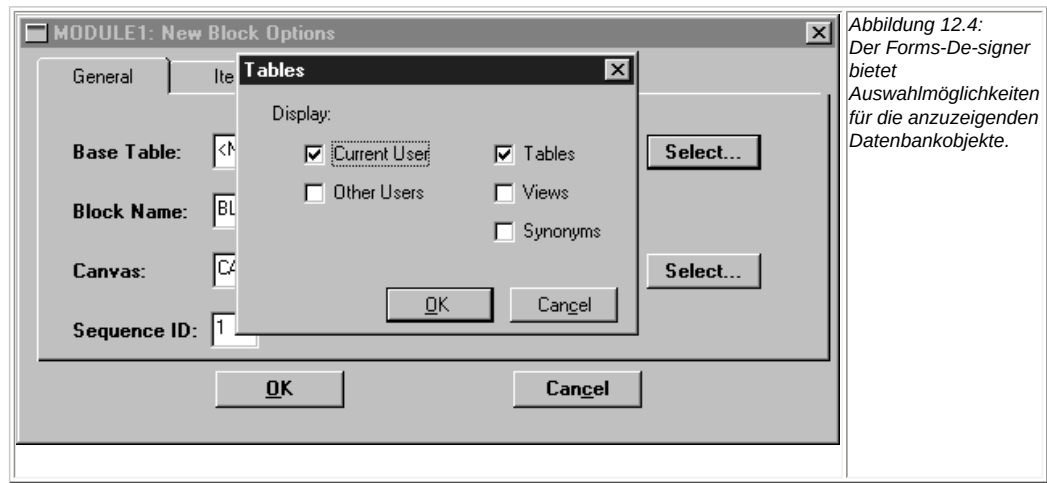


Abbildung 12.4:  
Der Forms-De-signer  
bietet  
Auswahlmöglichkeiten  
für die anzuzeigenden  
Datenbankobjekte.

Der Forms-Designer zeigt nun ein Fenster an, das eine Liste der Tabellen enthält, auf die flugle Rechte besitzt. Blättern Sie in dieser Liste nach unten, bis die Student-Tabelle erscheint. Wählen Sie diese mit Maus aus, und klicken Sie auf OK (siehe Abbildung 12.5).

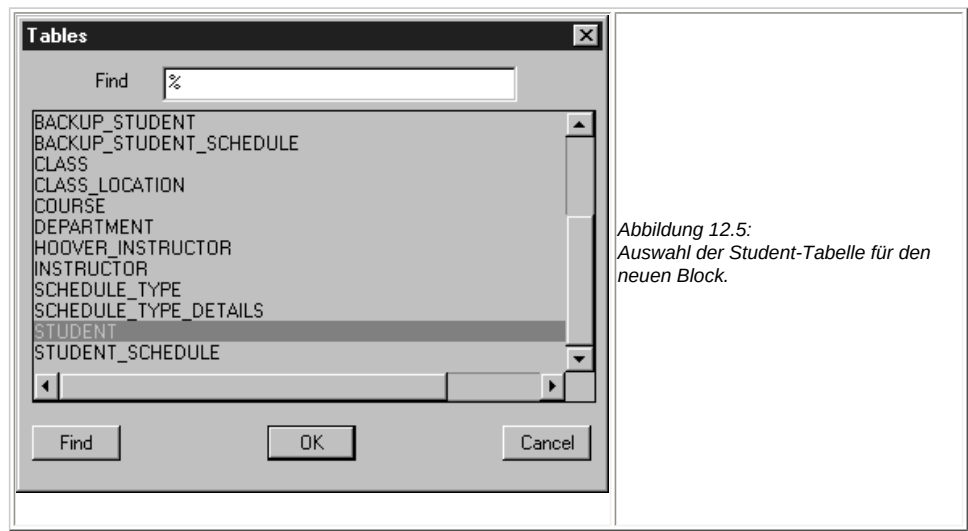
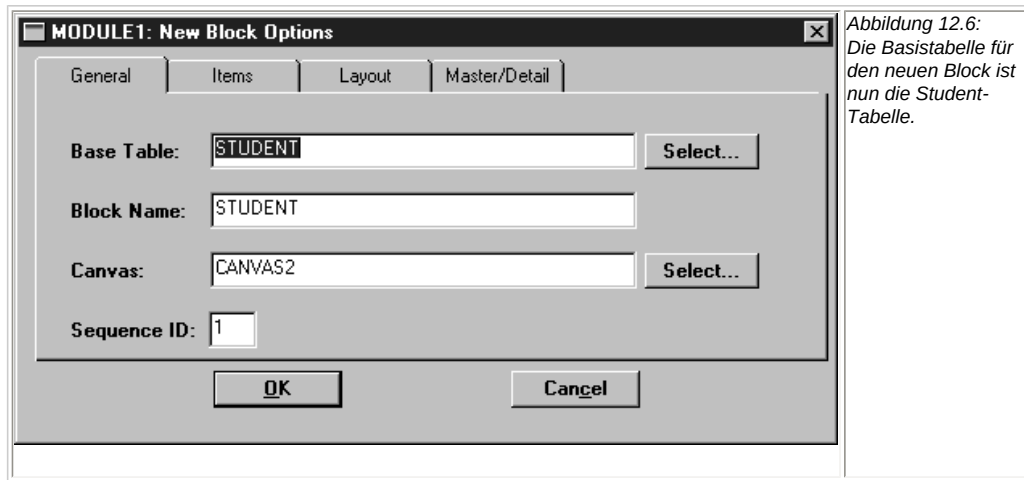


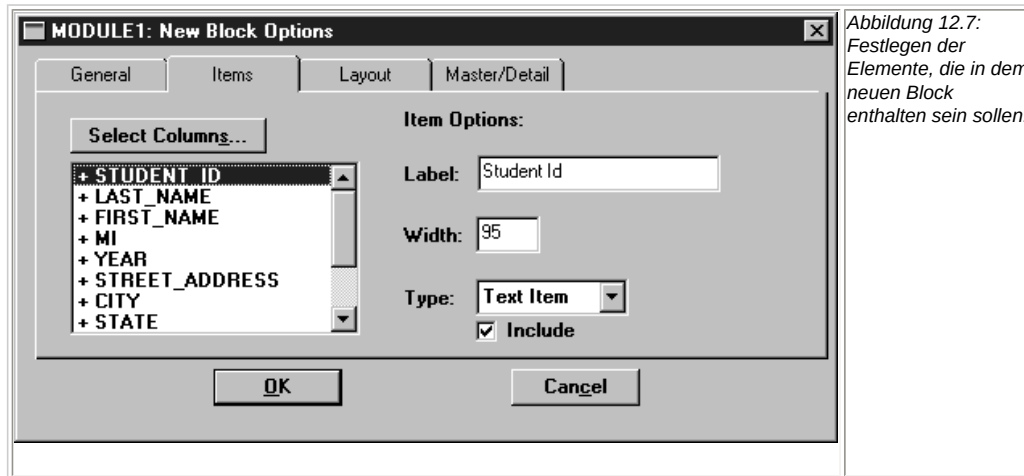
Abbildung 12.5:  
Auswahl der Student-Tabelle für den  
neuen Block.

Der Forms-Designer kehrt dann zum Fenster New Block Options zurück. Jetzt enthält das Feld Base Table jedoch die von Ihnen gewählte Tabelle - Student (siehe Abbildung 12.6).



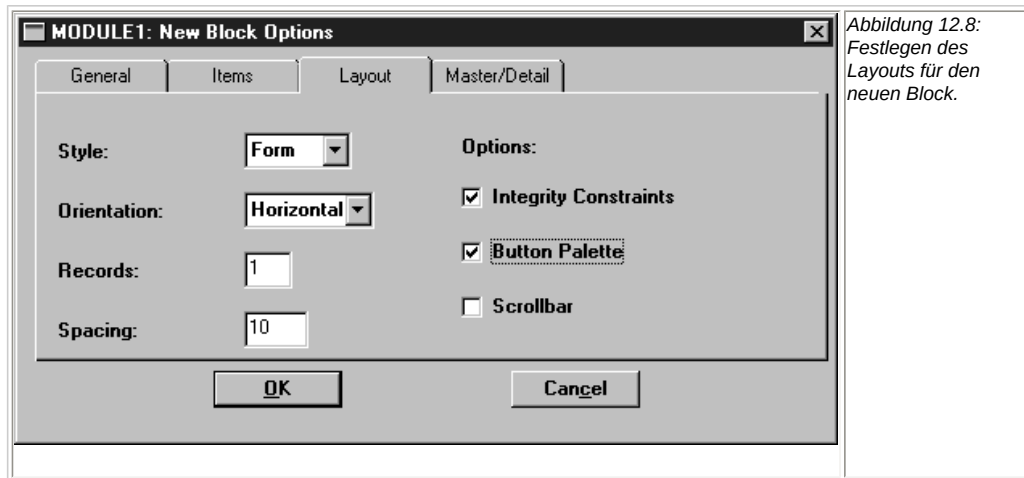
Schauen Sie sich nun die beiden weiteren Register dieses Fensters an. Wählen Sie das Register Items. Noch ist das Listenfeld neben der Select-Columns-Schaltfläche leer. Klicken Sie auf die Select-Columns-Schaltfläche. Durch die Standardeinstellung werden alle Spalten der ausgewählten Tabelle in das Formular übernommen (siehe Abbildung 12.7).

Es gibt noch einige wissenswerte Informationen zum Ordner des Items-Registers. Steht vor einer Spalte ein +, wird sie in das Formular übernommen; steht ein - davor, wird sie nicht übernommen. Ein Unterstrich im Namen einer Spalte wird für die Spaltenüberschrift in eine Leerstelle umgewandelt. Der Anfangsbuchstabe jedes Wortes für die Spaltenüberschrift wird großgeschrieben. Jedes Element ist von einem bestimmten Typ und jedes ausgewählte Element standardmäßig ein Textelement.

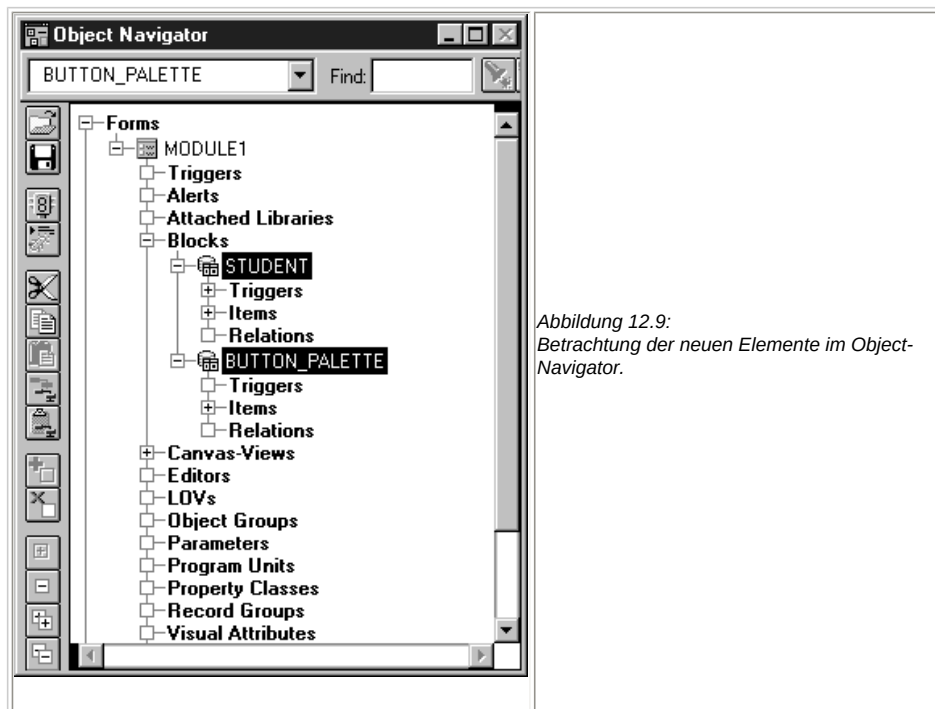


Schauen Sie als nächstes das Layout des neuen Blocks an. Wählen Sie das Layout-Register. Aus der Liste im Style-Feld können zwei Layoutstile, Tabular und Form angegeben werden; wählen Sie Form. Der Stil Form dient der Anzeige eines einzelnen Datensatzes. Von den beiden möglichen Ausrichtungen des neuen Blocks wählen Sie Horizontal.

Unter Options finden Sie drei Kontrollkästchen vor. Durch seine Anwahl generiert das Kontrollkästchen Integrity Constraints die notwendigen Trigger, um die für diese Tabelle geltenden Integritätsregeln für die Tabelle und Spalten zu erzwingen. Wählen Sie das Kontrollkästchen Integrity Constraints aus. Durch seine Anwahl generiert das Kontrollkästchen Button-Palette eine Reihe von Schaltflächen, die dazu dienen, im Formular durch Datensätze zu blättern oder die im Formular angezeigten Datensätze zu ändern. Wählen Sie das Kontrollkästchen Button-Palette aus. Wählen Sie aber nicht das Kontrollkästchen Scrollbar aus, weil dieser Block jeweils nur einen Datensatz anzeigt und somit ein Rollbalken nicht erforderlich ist. Abbildung 12.8 zeigt, wie Ihre Auswahl im Register Layout aussehen sollte.



Zum jetzigen Zeitpunkt können Sie das Register Master/Detail ignorieren; Sie werden es erst in der nächsten Lektion einsetzen. Um den neuen Block zu erstellen, wählen Sie OK. Der Forms-Designer kehrt nun zum Object Navigator zurück, in dem nun jedoch zwei Blöcke angezeigt werden sollten - Student und Button\_Palette (siehe Abbildung 12.9). Beachten Sie auch, daß links des Canvas-View-Elements ein + angezeigt wird und, wenn Sie nach unten blättern, werden Sie links des Window-Elements ebenfalls ein + sehen. Das + zeigt an, daß eine Canvas-Ansicht und ein Fenster automatisch erzeugt wurden, als Sie den Block Student erstellt haben.



### Testen des neuen Formulars

Das Formular kann nun getestet werden. Wählen Sie den Menüeintrag File | Run, oder klicken Sie in der Werkzeugleiste auf die grüne Ampel. Es wird ein Fenster mit dem Titel Developer/2000 Forms Runtime for Windows95/NT [WINDOW0] und dem neuen Formular angezeigt. Maximieren Sie beide Fenster (siehe Abbildung 12.10). Sie werden feststellen, daß oben im Fenster ein Standardmenü angezeigt wird.



Schauen Sie jetzt das Standardverhalten des Formulars an. Um eine Abfrage durchzuführen, wählen Sie den Menüeintrag Query | Execute. Das Formular fragt die Tabelle Student ab. Sie werden den ersten Datensatz sehen, der aus der Tabelle geliefert wird. In der linken unteren Fensterecke müßten Sie eine Bezeichnung Count: 1 sehen, die die Reihenfolge des soeben gelieferten Datensatzes angibt. Die Zeile in der diese Anzeige zu sehen ist, wird *Statuszeile* genannt.



Die Statuszeile ist (zur Laufzeit) der Bereich am unteren Rand des Formulars, in dem wichtige Informationen angezeigt werden, wie z.B. die Nummer des momentan angezeigten Datensatzes.



Die Meldungszeile erscheint oberhalb der Statuszeile und liefert dem Benutzer Rück- und Fehlermeldungen.

Die Zeile oberhalb der Statuszeile wird als Meldungszeile bezeichnet. Sie möchten z.B. wissen, wie viele Datensätze von der Abfrage des Tabellenblocks geliefert wurden und wählen den Menüeintrag Query | Count hits. Für die Tabelle Student zeigt die Meldungszeile an: FRM-40355: will retrieve 31 records.

Developer/2000 Forms Runtime for Windows 95 / NT - [WINDOW0]

Action Edit Block Field Record Query Window Help

STUDENT

Student Id

First Name

Year

City

Zipcode

Fax

Last Name

Mi

Street Address

State

Telephone

Email

<<

<

>

>>

Query

Save

Count: \*0

Abbildung 12.10:  
Testen des neuen  
Formulars.

Bewegen Sie sich mit der Tabulator-Taste zum nächsten Feld. Um z.B. vom Feld Student ID ins Feld Last Name zu wechseln, drücken Sie einmal die Tabulatortaste. Das Fenster rollt nach rechts, und das Feld wird hervorgehoben, wie es in Abbildung 12.11 zu sehen ist. Um zum vorherigen Feld zurückzukehren, drücken Sie die Umschalt- und Tabulatortaste gleichzeitig.



*Da Sie keine Reihenfolge angegeben haben, in der die Felder aus der Tabelle Student gelesen werden sollen, geschieht dies willkürlich - möglicherweise absteigend nach Student ID. Sie werden noch lernen, die Reihenfolge anzugeben, in der ein Block Felder einlesen soll.*

Durch ein Formular navigieren

Sie können sich in einem Formular auf zwei Wegen zum nächsten Datensatz bewegen. Entweder wählen Sie Record | Next oder klicken Sie auf > in der Button Palette. So oder so wird der nächste Datensatz im Formular angezeigt werden. Außerdem wird der in der linken unteren Ecke angezeigte Zähler erhöht - in diesem Fall auf 2. Ebenso können Sie die Tasten PFEIL-NACH-OBEN und PFEIL-NACH-UNTEN drücken, um sich in einem Formular durch den Datensatz zu bewegen. Die PFEIL-NACH-UNTEN-Taste zeigt den nächsten Datensatz und die PFEIL-NACH-OBEN-Taste den vorherigen Datensatz an. Versuchen Sie, sich vor den ersten Datensatz zu bewegen, zeigt die Meldungszeile an: FRM-40100: At first record. Versuchen Sie, sich über den letzten Datensatz hinaus zu bewegen, wird dementsprechend angezeigt: FRM-40352: Last record of query retrieved.

Developer/2000 Forms Runtime for Windows 95 / NT - [WINDOW0]

Action Edit Block Field Record Query Window Help

Last Name

SMYTHE

Mi

R

Street Address

144 OLIVE AVE.

State

CA

Telephone

8055554433

Email

JSMYTHE@FLUGLE.EDU

Save

Count: 1

Abbildung 12.11:  
Bewegen durch die  
Felder des neuen  
Formulars.

In der Statuszeile werden noch einige weitere Zeichen angezeigt. Existieren vor dem angezeigten Datensatz weitere Datensätze, zeigt die Statuszeile ein ^ an. Existieren nach dem angezeigten Datensatz weitere Datensätze, zeigt die Statuszeile ein · an. Nehmen Sie z.B. an, daß eine Abfrage 10 Datensätze zurückgeben würde und das Formular den ersten Datensatz anzeigen würde, stünde in der Statuszeile nur ein ·. Nachdem Sie die PFEIL-NACH-UNTEN-Taste

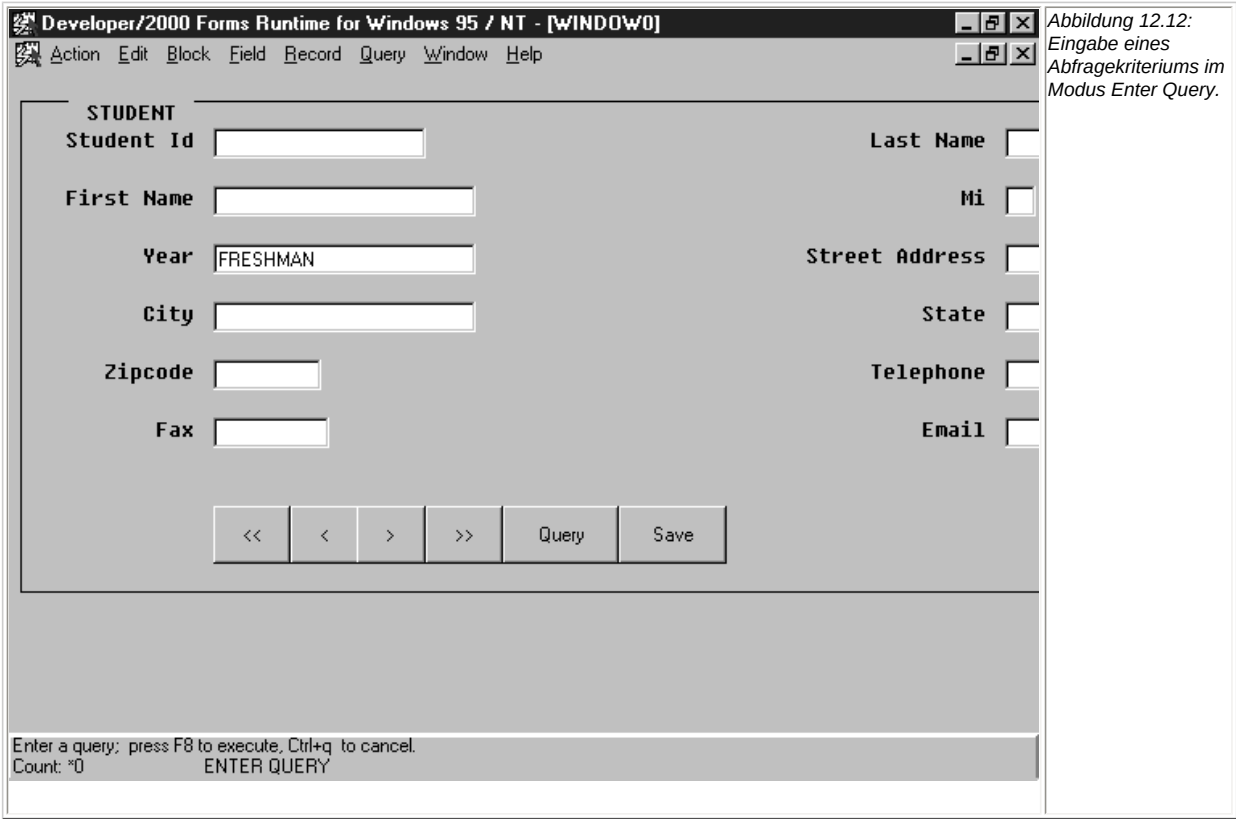
gedrückt haben, um zum nächsten Datensatz zu gelangen, zeigt die Statuszeile ein ^ und ein · an, da es sowohl vor als auch nach dem aktuellen Datensatz einen weiteren gibt. Sind sie zum letzten Datensatz gelangt, zeigt die Statuszeile ein ^ an.

Abfrageverhalten von Oracle Forms

Oracle Forms stellt einige grundlegenden Abfragemöglichkeiten zur Verfügung, die vom Entwickler keine weitere Programmierung erfordern. Ein Formular kann eine Abfrage in zwei Schritten durchführen. Der erste Schritt setzt voraus, daß für das Formular der ENTER-QUERY-Modus eingestellt ist. In diesem Modus kann der Benutzer in ein oder mehrere Felder des Formulars Suchkriterien eingeben. Mit dem nächsten Schritt führt das Formular die Abfrage aus, wobei die Suchkriterien verwendet werden. Schauen Sie sich einige Beispiele anhand des Formulars Student an.

Sie wechseln in den ENTER-QUERY-Modus, entweder durch die Auswahl des Menüeintrags Query | Enter, Drücken der (F7)-Taste oder Anklicken der Query-Schaltfläche in der Button-Palette. Sie können erkennen, ob sich das Formular im ENTER-QUERY-Modus befindet - in der Meldungszeile steht: Enter a query; press F8 to execute, Ctrl+q to cancel und in der Statuszeile: Zähler: +0 ENTER QUERY.

Bewegen Sie sich mit der Tabulatortaste in das Feld Year, und geben Sie FRESHMAN ein (siehe Abbildung 12.12). Um die Abfrage auszuführen, wählen Sie entweder den Menüeintrag Query | Execute, drücken (F8), oder klicken erneut auf die Query-Schaltfläche.



Schauen Sie sich ein Beispiel an, bei dem in mehr als einem Feld ein Suchkriterium eingegeben wird. Wechseln Sie in den ENTER-QUERY-Modus, entweder durch die Auswahl des Menüeintrags Query | Enter, Drücken der (F7)-Taste oder Anklicken der Query-Schaltfläche. Geben Sie im Feld Year SENIOR und im Feld City SPRINGFIELD ein. Abbildung 12.13 zeigt, was bei Ausführung der Abfrage angezeigt werden sollte. In der Statuszeile wird Ihnen auffallen, daß vor dem Zähler ein Sternchen angezeigt wird. Dies bedeutet, daß die Suchkriterien nur auf einen Studenten zutreffen.

Im ENTER-QUERY-Modus können Sie auch % als Platzhalter in einem Feld verwenden. Wechseln Sie für die Ausführung einer Beispielabfrage in den Abfragemodus. Geben Sie im Feld Last Name L%M% ein, und führen Sie die Abfrage durch. Dieses Suchkriterium wird alle Studenten ausgeben, deren Nachname mit L beginnt und ebenfalls den Buchstaben M enthält. Diese Abfrage gibt zwei Studenten mit den Nachnamen MALLARD und MICHAELS aus.

Eine weitere leistungsfähige Eigenschaft von Oracle Forms sind die Einsatzmöglichkeiten der logischen Operatoren >, >=, <, <= und !. Wenn Sie z.B. alle Freshmen finden, die nicht in DOVER wohnen und deren Vornamen mit dem Buchstaben G oder höher beginnen, würden Sie > G im Feld First Name, FRESHMEN im Feld Year und != 'DOVER' im Feld City eingeben. Nach Ausführung der Abfrage müßten fünf Datensätze abgerufen werden - Freshmen mit den Vornamen Paula, Henry, Linda, Richard und Ivan, die alle in Springfield wohnen.

Wenn Sie sich im ENTER-QUERY-Modus befinden und mit der Abfrage nicht fortfahren wollen, können Sie diesen Modus verlassen, indem Sie entweder den Menüeintrag QUERY | Cancel wählen, oder Strg+q drücken. Möchten Sie nur die Anzahl der Datensätze wissen, die eine Abfrage abrufen würde, ohne die Datensätze sehen zu wollen, erreichen Sie dies mit dem Befehl Count Hits.

Developer/2000 Forms Runtime for Windows 95 / NT - [WINDOW0]

Action Edit Block Field Record Query Window Help

STUDENT

Student Id10231301

Last NameHARRIS

First NameCOREY

MiY

YearSENIOR

Street Address938

CitySPRINGFIELD

StateCA

Zipcode900004321

Telephone805

Fax

EmailCHARRIS@SPR

<<

<

>

>>

Query

Save

Count: \*1

Abbildung 12.13:  
Ein einzelner  
Datensatz, der von  
einer Abfrage  
zurückgeliefert  
wurde, bei der in  
zwei Feldern  
Abfragekriterien  
eingegeben wurden.

So möchten Sie z.B. wissen, wie viele Sophomores es gibt. Stellen Sie für das Formular den ENTER-QUERY-Modus ein und geben Sie in das Feld Year SOPHOMORE ein. Die Anzahl der Datensätze können Sie ermitteln, indem Sie den Menüeintrag Query | Count Hits wählen oder (Umschalt)+(F2) drücken. In der Meldungszeile wird die Anzahl der Datensätze angezeigt, die den Abfragekriterien entsprechen.

Ändern eines Datensatzes mit Oracle Forms

Ein Formular befindet immer in einem der beiden Modi, NORMAL oder ENTER QUERY. Sie können keine Änderungen an Datensätzen vornehmen, wenn sich das Formular im ENTER-QUERY-Modus befindet. Nur im NORMAL-Modus können Sie einen Datensatz ändern, löschen oder neu anlegen. Wenn Sie einen vorhandenen Datensatz ändern wollen, bewegen Sie sich zum entsprechenden Datensatz und Feld, das Sie ändern möchten. Möchten Sie z.B. den Vornamen von Peter Pinkwater in Pierre ändern, fragen Sie die Tabelle ab, um seinen Datensatz abzurufen und geben Sie im Feld First Name seinen neuen Vornamen ein.

Um einen neuen Datensatz anzulegen, wählen Sie den Menüeintrag Records | Insert oder drücken Sie (F6). Nehmen Sie als Beispiel das gerade von Ihnen erstellte Formular, um der Student-Tabelle einen neuen Studenten hinzuzufügen. Nachdem Sie den Menüeintrag Records | Insert gewählt oder (F6) gedrückt haben, steht die Schreibmarke im ersten Feld des Datensatzes - Student ID. Geben Sie folgende Werte in die Felder ein:

Student ID 99991234

Last Name **THOMAS**  
First Name **KAREN**  
MI **G**  
Year **FRESHMAN**  
Street Address **133 SEQUOIA DR.**  
City **SPRINGFIELD**  
State **CA**  
Zipcode **900004321**  
Telephone **8055551291**  
Fax **<NULL>**  
[Email KTHOMAS@FLUGLE.EDU](mailto:KTHOMAS@FLUGLE.EDU)

Um den soeben erstellten Datensatz zu speichern, stehen Ihnen mehrere Möglichkeiten zur Auswahl bereit:

- Sie können den Menüeintrag Action | Save wählen.
- Sie können (F10) drücken.
- Sie könne solange warten, bis Sie eine weitere Abfrage durchführen oder das Formular verlassen. Sie werden dann in beiden Fällen gefragt, ob die vorgenommenen Änderungen gespeichert sollen (siehe Abbildung 12.14).

Developer/2000 Forms Runtime for Windows 95 / NT - [WINDOW0]

Action Edit Block Field Record Query Window Help

STUDENT

Student Id99991234

First NameKAREN

YearFRESHMAN

City

Zipcode

Fax

Last NameTH

MiG

Street Address133

StateCA

Telephone805

EmailKTH

Forms

Do you want to save the changes you have made?

YesNoCancel

<<<>>>

Query

Save

Press Ctrl+Tab to move between choices, Enter to select.  
Count: \*31 ^v

Abbildung 12.14:  
Der Benutzer wird  
gefragt, ob die  
Änderungen  
gespeichert werden  
sollen.



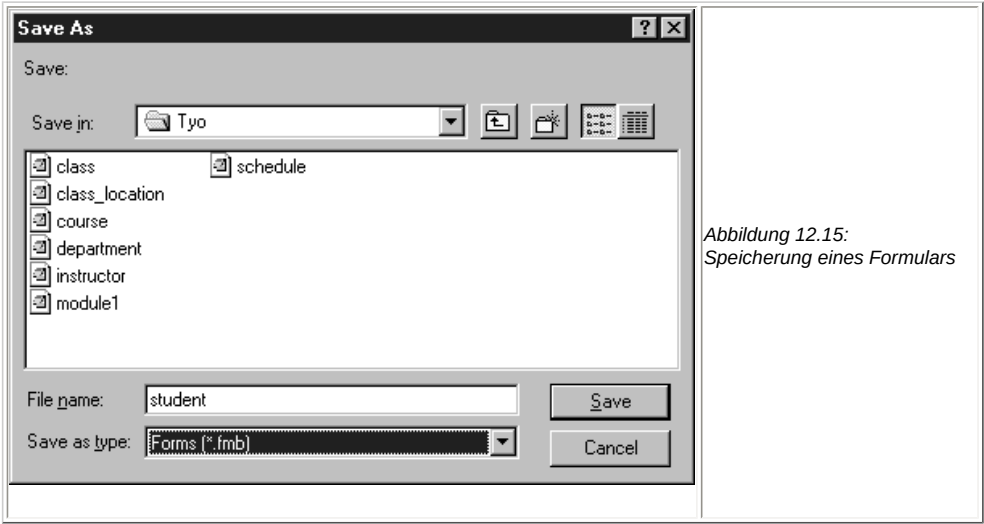
In der Standardeinstellung speichert Oracle vom Benutzer vorgenommene Änderungen nicht automatisch. Im Hintergrund nutzt Oracle Forms Oracles Fähigkeit, Transaktionen zu verwalten. Wenn Sie Ihre Änderungen an

Datensätzen in einem Formular speichern, führt Oracle Forms für Sie eine COMMIT-Anweisung aus. Geben Sie an, keine Änderungen speichern zu wollen, führt Oracle Forms eine ROLLBACK-Anweisung für Sie aus. Machen Sie sich klar, daß Oracle Forms alle Änderungen seit dem Beginn der Sitzung oder dem letzten Speichern rückgängig macht.

Um einen Datensatz zu löschen, gehen Sie zu dem entsprechenden Datensatz und wählen den Menüeintrag Record | Remove. Es wird der vorstehende Datensatz angezeigt. Der Datensatz wird erst wirklich aus der Tabelle entfernt, wenn Sie einen Speichervorgang durchführen.

Speichern eines Formulars

Nachdem Sie nun eine Einführung in das Verhalten von Formularen erhalten haben, verlassen wir das Formular und kehren zum Forms-Designer zurück. Um das von Ihnen erstellte Formular zu speichern, wählen Sie den Menüeintrag im Forms-Designer File | Save. Standardmäßig speichert Forms-Designer das Formular im Verzeichnis FORMS45 unter dem Stammverzeichnis von Oracle. Abbildung 12.15 zeigt das vom Forms-Designer verwendete Fenster zum Speichern der Datei. Im Feld »Save as type« wählen Sie in der Liste FORMS (\*.fmb) aus, geben Sie als Dateiname student ein und klicken Sie auf Save. Beachten Sie, daß der Object Navigator das Formular unter dem Namen STUDENT anzeigt.



Es ist nicht ratsam, anwendungsspezifische Dateien im Verzeichnisbaum von Oracle zu speichern. Sie verlieren dann u.U. die Übersicht, welche Dateien zu welcher Anwendung gehören. Sie sollten statt dessen ein separates Verzeichnis einrichten, in dem Sie Ihre Anwendungsdateien speichern.

Elemente von Oracle Forms

Oracle Forms besteht aus drei Werkzeugen - Forms-Designer, Forms-Generator und Forms-Runtime. Forms-Designer ermöglicht Ihnen das Erstellen und Verändern von drei Modultypen: Formulare, Menüs und Bibliotheken. Die meisten Forms-Anwendungen bestehen in erster Linie aus Formularen mit einem oder mehreren Menüs und manchmal einer oder mehreren Bibliotheken.

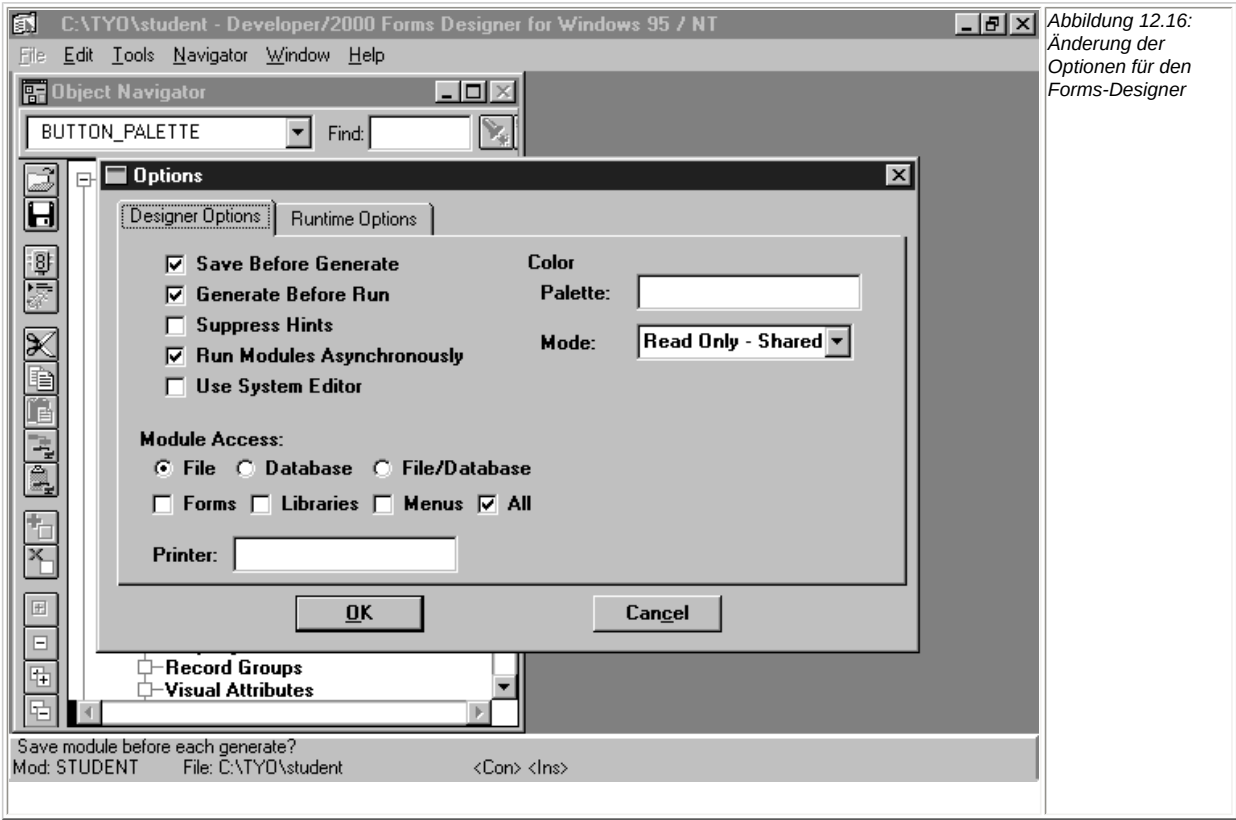
Die von Ihnen entworfenen Module werden vom Forms-Designer standardmäßig entweder in einer Datei oder Oracle-Datenbank gespeichert. Um ein Modul in einer Datenbank zu speichern, müssen Sie oder Ihr DBA das von Developer/2000 benötigte Datenbank-Schema laden. Hierüber erfahren Sie mehr in den Administrative Notes des Developer/2000 Admin for Win95.

Das Speichern Ihrer Anwendungsmodule in einer Oracle-Datenbank hat Vor- und Nachteile. Wenn Sie mit anderen, geographisch voneinander getrennten Entwicklern zusammenarbeiten, kann es sinnvoll sein, eine Oracle-Datenbank als gemeinsames Repository für Ihre Anwendungsmodule einzusetzen. Wenn Sie jedoch ein Versionskontrollpaket wie PVCS, zur Verwaltung der einzelnen Versionen Ihrer Formulare, Berichte und Grafiken, einsetzen möchten, wollen Sie Ihre Anwendungsmodule wahrscheinlich in Dateien speichern. Wenn dem so ist, wollen Sie wahrscheinlich die Standardeinstellungen ändern, damit Forms-Designer Module aus einer Datei öffnet oder in eine Datei speichert. Wählen Sie dazu den Menüeintrag Tools | Options. Im Options-Fenster ändern Sie das Optionsfeld Module-Access von File/Database zu File (siehe Abbildung 12.16). Außerdem sollten Sie das Kontrollkästchen Save Before Generate markieren - der Vorgang der Formulargenerierung wird bald erläutert.

Forms-Designer erlaubt Ihnen, mehrere Module gleichzeitig geöffnet zu haben. So können Sie z.B. fünf Formulare, ein Menü und eine Bibliothek zur gleichen Zeit geöffnet haben.

Formulare

Wenn Sie das Dateisystem zum Öffnen und Speichern Ihrer Formulare einsetzen, verwendet Forms-Designer die Dateierweiterung .fmb. Eine .fmb-Datei ist eine Binärdatei mit einer komplexen Beschreibung des Formulars. Da Sie eine .fmb-Datei aber nicht ausführen können, müssen Sie eine Laufzeitversion der Datei mit der Endung .fmX erzeugen, wozu Forms-Generator die .fmb-Datei verwendet. Wenn Sie ein Formular verwenden, benutzen Sie zum Ausführen einer .fmX-Datei das Programm Forms-Generator. Außerdem können Sie eine binäre .fmb-Datei in eine .fmt-Datei umwandeln, die dann eine Textdatei ist. Um Änderungen an .fmt-Dateien zu verwalten, können Sie ein Versionskontrollsystem einsetzen. Dateien vom Typ .fmb und .fmt lassen sich portieren - Sie können diese von einer Umgebung in eine andere übertragen und die .fmX-Datei generieren.



Menüs

Ein Menü wird vom Forms-Designer in eine Datei mit der Endung .mmb gespeichert, die dann beim Erzeugen des Menüs zur Generierung der .mmb-Datei verwendet wird. Benutzen Sie ein Formular, das ein Menü besitzt, führt Forms-Runtime die .mmX-Datei aus. Sie können auch eine binäre .mmb-Datei in eine .mmt-Datei umwandeln, die dann eine Textdatei ist. Änderungen an .mmt-Dateien können Sie mit einem Versionskontrollsystem verwalten.

## Bibliotheken

Eine Bibliothek besteht aus PL/SQL-Prozeduren, Funktionen oder Paketen. Forms-Designer speichert eine Bibliothek in eine Datei mit der Endung .pll, die dann beim Erzeugen der Bibliothek zur Generierung der .plx-Datei verwendet wird. Benutzen Sie ein Formular, dem eine Bibliothek zugeordnet ist, führt Forms-Runtime den in der .plx-Datei enthaltenen Code aus. Sie können auch eine binäre .pll-Datei in eine .pld-Datei umwandeln, die dann eine Textdatei ist. Änderungen an .pld-Dateien können Sie mit einem Versionskontrollsystem verwalten.

## Elemente eines Formulars

Ein Formular wird durch eine Sammlung von Objekten definiert. Einige dieser Objekte sind zwingend erforderlich, andere sind optional. Ein Objekt kann folgendes sein:

- Block
- Canvas-Ansicht
- Fenster
- Trigger
- Werteliste (LOV)
- Datensatzgruppe
- Warnung
- Parameter

Zu einem späteren Zeitpunkt dieses Kapitels schauen Sie sich jedes dieser Objekte genauer an. Zuvor wird im folgenden Abschnitt die Verwendung des Object Navigators untersucht.

### Mit dem Object Navigator arbeiten

Der Object Navigator stellt eine intuitive und bequeme Schnittstelle zur Erstellung und Bearbeitung von Formularen, Menüs und Bibliotheken dar. Der Object Navigator zeigt Objekte in hierarchischer Form an. Für jeden Modultyp - Formular, Menü oder Bibliothek - zeigt er eine Reihe von Knoten unterhalb eines jeden Moduls an. So werden z.B. unter jedem Formular die folgenden Knoten angezeigt: Trigger, Warnungen, zugeordnete Bibliotheken, Blöcke, Canvas-Ansichten, Editoren, LOVs, Objektgruppen, Parameter, Programmeinheiten und Fenster. Entweder mit der Maus oder den NACH-OBEN- und NACH-UNTEN-Pfeiltasten wechseln Sie zwischen den Knoten.

Links von jedem Knoten befindet sich ein Kästchen. Ist dieses Kästchen leer, gibt es für das Formular keine Objekte dieses Knotentyps. Steht in dem Kästchen ein +, besitzt das Formular Objekte dieses Knotentyps. Das + zeigt an, daß die Anzeige dieses Knotens erweitert werden kann -- zur Zeit erscheint es ausgeblendet. Es gibt verschiedene Möglichkeiten, die Anzeige des Knotens zu erweitern:

- Sie können das Kästchen anklicken.
- Sie können den Menüeintrag Navigator | Expand wählen.

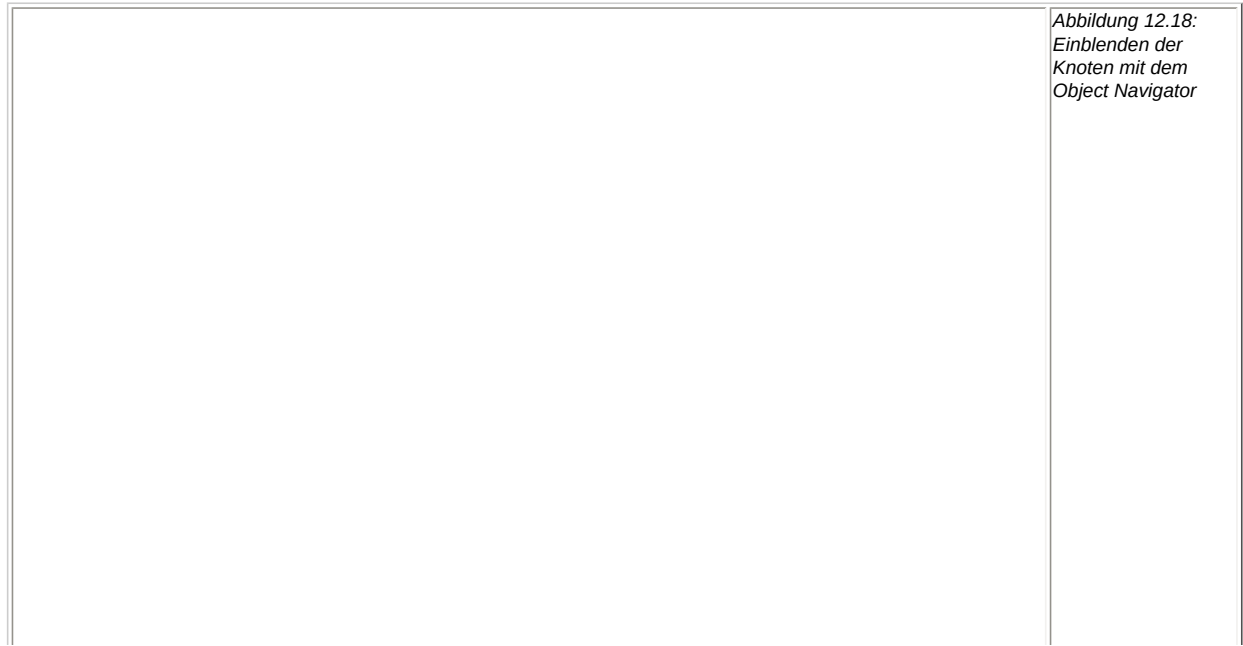
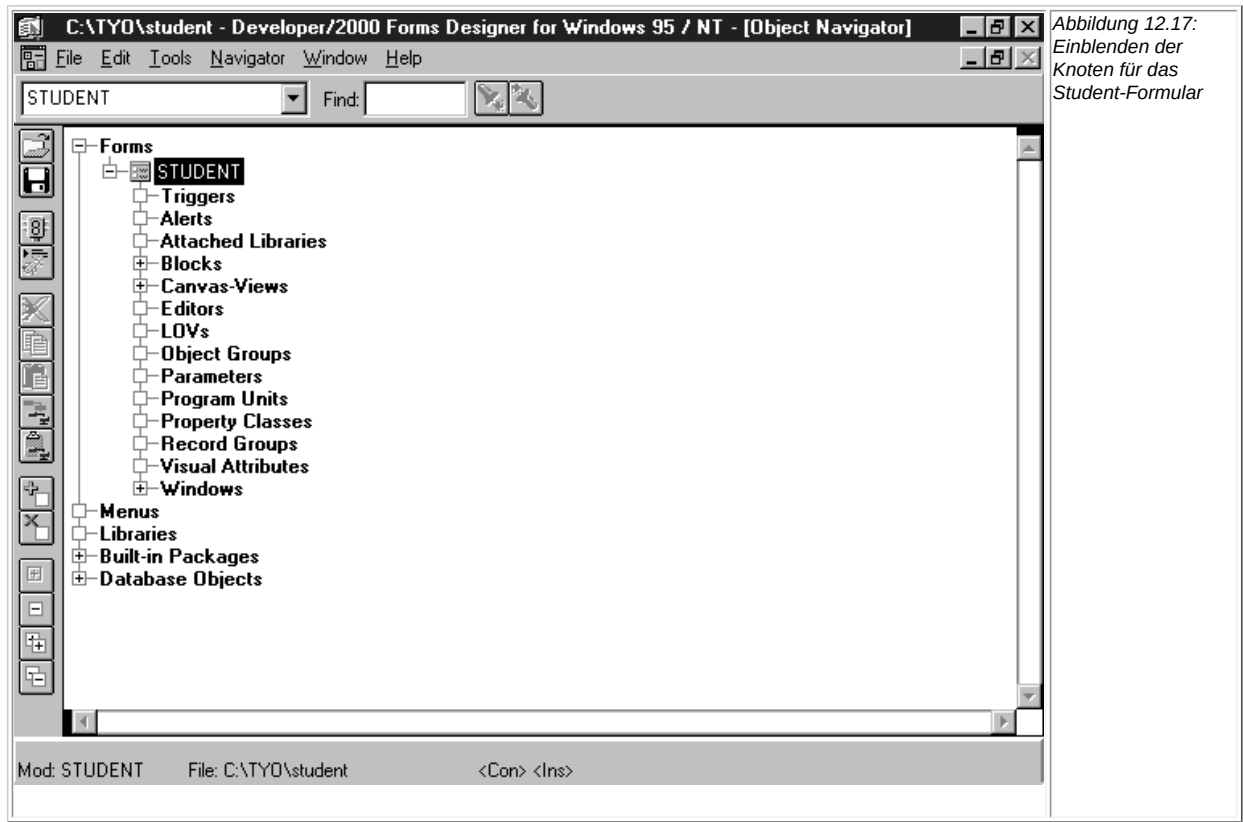
Wird in dem Kästchen ein - angezeigt, existieren Objekte dieses Knotentyps, die aber bereits vollständig zu sehen sind. Es gibt verschieden Möglichkeiten, die Anzeige des Knotens auszublenden:

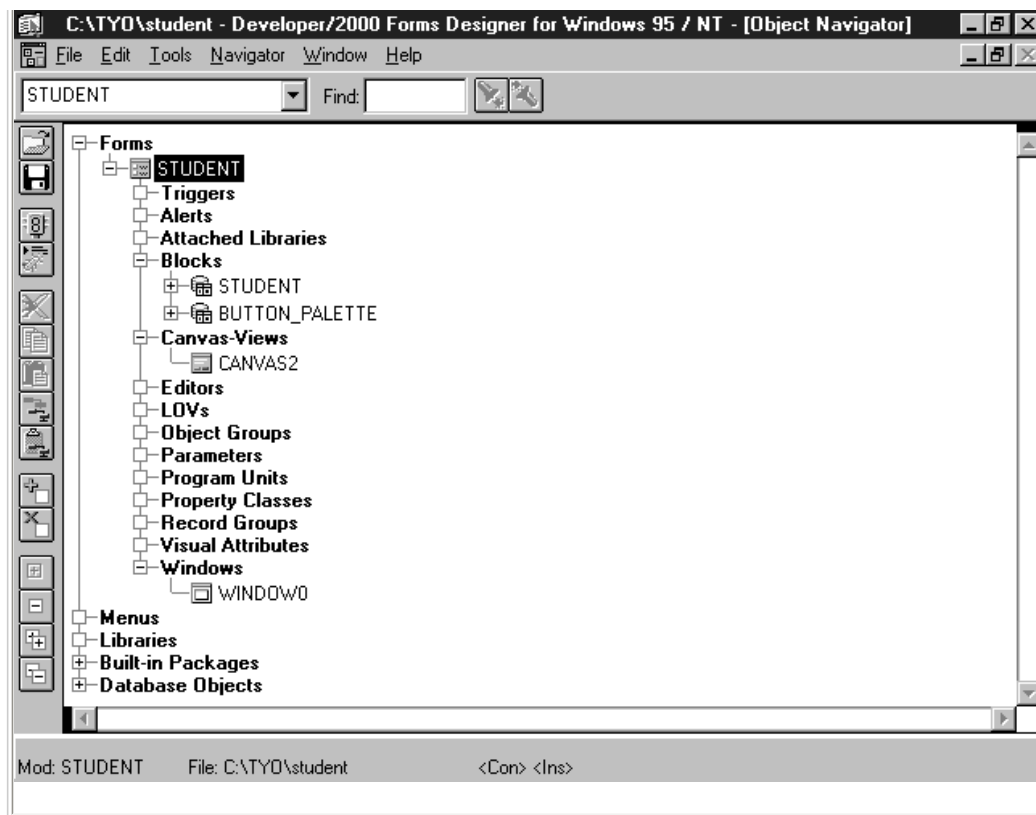
- Sie können das Kästchen anklicken, um den Knoten auszublenden.
- Sie können den Menüeintrag Navigator | Collapse wählen.

Nehmen wir das Formular Student, das drei Knoten mit Objekten enthält: Blöcke, Canvas-Ansichten und Fenster (siehe Abbildung 12.17).

Um jeden dieser drei Knoten einzublenden, klicken auf das Kästchen links von jedem Knoten (siehe Abbildung 12.18). Wie Sie sehen, steht vor den Blöcken Student und Button\_Palette jeweils ein + in den Kästchen, was bedeutet, daß diese nochmals erweitert werden können.







Jedes dieser Objekte besitzt eine Reihe von Eigenschaften. Die Eigenschaften eines bestimmten Objekts lassen Sie sich auf einem der folgenden Wege anzeigen:

- Klicken Sie mit der rechten Maustaste und dann auf Properties.
- Wählen Sie den Menüeintrag Tools | Properties.
- Klicken Sie auf das links des Objektnamens angezeigte Symbol.

Um z.B. die Eigenschaften des Blocks Student zu sehen, bewegen Sie sich mit der Maus oder den NACH-OBEN- und NACH-UNTEN-Pfeiltasten und klicken Sie mit der rechten Maustaste und dann auf Properties (siehe Abbildung 12.19).

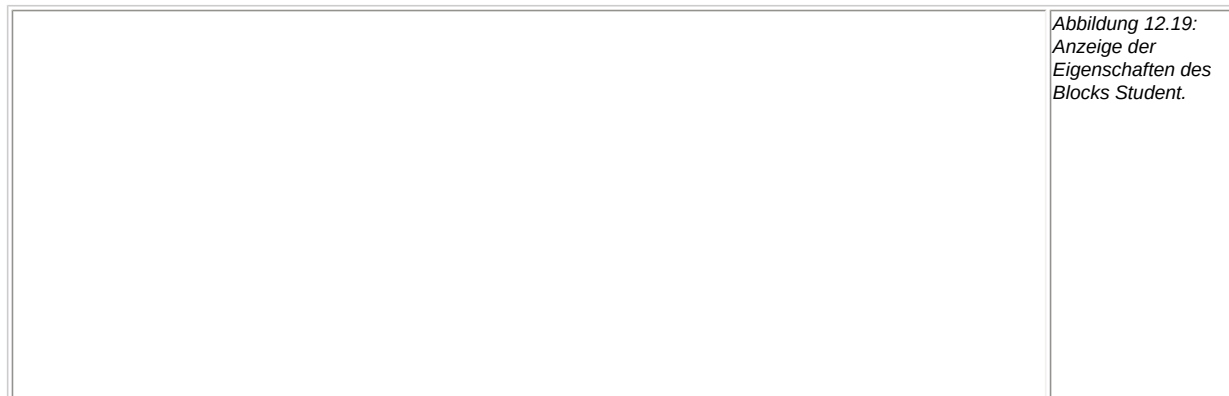
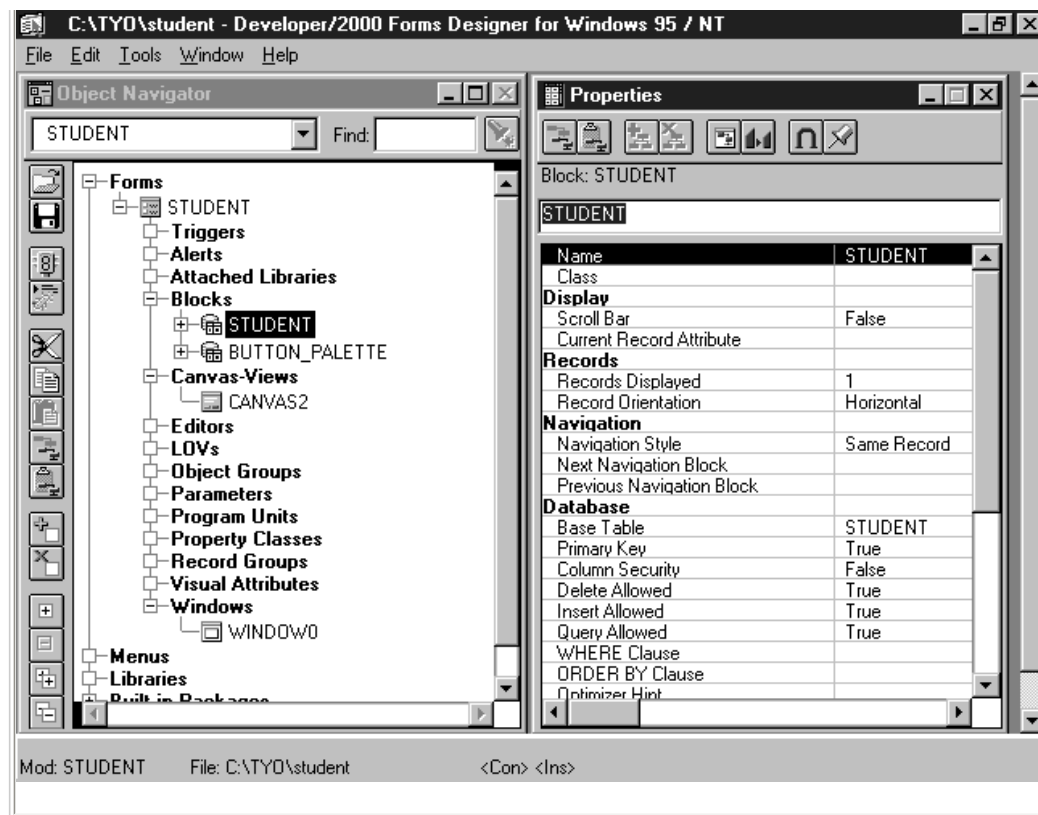


Abbildung 12.19:  
Anzeige der  
Eigenschaften des  
Blocks Student.



## Objekteigenschaften

Im Object Navigator können Sie die Eigenschaften eines Objekts einsehen und ändern. Dazu klicken Sie mit der linken Maustaste auf den Objektknoten, wählen das Objekt mit der Maus an und den Menüeintrag Tools | Properties oder Sie klicken mit der rechten Maustaste und dann auf Properties.

Welche Eigenschaften für ein Objekt angezeigt werden, hängt von seinem Typ ab. Auf den nächsten Seiten sehen Sie einige Beispiele von Objekteigenschaften.

## Blöcke

Fast jedes Formular enthält mindestens einen Block. Es gibt zwei Arten von Blöcken: *Basistabellenblöcke* und *Steuerblöcke*. Ein Basistabellenblock hängt mit einer bestimmten Tabelle oder Datensicht zusammen, während ein Steuerblock nicht mit einer bestimmten Tabelle oder Datensicht zusammenhängt. Fast jeder Block besitzt ein oder mehrere Elemente. Mindestens ein Element eines Basistabellenblocks hängt mit einer Spalte der Tabelle oder Datensicht des Blocks zusammen.



Ein Basistabellenblock ist ein Block von Oracle Forms, der mit einer Datenbanktabelle oder Datensicht zusammenhängt.



Ein Steuerblock ist ein Block von Oracle Forms, der nicht mit einer Datenbanktabelle zusammenhängt und normalerweise Steuerobjekte wie Schaltflächen enthält.

Zu den Blockeigenschaften gelangen Sie, indem Sie mit der rechten Maustaste und dann auf Properties klicken. Es wird ein Fenster mit den Eigenschaften des Blocks angezeigt. Möchten Sie z.B. eine ORDER-BY-Klausel für den Block Student bestimmen, bewegen Sie sich mit der Pfeiltaste zur Eigenschaft ORDER-BY-Klausel. Sie können die ORDER-BY-Klausel oben im Eigenschaftfenster eingeben und die Schaltfläche mit dem gelben Ballon rechts neben dem Feld anklicken. Möchten Sie z.B. die Datensätze im Block Student erst nach dem Jahr und dann nach den Namen der Studenten anordnen, geben Sie die in Abbildung 12.20 gezeigte ORDER-BY-Klausel ein und drücken Sie die Eingabetaste, um der Eigenschaft den Wert zuzuweisen.

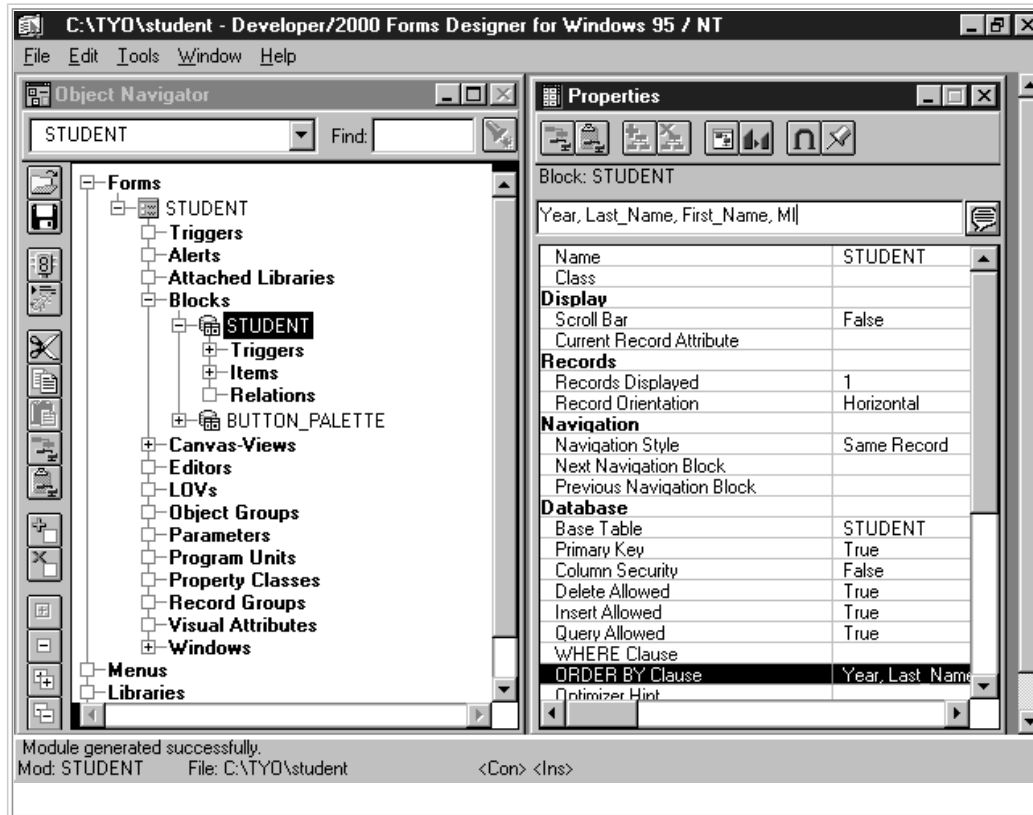


Abbildung 12.20:  
Setzen der  
Blockeigenschaft  
ORDER BY.

## Basistabellenblöcke

Ein Basistabellenblock enthält Elemente, die den Spalten einer Tabelle oder Datensicht zugeordnet sind. Jedes Element hat eine Reihe eigener Eigenschaften, auf die Sie zugreifen, indem Sie das Element mit der rechten Maustaste anwählen und dann auf Properties klicken. Eine wichtige Eigenschaft ist Item Type. Um den Typ eines Elements zu ändern, wählen Sie die Eigenschaft Item Type und dann oben im Fenster Properties den gewünschten Typ aus der Pop-up-Liste, wie in Abbildung 12.21 gezeigt, aus.

Eine ganze Reihe weiterer Elementeeigenschaften beeinflussen das Erscheinungsbild und Verhalten von Oracle-Forms-Anwendungen. Einige dieser Elementeeigenschaften steuern, was vom Benutzer in diesem Element eingegeben wird. So ist z.B. die Standardeinstellung für Case Restriction auf Mixed gesetzt. Um dies zu ändern, wählen Sie das Element, und klicken Sie darauf, bis sich die Eigenschaft in Upper geändert hat.

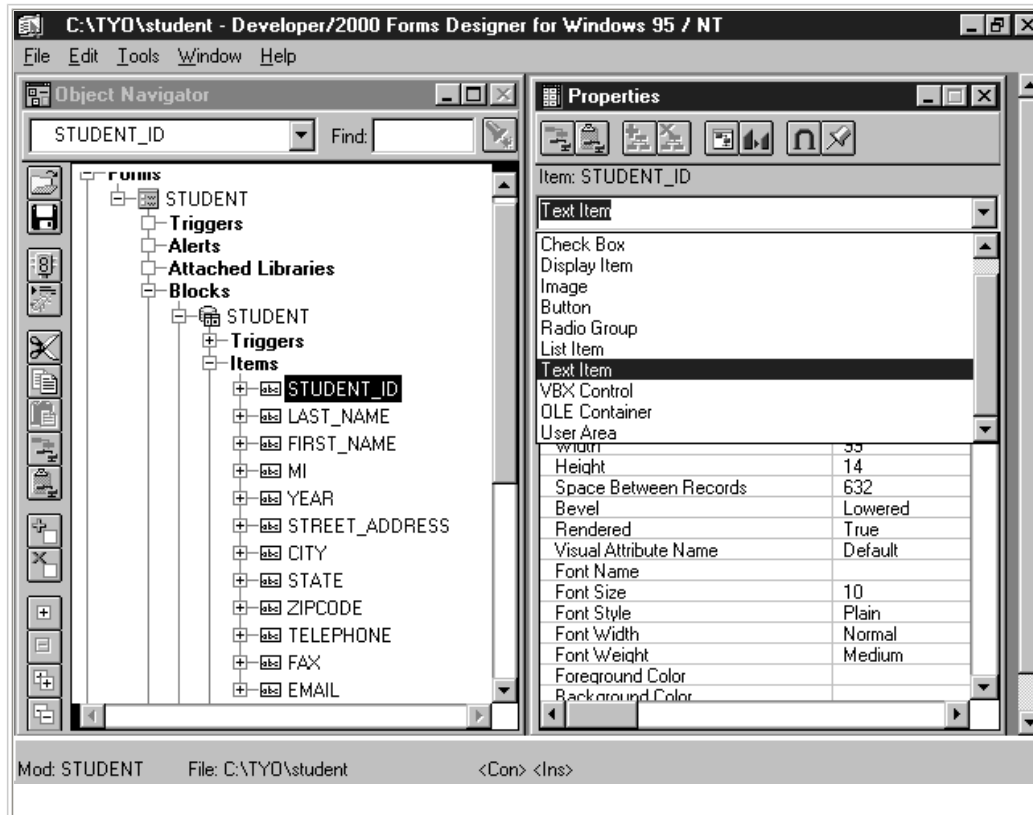


Abbildung 12.21:  
Setzen eines  
Elementtyps

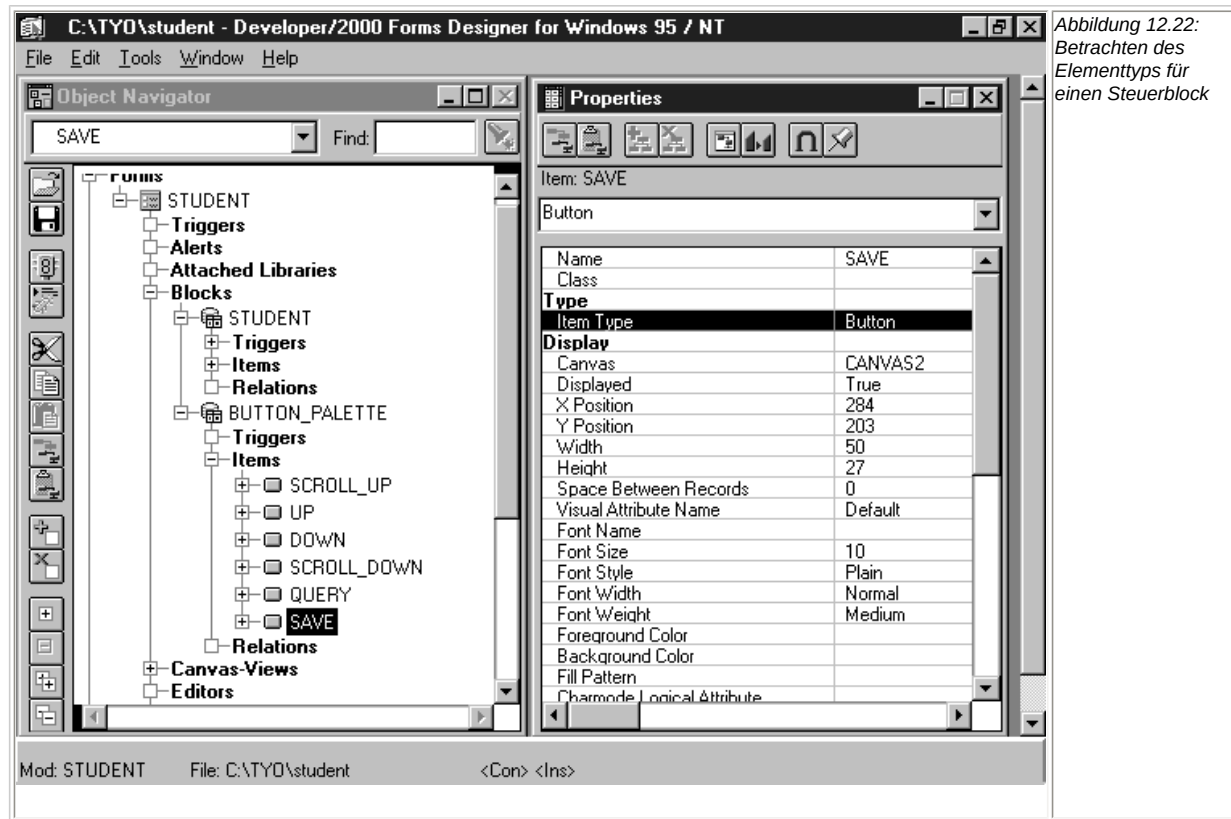


Besitzt eine Elementeigenschaft eine Reihe vordefinierter Werte, können Sie den Wert der gewählten Eigenschaft durch wiederholtes Drücken der Eingabetaste oder Anklicken mit der Maus ändern. Eine Eigenschaft hat eine Reihe vordefinierter Werte, wenn oben im Fenster Properties rechts neben dem Feld eine Pop-up-Liste zu sehen ist.

## Steuerblöcke

Damit Sie den Unterschied zwischen einem Basistabellenblockelement und einem Steuerblockelement sehen können, schauen Sie sich die Eigenschaften des Elements Save im Steuerblock BUTTON\_PALETTE an. Erstens beachten Sie, daß das Element vom Typ Button ist (siehe Abbildung 12.22) und zweitens, ungleich einem Basistabellenblockelement, besitzt ein Steuerblockelement keine mit der Datenbank zusammenhängenden Eigenschaften.

Vorausgesetzt, daß das Element Save eine Schaltfläche ist, muß diese bei ihrer Aktivierung natürlich irgendeine Aktion ausführen. Wenn Sie die Anzeige für das Element Save im Object Navigator erweitern, sehen Sie, daß zu diesem Element ein Trigger namens WHEN-BUTTON-PRESSED gehört. Viele Steuerblockelemente sind Schaltflächen, die das Verhalten anderer Blöcke eines Formulars »kontrollieren«.



## Canvas-Ansichten

Ein Canvas ist ein Hintergrundobjekt, auf dem Blockelemente platziert sind. Jedes Canvas besitzt eine Ansicht, in der definiert ist, wie es durch ein Fenster betrachtet wird. Mit dem Layout Editor können Sie die Elemente auf dem Canvas bearbeiten. Wählen Sie das zu ändernde Canvas aus, und klicken Sie mit der rechten Maustaste und dann auf den Layout Editor. Beim Erstellen eines neuen Blocks erzeugt Forms-Designer eine Canvas-Ansicht, wenn noch keine besteht, und platziert die Blockelemente darauf. Abbildung 12.23 zeigt das Standardlayout des Formulars Student.

## Fenster

Ein Fenster ist ein leerer Rahmen, der über der Canvas-Ansicht liegt. Jedes Formular hat mindestens ein Fenster, aber es kann mehrere enthalten. Abbildung 12.24 zeigt das Fenster Properties für das Standardfenster WINDOW0, das zusammen mit dem Block Student erzeugt wurde. Einige der Eigenschaften, die typischerweise verändert werden, sind die Breite, Höhe und Überschrift.

## Trigger

Ein Trigger ist eine oder mehrere PL/SQL-Anweisungen, die beim Auftreten der Trigger-auslösenden Aktion ausgeführt werden. Ein Trigger kann mit vielen unterschiedlichen Ebenen der Hierarchie von Formularobjekten verbunden sein. So können Sie z.B. einen Trigger definieren, der beim Aktivieren eines Formulars ausgelöst wird. Sie können einen Trigger definieren, der vor der Ausführung einer Abfrage in einem Block ausgelöst wird. Oder Sie definieren einen Trigger, der ausgelöst wird, wenn sich der Inhalt eines Elements ändert. Es gibt eine ganze Reihe vordefinierter Trigger-Ereignisse. Abbildung 12.25 zeigt die Liste der Trigger-Ereignisse, die angezeigt wird, wenn Sie die Pop-up-Liste für die Eigenschaft des Trigger Name auswählen.

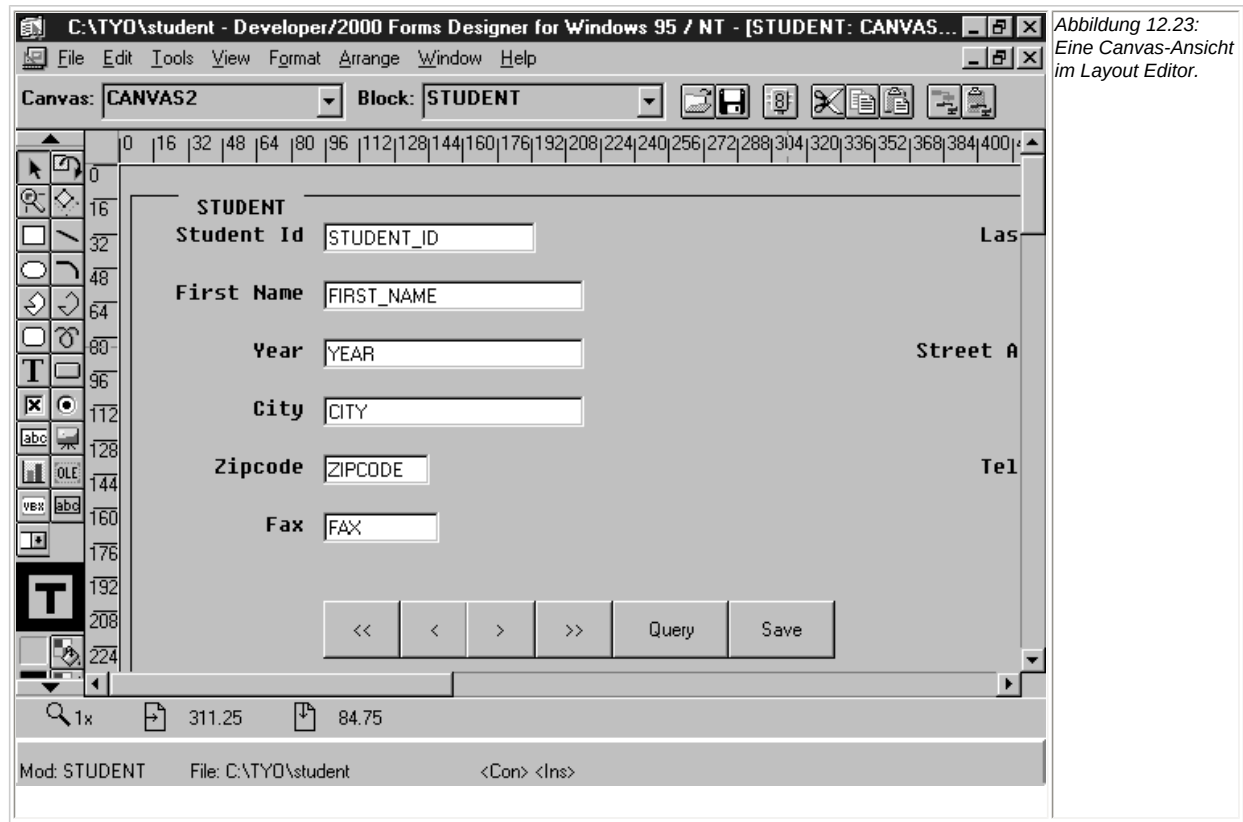


Abbildung 12.23:  
Eine Canvas-Ansicht  
im Layout Editor.

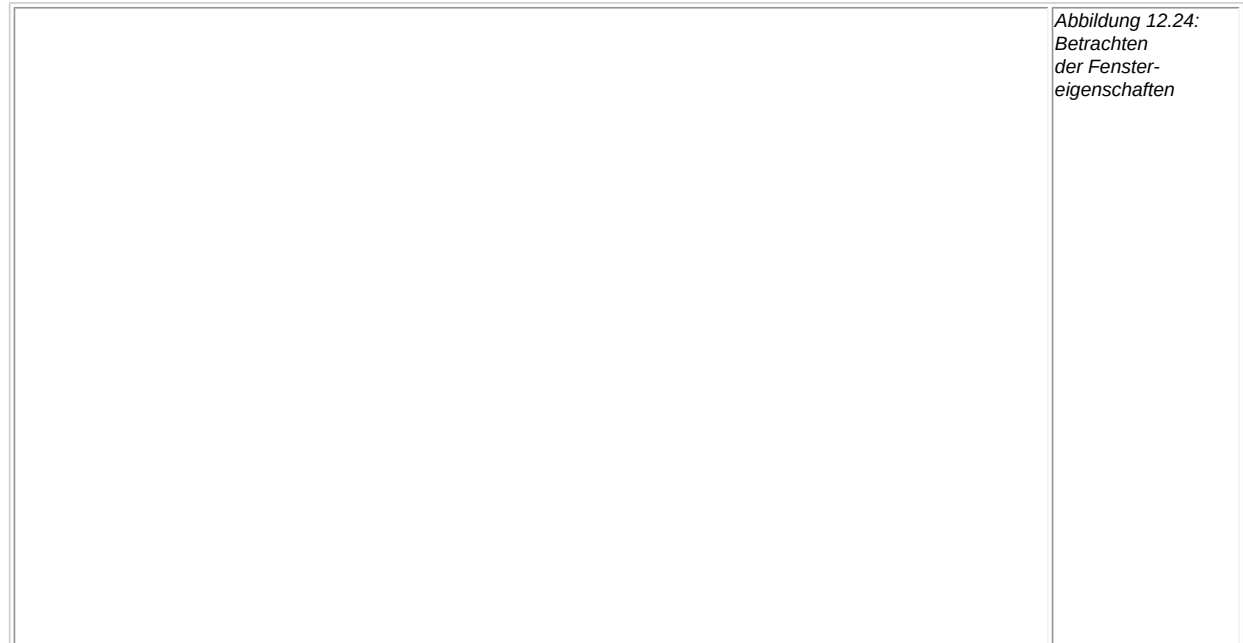


Abbildung 12.24:  
Betrachten  
der Fenster-  
eigenschaften

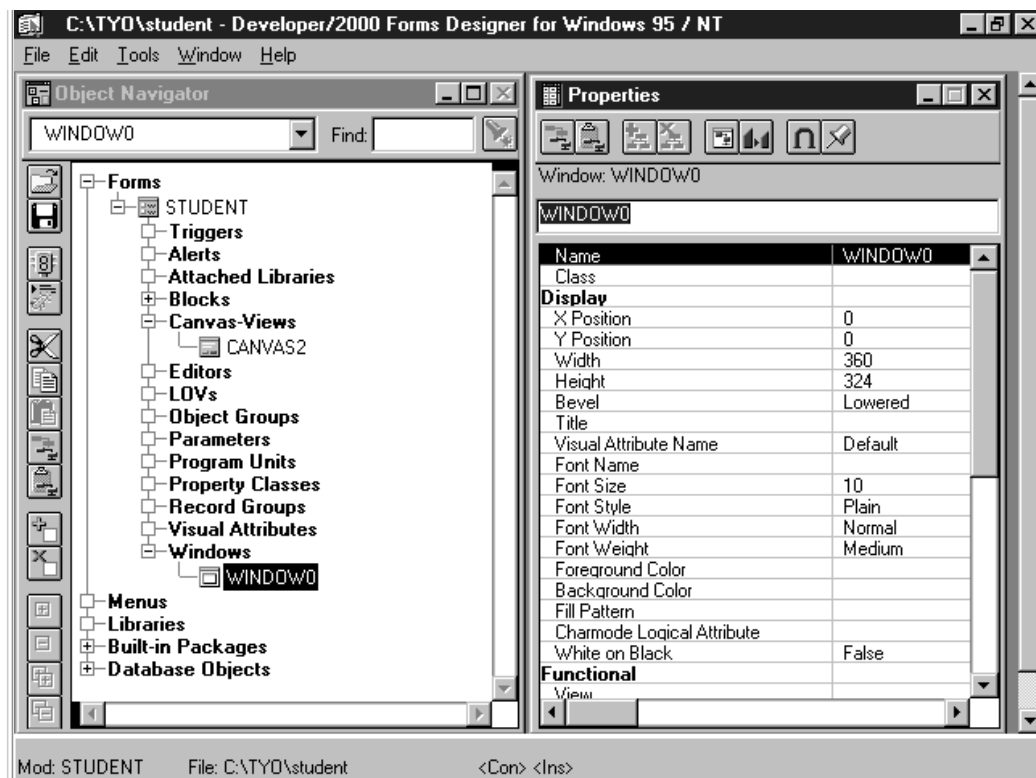
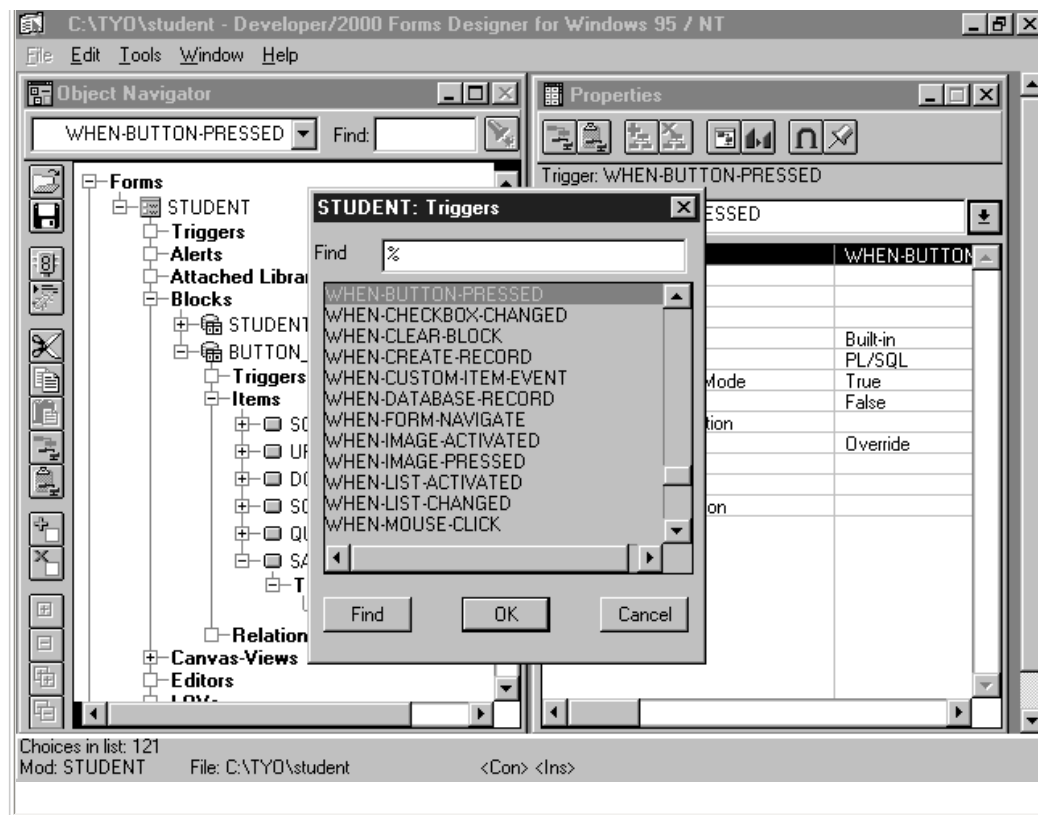


Abbildung 12.25:  
Einen Trigger  
auslösende  
Ereignisse





## Datensatzgruppen

Eine Datensatzgruppe ist eine Gruppe von Werten, die definiert werden kann:

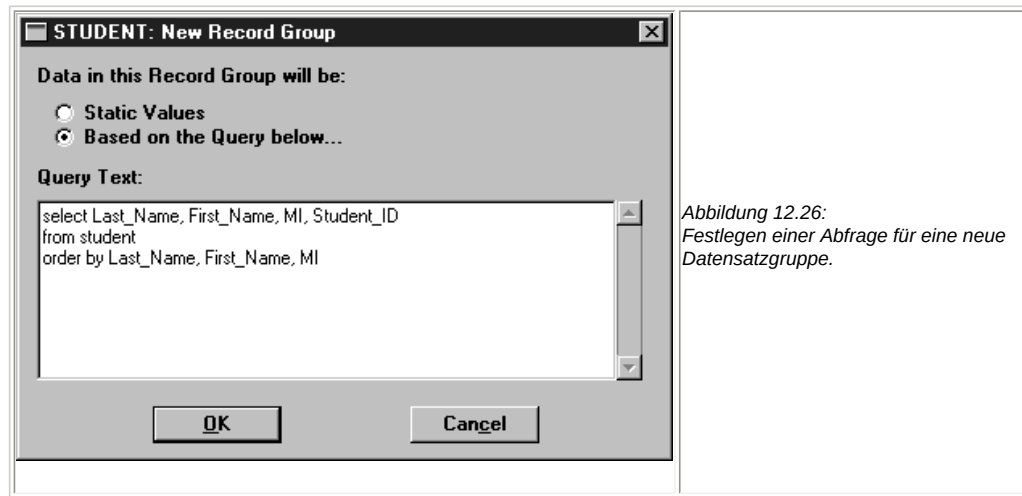
- Dynamisch, durch eine SELECT-Anweisung, die zur Laufzeit ausgeführt wird.
- Statisch, durch eine Gruppe statischer Werte.
- Programmgesteuert, durch ein PL/SQL-Unterprogramm, das zur Laufzeit eine SELECT-Anweisung oder eine Gruppe von Werten aufbaut.

Nehmen Sie z.B. an, daß Sie basierend auf den bestehenden Studenten-IDs der Tabelle Student eine Datensatzgruppe erzeugen wollen. Dazu verwenden Sie den Object Navigator, um den Knoten Record Groups im Formular STUDENT auszuwählen und das + in der vertikalen Werkzeugleiste an der linken Fensterseite anzuklicken. Es wird ein Dialogfenster angezeigt, in das Sie die Abfrage eingeben, auf der die neue Datensatzgruppe basieren soll (siehe Abbildung 12.26).

Nachdem Sie Ihre Abfrage eingegeben haben, klicken Sie auf OK und die Datensatzgruppe wird erzeugt. Eine Datensatzgruppe wird typischerweise von einer Werteliste (LOV) verwendet.

## Wertelisten

Eine Werteliste (LOV) ist ein auf einer Datensatzgruppe basierendes Objekt. Eine LOV wird üblicherweise zur Anzeige einer Liste mit möglichen Werten eingesetzt, aus der der Benutzer auswählen kann. Mit einer LOV ist eine Reihe von Blockelementeigenschaften verbunden. So können Sie z.B. auch festlegen, ob eine LOV auch zur Validierung einer Eingabe durch den Benutzer verwendet werden soll. Damit wird sichergestellt, daß der Benutzer keinen Wert in ein Feld eingeben kann, wenn dieser nicht in der angegebenen LOV vorkommt. An Tag 13, »Developer/2000: Entwicklung einer Benutzeroberfläche mit Hilfe«, und an Tag 14, »Developer/2000: Anwendungsentwicklung mit Hilfe von Oracle Forms«, lernen Sie mehr über die Verwendung von LOVs.



## Warnungen

Eine Warnung ist ein Fenster, das den Benutzer über ein Ereignis informiert. Dieses ist ein modales Fenster - ehe er fortfahren kann, muß der Benutzer eine der beiden Schaltflächen wählen.

## Parameter

Sie können für ein Formular Parameter definieren, die aus einem anderen Formular gesetzt werden können. Unter Verwendung eines Parameters können Sie einen Trigger erzeugen, der den Wert eines Parameters überprüft, und der, abhängig vom Vorhandensein oder Fehlen seines Werts, eine bestimmte Aktion auslöst. Nehmen Sie an, Sie wollen, daß es möglich ist, ein Formular aus einem anderen heraus aufzurufen. Sie möchten dem Formular Student eine Student-ID übergeben und den entsprechenden Datensatz des Studenten angezeigt bekommen. Dazu würden Sie einen Parameter Student\_ID vom Typ CHAR definieren. Dann würden Sie einen Trigger schreiben, der beim Aktivieren des Fensters ausgelöst würde und der den Datensatz, entsprechend der im Parameter enthaltenen Student-ID, abrufen würde, vorausgesetzt der Parameter enthält einen Wert.

## Ändern von Oracle-Komponenten mit Hilfe des Registrierungseditors

Es ist ratsam, Ihre gesamten Laufzeitformulare, Berichte und Grafikdateien in einem speziell für Ihre Anwendung eingerichteten Verzeichnis zu speichern. Sie in irgendeinem der Oracle-Verzeichnisse zu speichern, ist aus den folgenden Gründen hingegen keine gute Idee:

- Da im Verzeichnisbaum von Oracle (Orawin95) sehr viele Verzeichnisse liegen, kann es schnell geschehen, daß eine Datei falsch abgelegt wird.
- Bei der Installation einer neuen Version der Oracle-Werkzeuge auf Ihrem PC, können Ihre Dateien entfernt werden.

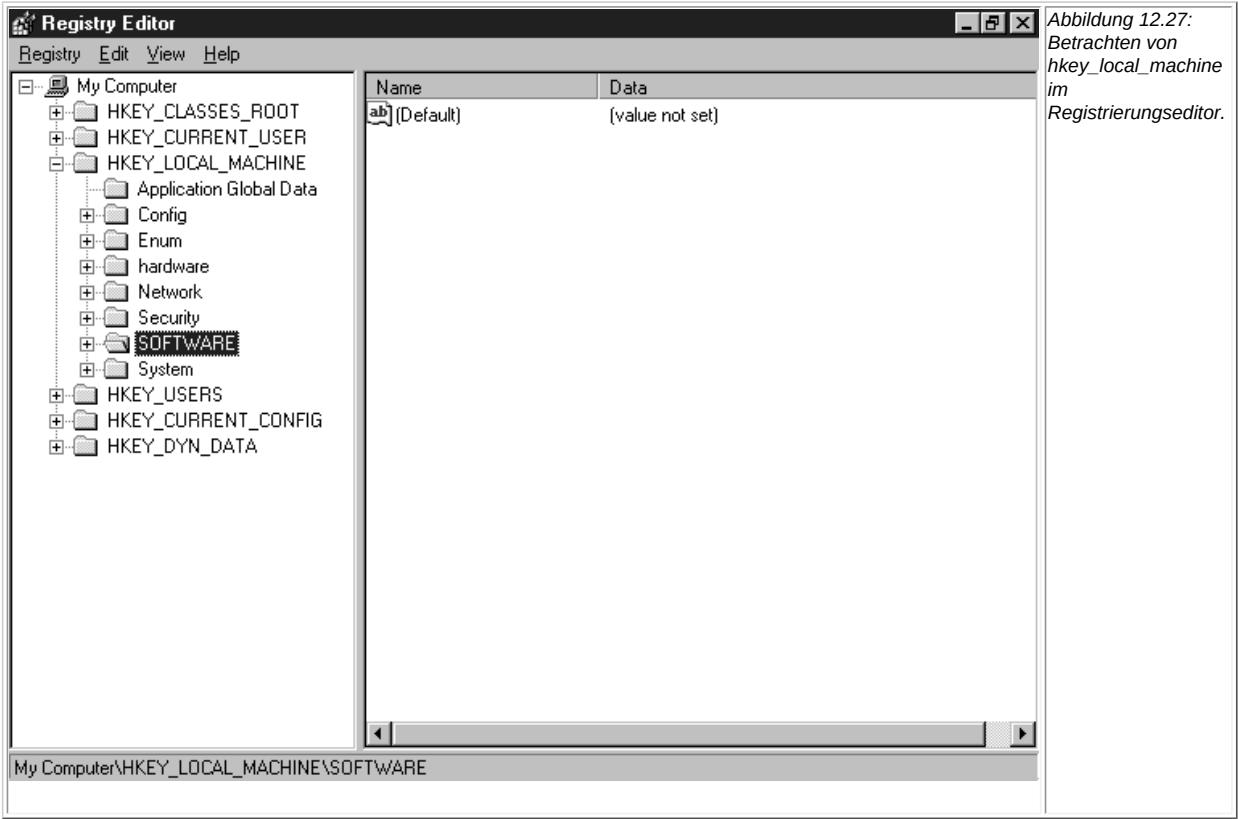
Richten Sie dagegen ein anderes Verzeichnis ein - z.B. C:\Tyo - müssen Sie drei Einträge in der Windows-95-Registrierdatenbank ändern. Die Laufzeitkomponenten von Developer/2000 - Forms Runtime, Reports Runtime und Graphics Runtime - verwenden nicht die Anweisung PATH, um nach einer Formular-, Berichts- oder Grafikdatei zu suchen, um diese auszuführen. Statt dessen verwendet jede Laufzeitkomponente von Developer/2000 einen Zeichenfolgewart des Oracle-Registrierschlüssels, um nach einer bestimmten Datei zu suchen. Im einzelnen sind dies:

- Forms Runtime (F45run32) verwendet FORMS45\_PATH
- Reports Runtime (R25run32) verwendet REPORTS25\_PATH
- Graphics Runtime (G25run32) verwendet GRAPHICS25\_PATH

In den Standardeinstellungen enthalten FORMS45\_PATH, REPORTS25\_PATH und GRAPHICS25\_PATH nur Verzeichnisse des Oracle-Verzeichnisbaums. Wenn Sie versuchen, ein Formular auszuführen, das im Verzeichnis C:\Tyo abgespeichert ist, zeigt Forms Runtime eine Meldung an, die besagt, daß es dieses Formular nicht öffnen kann. Damit Sie Ihre Formulare, Berichte oder Grafiken ausführen können, müssen Sie das diese Dateien enthaltende Verzeichnis

Developer/2000: Einführung in Oracle Forms  
mit dem Registrierungseditor hinzufügen.

- Dazu wählen Sie Start | Ausführen.
- Geben Sie regedit ein, und klicken Sie auf OK.
- Erweitern Sie die Anzeige von HKEY\_LOCAL\_MACHINE, indem Sie auf das links davon stehende + klicken (siehe Abbildung 12.27).



- Erweitern Sie die Anzeige von SOFTWARE, indem Sie auf das links davon stehende + klicken. Klicken Sie auf ORACLE.
- In der rechten Fensterhälfte sollten Sie eine Liste mit Zeichenfolgen sehen können. Blättern Sie bis zum Eintrag FORMS45\_PATH nach unten. Klicken Sie mit der rechten Maustaste und dann auf Ändern. Ein Fenster mit dem Titel Zeichenfolge bearbeiten wird angezeigt. Klicken Sie in das Feld Wert und geben Sie ;c:\Tyo ein, wie in Abbildung 12.28 gezeigt.
- Führen Sie die gleichen Schritte durch, um die Verzeichnisse GRAPHICS25\_PATH und REPORTS25\_PATH hinzuzufügen. Danach schließen Sie den Registrierungseditor.

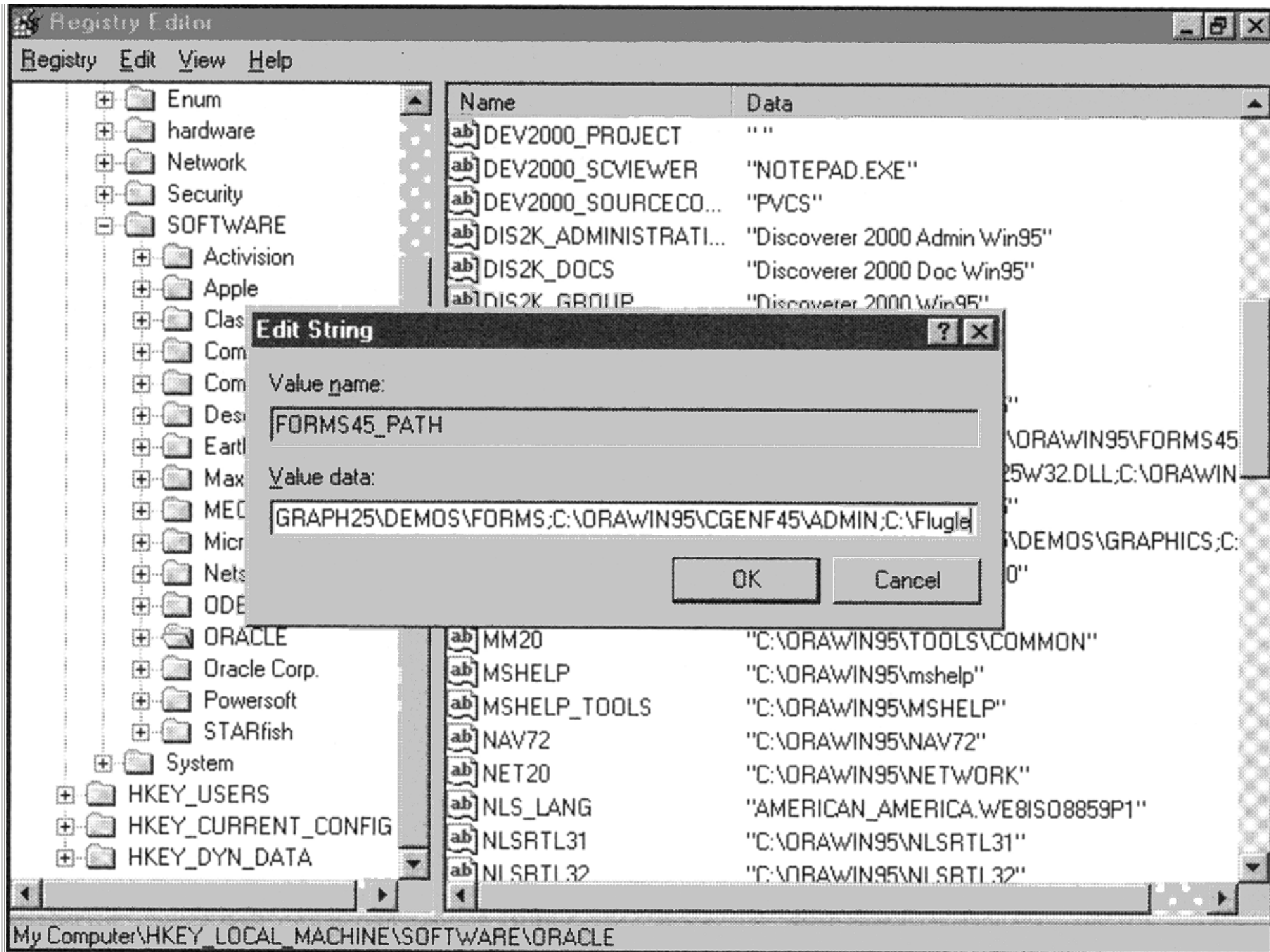


Abbildung 12.28:  
Änderung von  
forms45\_path mit  
dem  
Registrierungseditor.

## Zusammenfassung

Dies war eine in die Tiefe gehende Einführung in Oracle Forms. Dies waren die Hauptpunkte der Lektion:

- Developer/2000 besteht in erster Linie aus den vier Komponenten Oracle Forms, Oracle Reports, Oracle Graphics und Procedure Builder.
- Der Forms-Designer wird zum Entwurf und Testen und Forms Runtime zum Ausführen von Formularen eingesetzt. Forms-Designer speichert ein Formular als binäre .fmb-Datei, die auf verschiedene andere Plattformen portiert werden kann. Der Forms-Designer kann den Forms-Generator aufrufen, um aus einer .fmb-Datei eine .fmx-Datei zu erzeugen, die wiederum von Forms Runtime ausgeführt wird.
- Zur Validierung von Formularobjekten setzen Sie den Object Navigator ein. Jedes Objekt besitzt eine Gruppe von Eigenschaften, die angezeigt und geändert werden kann.
- Forms-Designer erzeugt drei Arten von Modulen: Formulare, Menüs und Bibliotheken.
- Forms-Designer kann mehrere Module aller drei Arten gleichzeitig geöffnet haben.

## Wie geht es weiter?

An Tag 13 lernen Sie, das Erscheinungsbild und Verhalten eines Formulars anzupassen. Sie lernen ein Menü und ein Master-/Detail-Formular zu erstellen und die unterschiedlichen Eigenschaften von Objekten zu ändern.

## Fragen und Antworten

Frage:

Kann ein Basistabellenblock ein Element enthalten, das nicht Teil der Blocktabelle ist?

Antwort:

*Auf jeden Fall. Solche Elemente werden dazu verwendet, aus anderen Elementen des Blocks berechnete Werte anzuzeigen. So hat z.B. ein Bestellformular üblicherweise eine Summe der Zeilenelemente, das dem Benutzer zwar angezeigt werden soll, aber nicht in der Tabelle gespeichert ist.*

Frage:

Kann ein Block auf einer Datensicht statt einer Tabelle basieren?

Antwort:

*Ja. Oft sind dies nur Anzeigeblöcke und nicht dazu gedacht, vom Benutzer geändert zu werden.*

Frage:

Kann eine Forms-Anwendung andere Produkte wie einen Bericht oder eine Grafik aufrufen?

Antwort:

*Ja. Sie können Menüelemente definieren, die einen Bericht drucken oder eine Grafik anzeigen. Wie dies gemacht wird, sehen Sie an Tag 15, »Developer/2000: Entwicklung von Berichten mit Hilfe von Oracle Reports;« und an Tag 16, »Developer/2000: Oracle Graphics und den Procedure Builder verwenden«.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

### Test

1. Richtig oder falsch? Oracle Forms Runtime führt nur eine .fmb-Datei aus.
2. Welches sind die drei Modultypen, die mit dem Forms-Designer erzeugt werden können?
3. Richtig oder falsch? Der Forms-Designer kann ein Formular nur in einer Oracle-Datenbank speichern, wenn dieses einen Basistabellenblock enthält.
4. Richtig oder falsch? Sie können einen Block erzeugen, der auf einer Tabelle basiert, auf die ein anderer Oracle-Benutzer die Rechte besitzt.

### Übung

Erzeugen Sie für die Tabelle Instructor ein Formular. Geben Sie eine ORDER-BY-Klausel ein, damit die abgerufenen Datensätze nach der Abteilungs-ID, dem Nachnamen, Vornamen und Anfangsbuchstaben des Mittelnamens des Unterrichtenden sortiert werden. Testen Sie das Formular.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

# 13

## Developer/2000: Entwicklung einer Benutzeroberfläche

[Erstellen eines Master-/Detail-Formulars](#)

[Mit dem Layout-Editor arbeiten](#)

[Ausführen des Master-/Detail-Formulars](#)

[Verbessern des Formulars](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

[Übungen](#)

In der vorhergehenden Lektion wurden Sie in einige grundlegenden Konzepte des Developer/2000 eingeführt, und Sie haben mit Oracle Forms zu arbeiten begonnen. Sie haben gelernt, wie ein einfaches Formular auf der Basis einer einzelnen Tabelle erstellt wird. In dieser Lektion fahren Sie mit der Arbeit mit Forms-Designer fort und erstellen sich ein Master-/Detail-Formular. Außerdem lernen Sie, wie mit dem Layout-Editor das Aussehen eines Formulars angepaßt werden kann.



*Ein Master-/Detail-Formular ist ein aus zwei Blöcken bestehendes Formular, dem Master- und dem Detail-Block, zur Anzeige der Datensätze. Master- und Detail-Block sind synchronisiert; wird im Master-Block ein Datensatz angezeigt, werden im Detail-Block die dazugehörigen Detail-Datensätze angezeigt. Ein Beispiel: Ein Master-/Detail-Formular enthält einen Master-Block, der mit einer Abteilungstabelle verbunden ist und einen Detail-Block, der mit einer Lehrkräfte-Tabelle verbunden ist; wird nun im Abteilungs-Block der Datensatz einer bestimmten Abteilung angezeigt, werden im Lehrkräfte-Block die mit der Abteilung verbundenen Datensätze im Lehrkräfte-Block angezeigt.*



*Für jeden Objekttyp bestehen in Oracle Forms eine Reihe von Eigenschaften. Die Seiten dieses Buches reichen einfach nicht aus, um jede Nuance dieser Eigenschaften zu untersuchen. Statt dessen konzentriere ich mich auf die wichtigsten Objekteigenschaften.*

---

## Erstellen eines Master-/Detail-Formulars

An Tag 12, »Developer/2000: Einführung in Oracle Forms«, haben Sie Forms-Designer dazu benutzt, ein einfaches Formular zur Anzeige und Bearbeitung von Studenteninformationen zu erstellen. Jetzt werden Sie erfahren, wie Sie ein Master-/Detail-Formular erstellen. Genauer gesagt, werden Sie ein Formular erstellen, mit dem der Unterricht einer Abteilung geplant werden soll. Im Master-Block sollen die Abteilung und im Detail-Block die von dieser Abteilung angebotenen Fächer angezeigt werden. Beginnen Sie, indem Sie Forms-Designer über Start | Programme | Developer/2000 for Win95 aufrufen. Stellen Sie als nächstes die Verbindung zur Oracle-Datenbank mit dem Benutzernamen, Kennwort und dem Datenbank-Alias der vorherigen Version her. In der Standardeinstellung erstellt Forms-Designer ein Formular namens Modul1.

### Definieren des Master-Blocks

Um den Master-Block Ihres Formulars zu erstellen, führen Sie die folgenden Schritte aus:

1. Sie beginnen, indem Sie den Menüeintrag Tools | New Block wählen. Das Fenster New Block Options erscheint.
2. Klicken Sie auf die Select-Schaltfläche rechts neben dem Feld Base Table. Ein weiteres Fenster öffnet sich, in dem Sie angeben können, welche Datenbank-Objekte für Tabellen zur Auswahl angezeigt werden (siehe Abbildung 13.1).
3. Klicken Sie auf Objektklasse. Die dem aktuellen Benutzer gehörende Tabelle - entsprechend des bei der Verbindung mit der Datenbank eingegebenen Benutzernamens - wird angezeigt.
4. Blättern Sie nach unten zur Abteilungstabelle, wählen Sie sie aus und klicken Sie auf OK (siehe Abbildung 13.2).
5. Standardmäßig gibt Forms-Designer dem Block den Namen der Tabelle. Wählen Sie das Register Items, und klicken Sie die Select-Columns-Schaltfläche an. Standardmäßig werden beide Spalten der Tabelle ausgewählt: Depart-ment\_ID und Department\_Name (siehe Abbildung 13.3).
6. Jetzt wählen Sie das Register Layout. Im Feld Style wählen Sie Form aus der Liste. Markieren Sie sowohl Integrity Constraints und Button Palette, wie es in der Abbildung 13.4 zu sehen ist, und klicken Sie auf OK.

*Abbildung 13.1:  
Einschränkung der  
Liste von  
Datenbankobjekten  
auf Tabellen, die im  
Besitz des aktuellen  
Benutzers sind.*



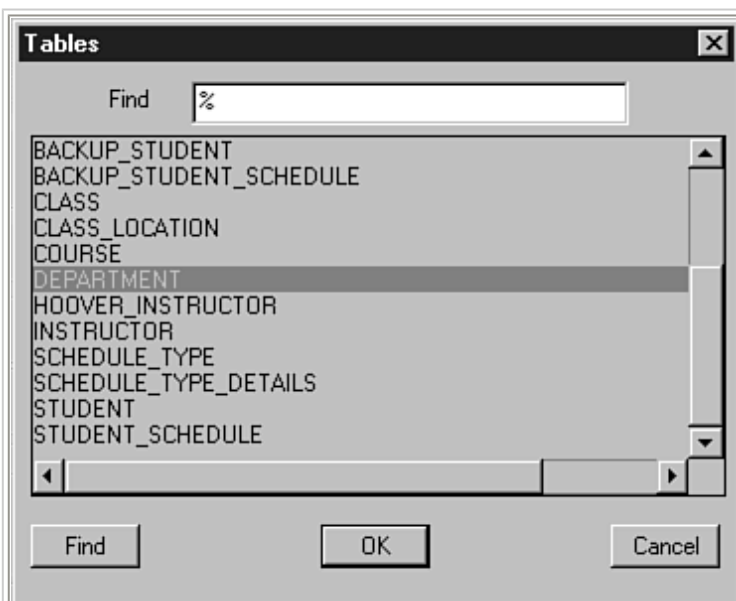
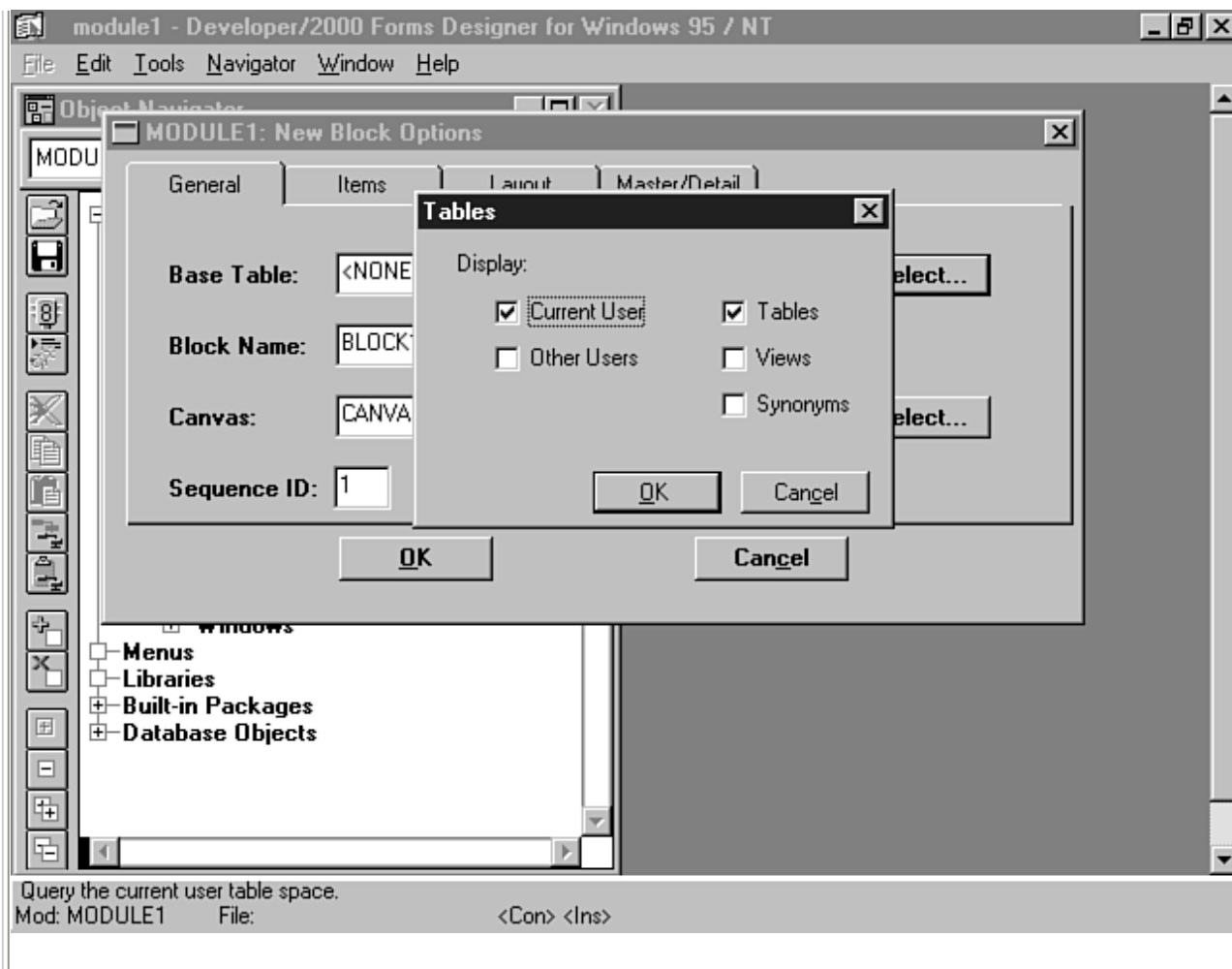


Abbildung 13.2:  
Auswahl der Department-Tabelle für  
den Master-Block

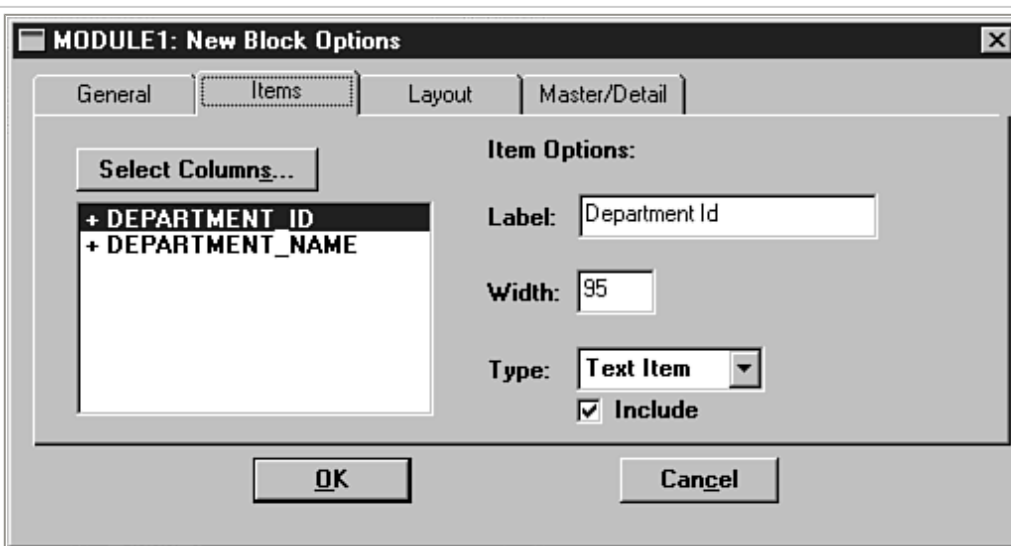


Abbildung 13.3:  
Auswahl der Spalten  
aus der Department-  
Tabelle, die im  
Master-Block  
enthalten sein sollen.

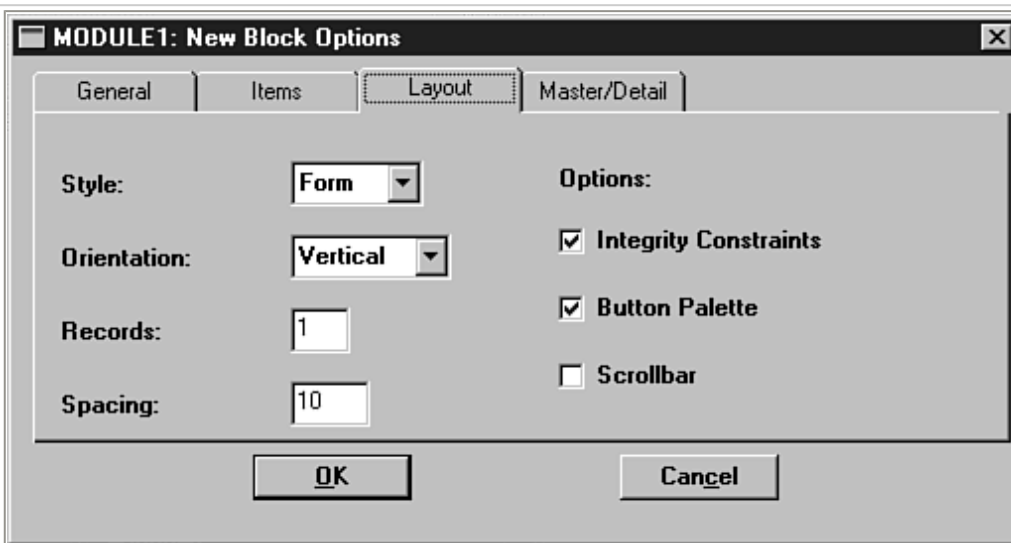
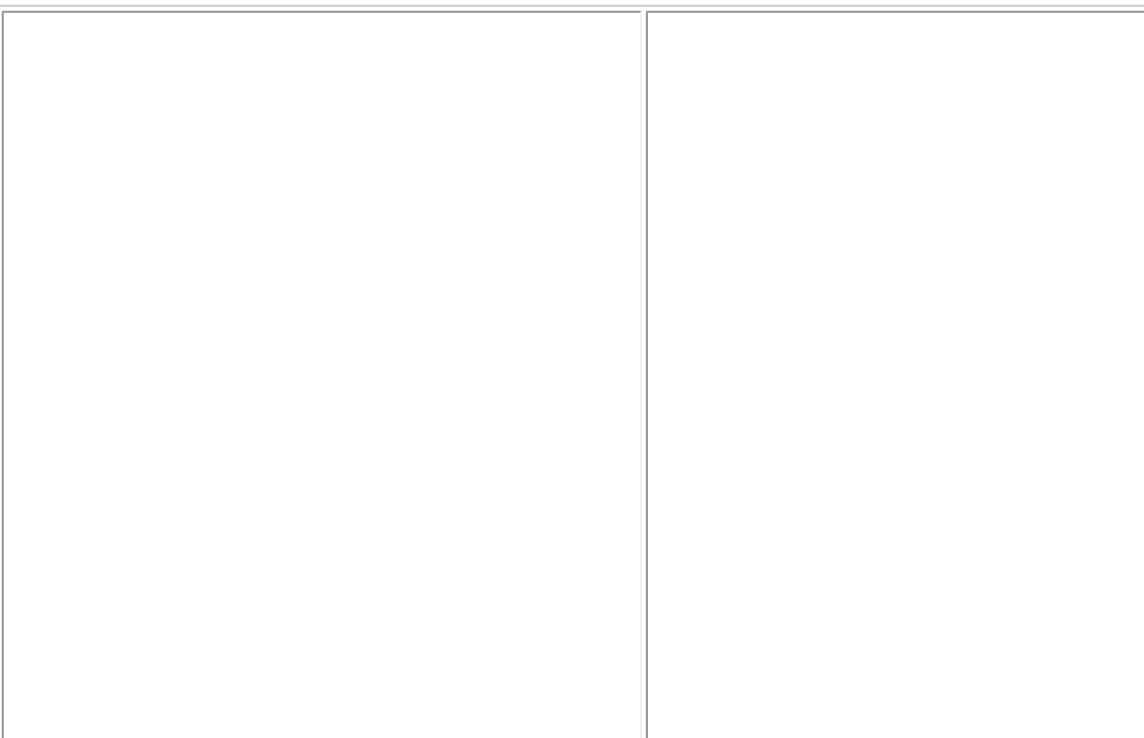


Abbildung 13.4:  
Festlegen der  
Layoutoptionen für  
den Master-Block.

Wie in Abbildung 13.5 zu sehen ist, zeigt der Object Navigator die beiden neuen Blöcke an: Department und Button Palette (der durch die Auswahl von Button-Palette erzeugte Steuerblock). Somit haben Sie den Master-Block erstellt.



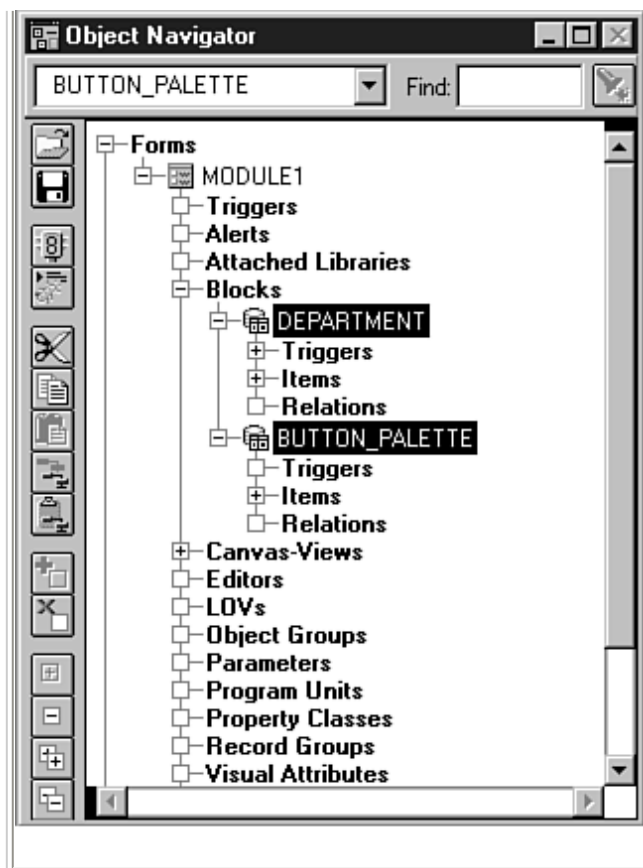


Abbildung 13.5:  
Der Object-Navigator zeigt die zwei neuen  
Blöcke an.

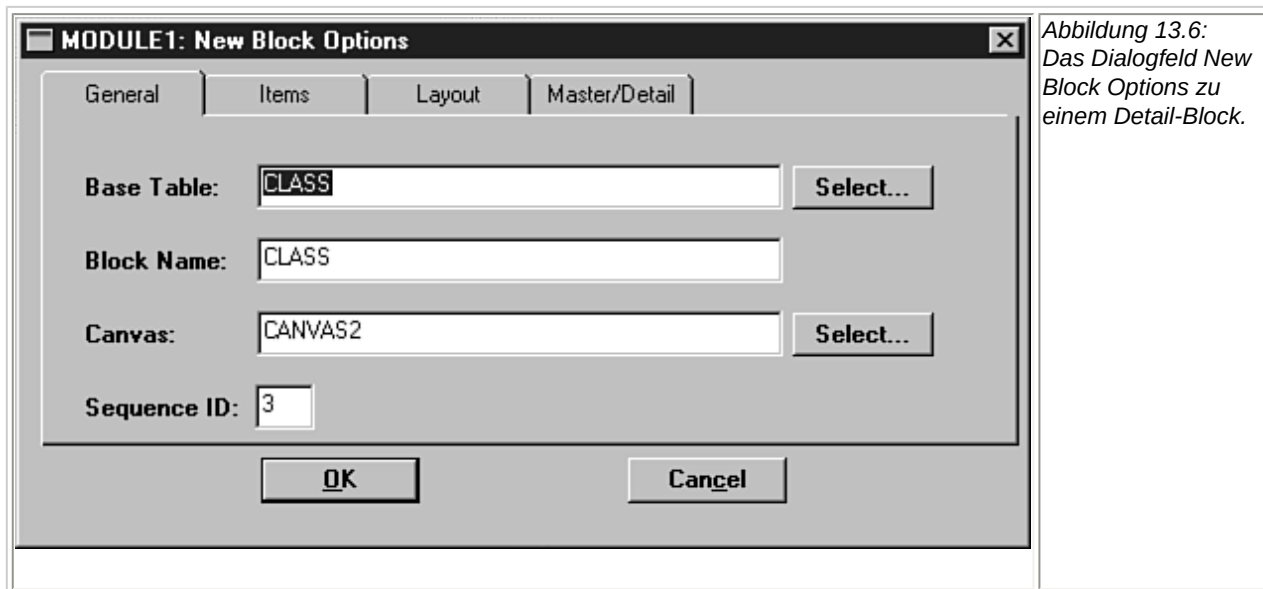


Beachten Sie, daß Sie keine Eigenschaften eingestellt haben, um diesen Block als Master-Block festzulegen. Dieses ist der Master-Block, weil Sie im Begriff sind, einen anderen Block zu erzeugen, der Department als den Master-Block identifiziert.

## Definieren des Detail-Blocks

Um den Detail-Block zu erzeugen führen Sie die gleichen Schritte aus, wie sie zur Erstellung des Master-Blocks nötig waren:

1. Sie beginnen, indem Sie den Menüeintrag Tools | New Block wählen. Klicken Sie wieder auf die Select-Schaltfläche rechts neben dem Feld Base Table, um alle verfügbaren Tabellen aufzulisten.
2. Klicken Sie im Tabellen-Fenster auf OK, um alle Tabellen des aktuellen Eigentümers anzuzeigen.
3. Wählen die Tabelle Class und klicken Sie auf OK. Es müßte das Fenster New Block Options aus der Abbildung 13.6 zu sehen sein.
4. Wählen Sie das Register Items und dann Select Columns. Klicken Sie auf die Spalten Semester und School Year, um diese aus der Auswahl zu nehmen. Links der beiden Spalten müßten Sie ein - sehen, das anzeigt, daß Forms-Designer für diese Spalten keine Elemente erzeugen soll; für dieses Beispiel werden Sie die Spalten Semester und School Year nicht brauchen.

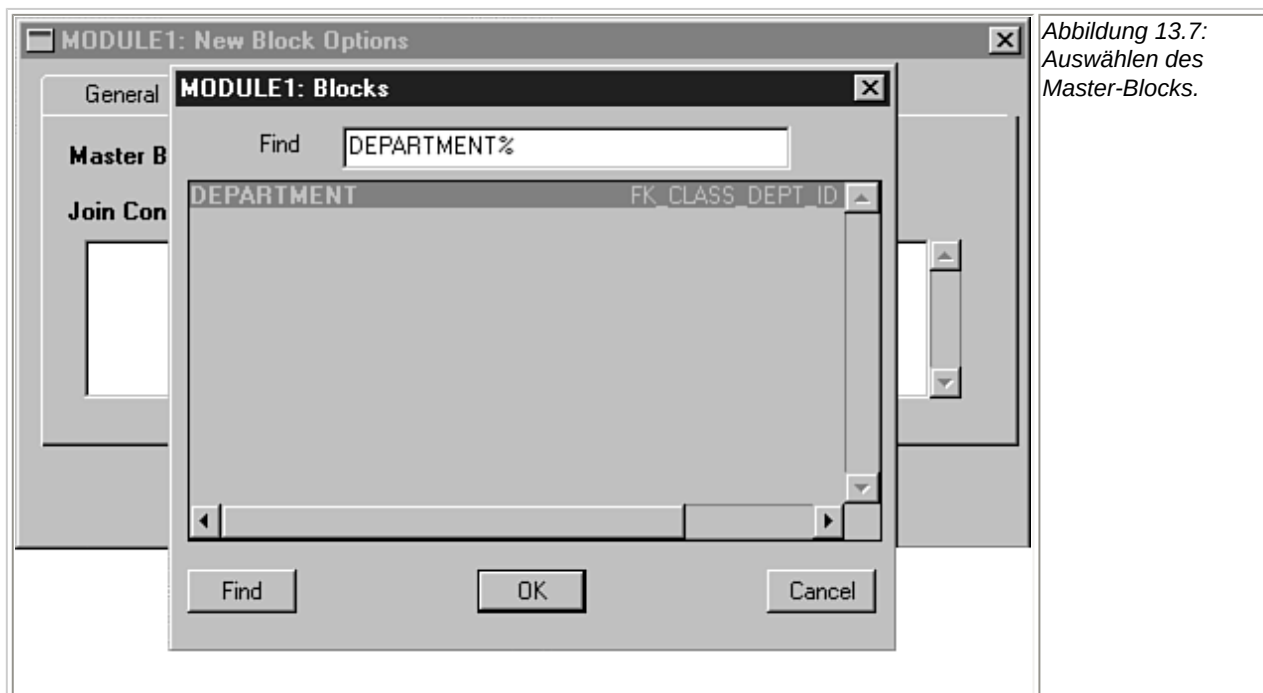


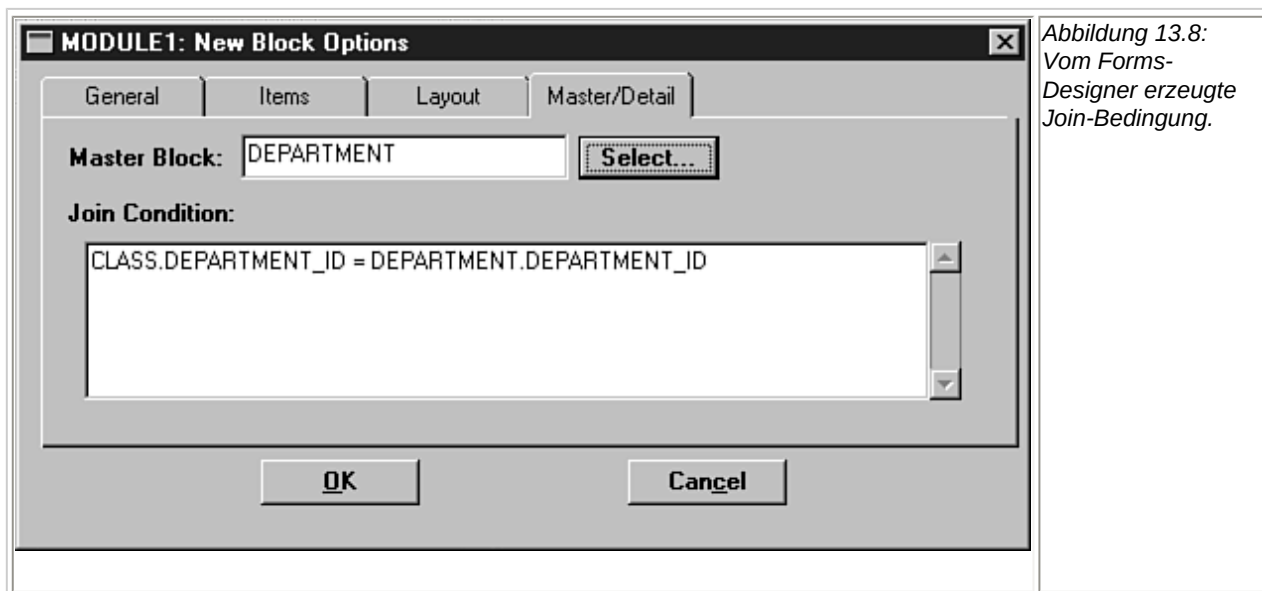
Wählen Sie dann das Register Layout. Vergewissern Sie sich, daß im Feld Style Tabular und als Orientation Vertical ausgewählt sind. Stellen Sie Records auf 10 ein. Und markieren Sie die Kontrollkästchen Integrity Constraints und Button-Palette.

### Definieren der Beziehung

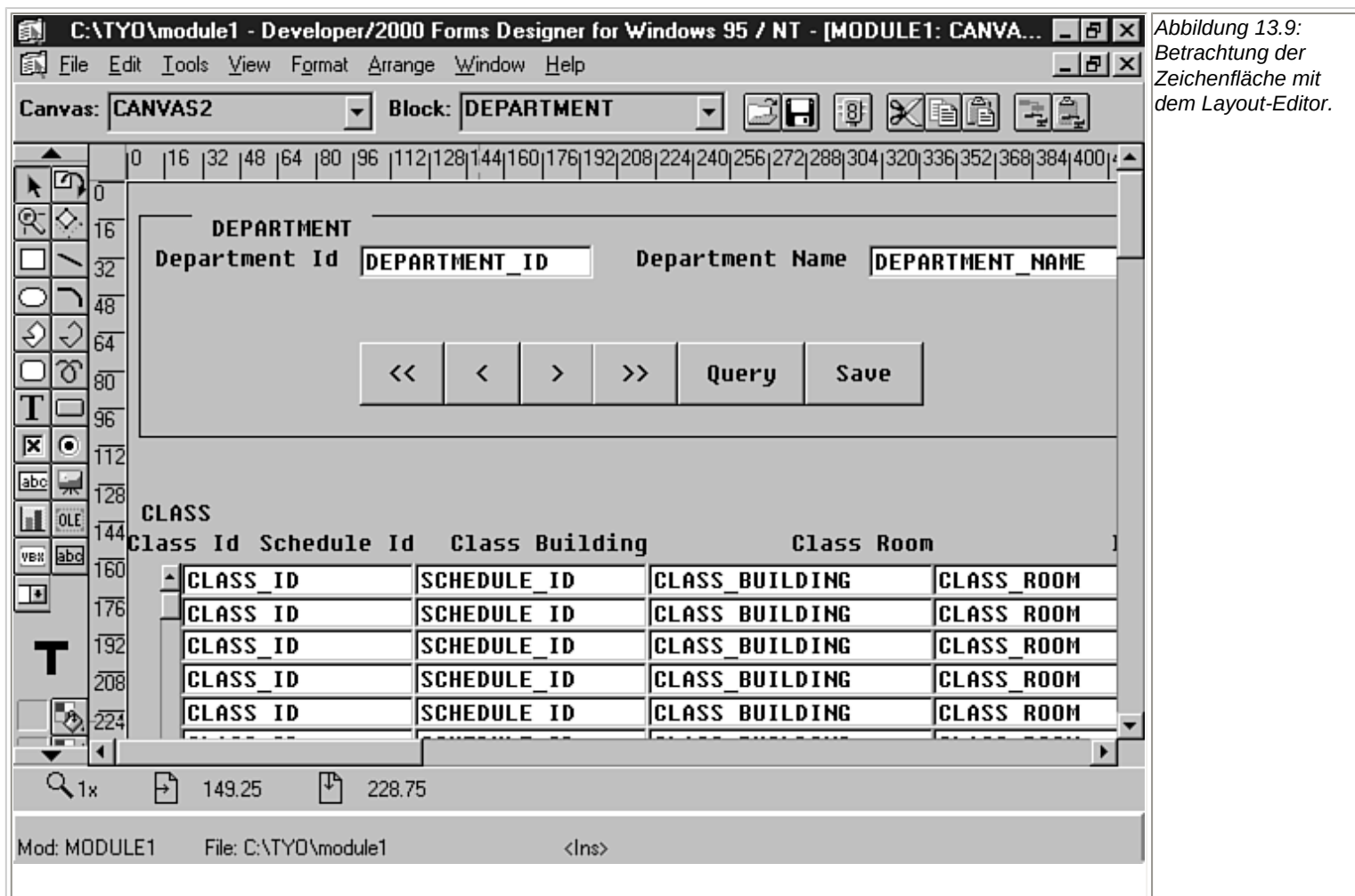
Mit den folgenden Schritten legen Sie die Beziehung zwischen dem Master- und dem Detail-Block fest:

1. Um den Master-Block auszuwählen, wählen Sie das Register Master-/Detail.
2. Rechts vom Feld Master Block klicken Sie auf die Select-Schaltfläche. Im jetzt erscheinenden Fenster ist der Department-Block standardmäßig ausgewählt (siehe Abbildung 13.7). Klicken Sie auf OK.
3. Unter Verwendung des für die Class-Tabelle bestehenden Fremdschlüssels wird Forms-Designer die Verknüpfungsbedingung zwischen dem Master- und dem Detail-Block herstellen, wie es in Abbildung 13.8 gezeigt wird. Klicken Sie auf OK, um den Detail-Block Class zu erstellen.





Schauen wir uns die erzeugte Ansicht näher an. Wählen Sie den Menüeintrag Select Tools | Layout Editor. Abbildung 13.9 zeigt auf, was Sie im Layout-Editor sehen sollten. Wie Sie feststellen, müssen Sie an dem Formular noch eine Reihe von Änderungen vornehmen, bevor Sie es verwenden können. Im nächsten Abschnitt erlernen Sie den Einsatz des Layout-Editors zur Bearbeitung des Aussehens und der Eigenschaften des Blockelements.



## Mit dem Layout-Editor arbeiten

Beim Aufruf des Layout-Editors stellen Sie fest, daß sich die Menüpunkte verändern. Im Layout Editor können Sie unter Format und Arrange eine Reihe von Menüpunkten aufrufen. Lassen Sie uns zum Anfang die Schriftart für alle Textelemente ändern.

## Text formatieren

Um alle Elemente in der Ansicht auszuwählen, drücken Sie die Tasten (Strg)+(A). Um für alle ausgewählten Elemente die Schriftart zu ändern, wählen Sie Format | Font, wonach das Fenster Font erscheint. Wählen Sie MS Sans Serif, Regular als Schriftattribut und 10 Punkt aus (siehe Abbildung 13.10). Klicken Sie auf OK, damit die Änderungen wirksam werden.



Um mehrere Elemente auszuwählen, können Sie mit der Maus um die auszuwählenden Elemente ein Rechteck ziehen, indem Sie die linke Maustaste gedrückt halten und die Maus solange ziehen, bis alle Elemente vom Rechteck umgeben sind. Alternativ dazu können Sie auch mehrere Elemente anklicken, während Sie die Umschalt-Taste gedrückt halten.

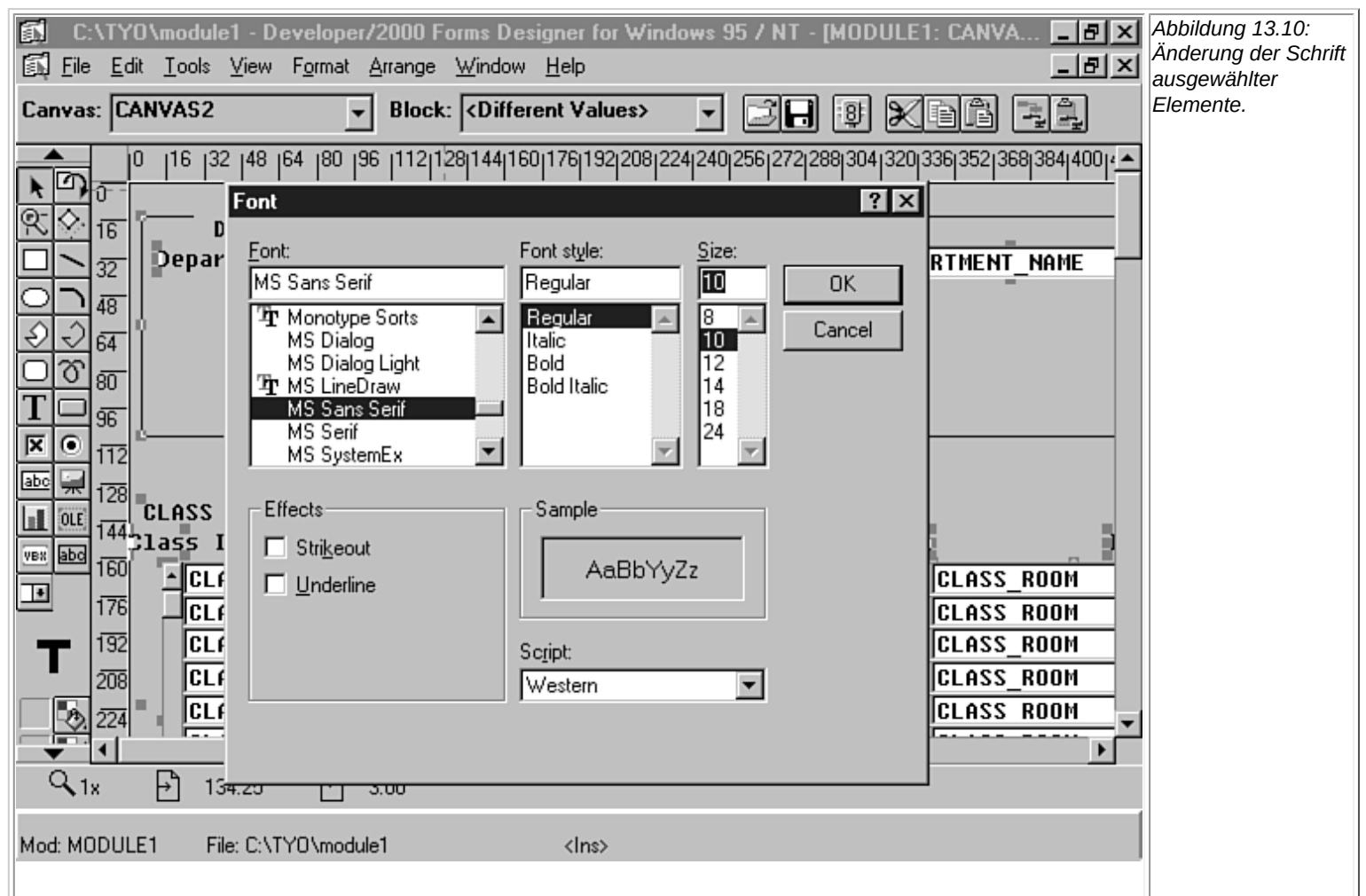


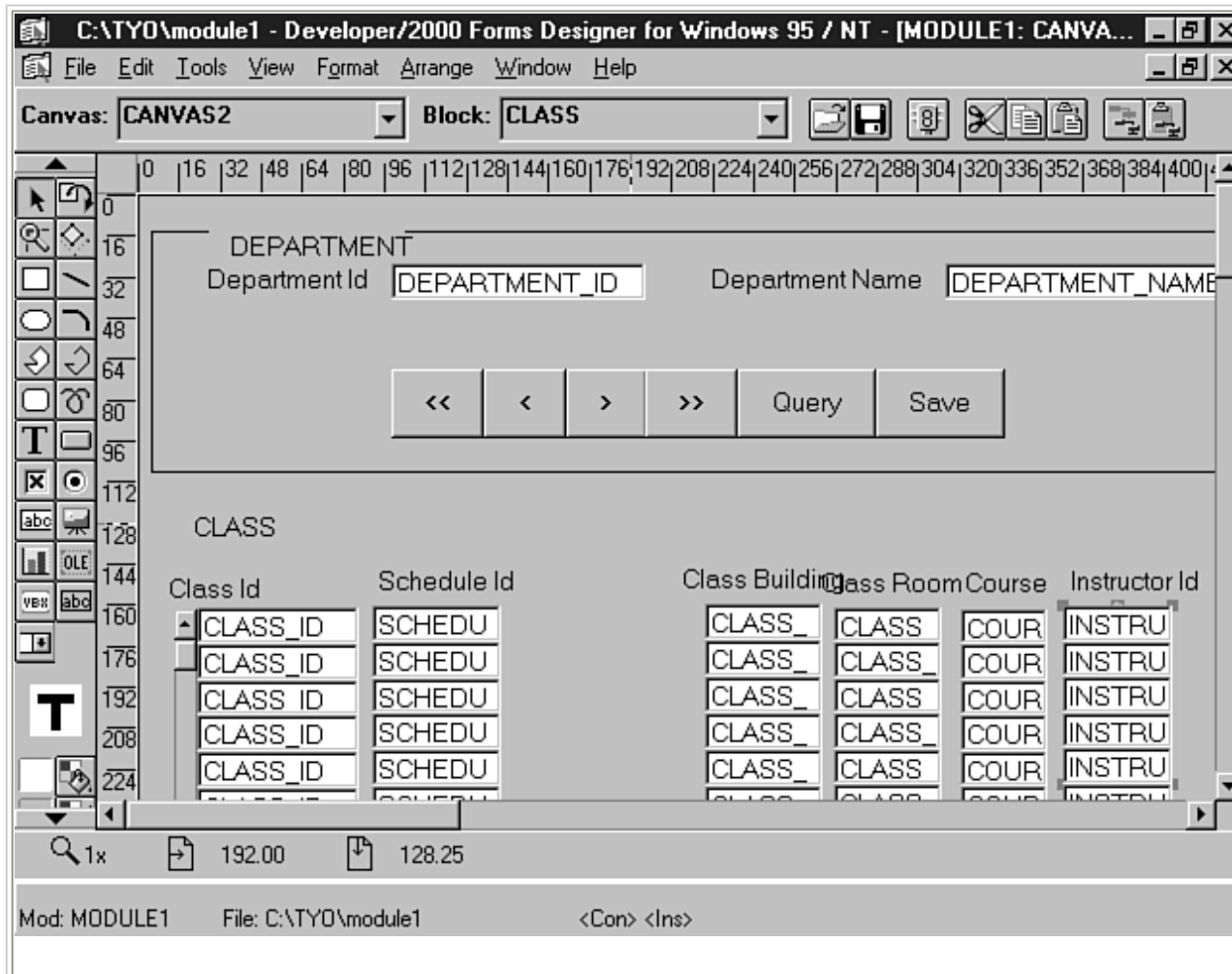
Abbildung 13.10:  
Änderung der Schrift  
ausgewählter  
Elemente.

## Die Größe von Objekten einstellen

Im Layout erkennen Sie, daß die Größe einiger Felder geändert werden muß. So können z.B. die Felder Class\_Building und Class\_Room schmaler eingestellt werden. Um die Breite eines Feldes zu verändern, wählen Sie es mit der Maus aus. Erfassen Sie einen Punkt auf der rechten Seite des Elements, und ziehen Sie ihn bei gedrückter linker Maustaste nach links, bis das Element die gewünschte Breite hat. Verringern Sie die Breite der Elemente Class\_ID, Schedule\_ID, Class\_Building, Class\_Room und Instructor\_ID (siehe Abbildung 13.11). Stören Sie sich nicht daran, daß die Elemente nicht mehr zueinander ausgerichtet sind; in Kürze werden Sie dies berichtigen.

Um nun das Rechteck um den Master-Block zu entfernen, wählen Sie die untere Seite des Rechtecks an, und drücken Sie die Entfernen-Taste. Rücken Sie die Felder mit ihren Text-Labels wieder zurecht. Bewegen Sie die Button-Palette-Elemente nach oben auf dem Canvas. Und lassen Sie uns Rechtecke um die Button-Palette-Elemente, die Felder der Department-Tabelle und deren Labels hinzufügen.

1. Wählen Sie dazu das Werkzeug Rectangle aus der vertikalen Werkzeugleiste.
2. Bewegen Sie die Maus in die linke, obere Ecke des Rechtecks, halten Sie die linke Maustaste gedrückt, und ziehen Sie die Maus in die rechte, untere Ecke des Rechtecks, bevor Sie die Maustaste loslassen. Es hängt jetzt von Ihren vorhergehenden Aktionen im Layout-Editor ab, ob Sie das Rechteck jetzt sehen können oder nicht.
3. Um das Rechteck sichtbar zu machen, wählen Sie das Werkzeug Line Color, wie in Abbildung 13.12 zu sehen ist (achten Sie auf die Sprechblasen-Hilfe, wenn Sie die Maus über das Werkzeug bewegen). Es erscheint ein Fenster mit verfügbaren Farben - wählen Sie Schwarz (siehe Abbildung 13.13).
4. Um für das Rechteck einen Betrachtungswinkel einzurichten, wählen Sie den Menüeintrag Format | Bevel. Das Rechteck sollte danach einen Betrachtungswinkel, entsprechend der Abbildung 13.14, aufweisen. Natürlich ist dies eine Frage des persönlichen Geschmacks, und Sie können jedes individuelle Aussehen wählen.





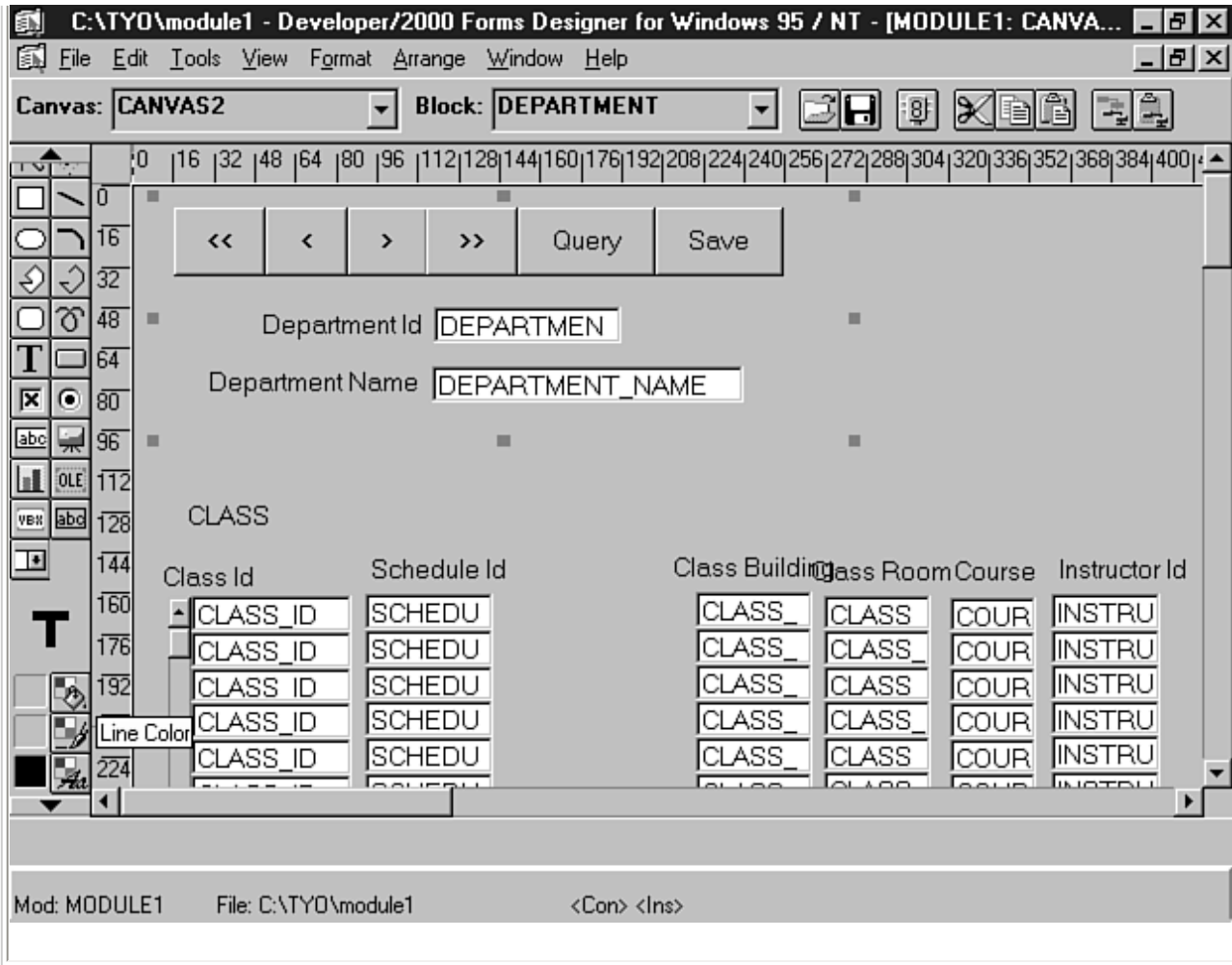


Abbildung 13.12:  
Mit dem Line-Color-  
Werkzeug kann das  
Rechteck sichtbar  
gemacht werden.

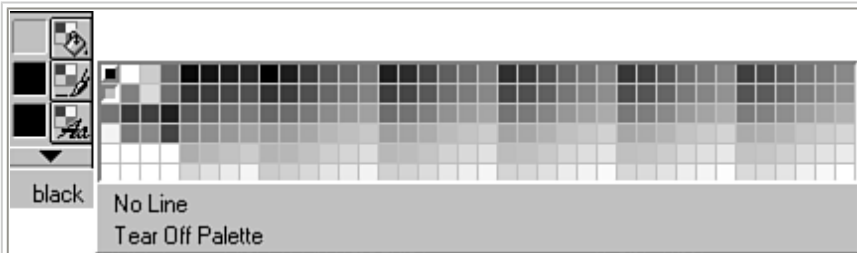


Abbildung 13.13:  
Setzen der Linienfarbe für das  
Rechteck.

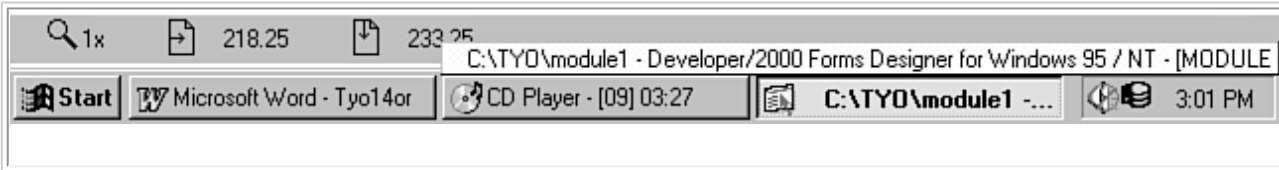


Abbildung 13.14:  
Festlegen der  
Einfügeposition für  
das Rechteck.

### Text ändern

Einige der Überschriften des Blocks Class müssen noch geändert werden: in einigen Fällen sollten passendere Überschriften gewählt werden, oder sie sollten formatiert werden. Um z.B. Class Building in Building zu ändern, wählen Sie den Text mit der Maus und dann das Textwerkzeug. Der Zeiger wird zum Fadenkreuz. Bewegen Sie den Zeiger auf Class Building, und klicken Sie die Überschrift mit der linken Maustaste an, worauf um diese ein Kasten und eine Einfügemarke, die Ihre Position angibt, angezeigt werden (siehe Abbildung 13.15). Entfernen Sie das Wort Class aus der Überschrift.



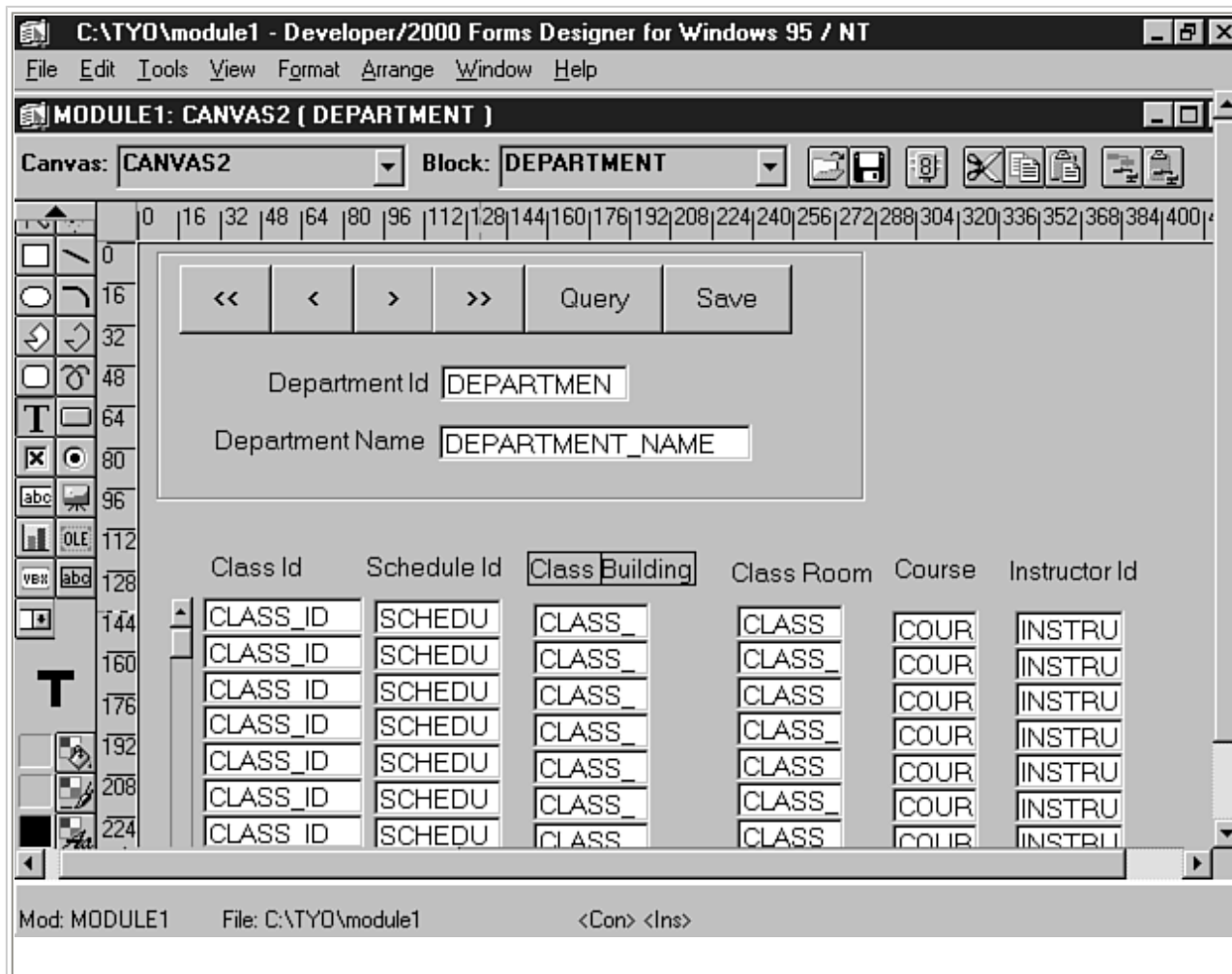


Abbildung 13.15:  
Änderung einer  
Spaltenüberschrift

Ist eine Spaltenüberschrift breiter als die dazugehörige Spalte, können Sie die Wörter in separate Zeilen setzen. Dadurch können Sie die Spalten schmaler einstellen. So können Sie z.B. Instructor\_ID so ändern, daß Instructor und ID in zwei Zeilen stehen. Dazu wählen Sie die Spaltenüberschrift aus, dann das Textwerkzeug und klicken mit der linken Maustaste auf die Spaltenüberschrift. Bewegen Sie die Einfügemarke vor ID, und drücken Sie die Eingabetaste.

Um eine Spaltenüberschrift zu zentrieren, wählen Sie den Menüeintrag Format | Alignment | Center.

### Größe, Abstand und Ausrichtung von Objekten einstellen

Als Sie im Block Class die Elemente zurückgesetzt haben, verloren diese ihre Ausrichtung. Ebenso kann es vorkommen, daß Sie einige der Elemente unbeabsichtigt in ihrer Größe verändern. Um eine Gruppe von Elementen auf die gleiche Größe einzustellen, wählen Sie alle aus, indem Sie entweder mit der Maus ein Rechteck um die Objekte ziehen oder jedes davon bei gedrückt gehaltener Umschalttaste anklicken.

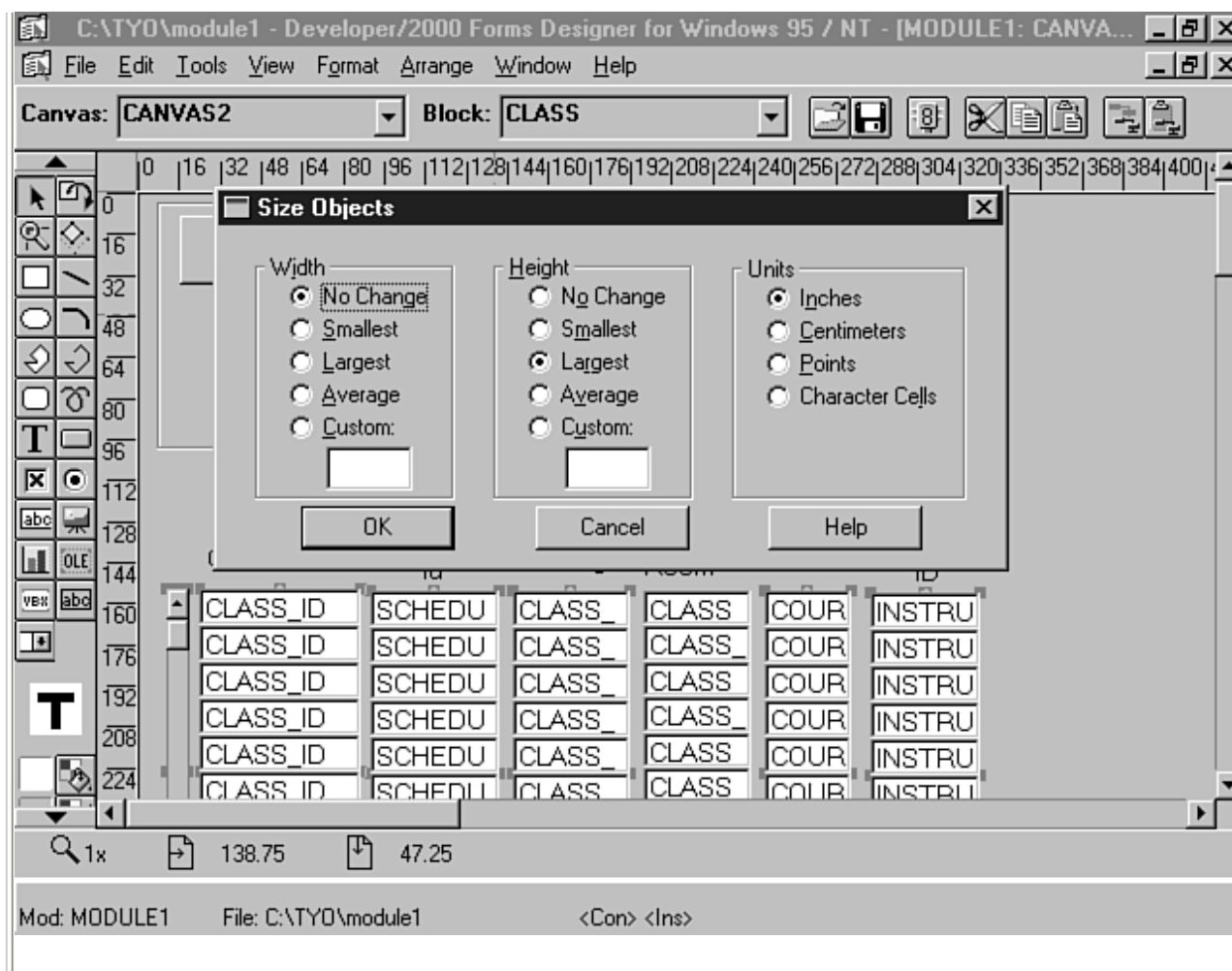


Abbildung 13.16:  
Festlegen der Größe  
einer Gruppe von  
Elementen.

Wählen Sie z.B. die folgenden sieben Elemente des Blocks Class aus: den Rollbalken, Class\_ID, Schedule\_ID, Class\_Building, Class\_Room, Course\_ID und In-structor\_ID. Um diesen Elementen die gleiche vertikale Größe zuzuweisen, wählen Sie den Menüeintrag Arrange | Size Objects. Ein Fenster mit dem Namen Size Objects wird geöffnet. Da Sie für keines der Elemente die Breite ändern wollen, lassen Sie die Einstellung No Change für Width unverändert. Für Height wählen Sie das Kontrollkästchen Largest, damit alle Elemente die Höhe des größten Kontrollkästchens erhalten (siehe Abbildung 13.16). Klicken Sie auf OK, damit die Änderungen wirksam werden.

Jetzt werden Sie die Elemente ausrichten. Stellen Sie sicher, daß die Elemente noch ausgewählt sind. Wählen Sie Arrange | Align Objects, und ein Fenster namens Align Objects erscheint. Wählen Sie Each Other in der Gruppe Align To und Distribute in der Gruppe Horizontally; wodurch der horizontale Raum zwischen den Elementen gleichmäßig verteilt wird. Wählen Sie Align Top in der Gruppe Vertically, und klicken Sie auf OK (siehe Abbildung 13.17).



*Wenn die Elemente in horizontaler oder vertikaler Richtung nicht gleichmäßig verteilt werden, haben Sie wahrscheinlich ein anderes Element versehentlich ausgewählt, das diese Ausrichtung beeinflusst.*

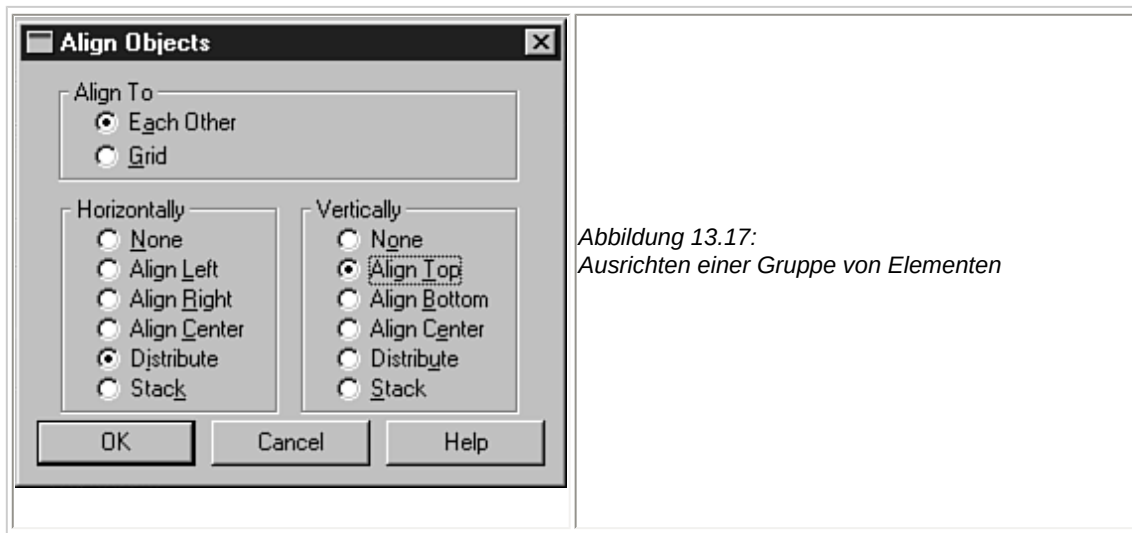


Abbildung 13.17:  
Ausrichten einer Gruppe von Elementen

Lassen Sie uns jetzt die Spaltenüberschriften bearbeiten. Da die Textelemente des Blocks Class horizontal neu verteilt wurden, kann es sein, daß Sie die Spaltenüberschriften umstellen müssen.

1. Stellen Sie sicher, daß Sie nur die Spaltenüberschriften ausgewählt haben.
2. Wählen Sie den Menüeintrag Arrange | Align Objects.
3. Im Fenster Align Objects wählen Sie Each Other für die Gruppe Align To.
4. Wählen Sie None für die Gruppe Horizontally; der Raum zwischen den Spaltenüberschrift soll nicht gleichmäßig verteilt werden, da sonst diese nicht richtig über den Spalten stehen.
5. Wählen Sie Align top für die Gruppe Vertically und klicken Sie auf OK.
6. Ziehen Sie um den Block Class ein Rechteck. Abbildung 13.18 zeigt die Gestaltung des Master-/Detail-Formulars so, wie es auf Ihrem Bildschirm dargestellt werden sollte.

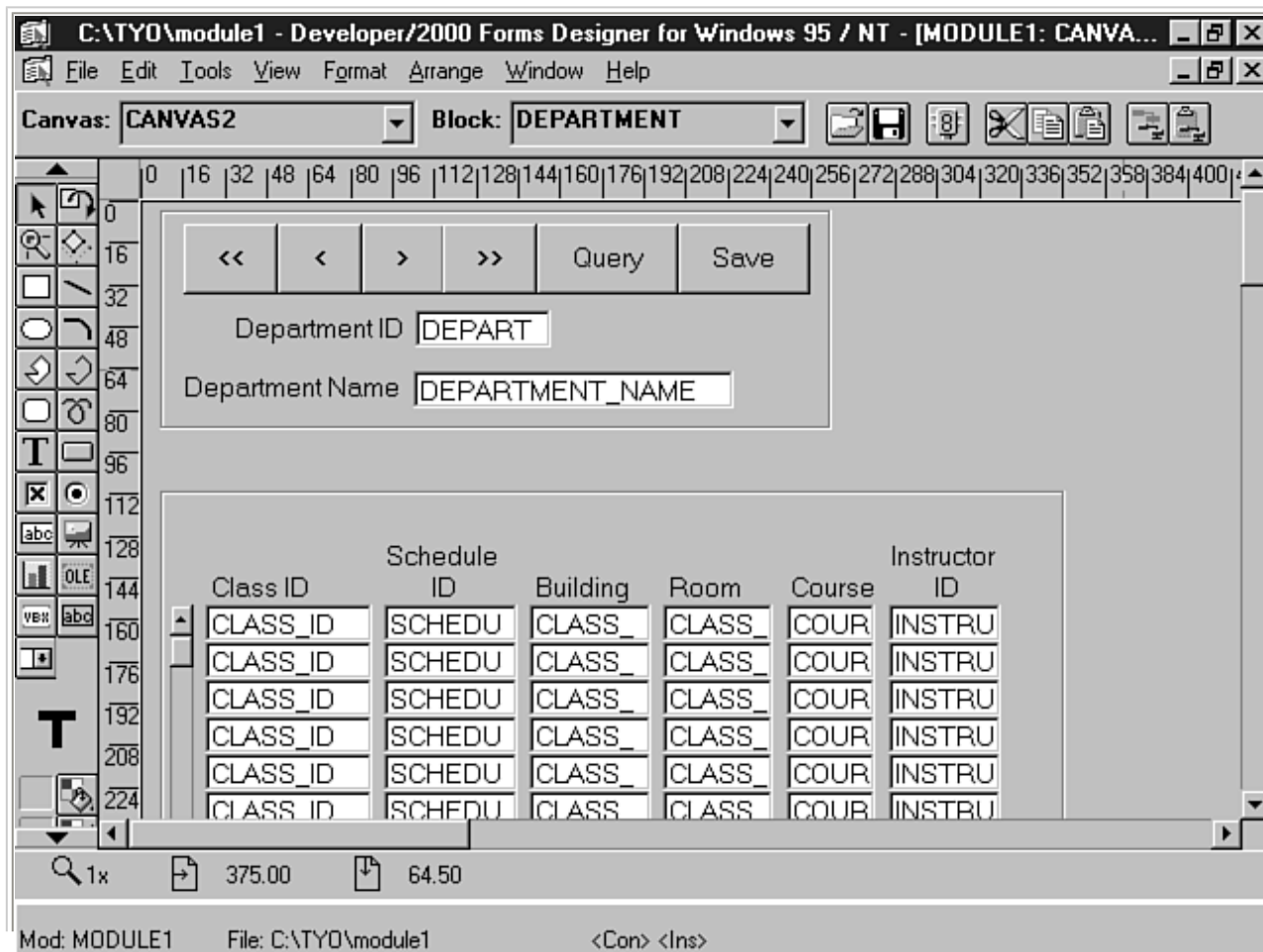
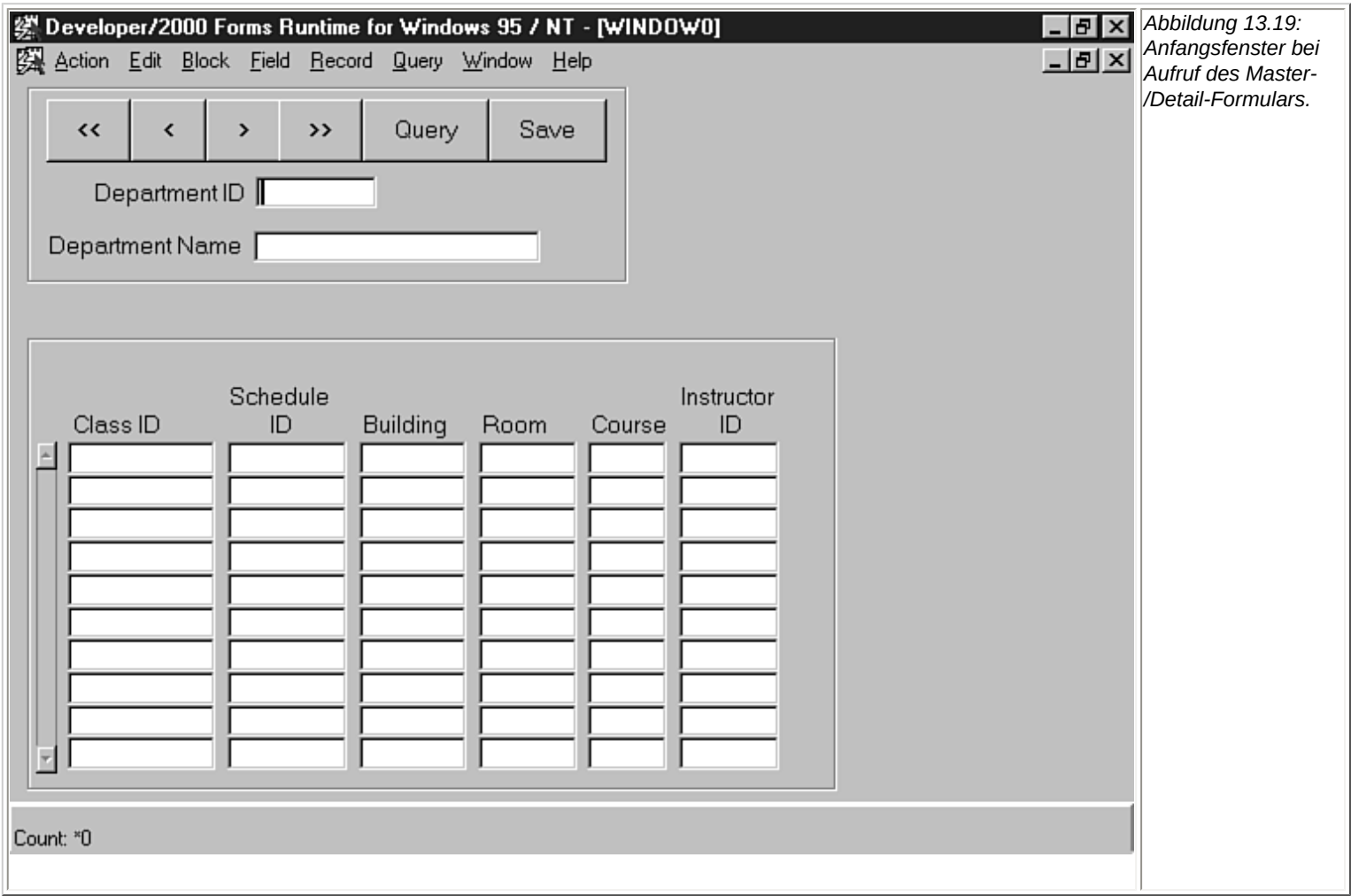


Abbildung 13.18:  
Layout des  
Master-/Detail-  
Formulars.

Mod: MODULE1      File: C:\TYO\module1      <Con> <Ins>

## Ausführen des Master-/Detail-Formulars

Wahrscheinlich sind Sie gespannt, wie das Ergebnis Ihrer Arbeit aussehen wird. Die beste Methode, herauszufinden, ob ein Formularentwurf korrekt ist, besteht darin, ihn zu testen. Dazu wählen Sie entweder den Menüeintrag File | Run oder klicken auf die grüne Ampel in der horizontalen Werkzeugleiste. Zuerst wird von Forms-Designer eine ausführbare FMX-Datei erzeugt. Liegen keine Fehler vor, ruft Forms-Designer zur Ausführung der FMX-Datei Forms Runtime auf. Es müsste ein Fenster namens Developer/2000 Forms Runtime for Windows 95/NT[WINDOW0] erscheinen. In diesem Fenster müsste ein Unterfenster mit dem Titel WINDOW0 geöffnet sein. Maximieren Sie beide Fenster. Abbildung 13.19 zeigt, wie es auf Ihrem Bildschirm aussehen sollte.



Um eine Abfrage durchzuführen, klicken Sie zweimal auf die Query-Schaltfläche; nach dem ersten Klick können Sie die Suchkriterien eingeben, und nach dem zweiten Klick wird die Abfrage ausgeführt. Wie Sie sehen, stehen die Abteilungen in keiner bestimmten Reihenfolge. Genausowenig stehen die zugehörigen Klassen in einer bestimmten Reihenfolge. Im nächsten Abschnitt werden Sie die Eigenschaft Order by für den Master- und den Detail-Block ändern. Dazu verlassen Sie Forms Runtime, indem Sie File | Exit wählen.

## Verbessern des Formulars

Im Forms-Designer wechseln Sie in das Fenster Object Navigator, indem Sie den Menüeintrag Window | Object Navigator wählen. Wählen Sie den Block Department an und nach einem Klick mit der rechten Maustaste Properties aus dem Kontextmenü. Blättern Sie nach unten bis zur Eigenschaft Order By, und geben Sie Department\_ID ein (siehe Abbildung 13.20). Drücken Sie die Eingabetaste, um die Eigenschaft zu übernehmen.

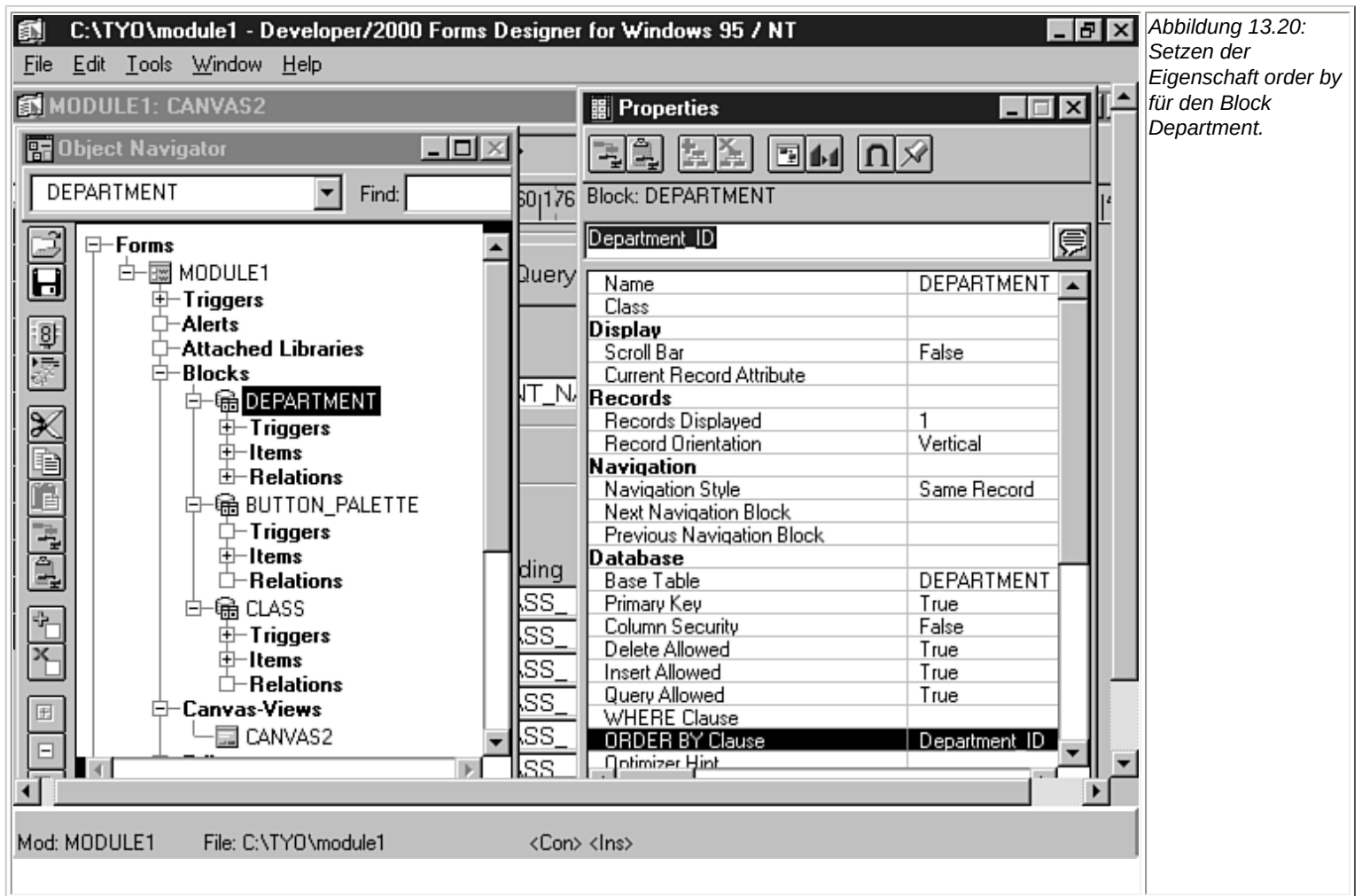


Abbildung 13.20:  
Setzen der  
Eigenschaft order by  
für den Block  
Department.

Wählen Sie den Block Class, und klicken Sie mit rechten Maustaste und dann auf Properties. Es ist wahrscheinlich sinnvoller, den Block nach Course\_ID als nach Class\_ID zu sortieren, weil die Course-ID jedem vertraut ist: der Verwaltung, den Lehrkräften und den Studenten. Blättern Sie nach unten zur Eigenschaft Order by, und geben Sie Course\_ID ein. Drücken Sie die Eingabetaste, um die Eigenschaft zu übernehmen.

An dem Formular sollten noch weitere Änderungen vorgenommen werden. Erst einmal sind einige Felder zu breit - diese sind Class\_ID, Schedule\_ID und Class\_Room. Ein Feld ist zu schmal: Class\_Building. Außerdem wäre es sicherlich sinnvoller, Course\_ID direkt rechts von Class\_ID zu setzen. Nehmen wir diese Änderungen vor und lassen das Formular erneut laufen.

Führen Sie eine Abfrage durch, und blättern Sie zu History Department, indem Sie auf > klicken oder den Menüeintrag Record | Next wählen, bis diese im Block Department angezeigt wird (siehe Abbildung 13.21).

Wenn Sie den Raum, in dem Class 108400 (Geschichtskurs 115) stattfinden soll, ändern möchten, müssen Sie zum Block Class wechseln. Standardmäßig ist der aktuelle Block, Department, der erste Block. Um zum nächsten Block zu gelangen, drücken Sie die Tasten STRG+BILD-NACH-UNTEN. Hierbei stellen Sie jedoch fest, daß die ganz linksstehende Fläche der Button-Palette aktiviert wird. Dieses Verhalten läßt sich durch die Reihenfolge der Blöcke erklären.

Bei Verwendung des Object Navigator zur Anzeige der Blöcke erkennen Sie die Reihenfolge der Blöcke: Department, Button\_Palette und Class (siehe Abbildung 13.22); in dieser Reihenfolge werden die Blöcke mit Befehl Next Block angesprungen. Um die Reihenfolge der Blöcke zu ändern, wählen Sie den Block an und ziehen so weit, bis der Zeiger über dem Block steht, der dem ausgewählten Block vorhergehen soll. Um z.B. den Block Class umzustellen, wählen sie ihn mit der Maus an und ziehen bei gedrückt gehaltener linken Maustaste, bis der Zeiger über dem Block Department steht, und lassen die linke Maustaste los.

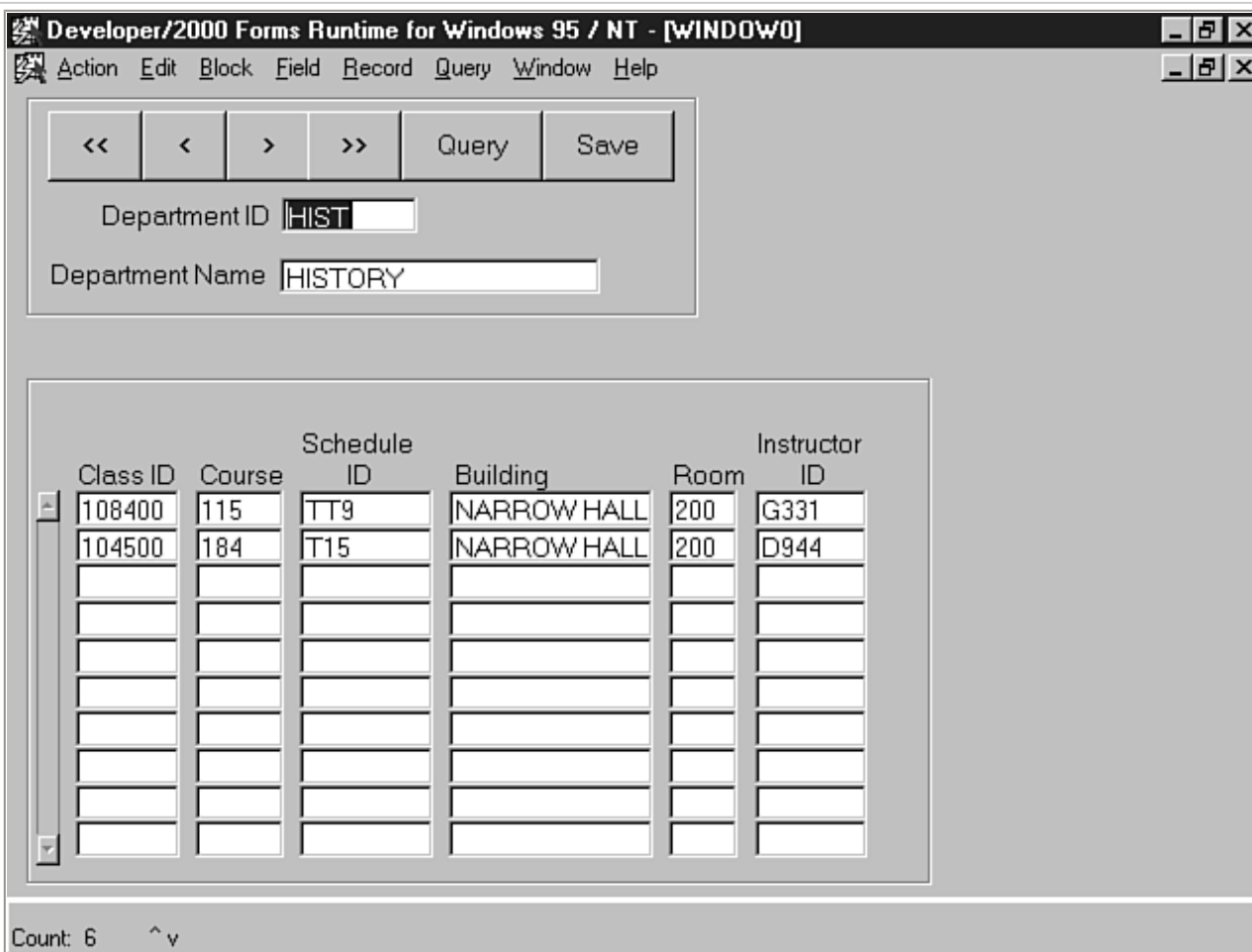


Abbildung 13.21:  
Betrachtung von  
Vorlesungen, die  
vom History  
Department  
(Geschichtsseminar)  
angeboten werden.

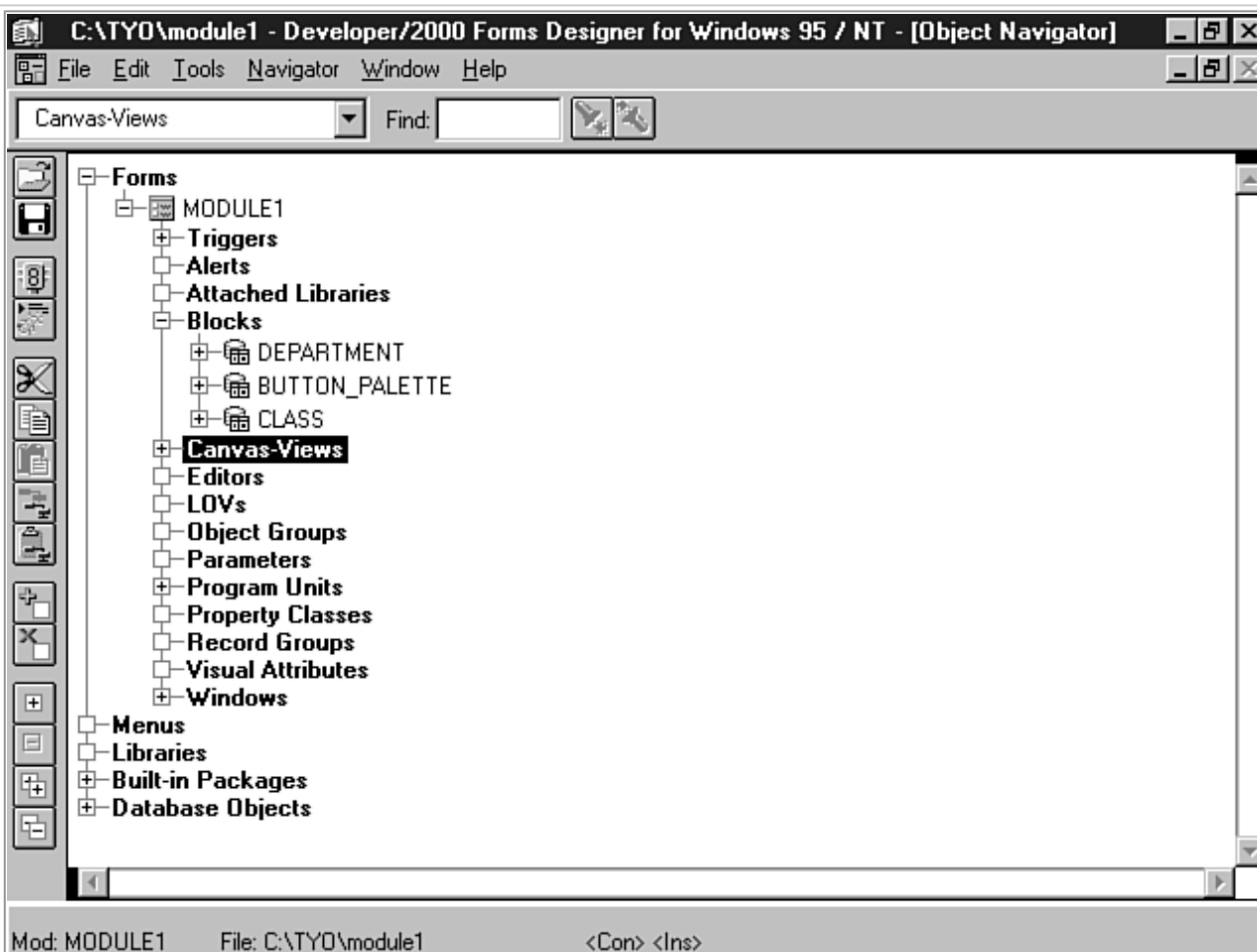
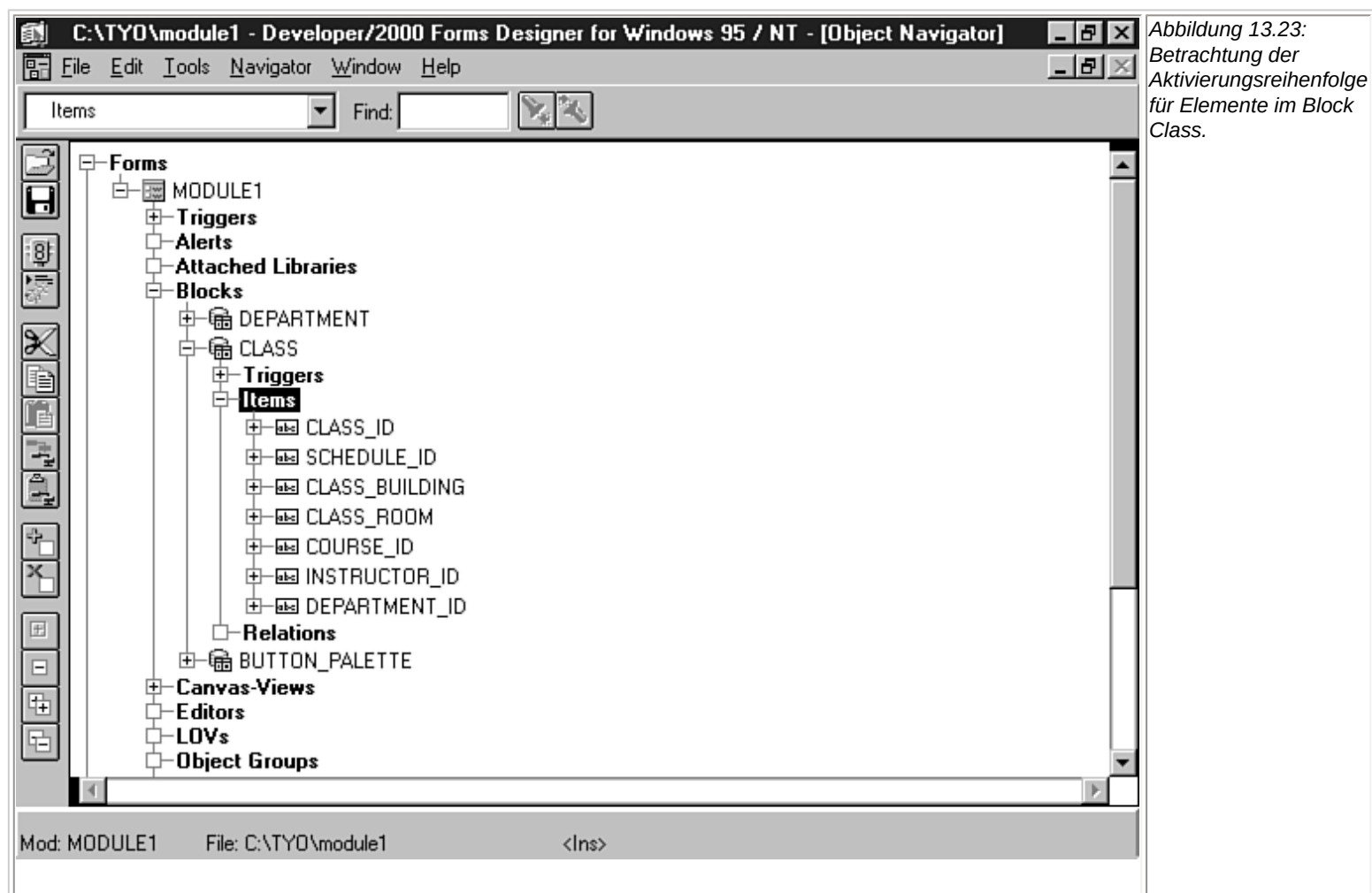


Abbildung 13.22:  
Betrachtung der  
Blockreihenfolge mit  
dem Object  
Navigator.



## Die Aktivierreihenfolge von Blockelementen steuern

Wenn Sie sich zum Block Class bewegen, können Sie mit der Tabulatortaste zum nächsten Element gelangen. Zum vorherigen Element gelangen Sie mit Umschalt+ Tabulator. Beachten Sie, daß Sie, wenn Class\_ID aktiviert ist und Sie die Tabulatortaste drücken, Schedule\_ID aktiviert wird. Als Sie Course\_ID rechts von Class\_ID gestellt haben, hat sich die Reihenfolge, in der die Felder mit der Tabulatortaste angesprungen werden, nicht geändert. Der bequemste Weg, diese Reihenfolge zu ändern, führt über den Object Navigator. Abbildung 13.23 zeigt die vorhandene Reihenfolge der Elemente im Block Class.



Damit Course\_ID zum zweiten Element wird, das mit der Tabulatortaste aktiviert wird, wählen Sie es mit der Maus an, halten Sie die linke Maustaste gedrückt, ziehen Sie den Zeiger über Class\_ID, und lassen Sie die linke Maustaste los.

## Erstellen einer Werteliste

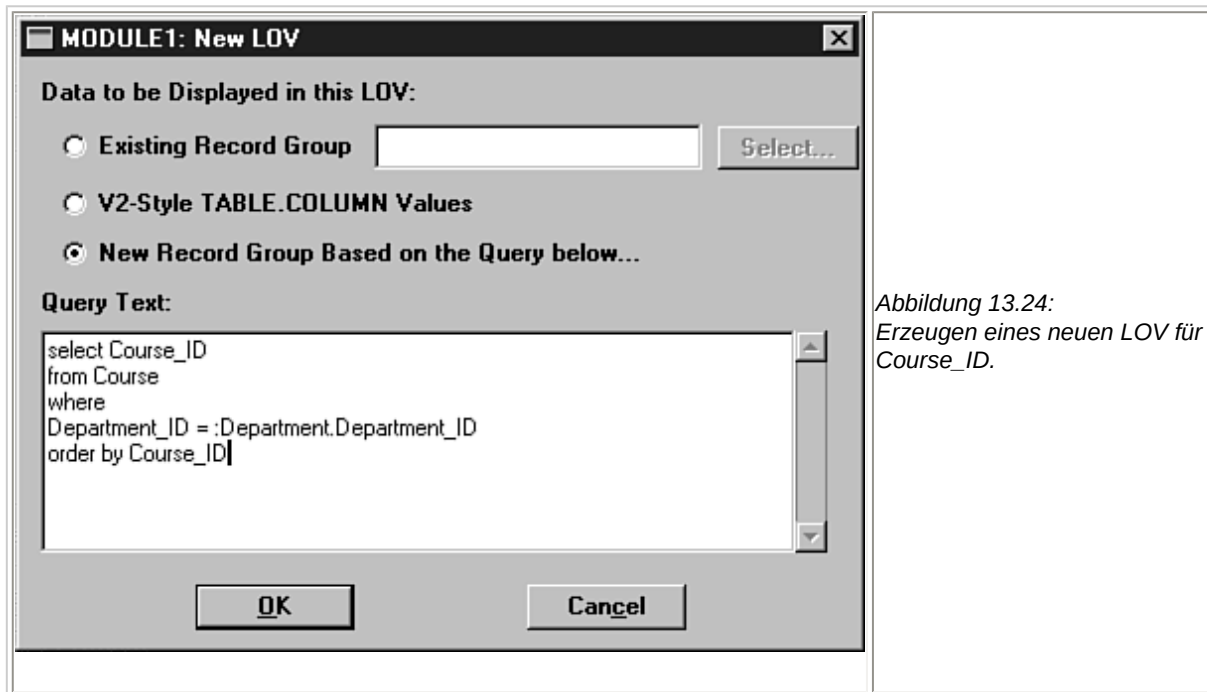
Was könnte noch an diesem Master-/Detail-Formular verbessert werden? Eine Möglichkeit wäre die Verwendung einer LOV für die Elemente mit einer Reihe einzelner Werte. Course\_ID sollte z.B. auf die Werte für die aktuelle Department\_ID beschränkt sein, wie sie in der Tabelle Course zu finden sind. Um unter Verwendung des Object Navigator eine LOV für Course\_ID anzulegen, wählen Sie Knoten der LOV und klicken auf das + in der vertikalen Werkzeugleiste. In dem dann erscheinenden Fenster können Sie die gewünschte, für die neue LOV durchzuführende Abfrage eingeben. Forms-Designer erzeugt dann eine neue Datensatzgruppe aufgrund der eingegebenen Abfrage. Abbildung 13.24 enthält den Text der Abfrage. Beachten Sie, daß Department\_ID auf den Wert beschränkt ist, der im Element Department\_ID des Blocks Department steht. Klicken Sie auf OK, um die LOV zu erstellen.



Der in diesem Listing und der Abbildung aufgeführte Code kann in der .fmb-Datei auf der CD-ROM eingesehen werden. Sie können das gesamte Formular öffnen und alle Objekte und Eigenschaften nachschauen.

**Listing 13.1: Abfrage zur Erstellung einer neuen Werteliste für Course\_ID.**

```
select Course_ID
from Course
where
Department_ID = :Department.Department_ID
order by Course_ID
```



Müssen Sie in einer SQL- oder PL/SQL-Anweisung auf ein Blockelement Bezug nehmen, benutzen Sie die Syntax: *block.item*.

Wenn Sie mit dem Object Navigator die Knoten der Objekte betrachten, fallen Ihnen zwei neue Objekte auf: eine LOV namens LOV15 und eine Datensatzgruppe namens LOV15. Machen Sie sich bitte klar, daß diese Objekte durchaus mit anderen Standardnamen versehen werden können. Um den Namen der Datensatzgruppe zu ändern, wählen Sie sie mit der Maus an und klicken Sie noch einmal mit der linken Maustaste darauf. Der Name erscheint in Blau, und der Zeiger wird zur Einfügemarke. Nennen Sie die Datensatzgruppe COURSE\_ID\_RG, und die LOV COURSE\_ID\_LOV, wie es in Abbildung 13.25 zu sehen ist.



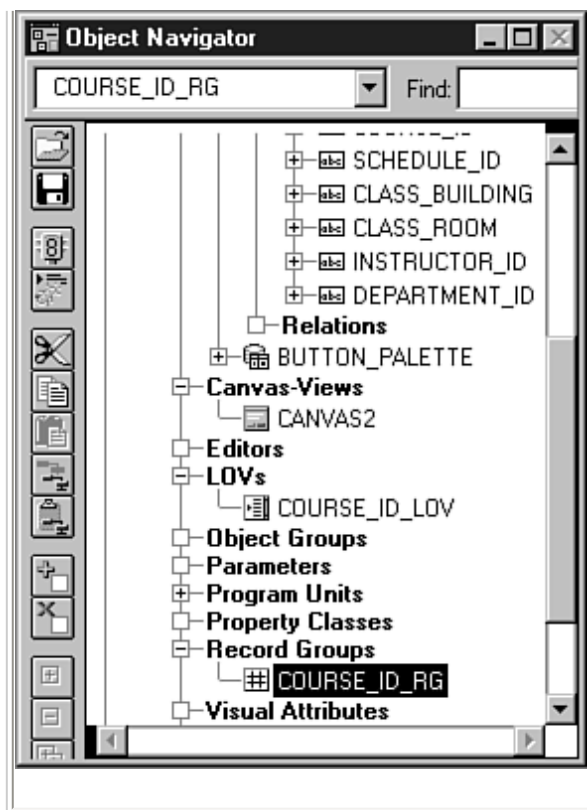


Abbildung 13.25:  
Umbenennen der LOV und der Datensatzgruppe.

Sie möchten noch eine Eigenschaft von `COURSE_ID_LOV` ändern. Dazu wählen Sie `COURSE_ID_LOV` an und dann mit der rechten Maustaste Properties. Blättern Sie nach unten, bis Sie die Eigenschaft Column Mapping erreicht haben. Klicken Sie oben im Eigenschaftfenster auf die More-Schaltfläche (siehe Abbildung 13.26).

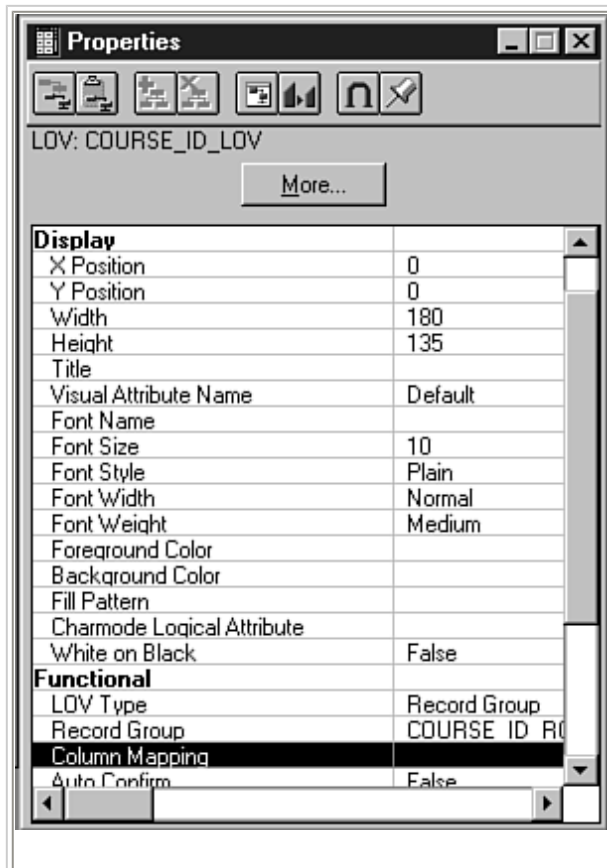
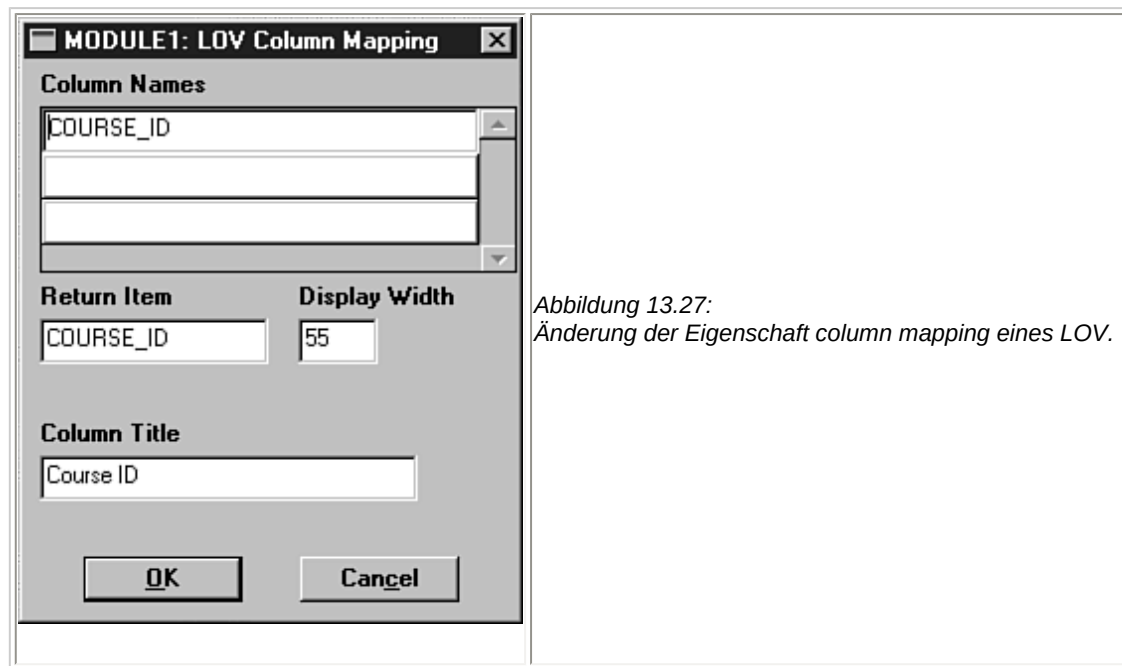


Abbildung 13.26:  
Änderung der Eigenschaften eines LOV.

Das Fenster LOV Column Mapping erscheint mit vier Feldern. Das Feld Column Name bezeichnet die Spalten, die durch Abfrage der Datensatzgruppe zurückgegeben werden. Das Feld Return Item gibt das Blockelement an, in dem der aktuelle Wert im Feld Column Names abgelegt werden soll. Das Feld Display Width zeigt die in der LOV angezeigte Breite des Feldes an. Schließlich enthält das Feld Column Title die Überschrift, so wie sie bei Aufruf der LOV angezeigt wird.

Wie in Abbildung 13.27 zu sehen, geben Sie `Course_ID` in das Feld Return Item ein; dadurch wird die vom Benutzer ausgewählte

Course\_ID in das Element Course\_ID gesetzt. Erhöhen Sie Display Width auf 55 Punkte, damit die LOV-Überschrift für Course\_ID nicht abgeschnitten wird. Ändern Sie die Spaltenüberschrift in Course ID. Klicken Sie auf OK, damit die Veränderungen übernommen werden.



Es gibt noch einige durchzuführende Schritte, ehe die LOV tatsächlich in diesem Formular benutzt wird:

1. Verwenden Sie den Object Navigator, und wählen Sie Course\_ID und mit der rechten Maustaste Properties.
2. Blättern Sie nach unten, bis Sie eine Gruppe von Eigenschaften sehen, die mit Miscellaneous betitelt ist, und wählen Sie die Eigenschaft LOV aus.
3. Wählen Sie oben im Fenster Properties Course\_ID\_LOV aus der Pop-up-Liste, wie in Abbildung 13.28 gezeigt.
4. Durch Anklicken ändern Sie die Eigenschaft LOV for Validation in True; damit wird verhindert, daß ein Wert in das Feld eingegeben wird, der nicht in der LOV vorhanden ist.

Lassen Sie uns das geänderte Formular ausprobieren.

1. Klicken Sie auf die grüne Ampel in der vertikalen Werkzeugleiste, oder wählen Sie den Menüeintrag File | Run.
2. Wenn das Laufzeit-Formular erscheint, maximieren Sie beide Fenster.
3. Führen Sie eine Abfrage durch, indem Sie zweimal auf die Query-Schaltfläche klicken - einmal, um in den Query-Modus zu gelangen und ein zweites Mal, um die Abfrage auszuführen. Im Block Department erscheint die erste Abteilung: Anthropologie. Aber es werden keine Kurse von dieser Abteilung angeboten.
4. Mit der Maus klicken Sie in die erste Zeile des Blocks Class.
5. Geben Sie **222333** in Class\_ID ein.
6. Drücken Sie die Tabulatortaste, um zu Course\_ID zu gelangen. In der Statuszeile am unteren Bildschirmrand müßten Sie <List> stehen sehen, dies zeigt an, daß es für Course\_ID eine LOV gibt.
7. Um die LOV anzuzeigen, drücken Sie F9). In der linken, oberen Ecke des Fensters wird die LOV angezeigt (siehe Abbildung 13.29).

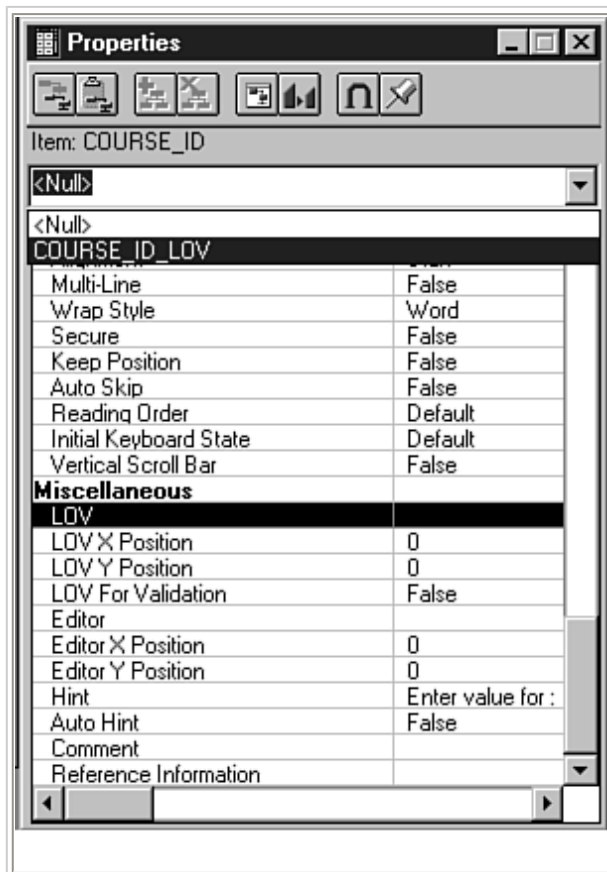


Abbildung 13.28:  
Anpassen der Eigenschaften des Elements  
Course\_ID auf die LOV.

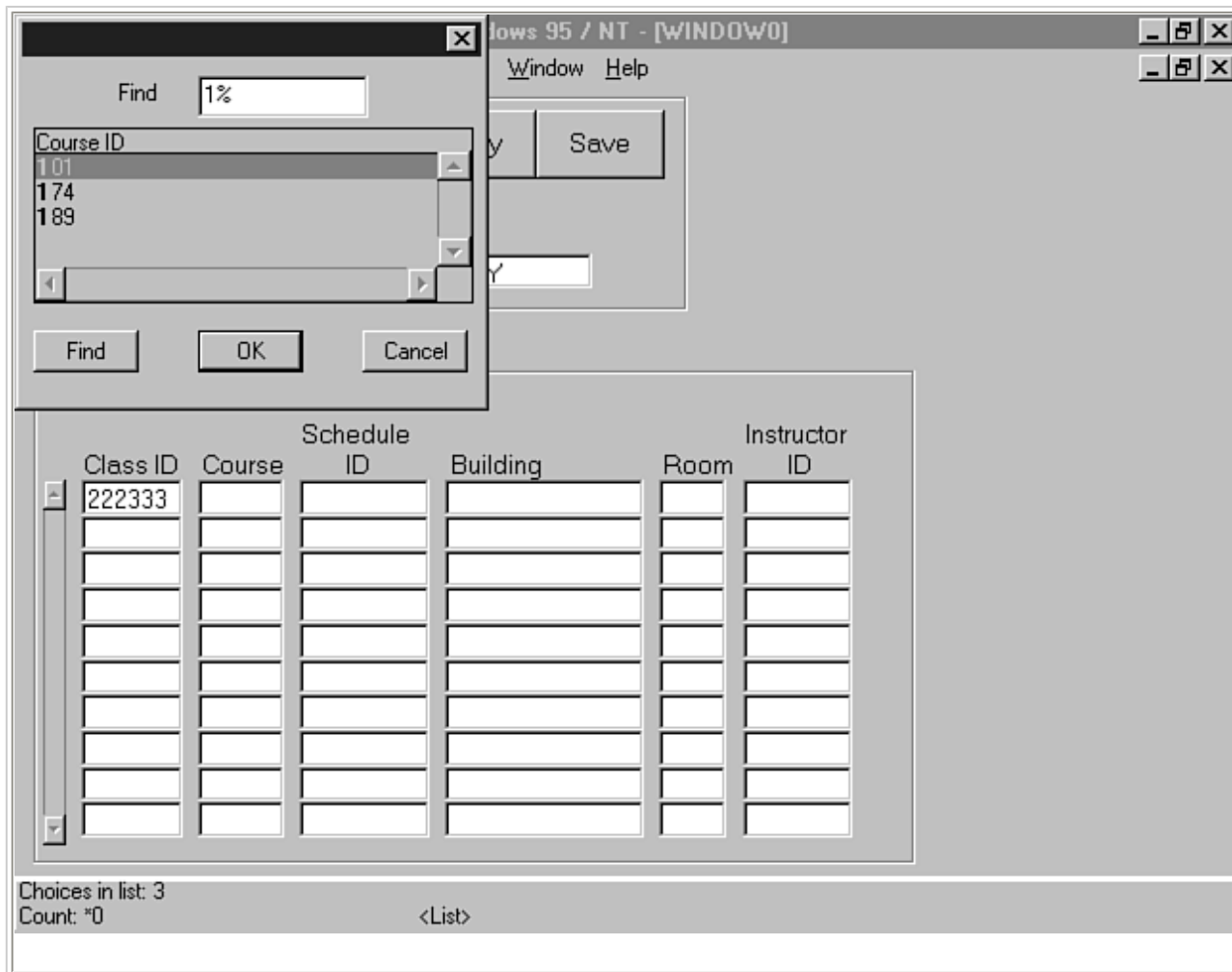


Abbildung 13.29:  
Anzeigen des LOV  
für das Element  
Course\_ID.

Blättern Sie nach unten bis zum Kurs 189, und drücken Sie die Eingabetaste; 189 wird im Element Course\_ID erscheinen. Wenn Sie einen in der Abteilung nicht existierenden Kurs eingeben, z.B. 1234, und die Tabulatortaste drücken, um zum nächsten Element zu gelangen, zeigt das Formular die LOV an, sowie die folgende Meldung in der Statuszeile: FRM:40212: Invalid value for field COURSE\_ID (siehe Abbildung 13.30).

The screenshot shows the 'Developer/2000 Forms Runtime for Windows 95 / NT - [WINDOW0]' window. The menu bar includes Action, Edit, Block, Field, Record, Query, Window, and Help. Below the menu is a toolbar with navigation buttons (<<, <, >, >>) and 'Query' and 'Save' buttons. The form contains two text fields: 'Department ID' with the value 'ANTHRO' and 'Department Name' with the value 'ANTHROPOLOGY'. Below these is a table with the following columns: Class ID, Course, Schedule ID, Building, Room, and Instructor ID. The first row of the table has the values 222333, 1234, and empty cells for the remaining columns. A status bar at the bottom displays the error message: 'FRM-40212: Invalid value for field COURSE\_ID. Count: \*0 <List>'. To the right of the screenshot, a caption reads: 'Abbildung 13.30: Die LOV erzwingt gültige Werte für das Element Course\_ID.'



Selbst wenn Sie zur Gültigkeitsüberprüfung keine LOV einsetzen, setzt die Oracle-Datenbank Integritätsregeln durch. Trotzdem hat eine LOV den Vorteil, daß sie eine Gruppe erlaubter Werte zur Verfügung stellt und der Benutzer sich die möglichen Werte für ein Element nicht merken muß.

Bevor Sie diese Lektion abschließen, speichern Sie das Formular unter dem Namen Department\_Class.fmb, indem Sie den Menüeintrag File | Save wählen.

## Zusammenfassung

In dieser Lektion haben Sie folgendes gelernt:

- Mit Forms-Designer lassen sich leicht Master-/Detail-Formulare erstellen.
- Durch die Erzeugung eines Detail-Blocks wird der Master-Block definiert.
- Mit Layout-Editor ändern Sie das Aussehen Ihrer Formulare.
- Die Eigenschaften mehrerer Elemente, z.B. deren Schriftart, lassen sich mit einem Schritt ändern, wenn Sie sie auswählen.
- Mit dem Layout-Editor können Sie Formularen grafische Elemente hinzufügen, indem Sie Werkzeuge aus der vertikalen Werkzeugleiste auswählen.
- Bei der Erzeugung einer LOV können Sie automatisch eine Datensatzgruppe für die LOV erzeugen, indem Sie für die Datensatzgruppe eine Abfrage bestimmen.
- Um eine LOV einzusetzen, muß die Eigenschaft LOV für ein Element eingestellt sein. Wollen Sie, daß die LOV Benutzereingaben auf deren Gültigkeit überprüft, muß die Eigenschaft LOV for Validation auf den Wert True gesetzt sein.

## Wie geht es weiter?

An Tag 14, »Developer/2000: Anwendungsentwicklung mit Hilfe von Oracle Forms«, lernen Sie, ein Menü zu erstellen, einem Block Elemente hinzuzufügen, die abgeleitete Informationen anzeigen, Trigger auf Formular-, Block- und Elementebene zu erstellen und ein Formular aus einem anderen Formular aufzurufen und dem aufgerufenen Formular einen Parameter zu übergeben.

## Fragen und Antworten

Frage:

Kann ein Master-/Detail-Formular aus mehr als einem Detail-Block bestehen?

Antwort:

*Ja. Sie erstellen einfach jeden gewünschten Detail-Block, und im Register Master-/Detail bestimmen Sie den Master-Block.*

Frage:

Können sich mehrere LOVs auf dieselbe Datensatzgruppe beziehen?

Antwort:

*Ja. Es gibt keine Einschränkung, die für jede LOV eine eigene Datensatzgruppe verlangt.*

Frage:

Kann ein Formular mehrere Fenster und Canvas-Ansichten haben?

Antwort:

*Ja. Ein einzelnes Formular kann mehrere Fenster und Canvas-Ansichten haben. Wie Sie jedoch in der nächsten Lektion sehen werden, ist es für gewöhnlich sinnvoller, eine Anwendung mit mehreren Formularen zu erstellen, in der jedes Formular nur ein Fenster und nur eine Canvas-Ansicht hat.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

### Test

1. Richtig oder falsch? Ein Master-Block kann nicht auf einer Datensicht basieren, es muß eine Tabelle sein.
2. Richtig oder falsch? Man kann das Erscheinungsbild eines Elements mit dem Layout-Editor oder durch Ändern der Eigenschaften des Elements verändern.
3. Wie kann die Reihenfolge der Aktivierung in einer Gruppe von Elementen geändert werden?

## Übungen

1. Erstellen Sie ein Master-/Detail-Fenster, um Schedule\_Type und Schedule-\_Type\_Details einzugeben.
2. Erstellen Sie im Formular Department\_Class eine LOV für Schedule\_ID, die die Benutzereingabe überprüft.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖Kapitel❖ [Top](#) [Inhalt](#) [Index](#) **SAMS** Kapitel❖

# 14

# Developer/2000: Anwendungsentwicklung mit Hilfe von Oracle Forms

[Eine Übersicht über Trigger](#)

[Abrufen von Werten aus einer anderen Tabelle mit Hilfe eines Trigger](#)

[Eine Abfrage mit Hilfe eines Trigger auf Formularebene](#)

[Benutzereingaben mit Hilfe eines Trigger auf ihre Gültigkeit überprüfen](#)

[Elemente eines Menüs](#)

[Entwickeln eines Menüs](#)

[Integrieren neuer Menüs](#)

[Entwickeln einer Multiform-Anwendung](#)

[Erzeugen eines Windows-Shortcuts zum Starten einer Anwendung](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

In den beiden vorhergehenden Lektionen haben Sie zwei Formulare erstellt: ein auf einer einzelnen Tabelle basierendes Formular und ein Master-/Detail-Formular. Außerdem haben Sie gelernt, wie Sie das Standardlayout verändern können, das vom Forms-Designer bei der Erstellung eines neuen Blocks erzeugt wird.

Dieses ist Ihre letzte Lektion zu Oracle Forms. Hier werden Sie die Verwendung von PL/SQL zur Erstellung von *Anwendungs*-Trigger für eine Reihe von Einsatzzwecken untersuchen. Und Sie werden die Entwicklungsschritte für eine Anwendung mit mehreren

Formularen durchgehen. Gleichzeitig lernen Sie, wie ein Menü erstellt wird, aus dem unterschiedliche Formulare aufgerufen werden.

## Eine Übersicht über Trigger



*Ein Trigger ist eine PL/SQL-Anweisung, die ausgeführt wird, wenn ein Ereignis eintritt. Das Ereignis kann direkt mit einer vom Benutzer durchgeführten Aktion zusammenhängen, wie das Betätigen einer Schaltfläche. Oder das Ereignis ist indirekt mit einer vom Benutzer durchgeführten Aktion verbunden, wie der Zeitraum vor der Ausführung einer Abfrage.*

Trigger können auf drei Ebenen definiert werden:

- Auf Formularebene
- Auf Blockebene
- Auf Elementebene

Häufig enthält ein Formular auf jeder Ebene einen oder mehrere Trigger.

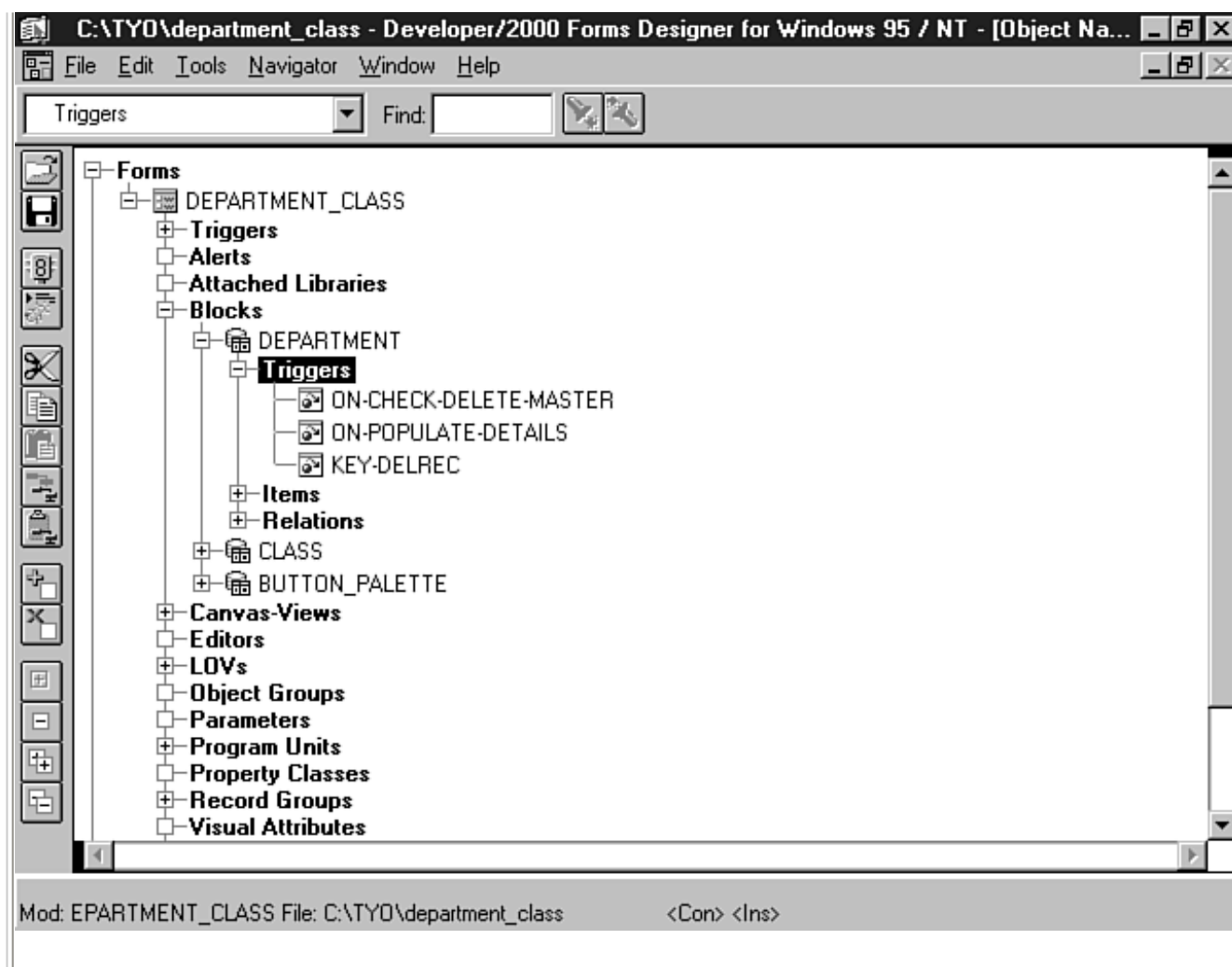
### Standard-Trigger

In der vorherigen Lektion haben Sie ein Master-/Detail-Formular namens Department\_Class erzeugt. Als Sie die einzelnen Blöcke erzeugt haben, haben Sie die Kontrollkästchen im Register Layout markiert, um anzugeben, daß Integritätsregeln eingehalten werden sollen. Wenn Sie den Object Navigator zur Erweiterung des Blocks Department verwenden, werden Sie drei Trigger sehen, die erzeugt wurden, als Sie den Block Department definiert haben (siehe Abbildung 14.1).

- ON-CHECK-DELETE MASTER
- ON-POPULATE-DETAILS
- KEY-DELREC

*Abbildung 14.1:  
Trigger auf  
Blockebene, mit  
denen die  
Datenintegrität  
erzwungen wird.*





Wie erzwingen diese Trigger die referentielle Integrität? Schauen wir uns den Trigger KEY-DELREC näher an. Dieser Trigger wird ausgelöst, wenn der Benutzer die Entfernen-Taste drückt. Listing 14.1 enthält den Text dieses Trigger. Beachten Sie, daß der Trigger sich aus zwei anonymen PL/SQL-Blöcken zusammensetzt. Der erste Block stellt fest, ob es mit der zu löschenden Abteilung verbundene Kurse gibt, und, wenn dem so ist, wird eine Fehlermeldung ausgegeben und eine Exception ausgelöst. Der zweite Block ähnelt dem ersten sehr; er stellt fest, ob es mit der zu löschenden Abteilung verbundene Lehrkräfte gibt, und, wenn das zutrifft, werden die entsprechenden Schritte ausgeführt. Im Object Navigator kann ein Trigger durch einen Doppelklick auf denselben betrachtet werden.

**Listing 14.1: Code für den Trigger key-delrec zum Block Department.**

```
declare
cursor primary_cur is select 'x' from FLUGLE.CLASS where DEPARTMENT_ID =
:DEPARTMENT.DEPARTMENT_ID; primary_dummy char(1);
begin
if ( ( :DEPARTMENT.DEPARTMENT_ID is not null ) ) then open primary_cur;
fetch primary_cur into primary_dummy;
if ( primary_cur%found ) then
message('Cannot delete master record when matching detail [ccc]records exist.');
```

```
close primary_cur;
raise form_trigger_failure;
end if;
close primary_cur;
end if;
end;
declare
cursor primary_cur is select 'x' from FLUGLE.COURSE where DEPARTMENT_ID =
:DEPARTMENT.DEPARTMENT_ID; primary_dummy char(1);
begin
if ( ( :DEPARTMENT.DEPARTMENT_ID is not null ) ) then open primary_cur;
```

```
fetch primary_cur into primary_dummy;
if ( primary_cur%found ) then
message('Cannot delete master record when matching detail [ccc]records exist. ');
close primary_cur;
raise form_trigger_failure;
end if;
close primary_cur;
end if;
end;
_ XE "triggers:default:KEY-DELREC trigger code listing" __ XE "2" __ XE
"listings:triggers:KEY-DELREC" _
declare
cursor primary_cur is select 'x' from FLUGLE.INSTRUCTOR
where DEPARTMENT_ID = :DEPARTMENT.DEPARTMENT_ID;
primary_dummy char(1);
begin
if ( ( :DEPARTMENT.DEPARTMENT_ID is not null ) ) then
open primary_cur;
fetch primary_cur into primary_dummy;
if ( primary_cur%found ) then
message('Cannot delete master record when matching detail
[ccc]records exist. ');
close primary_cur;
raise form_trigger_failure;
end if;
close primary_cur;
end if;
end;
delete_record;
```

## Abrufen von Werten aus einer anderen Tabelle mit Hilfe eines Trigger

Jetzt wollen wir ein Beispiel durcharbeiten, wie in dem Formular Depart-ment\_Class ein Trigger eingesetzt werden kann. Schauen Sie sich den Block Class an, in dem zwar die Course\_ID angezeigt wird, der Benutzer aber keine Möglichkeit hat, mehr über den Kurs zu erfahren, wenn er sich die Course\_IDs nicht gemerkt hat. Es würde dem Benutzer wahrscheinlich helfen, wenn der Titel des Kurses angezeigt würde.

1. Benutzen Sie den Layout-Editor, und bewegen Sie Building\_ID, Room und Instructor\_ID nach rechts, damit der Kurstitel rechts von Course\_ID angezeigt werden kann.
2. Dann wählen Sie das Werkzeug Text Item aus der vertikalen Werkzeugleiste, und klicken Sie mit der linken Maustaste rechts von Course\_ID (siehe Abbildung 14.2).  
Kümmern Sie sich nicht um die Größe und den Zwischenraum des neuen Textelements; Sie werden es gleich ändern.
3. Klicken Sie mit der rechten Maustaste und dann auf Properties, um die Eigenschaften des neuen Textelements zu ändern.
4. Geben Sie für die Eigenschaft Name Course\_Title ein. Blättern Sie nach unten, bis zu der Eigenschaftengruppe mit dem Namen Navigation. Setzen Sie die Eigenschaft Navigable auf False, indem Sie die Eingabetaste drücken.
5. Das Textelement Course\_Titel wird nur zur Anzeige des Kurstitels benutzt; es gibt für den Benutzer keinen Grund, dieses zu aktivieren. Blättern Sie weiter nach unten zu der Eigenschaft Database.
6. Setzen Sie die Eigenschaft Base Table auf False, indem Sie die Eingabetaste drücken (siehe Abbildung 14.3).  
Der Grund, die Eigenschaft Base Table auf False zu setzen, besteht darin, daß der Block Class auf der Tabelle Class basiert und Course\_Title keine Spalte dieser Tabelle ist.
7. Kehren wir zum Object Navigator zurück. Ziehen Sie das Textelement COURSE\_TITLE neben das Textelement COURSE\_ID.

8. Wählen Sie das Textelement COURSE\_ID an, klicken Sie mit der rechten Maustaste und dann auf Properties. blättern Sie nach unten zur Eigenschaft Height Property.
9. Im Fenster des Object Navigators klicken Sie das Textelement COURSE\_TITLE an. Beachten Sie, daß das Fenster Properties jetzt die Eigenschaften für COURSE\_TITLE anzeigt und daß immer noch Height Property aktiviert ist.
10. Setzen Sie für die Höhe den gleichen Wert wie für COURSE\_ID: 14.

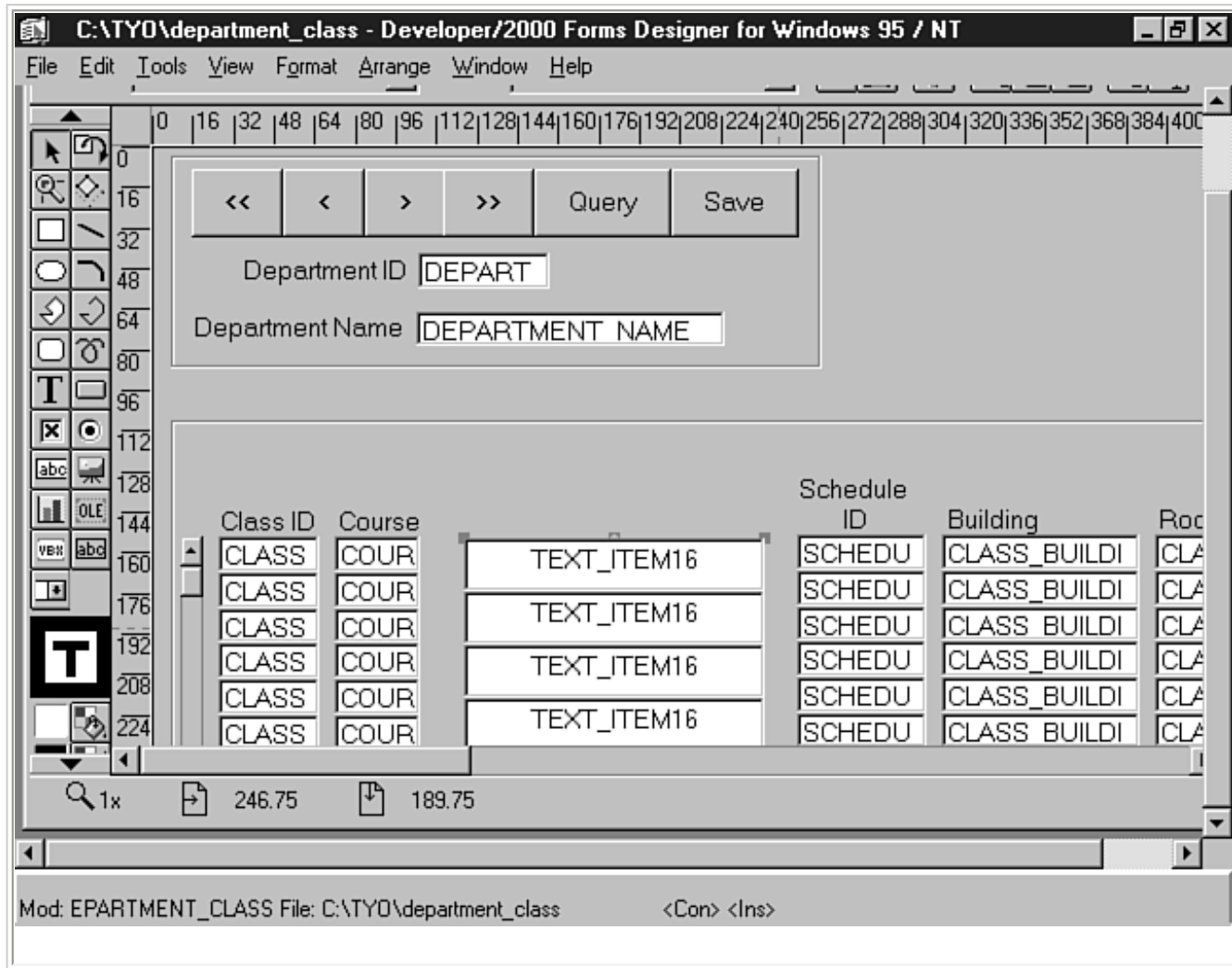


Abbildung 14.2:  
Hinzufügen eines  
Textelements  
zum Block Class.



Abbildung 14.3:  
Setzen der Eigenschaften für das  
Textelement Course\_Title.

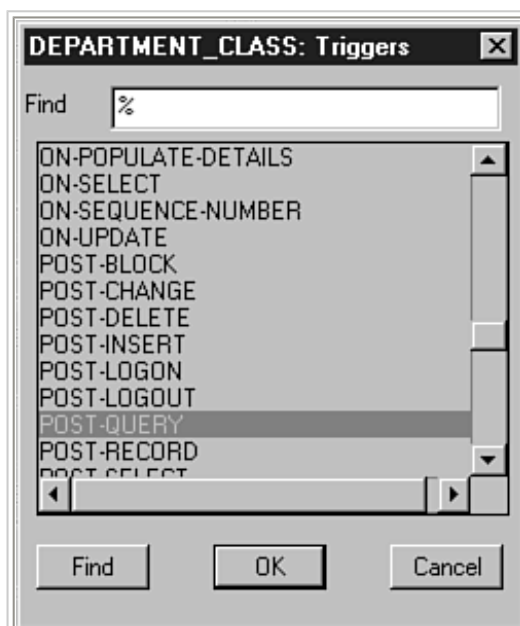


Abbildung 14.4:  
Erstellen eines post-query-Trigger für den Block  
Department.

Da Sie jetzt eine Stelle zur Anzeige der Kursüberschrift haben, benötigen Sie noch einen Trigger, der diese anzeigen läßt.

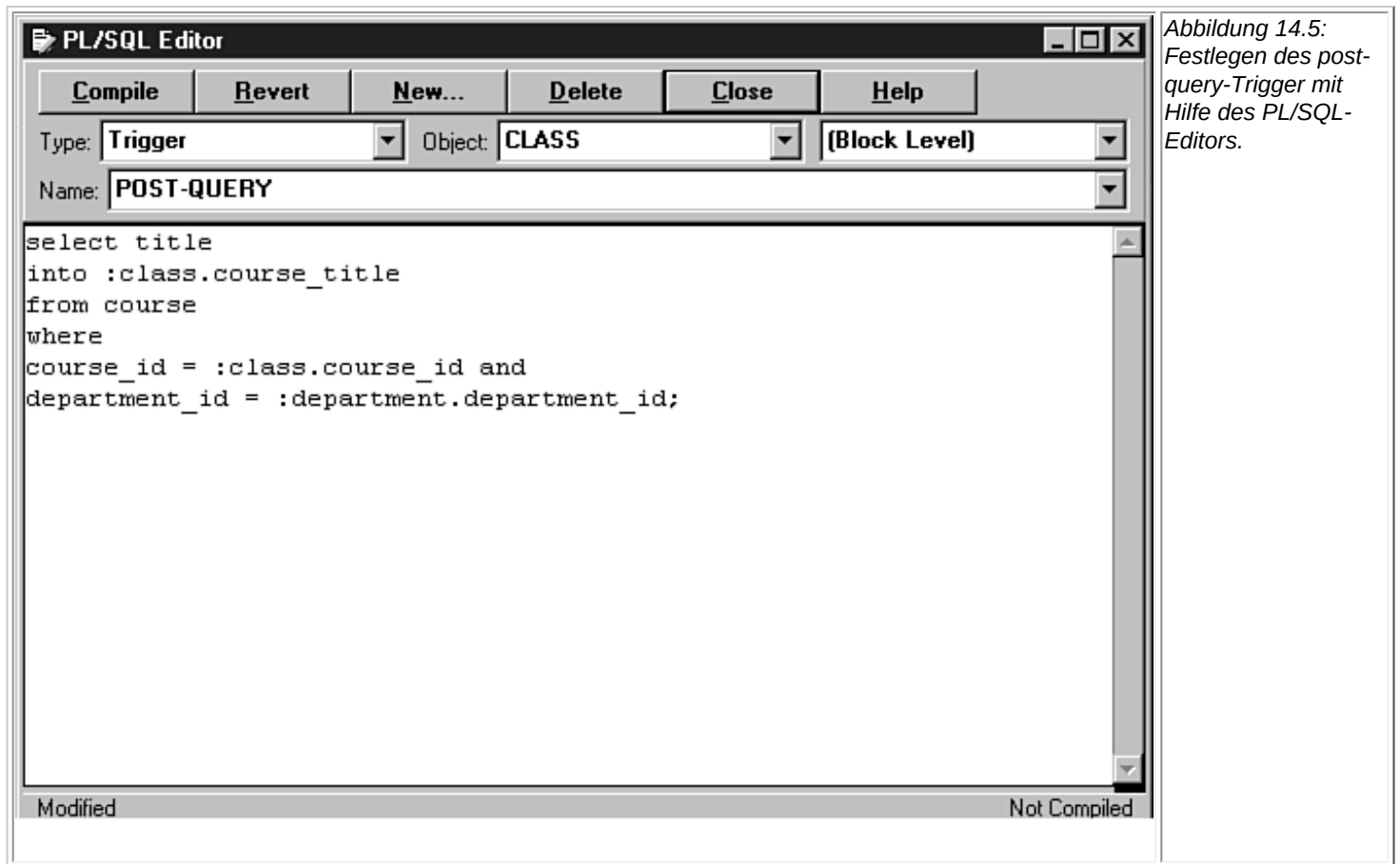
1. Verwenden Sie den Object Navigator, um die zum Block Department gehörenden Trigger zu erweitern, und klicken Sie auf das + in der vertikalen Werkzeugleiste, um einen neuen Trigger auf Blockebene zu erzeugen.

2. In dem neu erscheinenden Fenster können Sie das Ereignis auswählen, das den neuen Trigger auslösen soll; wählen Sie POST-QUERY, und klicken Sie auf OK (siehe Abbildung 14.4).

3. Der PL/SQL-Editor erscheint. Geben Sie wie Abbildung 14.5 gezeigt, eine SELECT-Anweisung ein, die Course\_Titel für den Wert aus Course\_ID, der im Element Course\_ID des Blocks Class gefunden wird, abrufen, und Department\_ID, der im Element Department\_ID des Blocks Department gefunden wird.

4. Geben Sie die SELECT-Anweisung aus Listing 14.2 ein, und klicken Sie auf Compile. Wenn Sie die Anweisung richtig

eingetragen haben, sollten Sie keine Fehlermeldung angezeigt bekommen. Klicken Sie auf Close.



Beachten Sie, daß vor allen Bezügen auf Blöcke oder Elemente ein Komma steht.

#### Listing 14.2: Festlegen eines post-query-Trigger.

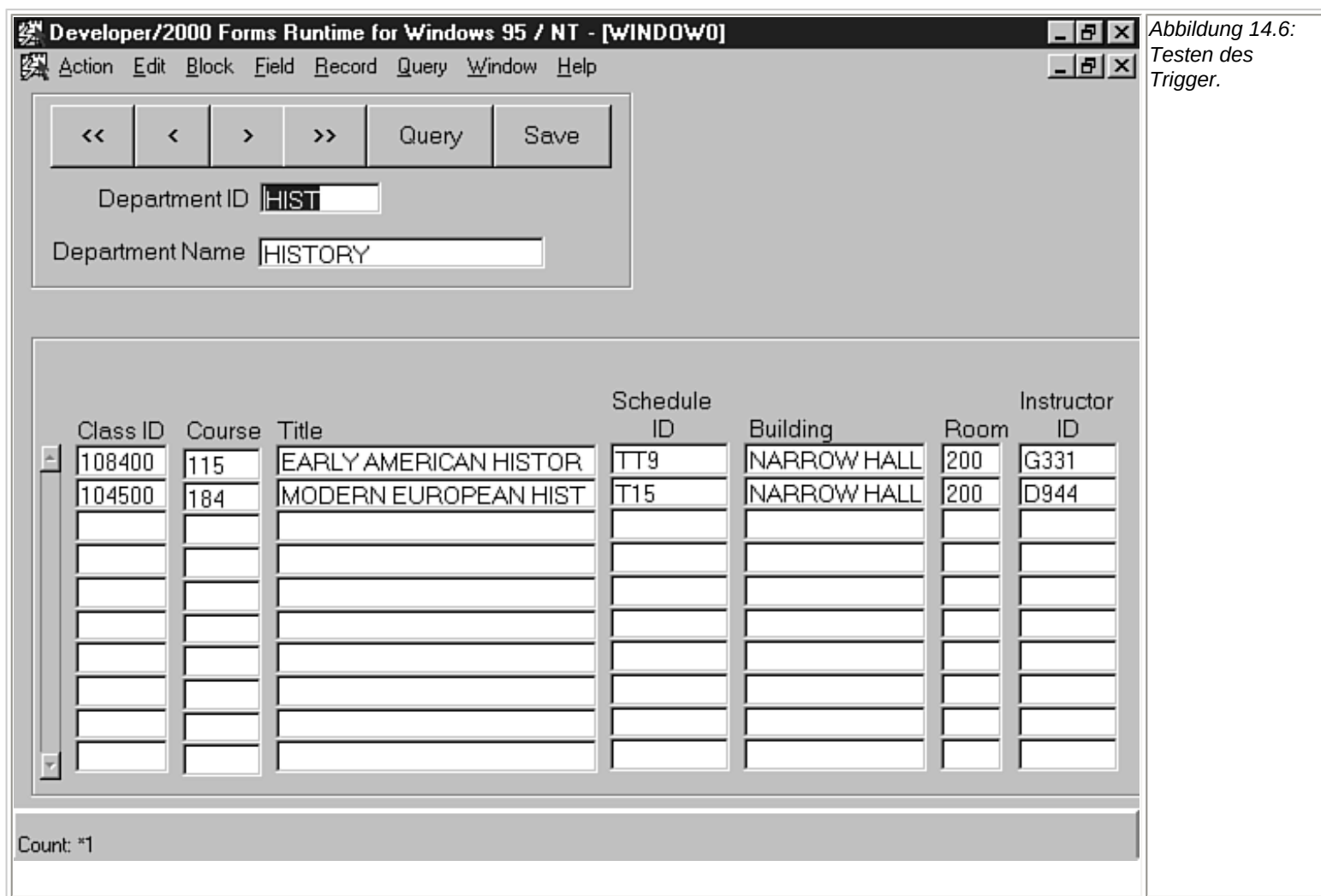
```
select title
into :class.course_title
from course
where
course_id = :class.course_id and
department_id = :department.departmen_id;
```

Versuchen wir, dieses Formular auszuführen, um zu sehen, ob der Trigger korrekt arbeitet.

1. Klicken Sie auf die grüne Ampel in der vertikalen Werkzeugleiste.
2. Um den Block Department abzufragen, klicken Sie auf die Query-Schaltfläche.
3. Für Department\_ID geben Sie **HIST** ein.
4. Klicken Sie erneut auf die Query-Schaltfläche. Wie Sie in Abbildung 14.6 sehen können, hat der Trigger wie erwartet gearbeitet - der Titel für jeden Kurs wurde aus der Tabelle Course abgerufen.

Sehen wir jetzt, was passiert, wenn ein neuer Kurs eingegeben wird.

1. Klicken Sie auf Class\_ID. Wie es in Abbildung 14.7 gezeigt wird, wird der Title für Course\_ID 199 *nicht* abgerufen. Sie sehen, daß für Course\_ID ein Trigger auf Elementebene fehlt.
2. Verlassen Sie Forms Runtime.
3. Im Object Navigator erweitern Sie das Element Course\_ID.
4. Im Knoten des zu Course\_ID gehörenden Trigger klicken Sie auf das + in der vertikalen Werkzeugleiste, um einen Trigger hinzuzufügen.
5. Im jetzt erscheinenden Fenster wählen Sie den Trigger WHEN-VALIDATE-ITEM. Obwohl die gleiche SELECT-Anweisung wie für den Trigger auf Blockebene - POST-QUERY - für diesen Trigger auf Elementebene benutzt wird, wird die Anwendung in eine anonymen Block gesetzt, wie es in Abbildung 14.8 zu sehen ist.
6. Zusätzlich sollten Sie dem Trigger einen Exception-Abschnitt hinzufügen, um die Situation zu verarbeiten, daß der Benutzer eine ungültige Course\_ID eingibt. Da Sie jedoch vorgegeben haben, daß Course\_ID\_LOV zur Gültigkeitsüberprüfung des Felds Course\_ID benutzt werden soll, sollte diese Exception nie auftreten.



Developer/2000 Forms Runtime for Windows 95 / NT - [WINDOW0]

Action Edit Block Field Record Query Window Help

<< < > >> Query Save

Department ID HIST

Department Name HISTORY

| Class ID | Course | Title                 | Schedule ID | Building    | Room | Instructor ID |
|----------|--------|-----------------------|-------------|-------------|------|---------------|
| 108400   | 115    | EARLY AMERICAN HISTOR | TT9         | NARROW HALL | 200  | G331          |
| 104500   | 184    | MODERN EUROPEAN HIST  | T15         | NARROW HALL | 200  | D944          |
| 109999   | 199    |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |

Count: \*2

Abbildung 14.7:  
Post-Query  
funktioniert nicht,  
wenn eine neuer  
Datensatz  
eingefügt wird.

C:\TYO\department\_class - Developer/2000 Forms Designer for Windows 95 / NT

File Edit Tools Window Help

PL/SQL Editor

Compile Revert New... Delete Close Help

Type: Trigger Object: CLASS COURSE\_ID

Name: WHEN-VALIDATE-ITEM

```
begin
  select title
  into :class.course_title
  from course
  where
    course_id = :class.course_id and
    department_id = :department.department_id;
exception
  when no_data_found then
    message ('Invalid Course ID. Please enter a valid Course ID.');
```

end;

Not Modified Successfully Compiled

Mod: EPARTMENT\_CLASS File: C:\TYO\department\_class <Con> <Ins>

Abbildung 14.8:  
Festlegen des  
when-validate-  
item-Trigger für  
das Element  
Course\_ID.

**Listing 14.3: when-validate-item-Trigger für das Feld Course\_ID.**

```
begin
select title
into :class.course_title
from course
where
course_id = :class.course_id and
department_id = :department.department_id;
exception
when no_data_found then
message ('Invalid Course ID. Please enter a valid Course ID. ');
raise form_trigger_failure;
end;
```

Um den Trigger auf Elementebene zu testen, lassen wir das Formular erneut ausführen.

1. Um den Block Department abzufragen, klicken Sie auf die Query-Schaltfläche.
2. Für Department\_ID geben Sie **HIST** ein.
3. Klicken Sie erneut auf die Query-Schaltfläche.
4. Klicken Sie Class\_ID an.
5. Geben Sie in Class\_ID **109999** ein.
6. Gehen Sie mit der Tabulatortaste zu Course\_ID, und geben Sie **199** ein.
7. Drücken Sie wieder die Tabulatortaste. Abbildung 14.9 zeigt, daß der Trigger korrekt arbeitet; der richtige Titel für History 199 - »WORKSHOP ON JEFFERSON« - wird in Course\_Title angezeigt.

*Abbildung 14.9:  
Testen des when-  
validate-item-  
Trigger für das  
Feld Course\_ID.*



| Class ID | Course | Title                 | Schedule ID | Building    | Room | Instructor ID |
|----------|--------|-----------------------|-------------|-------------|------|---------------|
| 108400   | 115    | EARLY AMERICAN HISTOR | TT9         | NARROW HALL | 200  | G331          |
| 104500   | 184    | MODERN EUROPEAN HIST  | T15         | NARROW HALL | 200  | D944          |
| 109999   | 199    | WORKSHOP ON JEFFERSO  |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |
|          |        |                       |             |             |      |               |

Count: \*2

## Eine Abfrage mit Hilfe eines Trigger auf Formularebene

- Wir wollen einen weiteren Beispieltrigger durcharbeiten. Führen Sie die folgenden Schritte aus, um ein Formular zu erstellen, das, wenn es aufgerufen wird, automatisch eine Abfrage ausführt:
1. Erstellen Sie mit Forms-Designer ein neues Formular, indem Sie den Menüeintrag File | New | Form wählen.
  2. Nennen Sie das neue Formular INSTRUCTOR, und erstellen Sie einen neuen Block auf Basis der Tabelle Instructor. Wenn Sie die Gestaltung Ihres Formulars abgeschlossen haben, sollte es dem aus Abbildung 14.10 ähneln.
  3. Wenn Sie Object Navigator verwenden, um das Formular Instructor zu betrachten, erkennen Sie, daß auf Formularebene, direkt unterhalb des Formular selbst, keine Trigger vorhanden sind. Um einen Trigger auf Formularebene zu erstellen, wählen Sie den Trigger-Knoten an, und klicken Sie auf das + in der vertikalen Werkzeugleiste.
  4. Wählen Sie im dann erscheinenden Fenster WHEN-WINDOW-ACTIVATED, und klicken Sie auf OK. Der Trigger wird ausgelöst, wenn das Formular aufgerufen wird.
  5. Der PL/SQL-Editor wird aufgeblendet, und Sie brauchen nur eine einzige Zeile SQL-Code für diesen Trigger zu schreiben: einen execute\_query-Aufruf; der aus einer in Oracle Forms enthaltenen Prozedur besteht, die eine Abfrage im aktuellen Block, hier Instructor, ausführt.
  6. Klicken Sie auf Close, und testen Sie das Formular, indem Sie auf die grüne Ampel in der vertikalen Werkzeugleiste klicken. Wie Sie der Abbildung 14.11 entnehmen können, wird das Formular bei seinem Aufruf jedesmal automatisch ausgeführt.

C:\TYO\instructor - Developer/2000 Forms Designer for Windows 95 / NT - [INSTRUCTOR: CA...

File Edit Tools View Format Arrange Window Help

Canvas: CANVAS2 Block: INSTRUCTOR

<< < > >> Query Save

Instructor ID INSTRUCTOR\_ID Department ID DEPARTMENT\_I

Last Name LAST\_NAME First Name FIRST\_NAME

MI M Position POSITION

Telephone TELEPHO Fax FAX

Email EMAIL

Mod: INSTRUCTOR File: C:\TYO\instructor <Con> <Ins>

Abbildung 14.10:  
Erstellen eines  
Formulars für die  
Instructor-Tabelle.

Developer/2000 Forms Runtime for Windows 95 / NT - [WINDOW0]

Action Edit Block Field Record Query Window Help

<< < > >> Query Save

Instructor ID M101 Department ID ANTHRO

Last Name HITCHCOCK First Name BORIS

MI Position PROFESSOR

Telephone 805555012 Fax 805555032

Email BHITCHCOCK@FLUGLE.EDU

Count: 1 v

Abbildung 14.11:  
Testen des when-  
window-activated-  
Trigger.

## Benutzereingaben mit Hilfe eines Trigger auf ihre Gültigkeit überprüfen

In diesem Abschnitt untersuchen Sie eine weitere Verwendung eines Trigger auf Elementebene: zur Gültigkeitsüberprüfung von Benutzereingaben. Nehmen Sie an, der Schulleiter hätte Sie informiert, daß eine zusätzliche Lehrkraft für eine Abteilung, der schon mindestens ein Professor als Lehrkraft angehört, nicht auch ein Professor sein kann - er oder sie können höchstens Assistent oder beratender Professor sein.

Dazu verändern Sie den bestehenden Trigger WHEN-WINDOW-ACTIVATED für das Textelement Position, indem Sie es erweitern und einen Doppelklick auf das Symbol links des Trigger WHEN-WINDOW-ACTIVATED ausführen. Aus Abbildung 14.12 können Sie ersehen, daß der bestehende Trigger die gleiche Logik aufweist, wie die CHECK-Regel für die Spalte Position in der Instructor-Tabelle. Dieser Trigger wurde automatisch erzeugt, als bei der Erstellung des Blocks Integrity Constraints gewählt haben.

Listing 14.4 zeigt den Trigger WHEN-WINDOW-ACTIVATED, nachdem er entsprechend verändert wurde.

### *Listing 14.4: Geänderter when-validate-item-Trigger.*

```
declare
num_full_professors integer;
begin
if not( :INSTRUCTOR.POSITION IN ('ASSISTANT PROFESSOR', [ccc]'ASSOCIATE PROFESSOR',
'PROFESSOR') ) then
message( 'WHEN-VALIDATE-ITEM trigger failed on field - ' [ccc]|| :system.trigger_field );
raise form_trigger_failure;
end if;
select count(*) into num_full_professors
from Instructor
where Department_ID = :Instructor.Department_ID and [ccc]Position = 'PROFESSOR';
if num_full_professors > 0 and :Instructor.Position = 'PROFESSOR' then
message ('Cannot add another full professor to this department.');
```



Als erstes stehen die PL/SQL-Anweisungen in einem anonymen Block, und eine Variable namens num\_full\_professor wird deklariert. Im ersten Teil des Blocks werden die CHECK-Regeln durchgesetzt. Dann wird eine SELECT-Anweisung eingesetzt, um die Anzahl der schon besetzten vollen Professorenstellen in der Abteilung zu ermitteln. Ist diese Zahl größer als Null, wird eine Meldung angezeigt und eine integrierte Exception, FORM\_TRIGGER\_FAILURE, wird ausgelöst, die den Benutzer daran hindert, fortzufahren, solange der Wert nicht berichtigt wurde.

Wir wollen den Trigger jetzt testen. Wählen Sie den Menüeintrag Record | Insert, um eine neue Lehrkraft einzurichten. Versuchen Sie, eine neue Lehrkraft für die Anthropologie-Abteilung einzugeben, die schon zwei volle Professorenstellen besitzt, Boris Hitchcock und Ranier Poulson. Abbildung 14.13 zeigt den Trigger in Aktion. Wie beabsichtigt, hindert der Trigger Sie nicht daran, für die Mathematikabteilung eine neue Professorenstelle einzugeben, da in dieser noch keine volle Professorenstellen besetzt ist.

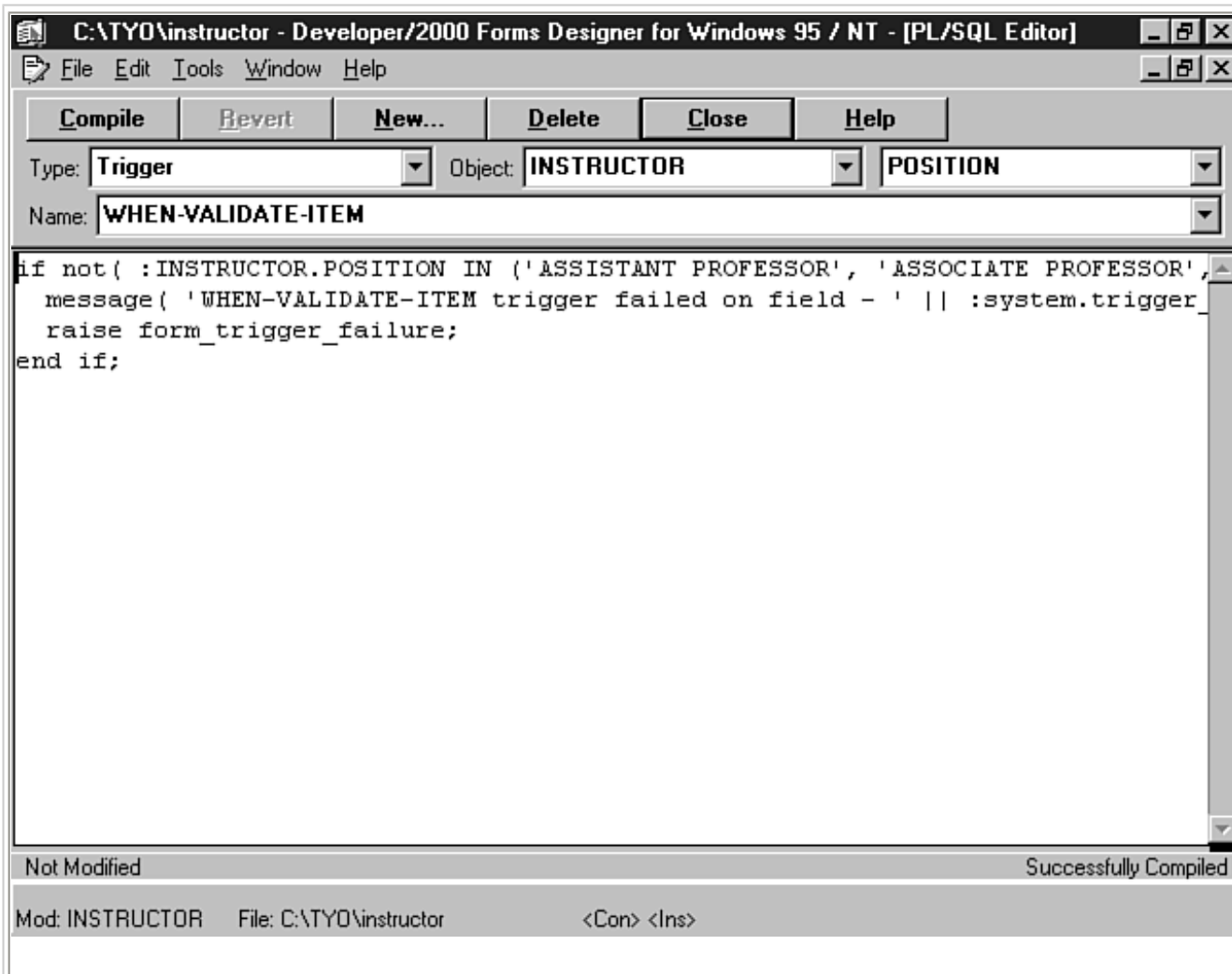


Abbildung 14.12:  
Im when-window-  
activated-Trigger  
vorhandener  
Code.

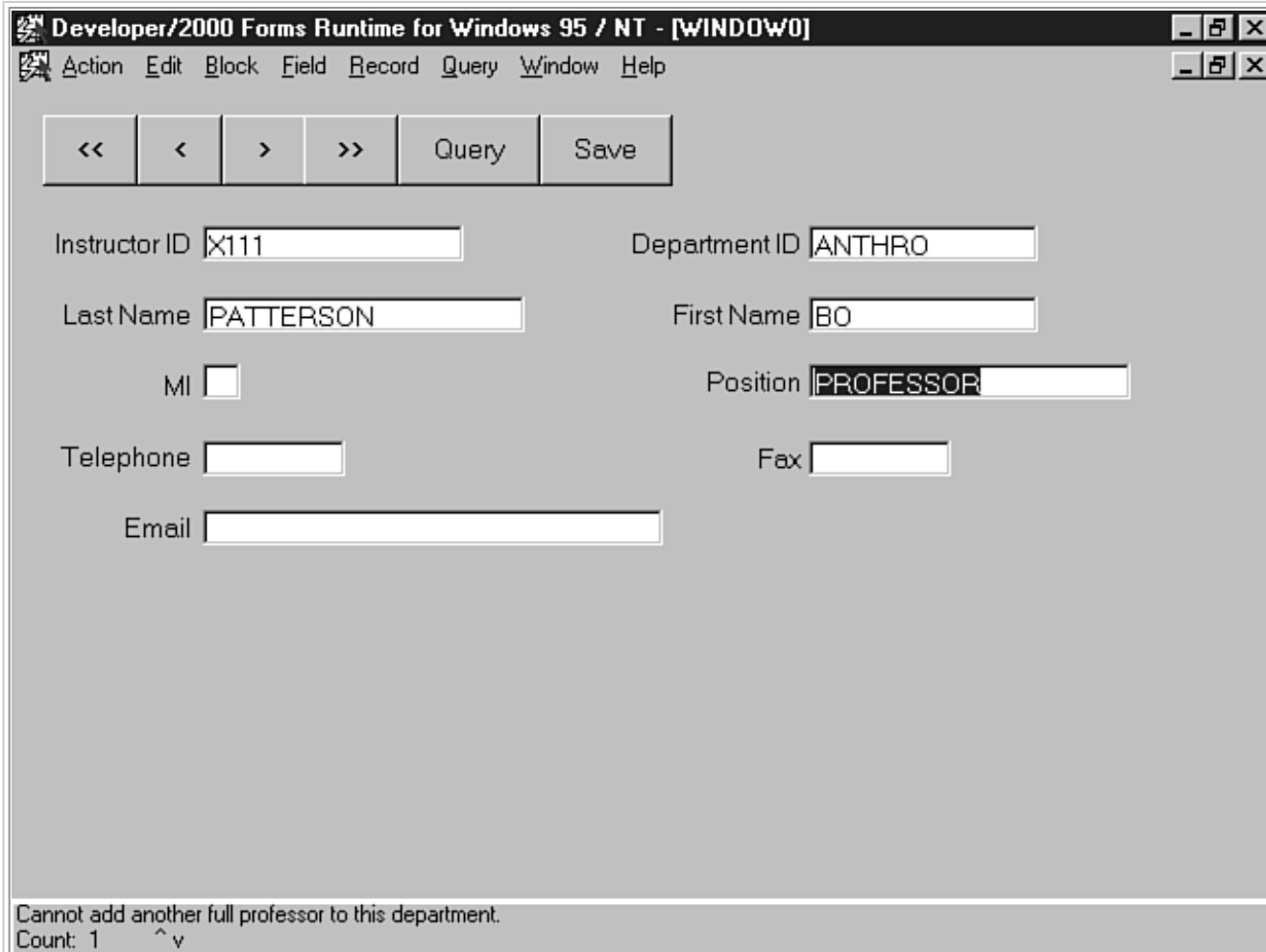


Abbildung 14.13:  
Testen des when-  
window-activated-  
Trigger.

## Elemente eines Menüs

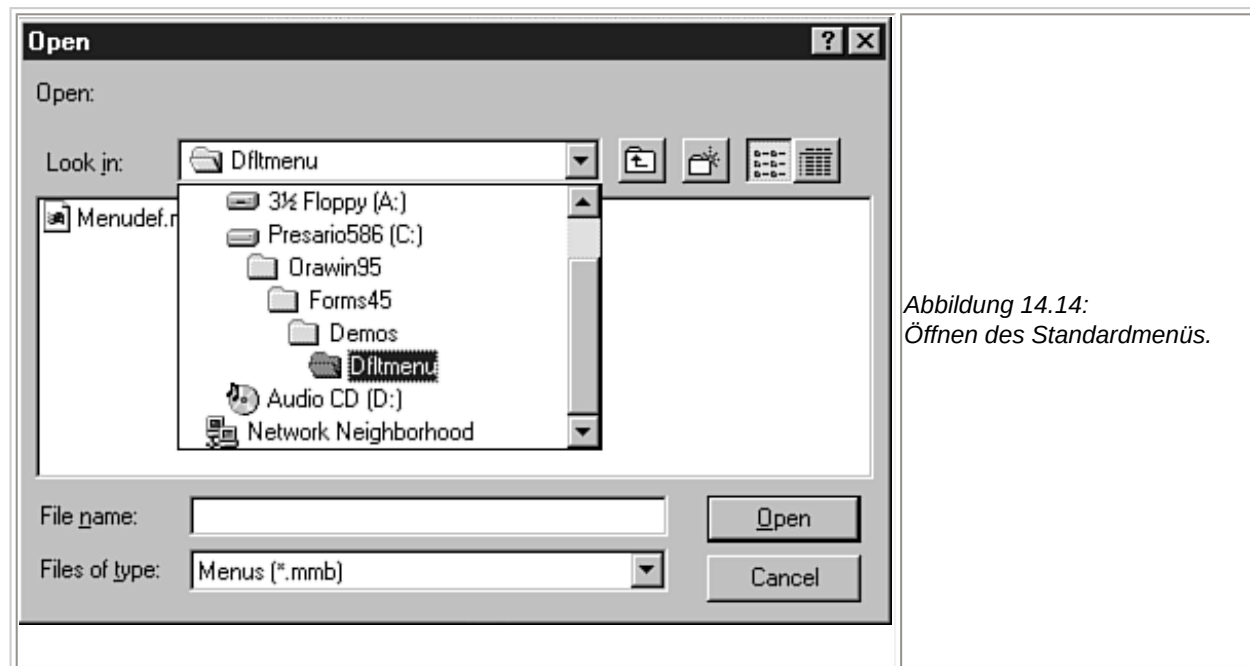
Ein Menü wird wie ein Formular durch eine Sammlung von Objekten definiert. Einige dieser Objekte sind erforderlich, andere optional. Jedes Objekt kann von einer der folgenden Art sein:

- Menü
- Objektgruppe
- Programmeinheit
- Parameter
- Eigenschaftsklasse
- Zugeordnete Bibliothek
- Optisches Attribut

Sie werden jetzt ein Menü für eine Multiform-Anwendung mit dem Namen MAIN\_MENU erstellen.

## Entwickeln eines Menüs

Beginnen Sie, indem Sie im Forms-Designer den Menüeintrag File | Open wählen (siehe Abbildung 14.4), um C:\Orawin95\Forms45\Demos\Dfltmnu\menudef öffnen. Dies ist der Standardpfad, wenn Sie Developer/2000 unter Windows 95 installiert haben. Im Object Navigator erweitern Sie die Anzeige des Knoten des Menüs-Objekts. Unter den Eigenschaften von MENUDEF finden Sie eine Eigenschaft namens Main Menu, die auf Main\_Menu gesetzt ist. Die Eigenschaft bezeichnet das übergeordnete Menü auf oberster Ebene.



Neben dem Menüs-Objektknoten finden Sie folgende Menüs:

- ACTION
- BLOCK
- MAIN\_MENU
- EDIT
- FIELD
- HELP
- QUERY
- RECORD

Diese Menüs stellen die Menüs auf oberster Ebene dar. Jedoch ist dies das Standardmenü, und Sie möchten drei Menüs auf dieser obersten Ebene hinzufügen:

- Ein Menü, das Menüpunkte enthält, über die der Benutzer Daten in unterschiedliche Tabellen eingeben kann.
- Ein Menü, das Menüpunkte enthält, über die der Benutzer Datensätze einer Datenbank durchsehen aber nicht verändern kann.
- Ein Menü zum Aufruf von Berichten und Diagrammen.

Um der obersten Ebene drei Menüs hinzuzufügen, wählen Sie mit der Maus oder den Pfeiltasten den Objektknoten Menüs aus, und klicken Sie auf das + in der vertikalen Werkzeugleiste. Es erscheint eine Menü mit dem Namen MENU0, das Sie anklicken, bis es in blauer Farbe angezeigt wird. Jetzt ändern Sie den Namen in DATA\_ENTRY. Zur Erstellung der Menüs BROWSE und REPORTS verfahren Sie genauso (siehe Abbildung 14.15). Ziehen Sie die drei Menüs unter RECORD.

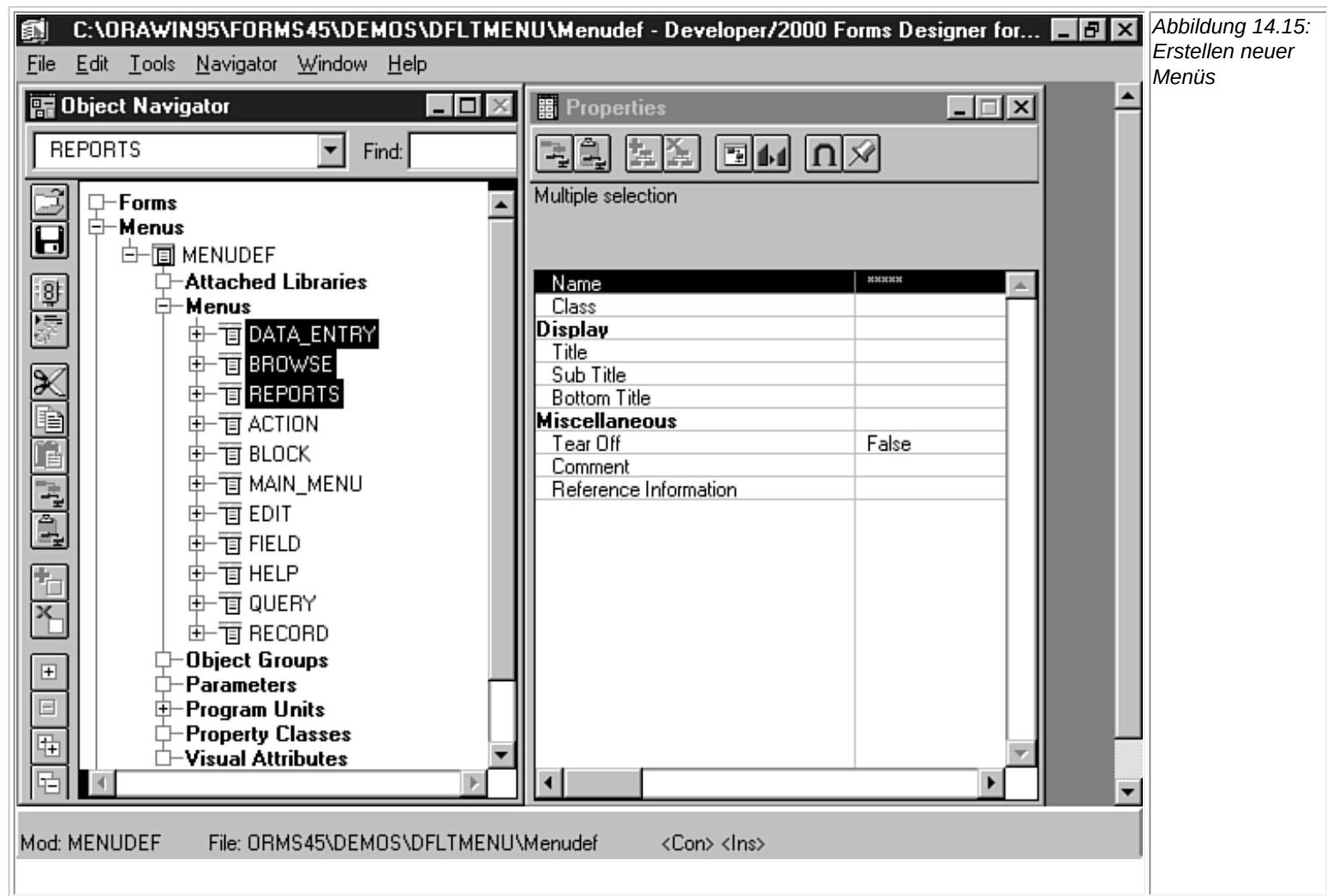


Abbildung 14.15:  
Erstellen neuer  
Menüs

## Integrieren neuer Menüs

Auch wenn es durch die Betrachtung der Menüs im Object Navigator schwer zu erkennen ist, ist MAIN\_MENU das Menü auf der obersten Ebene.

1. Erweitern Sie die Anzeige des Objektknotens Items unter MAIN\_MENU.
2. Um dem Hauptmenü das Menü Data Entry hinzuzufügen, wählen Sie den Objektknoten Items, und klicken Sie auf das + in der vertikalen Werkzeugleiste. Sie sehen jetzt ein neues Element namens ITEM, gefolgt von einer Zahl.
3. Klicken Sie das neue Element an, und benennen Sie es um in DATA\_ENTRY.
4. Wenn die Eigenschaften für DATA\_ENTRY nicht angezeigt werden, klicken Sie mit der rechten Maustaste auf Properties und blättern Sie dann nach unten bis zur Eigenschaft Command Text, und ändern Sie sie um in Menu (siehe Abbildung 14.16).

5. Blättern Sie nach unten bis zur Eigenschaft Command Text, und klicken Sie oben im Eigenschaftfenster auf die Schaltfläche More. Die Eigenschaft Command Text enthält den Befehl, der bei Auswahl des Menüpunkts ausgeführt wird. Es erscheint ein Fenster namens Command Text.

6. Geben Sie DATA\_ENTRY ein, und klicken Sie auf OK (siehe Abbildung 14.17).

7. Blättern Sie nach unten bis zur Eigenschaft Label, und geben Sie Data Entry ein. Dies wird der in der Menüleiste angezeigte Text sein.

8. Wenn Sie sich die zu MAIN\_MENU gehörenden Elemente anschauen, sehen Sie, daß DATA\_ENTRY das erste Element ist. Sie möchten aber nicht, daß es dort in der Menüleiste erscheint.

Um es zwischen EDIT und BLOCK zu stellen, wählen Sie DATA\_ENTRY mit der Maus an und ziehen es bei gedrückt gehaltener linker Maustaste, bis unter EDIT eine horizontale Linie erscheint, und lassen die linke Maustaste los.

9. Um die Menüs BROWSE und REPORTS in die oberste Ebene von MAIN\_MENU einzuordnen, führen Sie die Schritte 3 bis 8 aus. Doppelklicken Sie auf das Icon links neben MENUDEF, um das Aussehen des Menüs zu überprüfen. Abbildung 14.18 zeigt, wie es aussehen sollte.

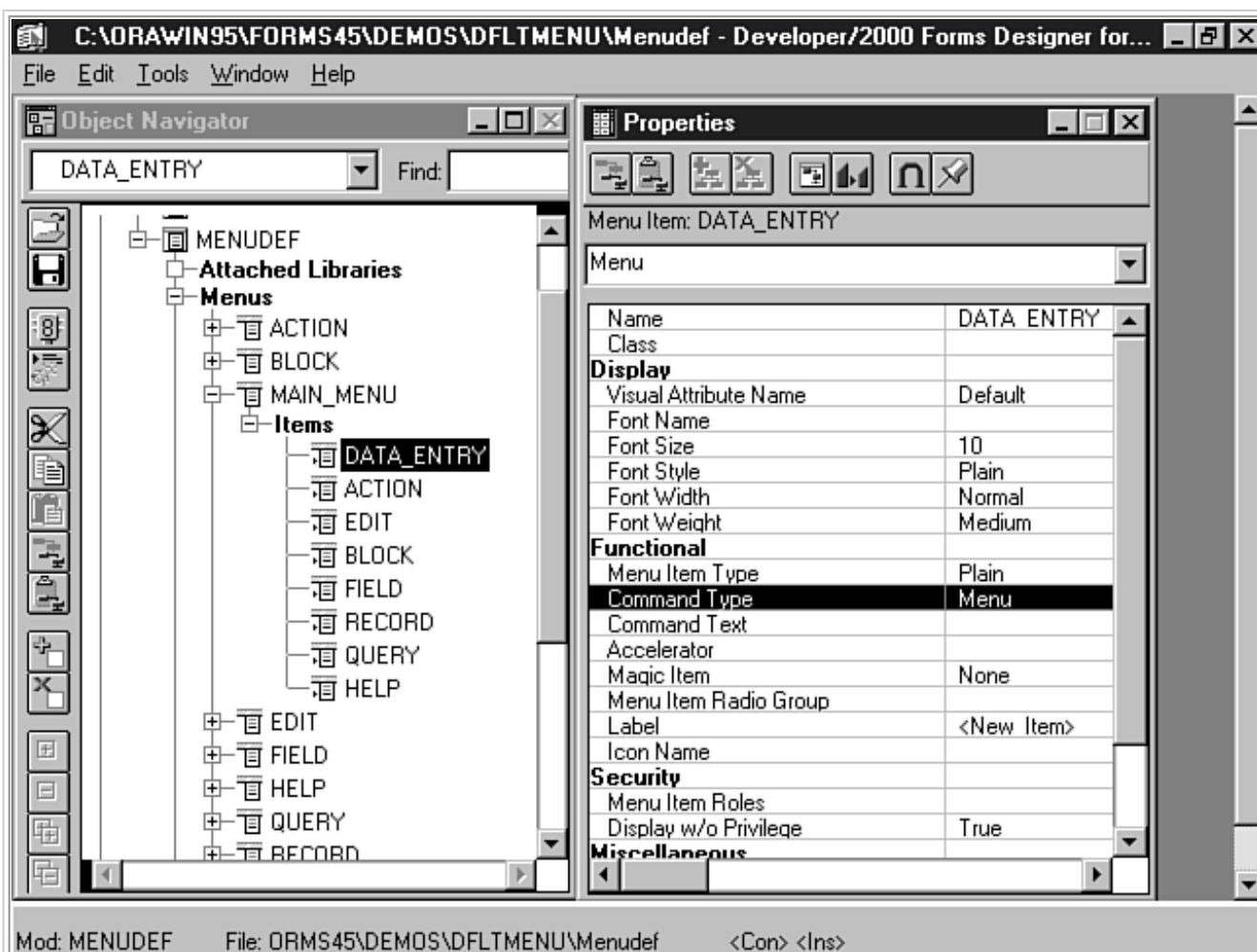


Abbildung 14.16: Setzen der Eigenschaft Command Type für das neue Menüelement.



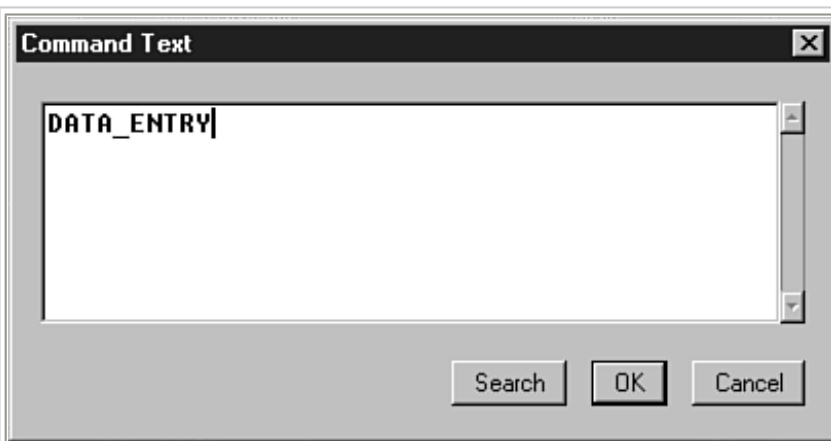


Abbildung 14.17:  
Setzen der Eigenschaft Command  
Text für das neue Menüelement.

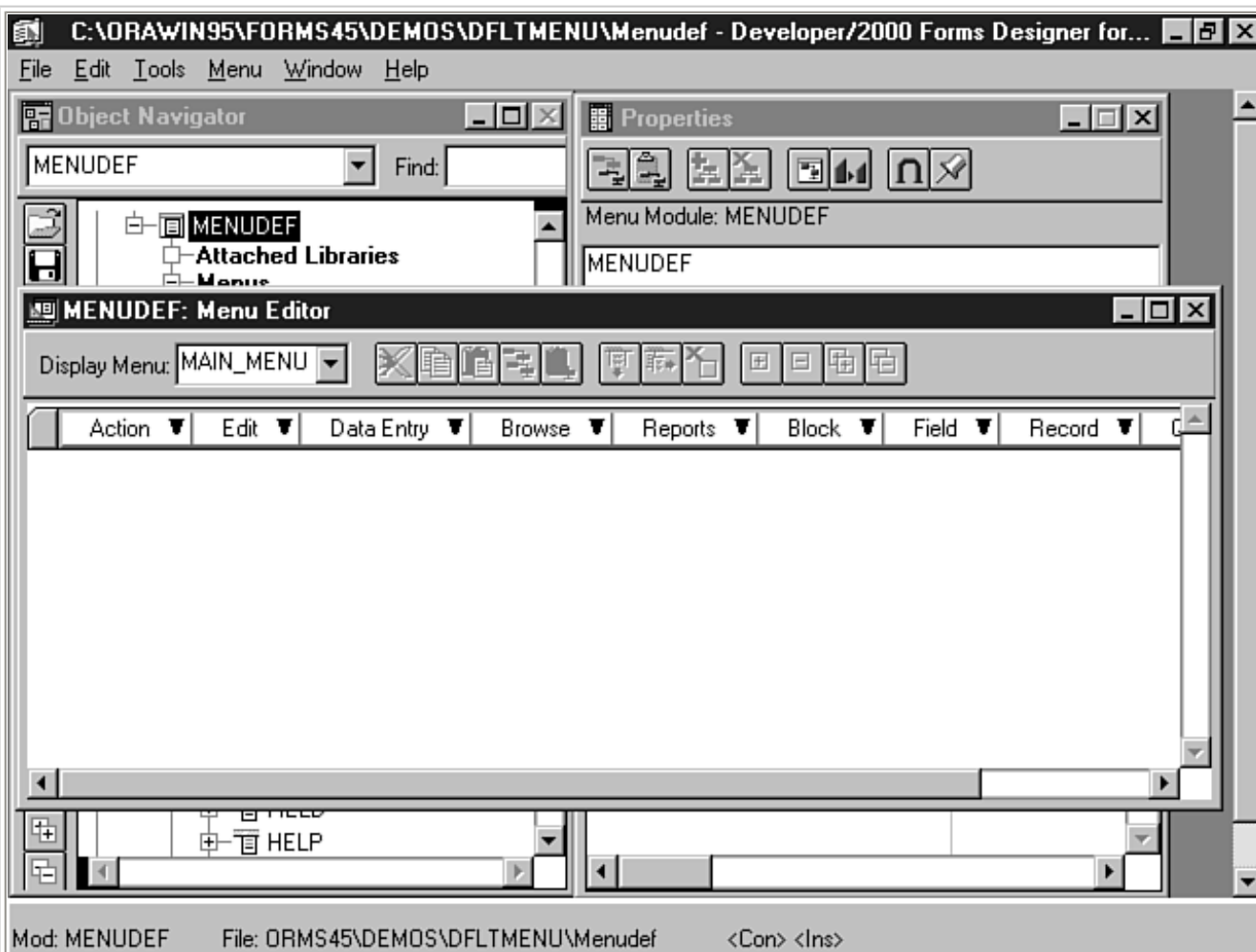


Abbildung 14.18:  
Testen des  
Erscheinungsbilds  
des Menüs.

## Elemente zu einem Menü hinzufügen

Sie haben zwar drei neue Menüs erstellt, aber keines enthält irgendwelche Menüpunkte. Wir wollen nun zwei Menüpunkte dem Menü Data Entry hinzufügen.

1. Verwenden Sie den Object Navigator, um die Anzeige des Menüs DATA\_ENTRY zu erweitern (nicht Element in MAIN\_MENU).
2. Wählen Sie den Objektknoten Items, und klicken Sie auf + in der vertikalen Werkzeugleiste.
3. Im Properties-Fenster ändern Sie den Namen des Elements zu INSTRUCTOR.
4. Blättern Sie nach unten bis zur Eigenschaft Command Text, der, wie Sie sehen können, auf PL/SQL gesetzt ist. Ändern Sie diese



Einstellung nicht, da es der gewünschte Wert ist.

5. Blättern Sie nach unten bis zur Eigenschaft Command Text, und klicken Sie oben im Fenster Properties auf die Schaltfläche More. Es erscheint der PL/SQL-Editor.

6. Geben Sie `call_form('Instructor',NO_HIDE,NO_REPLACE)` ein. Diese PL/SQL-Anweisung ruft die integrierte Forms-Prozedur, `call_form` auf, die das Formular Instructor anzeigt.

7. Klicken auf Compile and Close.

8. Blättern Sie im Properties-Fenster nach unten bis zur Eigenschaft Label, und geben Sie Instructor ein.

Außerdem müssen Sie ein Element für die Menüs Browse und Reports erstellen. Tun Sie dies nicht, erzeugt Forms Generator eine Fehlermeldung, wenn Sie versuchen, eine MMX-Datei für das Menü zu erzeugen.

1. Erstellen Sie für das Menü Browse ein Element namens INSTRUCTOR-\_BROWSE.

2. Als Command Text-Eigenschaft für dieses Element geben Sie ein: `call_form('Instructor_Browse',NO_HIDE,NO_REPLACE)`.

3. Erstellen Sie für das Menü Reports ein Element namens COURSE\_CATALOG.

4. Als Command-Text-Eigenschaft für dieses Element geben Sie Null ein; durch die Angabe NULL wird der Menüpunkt zwar im Menü erscheinen aber noch keine Aktion ausführen.

In der nächsten Lektion lernen Sie, wie von einem Menüpunkt ein Oracle-Bericht aufgerufen werden kann.

## Speichern des Menüs

Bevor Sie fortfahren, sollten Sie das Menü umbenennen.

1. Wählen Sie MENUDEF.

2. Wählen Sie den Menüeintrag File | Save as.

3. Achten Sie darauf, daß Sie als Verzeichnis, in dem gespeichert werden soll, das für Ihre Anwendung bestimmte Verzeichnis angeben.

4. Im Feld Save as wählen Sie Menus (\*.mmb).

5. Im Feld File name geben Sie Main\_menu ein.

6. Klicken Sie auf Save.

## Erzeugen des Menüs

Wird ein Menü im Dateisystem gespeichert, erhält die Datei die Endung .mmb. Forms Runtime kann jedoch nur eine MMX-Menüdatei ausführen, die aus einer MMB-Datei erzeugt wird. Um für Main\_menu eine MMX-Datei zu erzeugen, wählen Sie den Menüeintrag File | Administration | Generate, oder drücken Sie die Tasten (STRG) + (t).

## Entwickeln einer Multiform-Anwendung



*Bislang haben Sie nur einzelne Formulare innerhalb des Forms-Designers getestet. Jetzt werden Sie die Schritte durchführen, die zur Entwicklung einer Multiform-Anwendung erforderlich sind. Im ersten Schritt muß ein Formular erstellt werden, welches das Hauptfenster der Anwendung enthält; dieses Fenster soll immer das Menü und die Steuerung, die aus Meldung- und Statuszeile besteht, anzeigen.*

1. Dazu wählen Sie den Menüeintrag File | New | Form. Im Object Navigator erscheint ein neues Fenster.
2. Ändern Sie den Namen dieses Formulars in MDI\_FRAME.
3. Erweitern Sie die Anzeige des Window-Objektknotens; wählen Sie das einzige, automatisch erzeugte Fenster, klicken Sie mit der rechten Maustaste und dann auf Properties.
4. Ändern Sie den Namen des Fenster in ROOT\_WINDOW.
5. Damit die Fenstergröße bei einer Darstellung von 600 x 480 richtig angezeigt wird, stellen Sie Width auf 473 und Length auf 289. Wenn Sie die Verwendung einer anderen Bildschirmauflösung planen, müssen Sie Width und Length entsprechend anpassen.
6. Blättern Sie zur Eigenschaft Title X, und setzen Sie diese auf Flugle Information System. Abbildung 14.19 zeigt die für das Fenster gesetzten Eigenschaften.

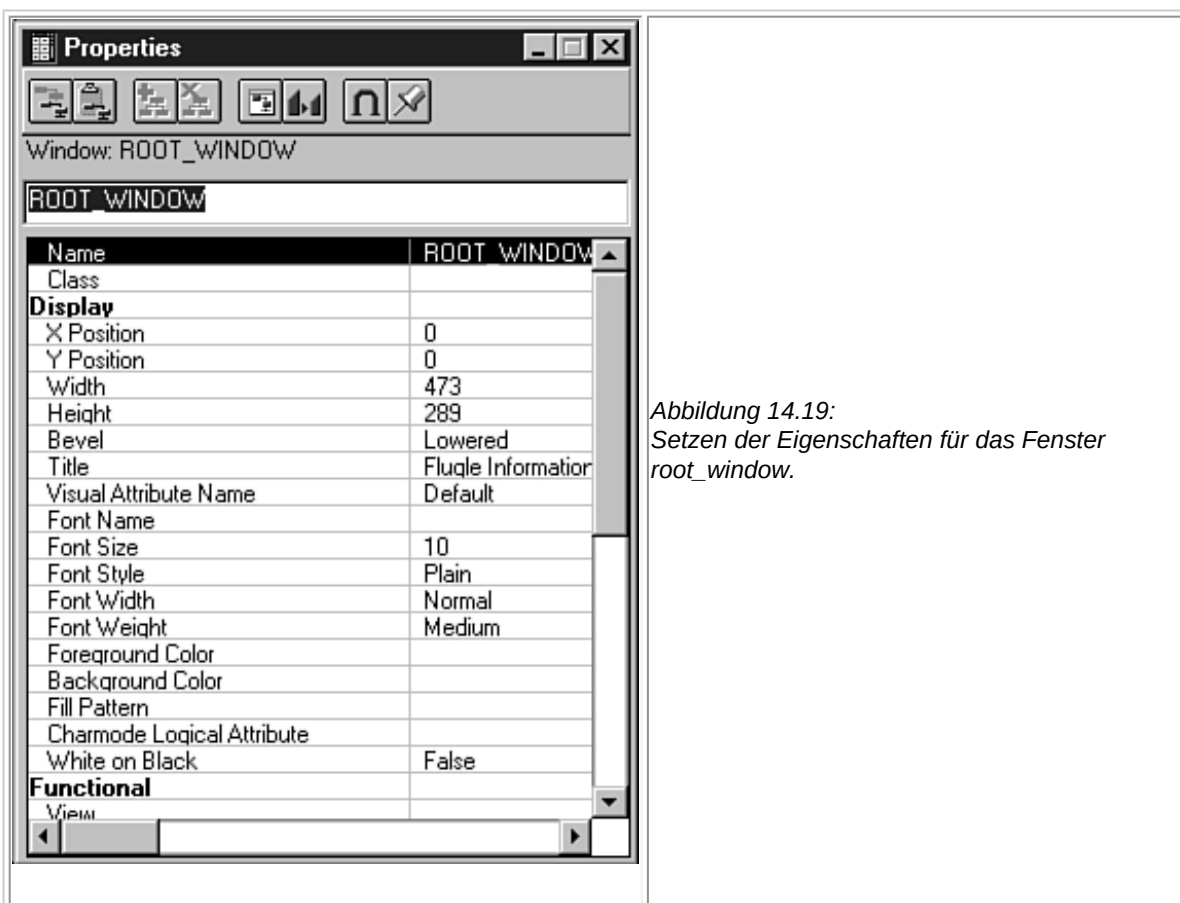


Abbildung 14.19:  
Setzen der Eigenschaften für das Fenster  
root\_window.

Jetzt müssen Sie die Canvas für das Formular erstellen.

1. Wählen Sie den Canvas-Ansicht-Objektknoten, und klicken Sie auf + in der vertikalen Werkzeugleiste.
2. Wählen Sie den erstellten Canvas aus, klicken Sie mit der rechten Maustaste und dann auf Properties.

3. Setzen Sie Width auf 473 und Length auf 289; die gleichen Werte, wie Sie sie für ROOT\_WINDOW festgelegt haben. Dies sind die einzigen für den Canvas benötigten Eigenschaften.
4. Wählen Sie jetzt das Formular MDI\_FRAME, klicken Sie mit der rechten Maustaste und dann auf Properties.
5. Setzen Sie die Eigenschaft Title auf Flugle Information System.
6. Blättern Sie nach unten zur Eigenschaft Menu Module, und setzen Sie sie auf MAIN\_MENU. Dies ist das soeben von Ihnen erstellte Menü.
7. Blättern Sie nach unten bis zur Eigenschaft Starting Menu, und setzen Sie sie auf MAIN\_MENU. Dies wird das zuerst verwendete Menü, wenn das Formular MDI\_FRAME aufgerufen wird. Abbildung 14.20 zeigt die Eigenschaftswerte des Formulars MDI\_FRAME.
8. Speichern Sie das Formular MDI\_FRAME als MDI\_FRAME in dem für die Anwendungskomponenten eingerichteten Verzeichnis.
9. Erzeugen Sie die .fmx-Datei für das Formular MDI\_FRAME, indem Sie die Tasten (STRG)+(T) drücken.

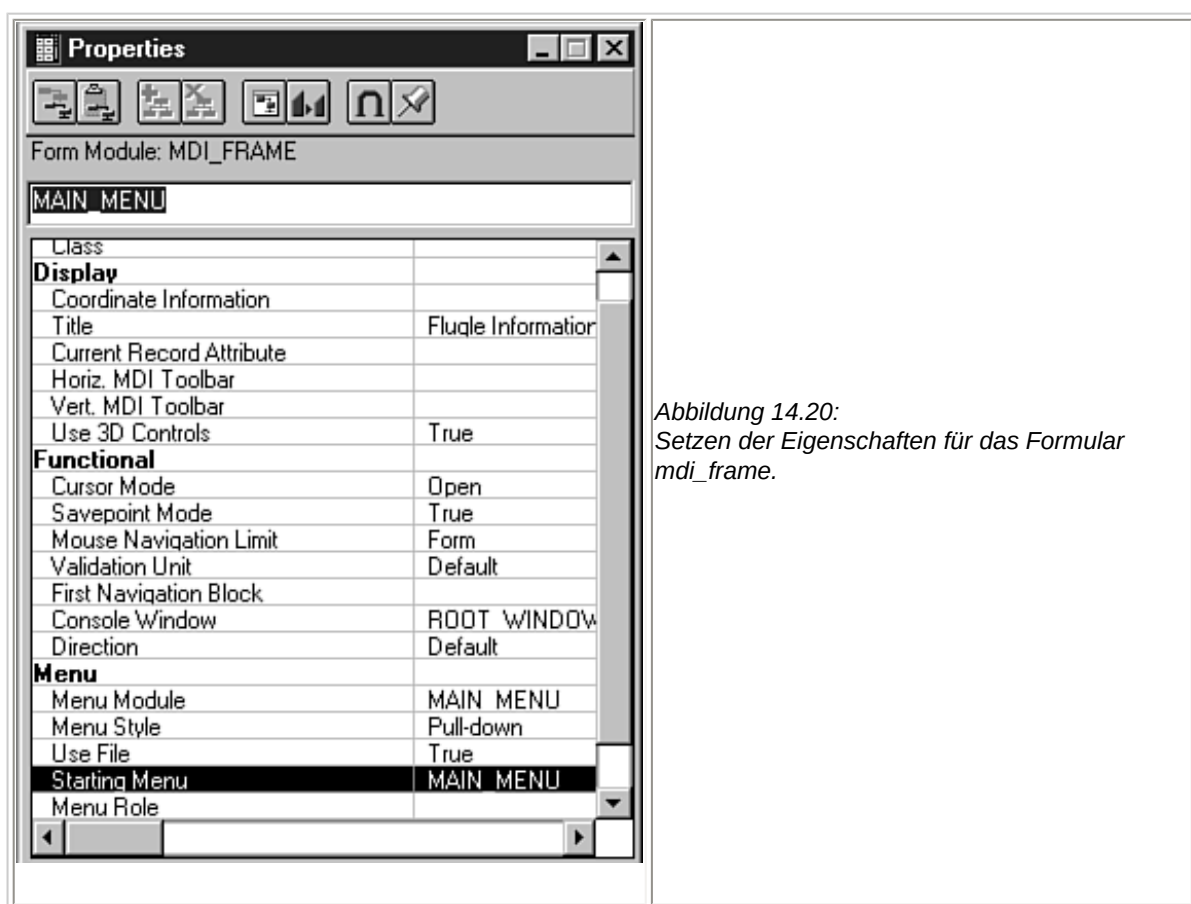


Abbildung 14.20:  
Setzen der Eigenschaften für das Formular  
mdi\_frame.

## Erzeugen eines Windows-Shortcuts zum Starten einer Anwendung

Ob Sie es glauben oder nicht, Sie haben die für eine Multiform-Anwendung benötigten Basiselemente erstellt. Um Ihre Anwendung zu testen, richten Sie einen Shortcut ein. Dazu klicken Sie auf dem Desktop von Windows 95 mit der rechten Maustaste und dann auf Neu | Verknüpfung. Im Fenster Verknüpfung erstellen müssen Sie die Befehlszeilenargumente eingeben. Ehe Sie diese für den Shortcut eingeben, besprechen wir einige der von Forms Runtime verwendeten Argumente:

- `userid` identifiziert den Benutzer, das Kennwort und, wenn vorhanden, den Datenbank-Alias zur Verbindung mit der Datenbank.
- `module` identifiziert das Formular, das Sie ausführen möchten (dies sollte das Formular sein, das das Hauptfenster MDI\_FRAME enthält).
- `window_state` ist ein optionaler Parameter, der die Darstellung des in `module` angegebenen Fensters bestimmt: Maximiert, Minimiert oder Wiederhergestellt.

Um Ihre Formular-Anwendung auszuführen, geben Sie im Feld Befehlszeile folgendes ein:

```
c:\0rawin95\bin\F4Srun32 module=c:\Tyo\MDI_FRAME  
userid=flugle/flugle@flugle window_state=maximize
```

Klicken Sie auf die Schaltfläche Weiter. Geben Sie dem Shortcut einen Namen, wie z.B. Flugle Information System, und klicken Sie auf die Schaltfläche Beenden. Der neue Shortcut sollte auf Ihrem Desktop erscheinen. Versuchen Sie, den Shortcut auszuführen. Rufen Sie den Menüeintrag Data Entry | Instructor auf; das Formular Instructor sollte Ihnen angezeigt werden.



*Ein Nachteil des Shortcut ist es, daß das Kennwort im Shortcut enthalten ist und von anderen eingesehen werden kann. Wenn Sie für userid keinen Wert angeben, fordert Forms Runtime bei Aktivierung des Shortcuts den Benutzer zur Eingabe des Benutzernamens, Kennworts und der Datenbank auf.*

---

## Schließen eines Formulars

Wenn Sie den Formular-Instructor aus dem Menü Data Entry aufrufen, entdecken Sie vielleicht, daß ein Weg, das Formular zu schließen, die Auswahl des Menüeintrags Action | Exit ist. Klicken Sie jedoch auf das Schließen-Symbol in der rechten oberen Ecke des Formulars Instructor (nicht im Formular MDI\_FRAME), schließt sich das Formular Instructor nicht. Idealerweise möchten Sie, daß sich das Formular schließt, wenn der Benutzer dieses Symbol anklickt. Dazu müssen Sie im Formular-Instructor einen weiteren Trigger erstellen.

1. Öffnen Sie mit dem Object Navigator das Formular Instructor.
2. Erweitern Sie das Formular, und wählen Sie den Objektknoten-Trigger auf Formularebene.
3. Klicken Sie auf + in der vertikalen Werkzeugleiste, um einen neuen Trigger zu erstellen.
4. Im darauf erscheinenden Fenster wählen Sie das Ereignis WHEN-WINDOW-CLOSED, und klicken Sie auf OK.
5. Doppelklicken Sie auf das im Object Navigator links neben dem Trigger WHEN-WINDOW-CLOSED angezeigte Symbol.
6. Im PL/SQL-Editor geben Sie eine einzige Zeile ein: exit\_form. Diese integrierte Prozedur wird immer dann aufgerufen, wenn das Fenster geschlossen wird.
7. Klicken Sie auf die Schaltfläche Close.
8. Speichern Sie das Formular Instructor, und lassen Sie es neu aufbauen.
9. Benutzen Sie den Shortcut, um das Formular zu testen, und überprüfen Sie, ob das Formular bei einem Klick auf das Schließen-Symbol geschlossen wird.

## Zusammenfassung

Dies waren die wesentlichen Punkte dieser Lektion:

- Ein Trigger ist ein Gruppe von PL/SQL-Anweisungen, die beim Eintreten bestimmter Ereignisse ausgeführt werden.
- In einem Formular können Trigger auf unterschiedlichen Ebenen definiert werden: auf Formular-, Block- und Elementebene.
- Die Elemente, die einen Trigger auslösen, unterscheiden sich durch die Ebene des Trigger. So lösen z.B. bestimmte

Fensterereignisse nur Trigger auf Formularebene aus.

- Einige Trigger werden von Forms-Designer automatisch erzeugt, wenn Sie für einen neuen Block die Einhaltung von Integritätsregeln erzwingen wollen.
- Zur Festlegung der PL/SQL-Anweisungen für einen Trigger verwenden Sie den PL/SQL-Editor.
- Ein Trigger kann zum Abrufen von Werten aus anderen Tabellen eingesetzt werden.
- Ein Trigger kann zur Gültigkeitsüberprüfung von Benutzereingaben eingesetzt werden. Ein solcher Trigger kann die integrierte Exception FORM\_TRIG-GER\_FAILURE aufrufen, wenn die Benutzereingabe die Gültigkeitsprüfung nicht besteht.
- Ein Trigger kann zur automatischen Ausführung einer Abfrage eingesetzt werden, wenn ein Formular aufgerufen wird.
- Zur Erstellung und Anpassung von Menüs verwenden Sie Forms-Designer.
- Ein Menü besteht aus mehreren Objekttypen, einschließlich einem Hauptmenü und anderen Menüs, einer Objektgruppe, einer Programmeinheit, Parametern, Eigenschaftenklassen, verbundenen Bibliotheken und optischen Attributen.

## Wie geht es weiter?

An Tag 15, »Developer/2000: Entwicklung von Berichten mit Hilfe von Oracle Reports«, lernen Sie Oracle Reports kennen. Sie werden die Schritte zur Erstellung von zwei Berichten durchgehen: einen einfachen, auf einer Tabelle basierenden Bericht und einen Master-/Detail-Bericht. Außerdem sehen Sie, wie ein Format-Trigger angegeben wird, der PL/SQL-Anweisungen verwendet, um das Aussehen von Berichtselementen von ihren Werten abhängig zu steuern.

## Fragen und Antworten

Frage:

Wie verarbeitet Oracle Forms auf unterschiedlichen Ebenen definierte Trigger, die auf dasselbe Ereignis reagieren?

Antwort:

*Wenn für ein Ereignis zwei oder mehr Trigger definiert wurden, erhält der Trigger auf der untersten Ebene den Vorzug. Nehmen Sie z.B. an, ein Formular hätte zwei Trigger, einen auf Block- und einen auf Elementebene. Wenn beide Trigger für das Ereignis **WHEN-NEW-ITEM-INSTANCE** definiert wurden, wird der Trigger auf Elementebene beim Eintreten des Ereignisses ausgelöst.*

Frage:

Wie heißen einige der integrierten Prozeduren, die in Oracle Forms zur Verfügung stehen?

Antwort:

*Es gibt eine Reihe von integrierten Prozeduren, die aus PL/SQL-Unterprogrammen in Triggern oder Bibliotheksmodulen aufgerufen werden können. Diese Prozeduren schließen ein:*

- Prozeduren zum Navigieren, wie GO\_BLOCK, GO\_ITEM und NEXT\_ITEM
- Prozeduren, die den Inhalt eines Blocks ändern, wie CLEAR\_RECORD, DELETE\_RECORD und INSERT\_RECORD
- Prozeduren, die Abfrageprozesse steuern: ENTER\_QUERY und EXECUTE\_QUERY

Frage:

Wie heißen einige der in Oracle Forms verfügbaren Systemvariablen?

Antwort:

*Es gibt einige Systemvariablen, die in einem Trigger nützlich sein können:*

- \$\$DATE\$\$ und \$\$TIME\$\$ enthalten aktuelles Datum und Zeit:

- Variablen, die die aktuelle Auswahl in einem Formular beschreiben:  
SYSTEM.CURRENT\_FORM, SYSTEM.CURRENT\_BLOCK, SYSTEM.CURRENT\_ITEM

Frage:

Unterstützt Oracle Forms die Verwendung von globalen Variablen?

Antwort:

*Ja. Sie können zur Speicherung von Zeichenfolgen mit bis zu 255 Zeichen globale Variablen verwenden. Eine globale Variable wird nicht deklariert; sie wird implizit deklariert, wenn Sie ihr einen Anfangswert zuweisen:*

**:GLOBAL.Instructor\_ID \_: #a1234';**

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Richtig oder falsch? Ein Trigger auf Elementebene eines Blocks kann nicht auf Elemente eines anderen Blocks verweisen.
2. Wie wird in einem Trigger auf Blockelemente Bezug genommen?
3. Welche Exception sollte bei einer ungültigen Benutzereingabe hervorgerufen werden, wenn Sie einen Trigger auf Elementebene zur Gültigkeitsüberprüfung von Benutzereingaben erstellen?

## Übungen

1. Ändern Sie das Formular Instructor dahingehend ab, daß es die gleiche Breite und Höhe wie das MDI\_Frame-Formular besitzt.
2. Erstellen Sie ein Element für das Browse-Menü, das ein Formular aufruft, in dem das Formular Instructor sortiert nach dem Nachnamen, Vornamen und Mittelinitialen in einer Tabelle angezeigt wird.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖ Kapitel ❖ Top Inhalt Index **SAMS** Kapitel ❖

# 15

# Developer/2000: Entwicklung von Berichten mit Hilfe von Oracle Reports

[Elemente eines Oracle-Berichts](#)

[Die Programme von Oracle Reports](#)

[Erstellen eines tabellarischen Berichts auf Basis einer einzelnen Tabelle](#)

[Erstellen eines Master-/Detail-Berichts](#)

[Verwenden eines Format-Trigger](#)

[Erzeugen einer Verknüpfung zum Starten eines Berichts](#)

[Aufrufen eines Berichts aus einem Oracle-Formular](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

In dieser Lektion lernen Sie die grundlegenden Funktionen von Oracle Reports kennen. Sie lesen über die beiden grundlegenden Berichtstypen - einen auf einer einzigen Tabelle basierenden tabellarischen Bericht und einen Master-/Detail-Bericht. Oracle Reports steckt voller Leistungsmerkmale, und diese Lektion soll eine Einführung in die wichtigsten von ihnen leisten.

## Elemente eines Oracle-Berichts

Wie auch Oracle Forms besitzt Oracle Reports einen Object Navigator, den Sie zur Erstellung und Anpassung von Berichtselementen verwenden. Der Object Navigator stellt Objekte in hierarchischer Anordnung dar. Für jeden Bericht zeigt Object Navigator einen Satz von Knoten an:

- Datenmodell

- Layout
- Parameterformular
- Berichts-Trigger
- Programmeinheiten
- Verbundene Bibliotheken

Angaben zur Verwendung des Object Navigator finden Sie an Tag 12, »Developer/2000: Einführung in Oracle Forms«. Die folgende Lektion befaßt sich näher mit den Objekten eines typischen Berichts.

## Datenmodell

Jeder Bericht benötigt ein Datenmodell. Das Datenmodell beschreibt die Abfragen, die Datensätze aus der Datenbank abrufen. Ein Datenmodell kann sich aus einer oder mehreren Abfragen zusammensetzen. Eine Abfrage kann aus jeder gültigen SELECT-Anweisung bestehen; mit anderen Worten, Sie können aus verschiedenen Tabellen und sogar aus einer Datensicht auswählen. Außerdem können Sie eine ORDER BY-Klausel verwenden, um die Reihenfolge anzugeben, in der die Zeilen abgerufen werden sollen.

## Layout

Jeder Bericht muß auch ein Layout aufweisen. Sie können sogar bis zu vier Layoutkomponenten beschreiben - Header, Body, Footer und Margin. Die Beispiele dieses Kapitels umfassen nur das Body-Layout. Sie können aber für den Header eines Berichts, der nur einmal ganz zu Anfang eines Berichts gedruckt wird, ein Layout definieren. Ein Beispiel für den Header eines Berichts könnte die Beschreibung des Inhalts eines Berichts und die Warnung über die Gewähr für die Richtigkeit der im Bericht angezeigten Informationen sein.

Das Layout eines Berichts kann aus vielen verschiedenen Elementen bestehen, einschließlich Rahmen, Feldern, Textelementen und grafischen Elementen, wie z.B. Linien und Rechtecken. Ein Rahmen wird zur Beschreibung der Merkmale der in ihm enthaltenen Elemente verwendet - z.B. wann und in welcher Richtung sie gedruckt werden sollen. Ein Rahmen kann mehrere Felder enthalten. Jedes Feld kann der Spalte einer Abfrage zugeordnet sein - was als die *Quelle* des Feldes bezeichnet wird. Die Quelle eines Feldes kann auch das aktuelle Datum oder eine physikalische Seitennummer sein.

## Parameterformular

Oracle Reports ermöglicht Ihnen, ein angepaßtes Runtime-Parameter-Formular zu entwerfen, das Parameter mit vom Benutzer zur Laufzeit angegebenen Werten enthalten kann. So können Sie z.B. ein Parameter-Formular entwerfen, das es dem Benutzer ermöglicht, eine Department ID anzugeben, die die Berichtsausgabe auf diese Abteilung beschränkt. Geben Sie kein Parameter-Formular an, zeigt Oracle Reports das normale Runtime-Parameter-Formular an.

## Die Programme von Oracle Reports

Oracle Reports beinhaltet zwei für diese Lektion relevante Werkzeuge - Reports Designer und Reports Runtime. Reports Designer läßt Sie drei Modultypen erstellen und anpassen: Berichte, externe Abfragen und Bibliotheken.

Standardmäßig speichert Reports Designer die von Ihnen entworfenen Module in eine Datei oder eine Oracle-Datenbank. Um ein Modul in eine Datenbank zu speichern, müssen Sie oder Ihr DBA das von Developer/2000 benötigte Datenbankschema installieren. Mehr darüber können Sie in den *Administrative Notes in Developer/2000 Admin for Win95* erfahren.

Reports Designer speichert einen Bericht als eine RDF-Datei: Reports Definition File. Dieses binäre Dateiformat ist portierbar, was bedeutet, daß es vom Reports Designer unter Windows 95, Mac OS oder Unix gelesen werden kann. Der Reports Designer generiert eine binäre REP-Datei, die zur tatsächlichen Ausführung eines Berichts von Reports Runtime verwendet wird. Eine RDF-Datei kann auch in eine REXX-Datei konvertiert werden, die eine vollständige Textbeschreibung des Berichts darstellt, die aus Code heraus gesteuert werden kann.

## Erstellen eines tabellarischen Berichts auf Basis einer einzelnen Tabelle



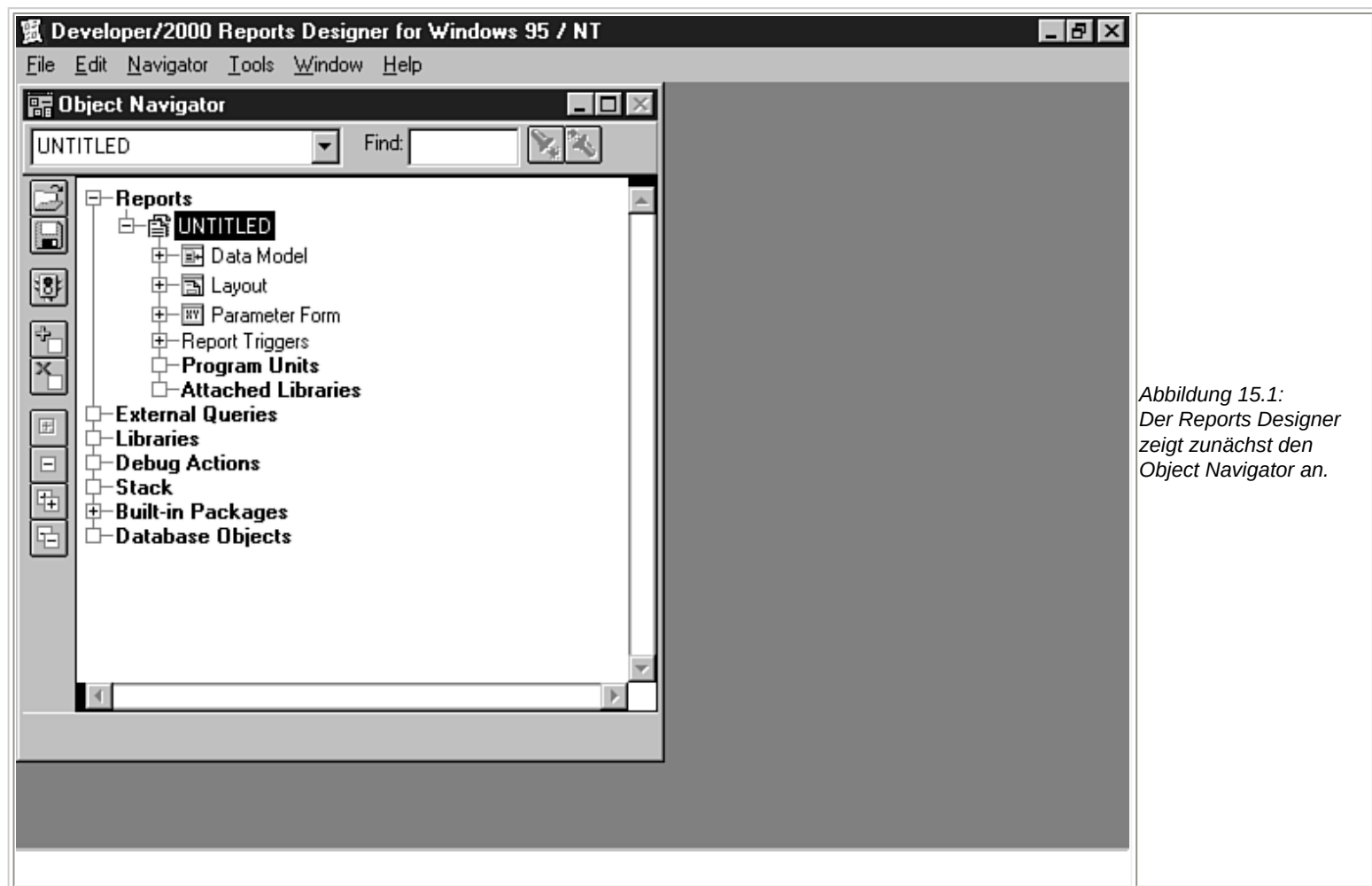
Beginnen wir, Oracle Reports einzusetzen, um einen tabellarischen Bericht auf der Basis einer einzigen Tabelle zu entwerfen. Sie werden einen Bericht erstellen, der Kursinformationen darstellt. Zum Aufruf von Reports Designer über die Start-Schaltfläche klicken Sie auf Oracle Designer for Win95 | Reports Designer. Reports Designer zeigt als erstes den Object Navigator an (siehe Abbildung 15.1).

Beachten Sie, daß im Object Navigator ein Bericht namens UNTITLED angezeigt wird. Ehe Sie fortfahren, stellen Sie eine Verbindung zu einer Oracle Datenbank her.

### Aufbau einer Verbindung zu einer Oracle-Datenbank

Um Reports Designer mit einer Oracle-Datenbank zu verbinden, führen Sie folgende Schritte durch:

1. Wählen Sie den Menüeintrag File | Connect. Im dann erscheinenden Dialogfenster werden Sie zur Eingabe des Benutzernamens, Kennwortes und der Datenbank aufgefordert, um eine Verbindung zu einer Oracle-Datenbank aufzubauen.
2. Verwenden Sie für die Entwicklung Personal Oracle, benutzen Sie flugle als Benutzernamen und Kennwort. Verwenden Sie für die Entwicklung einen Oracle Server, benutzen Sie flugle als Benutzernamen, Kennwort und Datenbank.



3. In beiden Fällen klicken Sie auf die Schaltfläche Connect. Erscheinen keine Fehlermeldungen, können Sie davon ausgehen, erfolgreich eine Verbindung mit der Datenbank aufgebaut zu haben.

### Erstellen eines Datenmodells

Der erste Schritt zur Berichtserstellung ist die Spezifizierung des Datenmodells für den Bericht. Zu diesem Zweck wird der Data Model Editor verwendet. Sie können den Data Model Editor aufrufen, indem Sie entweder auf das Symbol links des Data-Model-Knotens im Object Navigator klicken oder den Menüeintrag Tools | Data Model Editor wählen. Maximieren Sie das Fenster.

Um eine Abfrage zu erstellen, klicken Sie in der Werkzeugleiste am linken Rand des Fensters auf die mit SQL beschriftete Schaltfläche. Bewegen Sie die Maus in den Hauptteil des Fensters, und halten Sie die Maustaste gedrückt. Sie sehen die Umrisse eines Rechtecks. Wenn das Rechteck eine vernünftige Größe erreicht hat, lassen Sie die Maustaste los. Jetzt sollten Sie ein abgerundetes schwarzes Rechteck mit der Beschriftung Q\_1 sehen (siehe Abbildung 15.2).

Klicken Sie doppelt auf das die Abfrage Q\_1 darstellende Rechteck. Es wird ein Fenster angezeigt, in dem die SELECT-Anweisung eingegeben werden kann. Ändern Sie den Namen der Abfrage in Q\_COURSE. Klicken Sie die Schaltfläche mit der Aufschrift Tables/Columns an. Ein weiteres Fenster mit der Überschrift Table and Column Names wird angezeigt. In diesem Fenster können Sie die für die Abfrage benutzte Tabelle und die Spalten auswählen. Im Listenfeld namens Database Objects blättern Sie nach unten bis zur COURSE-Tabelle und wählen diese aus. Wählen Sie im Listenfeld Columns alle Spalten aus (siehe Abbildung 15.3).

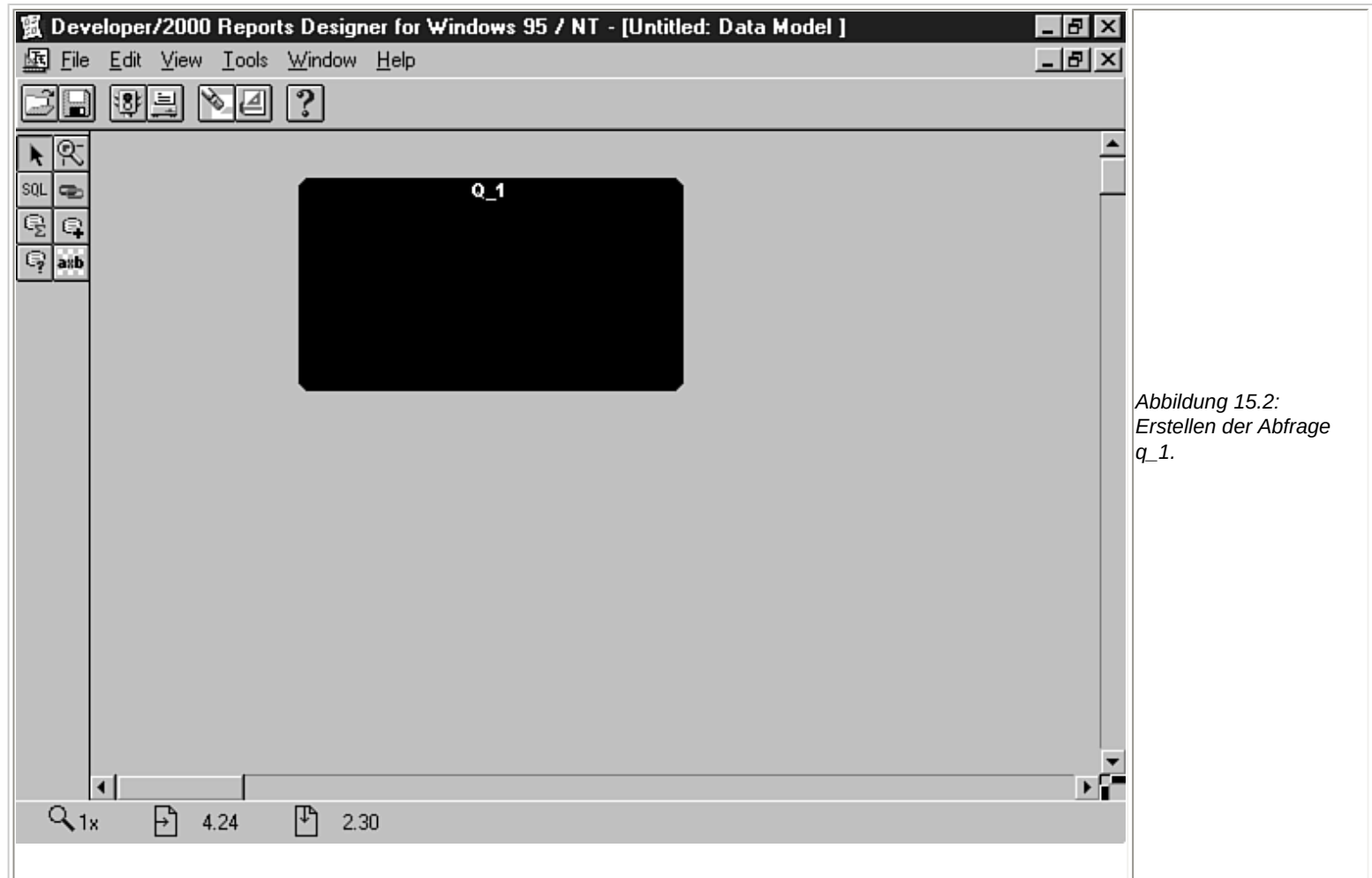
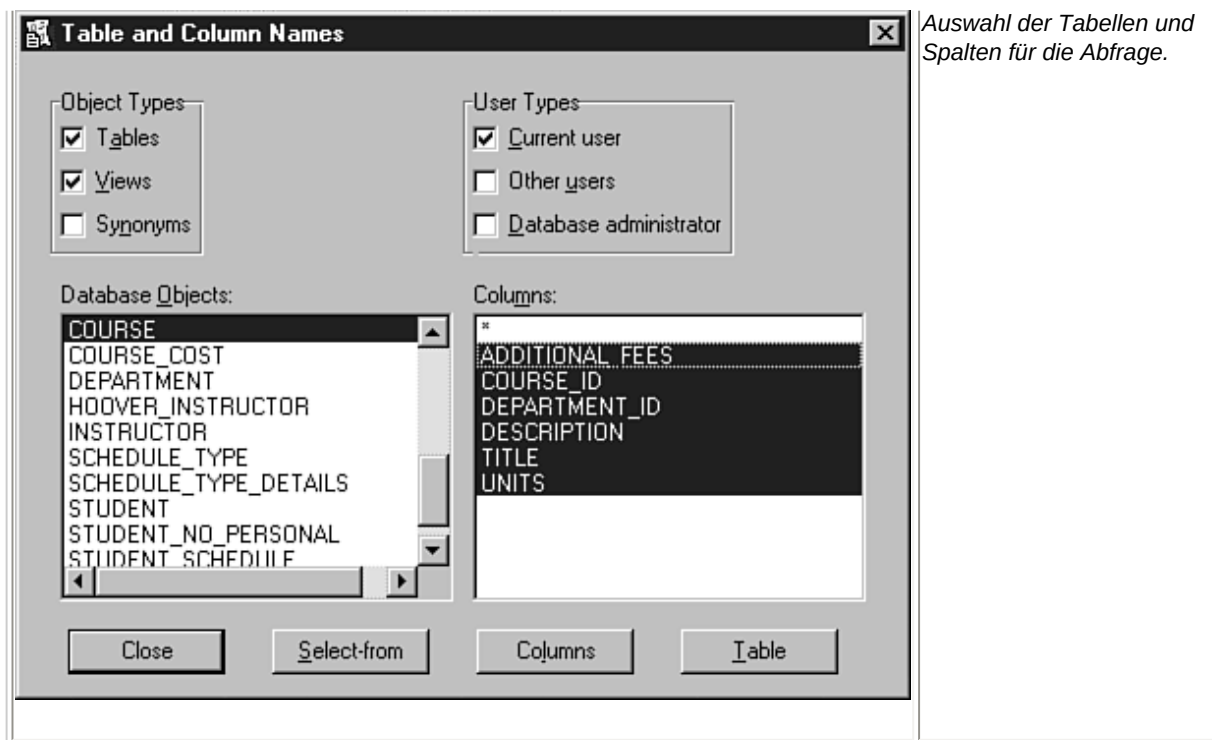
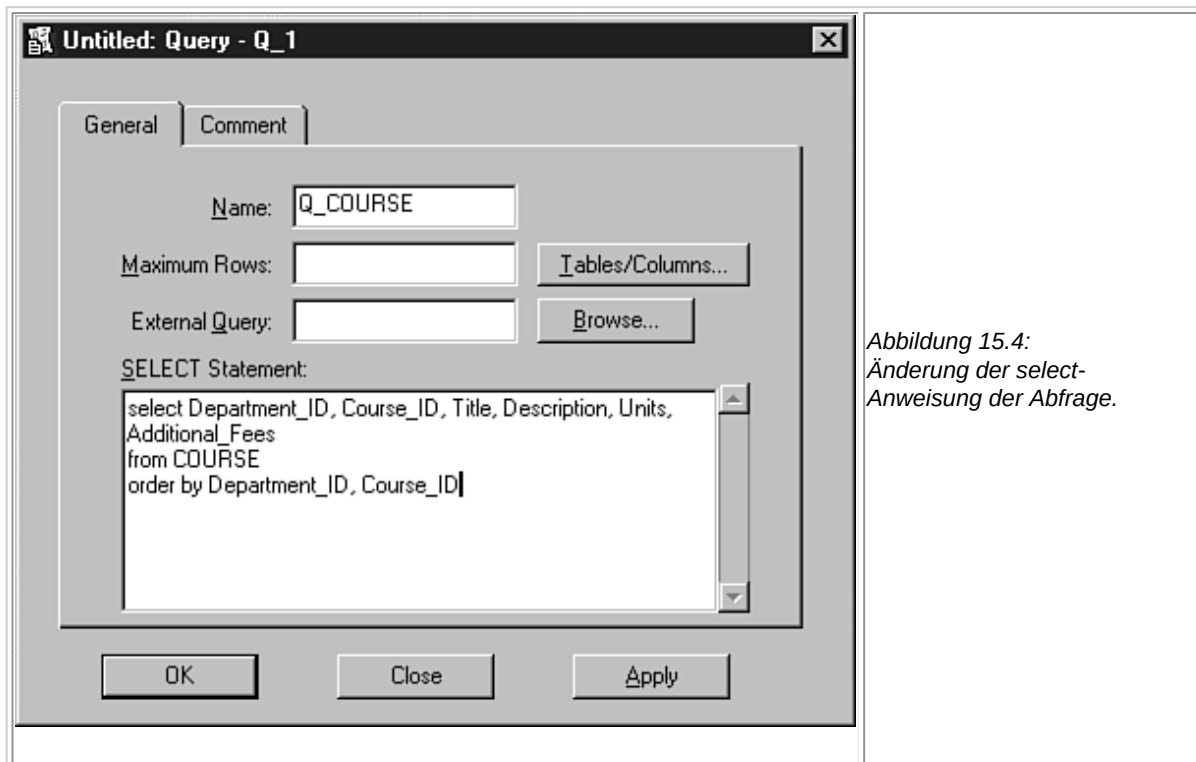


Abbildung 15.3:



Um die ausgewählte Tabelle und die Spalten in der SELECT-Anweisung der Abfrage zu verwenden, befolgen Sie diese Schritte:

1. Klicken Sie auf SELECT-FROM und CLOSE. Sie sehen die aus der gewählten Tabelle und den Spalten erstellte SELECT-Anweisung.
2. Stellen Sie die Spaltennamen entsprechend um, damit sie die von Ihnen im Bericht gewünschte Reihenfolge haben: Department\_ID, Course\_ID, Title, Description, Units und Additional Fees.
3. Fügen Sie der SELECT-Anweisung eine ORDER BY-Klausel hinzu, damit die Zeilen, wie in Abbildung 15.4 gezeigt, nach Department\_ID und Course\_ID sortiert werden.
4. Klicken Sie auf OK, um zum Data Model Editor zurückzukehren.



Als Sie die SELECT-Anweisung für die Abfrage definiert haben, wurde vom Reports Designer eine Gruppe für die Abfrage erstellt. Wie aus Abbildung 15.5 zu ersehen ist, wurde der Name der Gruppe aus dem Namen der Abfrage konstruiert: Q\_COURSE.

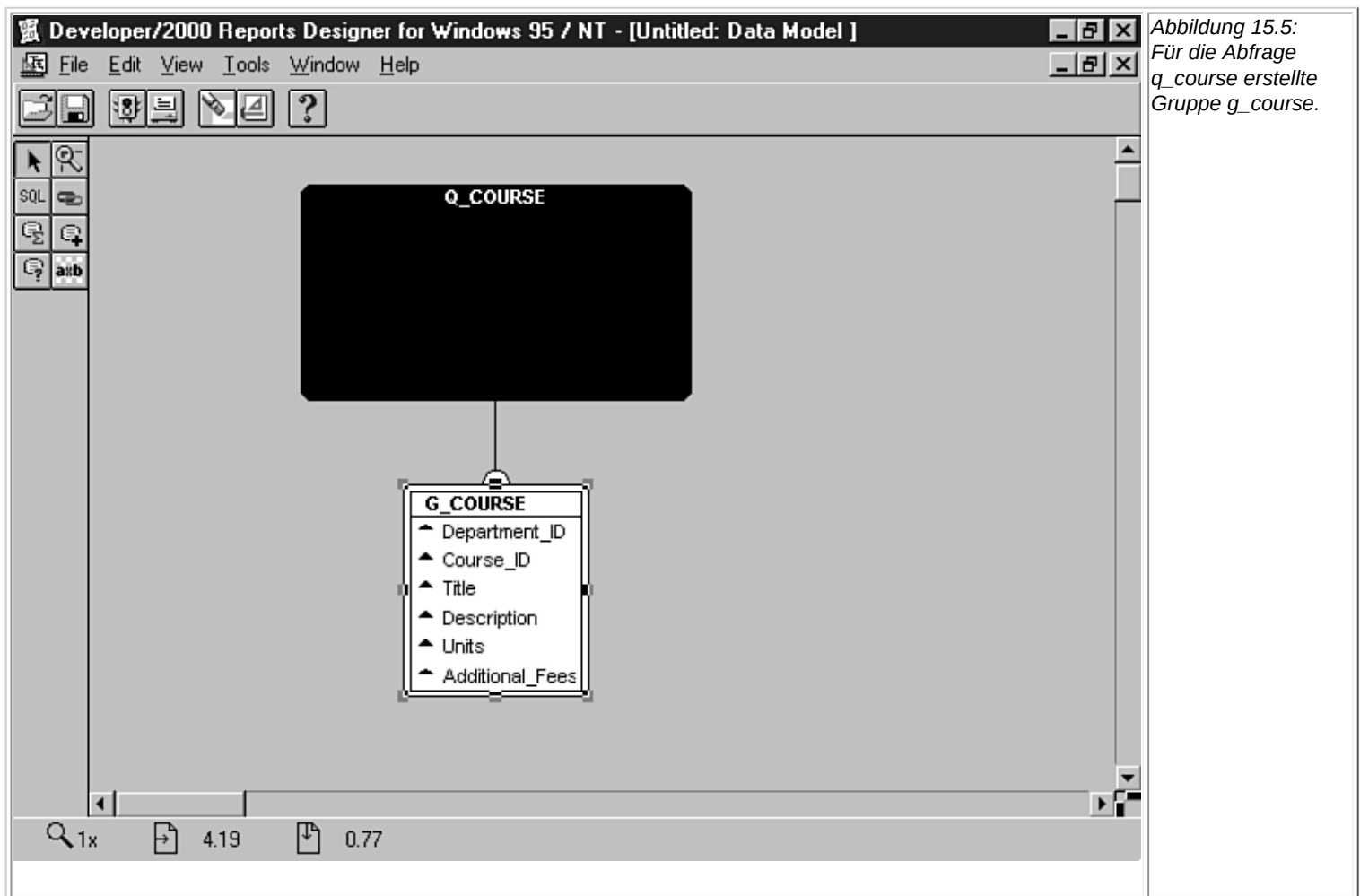


Abbildung 15.5:  
Für die Abfrage  
q\_course erstellte  
Gruppe g\_course.

## Festlegen eines Standard-Layouts

Somit haben Sie das Datenmodell für den Bericht erfolgreich erstellt. Im nächsten Schritt muß das Berichtslayout bestimmt werden.

1. Zur Vereinfachung dieser Aufgabe rufen Sie das Dialogfeld Default Layout auf, indem Sie entweder auf das grüne Dreieck in der horizontalen Werkzeugleiste klicken oder den Menüeintrag Tools | Layout wählen.

2. Im Dialogfeld Default Layout finden Sie zwei Register vor. Aus dem Register Style wählen Sie den Stil Tabular aus (siehe Abbildung 15.6).

3. Wählen Sie das Register Data/Selection, in dem jede Gruppe und die ihnen zugeordneten Spalten enthalten sind.

In der Standardeinstellung erstellt Reports Designer für jede Spalte eine Beschriftung, wobei Unterstriche in Spaltenbezeichnungen durch Leerstellen ersetzt werden und der erste Buchstabe eines jeden Wortes groß geschrieben wird. So bekommt die Spalte Department\_ID die Beschriftung Department Id. Diese Beschriftungen können Sie im Register Data/Selection ändern (siehe Abbildung 16.7).

4. Klicken Sie auf OK, um für diesen Bericht das Standardlayout zu erzeugen, das danach angezeigt werden sollte (siehe Abbildung 15.8). Zugegebenermaßen ist nicht sofort zu erkennen, was durch all die Kästchen und Linien dargestellt wird. Wenn Sie sich das Layout im Object Navigator anschauen, bekommen Sie eine andere Ansicht aller Elemente gezeigt (siehe Abbildung 15.9).

5. Um sich zum Object Navigator zu bewegen, wählen Sie den Menüeintrag Window | Object Navigator. Wie Sie hier erkennen können, kann das Layout eines Berichts aus einem Header, Trailer, Body und Margin bestehen. Durch das Standardlayout wurde nur das Objekt Body erzeugt, unter dem sich ein Objekt befindet namens M\_COURSE\_GRPFR - ein Rahmen für die gesamte Gruppe. Darunter befinden sich zwei weitere Objekt - M\_COURSE\_HDR und R\_COURSE. M\_COURSE\_HDR ist ein Rahmen, der den Header enthält und R\_COURSE ein sich wiederholender Rahmen, der die eigentlichen Felder jedes Datensatzes enthält. R\_COURSE besteht wiederum aus drei weiteren Objekten:

F\_ADDITIONAL\_FEES - Ein Feld, das den Wert von Additional\_Fees enthält.

M\_COURSE\_EXP - Ein Rahmen, der weitere Felder enthält.

M\_COURSE\_EXP1 - Ein weiterer Rahmen, der die für die Abfrage Q\_COURSE bestimmten, verbliebenen Felder enthält.

6. Dieses Standardlayout muß natürlich angepaßt werden. In diesem Bericht wird M\_COURSEEXP oder M\_COURSE\_EXP1 nicht benötigt. Deshalb markieren Sie diese Rahmen im Object Navigator und drücken die Entfernen-Taste.

7. Die Frage des Object Navigator, ob Sie diesen Rahmen tatsächlich entfernen wollen, beantworten Sie, indem Sie auf Yes klicken. Wie in Abbildung 15.10 gezeigt wird, sollten die Felder aus M\_COURSE\_EXP und M\_COURSE\_EXP1 nun direkt unterhalb R\_COURSE zu sehen sein.

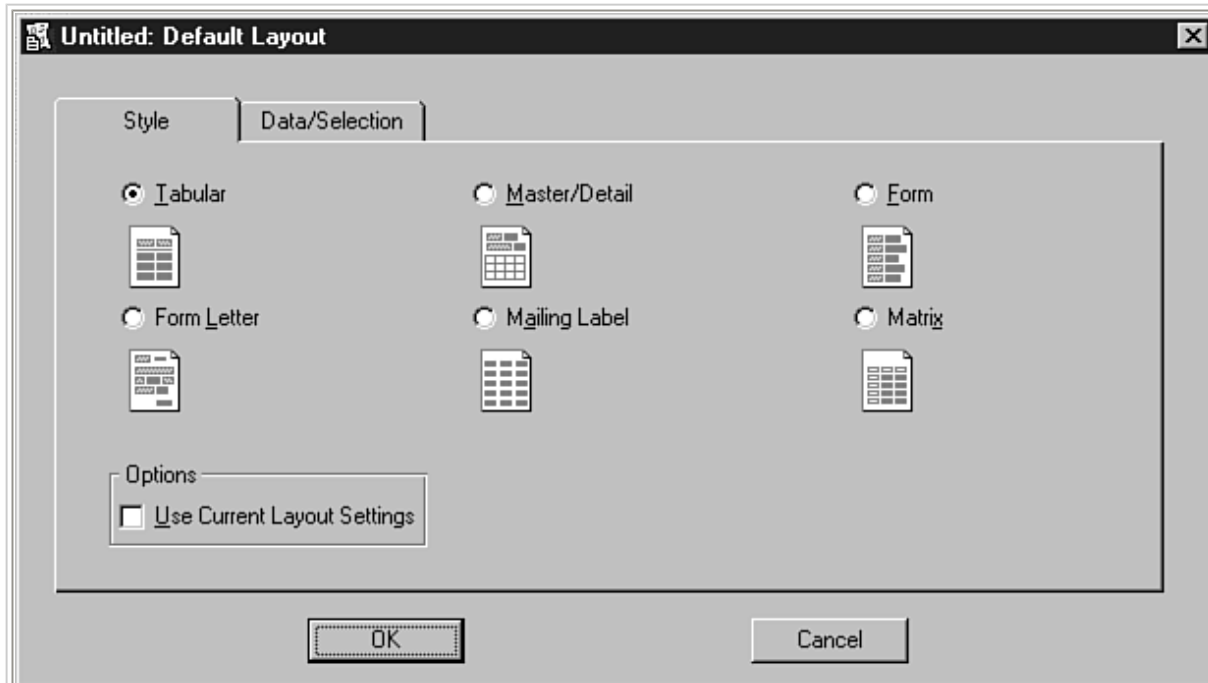


Abbildung 15.6:  
Dialogfeld Default  
Layout.

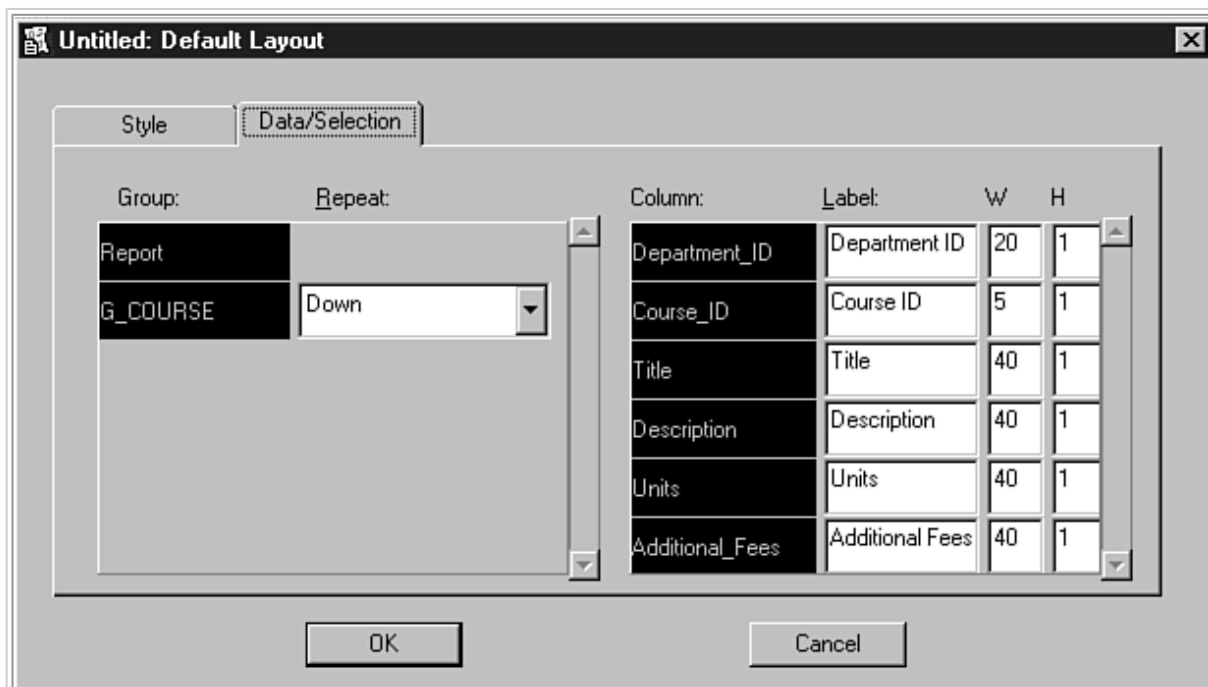


Abbildung 15.7:  
Änderung einer  
Spaltenbeschriftung  
auf der Registerkarte  
Data/Selection.

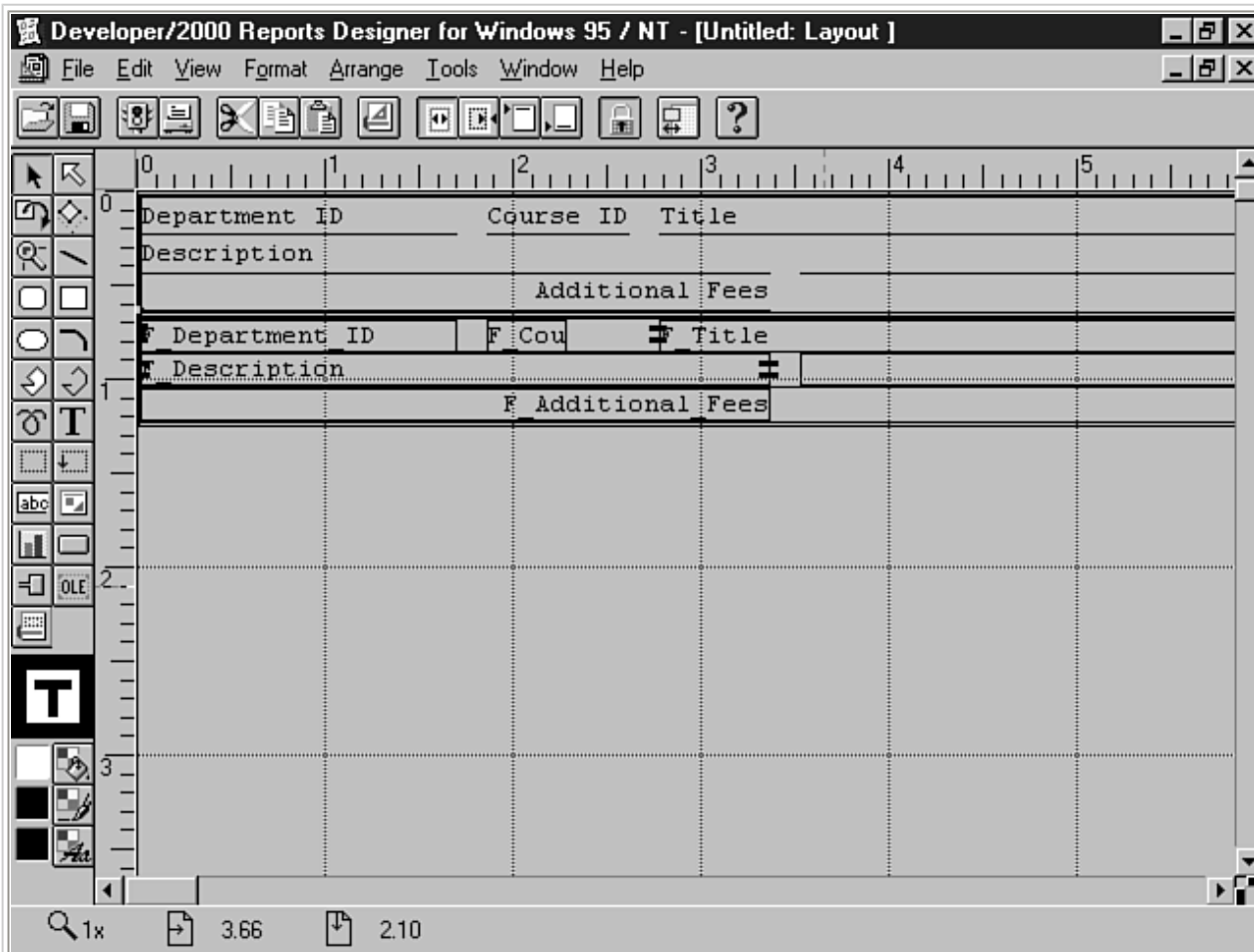


Abbildung 15.8:  
Für den Bericht  
erzeugtes  
Standardlayout.

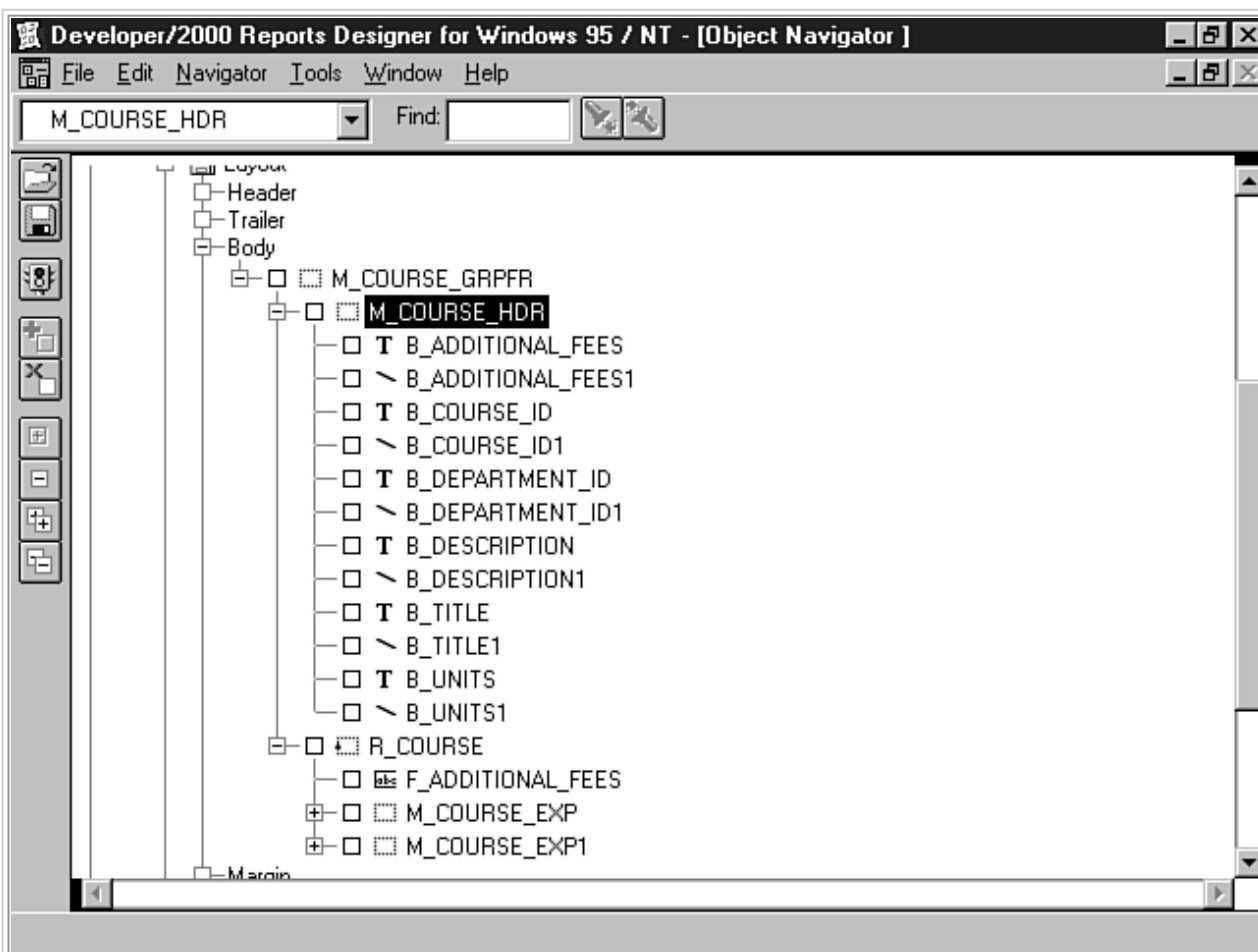
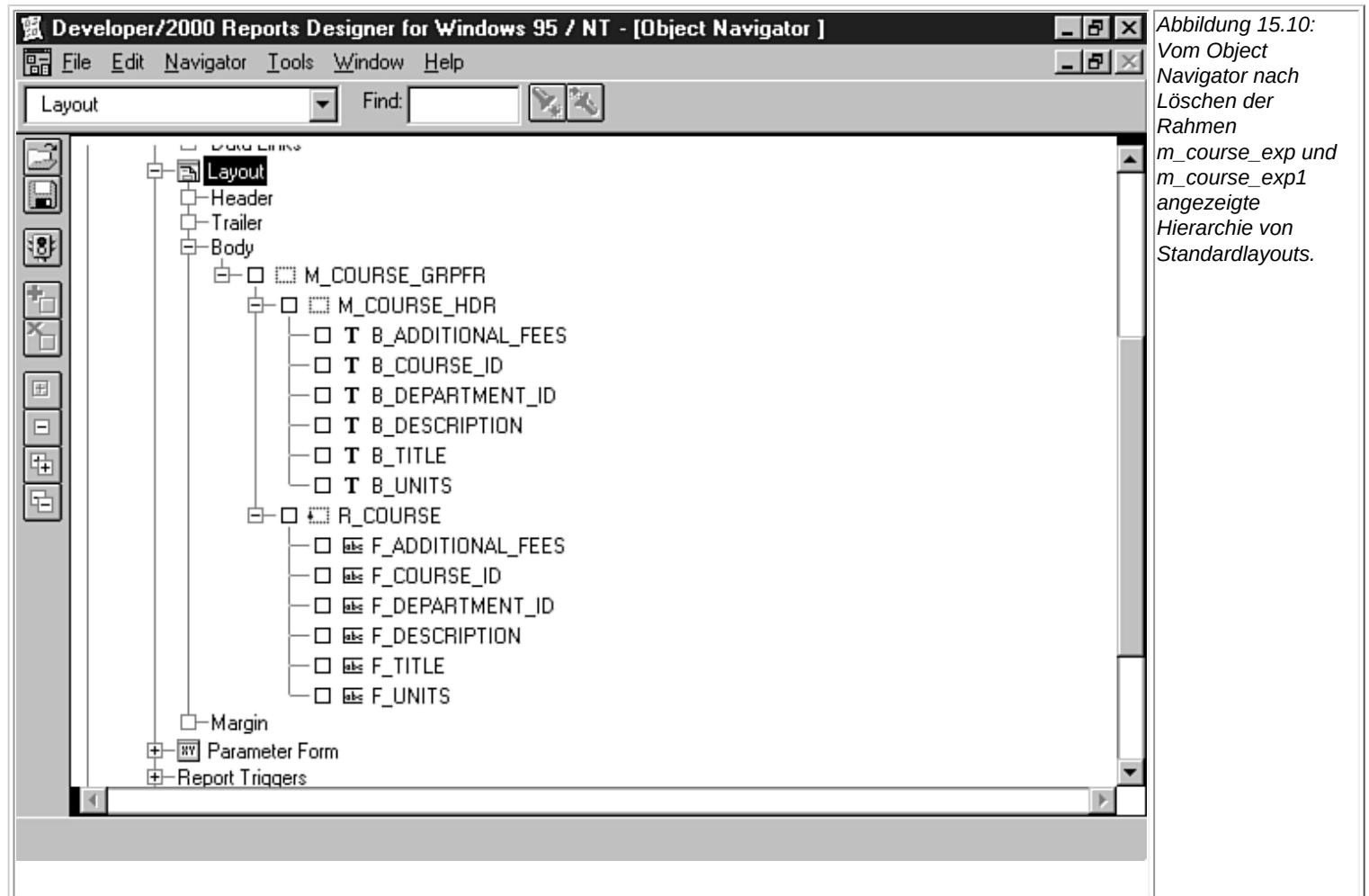


Abbildung 15.9:  
Vom Object  
Navigator  
angezeigte  
Hierarchie von  
Standardlayouts.

Als nächstes muß das Layout mit folgenden Schritten neu formatiert werden:

1. Kehren Sie zum Layout zurück, indem Sie den Menüeintrag Window | Untitled: Layout wählen.
2. Als erstes löschen Sie die Linien im Header-Rahmen; die Spaltenüberschriften können Sie vom Body des Berichts durch das Schriftattribut fett abheben.
3. Wählen Sie jede Linie aus, und drücken Sie die Entfernen-Taste; achten Sie darauf, daß Sie eine Linie und nicht den Header ausgewählt haben.
4. Passen Sie jetzt die Spaltenüberschriften an, indem Sie diese markieren und die linke Maustaste so lange gedrückt halten, bis die richtige Größe erreicht ist.



Als nächstes müssen alle drei Rahmen so angepaßt werden, daß sie auf jeder Seite ein halbes Zoll Rand haben. Außerdem wollen Sie die Spaltenüberschriften neu platzieren.

1. Passen Sie die Größe der Felder an, und stellen Sie sie unter die Spaltenüberschriften.
2. Einige der Spaltenüberschriften sind zu lang. So können Sie z.B. Department ID auf Dep verkürzen, indem Sie die Überschrift anwählen, die Text-Schaltfläche in der Werkzeugleiste (das große T) und erneut die Überschrift anklicken. Jetzt können Sie die Überschrift kürzen.
3. Das Resultat sollte der Darstellung in Abbildung 15.11 entsprechen.

Noch ist das Aussehen nicht zufriedenstellenden. Als erstes ändern Sie mit den folgenden Schritten die Schriftart der Spaltenüberschriften:

1. Klicken Sie mit der Maus alle Spaltenüberschriften an, während Sie die Umschalt-Taste gedrückt halten.
2. Wählen Sie den Menüeintrag Format | Font.
3. Wählen Sie MS Sans Serif, Fett, 12 Punkt, und klicken Sie auf OK.

4. Wiederholen Sie den dritten Schritt für jedes Feld, aber wählen Sie Normal statt Fett.

5. Da sich die Schriftart geändert hat, kann es erforderlich sein, daß Sie einige der Felder in ihrer Größe anpassen müssen, damit alle Zeichen angezeigt werden.

6. Außerdem wollen Sie die Formatmaske für das Feld ändern, das Additional\_Fees anzeigt. In der Standardeinstellung wird kein \$-Zeichen angezeigt. Doppelklicken Sie auf das Feld namens Additional\_Fees.

7. Im Feld Format Mask wählen Sie aus dem Drop-Down-Listefeld \$NNN,NN0 und klicken auf OK.

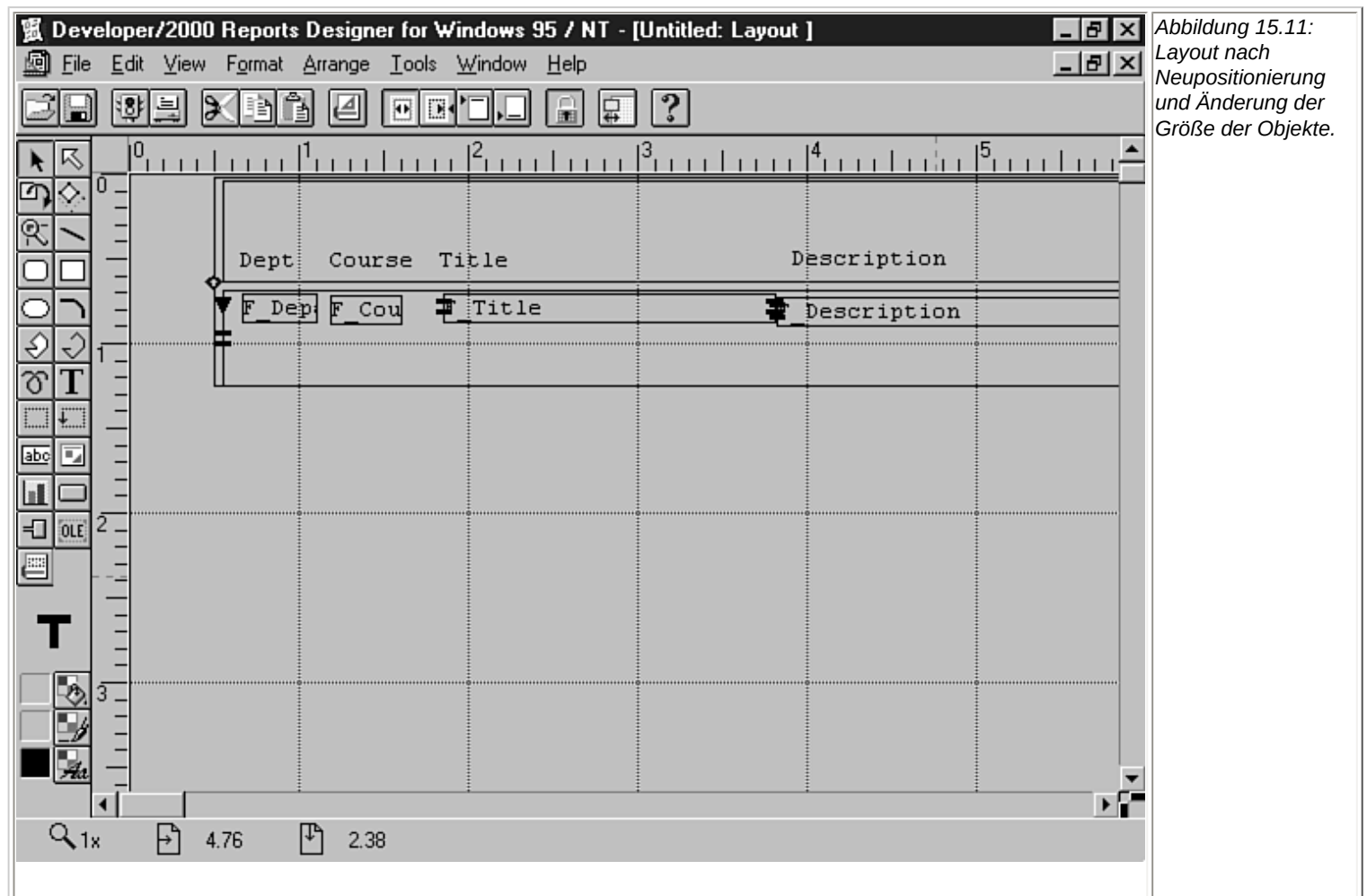


Abbildung 15.11:  
Layout nach  
Neupositionierung  
und Änderung der  
Größe der Objekte.

## Festlegen der Größe und Ausrichtung von Objekten

Sie können für alle Spaltenüberschriften und Felder die gleiche Höhe einrichten, indem Sie für eines der Objekte die Höhe heraufsetzen, so daß alle Zeichen vollständig angezeigt werden.

1. Wählen Sie die weiteren Objekte aus, für die Sie die gleiche Höhe einstellen möchten, und wählen Sie den Menüeintrag Arrange | Size Objects.

2. Im Optionsfeld Width stellen Sie None und im Optionsfeld Height Largest ein.

3. Klicken Sie auf OK. Jetzt sollte Ihr Layout ungefähr dem aus Abbildung 15.12 gleichen.



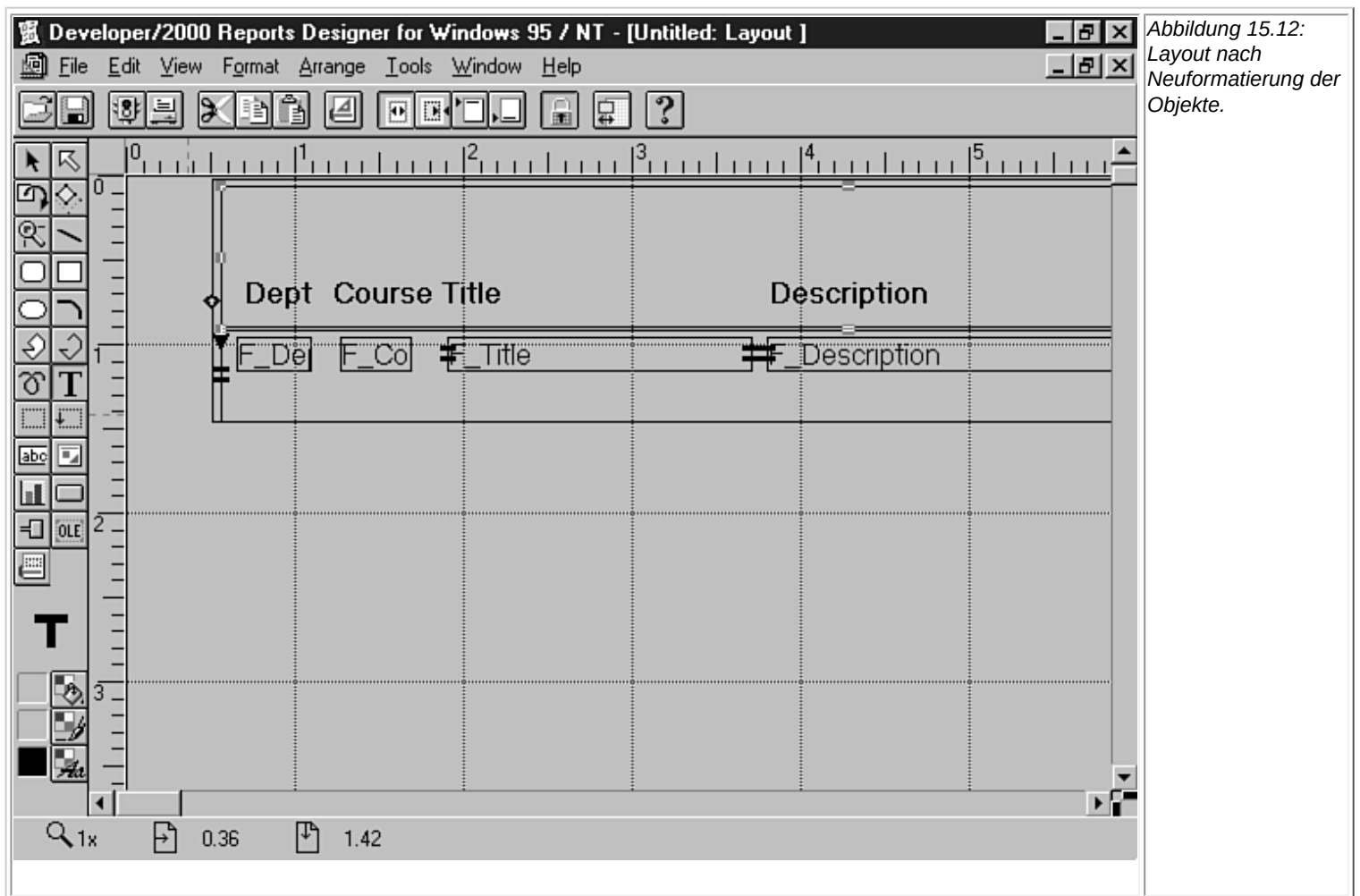


Abbildung 15.12:  
Layout nach  
Neuformatierung der  
Objekte.

### Titel, Datum und Seitenzahlen einem Bericht hinzufügen

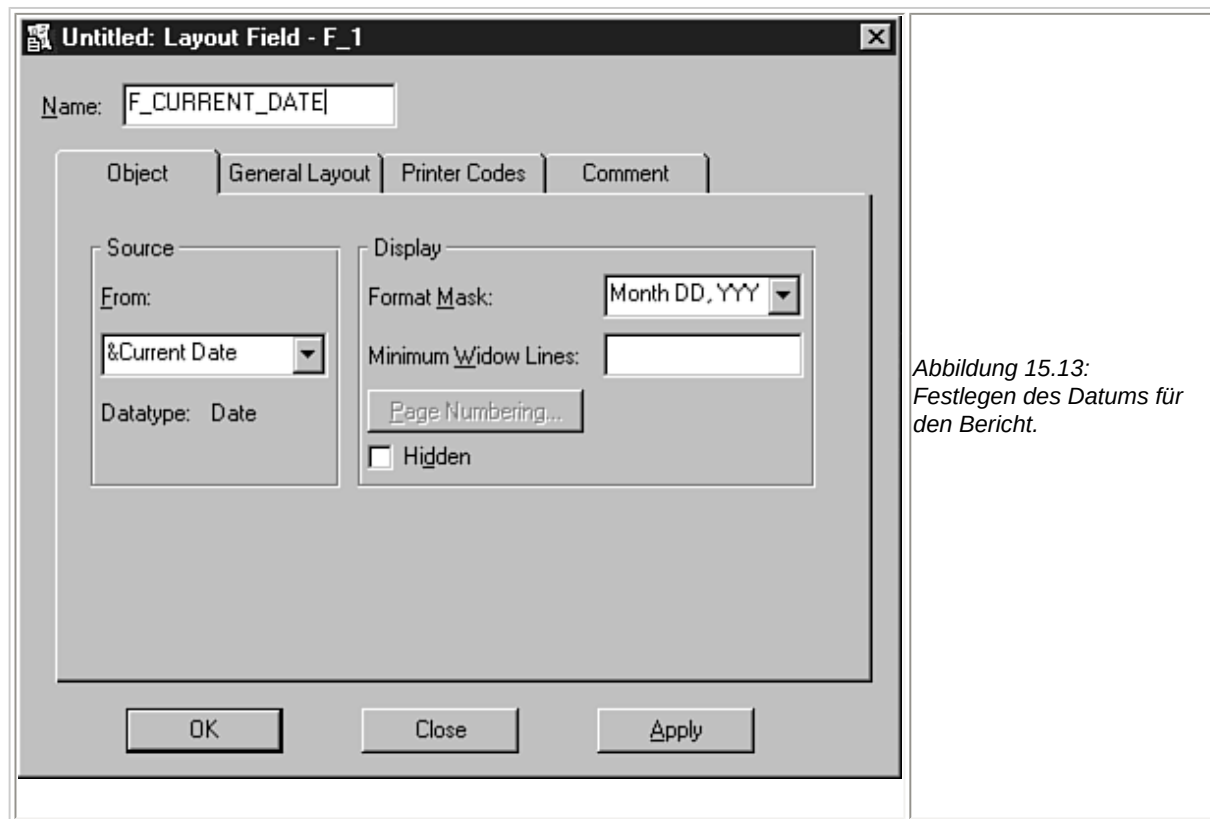
Ehe Sie diesen Bericht mit einem Titel versehen, achten Sie darauf, daß über den Spaltenüberschriften genug Raum für diesen, das Datum und die Seitenzahlen vorhanden ist. Mit den folgenden Schritten fügen Sie einen Titel hinzu:

1. Wählen Sie das Textwerkzeug aus der Werkzeugleiste, und ziehen Sie bei gedrückter linker Maustaste eine Fläche auf, in die Sie den Titel - FLUGLE COLLEGE Course Catalog - eingeben möchten. Drücken Sie zwischen FLUGLE COLLEGE und Course Catalog die Enter-Taste, damit sie untereinander stehen.
2. Um den Text zu zentrieren, wählen Sie den Menüeintrag Format | Alignment | Center.
3. Formatieren Sie den Titel so, daß FLUGLE COLLEGE in MS Sans Serif, Fett, 14 Punkt und Course Catalog mit MS Sans Serif, Fett, 12 Punkt angezeigt werden.

Um dem Bericht das aktuelle Datum hinzuzufügen, führen Sie die folgenden Schritte aus:

1. Wählen Sie das Feld-Werkzeug aus der Werkzeugleiste, und ziehen Sie bei gedrückter linker Maustaste eine Fläche auf, in der Sie das Datum anzeigen lassen möchten.
2. Doppelklicken Sie auf das Feld, und ein Fenster wird aufgeblendet.
3. Ändern Sie den Namen des Felds von F\_1 in F\_CURRENT\_DATE.
4. In der Drop-Down-Liste mit der Überschrift Source wählen Sie &Current Date.
5. In der Formatmaske wählen Sie Month DD,YYYY aus den Formaten der Drop-Down-Liste (siehe Abbildung 15.13).
6. Stellen Sie außerdem die Schriftattribute auf MS Sans Serif, Fett, 10 Punkt ein.

7. Klicken Sie auf OK, um das Feld mit diesen Einstellungen zu versehen.



Um den Bericht mit einer Seitenzahl zu versehen, führen Sie die folgenden Schritte aus:

1. Wählen Sie aus der Werkzeugleiste Field Tool, und ziehen Sie bei gedrückter linker Maustaste eine Fläche auf, in der Sie die Seitenzahl anzeigen lassen wollen.
2. Doppelklicken Sie auf das Feld, und es erscheint ein Fenster mit der Überschrift »Course: Layout Field - F\_1«.
3. Ändern Sie den Namen des Felds von F\_1 in F\_Page.
4. In der Drop-Down-Liste mit der Überschrift Source wählen Sie &Physical Page Number.
5. In der Formatierungsmaske geben Sie NNN ein.
6. Setzen Sie außerdem die Schriftattribute für die Seitenzahl auf MS Sans Serif, Fett, 10 Punkt. Nun sollte Ihr Berichtslayout dem aus Abbildung 15.14 sehr ähnlich sein.
7. Klicken Sie auf OK, um das Feld mit diesen Einstellungen zu versehen.

## Testen des Berichts

Jetzt ist es an der Zeit, den Bericht zu testen.

1. Klicken Sie in der horizontalen Werkzeugleiste auf die grüne Lampe, und ein Fenster mit dem Titel Runtime Parameter Form sollte erscheinen (siehe Abbildung 15.15).
2. In der Standardeinstellung ist Destination Type für den Bericht auf Screen eingestellt.
3. Klicken Sie auf Run Report. Es erscheint ein Fenster - Progress Report -, in dem die Aktivitäten sowohl auf dem Client als auch auf dem Server angezeigt werden. Nach kurzer Zeit müsste auf dem Bildschirm die erste Seite des Berichts wie in Abbildung 15.16 erscheinen.
4. Klicken Sie auf Close, wenn Sie die Überprüfung Ihres Berichts abgeschlossen haben.

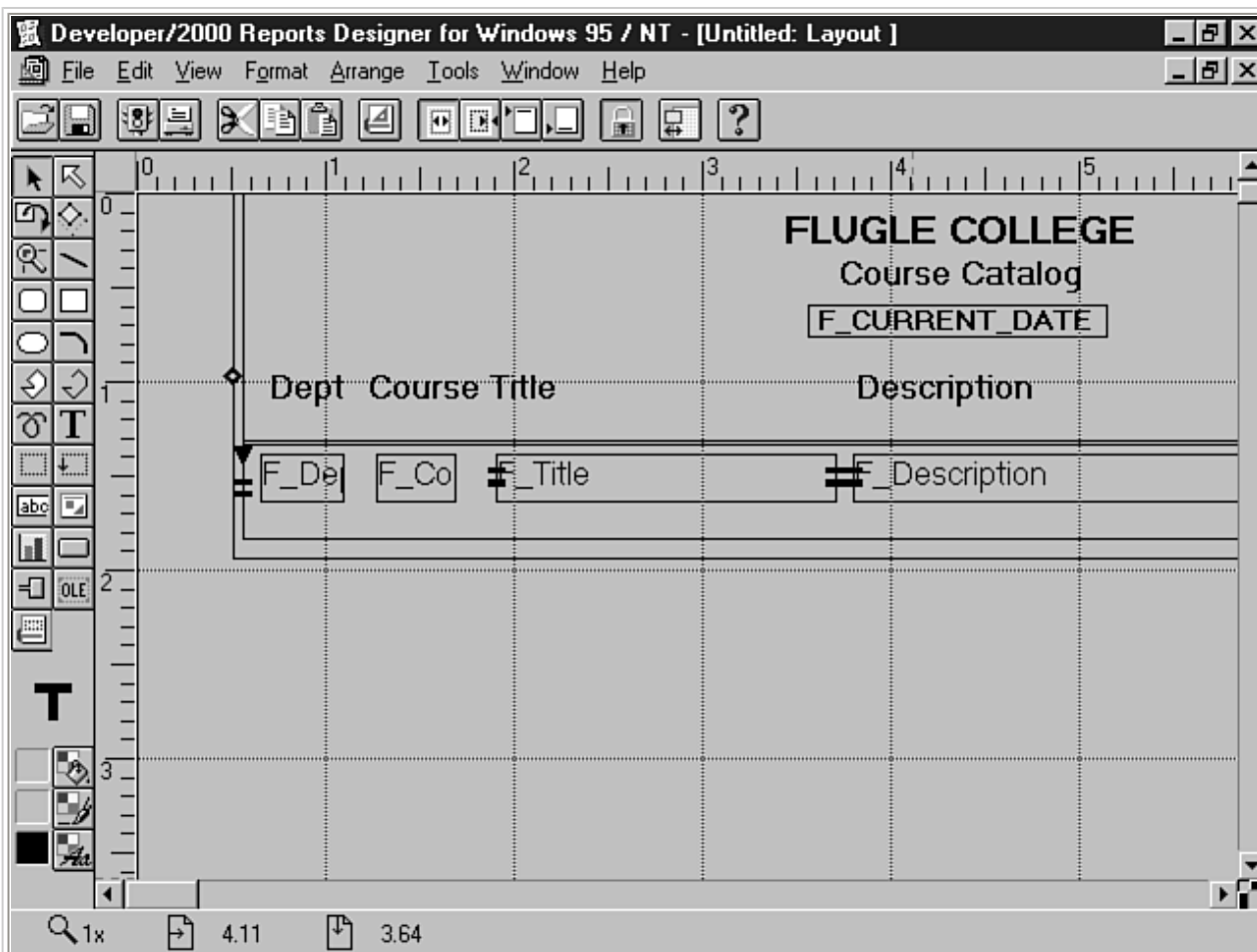


Abbildung 15.14:  
Aktuelles  
Berichtslayout.

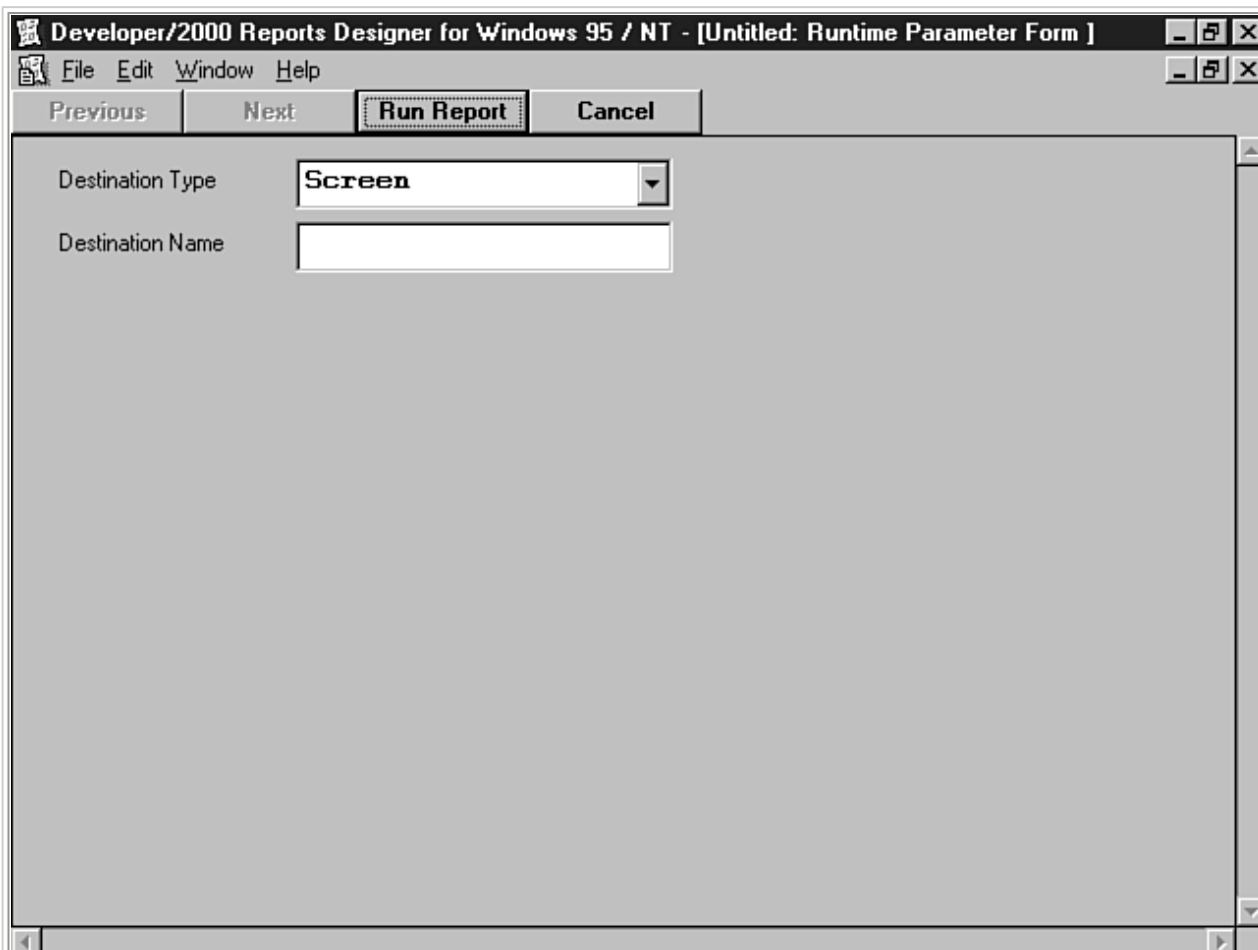


Abbildung 15.15:  
Anzeige des  
Formulars für  
Laufzeitparameter.

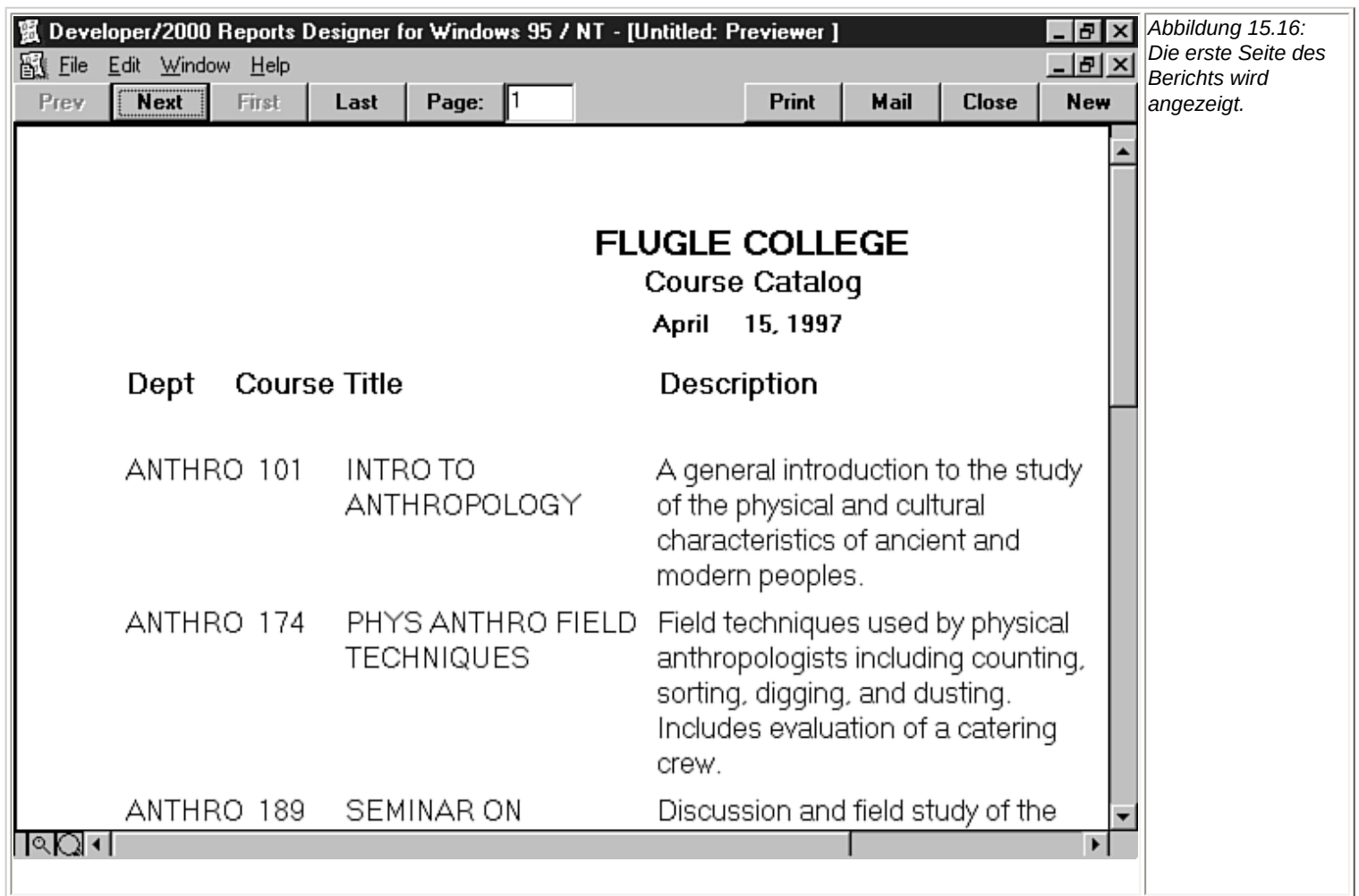


Abbildung 15.16:  
Die erste Seite des  
Berichts wird  
angezeigt.

## Drucken eines Berichts

Es gibt unterschiedliche Möglichkeiten, einen Bericht auszudrucken. Standardmäßig ist im Formular für Laufzeitparameter die Ausgabe auf dem Bildschirm eingestellt; mit anderen Worten, der Benutzer sieht immer erst eine Druckvorschau. Sie können Ihr eigenes Formular für Laufzeitparameter erstellen, in dem die Standardausgabe auf dem Drucker erfolgt. Aber auch aus der Vorschau heraus können Sie einen Bericht drucken, indem Sie die Drucken-Schaltfläche anklicken.

## Speichern des Berichts

Jetzt ist es ratsam, Ihren Bericht zu speichern. Reports Designer kann ebenso wie Forms-Designer eine Oracle-Datenbank als Speicherort benutzen. Um ein Modul in einer Datenbank zu speichern, müssen Sie oder Ihr DBA das von Developer/2000 benötigte Datenbankschema installieren. Mehr darüber können Sie in den *Administrative Notes* in *Developer/2000 Admin for Win95* erfahren.

Für die Beispiele dieses Buchs speichern Sie den Bericht jedoch in einer Datei. Um dies zur Standardeinstellung in Reports Designer zu machen, wählen Sie den Menüeintrag Tools | Tools Option. In der Gruppe Object Access finden Sie ein Feld namens Storage Type. Wählen Sie statt der Standardeinstellung File/Database in der Drop-Down-Liste File, und klicken Sie auf OK.

Führen Sie folgende Schritte zum Speichern eines Berichts aus:

1. Wählen Sie den Menüeintrag File | Save As.
2. Wählen Sie das von Ihnen zur Speicherung Ihrer anwendungsspezifischen Dateien verwendete Verzeichnis aus.
3. Geben Sie Course als Dateiname ein.
4. Klicken Sie auf OK. Reports Designer speichert die Datei unter dem Namen Course.rdf im angegebenen Verzeichnis.



*RD erlaubt Ihnen, mehrere Berichte gleichzeitig geöffnet zu haben. Sie werden dies als angenehme Arbeitserleichterung beim Kopieren von Objekten zwischen Berichten erleben.*

---

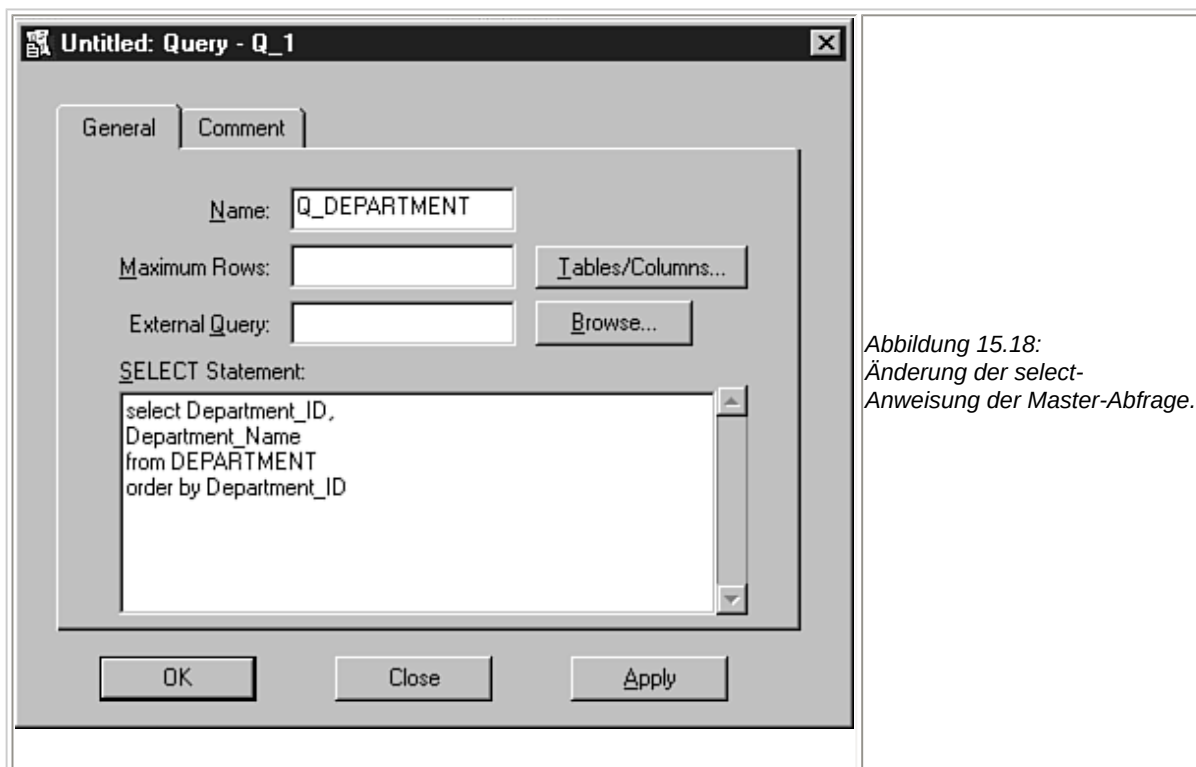
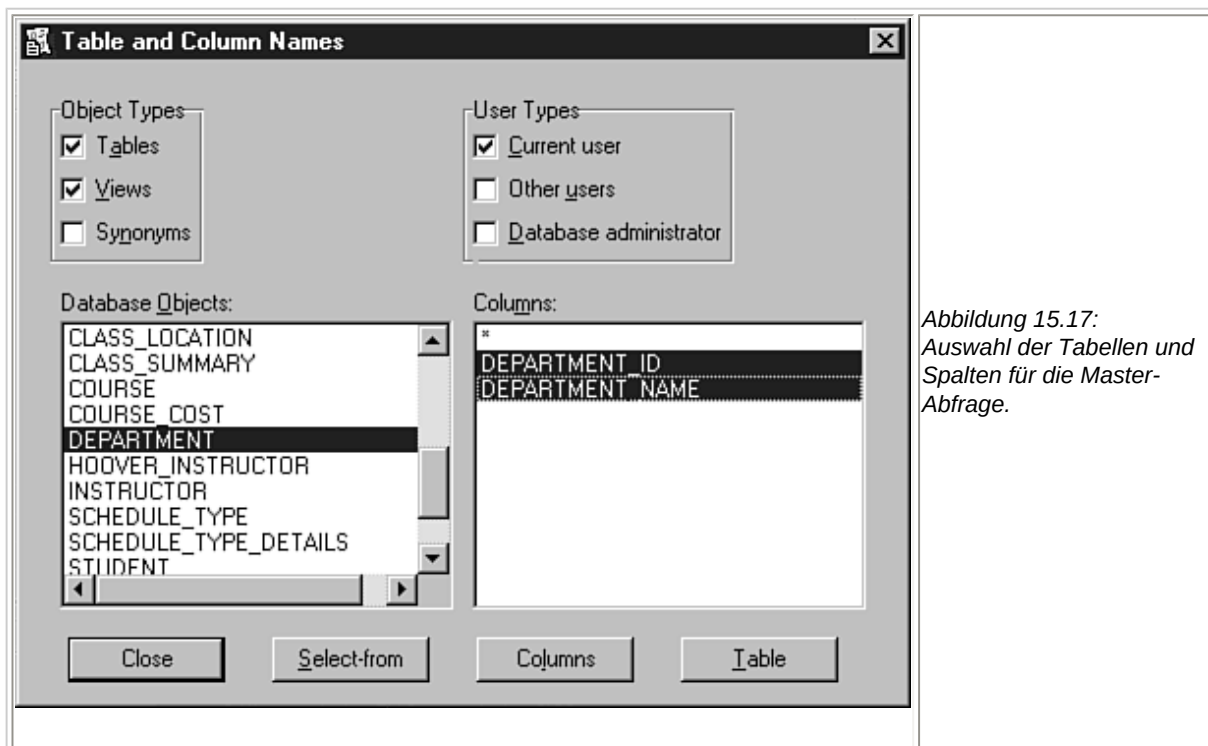
## Erstellen eines Master-/Detail-Berichts

Ein weiterer gebräuchlicher Berichtstyp ist der Master-/Detail-Bericht. Ein Beispiel wäre ein Bericht, der jede Abteilung mit den dazugehörigen Lehrkräften auflisten würde. Die Tabelle Department ist der Master-Datensatz und die Tabelle Instructor der Detail-Datensatz.

## Erstellen eines Datenmodells

Als erstes müssen Sie ein Datenmodell für den Bericht erstellen.

1. Rufen Sie den Data Model Editor auf, indem Sie entweder das Symbol links des Data Model-Knotens im Object Navigator anklicken oder den Menüeintrag Tools | Data Model Editor wählen.
2. Um eine Master-Abfrage zu erstellen, klicken Sie auf die Schaltfläche SQL in der Werkzeugleiste auf der linken Seite des Fensters.
3. Bewegen Sie die Maus in den Hauptteil des Fensters, halten Sie die linke Maustaste gedrückt, und ziehen Sie die Maus nach rechts unten, bevor Sie die Maustaste loslassen. Doppelklicken Sie auf Abfrage Q\_1.
4. Ändern Sie den Namen in Q\_DEPARTMENT um.
5. Klicken Sie auf die Schaltfläche Tables/Columns. In einem weiteren Fenster namens Table and Column Names können Sie die für die Abfrage zu verwendende Tabelle und die Spalten auswählen.
6. Im Listenfeld mit der Überschrift Database Objects blättern Sie nach unten bis zur DEPARTMENT-Tabelle und wählen diese aus.
7. Wählen Sie im Listenfeld Columns jede der Spalten aus (siehe Abbildung 15.17).
8. Um die ausgewählte Tabelle und die Spalten in der SELECT-Anweisung der Abfrage zu verwenden, klicken Sie auf SELECT-FROM und CLOSE. Jetzt sehen Sie die aus der Tabelle und den Spalten erstellte SELECT-Anweisung.
9. Bringen Sie die Spaltennamen in die im Bericht gewünschte Reihenfolge: Department\_ID und Department\_Name.
10. Fügen Sie der SELECT-Anweisung außerdem eine ORDER BY-Klausel hinzu, damit die Zeilen nach Department\_ID sortiert angezeigt werden, wie es in der Abbildung 15.18 zu sehen ist.
11. Klicken Sie auf OK, um zum Data Model Editor zurückzukehren. Als Sie die SELECT-Anweisung für die Abfrage definiert haben, hat Reports Designer für die Master-Abfrage eine Gruppe - G\_DEPARTMENT - erstellt.



Nun können Sie mit folgenden Schritten eine Detail-Abfrage erstellen:

1. Klicken Sie in der vertikalen Werkzeugleiste auf das SQL-Werkzeug. Die Abfrage wird Q-1 benannt sein, da Sie die Master-Abfrage umbenannt haben.
2. Doppelklicken Sie auf die Abfrage Q\_1.
3. Ändern Sie den Namen in Q\_INSTRUCTOR.
4. Klicken Sie auf die Schaltfläche Tables/Columns.
5. Wählen Sie im Listenfeld Columns jede der Spalten aus.

Um die ausgewählte Tabelle und die Spalten in der SELECT-Anweisung der Abfrage zu verwenden, führen Sie folgende Schritte aus:

1. Klicken Sie auf SELECT-FROM und CLOSE.

2. Stellen Sie die Spaltennamen in der SELECT-Anweisung in die im Bericht gewünschte Reihenfolge: Department\_ID, Instructor\_ID, Last\_Name, First\_Name, MI und Position.

Obwohl Department\_ID schon Bestandteil der Master-Abfrage ist, wird sie auch in der Detail-Abfrage benötigt, damit beide Abfragen auf Basis der gemeinsamen Spalte verknüpft werden können.

3. Fügen Sie der SELECT-Anweisung außerdem eine ORDER BY-Klausel hinzu, damit die Zeilen nach Last\_Name, First\_Name und MI sortiert angezeigt werden, wie es in Listing 15.1 zu sehen ist.

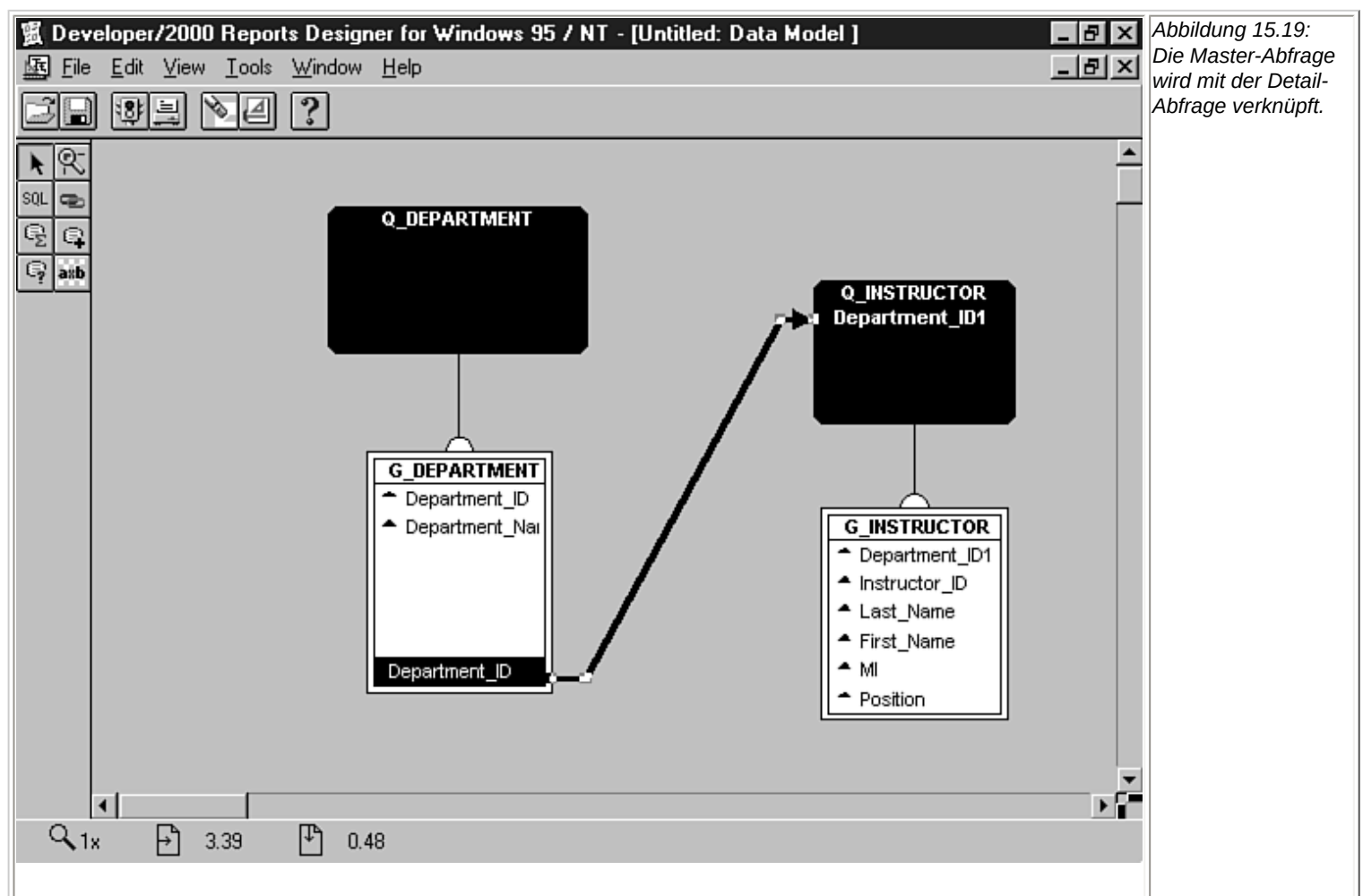
4. Klicken Sie auf OK, um zum Data Model Editor zurückzukehren.

**Listing 15.1: Die SELECT-Anweisung der Detail-Abfrage.**

```
Select Department_ID, Instructor_ID,
Last_Name, First_Name, MI, Position
from Instructor
order by Last_Name, First_Name, MI;
```

Als Sie die SELECT-Anweisung für die Abfrage definiert haben, hat Reports Designer für die Detail-Abfrage eine Gruppe - G\_INSTRUCTOR - erstellt.

Um die Master-Abfrage mit der Detail-Abfrage zu verknüpfen, wählen Sie in der vertikalen Werkzeugleiste das Data-Link-Werkzeug. Setzen Sie den Mauszeiger über Q\_DEPARTMENT, halten Sie die linke Maustaste gedrückt, ziehen Sie den Mauszeiger über Q\_INSTRUCTOR, und lassen Sie die Maustaste los. Vom Data Model Editor sollte eine Verknüpfung zwischen Q\_DEPARTMENT und Q\_INSTRUCTOR über die Department\_ID erstellt werden (siehe Abbildung 15.19).



## Festlegen eines Standard-Layouts



Da Sie nun ein Datenmodell für den Master-/Detail-Bericht erstellt haben, muß als nächstes das Berichtslayout bestimmt werden.

1. Rufen Sie das Dialogfeld Default Layout auf, indem Sie entweder auf die grüne Lampe in der horizontalen Werkzeugleiste klicken oder den Menüeintrag Tools | Default Layout wählen.
2. Im Register Style wählen Sie den Stil Master-/Detail aus (siehe Abbildung 19.20).
3. Wählen Sie das Register Data/Selection, in dem Sie eine Beschriftung ändern können. So könnten Sie z.B. die Breite eines Felds ändern, oder dessen Namen abkürzen, wie es in Abbildung 15.21 gezeigt wird.

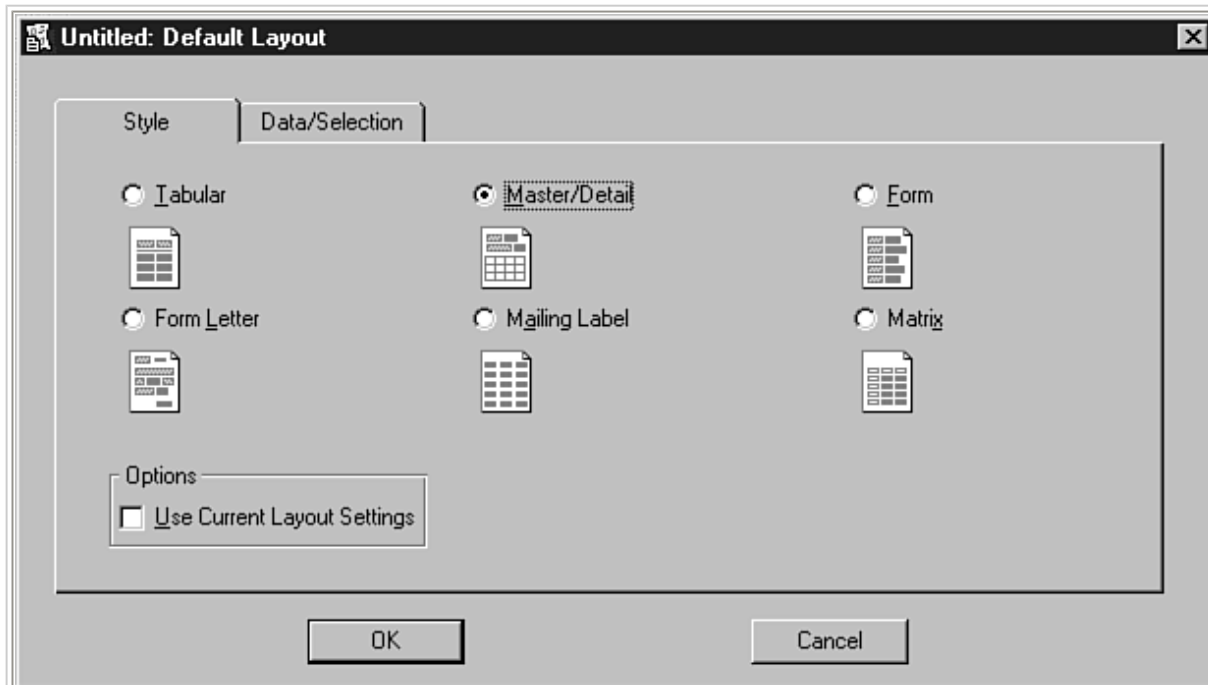


Abbildung 15.20:  
Festlegen von  
Master-/Detail als  
Berichtsform.

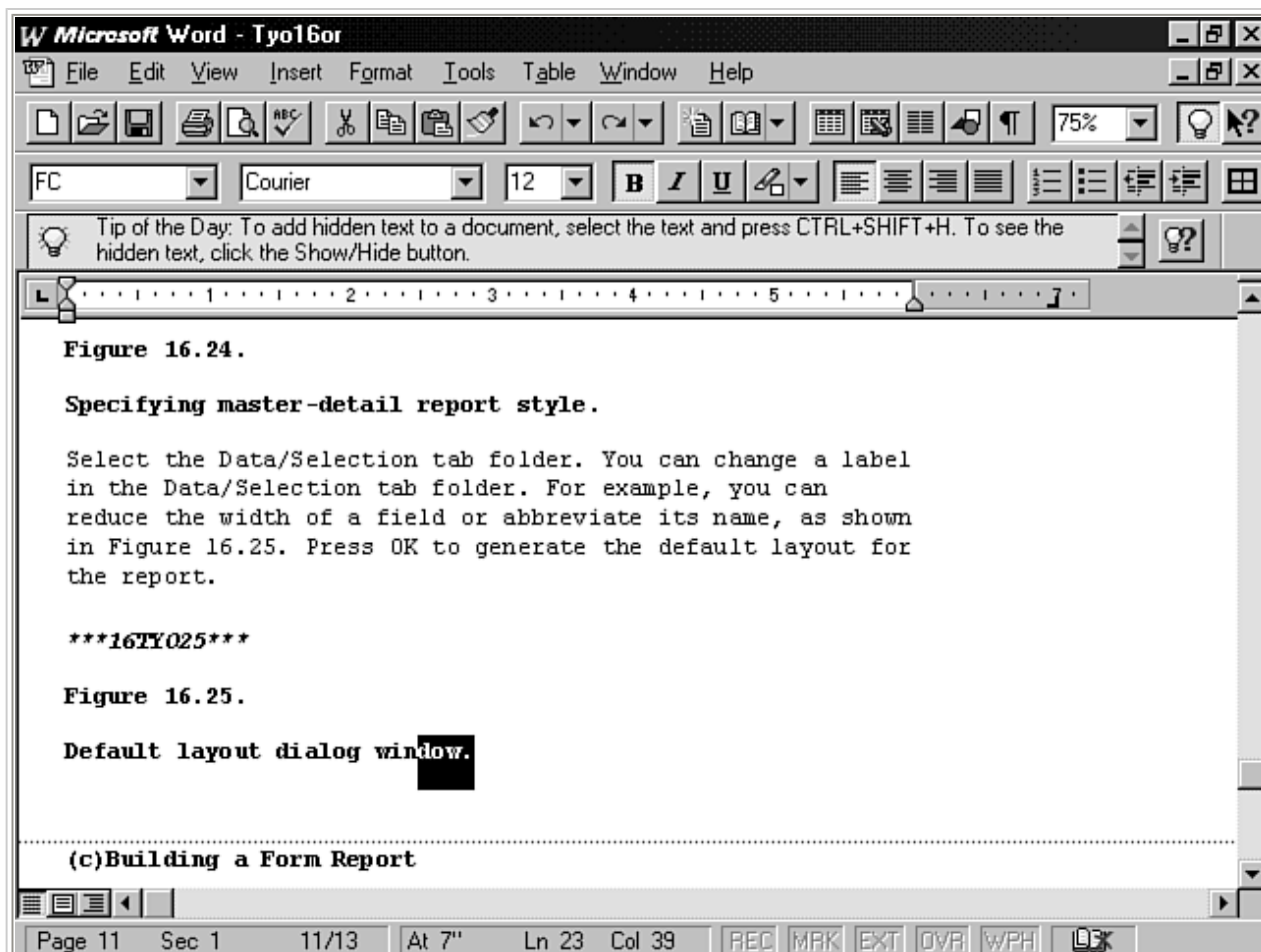


Abbildung 15.21:  
Änderung von  
Beschriftungen und  
Feldbreiten.



Das entstandene Layout muß geändert werden.

1. Im Object Navigator entfernen Sie die Detail-Spaltenüberschrift von Department\_ID.
2. Entfernen Sie alle Linien unter den Spaltenüberschriften.
3. Im Layout Editor stellen Sie alle Felder so um, daß sie zueinander und mit den Spaltenüberschrift bündig angeordnet sind.
4. Ändern Sie für alle Objekte die Schriftattribute auf MS Sans Serif, Normal und 12 Punkt.
5. Ändern Sie die Spaltenbeschriftungen in Fett. Abbildung 15.22 illustriert, wie das Berichtslayout aussehen sollte.

Eine Einstellung muß noch geändert werden. Der Detail-Rahmen - Q\_INSTRUCTOR - enthält die Department\_ID, damit die Detail-Abfrage mit der Master-Abfrage verknüpft werden kann. Sie muß aber nicht auch im Detail-Rahmen angezeigt werden, da sie bereits im Master-Rahmen angezeigt wird.

1. Um dieses Feld zu verbergen, klicken Sie im Detail-Rahmen doppelt auf F\_DepartmentID1. Im darauf erscheinenden Fenster können Sie angeben, daß das angezeigte Feld versteckt werden soll (siehe Abbildung 15.23).
2. Markieren Sie das mit Hidden betitelte Feld.
3. Klicken Sie auf OK.

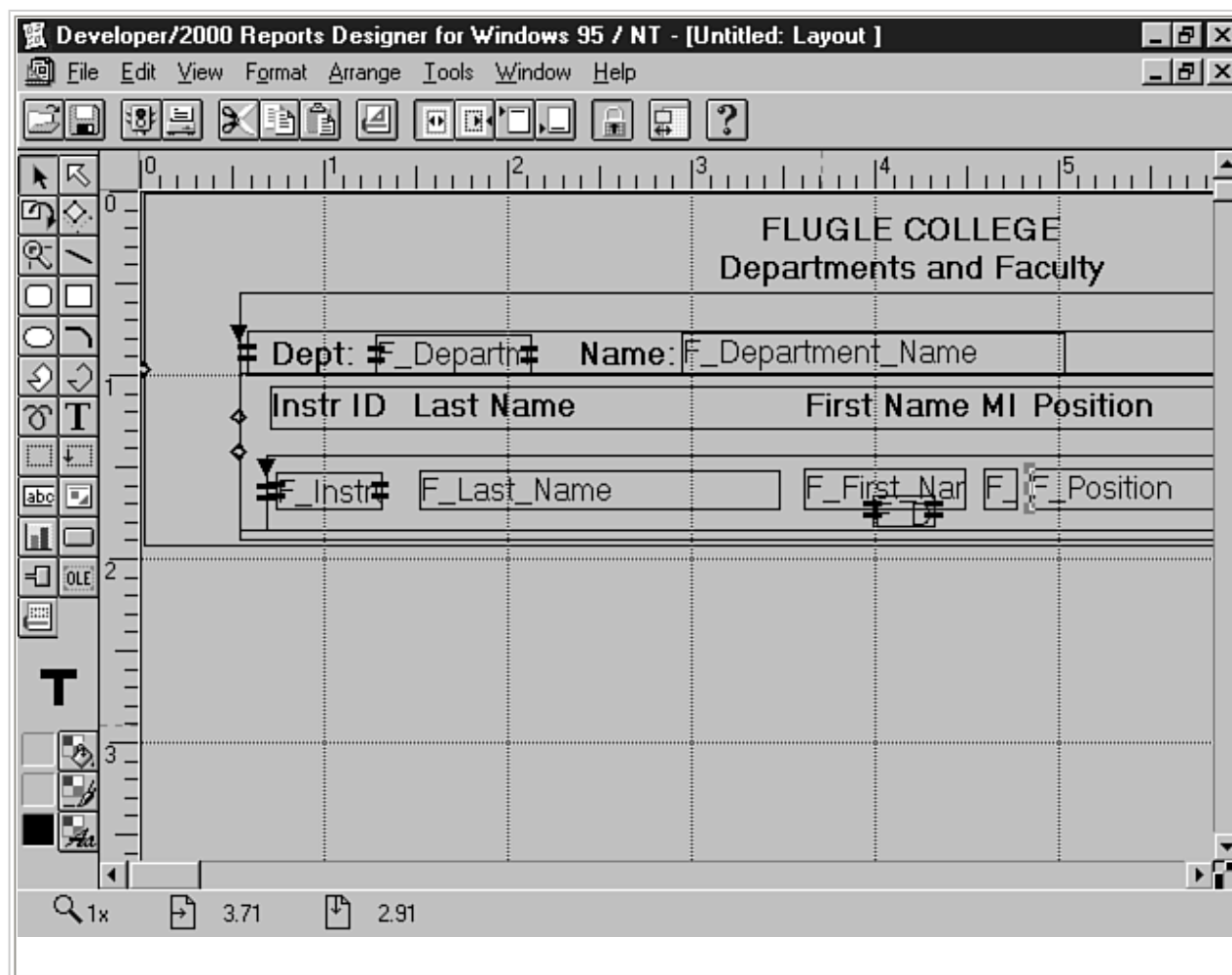


Abbildung 15.22:  
Vollständiges Layout  
für den Master-  
/Detail-Bericht.

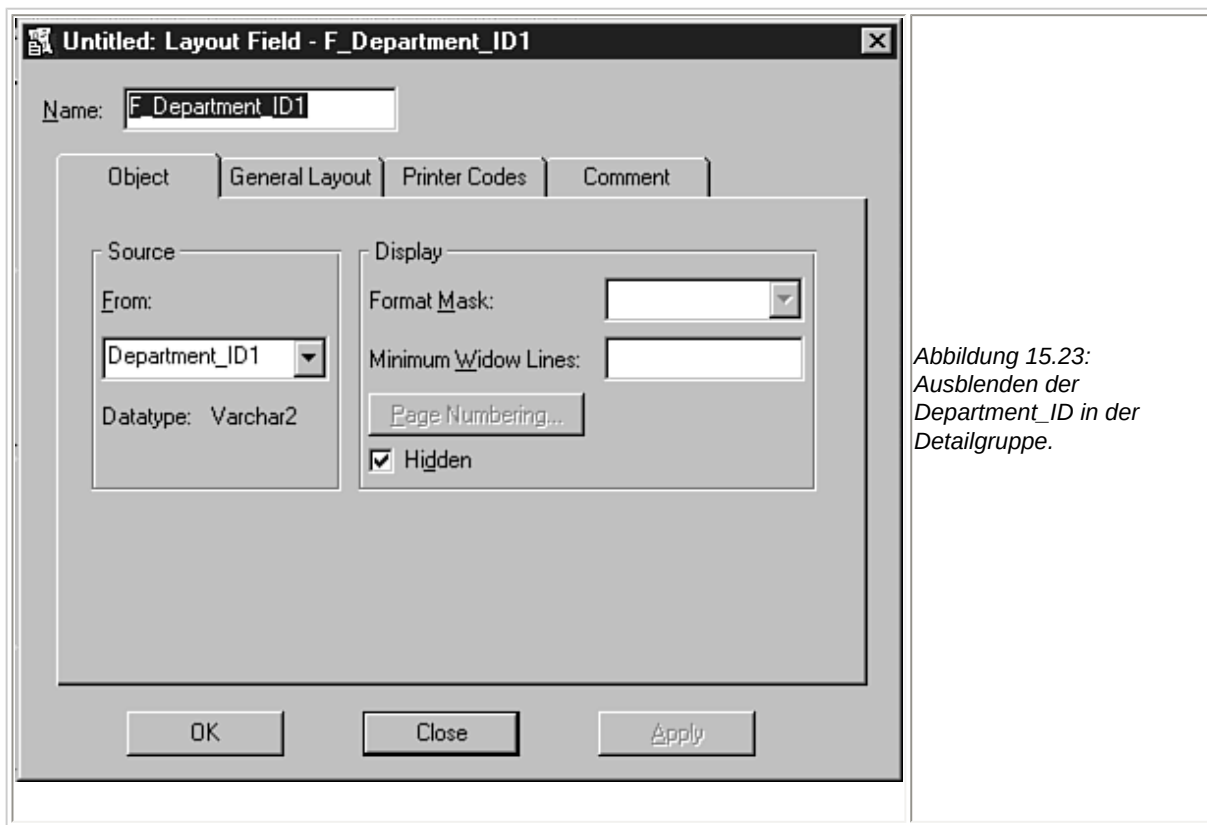


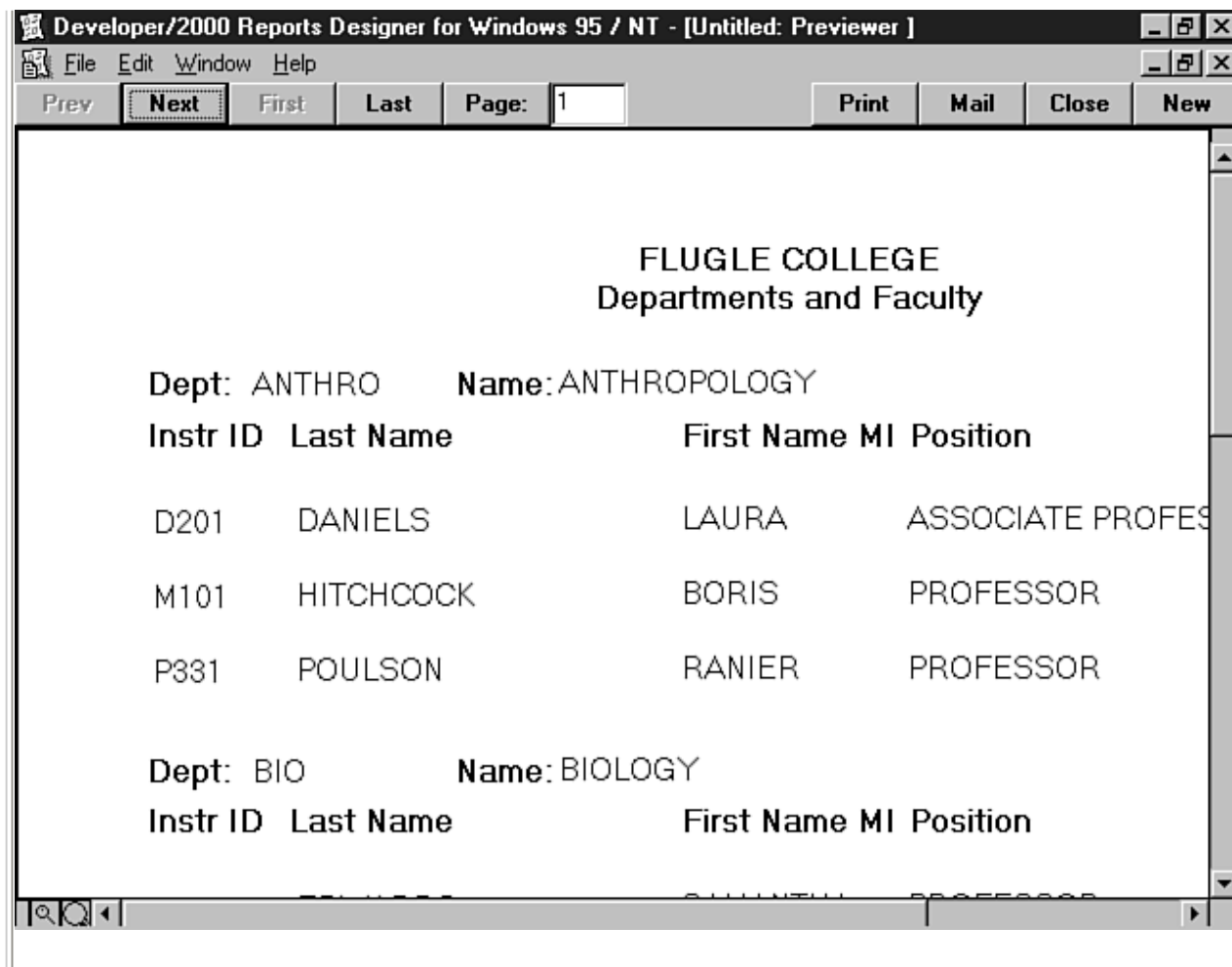
Abbildung 15.23:  
Ausblenden der  
Department\_ID in der  
Detailgruppe.

## Testen des Berichts

Dieser Abschnitt führt Sie durch den Test des Berichts.

1. Klicken Sie in der horizontalen Werkzeugleiste auf die grüne Lampe.
2. Im Formular für Laufzeitparameter klicken Sie auf Run Report. Das Fenster Report Progress wird angezeigt, in dem die Aktivitäten sowohl auf dem Client als auch auf dem Server angezeigt werden. Nach kurzer Zeit müßte auf dem Bildschirm die erste Seite des Berichts wie in Abbildung 15.24 erscheinen.
3. Klicken Sie auf Close, wenn Sie die Überprüfung Ihres Berichts abgeschlossen haben. Wenn Sie es wünschen, können Sie den Bericht im Anwendungsverzeichnis unter dem Namen Dept\_Instructor.rdf speichern.

Abbildung 15.24:  
Die erste Seite des  
Master-/Detail-  
Berichts wird  
angezeigt.



## Erzeugen einer Laufzeit-Version eines Berichts

Um ein Formular aus einer Formularanwendung heraus oder über eine Desktop-Verknüpfung auszuführen, müssen Sie eine Laufzeit-Version des Berichts erzeugen - die eine REP-Datei ist. Während im Reports Designer ein Bericht geöffnet ist, können Sie mit den Taste Strg+t oder über die Menüauswahl File | Administration | Generate eine Laufzeit-Version erzeugen. Standardmäßig geht Reports Designer davon aus, daß Sie für die REP-Datei den gleichen Namen wie für die RDF-Datei verwenden wollen.

## Verwenden eines Format-Trigger

Mit dem Reports Designer stehen Ihnen viele Möglichkeiten zur Anpassung eines Berichts zur Verfügung. So können Sie z.B. für ein Feld einen Formatierungs-Trigger bestimmen, damit das Aussehen eines Felds vom Inhalt abhängig wird. Wir wollen an dem von Ihnen erstellten Master-/Detail-Bericht ein Beispiel durchgehen.

Nehmen Sie an, im Bericht soll hervorgehoben werden, daß eine Lehrkraft Professor ist, indem die Schrift im Feld Instrucor\_ID vergrößert wird. Dazu doppelklicken Sie auf das Feld F\_Instructor\_ID. Im dann erscheinenden Fenster wählen Sie das Register General Layout. In der unteren linken Ecke des Registerblatts klicken Sie auf den mit Edit bezeichneten Schlüssel. Der Program Unit Editor wird angezeigt; standardmäßig zeigt der Editor eine Funktion namens F\_Instruc-tor\_IDFormatTrigger an. Das Fenster mit den Layouteigenschaften des Felds wird weiterhin angezeigt. Verschieben Sie dieses Fenster, um die Funktion bearbeiten zu können, wie es in Abbildung 15.25 gezeigt wird.

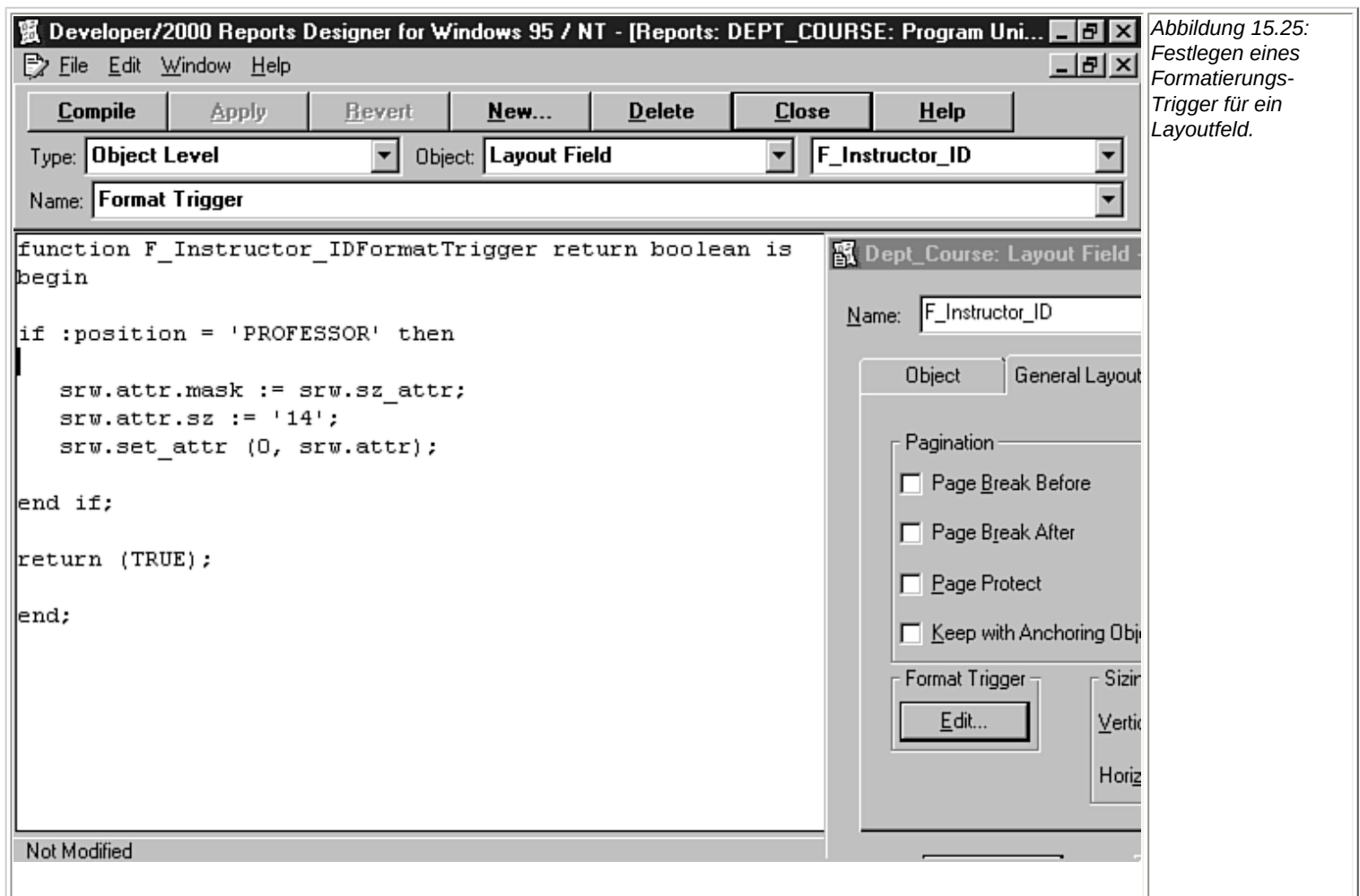


Abbildung 15.25:  
Festlegen eines  
Formatierungs-  
Trigger für ein  
Layoutfeld.

Die Funktion verwendet ein integriertes Paket namens SRW, das zur Änderung der Formateigenschaften eines mit einem Formatierungs-Trigger verbundenen Felds eingesetzt wird. Die if-Anweisung überprüft, ob der aktuelle Wert des Felds Position PROFESSOR ist; wenn dies der Fall ist, werden die drei SRW-Aufrufe ausgeführt. Die erste Zeile identifiziert das zu ändernde Formatattribut - sz\_attr -, das die Schriftgröße für das Feld darstellt. Die zweite Zeile stellt die Schriftgröße auf 14 Punkt. Die dritte Zeile ruft die Funktion set\_attr auf, die Attribute des Felds ändert, indem sie zwei Argumente zurückgibt:

- 0, das angibt, daß die Attribute auf das aktuelle Feld angewendet werden
- srw.attr, das die zu ändernden Attribute enthält.

Die Funktion gibt einen Wert vom Typ Boolean zurück. Ist dieser Wahr, wird das Feld im Bericht mit den jeweiligen vom Formatierungs-Trigger geänderten Attributen angezeigt; ist der Wert Falsch, wird das Feld im Bericht nicht angezeigt.

## Erzeugen einer Verknüpfung zum Starten eines Berichts

Soll der Benutzer einen Bericht mit einem einzigen Mausklick ausführen können, ist eine Verknüpfung eine echte Vereinfachung. In diesem Abschnitt werden Sie durch die erforderlichen Schritte zur Erstellung einer Verknüpfung geführt. Ehe Sie eine Verknüpfung erstellen können, müssen Sie aus der RDF-Datei eine REP-Datei, wie im vorhergehenden Abschnitt beschrieben, erzeugen.

Die Verknüpfung erstellen Sie wie folgt:

1. Klicken Sie auf dem Windows-Desktop mit der rechten Maustaste auf Neu | Verknüpfung.
2. Im Fenster Verknüpfung erstellen müssen Sie die Befehlszeilenargumente eingeben. Vorher schauen Sie sich noch einige von Reports Designer verwendete Argumente an:

- userid identifiziert den Benutzer, das Kennwort und, wenn vorhanden, den Datenbank-Alias zur Verbindung mit der Datenbank.
- report identifiziert den auszuführenden Bericht.
- paraform bestimmt, ob das Formular für Laufzeitparameter angezeigt werden soll.

- destype legt die Ausgabe fest; Bildschirm, Drucker, Datei oder Mail.

3. Um den Bericht Department Instructors zur Laufzeit auszuführen, geben Sie in das Befehlszeilenfeld folgendes ein:

c:\Orawin95\bin\r25run32 report=c:\Tyο\Dept\_Instructor paraform=no destype=screen

4. Klicken Sie auf die Schaltfläche Weiter.

5. Geben Sie der Verknüpfung z.B. den Namen Report - Department with Instructors.

6. Klicken Sie auf die Schaltfläche Beenden.

Die neue Verknüpfung sollte auf dem Desktop angezeigt werden. Probieren Sie die Verknüpfung aus. Sie werden zur Eingabe des Benutzernamens, des Kennworts und der Datenbank aufgefordert. Möchten Sie, daß der Benutzer nicht zur Eingabe dieser Werte aufgefordert wird, ändern Sie Verknüpfung wie folgt ab:

```
c:\Orawin95\bin\r25run32 userid=flugle/flugle report=c:\Tyο\Dept_Instructor paraform=no destype=screen
```



*Ein Nachteil dieser Verknüpfung ist, daß sie das Benutzerkennwort enthält und es von Dritten eingesehen werden kann.*

---

## Aufrufen eines Berichts aus einem Oracle-Formular

Wie schon erwähnt wurde, ist Developer/2000 eine integrierte Umgebung zur Anwendungsentwicklung. In diesem Abschnitt lernen Sie, wie Sie aus einer Formularanwendung einen Bericht ausführen lassen können. Als Beispiel werden Sie erfahren, wie Ihr Menü, MAINMENU, zur Ausführung des Berichts Course Catalog geändert werden muß.

Beginnen Sie, indem Sie in Forms-Designer MAINMENU öffnen. Verwenden Sie Object Navigator, um das Element Course\_Catalog\_Report anzuzeigen, das im Menü REPORTS steht. Rufen Sie das Eigenschaftfenster für Course\_Catalog\_Report auf, wie es in Abbildung 15.26 gezeigt wird.

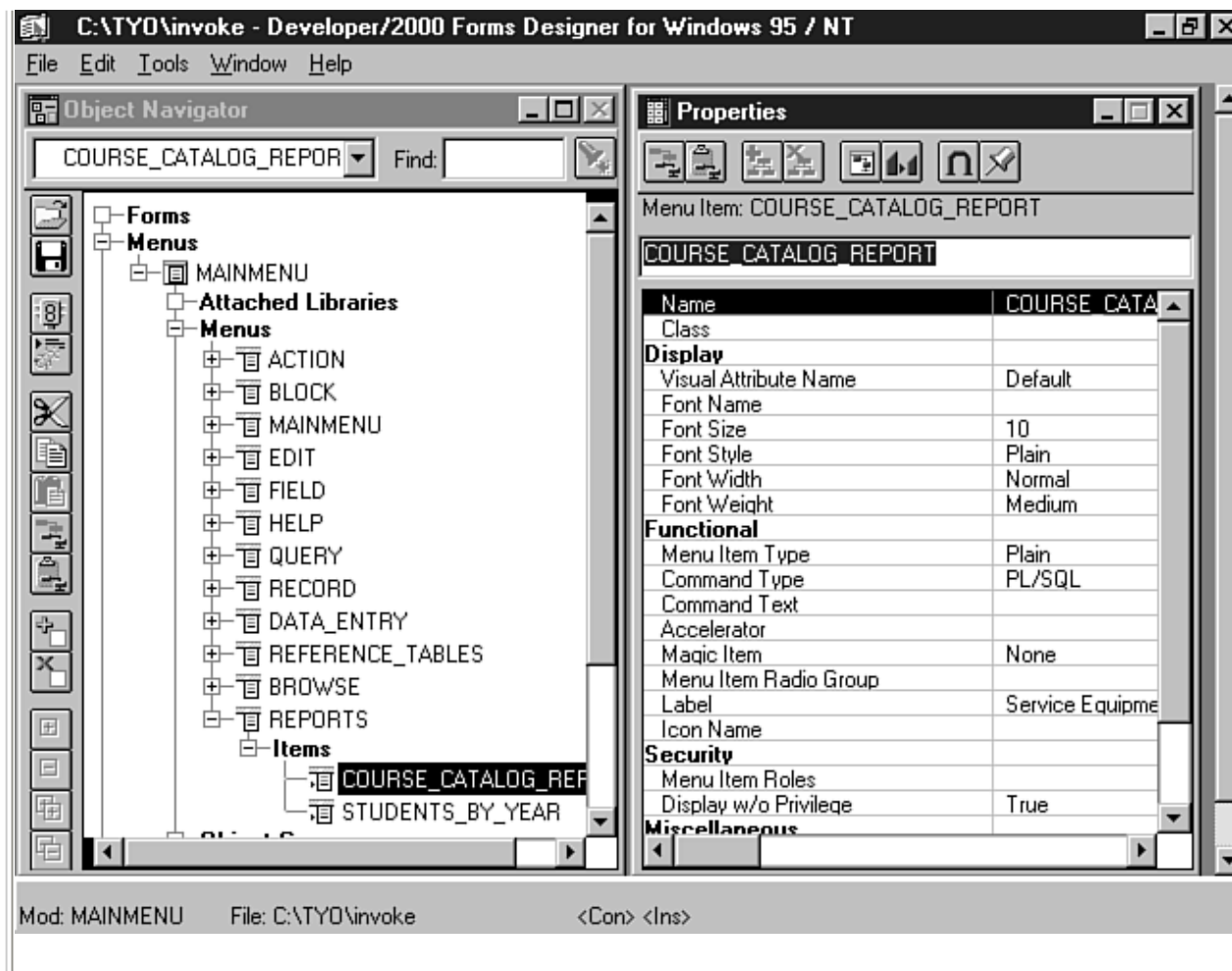


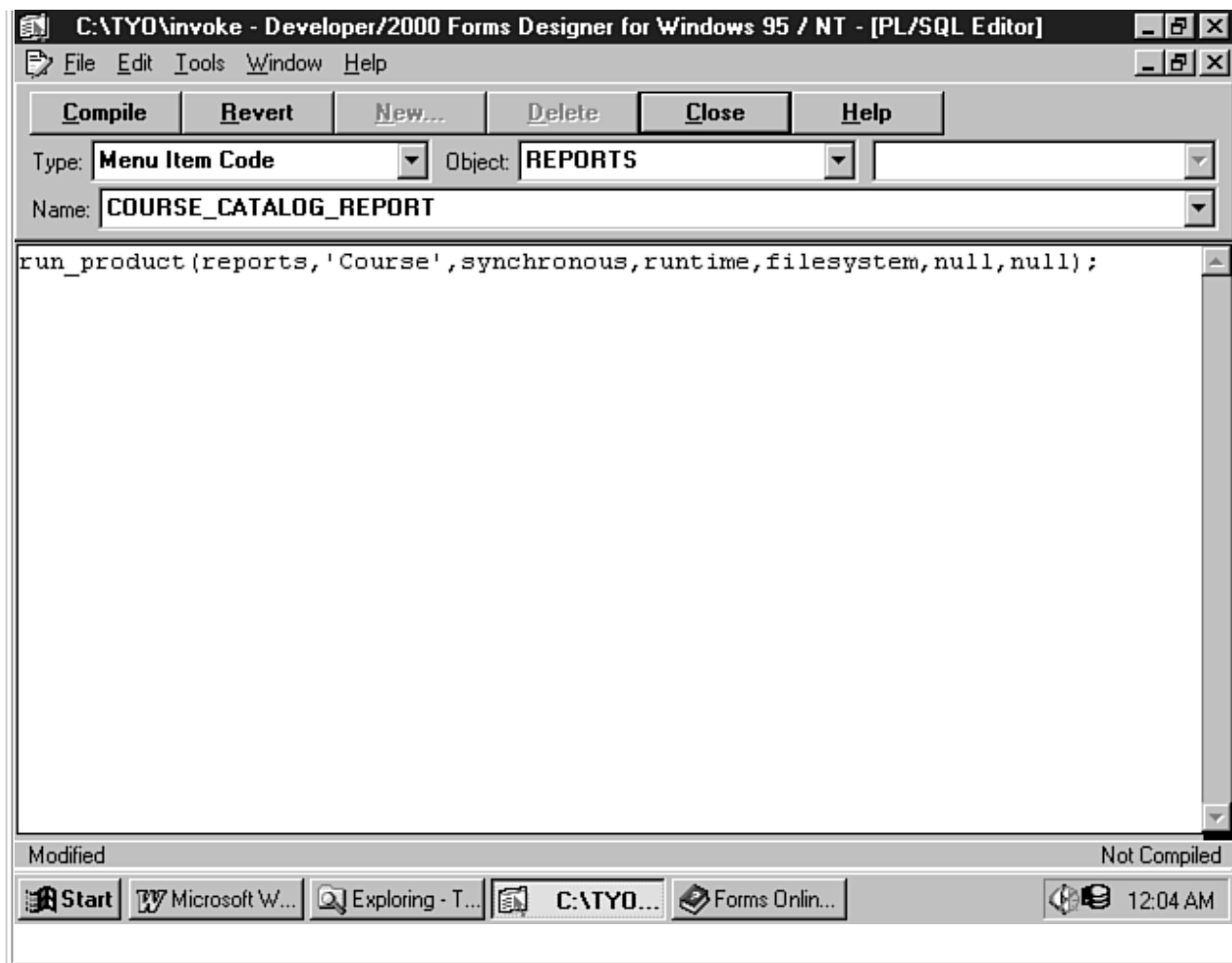
Abbildung 15.26:  
Eigenschaften des  
Menüelements  
Course\_Catalog\_Report.

Bewegen Sie sich jetzt zur Eigenschaft Text Command, und klicken Sie oben im Fenster auf More. Ein PL/SQL-Editor-Fenster wird angezeigt. Wie in Abbildung 15.31 gezeigt, verwenden Sie die integrierte Funktion Run\_Product, um Oracle-Berichte aufzurufen. Die Funktion Run\_Product benötigt sieben Argumente:

- Auszuführendes Produkt: Berichte
- Auszuführendes Objekt: Course
- Modus ist entweder synchron (was bedeutet, daß die Formular-Anwendung mit ihrer Fortführung wartet, bis das aufgerufene Produkt abgeschlossen ist) oder asynchron (was bedeutet, daß die Formular-Anwendung aktiv bleibt, während das aufgerufene Produkt seine Aufgaben ausführt): Synchron
- Ausführungsmodus, entweder Batch oder Laufzeit; Laufzeit
- Speicherort des Objekts, entweder Dateisystem oder Datenbank; Dateisystem
- Parameterliste, allenfalls 0, da Sie keine Parameter verwenden
- Name eines Grafikelements zur Anzeige eines Oracle-Diagramms; Null, da hier nicht anwendbar

Nachdem Sie die in Abbildung 15.27 angeführte Zeile eingegeben haben, klicken Sie auf Compile und Close. Speichern und erzeugen Sie das Menü, und testen Sie es auf seine korrekte Funktion.

Abbildung 15.27:  
Starten eines  
weiteren  
Developer/2000-  
Programms von  
einem Menüelement  
mit Hilfe von  
Run\_Product.



## Zusammenfassung

Nachfolgend werden noch einmal die Schwerpunkte dieser Lektion aufgeführt:

- Oracle Reports, als Teil von Developer/2000, besteht aus mehreren Programmen einschließlich Reports Designer und Reports Runtime.
- Sie können Reports Designer dazu einsetzen, die unterschiedlichsten Berichte, vom einfachen bis zum komplexen Bericht, zu entwickeln. Reports Designer kann Berichte als Binärdatei, RDF-Datei genannt, speichern.
- Reports Designer verwendet Object Navigator, um Berichtskomponenten in hierarchischer Form anzuzeigen.
- Auf der obersten Ebene wird ein Bericht durch ein Datenmodell, ein Layout, ein Parameterformular und Berichts-Trigger definiert. Ein Bericht enthält zumindest ein Datenmodell und ein Layout.
- Reports Designer enthält spezielle Editoren zur Definition der Berichtskomponenten. Diese Editoren umfassen den Data Model Editor und den Layout Editor.
- Ein Datenmodell besteht aus einer oder mehreren Abfragen, die die den Bericht ausmachenden Datensätze abrufen. Mit dem Data Model Editor können Sie Abfragen auf Basis gemeinsamer Spalten verknüpfen.
- Den Layout Editor verwenden Sie zur Gestaltung der äußeren Form eines Berichts. Das Layout hat vier Bestandteile: Header, Footer, Body und Margin. Üblicherweise enthält ein Bericht das Layout für den Body.

## Wie geht es weiter?

An Tag 16, »Developer/2000: Oracle Graphics und den Procedure Builder verwenden«, werden Sie zwei weitere Bestandteile von Developer/2000 kennenlernen: Oracle Graphics und den Procedure Builder. Sie werden lernen, aus dem Inhalt einer Tabelle ein Diagramm zu erstellen. Außerdem werden Sie erfahren, wie Sie Procedure Builder bei der Erstellung und Verwaltung von PL/SQL-Modulen in einer Oracle-Datenbank unterstützt.

## Fragen und Antworten

Frage:

Ist es möglich, die Ausführung eines Berichts aufzuzeichnen?

Antwort:

*Ja. Auch wenn es in dieser Lektion nicht behandelt wurde, ist es möglich, einen Berichts-Trigger zu definieren, der bei der Ausführung eines Berichts ausgelöst wird. Dieser Berichts-Trigger könnte einer Tabelle eine Zeile hinzufügen, in der der Name des Berichts, wer den Bericht angefordert hat und wann der Bericht angefordert wurde, festgehalten wird. Es können noch weitere Informationen über den Bericht gespeichert werden.*

Frage:

Kann man einen Benutzer am Ausdruck eines Berichts nach Geschäftsschluß hindern?

Antwort:

*Ja. Sie können auch einen Berichts-Trigger erstellen, der den Wochentag oder die Uhrzeit beachtet und feststellt, ob der Bericht zu dieser Zeit gedruckt werden soll.*

Frage:

Ist es angebracht, daß derselbe Bericht von mehr als einem Benutzer zur gleichen Zeit ausgeführt wird?

Antwort:

*Ja. Mehrere Benutzer können den Bericht ohne nachteilige Auswirkungen zeitgleich ausführen, es sei denn, der Trigger ändert die Daten auf ungewöhnliche Weise.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Richtig oder falsch? Sie können eine RDF-Datei in einem Code-Steuerungssystem wie PVCS speichern.
2. Nennen Sie die vier Elemente eines Berichtslayouts.
3. Richtig oder falsch? Ein Bericht kann nicht mehr als Abfragen miteinander verknüpfen.

## Übungen

Erstellen Sie einen Master-/Detail-Bericht, der jeden Klassenraum und die für diesen Raum geplanten Vorlesungen auflistet, ohne Berücksichtigung der Unterrichtszeiten (z.B. alle Klassen, die in einem bestimmten Raum stattfinden sollen).





# 16

# Developer/2000: Oracle Graphics und den Procedure Builder verwenden

[Die Rolle von Oracle Graphics](#)

[Die Elemente von Oracle Graphics](#)

[Erstellen eines Diagramms mit dem Graphics-Designer](#)

[Speichern des Diagramms](#)

[Die Rolle des Procedure Builder im Prozeß der Anwendungsentwicklung](#)

[Elemente des Procedure Builder](#)

[Den Object Navigator des Procedure Builder verwenden](#)

[Einen gespeicherten Programmblock betrachten und bearbeiten](#)

[Erstellen eines Datenbank-Trigger mit dem Editor für Datenbank-Trigger](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

In den vier vorangegangenen Lektionen haben Sie mit Oracle Forms and Reports gearbeitet, um Teile einer Anwendung zu entwickeln. In dieser Lektion werden Sie mit zwei weiteren Komponenten arbeiten: Developer/2000: Oracle Graphics und dem Procedure Builder.

## Die Rolle von Oracle Graphics

Oracle Graphics stellt eine Methode zur Erzeugung von Diagrammen zur Verfügung, die ihrer Werte aus einer oder mehreren Tabellen einer Oracle-Datenbank ableiten. Sie kennen das alte Sprichwort: »Ein Bild sagt mehr als tausend Worte.« Das Ziel von Oracle Graphics ist es, die

grafische Darstellung des Inhalts einer Tabelle aufzubereiten, damit Trends schnell erkannt, große Datenmengen addiert und Vergleiche zwischen Datensätzen gezogen werden können.

## Die Elemente von Oracle Graphics

Oracle Graphics verwendet das gleiche Muster wie Forms and Reports. Zum Entwerfen und Erzeugen einer Laufzeitversion eines Diagramms wird ein Werkzeug namens Graphics-Designer benutzt. Ein weiteres Programm namens Graphics Runtime führt das erzeugte Diagramm aus. In dieser Lektion sehen Sie, daß dem Menü des Formulars ein Menüpunkt hinzugefügt werden kann, um das Diagramm auszuführen.

Wie Forms and Reports verwendet Oracle Graphics einen Object Navigator zur Erzeugung, Änderung und Ansicht der Elemente eines Diagramms. Oracle Graph-ics verwendet den Begriff *Grafische Anwendung* um sich auf eine Grafikobjekt zu beziehen. Die Objektknoten einer grafischen Anwendung sind folgende:

- Layout
- Schablone
- Abfrage
- Parameter
- Klang
- Zeitgeber
- Programmblock
- Zugeordnete Bibliothek

Ein Diagramm enthält mindestens eine Abfrage und ein Layout.

## Erstellen eines Diagramms mit dem Graphics-Designer

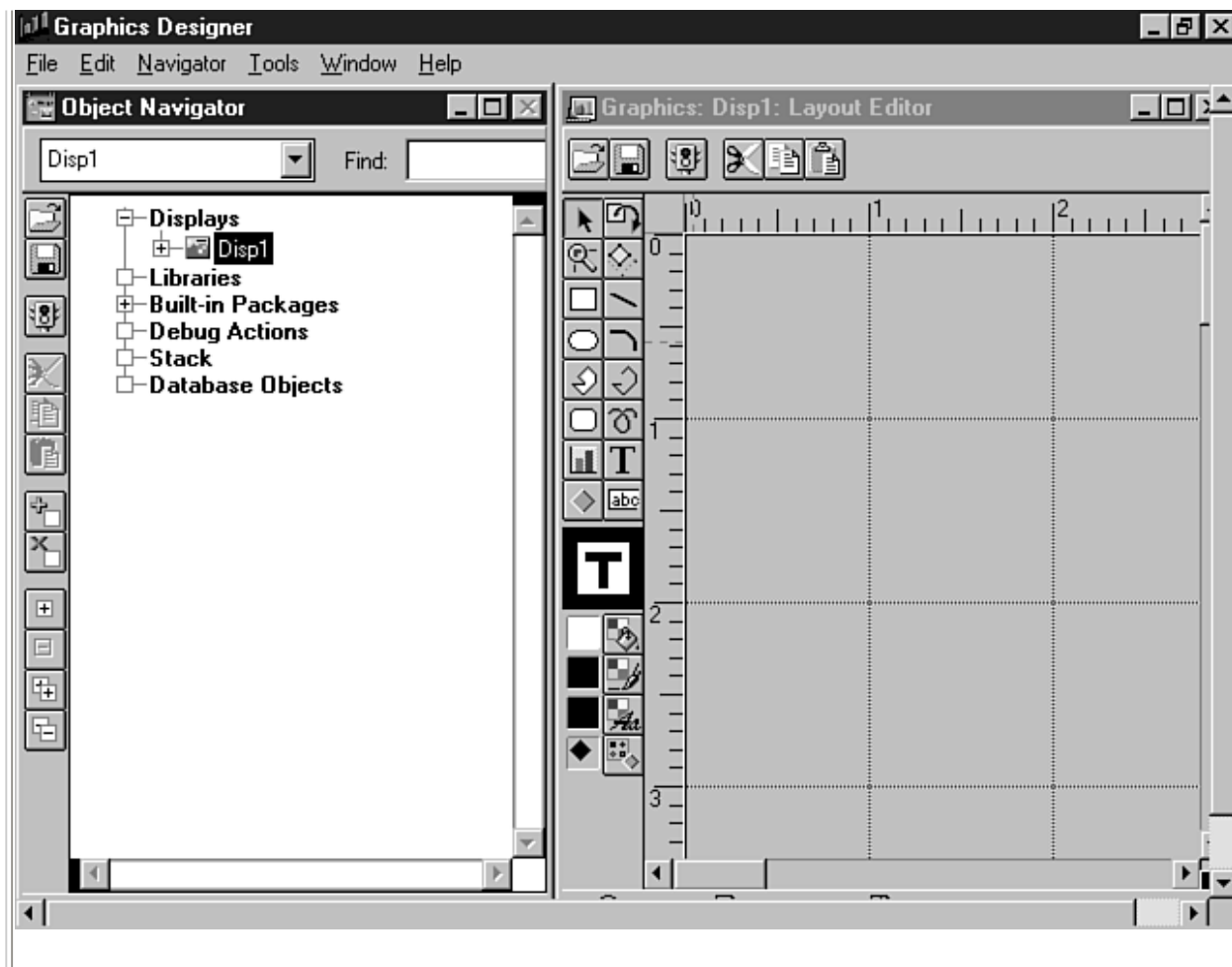
Zu Beginn wollen wir ein Tortendiagramm auf Basis einer einzigen Tabelle entwerfen. Sie werden ein Diagramm erstellen, das Angaben zu den Studenten eines Jahres darstellt. Zum Aufruf von Graphics-Designer wählen Sie Oracle Designer for Win95 | Graphics Designer aus dem Start-Menü. Graphics-Designer wird als erstes den Object Navigator anzeigen (siehe Abbildung 16.1).

Beachten Sie, daß das vom Object Navigator angezeigte Diagramm den Titel UNTITLED hat. Ehe Sie fortfahren können, müssen Sie eine Verbindung zu einer Oracle-Datenbank herstellen.

### Aufbau einer Verbindung zu einer Oracle-Datenbank

Um Graphics-Designer mit einer Oracle-Datenbank zu verbinden, wählen Sie den Menüeintrag File | Connect. Im dann erscheinenden Dialogfenster werden Sie zur Eingabe des Benutzernamens, Kennworts und der Datenbank aufgefordert, um eine Verbindung zu einer Oracle-Datenbank aufzubauen. Stellen Sie die Verbindung zu einer Oracle-Datenbank her; um Einzelheiten dazu zu sehen, schlagen Sie in Tag 12, »Developer/2000: Einführung in Oracle Forms«, nach.

*Abbildung 16.1:  
Der Graphics-  
Designer zeigt  
zunächst den Object-  
Navigator an.*



## Erstellen eines neuen Diagramms

Mit den folgenden Schritten erstellen Sie ein neues Diagramm:

1. Wählen Sie den Menüeintrag Windows | Graphics: Disp1: Layout Editor. Beachten Sie, daß sich das Menü verändert.
2. Wählen Sie den Menüeintrag Chart | Create - es erscheint das Chart Genie.
3. Ändern Sie im Namensfeld den Namen der Abfrage von query0 in Student\_Query.
4. Sie möchten ein Tortendiagramm erstellen, in dem die relative Prozentzahl von Studenten eines bestimmten Jahres dargestellt wird. Dazu brauchen Sie in der SELECT-Anweisung GROUP-BY-Klausel (siehe Abbildung 16.2). Im Listenfeld SQL-Statement geben Sie ein:

```
select Year, Count(*)
from Student
group by Year;
```

## Wählen des Diagrammtyps

Als nächstes zeigt Chart Genie ein Fenster an, in dem Sie die Chart Properties festlegen.

1. Geben Sie im Feld Name Student\_by\_Year ein.
2. Geben Sie im Feld Title Students by Year ein.
3. In der Auswahl Type wählen Sie Tortendiagramm. Eine weitere Gruppierung namens Subtype erscheint, in der Sie 3-D-Tortendiagramm auswählen.
4. Klicken Sie auf OK.

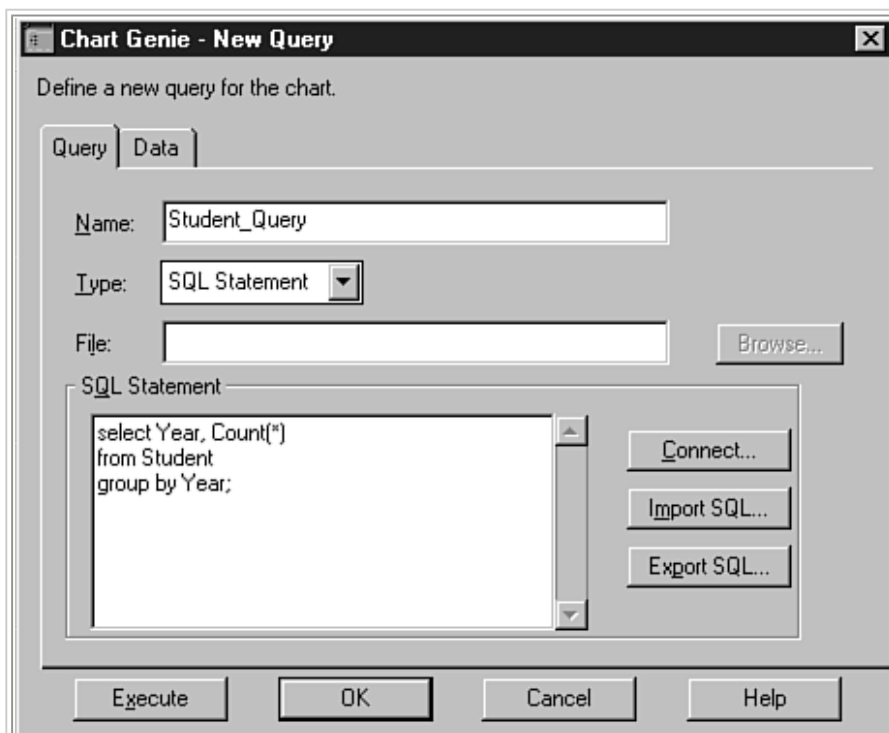


Abbildung 16.2:  
Festlegen der Abfrage im  
Chart Genie.

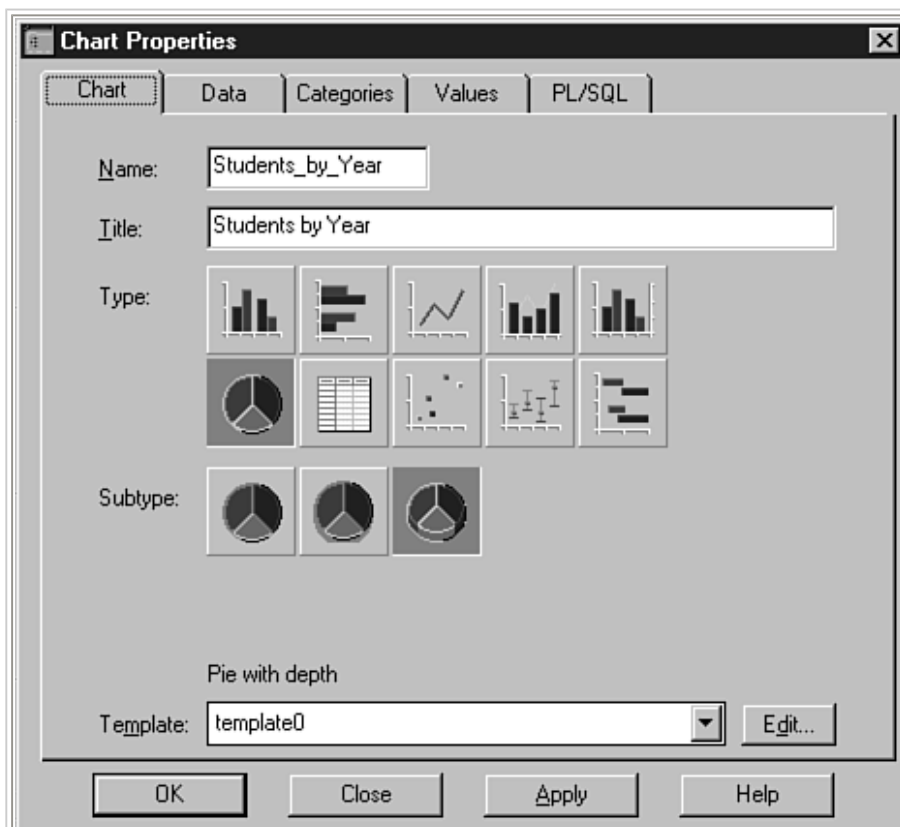


Abbildung 16.3:  
Festlegen der  
Diagrammeigenschaften  
im Chart Genie.

Das Layout des 3-D-Tortendiagramms erscheint entsprechend Ihren Angaben (siehe Abbildung 16.4). Machen Sie sich aber klar, daß dies nur das anfängliche Layout ist. Schauen wir uns einige Veränderungen an, die Sie hinzufügen möchten.

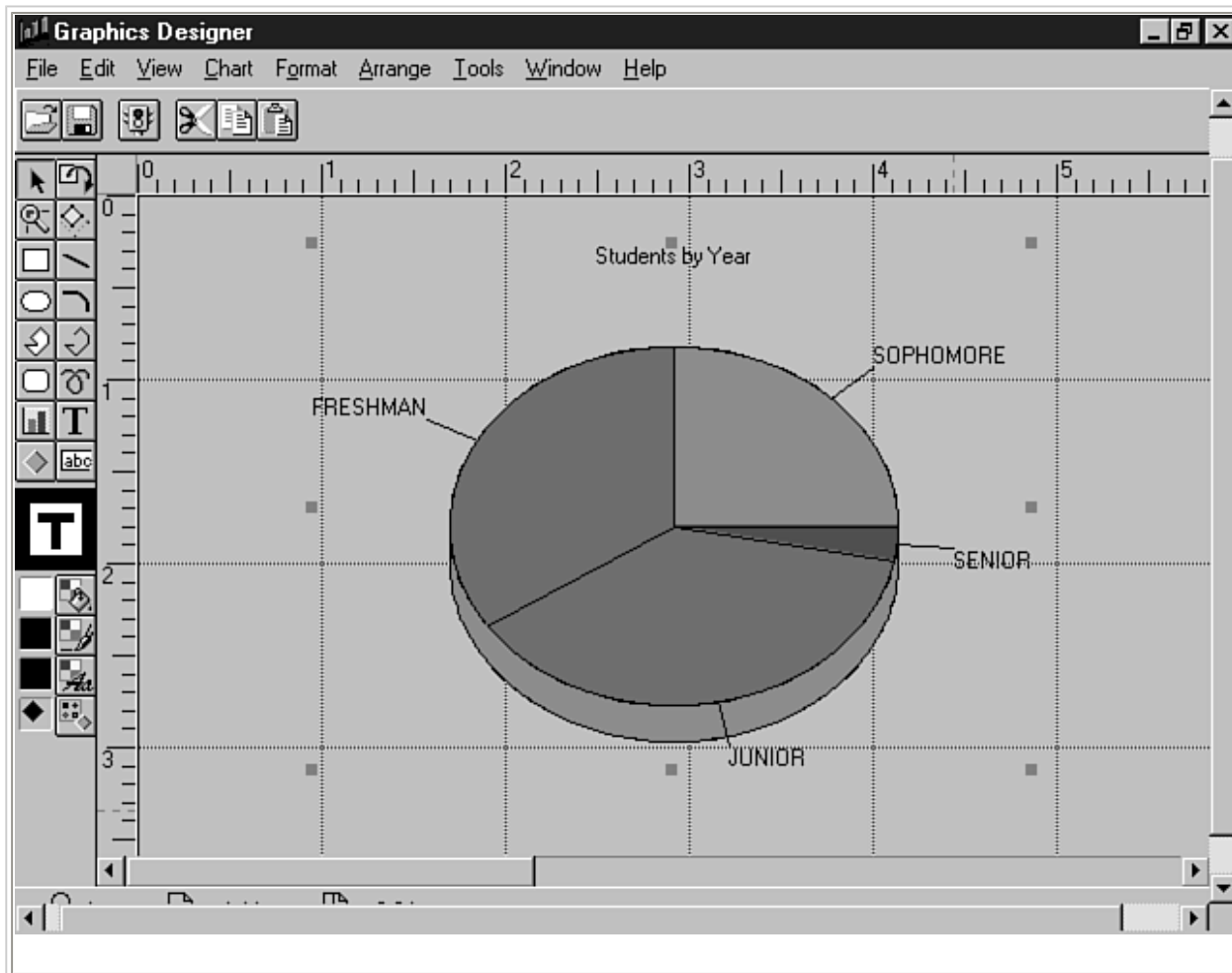


Abbildung 16.4:  
Layout des  
Tortendiagramms

## Anpassen des Diagramm-Layouts

Wie Sie in Abbildung 16.5 sehen können, sind im Tortendiagramm noch keine Prozentzahlen für die Datensegmente zu sehen. Mit den folgenden Schritten fügen Sie diese hinzu:

1. Wählen Sie mit der rechten Maustaste den Mittelpunkt des Diagramms an und dann den Menüeintrag **Frame**. es erscheint ein Fenster namens **Frame Properties**.
2. Wählen Sie das Register **Pie Frame**.
3. Markieren Sie das Kontrollkästchen mit der Bezeichnung **Show Percent Values** (siehe Abbildung 16.5).
4. Klicken Sie auf **OK**.

Jetzt müssten Sie die Prozentwerte im Layout sehen. Wir wollen die Größe der Überschrift heraufsetzen.

1. Wählen Sie die Überschrift mit der Maus an.
2. Wählen Sie den Menüeintrag **Format | Font**.
3. Stellen Sie **MS Sans Serif, Fett, 12 Punkt** für die Überschrift ein (siehe Abbildung 16.6).

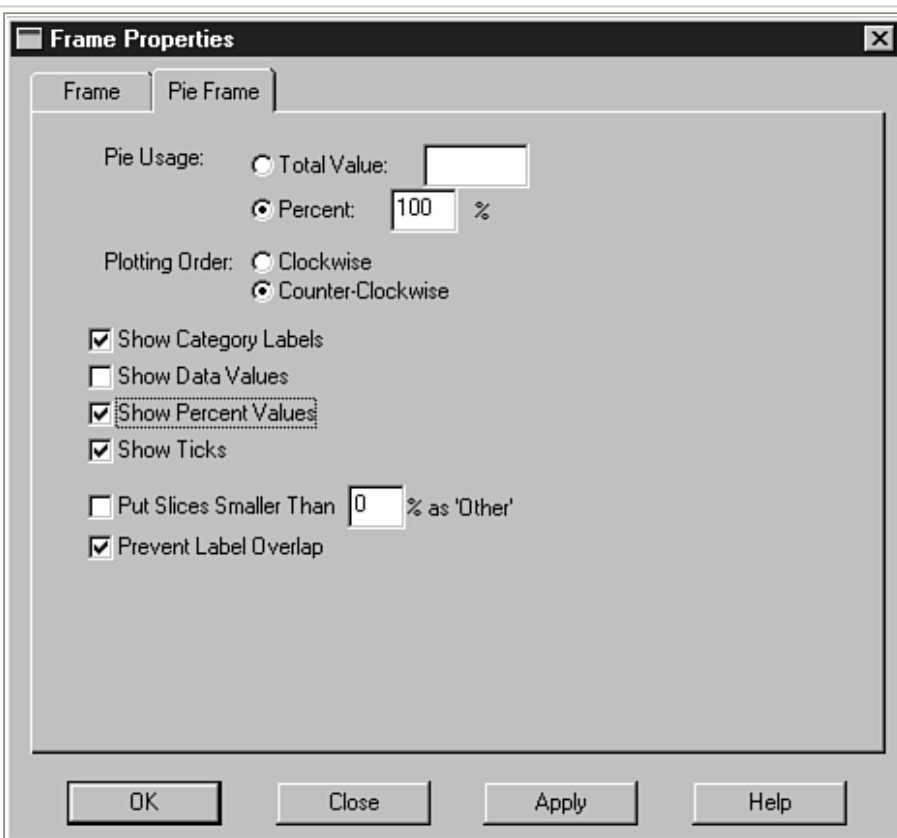


Abbildung 16.5:  
Änderung der  
Rahmeneigenschaften, so  
daß Prozentwerte  
angezeigt werden.

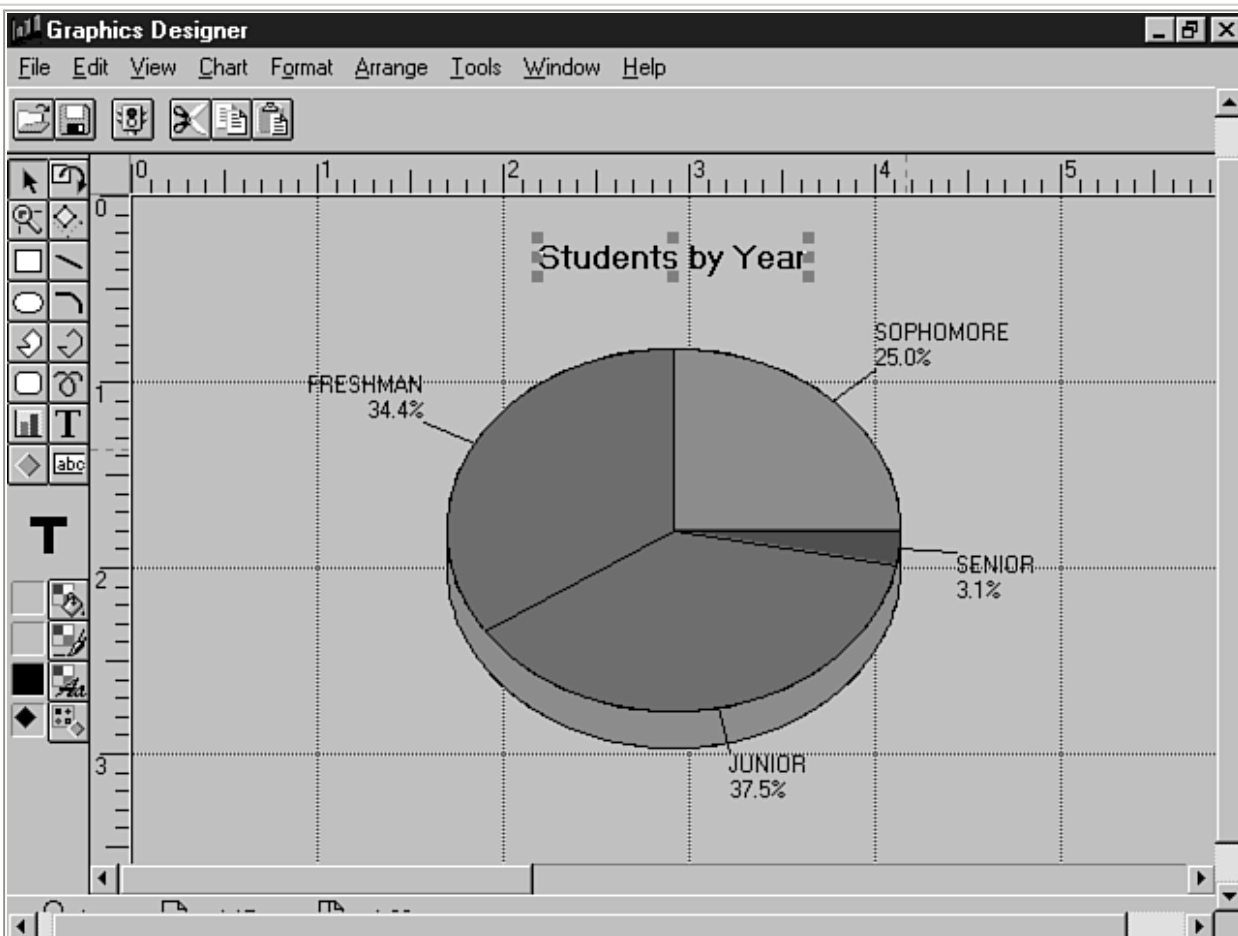


Abbildung 16.6:  
Formatierung des  
Diagrammtitels.

## Speichern des Diagramms

Um dieses Diagramm zu speichern, wählen Sie den Menüeintrag File | Save As | File System. Geben Sie im darauf erscheinenden Dialogfeld

den Namen der Datei ein: Students\_by\_Year. Graphics-Designer speichert eine Datei mit der Endung .ogd, die Abkürzung für Oracle-Graphics-Designer. Um jedoch ein Diagramm mit Graphics Runtime ausführen zu lassen, müssen Sie eine .ogr-Datei aus der entsprechenden .ogd-Datei erzeugen (.ogr ist die Abkürzung für Oracle Graphics Runtime). Dazu wählen Sie im Menü File | Administration | Generate | File System (alternativ können Sie auch die Tasten (Alt) + (t) drücken). Standardmäßig wird im Dialogfeld SaveAs der .ogr-Datei derselbe Dateiname wie der .ogd-Datei gegeben. Sie können Graphics-Designer verlassen - im nächsten Schritt muß für die Ausführung des Diagramms mit Graphics Runtime eine Verknüpfung erstellt werden.

---



*Achten Sie darauf, Ihre Diagrammdateien in einem Verzeichnis zu speichern, das nicht zum Oracle-Verzeichnisbaum gehört (unter C:\Orawin95). Denken Sie außerdem daran, das Verzeichnis mit dem Registry-Editor dem Pfad GRAPHICS25\_PATH hinzuzufügen, sonst kann Graphics Runtime Ihre Diagrammdateien nicht öffnen.*

---

### Erzeugen eines Shortcut zum Aufrufen eines Diagramms

Anwender freuen sich immer, wenn Sie es ihnen so leicht wie möglich machen, die gewünschten Ergebnisse zu erhalten. Eine Windows-Verknüpfung stellt eine solche Möglichkeit dar. Vielleicht ist sie keine so nahtlose Lösung wie eine integrierte Developer/2000-Anwendung, aber sie ist ein »billiger« Weg, dem Anwender zu helfen.

Ehe Sie diese Verknüpfung erstellen, sollten Sie in Graphics Runtime eine Voreinstellung vornehmen, damit beim Start eine Anmeldung in der Datenbank versucht wird. Dadurch wird Graphics Runtime Sie zur Eingabe des Benutzernamens, Kennworts und der Datenbank auffordern, wenn Sie diese nicht in der Befehlszeile angegeben haben. Dazu führen Sie folgende Schritte aus:

1. Rufen Sie Graphics Runtime auf, indem Sie den Menüeintrag Start | Developer/2000 for Win95 | Graphics Runtime wählen.
2. Wählen Sie den Menüeintrag Edit | Tools Options.
3. Aktivieren Sie Kontrollkästchen mit der Bezeichnung Logon at Startup.
4. Klicken Sie Save auf OK.
5. Verlassen Sie das Programm.
6. Zur Erstellung der Verknüpfung gehen Sie wie folgt vor:
7. Klicken Sie auf dem Windows-Desktop mit der rechten Maustaste auf Neu | Verknüpfung. Das Fenster Verknüpfung erstellen erscheint.
8. Damit Students by Year zur Laufzeit ausgeführt wird, geben Sie die folgende Zeile in das Feld Befehlszeile ein.  
  
`c:\Orawin95\bin\g25run32 openfile=c:\tyo\Students_by_Year.ogr`
9. Klicken Sie auf die Schaltfläche Weiter.
10. Geben Sie der Verknüpfung einen den Namen Diagramm - Students by Year.
11. Klicken Sie auf die Schaltfläche Beenden.

Die Verknüpfung sollte auf dem Desktop erscheinen. Doppelklicken Sie darauf. Sie werden zur Eingabe des Benutzernamens, Kennworts und der Datenbank aufgefordert. Wenn Sie wollen, daß der Anwender nicht zur Eingabe dieser Werte aufgefordert wird, ändern Sie die Befehlszeile folgendermaßen ab:

```
c:\Orawin95\bin\g25run32 openfile=c:\tyo\Students_by_Year.ogr  
userid=flugle/flugle
```

---





*Ein Nachteil dieser Verknüpfung ist, daß sie das Benutzerkennwort enthält und es von dritten eingesehen werden kann.*

---

## Die Rolle des Procedure Builder im Prozeß der Anwendungsentwicklung

Die letzte noch durchzusprechende Komponente von Developer/2000 ist der Procedure Builder, der zur Vereinfachung der eigentlichen Entwicklung und Verwaltung von PL/SQL-Quelltext (sowohl auf dem Client als auch auf dem Server) entworfen wurde.



*PB unterstützt nicht die Verwendung von Funktionen aus der Version 2 von PL/SQL, wie PL/SQL-Tabellen in PL/SQL-Modulen auf Client-Basis, die von Oracle Forms, Reports oder Graphics benutzt werden.*

---

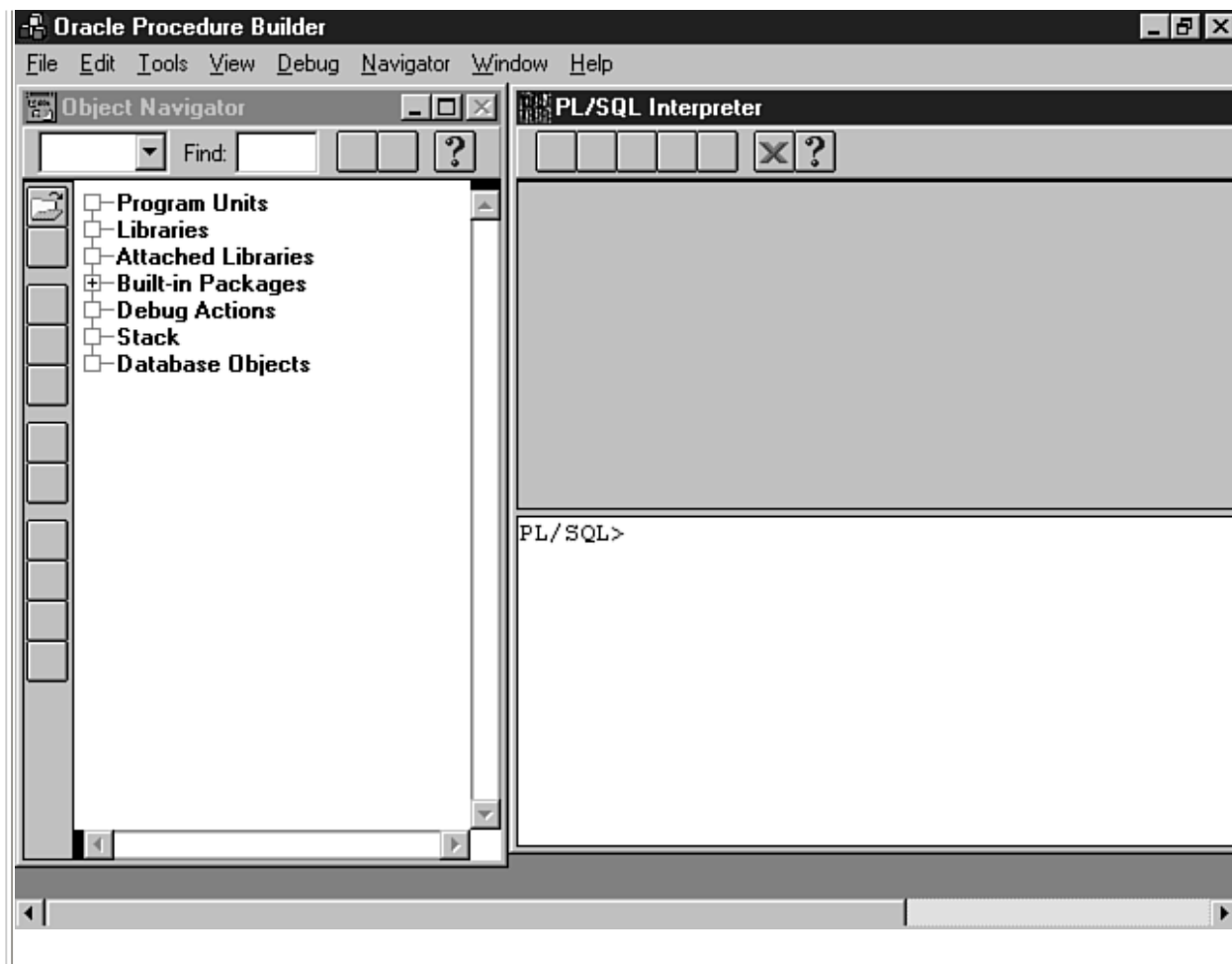
## Elemente des Procedure Builder

Procedure Builder enthält verschiedene Werkzeuge:

- Einen Program Unit Editor zum Erstellen, Ändern und Löschen von Programmblöcken, wie Prozeduren und Funktionen, die in Oracle Forms zugeordneten Bibliotheken benutzt werden.
- Einen PL/SQL-Interpreter, der Sie einen PL/SQL-Quelltextblock schreiben und testen läßt.
- Einen Stored Program Unit Editor, der die Erstellung und Verwaltung eines in einer Oracle-Datenbank gespeicherten Pakets, einer Prozedur oder Funktion unterstützt, mit der Procedure Builder verbunden ist.
- Einen Database Trigger Editor zur Erstellung und Verwaltung von Datenbank-Triggern.

Um Procedure Builder aufzurufen, wählen Sie Start | Programme | Developer/2000 for Win95 | Procedure Builder. Abbildung 16.7 zeigt das Anfangsfenster des Procedure Builder. Beachten Sie, daß Procedure Builder, wie auch die anderen Komponenten von Developer/2000, einen Object Navigator zur Verfügung stellt, um PL/SQL-Objekte auszuwählen und deren Eigenschaften zu verändern.

*Abbildung 16.7:  
Vom Procedure  
Builder angezeigtes  
Anfangsfenster*



*Wie auch die anderen Komponenten von Developer/2000 verwendet Procedure Builder den Object Navigator. Zu Einzelheiten des Einsatzes des Object Navigator schlagen Sie bei Tag 12 nach.*

Bevor Sie Werkzeuge des Procedure Builder einsetzen, sollten Sie zu einer Datenbank eine Verbindung aufbauen.

### Aufbau einer Verbindung zu einer Oracle-Datenbank

Um Procedure Builder mit einer Oracle-Datenbank zu verbinden, benutzen Sie die an Tag 12 beschriebene Methode.

## Den Object Navigator des Procedure Builder verwenden

Der Object Navigator des Procedure Builder arbeitet auf die gleiche Weise wie die anderen Komponenten von Developer/2000 und zeigt sieben Objektknoten an:

- Programmblöcke
- Bibliotheken
- Zugeordnete Bibliotheken
- Integrierte Pakete
- Debug-Aktionen
- Stacks
- Datenbankobjekte

In dieser Lektion konzentrieren Sie sich auf Programmblöcke und Datenbankobjekte.

## Der Programmblock-Editor

Den Program-Unit-Editor verwenden Sie, um Pakete, Prozeduren und Funktionen einzusehen, zu erstellen, zu ändern und zu löschen. Aufrufen können Sie den Program-Unit-Editor, indem Sie den Menüeintrag Tools | Stored Program Unit Editor wählen. Um z.B. eine neue Funktion zu erstellen, klicken Sie auf New; es erscheint ein Fenster, in dem Sie zur Eingabe des Namens und Typs des Programmblocks aufgefordert werden (siehe Abbildung 16.8). Nachdem Sie auf OK geklickt haben, erstellt Program-Unit-Editor eine Schablone für die Funktion, wie es in Abbildung 16.9 zu sehen ist.

## Der PL/SQL-Interpreter

Den PL/SQL-Interpreter rufen Sie auf, indem Sie den Menüeintrag Tools | PL/SQL-Interpreter wählen. Der PL/SQL-Interpreter arbeitet in zwei Zuständen:

- Nicht-modal, wenn Sie den PL/SQL-Interpreter direkt aufgerufen haben. In diesem Zustand machen Sie Ihre Eingabe über die Befehlszeile. Die Eingabe könnte aus Zeilen eines anonymen PL/SQL-Blocks bestehen. Abbildung 16.10 zeigt ein manuell eingegebenes und von PL/SQL-Interpreter ausgeführtes Beispiel eines anonymen PL/SQL-Blocks.
- Modal, wenn der PL/SQL-Interpreter durch von Ihnen in PL/SQL-Modulen gesetzte Programmstopps aufgerufen wird.

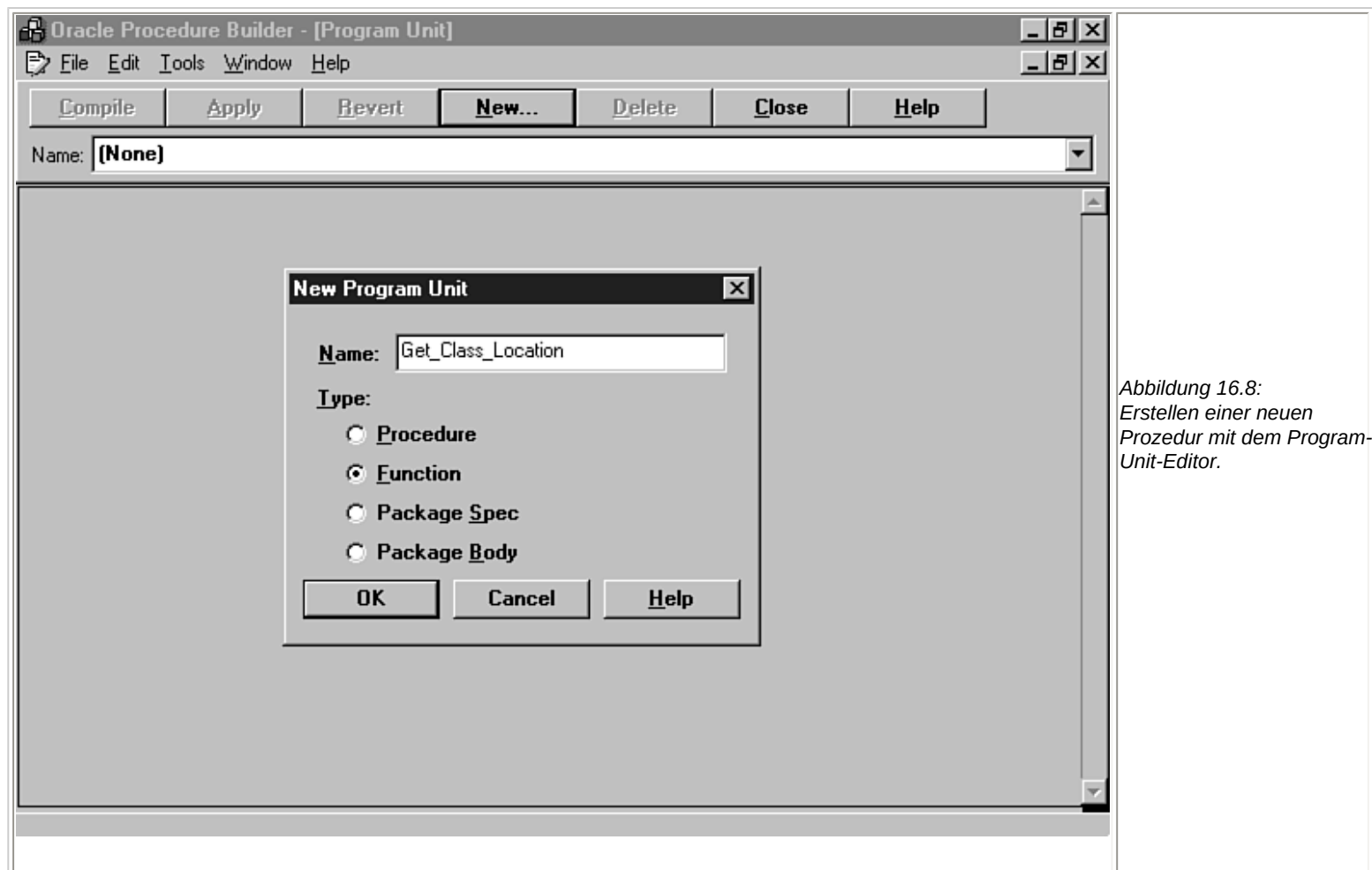


Abbildung 16.8:  
Erstellen einer neuen  
Prozedur mit dem Program-  
Unit-Editor.

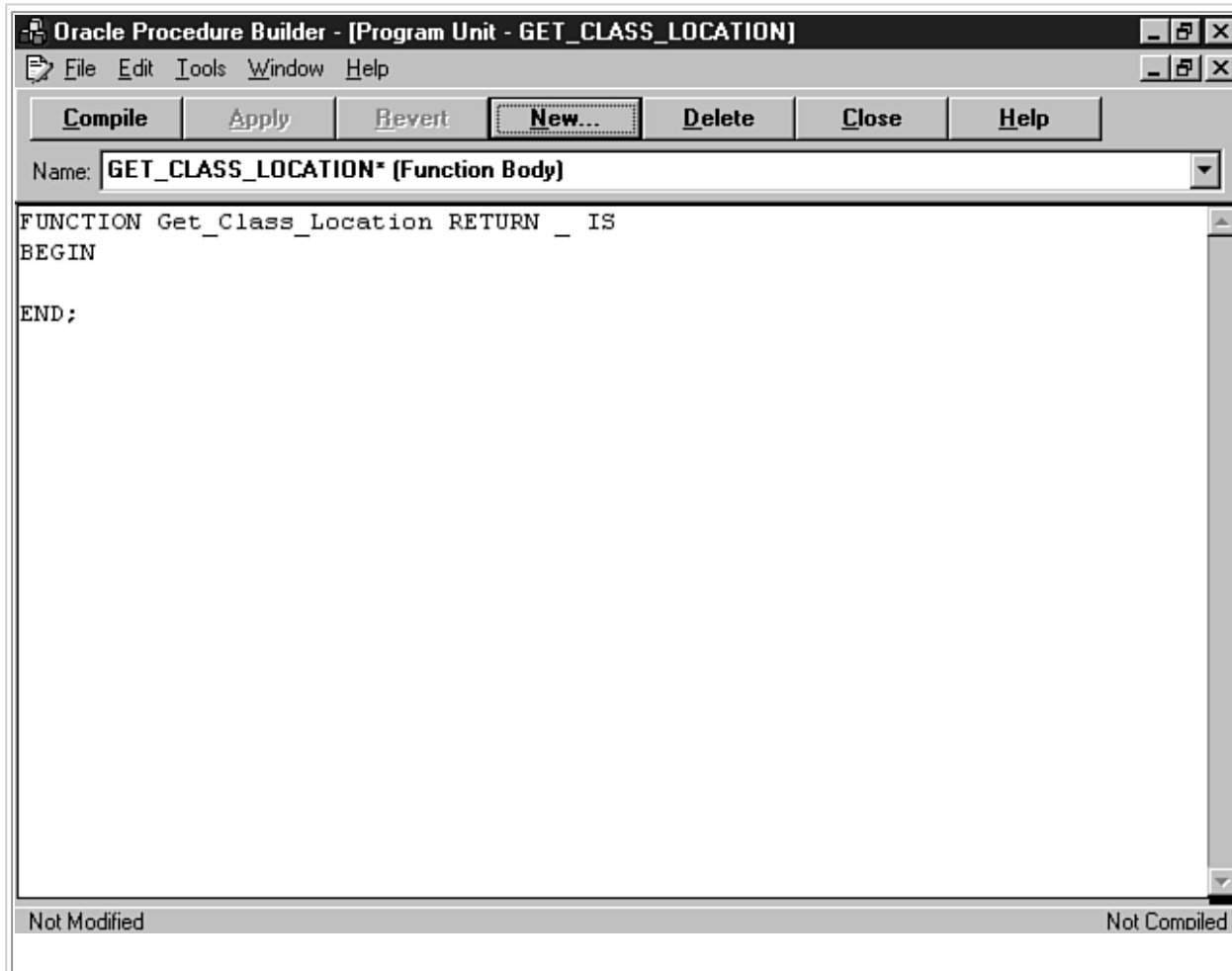


Abbildung 16.9:  
Mit dem Program-Unit-  
Editor erstellte Vorlage für  
eine Funktion.

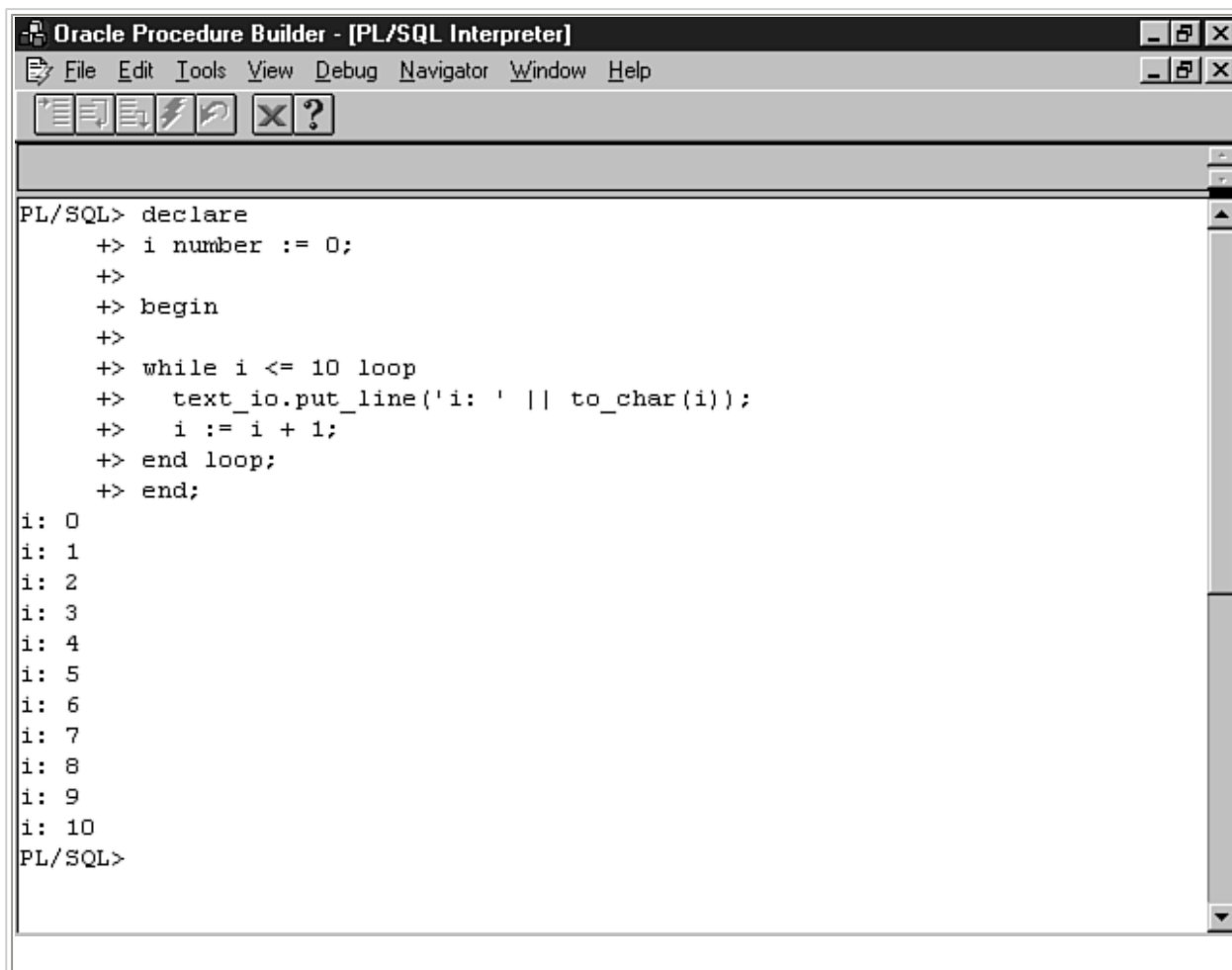


Abbildung 16.10:  
Eingabe und  
Ausführen eines  
anonymen Blocks  
mit Hilfe des PL/SQL-  
Interpreter.

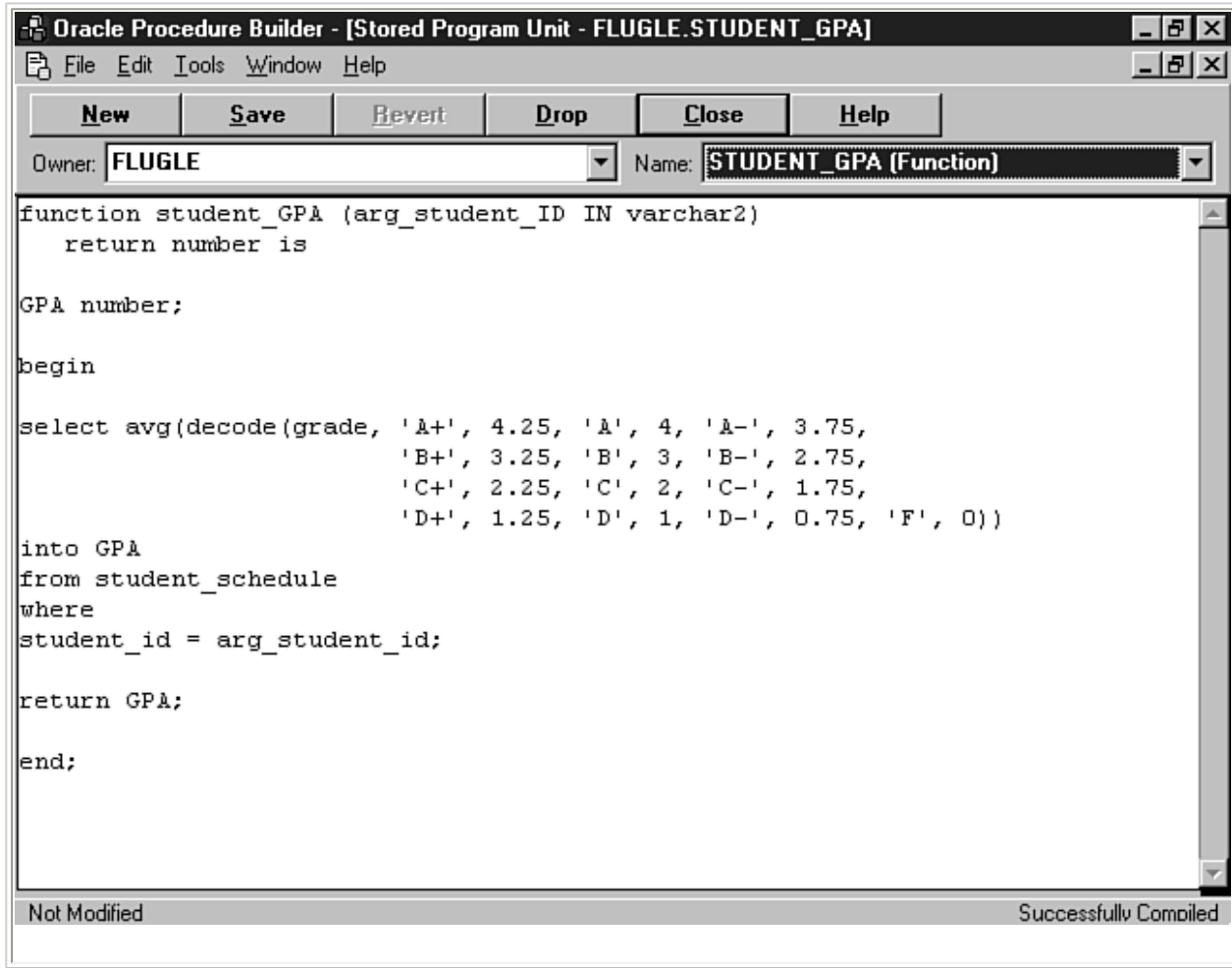
Der Editor für gespeicherte Programmblöcke

Den Stored-Program-Unit-Editor verwenden Sie, um in einer Oracle-Datenbank gespeicherte Pakete, Prozeduren und Funktionen einzusehen, zu erstellen, zu ändern und zu löschen. Um den Stored-Program-Unit-Editor benutzen zu können, müssen Sie bereits mit einer Oracle-Datenbank verbunden sein. Sie rufen den Stored-Program-Unit-Editor auf, indem Sie den Menüeintrag Tools | Stored Program Unit Editor wählen.

Wenn das Fenster des Editors erscheint, bemerken Sie oben im Fenster die beiden Felder: Owner und Name. Das Textfeld Owner enthält eine Drop-down-Liste der in der mit Procedure Builder verbundenen Oracle-Datenbank bestehenden Benutzer. Das Textfeld Name enthält eine Drop-down-Liste der den in Owner angezeigten Oracle-Benutzern gehörenden, gespeicherten Programmblöcke. Abbildung 16.11 illustriert, wie Sie einen bestimmten gespeicherten Programmblock im Textfeld Name auswählen, nachdem Sie den Oracle-Benutzer, flugle, angegeben haben.



Nicht jeder Oracle-Benutzer besitzt Pakete, Prozeduren oder Funktionen; einige Benutzer werden nichts davon besitzen. In einem solchen Fall wird Procedure Builder (None) im Feld Name anzeigen. Außerdem hängen Ihre Möglichkeiten, die Ihrem eigenen Account oder dem anderer Benutzer gehörenden Stored Program Units zu erstellen oder zu ändern, von den System- und Objektrechten Ihres Oracle Account ab.



Oracle Procedure Builder - [Stored Program Unit - FLUGLE.STUDENT\_GPA]

File Edit Tools Window Help

New Save Revert Drop Close Help

Owner: FLUGLE Name: STUDENT\_GPA (Function)

```
function student_GPA (arg_student_ID IN varchar2)
  return number is
GPA number;
begin
select avg(decode(grade, 'A+', 4.25, 'A', 4, 'A-', 3.75,
                    'B+', 3.25, 'B', 3, 'B-', 2.75,
                    'C+', 2.25, 'C', 2, 'C-', 1.75,
                    'D+', 1.25, 'D', 1, 'D-', 0.75, 'F', 0))
into GPA
from student_schedule
where
student_id = arg_student_id;
return GPA;
end;
```

Not Modified Successfully Compiled

Abbildung 16.11:  
Betrachtung einer  
gespeicherten  
Funktion mit Hilfe  
des Stored Program-  
Unit-Editors.

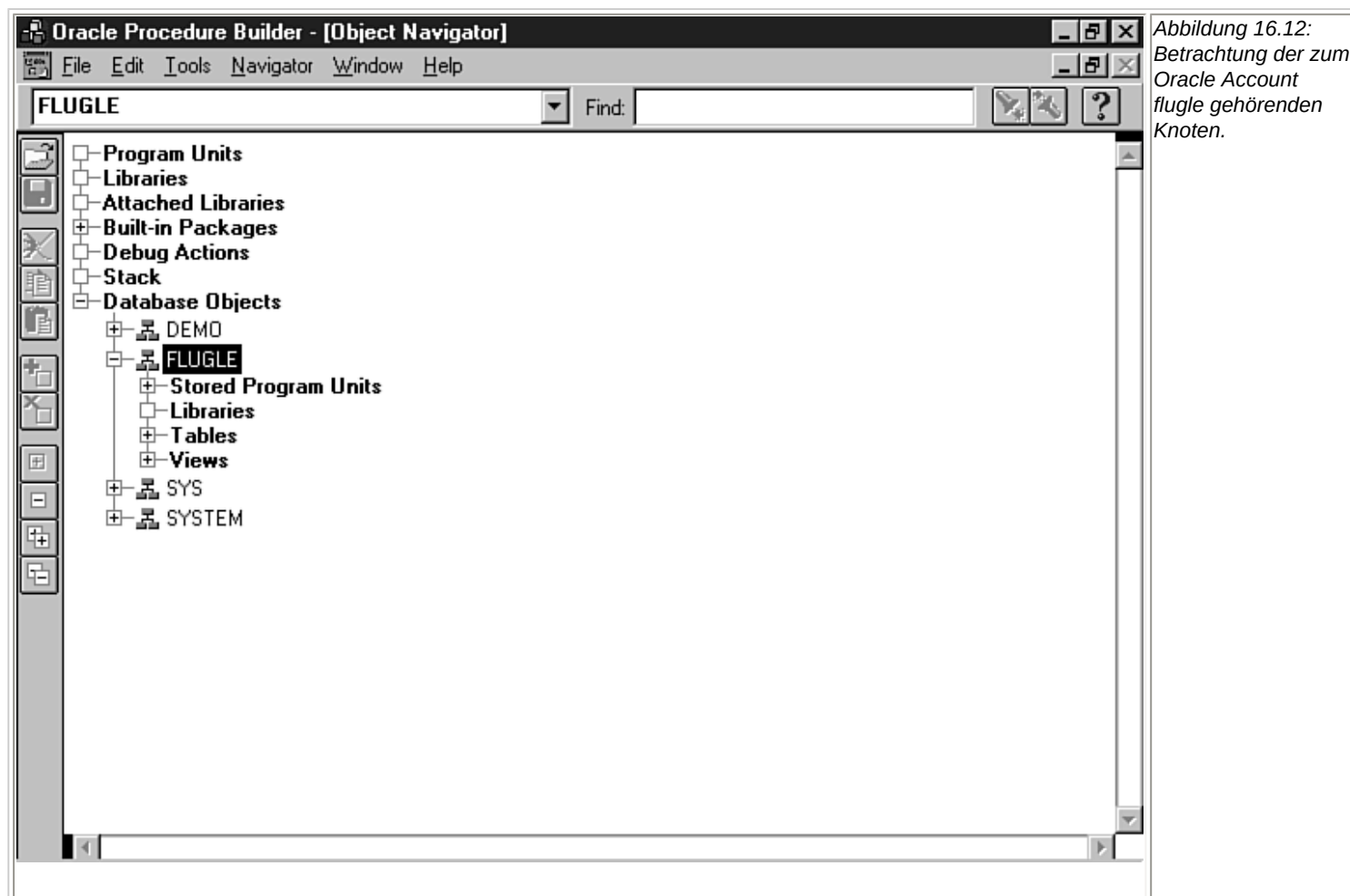
Der Editor für Datenbank-Trigger

Den Database-Trigger-Editor können Sie auf zwei Wegen aufrufen: Sie können mit dem Object Navigator einen vorhandenen Datenbank-Trigger anzeigen oder ändern, oder den Menüeintrag Tools | Database Trigger Editor wählen.

Um den Object Navigator zur Anzeige oder Änderung eines vorhandenen Datenbank-Trigger zu verwenden, erweitern Sie die Anzeige des Datenbankobjektknotens, und es wird eine Liste der Oracle Accounts angezeigt. Wenn Sie ein Oracle Account auswählen und erweitern,

sehen Sie eine andere Ebene an Knoten, einschließlich Stored Program Units, Programmblöcken, Bibliotheken, Tabellen und Ansichten (siehe Abbildung 16.12).

Wenn Sie den Tabellenknoten erweitern, sehen Sie eine Liste der dem ausgewählten Account gehörenden Tabellen. Wählen Sie eine Tabelle aus und erweitern Sie diese, sehen Sie darunter zwei Knoten: Trigger und Spalten. Erweitern Sie den Trigger-Knoten und doppelklicken Sie auf Symbol links dieses Trigger.



## Einen gespeicherten Programmblock betrachten und bearbeiten

Schauen wir uns ein Beispiel an, wie ein gespeicherter Programmblock angezeigt und verändert werden kann. Nehmen Sie an, Sie wollen die gespeicherte, dem Oracle Account flugle gehörende Funktion Student\_GPA ändern. Wie Sie sich erinnern werden, wird die Funktion Student\_GPA dazu benutzt, die in Buchstaben eingegebenen Zensuren der Studenten - A, B, C usw. - zu übersetzen und einen numerischen Notendurchschnitt zurückzugeben. Im Object Navigator erweitern Sie den Datenbankobjektknoten und den Knoten Flugle. Wählen Sie den Knoten Student\_GPA (siehe Abbildung 16.13).

Doppelklicken Sie auf das Symbol links vom Knoten Student\_GPA. Es sollte der PL/SQL-Quelltext der Funktion Student\_GPA vom Stored-Program-Unit-Editor angezeigt werden (siehe Abbildung 16.14).

Schauen wir uns einige der Elemente der Benutzerschnittstelle des Stored-Program-Unit-Editor an. Oben im Fenster befinden sich zwei Drop-down-Listen: Owner und Name. Wenn Sie auf Owner klicken, wird eine Liste aller vorhandenen Oracle Accounts in der mit Procedure Builder verbundenen Datenbank angezeigt, auch wenn diesem Account keine gespeicherten Programmblöcke gehören (siehe Abbildung 16.15). In ähnlicher Form zeigt die Drop-down-Liste Name alle gespeicherten Programmblöcke an - Paketmerkmale, Paketrümpfe, Prozeduren und Funktionen - die dem in Owner angezeigten Oracle Account gehören (siehe Abbildung 16.16).

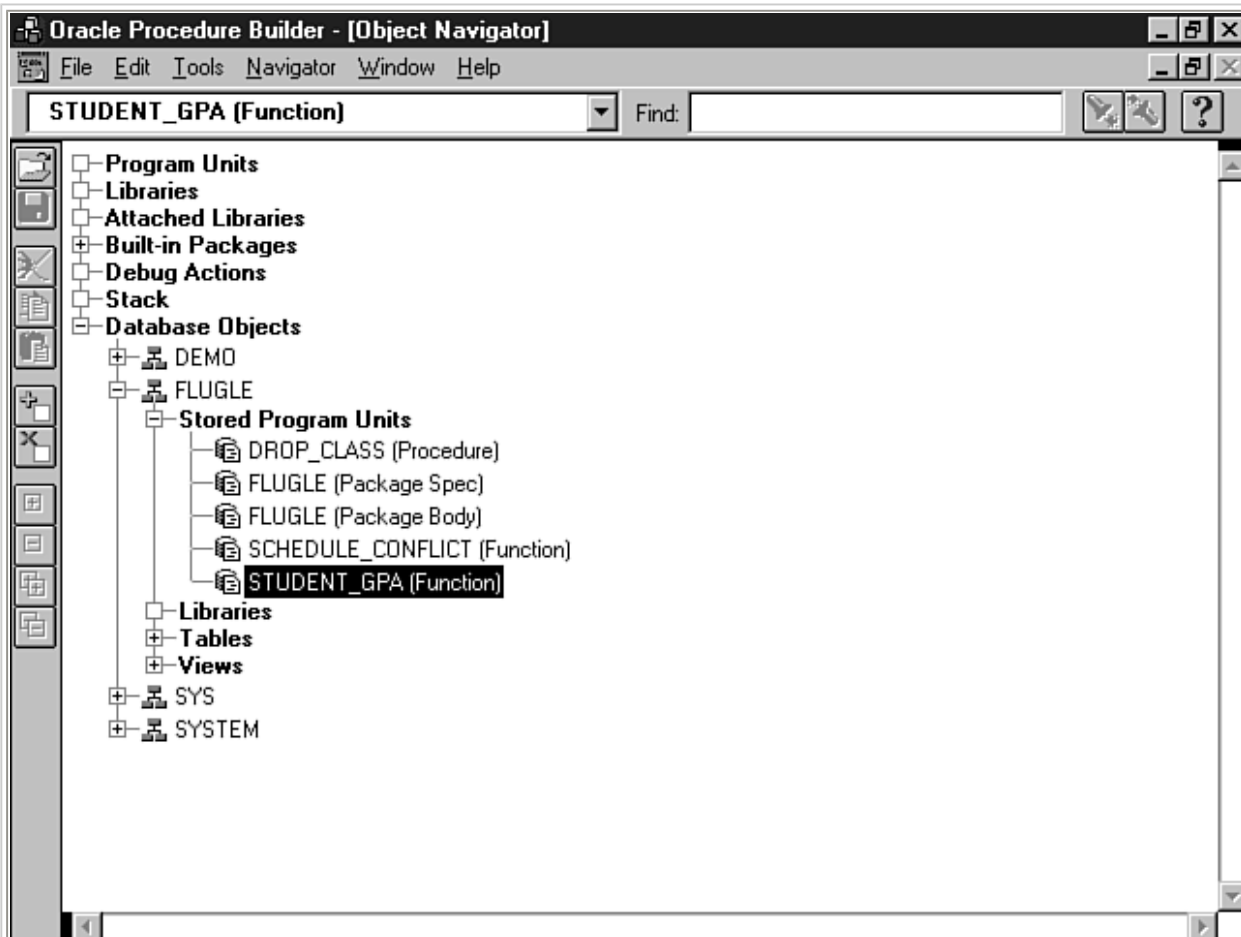


Abbildung 16.13:  
Auswahl eines  
gespeicherten  
Programmblocks mit dem  
Object Navigator.

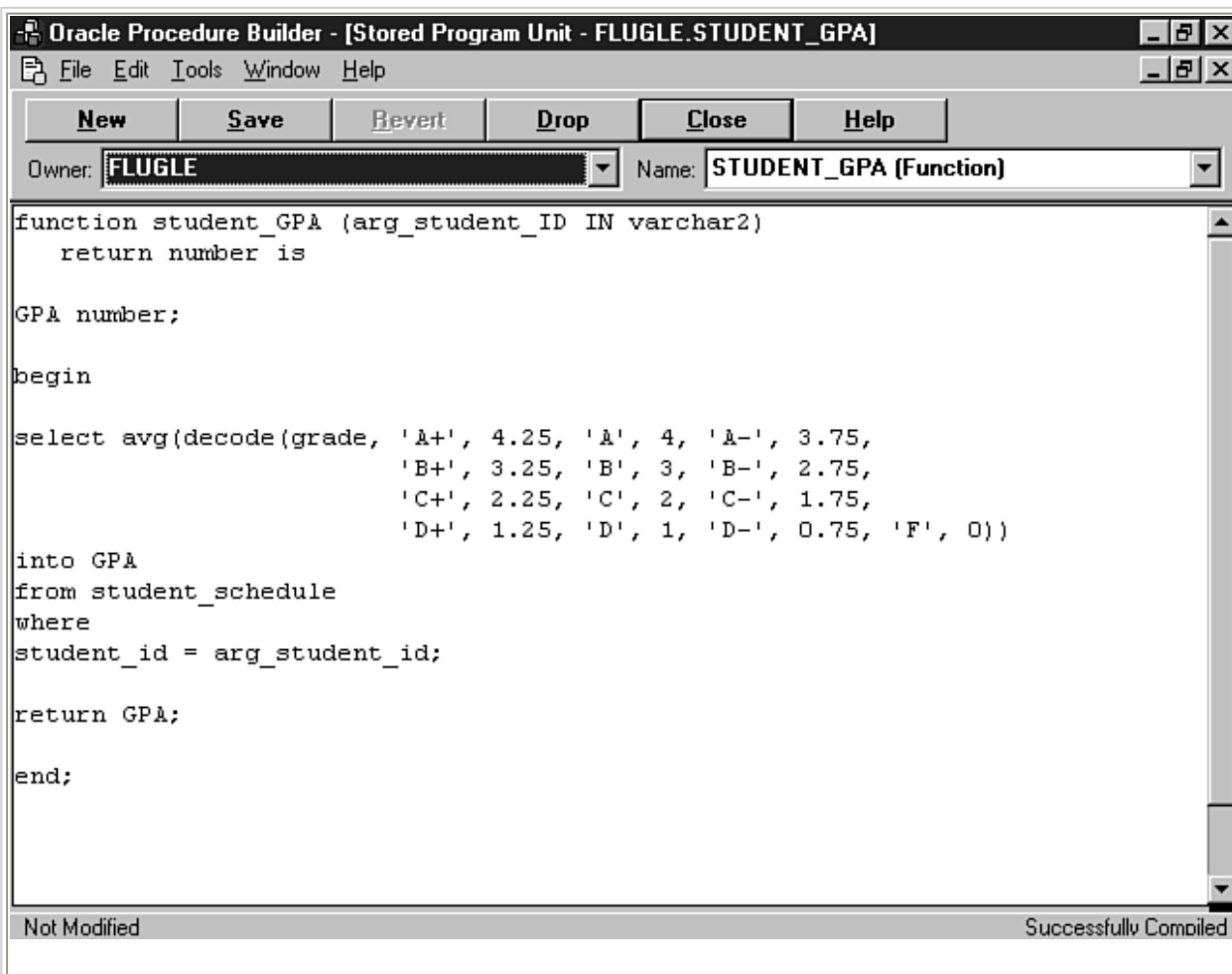


Abbildung 16.14:  
Betrachtung einer  
gespeicherten Funktion mit  
dem Stored-Program-Unit-  
Editor.

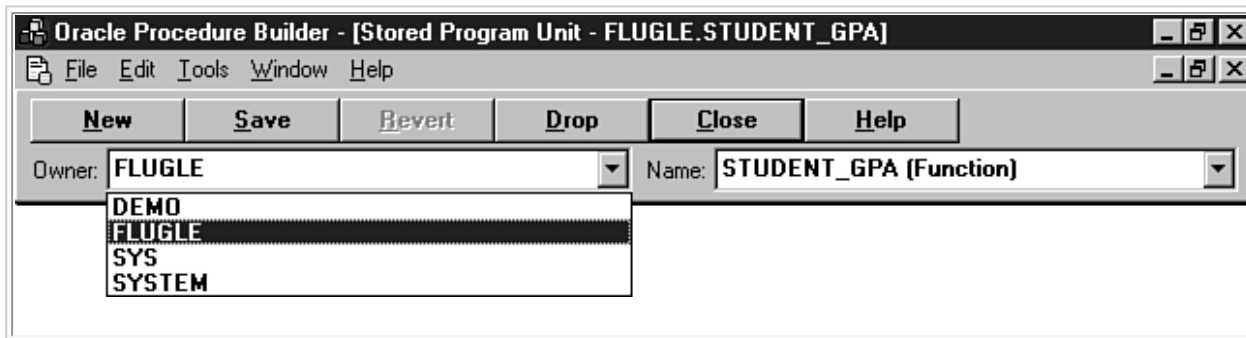


Abbildung 16.15:  
Die Drop-down-Liste  
Eigentümer zeigt alle  
Oracle Accounts an.

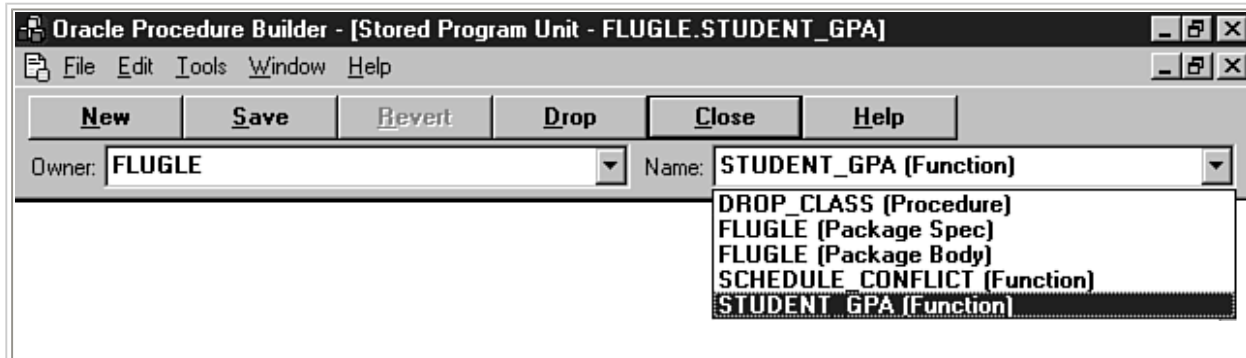


Abbildung 16.16:  
Die Drop-down-Liste  
zeigt alle  
Programmblöcke an, die  
im Besitz des in der  
Liste Eigentümer  
angezeigten Oracle  
Account sind.

Außerdem befinden sich oben in diesem Fenster sechs Schaltflächen:

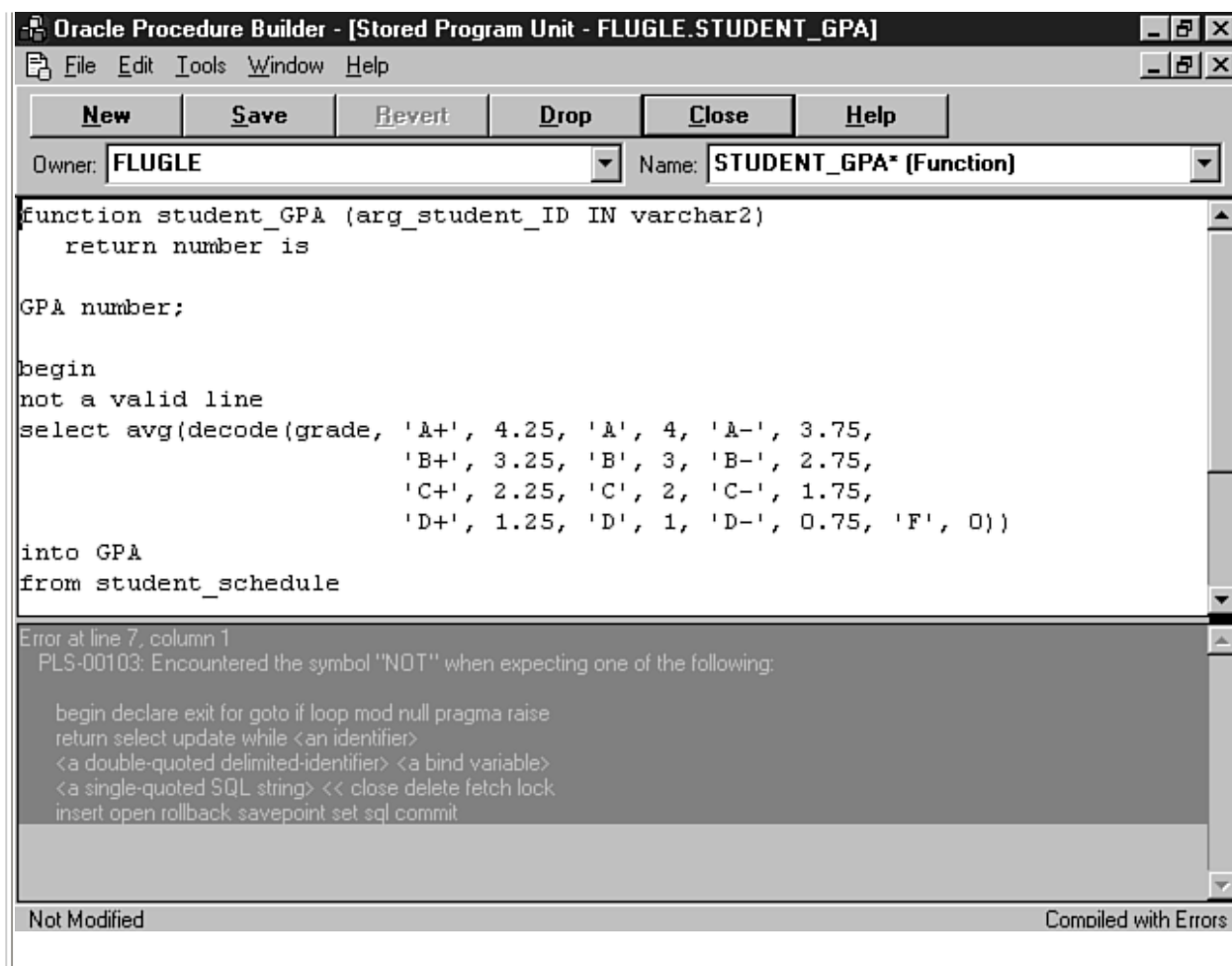
- New Erstellt einen neuen Programmblock.
- Save Speichert alle am Programmblock vorgenommenen Änderungen.
- Revert Widerruft alle am Programmblock vorgenommenen Änderungen.
- Drop Verläßt den aktuellen Programmblock.
- Close Schließt den Stored-Program-Unit-Editor.
- Help Zeigt die Online-Hilfe für den Stored-Program-Unit-Editor an.



Die Schaltfläche *Revert* steht erst zur Verfügung, nachdem Sie am Programmblock eine Änderung vorgenommen haben. Und wenn Sie die Schaltfläche *Close* wählen, werden Sie gefragt, ob die Änderungen am Programmblock vorgenommen oder rückgängig gemacht werden sollen.

Abbildung 16.17:  
Der Stored-Program-  
Unit-Editor zeigt  
PL/SQL-Fehler an.





Wenn Sie der Funktion Student\_GPA eine ungültige Zeile hinzufügen und Save anklicken, versucht der Stored-Program-Unit-Editor die Funktion zu kompilieren, und gibt am unteren Fensterrand eine Fehlermeldung aus (siehe Abbildung 16.17). Wenn Sie auf die Zeile hinter begin schauen, sehen Sie eine ungültige Anweisung: not a valid line.

Entfernen wir die störende Zeile, und lassen Sie uns die Funktion ändern, indem wir die Note F aus der Berechnung des GPA entfernen (siehe Listing 16.1). Klicken Sie auf Save, um die Änderungen zu kompilieren und die Funktion zu speichern. Klicken Sie auf Close, um den Stored-Program-Unit-Editor zu schließen.

**Listing 16.1: Änderung einer gespeicherten Funktion.**

```
function student_GPA (arg_student_ID IN varchar2)
return number is

GPA number;

begin

select avg (decode (grade, 'A+', 4.25, 'A', 4, 'A-', 3.75,
'B+', 3.25, 'B', 3, 'B-', 2.75,
'C+', 2.25, 'C', 2, 'C-', 1.75,
'D+', 1.25, 'D', 1, 'D-', 0.75))

into GPA
from student_schedule
where
student_id = arg_student_id;

return GPA;

end;
```

## Erstellen eines Datenbank-Trigger mit dem Editor für Datenbank-Trigger

Den Database-Trigger-Editor können Sie zur Erstellung, Anzeige, Änderung und Löschung eines Datenbank-Trigger einsetzen. Schauen wir uns ein Beispiel an. Angenommen, eine Anforderung an das Flugle-College-Student-Information-System wäre, daß ein Journal mit allen Noten eines Studenten, einschließlich aller Änderungen an diesen, geführt werden müßte. Wie Sie noch wissen, enthält die Tabelle Student\_Schedule jeden von einem Studenten abgeschlossenen Kurs, einschließlich der erreichten Noten und dem Datum, wann diese vergeben wurden.

Um die Anforderung zu erfüllen, erstellen Sie eine weitere Tabelle namens Student\_Schedule\_Journal, in der alle Spalten der Tabelle Student\_Schedule und drei zusätzliche Spalten enthalten sind:

- Operation Ein Buchstabe, der der DML-Operation entspricht: I für Einfügen, U für Update und D für Löschen
- Changed\_by Der Name des Oracle-Benutzers, der die Operation durchführt
- changed\_date Das Datum, an dem die Operation durchgeführt wurde

Verwenden Sie den Database-Trigger-Editor, um für die Tabelle Student\_Schedule einen Datenbank-Trigger zu erstellen, der die Informationen über die Zeile in der Tabelle Student\_Schedule\_Journal speichert, die entweder eingefügt, aktualisiert oder gelöscht wird.

1. Rufen Sie den Database-Trigger-Editor auf, indem Sie den Menüeintrag Tools | Database Trigger Editor wählen.
2. Wählen Sie FLUGLE aus der Drop-down-Liste unter der Beschriftung Table Owner (siehe Abbildung 16.19).
3. Wählen Sie STUDENT\_SCHEDULE aus der Drop-down-Liste unter der Beschriftung Table (siehe Abbildung 16.20).
4. Klicken Sie auf New, um den Datenbank-Trigger zu erstellen. Den standardmäßig vergebenen Namen für den Datenbank-Trigger sollten Sie ändern.
5. Rechts oben im Fenster ändern Sie unterhalb dem Feld Name den Namen des Datenbank-Trigger in AIUD\_STUDENT\_SCHEDULE. Diese Abkürzung bedeutet, daß dieser Trigger ein *After-Insert-Update-Delete*-Trigger für die Tabelle Student\_Schedule ist.
6. Aktivieren Sie das Kontrollkästchen After.
7. In der Gruppe Statement, markieren Sie alle Kontrollkästchen, UPDATE, INSERT und DELETE.
8. Markieren Sie das Kontrollkästchen For Each Row.

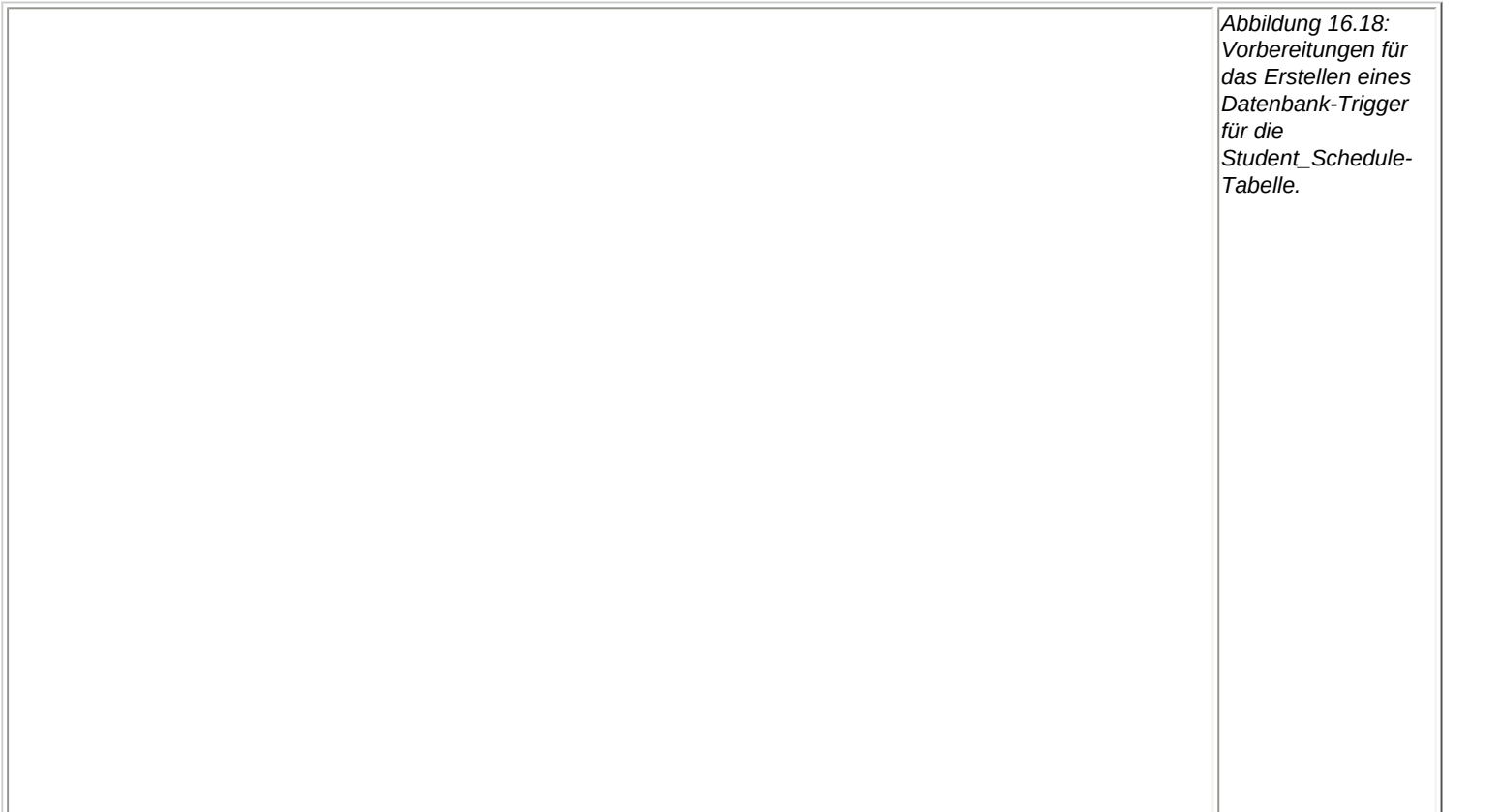


Abbildung 16.18:  
Vorbereitungen für  
das Erstellen eines  
Datenbank-Trigger  
für die  
Student\_Schedule-  
Tabelle.

**Oracle Procedure Builder - [Database Trigger]**

File Edit Tools Window Help

Table Owner: **FLUGLE** Table: **STUDENT\_SCHEDULE** Name: **(None)**

Triggering: ☒ Before ☐ After

Statement: ☐ UPDATE ☐ INSERT ☐ DELETE

Of Columns:

☐ For Each Row

Referencing OLD As: NEW As:

When:

Trigger Body:

New Save Revert Drop Close Help

**Oracle Procedure Builder - [Database Trigger]**

File Edit Tools Window Help

Table Owner: **FLUGLE** Table: **STUDENT\_SCHEDULE** Name: **AIUD\_STUDENT\_SCHEDULE**

Triggering: ☐ Before ☒ After

Statement: ☒ UPDATE ☒ INSERT ☒ DELETE

Of Columns: CLASS\_ID DATE\_GRADE\_ASSIGNED GRADE STUDENT\_ID

☒ For Each Row

Referencing OLD As: NEW As:

When:

Trigger Body:

```
BEGIN
END;
```

New Save Revert Drop Close Help

Abbildung 16.19: Setzen der Optionen für den neuen Datenbank-Trigger.

Abbildung 16.19 zeigt, was Sie abschließend im Database-Trigger-Editor sehen sollten.

Geben Sie Listing 16.2 in die Drop-down-Liste mit der Bezeichnung Trigger Body ein, klicken Sie auf Save, um den Trigger zu kompilieren und zu speichern.

**Listing 16.2: PL/SQL-Block für den Rumpf des Datenbank-Trigger.**

```
begin
if inserting then
insert into student_schedule_journal
(operation, student_ID, Class_ID, Grade, Date_Grade_Assigned,
Changed_By, Changed_Date)
values
('I', :new.Student_ID, :new.Class_ID, :new.Grade,
ew.Date_Grade_Assigned, user, sysdate);
elsif updating then
insert into student_schedule_journal
(operation, student_ID, Class_ID, Grade, Date_Grade_Assigned,
Changed_By, Changed_Date)
values
('U', :new.Student_ID, :new.Class_ID, :new.Grade,
ew.Date_Grade_Assigned, user, sysdate);
elsif deleting then
insert into student_schedule_journal
(operation, student_ID, Class_ID, Grade, Date_Grade_Assigned,
Changed_By, Changed_Date)
values
('I', :new.Student_ID, :new.Class_ID, :new.Grade,
ew.Date_Grade_Assigned, user, sysdate);
end if;
end;
```



Wie Sie in Listing 16.2 sehen, besteht der Trigger-Rumpf aus einem PL/SQL-Block mit einer IF-Anweisung. Jede Klausel der IF-Anweisung überprüft die ausgeführte Operation. So überprüft der erste Teil der IF-Anweisung, ob das Trigger-auslösende Ereignis INSERT ist; wenn dem so ist, wird in der Tabelle Student\_Schedule\_Journal eine Zeile mit den folgenden Werten eingefügt. Klicken Sie auf Close, um den Database-Trigger-Editor zu schließen.

Operation I, weil es eine INSERT-Operation ist

Student\_ID :new.student\_ID, enthält den Wert für den ersten Studenten, der der Tabelle Student\_Schedule hinzugefügt wird

Grade :new.Grade, enthält den Wert für die Note, die der Tabelle Student\_Schedule hinzugefügt wird

Date\_Grade\_Assigned :new.Date\_Grade\_Assigned, enthält den Wert für Date\_-Grade\_Assigned, der der Tabelle Student\_Schedule hinzugefügt wird

Changed\_by USER, eine integrierte Funktion, die den Namen des Oracle-Benutzers, der die Anweisung ausführt, zurückgibt

Changed\_Date SYSDATE, eine integrierte Funktion, die die aktuelle Zeit und das Datum zurückgibt

## Testen des neuen Datenbank-Trigger

Lassen Sie uns den Trigger ausprobieren. Dies können Sie entweder mit SQL\*Plus oder SQL-Worksheet tun.

1. Bauen Sie eine Verbindung zum flugle Account auf.

2. Zu Beginn überprüfen Sie, ob die Tabelle `Student_Schedule_Journal` leer ist (siehe Abbildung 16.20).

3. Rufen Sie aus der Tabelle `Student_Schedule` alle Datensätze mit der `Student_ID` 10231324 ab (das ist Anna Anastasia). Wie Sie Abbildung 16.21 entnehmen können, wird nur ein Kurs aufgelistet: 109100 (das ist Biologie 101).

4. Anna hat in diesem Semester schwer gearbeitet und verdient für diesen Kurs die Note A. Um Ihr für diesen Kurs die Note A zu erteilen, wählen Sie eine UPDATE-Anweisung, damit der Wert für `GRADE` auf A gesetzt wird (siehe Listing 16.3).

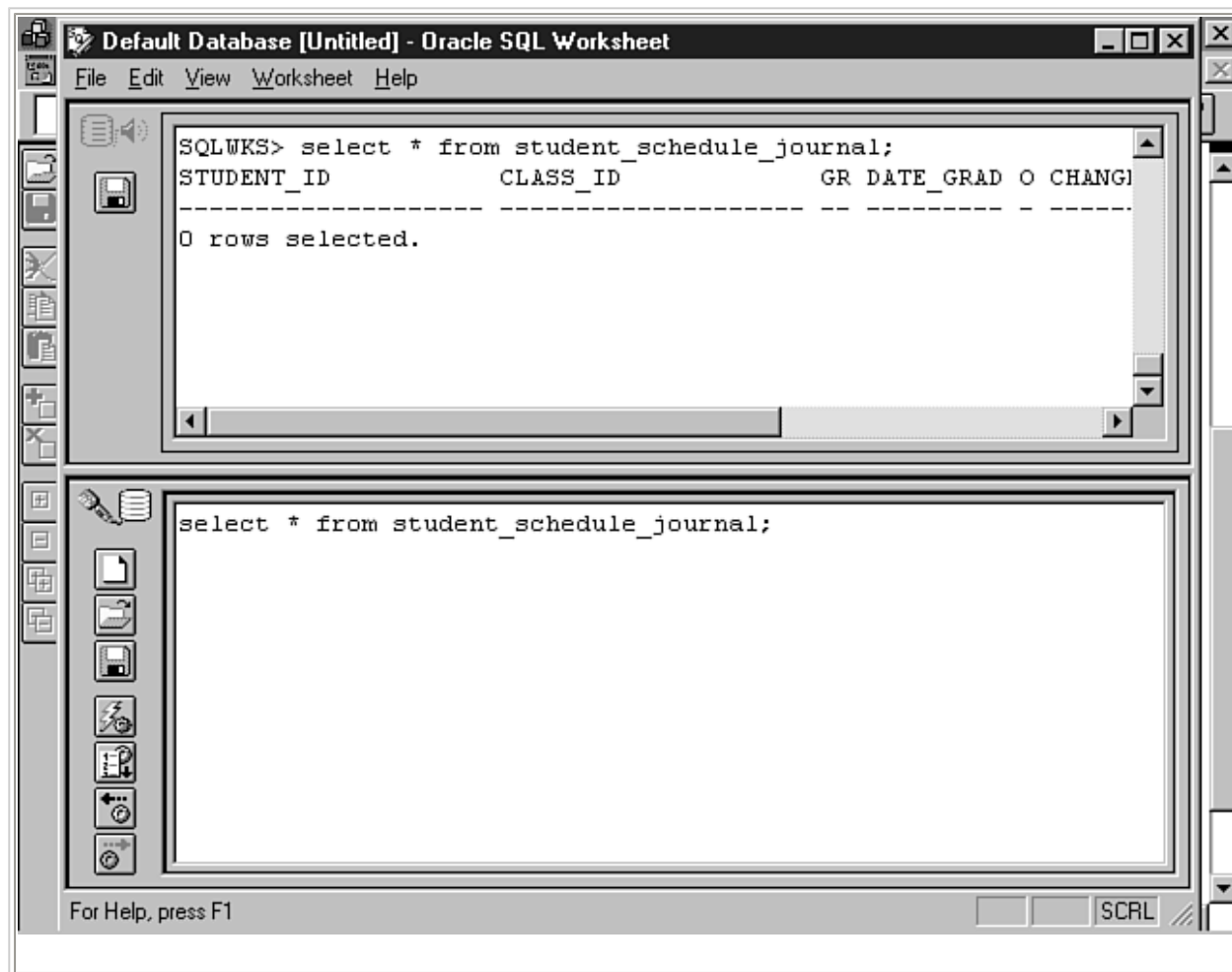


Abbildung 16.20:  
Die Tabelle  
`Student_Schedule_Journal`  
ist leer.

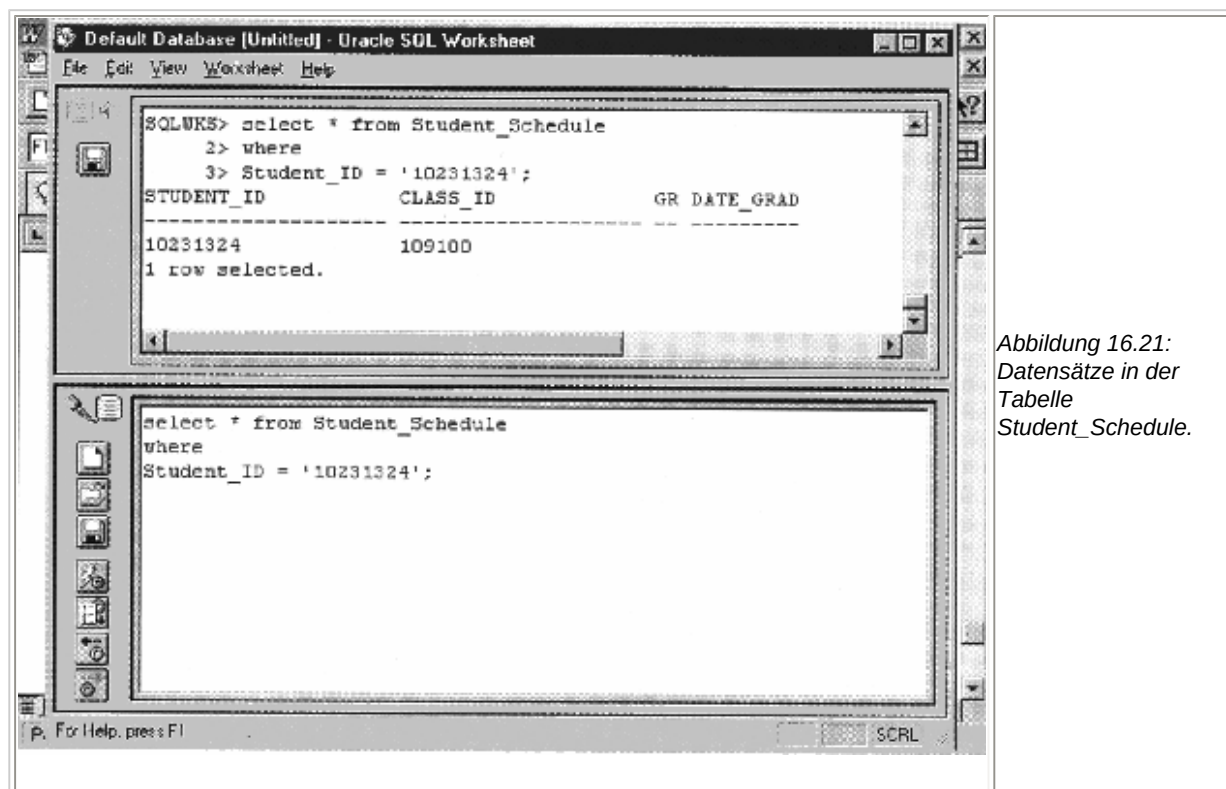


Abbildung 16.21:  
Datensätze in der  
Tabelle  
`Student_Schedule`.

**Listing 16.3: Zuweisung einer Note mit einer update-Anweisung.**

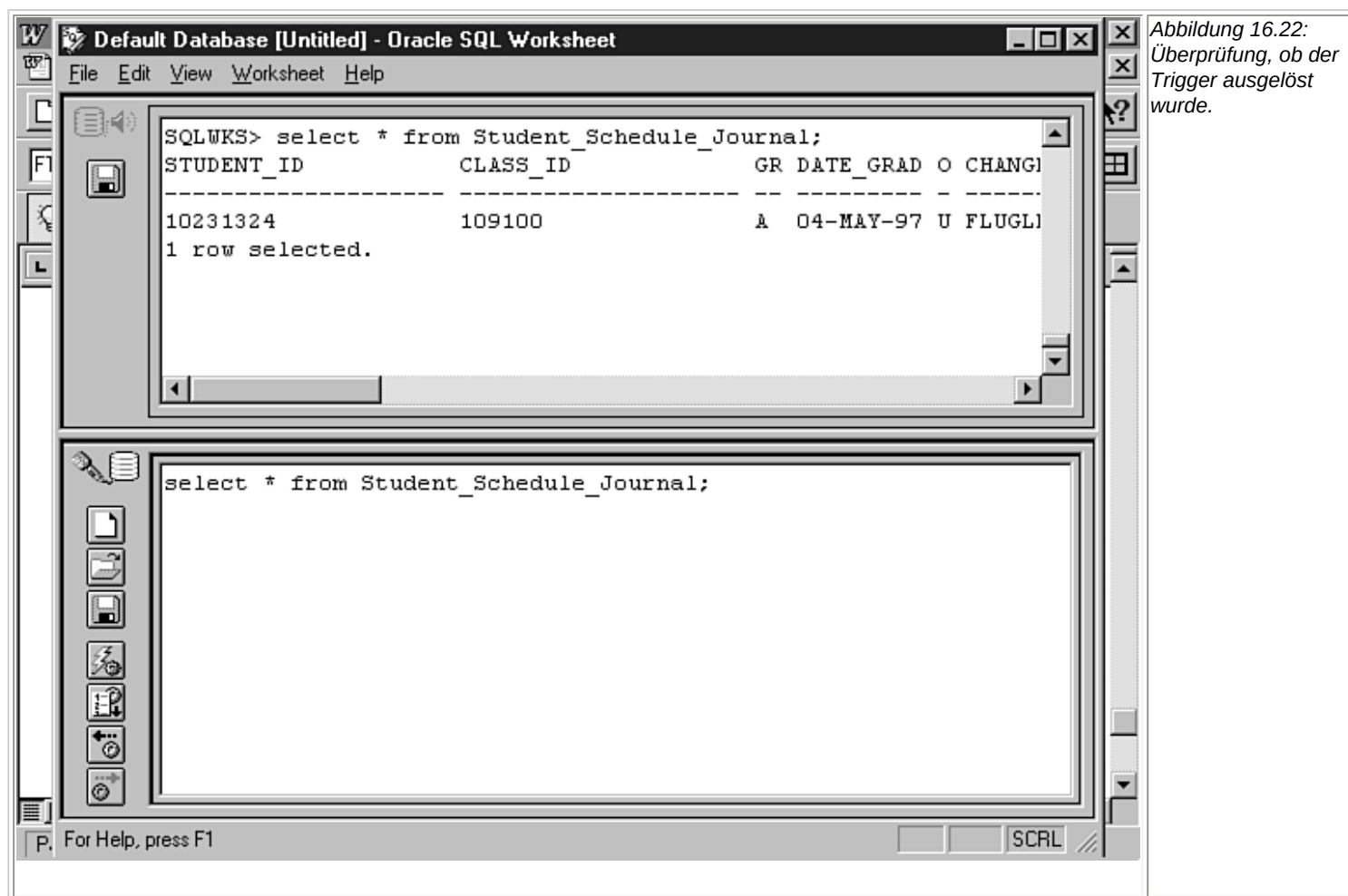
```

update Student_Schedule
set
Grade = 'A',
Date_Grade_Assigned = sysdate
where
Student_ID = '10231324' and
Class_ID = '109100';

```

Wie in Abbildung 16.22 gezeigt, enthält Student\_Schedule\_Journal jetzt eine einzige Zeile mit dem erwarteten Wert einschließlich:

- Operation U, entspricht der ausgeführten UPDATE-Anweisung
- Grade A, entspricht dem geänderten Wert
- Date\_Grade\_Assigned 04\_May-97, entspricht dem Datum, an dem die Zeile aktualisiert wurde



## Zusammenfassung

In dieser Lektion wurden die folgenden Schlüsselthemen zu Oracle Graphics und Procedure Builder erkundet:

- Oracle Graphics besteht aus zwei Programmen: Graphics-Designer und Graphics Runtime.
- Durch die Definition einer Abfrage und eines Diagrammtyps erstellen Sie ein Diagramm. Eine Abfrage kann auf einer Tabelle, zwei oder mehr verknüpften Tabellen oder einer Ansicht basieren.
- Oracle Graphics stellt viele Möglichkeiten zur Gestaltung und dem Layout eines Diagramms bereit.
- Procedure Builder stellt ein handliches Werkzeug zur Anpassung von PL/SQL-Unterprogrammen in einer Reihe von Quellen dar, einschließlich gespeicherter Programmblöcke, Bibliotheken und Paketrümpfen.
- Der Stored-Program-Unit-Editor erlaubt Ihnen das Erstellen, Betrachten, Ändern und Löschen von gespeicherten Programmblöcken, gespeicherten Funktionen, Paketmerkmalen und Paketrümpfen.
- Wenn Sie mit einem Oracle Account mit den entsprechenden Rechten verbunden sind, können Sie gespeicherte Programmblöcke und Datenbank-Trigger betrachten, die anderen Oracle Accounts gehören.

- Mit dem Database-Trigger-Editor können Sie einen Datenbank-Trigger erstellen, ändern und löschen. Sie können die den Trigger auslösende Operation und den Zeitpunkt (vor oder nach der Operation) festlegen.

## Wie geht es weiter?

An Tag 17, »Einführung in Oracle Power Objects«, lernen Sie dieses intuitive, objektorientierte Werkzeug zur Entwicklung von Client-Server-Anwendungen kennen. Power Objects unterscheidet sich merklich von Oracle Forms und verwendet eine Skript-Sprache namens Oracle Basics.

## Fragen und Antworten

Frage:

Bietet Oracle Graphics die Möglichkeit, in ein Diagramm hineinzugehen, um die Einzelheiten eines Diagrammelements zu sehen?

Antwort:

*Ja. Sie können eine Detail-Abfrage festlegen, die ausgeführt wird, wenn der Benutzer auf ein Element des Diagramms klickt. Das Hauptdiagramm wird als Master-Diagramm bezeichnet. Zur Definition der Beziehung zwischen Master- und Detaildiagramm wird ein Parameter benutzt.*

Frage:

Welches sind einige der integrierten Pakete, die mit dem Procedure Builder betrachtet werden können?

Antwort:

*Die integrierten Pakete umfassen:*

- TEXT\_IO um Dateien auszulesen oder in Dateien zu schreiben
- DDE, das den dynamischen Datenaustausch zwischen Oracle-Forms-, Reports-, Graphics- und Windows-basierten Programmen bereitstellt
- TOOL\_ENV, um die Werte aus Oracle-Umgebungsvariablen auszulesen

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Welche beiden Dateitypen verwendet Oracle Graphics?
2. Nennen Sie drei Diagrammtypen, die mit dem Graphics-Designer festgelegt werden können.
3. Richtig oder falsch? Sie können Stored-Program-Unit-Editor zur Änderung der Merkmale eines Pakets aber nicht des Pakettrumpfs verwenden.

## Übung

1. Erstellen Sie ein Tortendiagramm, das die von den einzelnen Fachrichtungen des Flugle College angebotenen Kurse darstellt.
2. Ändern Sie den Trigger für die Student\_Schedule-Tabelle in der Form, daß in der Student\_Schedule-Tabelle Änderungen nur gespeichert werden, wenn der Wert in der Spalte Grade A, B oder C ist.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖Kapitel❖ [Top](#) [Inhalt](#) [Index](#) **SAMS** Kapitel❖



# 17

# Einführung in Oracle Power Objects

[Elemente einer Oracle-Power-Objects-Anwendung](#)

[Erzeugen einer neuen Arbeitssitzung](#)

[Erstellen einer neuen Anwendung](#)

[Erstellen eines Formulars](#)

[Erstellen eines Master-/Detail-Formulars](#)

[Verwenden von Optionsschaltflächen](#)

[Anzeige eines Suchfeldes](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

Power Objects werden von allen drei Windows-Plattformen (Windows 3.11, Windows 95 und Windows NT) und der Macintosh-Plattform (System 7.1.1, 7.1.2, oder 7.5.3 und später) unterstützt. Das Material in dieser und der nächsten Lektion wird sich auf die Anwendung von Power Objects Version 2.0 unter Windows 95 konzentrieren.

## Elemente einer Oracle-Power-Objects-Anwendung

Um eine Power-Objects-Anwendung zu erstellen, werden Sie diese Objekte manipulieren:

- **Datenbank-Arbeitssitzung:** Eine Datenbank-Arbeitssitzung wird als Verbindung zu einer Oracle-Datenbank (oder ODBC-Datenquelle) mittels eines Benutzernamens, Kennworts und einem Datenbank-Alias definiert. Eine Datenbank-Arbeitssitzung bietet den Zugriff auf Tabellen, Datenansichten, Sequenzen, Synonyme und Indizes. Eine Datenbank-Arbeitssitzung wird als Datei mit der Erweiterung .pos gespeichert.
- **Anwendung:** Eine Sammlung von Formularen, Berichten, Klassen, Bitmap-Dateien und OLE-Objekten, die mit einer Datenbank

über eine Datenbank-Arbeitssitzung kommunizieren. Eine Anwendung wird als Datei mit der Erweiterung .poa gespeichert.

- **Bibliothek:** Eine Sammlung von Klassen und Bitmap-Dateien, die von vielen Anwendungen geteilt genutzt werden kann. Eine Bibliothek wird als Datei mit der Erweiterung .pol gespeichert.

Der beste Weg, etwas über Power Objects zu lernen, ist das schrittweise Erstellen einer Anwendung. Um den Power-Objects-Designer zu starten, wählen Sie Start | Programs | Oracle Power Objects 2.0 | Power Objects Designer; Abbildung 17.1 zeigt das Anfangsfenster des Power-Objects-Designer. Die Power-Objects-Arbeitsfläche besteht aus vier Hauptbereichen:

- Der Menüleiste
- Der Symbolleiste
- Dem Hauptfenster
- Dem Statusbereich am unteren Ende des Bildschirms



## Erzeugen einer neuen Arbeitssitzung

Der erste Punkt bei der Erstellung einer neuen Anwendung ist das Erzeugen einer neuen Datenbank-Arbeitssitzung. Um dieses zu tun, können Sie entweder den Menüeintrag File | New Session wählen oder das Symbol auf der Symbolleiste (sieht aus wie ein Netzstecker) anklicken. Ein Fenster mit dem Titel Create Session erscheint, welches vier Felder enthält:

Database Standardmäßig ist hier Oracle ausgewählt.

Username Der Name des Oracle Account, zu dem Sie die Verbindung herstellen wollen (geben Sie Flugle ein).

Password Das Kennwort für den in Benutzernamen enthaltenen Oracle Account (geben Sie auch hier Flugle ein).

Connect String Falls Sie Personal-Oracle verwenden, lassen Sie dieses Feld frei. Falls Sie Verbindung zu einem Oracle-Server oder einem anderen Rechner aufnehmen, geben Sie den entsprechenden Datenbank-Alias ein.

Wenn Sie die entsprechenden Werte in die Felder eingegeben haben, klicken Sie auf OK, um die Datenbank-Arbeitssitzung zu erzeugen.

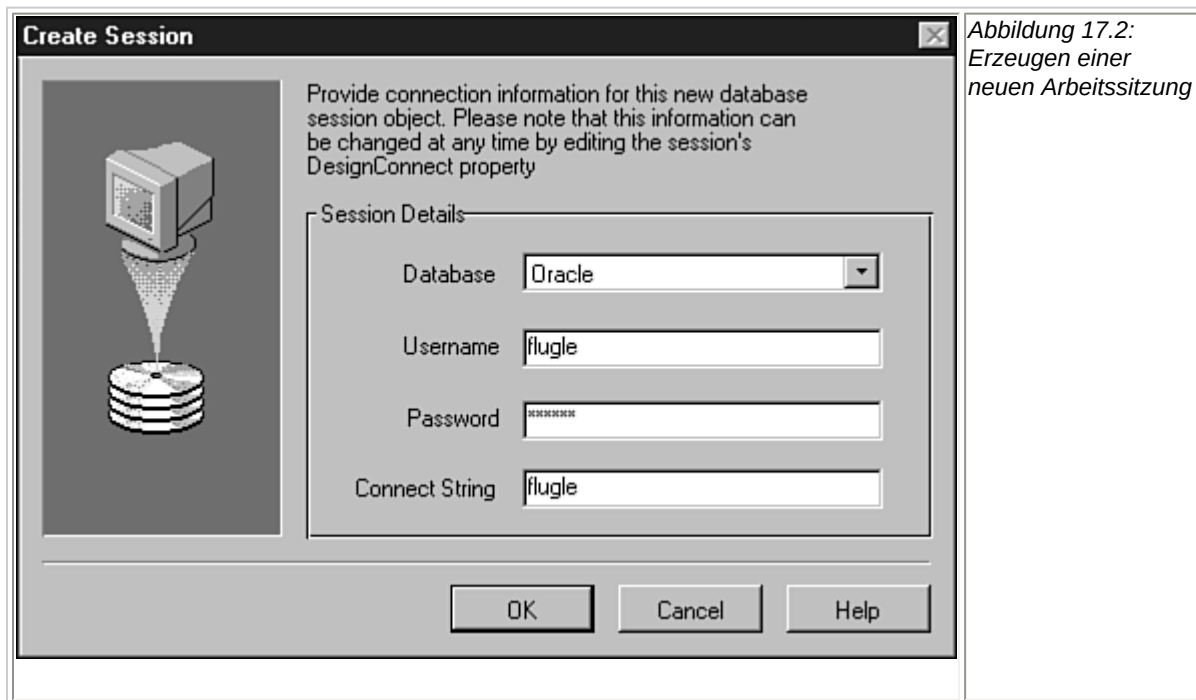


Abbildung 17.2:  
Erzeugen einer  
neuen Arbeitssitzung

Ist die Arbeitssitzung einmal erzeugt, erscheint das Attributblatt für die Arbeitssitzung mit der Flugle-Datenbank auf der Arbeitsfläche (siehe Abbildung 17.3).

Im Hauptfenster sehen Sie die neue Datenbank-Arbeitssitzung. Klicken Sie, wie üblich, das + an. Wie Sie in Abbildung 17.4 sehen können, werden alle zum Flugle Account gehörenden Datenbankobjekte im Hauptfenster angezeigt.

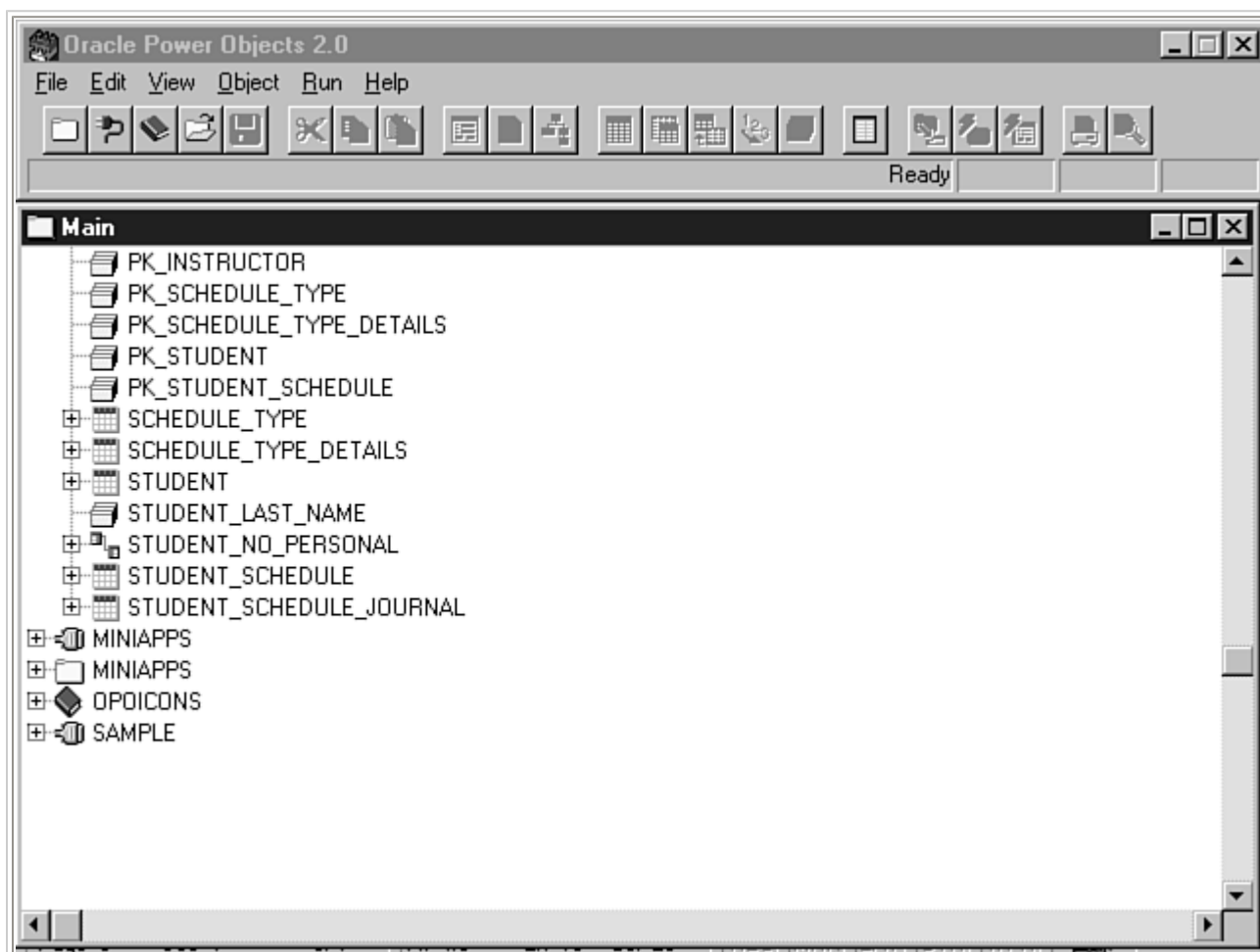


Abbildung 17.3:  
Das Attributblatt  
einer Arbeitssitzung  
mit Power Objects.



*Es ist wichtig, sich bewußt zu machen, daß jedes Objekt seinen eigenen Satz von Eigenschaften besitzt. Die beispielsweise für ein Arbeitssitzungsobjekt definierten Eigenschaften unterscheiden sich von den Eigenschaften, die für ein Anwendungsobjekt definiert wurden. (Selbstverständlich könnte die Eigenschaft eines Anwendungsobjekts ein Arbeitssitzungsobjekt benennen.)*

## Die Definition eines Datenbankobjekts betrachten

Jeder Objekttyp wird mit einem anderen Symbol angezeigt. Das Symbol für eine Tabelle zum Beispiel sieht aus wie ein Raster (z.B. Schedule\_Type). Das Symbol für eine Datenansicht sieht aus wie zwei zusammengesetzte Tabellen (z.B. Student\_No\_Personal). Um die Definition einer Tabelle anzuzeigen, doppelklicken Sie auf das Symbol links vom Tabellennamen; ein Fenster mit der Definition der Tabelle wird dann erscheinen. Abbildung 17.5 zeigt z.B. die Definition für eine Tabelle des Typs Schedule\_Type. Beachten Sie das Schlüsselsymbol links neben der Spalte Schedule\_ID, welches die Spalte als Primärschlüssel kennzeichnet. Sie können die Definition einer Tabelle unter Berücksichtigung der Einschränkungen verändern, die in der Lektion an Tag 4, »Implementierung Ihres logischen Modells: physischer Datenbankentwurf«, diskutiert wurden. Wenn Sie die Tabellendefinition modifizieren, fragt Power Objects nach Ihrer Zustimmung, ob Sie die gemachten Veränderungen speichern wollen.

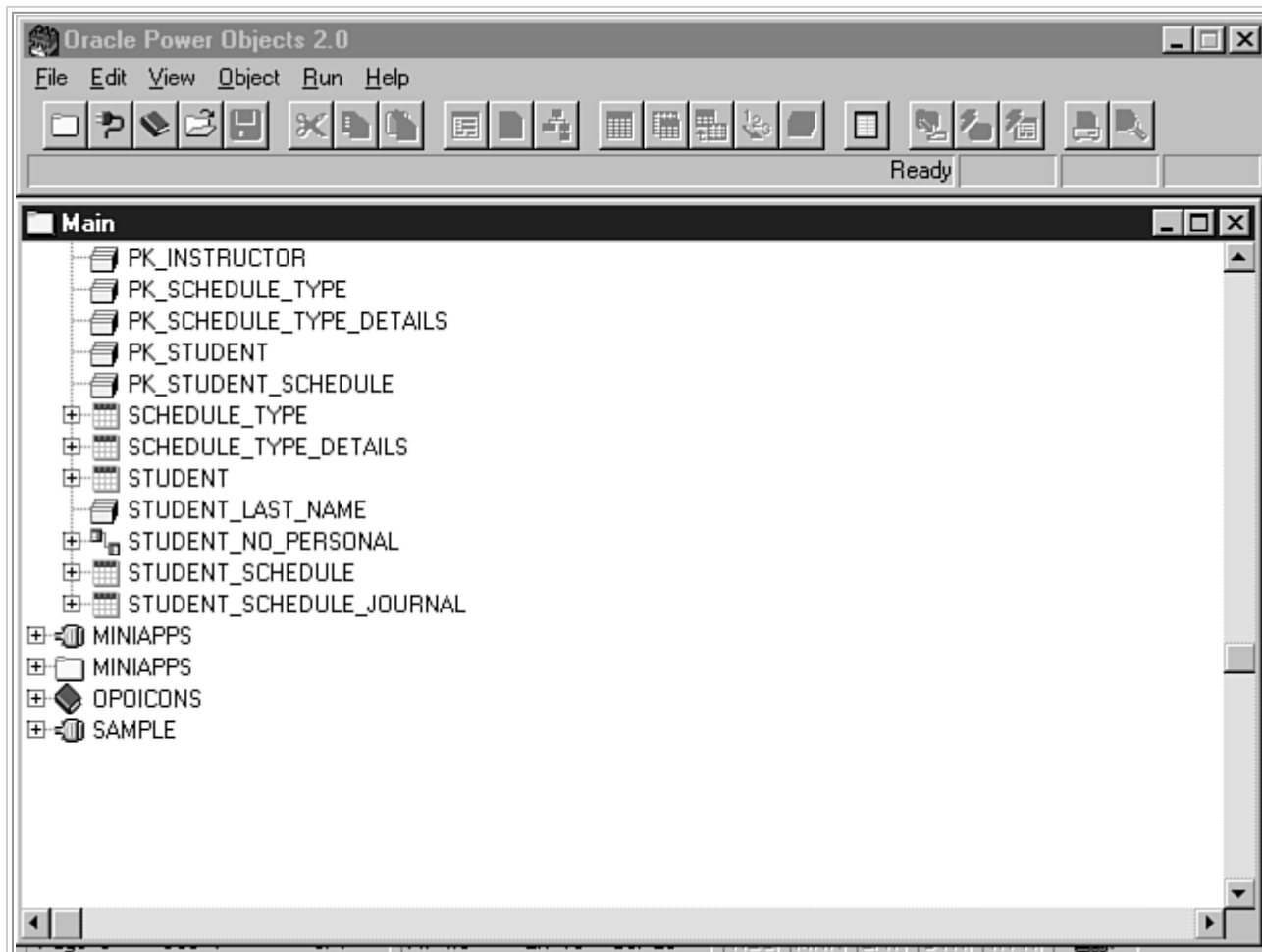


Abbildung 17.4:  
Das Attributblatt der  
Anwendung Power  
Objects.

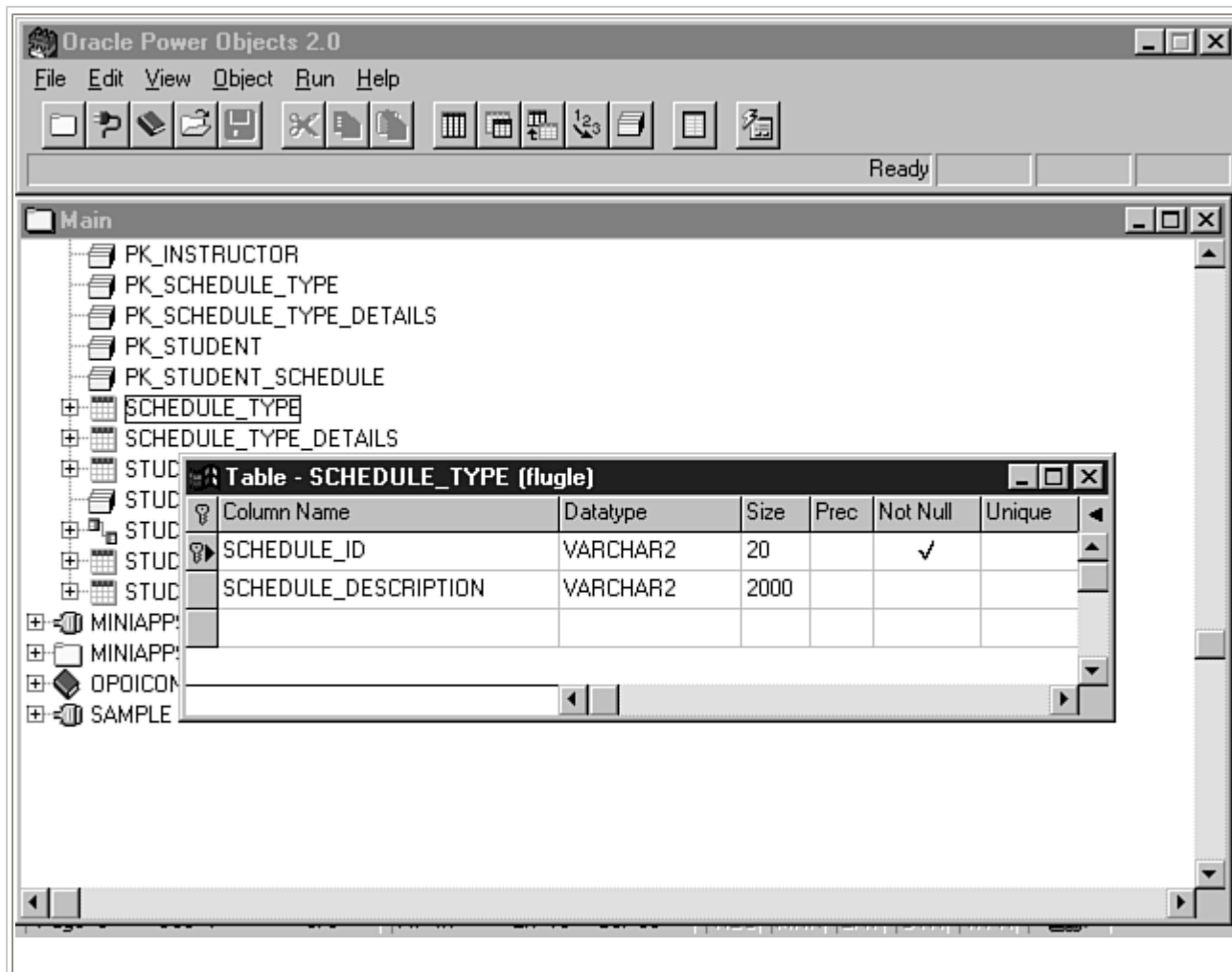


Abbildung 17.5:  
Betrachten einer  
Tabellendefinition im  
Hauptfenster.



Wenn Sie eine Datenbank-Arbeitssitzung erweitern, können Sie neue Datenbankobjekte wie Tabellen, Indizes und Datenansichten erzeugen, indem Sie entweder eine Menüwahl treffen (so wie Object | New Table), oder durch Anklicken des entsprechenden Symbols auf der Symbolleiste (die vierte Gruppe mit Symbolen von links).

## Den Inhalt eines Datenbankobjekts betrachten und ändern

Power Objects ermöglicht Ihnen ebenfalls das Betrachten und Verändern der Inhalte eines Datenbankobjekts, wie der einer Tabelle. Wenn Sie beispielsweise auf das Tabellensymbol links von der Student-Tabelle doppelklicken, zeigt Power Objects die Definition für die Student-Tabelle. Falls Sie den Inhalt dieser Tabelle betrachten oder verändern möchten, können Sie folgendes tun:

- (Strg) + (R) betätigen,
- den Menüeintrag Run | Run wählen,
- das Symbol Run auf der Symbolleiste anklicken (sieht aus wie ein Blitzstrahl).

Anschließend sehen Sie ein Fenster mit dem Titel Browsing Table - STUDENT, welches den Inhalt der Student-Tabelle enthält (siehe Abbildung 17.6).



Oracle Power Objects 2.0

File Edit View Object Run Database Help

Browsing Table - STUDENT (fluge)

| STUDENT_ID | LAST_NAME | FIRST_NAME | MI | YEAR      | STREET_ADDRESS      | CITY      |
|------------|-----------|------------|----|-----------|---------------------|-----------|
| 10231300   | SMYTHE    | JACKSON    | R  | FRESHMAN  | 144 OLIVE AVE.      | DOVER     |
| 10231301   | HAN       | COREY      | Y  | SENIOR    | 9388 MAPLE ST.      | SPRINGFIE |
| 10231302   | GORDON    | IVY        | Y  | SOPHOMORE | 544 ASH ST.         | DOVER     |
| 10231305   | REYNOLDS  | PAULA      | E  | FRESHMAN  | 7493 MAPLE ST.      | SPRINGFIE |
| 10231306   | PARKER    | FREDERICK  | W  | JUNIOR    | 833 ACORN AVE.      | DOVER     |
| 10231308   | TANAKA    | JEFFREY    | E  | FRESHMAN  | 838 PECAN RD.       | DOVER     |
| 10231309   | COEN      | JOSEPH     | W  | JUNIOR    | 994 WALNUT DR.      | DOVER     |
| 10231322   | NEWTON    | ELEANOR    | E  | JUNIOR    | 822 BIRCH RD.       | DOVER     |
| 10231314   | PURCELL   | STEVEN     |    | JUNIOR    | 3833 OAK DR.        | SPRINGFIE |
| 10231315   | JACKEL    | LINDA      | H  | SOPHOMORE | 493 ASH ST.         | DOVER     |
| 10231317   | ROSEN     | ROSEMARY   | E  | SOPHOMORE | 838 EUCALYPTUS AVE. | DOVER     |
| 10231319   | MALLARD   | HENRY      | Q  | FRESHMAN  | 123 WALNUT DR.      | SPRINGFIE |
| 10231320   | GUSSEY    | LISA       |    | JUNIOR    | 311 OLIVE AVE.      | DOVER     |
| 10231325   | MICHAELS  | MARCUS     |    | JUNIOR    | 201 OLIVE AVE.      | DOVER     |
| 10231326   | DEURRE    | PIERRE     | E  | SOPHOMORE | 555 ASH ST.         | SPRINGFIE |
| 10231328   | WONG      | LE-MING    |    | SOPHOMORE | 901 BIRCH RD.       | SPRINGFIE |
| 10231329   | ABBOT     | KENNETH    | I  | JUNIOR    | 827 PINE ST.        | SPRINGFIE |
| 10231327   | POSEN     | HUGO       | Y  | FRESHMAN  | 9100 MAPLE ST.      | DOVER     |
| 10231330   | CLAUSEN   | THOMAS     | Y  | FRESHMAN  | 901 BIRCH RD.       | DOVER     |

Abbildung 17.6:  
Betrachtung des  
Inhalts der Student-  
Tabelle.

Von diesem Fenster aus können Sie außerdem den Inhalt einer Tabelle ändern. Falls Sie zum Beispiel die mittleren Initialen von Anna Anastatia in Q (für Quinn) ändern wollen, würden Sie auf das Feld MI in dem entsprechenden Datensatz klicken und die Änderung durchführen. Wie in Abbildung 17.7 gezeigt, ermöglicht Power Objects es Ihnen, Veränderungen vorzunehmen und ebenso, den Datensatz zu sperren, was vom Sperren-Symbol links neben dem Datensatz angezeigt wird. Sie können ebenfalls eine Zeile einfügen oder löschen, indem Sie entweder den entsprechenden Menüpunkt innerhalb des Datenbank-Menüeintrags wählen oder durch Anklicken des Symbols + oder - auf der Symbolleiste.



Wenn Sie einen Datensatz modifizieren, indem Sie ihn entweder durch die Änderung eines der Felder erstellen, oder indem Sie ihn löschen, wird Power Objects eine Zeilensperre darauf verhängen. Diese Sperre hindert andere Benutzer nicht am Betrachten des Datensatzes, wohl aber an seiner Änderung, bis Sie diese festschreiben oder ein Rollback durchführen.

Falls Sie versuchen, das Fenster mit dem Titel Browsing Table zu schließen, nachdem Sie den Inhalt der Tabelle verändert haben, wird Power Objects Sie informieren, daß eine Transaktion ansteht - Sie müssen entweder die Änderungen festschreiben, die Sie gemacht haben, oder die Änderungen rückgängig machen (siehe Abbildung 17.8). Sie können eine Festschreibung oder ein Rollback entweder durch die Wahl des entsprechenden Menüpunkts innerhalb des Menüeintrags Database oder durch Betätigen des entsprechenden Symbols auf der Symbolleiste (das Festschreiben-Symbol ist eine grüne Kontrollmarke und das Rollback-Symbol ein lilafarbener gebogener Pfeil) ausführen.

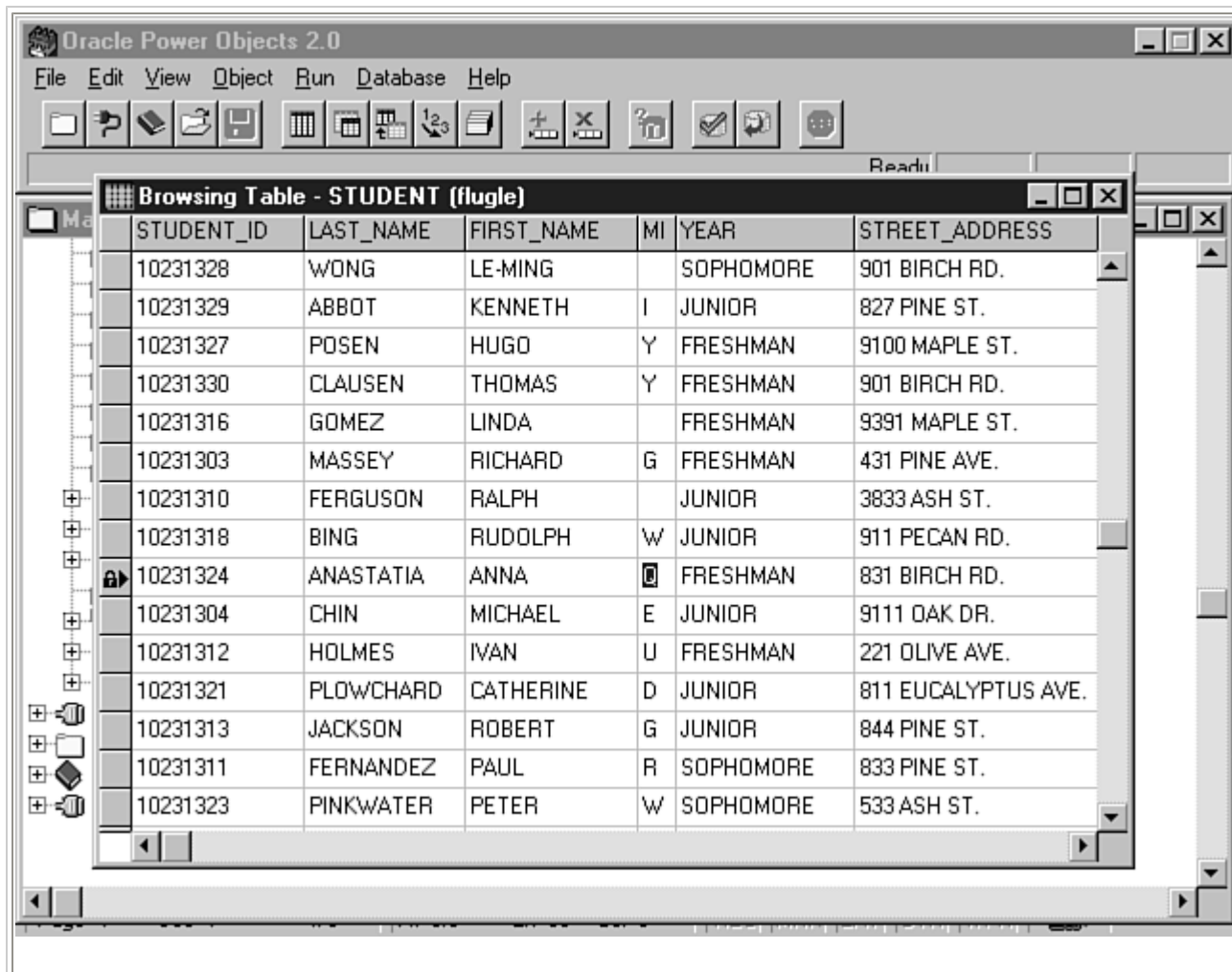


Abbildung 17.7:  
Änderung einer Zeile  
in der Student-  
Tabelle mit dem  
Power-Objects-  
Designer.

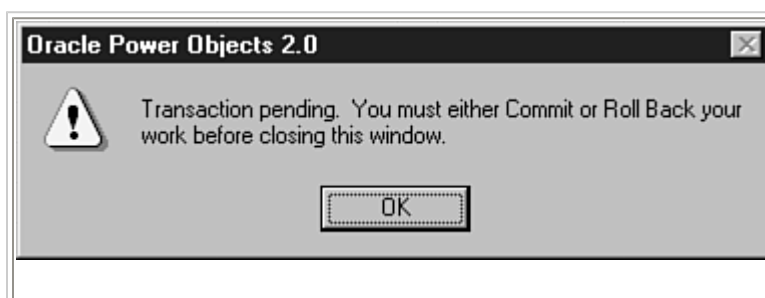


Abbildung 17.8:  
Der Power-Objects-Designer erfordert  
einen Transaktionsabschluß oder ein  
Rollback, bevor das Fenster  
geschlossen wird.

Um das Fenster Browsing Table zu verlassen, können Sie es entweder mit den typischen Methoden von Windows 95 schließen oder Sie beenden es, indem Sie das Stop-Symbol auf der Symbolleiste anklicken.

## Erstellen einer neuen Anwendung

Jetzt, wo Sie eine neue Datenbank-Arbeitssitzung erzeugt haben, lassen Sie uns sehen, wie ein neues Anwendungsobjekt erstellt wird. Um das neue Anwendungsobjekt - welches Sie Flugle nennen werden - zu erstellen, können Sie aus den folgenden drei Methoden wählen:

- (Strg) + (N) betätigen
- Wählen des Menüeintrags File | New Application
- Anklicken des Verzeichnis-Symbols auf der Symbolleiste (das Symbol ganz links)

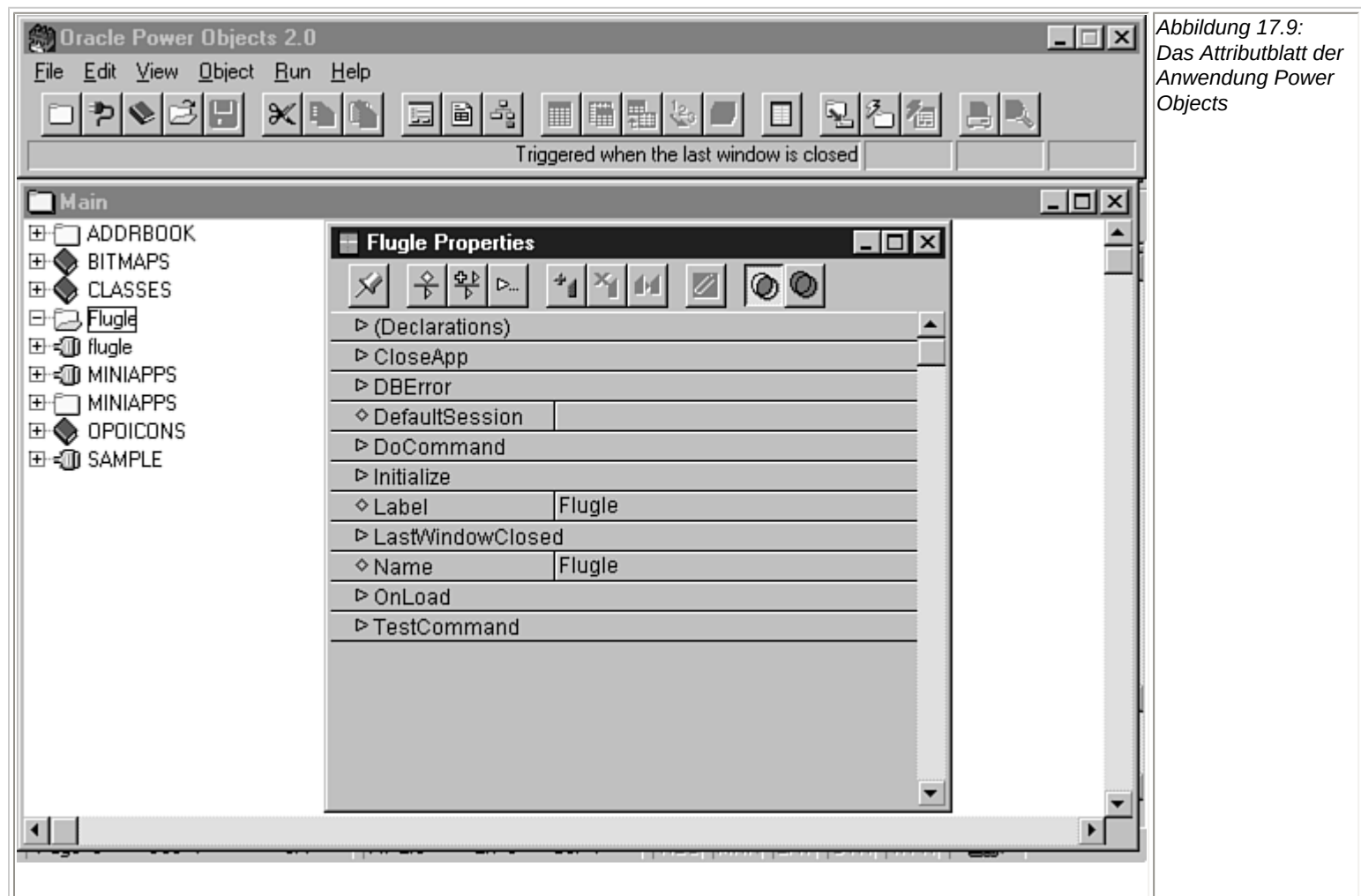
Ein Fenster, betitelt mit Create as, wird erscheinen, welches Sie zur Eingabe des Verzeichnisses auffordert, in dem das Anwendungsobjekt gespeichert werden soll. Als Standard wird Power Objects Ihre Objekte im Power-Objects-Verzeichnis speichern; Sie sollten jedoch Ihre Objekte in einem Verzeichnis speichern, das Sie speziell für diesen Zweck angelegt haben. Geben Sie Flugle in das Feld FileName ein, und klicken Sie auf Save. Sie sehen anschließend das Anwendungsobjekt Flugle im Hauptfenster (Anwendungsobjekte werden durch ein Verzeichnis dargestellt).



Um das Attributblatt für die Anwendung einzusehen, die Sie gerade erstellt haben, wählen Sie die Flugle-Anwendung im Hauptfenster und führen Sie einen der folgenden Schritte aus:

- Klicken mit der rechten Maustaste, um das Attributblatt zu öffnen
- Wählen des Menüeintrags View | Property Sheet
- Klicken auf das Symbol Edit Properties auf der rechten Seite der horizontalen Symbolleiste (sieht aus wie ein weißes Blatt mit einem blauen Rand)

Abbildung 17.9 zeigt das Attributblatt für die Flugle-Anwendung.



## Erstellen eines Formulars

Nachdem Sie das Anwendungsobjekt erzeugt haben, sind Sie bereit, Ihr erstes Formular zu erstellen. Sie werden ein einfaches Formular erstellen, das die Student ID (Studenten-Identifikationsnummer), last name (Nachname), und first name (Vorname) anzeigt, sortiert nach last name und first name. Bevor Sie ein Formular erstellen können, müssen Sie die Anwendung auswählen, zu der es gehören soll. Wählen Sie zu diesem Zweck die Flugle-Anwendung im Hauptfenster, und klicken Sie das Symbol New Form auf der linken Seite der Symbolleiste an (sieht aus wie ein Miniaturformular mit einem blauen Band an der Oberkante). Sie werden ein Fenster mit dem Titel Form1 erscheinen sehen (siehe Abbildung 17.10). Auf der linken Seite des Bildschirms können Sie außerdem die Objektpalette sehen.

Es gibt bestimmte Formulareigenschaften, die Sie üblicherweise festlegen wollen. Um die Eigenschaften des neuen Formulars zu ändern, versichern Sie sich, daß das Formular ausgewählt ist, und klicken Sie dann mit der rechten Maustaste auf den Menüpunkt Property Sheet. Die Eigenschaften-Dialogbox erscheint (siehe Abbildung 17.11). Blättern Sie die Eigenschaftenliste herunter, bis Sie die Eigenschaft RecSrcSession sehen. Geben Sie in das Feld rechts der Bezeichnung RecSrcSession **Flugle** ein. Blättern Sie die Eigenschaftenliste herauf, bis Sie die Eigenschaft RecordSource sehen. Geben Sie in das Feld rechts der Bezeichnung RecordSource **Student** ein; RecordSource kennzeichnet die Tabelle oder Datenansicht, von welcher das Formular seine Datensätze erhält.



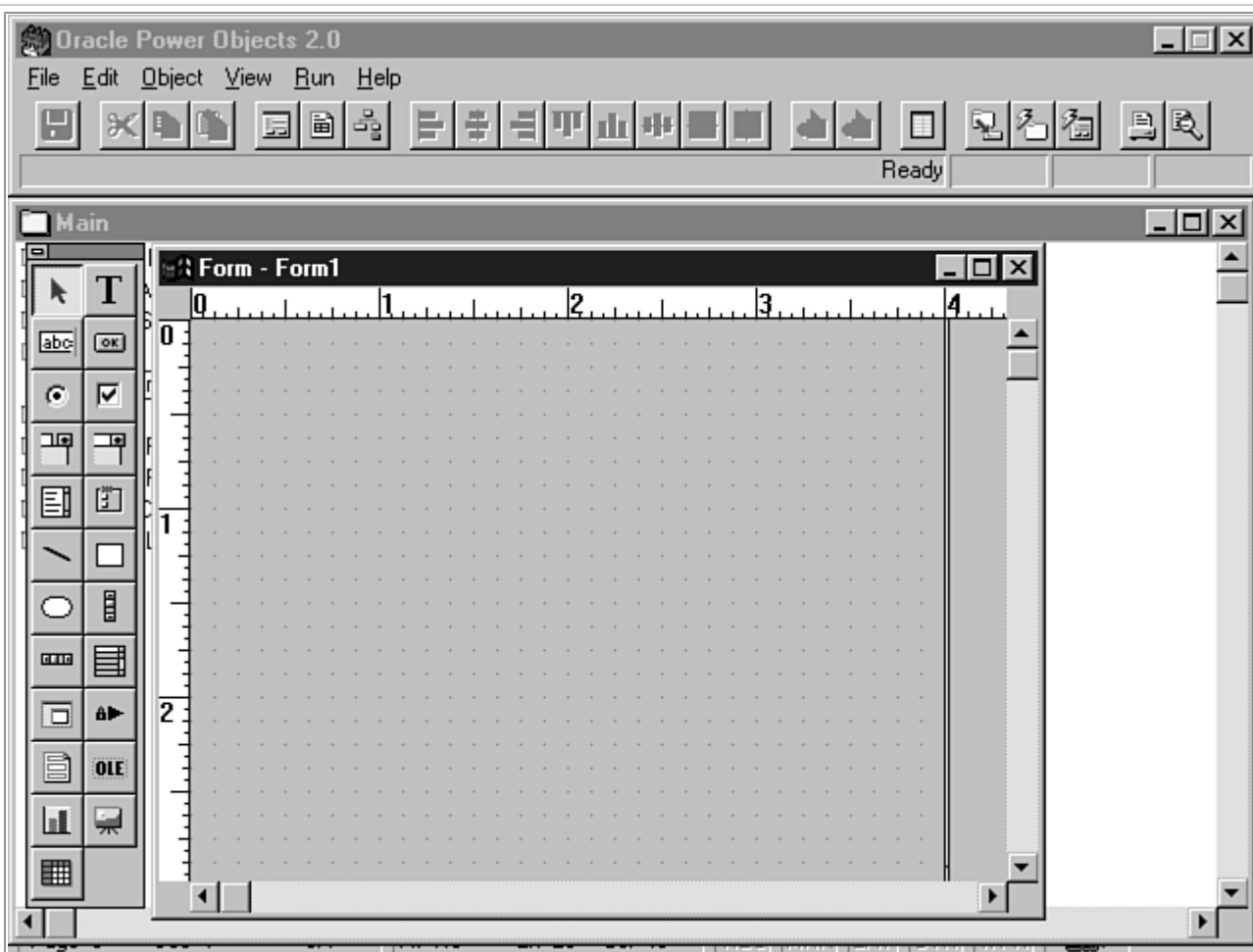


Abbildung 17.10:  
Das neue Formular  
wird angezeigt.

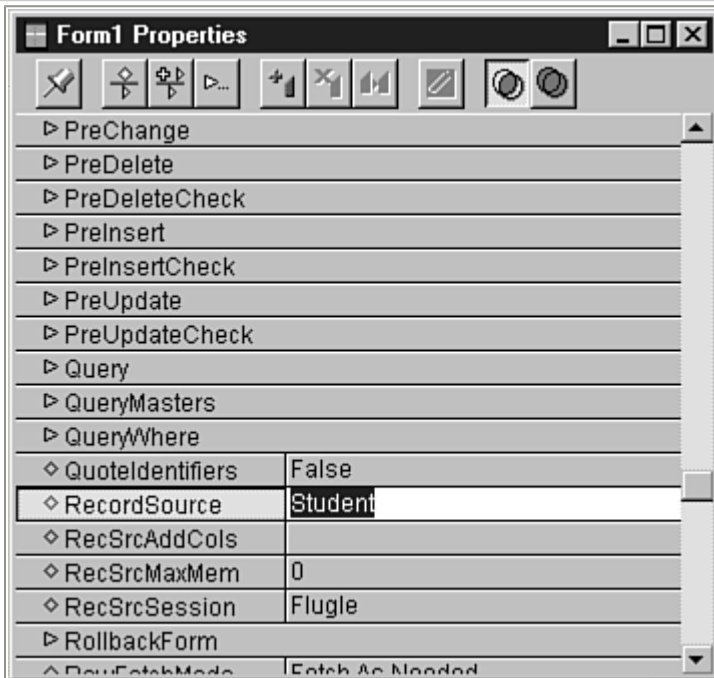


Abbildung 17.11:  
Setzen der Formulareigenschaften  
RecSrcSession und RecordSource.

Das Formular ist jedoch immer noch leer - es enthält keine Objekte. Es gibt drei Kategorien von Objekten, die in einem Formular platziert werden können:

**Container** Dies sind Objekte, die andere Objekte enthalten können, so wie eingebettete Formulare, Auswahllisten, Klassen, OLE-Objekte und Rechtecke/Ellipsen.

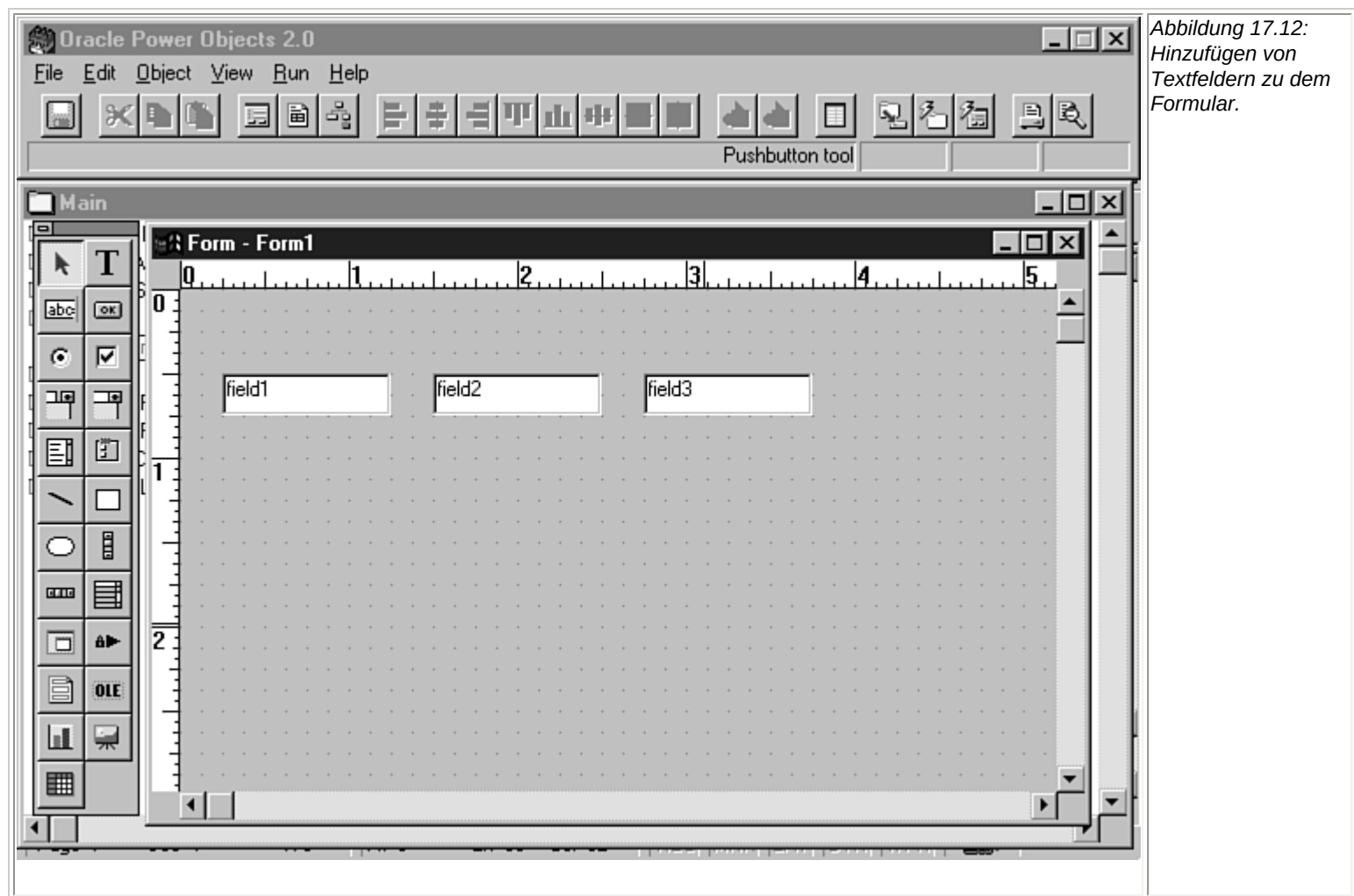
**Statische Objekte** Dies sind grafische Elemente wie Linien, Ellipsen und statischer Text.

Steuerelemente Dies sind Objekte, die entweder von der Datenbank erhalten oder abgeleitet, oder vom Benutzer eingegeben werden. Textfelder, Comboboxen und Pop-up-Listenfelder sind Beispiele für diesen Typ von Steuerelement. Steuerelemente umfassen ebenfalls Objekte, die das Verhalten eines Formulars beeinflussen, so wie Schaltflächen und Bildlaufleisten.

## Hinzufügen von Steuerelementen zu einem Formular

Lassen Sie uns mit Hilfe der Objektpalette dem Formular einige Steuerelemente hinzufügen.

1. Wählen Sie das Textfeld-Symbol aus der Objektpalette (es befindet sich unmittelbar unter dem roten Pfeil-Symbol in der linken oberen Ecke der Palette), und klicken Sie mit der linken Maustaste auf das Formular.
2. Machen Sie dieses dreimal, so daß sich drei Felder auf Ihrem Formular befinden.
3. Richten Sie die Felder so aus, daß das Formular so aussieht, wie in Abbildung 17.12 gezeigt wird.



Diese Textfelder wissen jedoch nichts über die Student-Tabelle; Sie müssen das Attributblatt für jedes Feld anzeigen.

4. Wählen Sie das erste Feld, und klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen.
5. Geben Sie in der Eigenschaft DataSource **Student\_ID** (siehe Abbildung 17.13) ein. Sie sollten ebenfalls den Namen des Feldes ändern; blättern Sie bis zur Eigenschaft Name herunter und geben Sie **Student\_ID** ein. Schließen Sie das Attributblatt für dieses Feld.

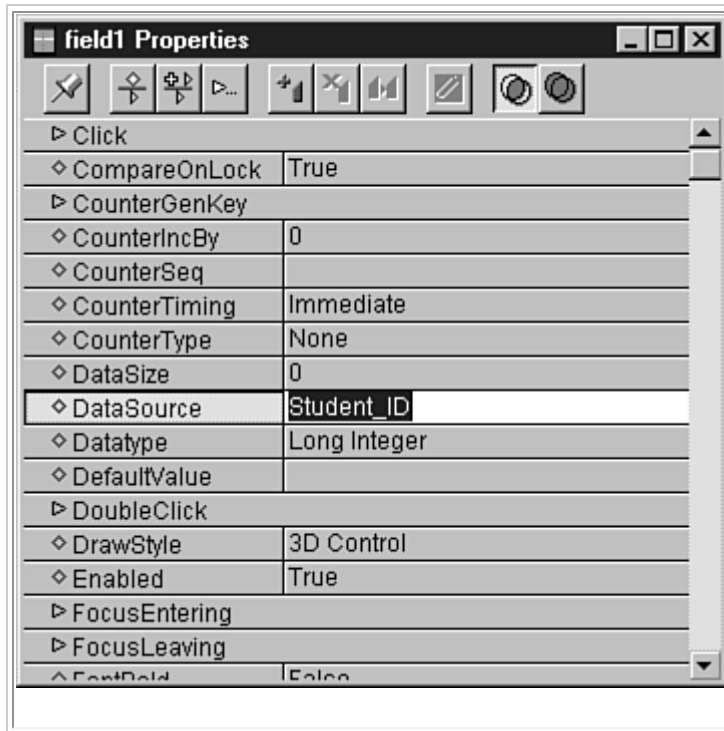


Abbildung 17.13:  
Setzen der Eigenschaft DataSource für  
das Textfeld.

6. Geben Sie für das zweite Feld **Last\_Name** in die beiden Eigenschaften DataSource und Name ein. Setzen Sie die Eigenschaft Datatype auf String.
7. Geben Sie für das dritte Feld **First\_Name** in die beiden Eigenschaften DataSource und Name ein. Setzen Sie, wie im zweiten Feld, die Eigenschaft Datatype auf String.
8. Um Überschriften zu den drei Textfeldern hinzuzufügen, wählen Sie das Objekt Statischer Text aus der Objektpalette (es befindet sich rechts vom roten Pfeil-Symbol).
9. Klicken Sie mit der linken Maustaste über das Student\_ID-Textfeld. Ändern Sie den Text von static1 in Student\_ID.
10. Erstellen Sie auf demselben Weg zwei weitere Überschriften: eine über dem Last\_Name-Textfeld und die andere über dem First\_Name-Textfeld.

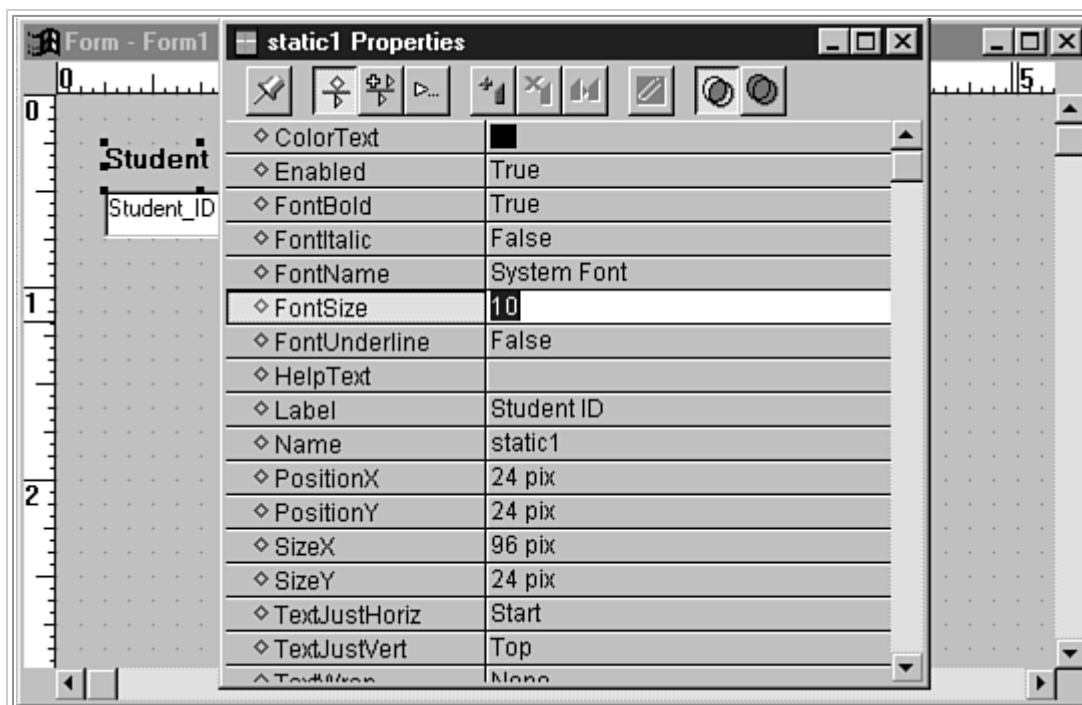
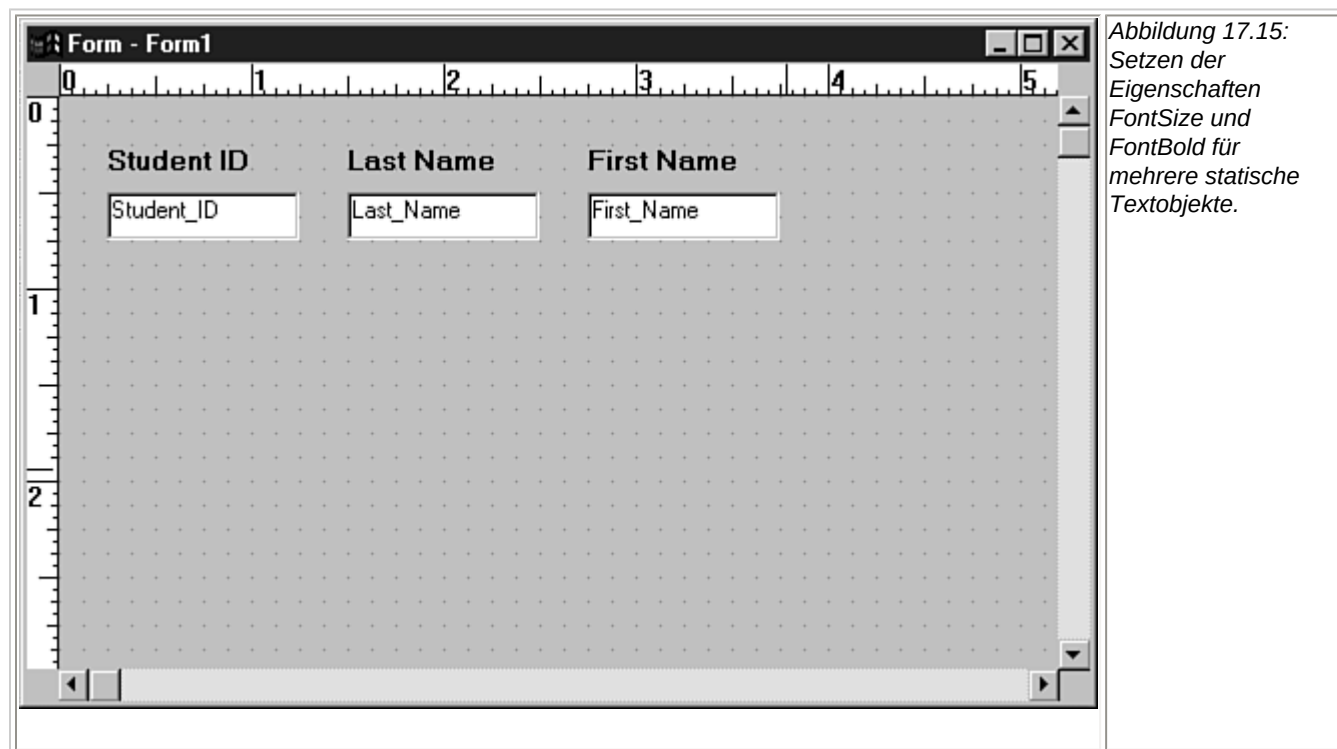


Abbildung 17.14:  
Formular mit  
Textfeldern und  
Beschriftungen

11. Um die Objekte mit statischem Text zu formatieren, wählen Sie sie durch Betätigen der Umschalttaste und Anklicken aller drei

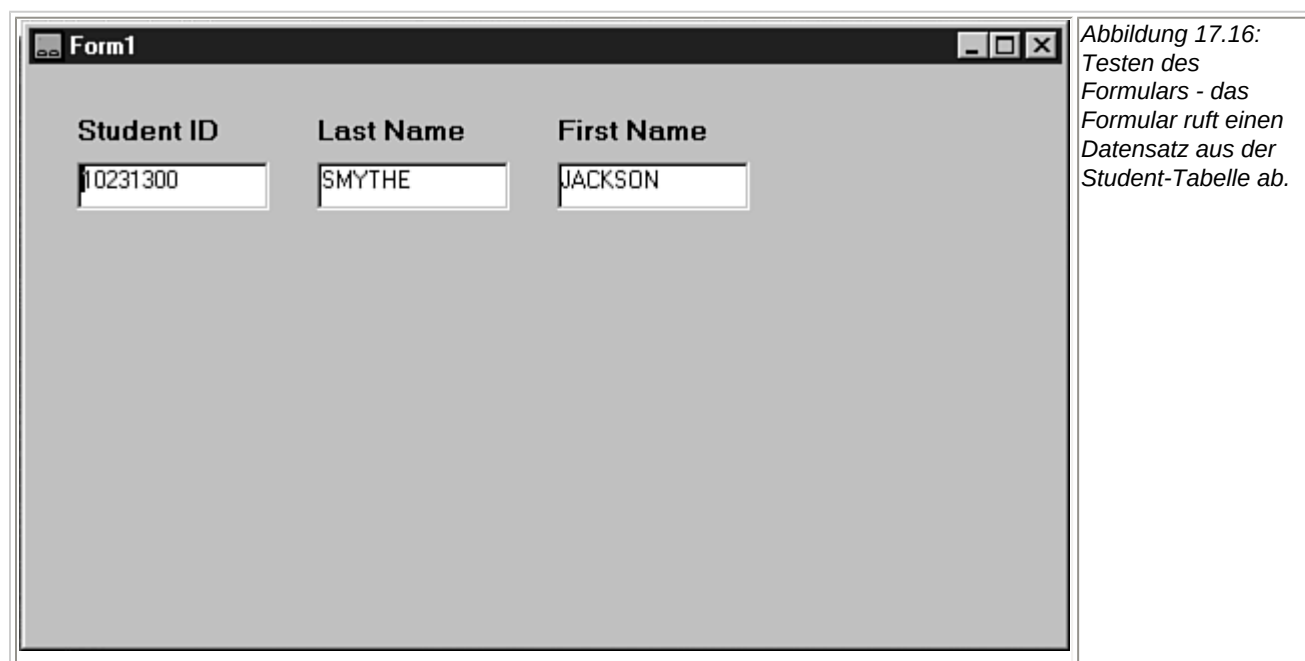
Objekte aus. Klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen; das Attributblatt wird mit einer Mehrfachauswahl von Eigenschaften erscheinen, was bedeutet, daß die Eigenschaften für alle drei Objekte mit statischem Text gelten. Setzen Sie die Eigenschaft **FontSize** auf 10 und die Eigenschaft **FontBold** auf **True** (siehe Abbildung 17.14).

Wenn Sie das Attributblatt-Fenster schließen, sollte Ihr Formular wie das in Abbildung 17.15 gezeigte aussehen.



## Testen des Formulars

Der Power-Objects-Designer bietet eine praktische Methode für das Testen eines einzelnen Formulars (es ist der Methode in Oracle-Formularen sehr ähnlich). Um das gegenwärtige Formular auszuführen, können Sie **Run Form** mit der rechten Maustaste auswählen oder die Befehlsschaltfläche auf der Symbolleiste (ähnelt sinnigerweise einem Blitzstrahl, der ein Formular trifft) betätigen. Wenn Sie irgendwelche Änderungen an dem Formular vorgenommen haben, seit es zum letzten Mal gespeichert wurde, wird Power-Objects-Designer Sie nach Ihrer Zustimmung zur Sicherung der zuletzt gemachten Änderungen fragen. Falls Sie **Yes** anklicken, wird der Power-Objects-Designer das Formular ausführen. Wie Sie in Abbildung 17.16 sehen können, enthalten die Felder einen Datensatz aus der Student-Tabelle. Es gibt jedoch keine Möglichkeit, um zum vorhergehenden oder dem nächsten Datensatz zu gelangen! Beenden Sie den Testmodus, indem Sie entweder die Befehlsschaltfläche **Terminate** (welche einem roten Verkehrszeichen ähnelt) betätigen oder indem Sie das Fenster **Form1** schließen.



## Hinzufügen einer Bildlaufleiste zu dem Formular

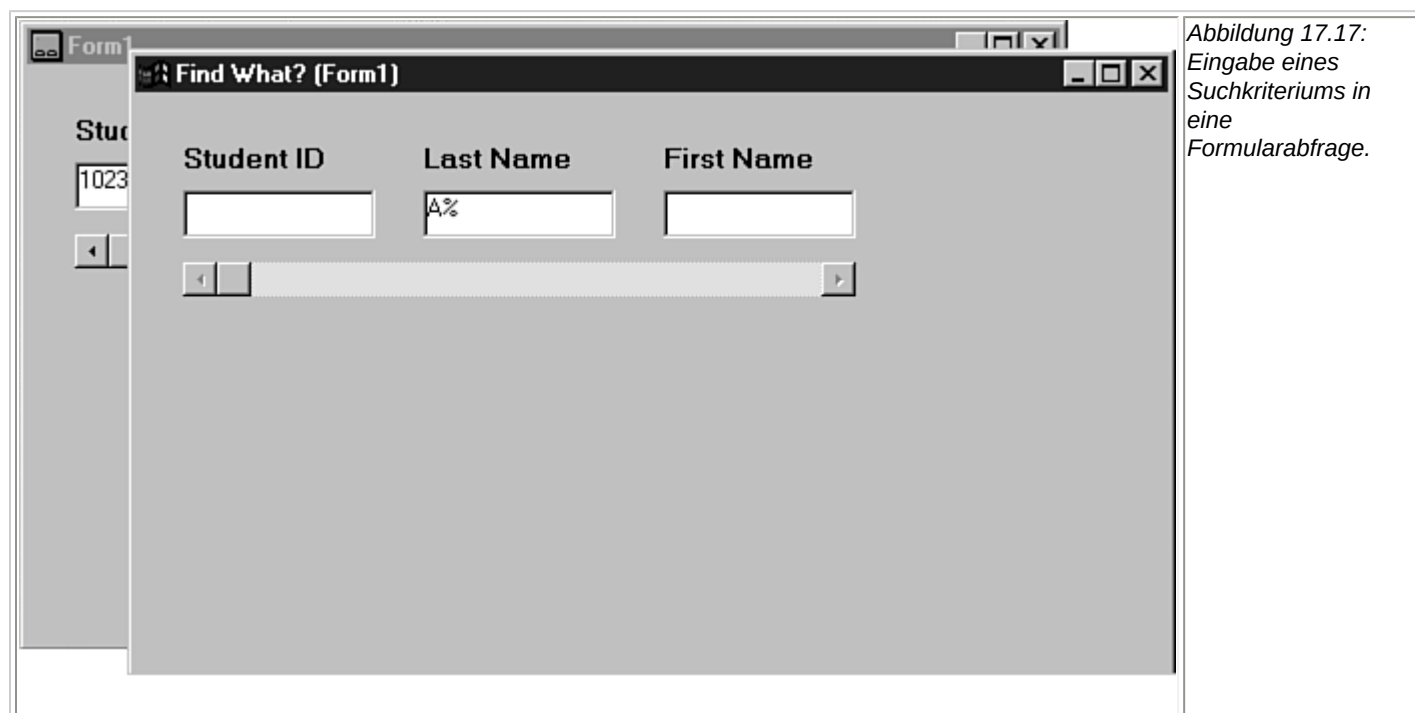
Power Objects bietet einen einfachen Weg, um Navigationsmöglichkeiten zu einem Formular hinzuzufügen. Wählen Sie das Symbol der horizontalen Bildlaufleiste aus der Objektpalette, und klicken Sie mit der linken Maustaste an der Stelle auf das Formular, an der die Bildlaufleiste erscheinen soll. Korrigieren Sie die Position und die Größe nach Ihren Wünschen.

Testen Sie das Formular jetzt erneut durch einen Klick mit der rechten Maustaste auf Run Form. Versuchen Sie, durch Klicken auf das jeweilige Ende der Bildlaufleiste zum nächsten und zum vorherigen Datensatz zu gelangen. Wie Sie sehen können, waren Sie in der Lage, eine Bildlaufleiste zu dem Formular hinzuzufügen, ohne irgendwelche Eigenschaften setzen zu müssen oder Code zu schreiben.

Es gibt einige andere Möglichkeiten, die Power Objects quasi als »Gratiszugabe« bietet; Sie können beispielsweise eine Abfrage durch die Wahl von Database | Query by Form ausführen. Ein anderes Fenster mit dem Titel »Find What?« wird erscheinen. Falls Sie alle Studenten finden wollen, deren Familienname mit einem A beginnt, geben Sie A% in das Last-Name-Feld in dem Suchen-nach-Formular ein (siehe Abbildung 17.17). Um die Abfrage mit dem Suchkriterium auszuführen, wählen Sie Database | Apply Form Query. Sie werden zwei Studenten finden, welche die Kriterien erfüllen - Anna Anastasia und Kenneth Abbot. Sie können ebenfalls durch ein Klicken mit der rechten Maustaste und der Wahl des entsprechenden Menüpunkts einen Datensatz aktualisieren, hinzufügen und löschen. Verlassen Sie das Formular, um in den Entwurfsmodus zurückzukehren.

## Die Abrufreihenfolge für das Formular festlegen

Sie werden die Reihenfolge bestimmen wollen, in welcher die Datensätze gelesen und im Formular dargestellt werden. Wie Sie sich erinnern, soll das Formular die Datensätze aus der Student-Tabelle nach Familien- und Vornamen sortiert wiedergeben. Wählen Sie das Formular, und klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen. Verschieben Sie den Bildlauf bis zur Eigenschaft OrderBy, und geben Sie Last\_Name, First\_Name ein. Versuchen Sie einmal mehr einen Test des Formulars. Die Datensätze sollten in der Reihenfolge angezeigt werden, die Sie festgelegt haben: Last\_Name, First\_Name.



## Benennen des Formulars

Anders als Oracle-Formulare speichert Power Objects ein Formular nicht in einer eigenen Datei; ein Formular wird als ein Teil des Anwendungsobjekts gespeichert. Um den Namen des Formulars in Student zu ändern, setzen Sie die Eigenschaft Label auf Student und die Eigenschaft Name auf frmStudent.

## Erstellen eines Master-/Detail-Formulars

Das Erstellen eines Master-/Detail-Formulars in Power Objects ist einfach. Lassen Sie uns als ein Beispiel das Department-Class-Formular erstellen, das Sie unter Verwendung von Oracle Forms erzeugt haben. Zunächst erstellen Sie ein neues Formular im Flugle-Anwendungsobjekt.

Erzeugen Sie zuerst die Objekte, die mit der Haupttabelle Department in Verbindung stehen.

1. Wählen Sie das Formular durch Anklicken, und klicken Sie dann mit der rechten Maustaste, um das Attributblatt zu öffnen.
2. Setzen Sie diese Eigenschaften für das Formular:

```
Name frmDepartment_Class
RecSrcSession Flugle
RecordSource Department
OrderBy Department_ID
```

3. Beenden Sie das Attributblatt des Formulars.

4. Wählen Sie das Textfeld-Symbol in der Objektpalette und platzieren Sie es im oberen Bereich des Formulars.

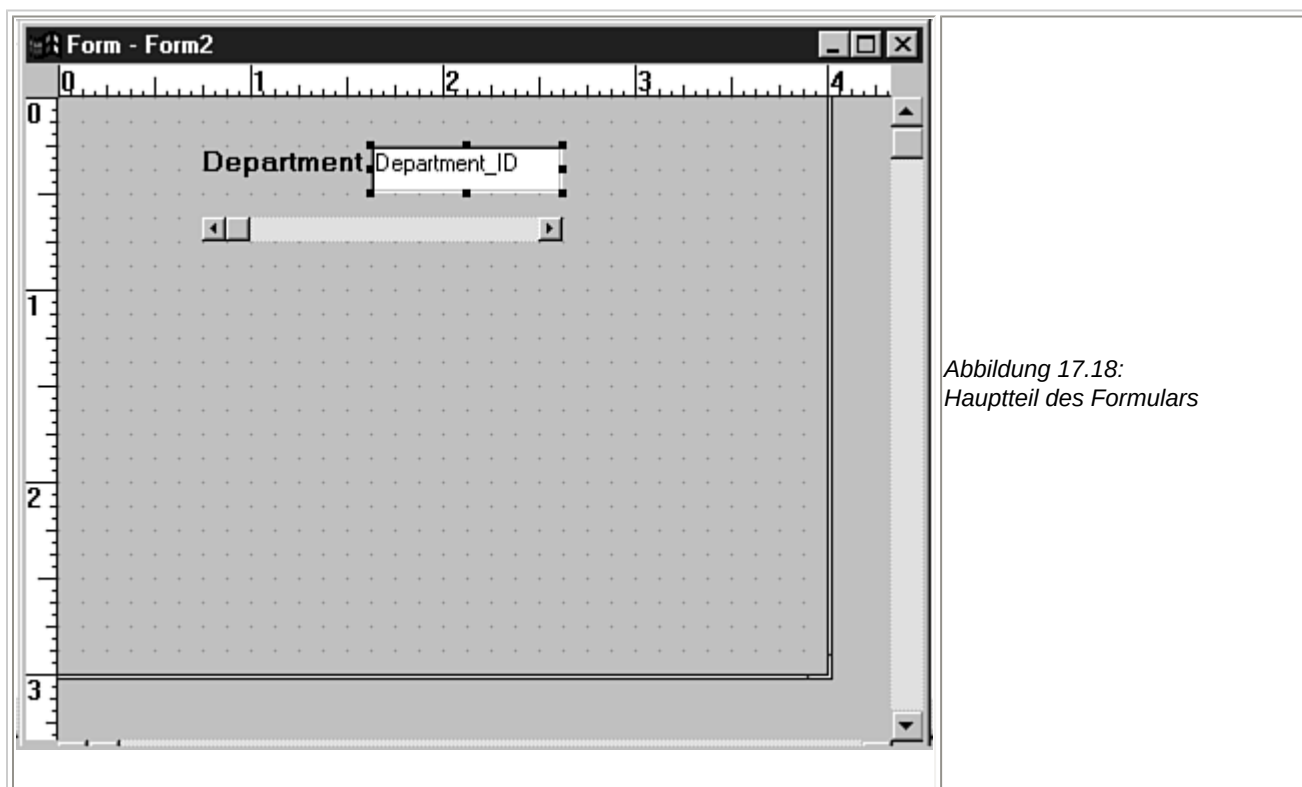
5. Klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen und setzen Sie die Eigenschaften DataSource und Name auf Department\_ID. Sie müssen außerdem die Eigenschaft Datatype auf String setzen (als Standard ist sie auf eine Long Integer gesetzt).

6. Schließen Sie das Attributblatt des Textfeldes.

7. Wählen Sie das Objekt Statischer Text aus der Objektpalette, und platzieren Sie eine Textbeschriftung links vom Department\_ID-Textfeld.

8. Setzen Sie den Namen der Textbeschriftung auf Department; ändern Sie die Eigenschaft FontBold in True und die Eigenschaft FontSize auf 10.

9. Zuletzt benötigen Sie eine Bildlaufleiste, um zwischen den Departments wechseln zu können. Wählen Sie die horizontale Bildlaufleiste aus der Objektpalette, und erstellen Sie die horizontale Bildlaufleiste auf dem Formular genau unter dem Department\_ID-Textfeld.



An dieser Stelle sollte Ihr Formular so aussehen wie das in Abbildung 17.18 gezeigte.

Als nächstes werden Sie den detaillierteren Teil des Formulars erstellen. Wählen Sie das Werkzeug Auswahlliste in der Objektpalette - es befindet sich in der rechten Spalte, das vierte von unten, und sieht aus wie ein graues Formular mit weißen Rändern. Plazieren Sie die Auswahlliste auf dem Formular unterhalb des Department\_ID-Textfelds. Die Auswahlliste wird benutzt, um die Datensätze aus der Tabelle Class anzuzeigen, die mit der gegenwärtig angezeigten Department\_ID korrespondieren. Eine Auswahlliste ist ein Behälter, der aus diesen Elementen besteht:

**Rahmen** Dies ist das Objekt, welches das gesamte Objekt einschließt.

**Primärer Bereich** Dies ist der Bereich im oberen Abschnitt der Auswahlliste, in welchem Sie andere Steuerelemente und statische Objekte platzieren. Der primäre Bereich ist typischerweise der Ort, an dem Sie die Textfelder positionieren, die in der Auswahlliste angezeigt werden.

**Sekundärer Bereich** Dieses sind die Bereiche, welche die zusätzlichen Instanzen von Objekten anzeigen, die im primären Bereich enthalten sind.

**Bildlaufleiste** Die Auswahlliste enthält eine vertikale Bildlaufleiste an ihrer rechten Seite.

Wenn Sie die Auswahlliste einmal auf dem Formular platziert haben, öffnen Sie mit einem Klick auf die rechte Maustaste das Attributblatt. Es gibt mehrere Eigenschaften, die Sie für eine einwandfreie Funktion der Auswahlliste setzen müssen:

**RecordSource Class** (dies ist die Tabelle, von der die Datensätze bezogen werden)

**RecSrcSession Flugle** (dies ist die Datenbank-Arbeitssitzung, die für den Zugriff auf die Datenbank benutzt wird)

**LinkDetailColumn Department\_ID** (dies ist der Name der Spalte in dem Detail-Recordset, welche für den Link auf den Master-Recordset benutzt wird)

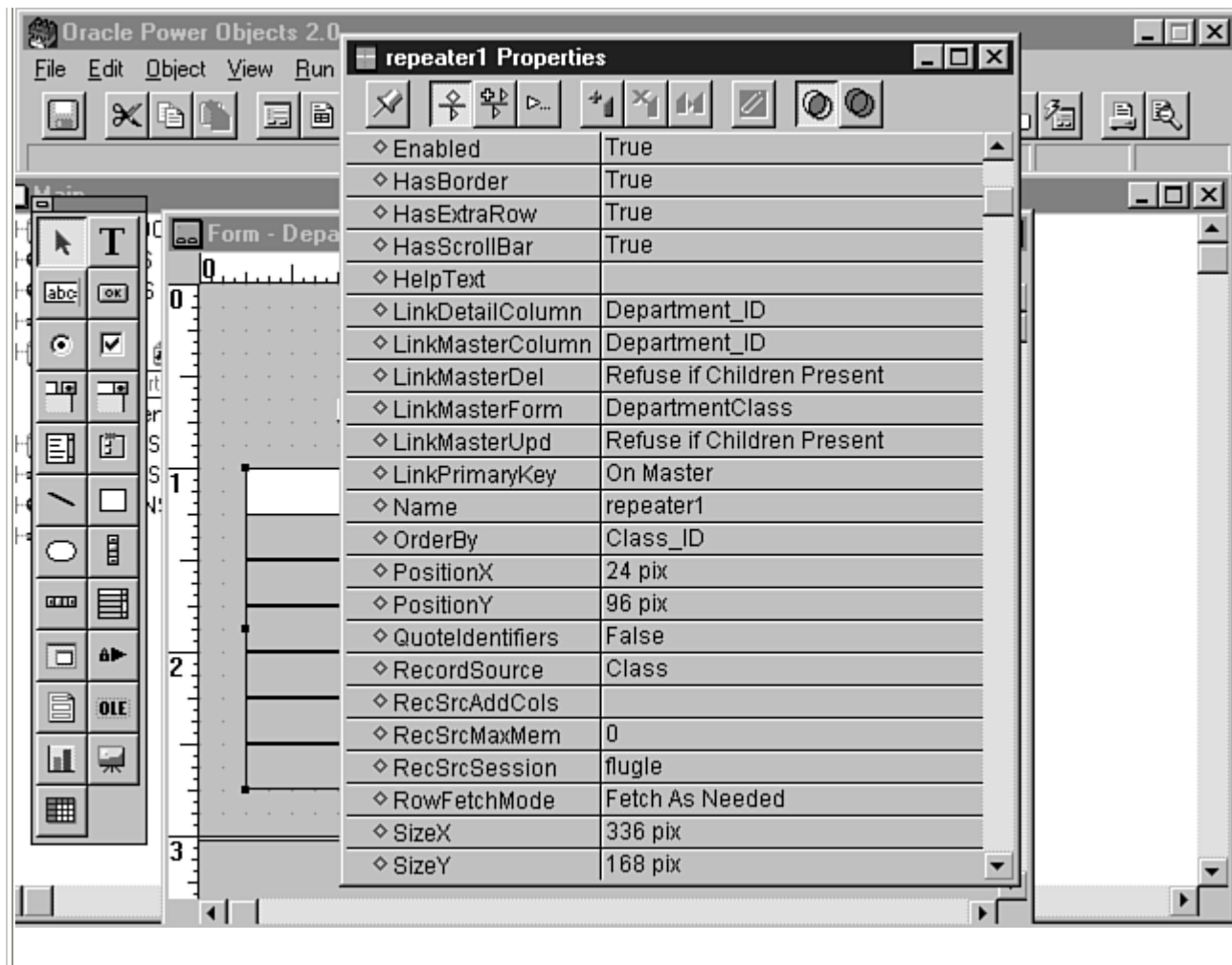
**LinkMasterColumn Department\_ID** (dies ist der Name der Spalte in dem Master-Recordset, zu dem der Detail-Recordset gelinkt wird)

**LinkMasterForm DepartmentClass** (dies ist der Name des Formulars, welches die Master-Datensätze enthält)

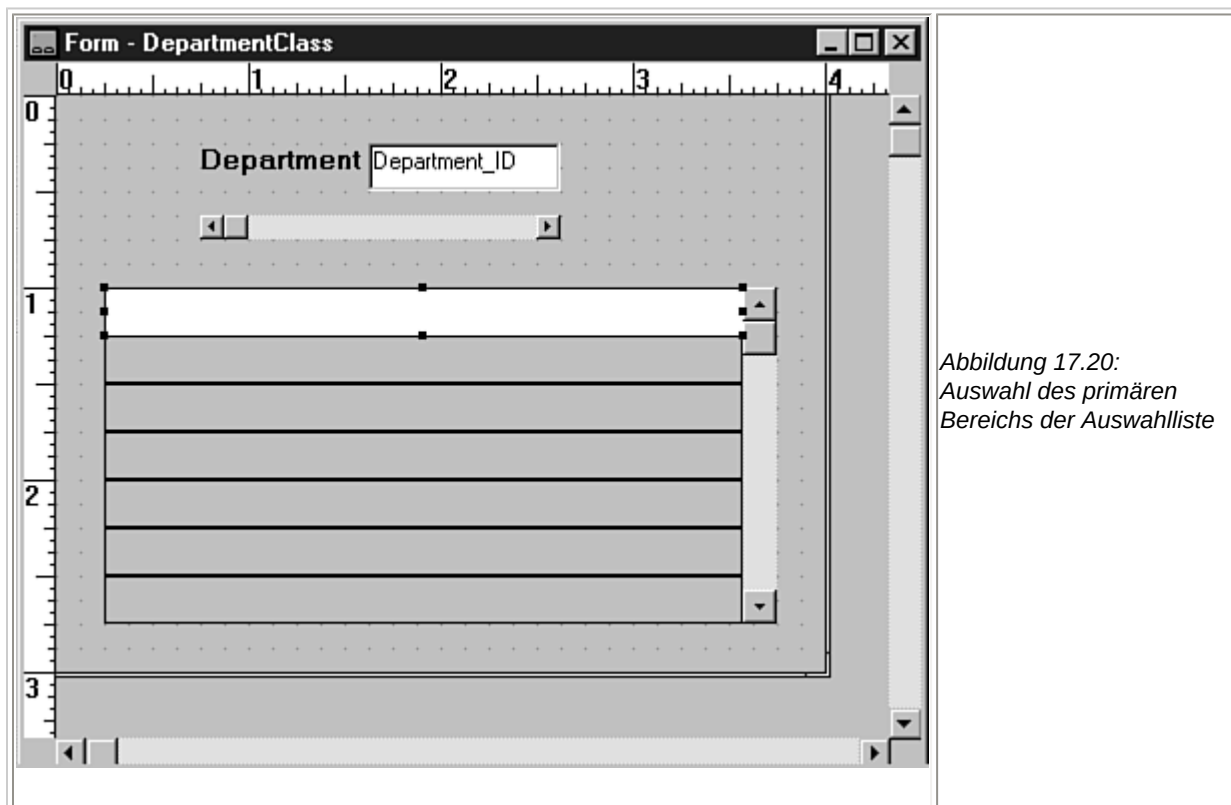
**OrderBy Class\_ID** (die Spalte/Spalten, welche zur Sortierung der Detail-Datensätze genutzt werden)

Abbildung 17.19 zeigt das Attributblatt für die Auswahlliste. Schließen Sie dieses Attributblatt.

*Abbildung 17.19:  
Attributblatt für die  
Auswahlliste*



Als nächstes müssen Sie die Textfelder dem primären Bereich der Auswahlliste hinzufügen. Wählen Sie den primären Bereich, wie in Abbildung 17.20 gezeigt wird.



Wählen Sie das Objekt Textfeld aus der Objektpalette, und erstellen Sie ein Textfeld durch Klicken innerhalb des primären Bereichs der Auswahlliste. Erzeugen Sie insgesamt sechs Felder. Machen Sie für jedes Textfeld folgendes:



- Setzen Sie die Eigenschaft DataSource auf den Namen der Spalte, die Sie in dem Textfeld anzeigen wollen.
- Setzen Sie die Eigenschaft Datatype auf den entsprechenden Wert für den Datentyp der Spalte.
- Setzen Sie die Eigenschaft Name auf den Namen der Spalte.
- Setzen Sie die Eigenschaft FontName auf Ms Sans Serif.

Abbildung 17.21 zeigt als ein Beispiel das Attributblatt für das Textfeld Class\_ID.

Nachdem Sie einmal die Eigenschaften für jedes der Textfelder im primären Bereich der Auswahlliste festgelegt haben, fügen Sie eine Überschrift zu jeder Spalte hinzu, indem Sie das Objekt Statischer Text auf der Objektpalette auswählen und dieses Objekt über dem Textfeld der Auswahlliste platzieren. Wenn Sie das alles beendet haben, testen Sie das Formular durch Klicken mit der rechten Maustaste auf Run Form. Abbildung 17.22 zeigt, wie Ihr Formular aussehen sollte, wenn es die Klassen für das Department Psychologie anzeigt. Beenden Sie den Testmodus und öffnen Sie das Student-Formular für die nächste Übung.

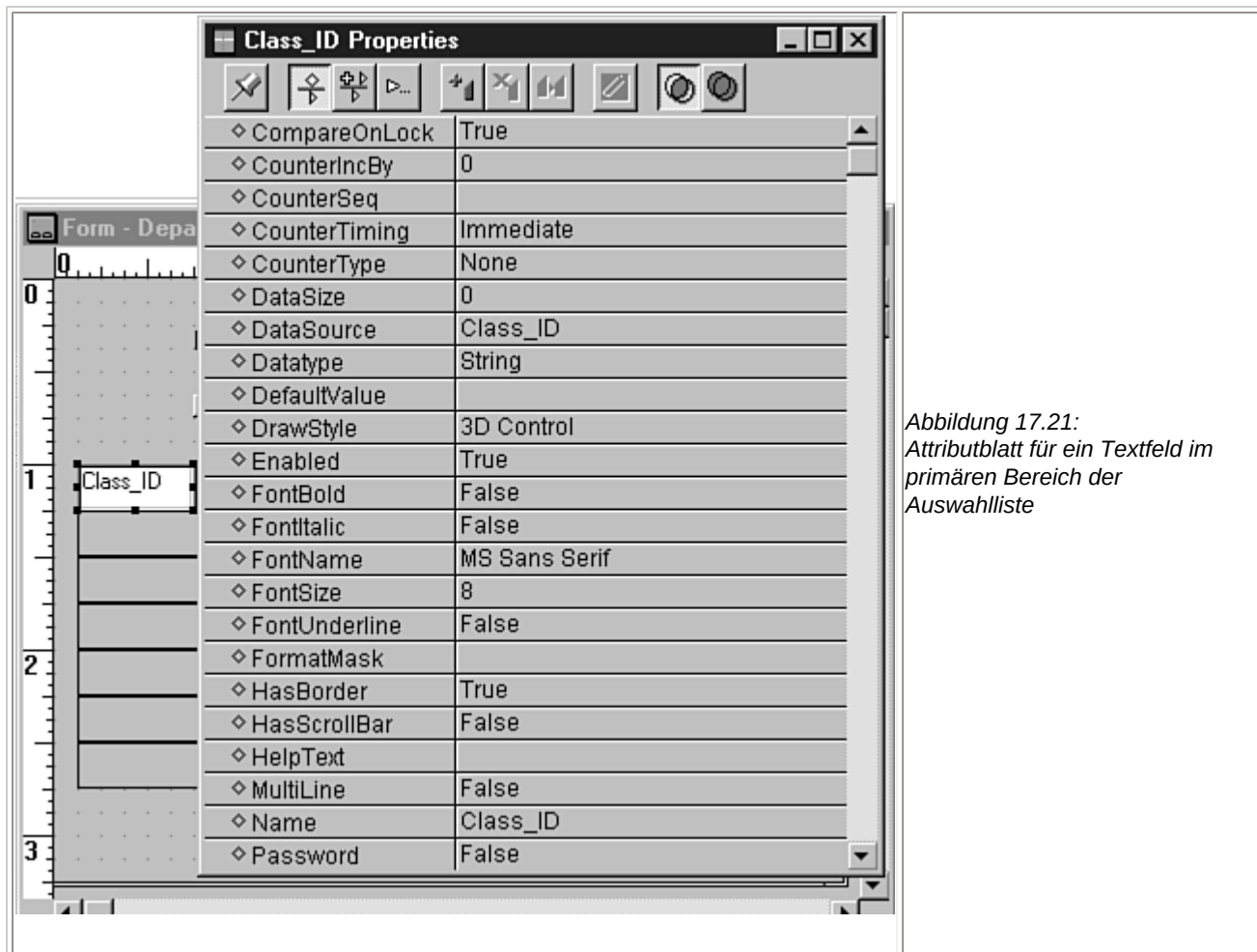


Abbildung 17.21:  
Attributblatt für ein Textfeld im  
primären Bereich der  
Auswahlliste

| Class ID | Course | Schedule | Building     | Room | Instructor ID |
|----------|--------|----------|--------------|------|---------------|
| 103400   | 183    | T13      | FLUGLE HALL  | 150  | J505          |
| 108600   | 183    | T10      | VILING TOWER | 240  | J505          |
|          |        |          |              |      |               |

Abbildung 17.22:  
Testen des Master-/Detail-  
Formulars

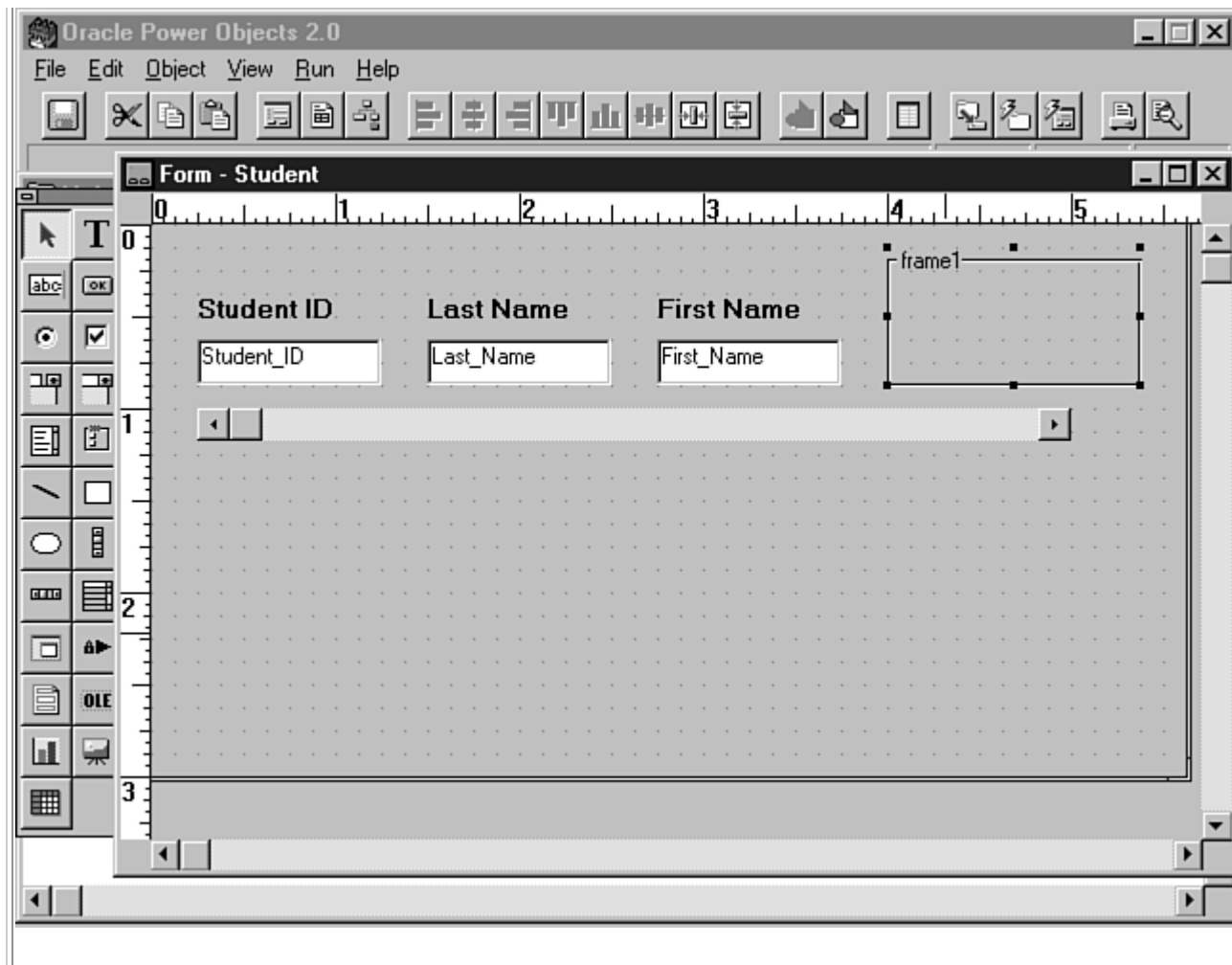
## Verwenden von Optionsschaltflächen

Lassen Sie uns einen Blick auf zwei andere Objekte werfen, die Sie in Ihre Formulare integrieren können: das Optionsgruppenfeld und die Optionsschaltfläche. Ein Optionsgruppenfeld enthält üblicherweise zwei oder mehr Optionsschaltflächen. Stellen Sie sich zum Beispiel vor, Sie wollen das Student-Formular verändern, indem Sie das Studienjahr als eine Gruppe von Optionsschaltflächen anzeigen wollen.

Der erste Schritt ist das Hinzufügen eines Optionsgruppenfeldes zum Formular.

1. Wählen Sie das Symbol des Optionsgruppenfeldes aus der Objektpalette (es befindet sich in der rechten Spalte, das fünfte von oben).
2. Klicken Sie mit der linken Maustaste an der Stelle auf das Formular, an der Sie das Feld platzieren wollen (siehe Abbildung 17.23).

Abbildung 17.23:  
Hinzufügen eines  
Optionsgruppenfeldes  
zu einem Formular.



3. Ist das Optionsgruppenfeld erst einmal erzeugt, können Sie es durch Anklicken einer Ecke und mit anschließendem Ziehen in der Größe verändern.

4. Richten Sie es mit den anderen Feldern auf dem Formular aus. öffnen Sie durch Klicken mit der rechten Maustaste das Attributblatt, um die Eigenschaften für das Optionsgruppenfeld festzulegen. Sie müssen genau die folgenden Eigenschaften setzen:

DataSource **Year** (die Spalte, welche durch die Optionsschaltflächen dargestellt wird, die im Optionsgruppenfeld enthalten sind)

DataType **String**  
 FontBold **True**  
 FontName **MS Sans Serif**  
 FontSize **10**  
 Label **Year**  
 Name **StudentYear**

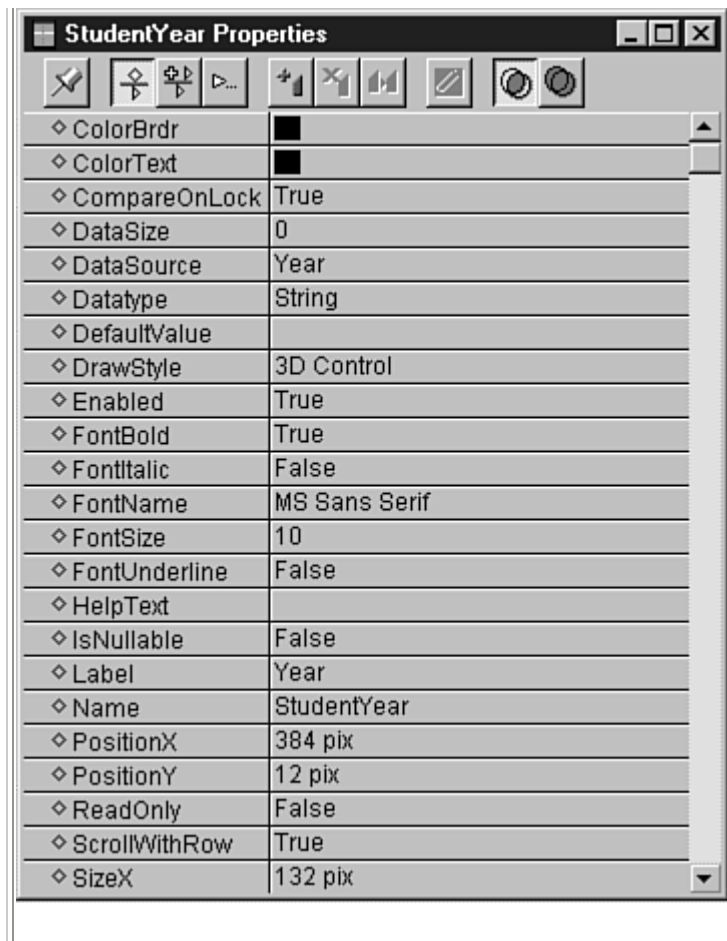


Abbildung 17.24:  
Festlegen der Eigenschaften für das  
Optionsgruppenfeld.

5. Wenn Sie alle Eigenschaften gesetzt haben, beenden Sie das Attributblatt.



Sie können das Optionsgruppenfeld nicht »Year« nennen, weil dies ein für Oracle Basic reserviertes Wort ist.

Der nächste Schritt ist das Hinzufügen von vier Optionsschaltflächen zum Optionsgruppenfeld - Freshman (Erstsemester), Sophomore (Student im zweiten Semester), Junior (Student im vorletzten Semester) und Senior (Student im letzten Semester). Um ein Optionsfeld einzufügen, müssen Sie das entsprechende Symbol aus der Objektpalette wählen (es befindet sich in der linken Spalte, das dritte von oben). Positionieren Sie den Zeiger innerhalb des Optionsgruppenfeldes, und klicken Sie mit der linken Maustaste. Ihr Formular sollte wie das in Abbildung 17.25 gezeigte aussehen.

Sie müssen die Eigenschaften für die Optionsschaltflächen durch einen Klick mit der linken Maustaste auf Properties setzen. Legen Sie die Eigenschaften des ersten Optionsfeldes folgendermaßen fest:

DataSource **Year**

Datatype **String**

Label **Freshman**

Name **Freshman**

ValueOn **FRESHMAN** (diese Eigenschaft enthält den Wert, der sich zu dem Zeitpunkt in der Spalte befindet, an dem die Optionsschaltfläche gesetzt wird)

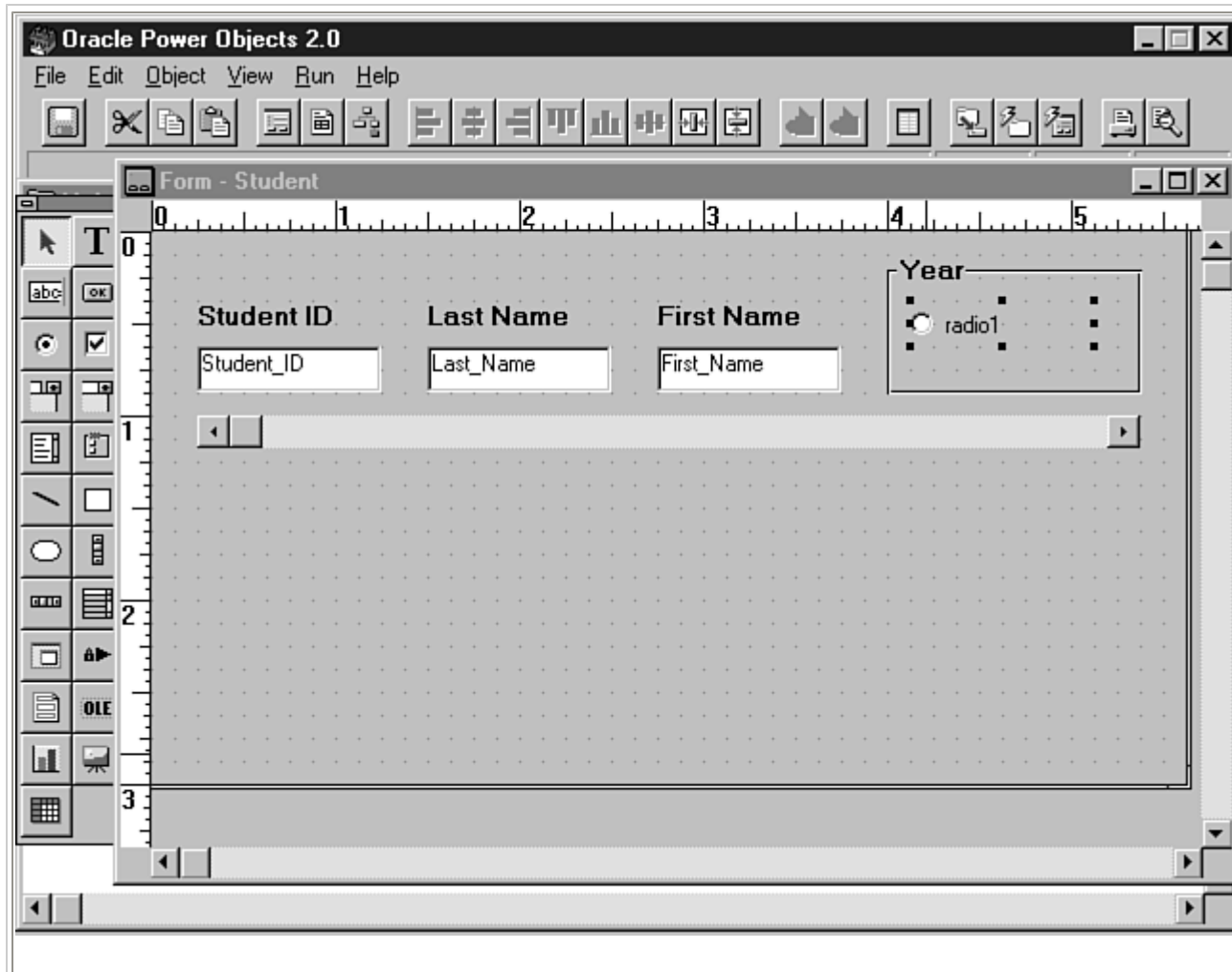


Abbildung 17.25:  
Einfügen eines  
Optionsfeldes in das  
Optionsgruppenfeld.

Abbildung 17.26 zeigt das Attributblatt für die Optionsschaltfläche. Wenn Sie das Setzen der Eigenschaften beendet haben, schließen Sie das Attributblatt. Führen Sie dieselben Schritte des Hinzufügens von Optionsfeldern für Sophomore, Junior und Senior aus. Wenn Sie das abgeschlossen haben, sollte Ihr Formular ähnlich dem in Abbildung 17.27 gezeigten aussehen.

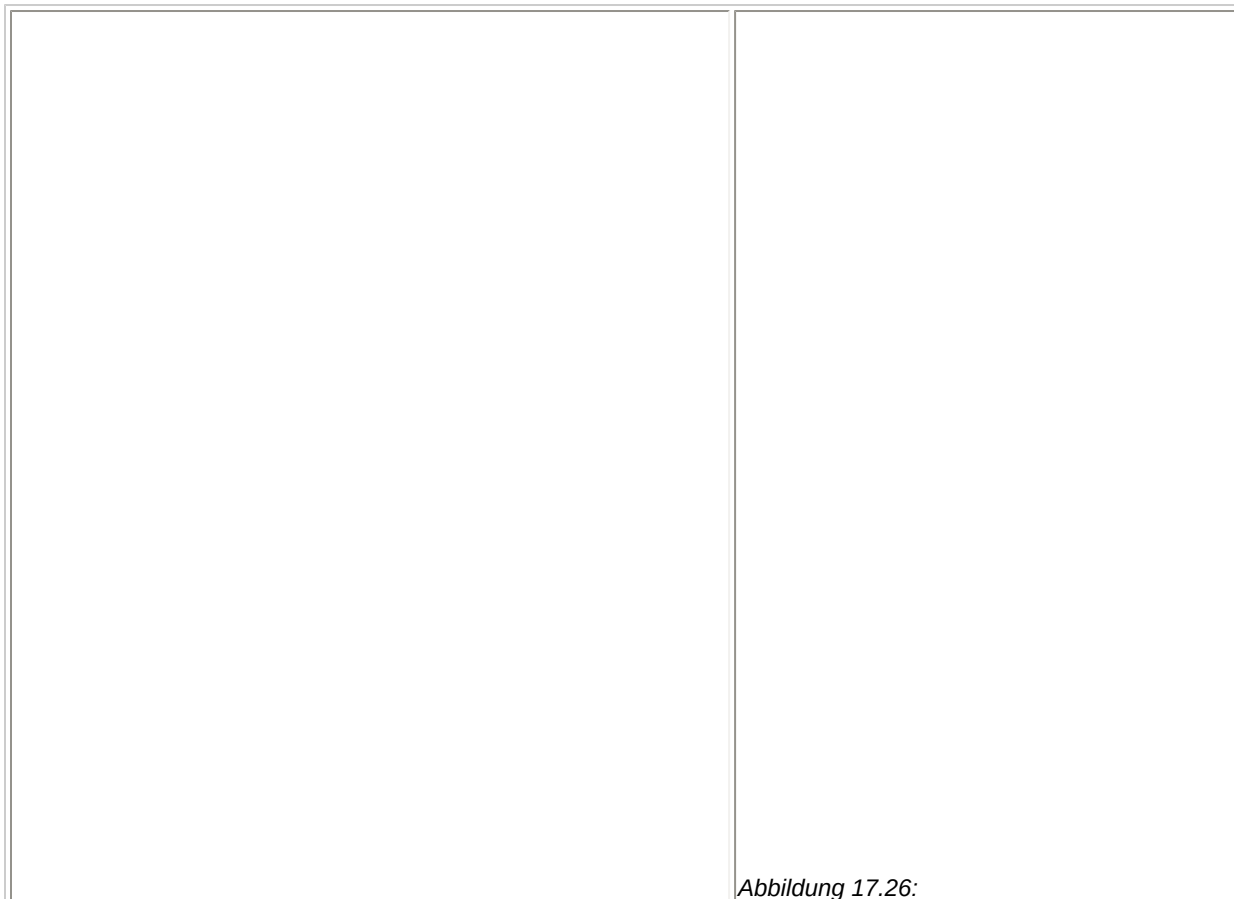


Abbildung 17.26:

| Property      | Value       |
|---------------|-------------|
| DataSource    | Year        |
| Datatype      | String      |
| DefaultValue  |             |
| DrawStyle     | 3D Control  |
| Enabled       | True        |
| FontBold      | False       |
| FontItalic    | False       |
| FontName      | System Font |
| FontSize      | 8           |
| FontUnderline | False       |
| HelpText      |             |
| Label         | Freshman    |
| Name          | Freshman    |
| PositionX     | 12 pix      |
| PositionY     | 24 pix      |
| ReadOnly      | False       |
| ScrollWithRow | True        |
| SizeX         | 96 pix      |
| SizeY         | 24 pix      |
| TabEnabled    | True        |
| TabOrder      | 1           |
| ValidateMsg   |             |
| ValueOff      |             |
| ValueOn       | FRESHMAN    |

Setzen der Eigenschaften für die Optionsschaltfläche Freshman.

**Form - Student**

Student ID:  Last Name:  First Name:

**Year**

☒ Freshman  
☐ Sophomore  
☐ Junior  
☐ Senior

Abbildung 17.27: Student-Formular mit einem Optionsgruppenfeld und vier Optionsschaltflächen.

Führen Sie einen Test mit dem geänderten Student-Formular durch. Wie Sie in Abbildung 17.28 sehen können, wird das Studienjahr mit der richtigen Optionsschaltfläche angezeigt.

Abbildung 17.28:  
Testen des  
Formulars mit dem  
Optionsgruppenfeld  
und den vier  
Optionsschaltflächen.

## Anzeige eines Suchfeldes

Sie werden oftmals ein Suchfeld zu einem Formular hinzufügen müssen. Power Objects bietet eine eingebaute Funktion namens `SQLLOOKUP`, die zur Ausführung einer Abfrage benutzt wird und eine einzelne Spalte zurückgibt. Lassen Sie uns als Beispiel einen Blick auf das `DepartmentClass`-Formular werfen. Stellen Sie sich vor, Sie wollen zusätzlich zur `Department_ID` auch die Zahl der Kurse anzeigen, die das Department anbietet.

1. Fügen Sie ein Textfeld zum Formular hinzu (siehe Abbildung 17.29).

Abbildung 17.29:  
Hinzufügen eines  
Textfeldes, das  
einen Suchwert  
anzeigt.

2. Setzen Sie die Eigenschaften für das neue Textfeld durch Klicken mit der rechten Maustaste, um das Attributblatt zu öffnen. Sie sollten diese Eigenschaften wie in Abbildung 17.30 gezeigt setzen:

`DataSource =sqllookup(flugle, "select count(*) from course where department_id = ' " + department_id.value + "'")`

Datatype **Long Integer**Name **NumberOfCourses**ReadOnly **True**

Wie Sie sehen können, enthält die Funktion SQLLOOKUP zwei Argumente. Das erste Argument ist die Datenbank-Arbeitssitzung, und das zweite ist die Anweisung SELECT, welche durch die Arbeitssitzung ausgeführt wird. In diesem Beispiel gibt die SELECT-Anweisung die Zahl der Kurse aus der Kurs-Tabelle zurück, in welcher die Department\_ID-Spalte gleich dem Wert ist, der zu diesem Zeitpunkt im Department\_ID-Textfeld angezeigt wird.



*Wenn Sie die Eigenschaft DefaultSession für eine Anwendung setzen, müssen Sie die Datenbank-Arbeitssitzung nicht angeben, wenn Sie SQLLOOKUP aufrufen, solange Sie nicht eine andere Arbeitssitzung benutzen wollen.*

3. Testen Sie das Formular. Wie Sie in Abbildung 17.31 sehen können, zeigt das NumberOfCourses-Feld den korrekten Wert; falls Sie den Inhalt der Kurs-Tabelle prüfen wollen, werden Sie sehen, daß vom Department Anthropologie drei Kurse angeboten werden. Der Detailbereich des Formulars ist jedoch leer, weil keiner der anthropologischen Kurse in diesem Semester in Form einer Klasse angeboten wird.

| Property      | Value                           |
|---------------|---------------------------------|
| CompareOnLock | True                            |
| CounterIncBy  | 0                               |
| CounterSeq    |                                 |
| CounterTiming | Immediate                       |
| CounterType   | None                            |
| DataSize      | 0                               |
| DataSource    | =sqllookup(flugle,"select count |
| Datatype      | Long Integer                    |
| DefaultValue  |                                 |
| DrawStyle     | 3D Control                      |
| Enabled       | True                            |
| FontBold      | False                           |
| FontItalic    | False                           |
| FontName      | System Font                     |
| FontSize      | 8                               |
| FontUnderline | False                           |
| FormatMask    |                                 |
| HasBorder     | True                            |
| HasScrollBar  | False                           |
| HelpText      |                                 |
| MultiLine     | False                           |
| Name          | NumberOfCourses                 |
| Password      | False                           |
| PasswordChar  | *                               |

Abbildung 17.30:  
Setzen der Eigenschaften für das Suchfeld.



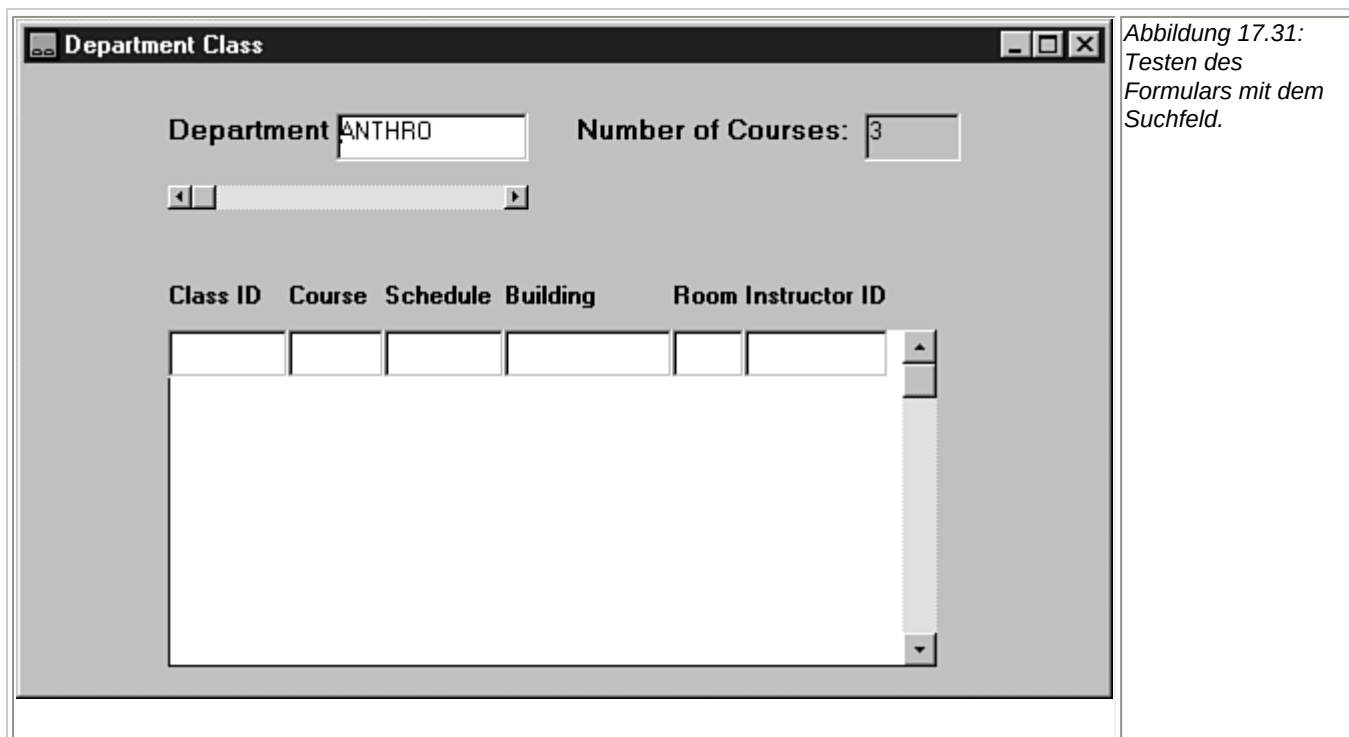


Abbildung 17.31:  
Testen des  
Formulars mit dem  
Suchfeld.

## Zusammenfassung

Die in dieser Lektion hauptsächlich gezeigten Sachverhalte sind die folgenden:

- Power Objects ist ein Entwicklungswerkzeug für Anwendungen, das von den Windows- und Macintosh-Plattformen unterstützt wird.
- Power Objects ist mehr mit Visual Basic und PowerBuilder verwandt als mit Oracle-Formularen. Seine Objektorientierung zeigt sich durch die Fähigkeit zur Erzeugung neuer Klassen und der Vererbung aus einer Klasse.
- Power Objects benutzt Oracle Basic als Skriptsprache zur Festlegung der für ein Objekt definierten Methoden.
- Auf dem höchsten Level verwaltet Power Objects drei Typen von Objekten: Anwendungen, Arbeitssitzungen und Bibliotheken.
- Eine Anwendung besteht aus Formularen, Berichten und Klassen.
- Power Objects bietet ein Attributblatt für jeden Typ von Objekt, der unterstützt wird.
- Power Objects benutzt zwei Modi: Entwurfsmodus und Ausführungsmodus. Zur Entwurfszeit können Sie die Objekte zusammensetzen, welche die Anwendung bilden. Zur Laufzeit können Sie das Verhalten eines Formulars testen.

## Wie geht es weiter?

An Tag 18, »Eine Anwendung mit Oracle Power Objects entwickeln«, werden Sie mehr über Power Objects, wie die Grundlagen von Oracle Basic, das Entwickeln von Berichten, das Erstellen eines Klassenobjekts, und andere Themen lernen.

## Fragen und Antworten

Frage:

Ist es möglich, die Eigenschaften für eine Gruppe von Objekten gleichzeitig zu setzen?

Antwort:

*Ja. Um dies zu tun, wählen Sie alle die Objekte, deren Eigenschaften Sie setzen wollen, durch das Aufspannen eines Rechteckes um Sie herum oder indem Sie die Umschalttaste betätigen und jedes Objekt mit der Maus anklicken. Wenn Sie durch Klicken mit der rechten Maustaste das Attributblatt öffnen, werden Sie bemerken, daß das Attributblatt-Fenster den Titel **Multiple Selection Properties besitzt**. Sie können diejenigen Eigenschaften setzen, die alle Objekte gemeinsam haben - zum Beispiel **FontSize** oder **FontName**.*

Frage:

Welche Dateien sind notwendig, um eine ausführbare Anwendung zu generieren?

Antwort:

*Mit Power Objects können Sie eine Laufzeitdatei (.PO) generieren, welche von dem Power-Objects-Laufzeitmodul namens **PWRRUN.EXE** ausgeführt werden kann. Alternativ dazu können Sie eine selbständig ausführbare Datei generieren, die **PWRRUN** nicht zur Ausführung benötigt. Falls Sie eine selbständig ausführbare Datei erzeugen wollen, benötigen Sie außerdem die entsprechenden Datenbanktreiber und DLLs.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

### Test

1. Richtig oder falsch? Eine Power-Objects-Anwendung verbindet sich mit einer Oracle-Datenbank über ein Transaktionsobjekt der Anwendung.
2. Nennen Sie drei Typen von Objekten, die Sie auf einem Formular plazieren können.
3. Geben Sie ein Beispiel für ein Container-Objekt.

### Übungen

1. Bauen Sie ein Formular für die Anzeige und Modifizierung von Informationen über jeden Dozenten auf. Zeigen Sie die Zahl der Klassen an, falls vorhanden, für die der Dozent zum Unterrichten eingeteilt ist.
2. Ändern Sie das Student-Formular so ab, daß das GPA des zum gegenwärtigen Zeitpunkt gewählten Studenten angezeigt wird.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

# 18

# Eine Anwendung mit Oracle Power Objects entwickeln

[Methoden und Methodencode](#)

[Benutzereingaben auf ihre Gültigkeit überprüfen](#)

[Berechnete Werte in einem Formular anzeigen](#)

[Erstellen einer Klasse](#)

[Eine Instanz einer Klasse zu einem Formular hinzufügen](#)

[Erstellen eines Oracle-Power-Objects-Berichts](#)

[Abschnitte eines Berichts](#)

[Festlegen von Berichtsobjekten](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

In der vorhergehenden Lektion konnten Sie sich mit Oracle Power Objects vertraut machen. Sie haben gelernt, wie man eine Datenbank-Arbeitssitzung definiert, wie man ein Anwendungsobjekt erzeugt, und wie man ein einfaches Formular und ein Master-/Detail-Formular erstellt. In dieser Lektion setzen Sie die Untersuchung einiger der Möglichkeiten von Power Objects einschließlich der folgenden fort:

- Benutzen einiger Standardmethoden für verschiedene Objekte
- Schreiben von Code für Methoden
- Anzeigen eines berechneten Werts in einem Formular
- Erstellen einer Klasse
- Vererbung aus einer Klasse
- Entwickeln eines Berichts

## Methoden und Methodencode



Eine Methode ist ein Satz von Instruktionen, die von einem Objekt als Antwort auf ein Ereignis oder den Aufruf des Objekts ausgeführt werden. Jeder Typ von Objekt besitzt eine Reihe von vordefinierten Methoden, die Standardmethoden genannt werden. Dieser Satz von Instruktionen - wobei es sich um Oracle-Basic-Anweisungen handelt -, die für eine Methode ausgeführt werden, wird als Methodencode bezeichnet. Einige Methoden können Standardabläufe besitzen, die ausgeführt werden, wenn der Entwickler keinerlei Methodencode geschrieben hat.

Die Methoden für ein Objekt werden im Attributblatt des Objekts unterhalb der Eigenschaften aufgelistet. Auf dem Attributblatt wird ein kleiner Diamant links von jeder Eigenschaft angezeigt, ebenso wie eine kleine Pfeilspitze links von jeder Methode. Lassen Sie uns einen Blick auf eine Methode für ein Formular werfen.

1. Erstellen Sie zu Beginn ein anderes Formular, und nennen Sie es MyForm.
2. Fügen Sie dem Formular zwei Textfelder hinzu: Xpos und Ypos.
3. Rufen Sie das Attributblatt für das Formular auf, und blättern Sie hinunter bis zur Methode MouseOver. Diese Methode wird ausgeführt, wenn der Mauszeiger sich über das Objekt hinwegbewegt.
4. Wie in Abbildung 18.1 gezeigt, klicken Sie auf die MouseOver-Methode und geben Sie folgendes ein:

```
XPos.value = x
YPos.value = y
```

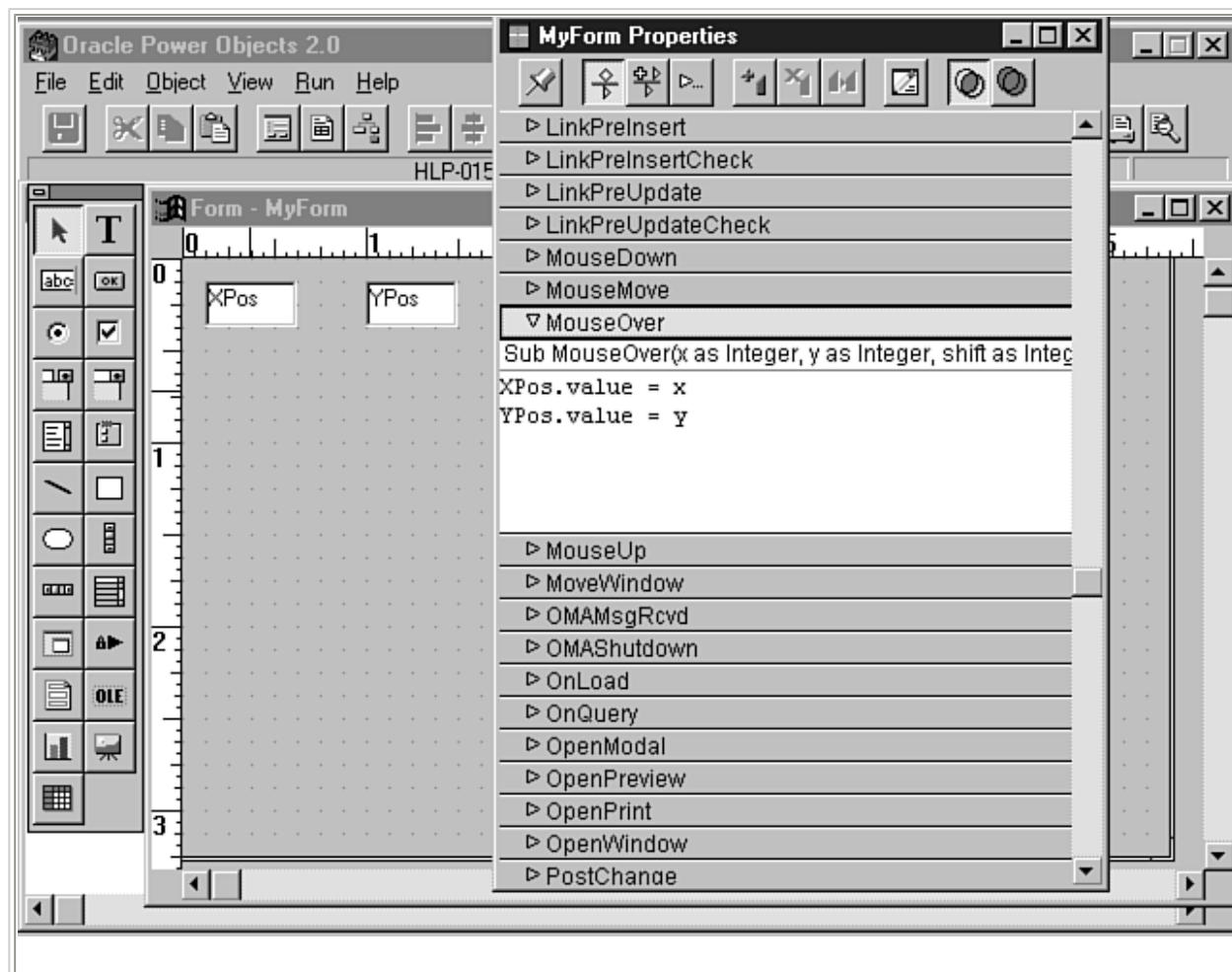
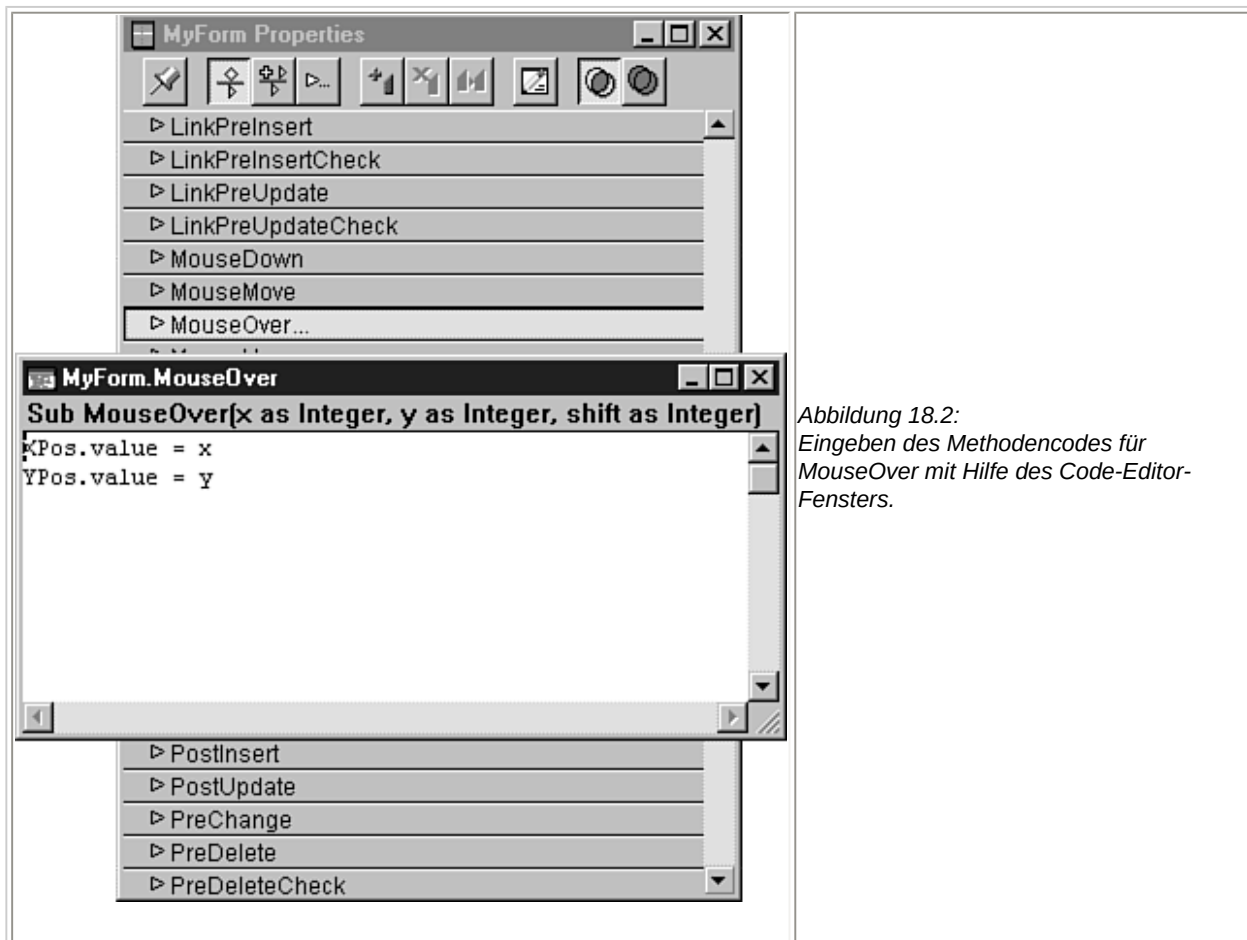


Abbildung 18.1:  
Methodencode für  
MouseOver

Dieses Skript weist die aktuelle x- und y-Position jeweils den Feldern XPos und YPos zu. Klicken Sie auf den Titel der MouseOver-Methode, um das Fenster zu schließen. Alternativ dazu können Sie das Fenster des Code-Editors durch Klicken der Code-Editor-Befehlsschaltfläche im oberen Bereich des Attributblatts (sie ähnelt einem gelben Stift auf einer Seite) aufrufen. Wie Sie in Abbildung 18.2 sehen können, ist die

Größe des Code-Editor-Fensters veränderbar, um es an die Eingabe eines langen Blocks von Methodencode anzupassen.



Schließen Sie das Attributblatt, und testen Sie das Formular durch Klicken mit der rechten Maustaste auf Run Form. Abbildung 18.3 zeigt,

was Ihr Formular im Ausführungsmodus anzeigen sollte.

## Benutzereingaben auf ihre Gültigkeit überprüfen

Lassen Sie uns einen Blick auf eine üblicherweise für ein Steuerelement verwendete Methode werfen: die Validate-Methode. Diese Methode wird zur Erzwingung von Firmenregeln für ein Objekt benutzt. Stellen Sie sich beispielsweise vor, daß eine der Firmenregeln für das Flugle College besagt, daß die Studenten-Identifikationsnummer mit einer 1 beginnen muß. Um diese Regel festzulegen, wählen Sie das Student\_ID-Textfeld auf dem Student-Formular aus und rufen das Attributblatt auf. Blättern Sie herunter zur Validate-Methode - es ist die letzte Methode für das Objekt. Geben Sie die in Listing 18.1 gezeigten Oracle-Basic-Anweisungen ein.

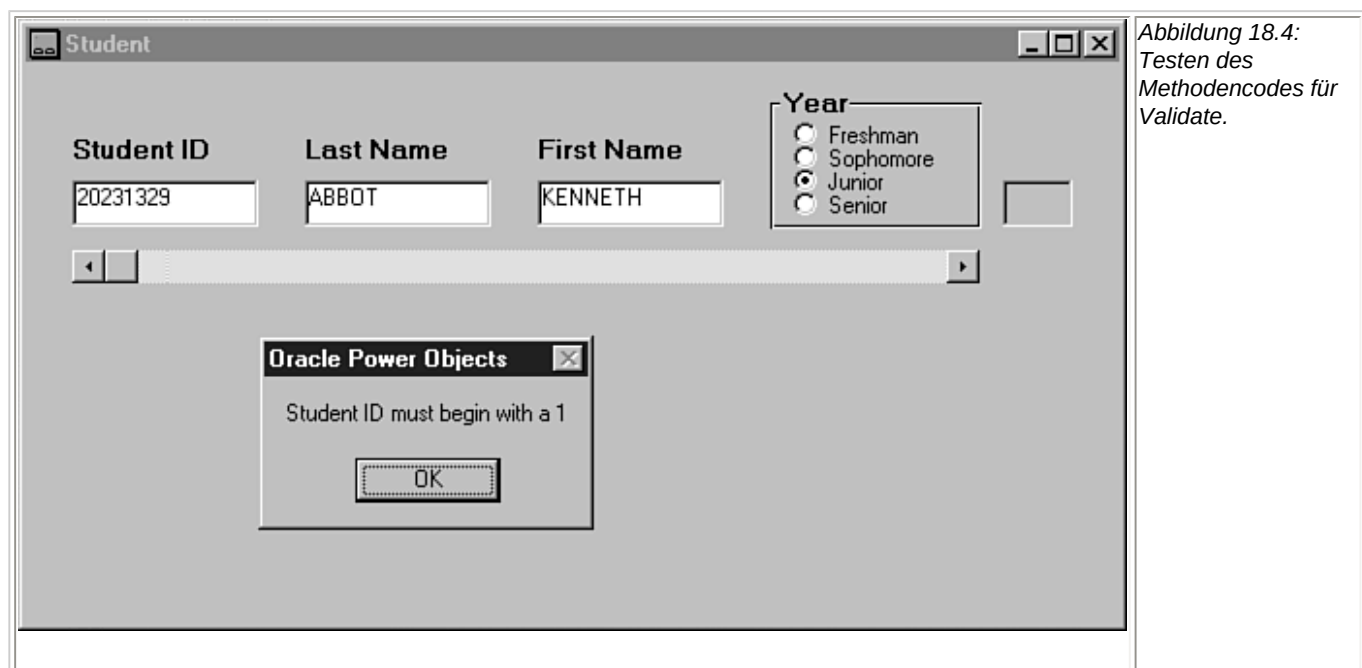
**Listing 18.1: Erzwingen einer Firmenregel im Methodencode für Validate.**

```
IF LEFT(newval,1) = "1" THEN
Validate = TRUE
ELSE
Validate = FALSE
MSGBOX("Student ID must begin with a 1"
END IF
```

Die Firmenregel wird in der Oracle-Basic-Anweisung IF-THEN-ELSE ausgedrückt. Die erste Zeile verwendet die LEFT-Funktion, um das erste Zeichen von Student\_ID zurückzugeben, welches geprüft werden soll. Falls dieses erste Zeichen gleich 1 ist, gibt die Validate-Funktion TRUE zurück, womit angezeigt wird, daß der neue Wert die Gültigkeitsprüfung bestanden hat; andernfalls wird die Validate-Funktion FALSE zurückgeben, um den Benutzer vor dem Wechsel zu einem anderen Steuerelement oder einer Bestätigung der Änderung zu bewahren. Außerdem wird die MSGBOX-Funktion benutzt, um dem Benutzer eine Fehlermeldung zu zeigen.

Lassen Sie uns den Methodencode testen.

1. Schließen Sie das Attributblatt für Student\_ID.
2. Klicken Sie die Befehlsschaltfläche Run Form, um das Formular zu testen.
3. Versuchen Sie, im ersten Datensatz das erste Zeichen von Student\_ID auf 2 zu setzen.
4. Versuchen Sie, ein anderes Feld anzuklicken. Wie Sie in Abbildung 18.4 sehen können, zeigt der Methodencode wie erwartet eine Fehlermeldung an.



5. Um das Formular zu beenden, klicken Sie auf die Rollback-Schaltfläche auf der Symbolleiste (sie ähnelt einem lila Pfeil) und anschließend die Stop-Schaltfläche.

## Berechnete Werte in einem Formular anzeigen

In der vorhergehenden Lektion haben Sie gelernt, wie man ein Suchfeld anzeigt, welches die eingebaute Funktion SQLLOOKUP verwendet. Jetzt werden Sie sehen, wie Sie ein Feld konstruieren können, das einen Wert anzeigt, der aus dem Wert eines anderen Steuerelements innerhalb des Formulars berechnet wurde.

Lassen Sie uns als Beispiel ein Formular konstruieren, das Informationen über jeden Kurs anzeigt.

1. Setzen Sie die folgenden Eigenschaften für das Formular:

RecSrcSession **Flugle**

RecordSource **Course**

OrderBy **Department\_ID, Course\_ID**

Label **Course Information**

Name **CourseInformation**

2. Fügen Sie dem Formular außerdem Textfelder und statischen Text hinzu, wie in Abbildung 18.5 gezeigt wird.

3. Das berechnete Feld hat den Namen **Total\_Cost**; es zeigt die Kosten für einen Kurs an, welche auf einem Preis von \$ 350 pro Einheit plus eventuell zusätzlicher Gebühren beruhen. Wählen Sie das berechnete Feld namens **Total\_Cost**, und klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen. Setzen Sie diese Eigenschaften, wie in Abbildung 18.6 gezeigt wird:

DataSource = **Units.value\*350 + Additional\_Fees.value**

Datatype **Long Integer**

Name **Total\_Cost**

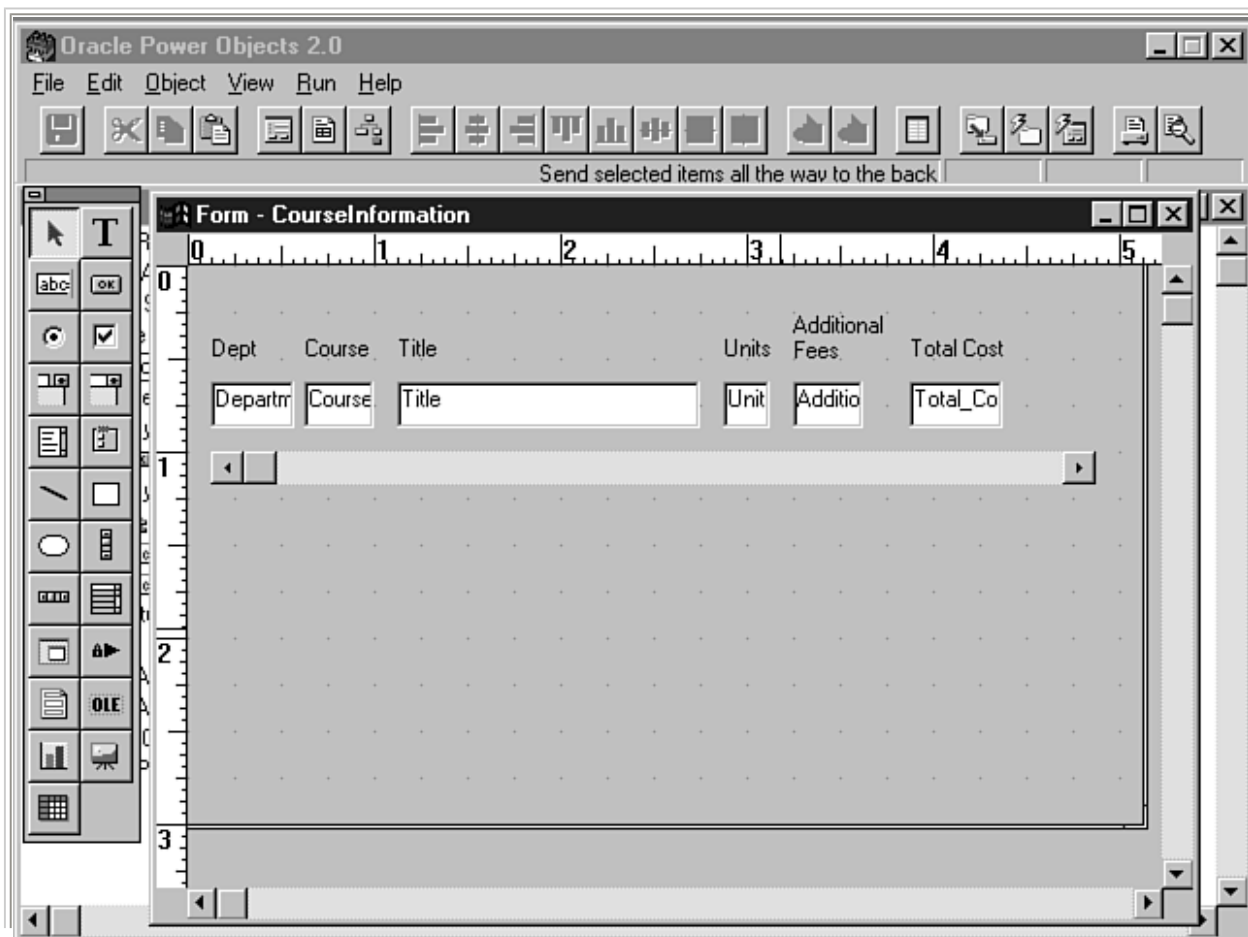


Abbildung 18.5:  
Das  
CourseInformation-  
Formular



**Total\_Cost Properties**

|               |                                |
|---------------|--------------------------------|
| CompareOnLock | True                           |
| CounterIncBy  | 0                              |
| CounterSeq    |                                |
| CounterTiming | Immediate                      |
| CounterType   | None                           |
| DataSize      | 0                              |
| DataSource    | = Units.value*350 + Additional |
| Datatype      | Long Integer                   |
| DefaultValue  |                                |
| DrawStyle     | 3D Control                     |
| Enabled       | True                           |
| FontBold      | False                          |
| FontItalic    | False                          |
| FontName      | System Font                    |
| FontSize      | 8                              |
| FontUnderline | False                          |
| FormatMask    |                                |
| HasBorder     | True                           |
| HasScrollBar  | False                          |
| HelpText      |                                |
| MultiLine     | False                          |
| Name          | Total_Cost                     |
| Password      | False                          |
| PasswordChar  | *                              |

Abbildung 18.6:  
Setzen der Eigenschaft DataSource für das berechnete Feld.

4. Schließen Sie das Attributblatt; es ist an der Zeit, das Formular zu testen, um zu sehen, ob das berechnete Feld funktioniert.

5. Klicken Sie die Befehlsschaltfläche Run Form auf der Symbolleiste an. Wie Sie in Abbildung 18.7 sehen können, zeigt das berechnete Feld die korrekten Summen für jeden Datensatz an.

**Course Information**

| Dept   | Course Title                | Units | Additional Fees | Total Cost |
|--------|-----------------------------|-------|-----------------|------------|
| ANTHRO | 174 PHYS ANTHRO FIELD TECHN | 3     | 55              | 1105       |

Abbildung 18.7:  
Testen des Formulars, das einen berechneten Wert anzeigt.

## Erstellen einer Klasse

Ein Hauptmerkmal von Power Objects ist die Unterstützung von Klassen.



1. Sie können eine Klasse zu einer Anwendung hinzufügen, indem Sie die Befehlsschaltfläche New Class auf der Symbolleiste betätigen, oder durch die Wahl des Menüeintrags Object | New Class. Abbildung 18.8 zeigt, wie die neue Klasse aussieht - wie Sie wahrscheinlich erwartet haben, ist sie am Anfang leer.

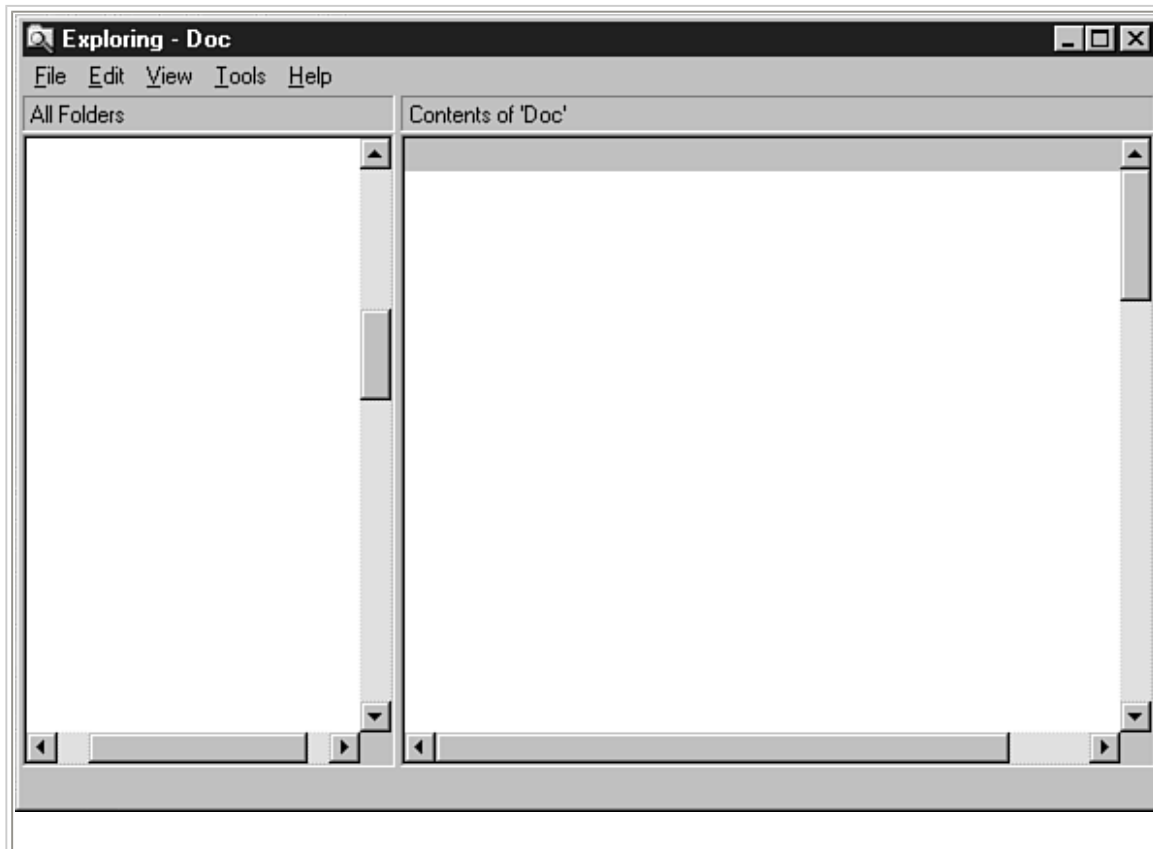


Abbildung 18.8:  
Erzeugung einer  
neuen Klasse

2. Sie werden als nächstes dieser Klasse eine Befehlsschaltfläche hinzufügen. Wählen Sie das entsprechende Symbol aus der Objektpalette, und erstellen Sie eine Befehlsschaltfläche auf der Klasse. Power Objects wird der Schaltfläche einen Namen zuweisen, wie in Abbildung 18.9 zu sehen ist.

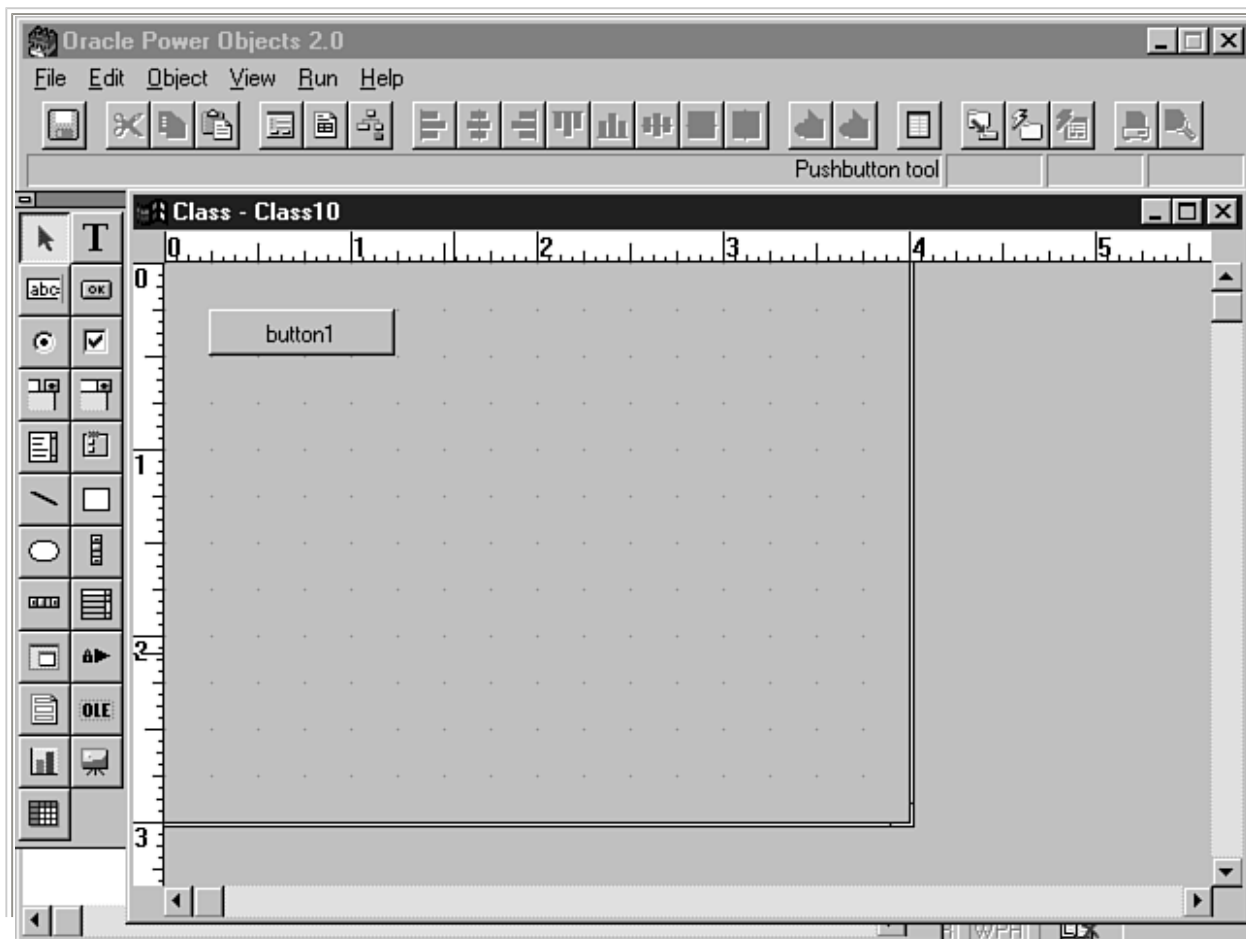
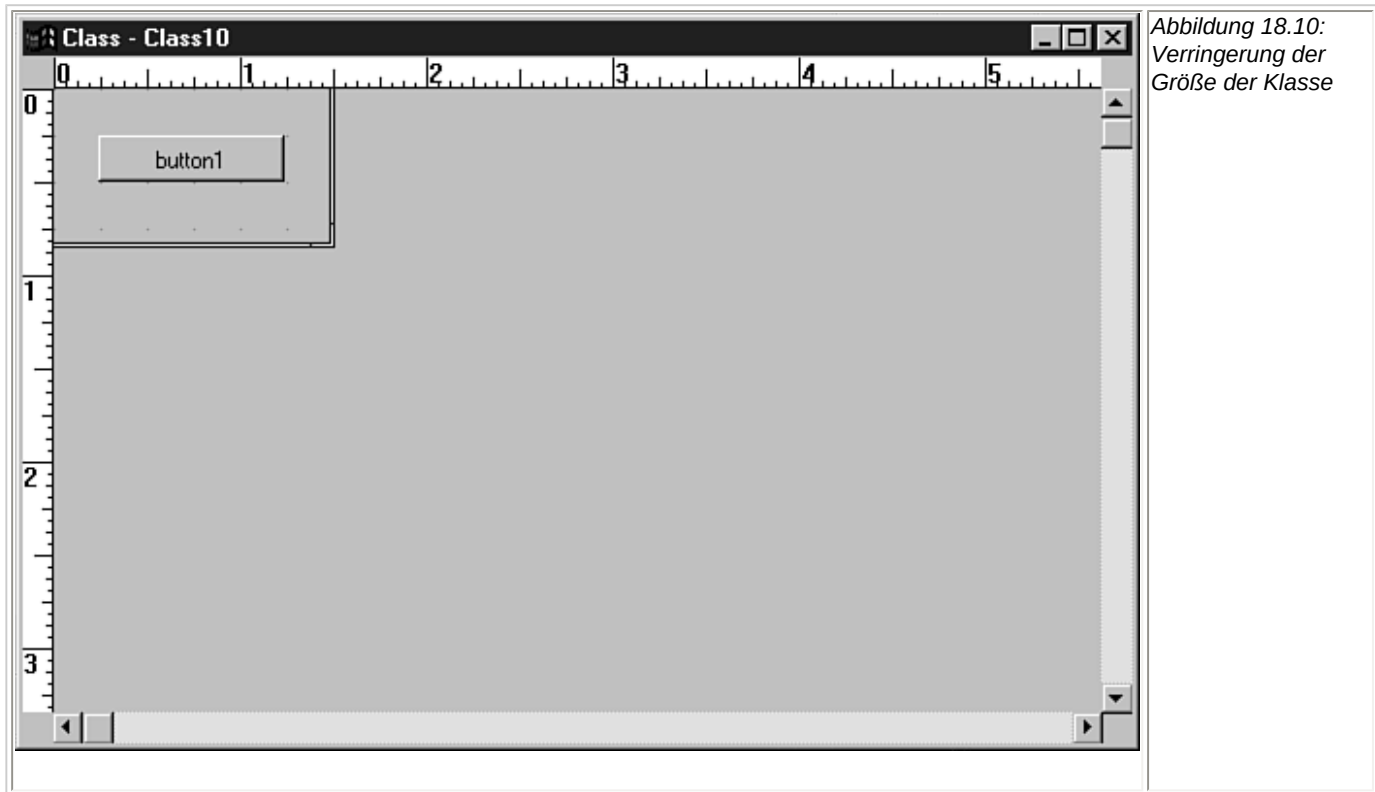


Abbildung 18.9:  
Hinzufügen einer  
Befehlsschaltfläche  
zu der Klasse.



3. Reduzieren Sie die Größe der Klasse, indem Sie die Ecke unten rechts ergreifen und sie in Richtung der linken oberen Ecke verschieben, wie in Abbildung 18.10 gezeigt wird.



Lassen Sie uns einige Eigenschaften der Befehlsschaltfläche ändern. Wählen Sie die Schaltfläche, klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen, und setzen Sie diese Eigenschaften:

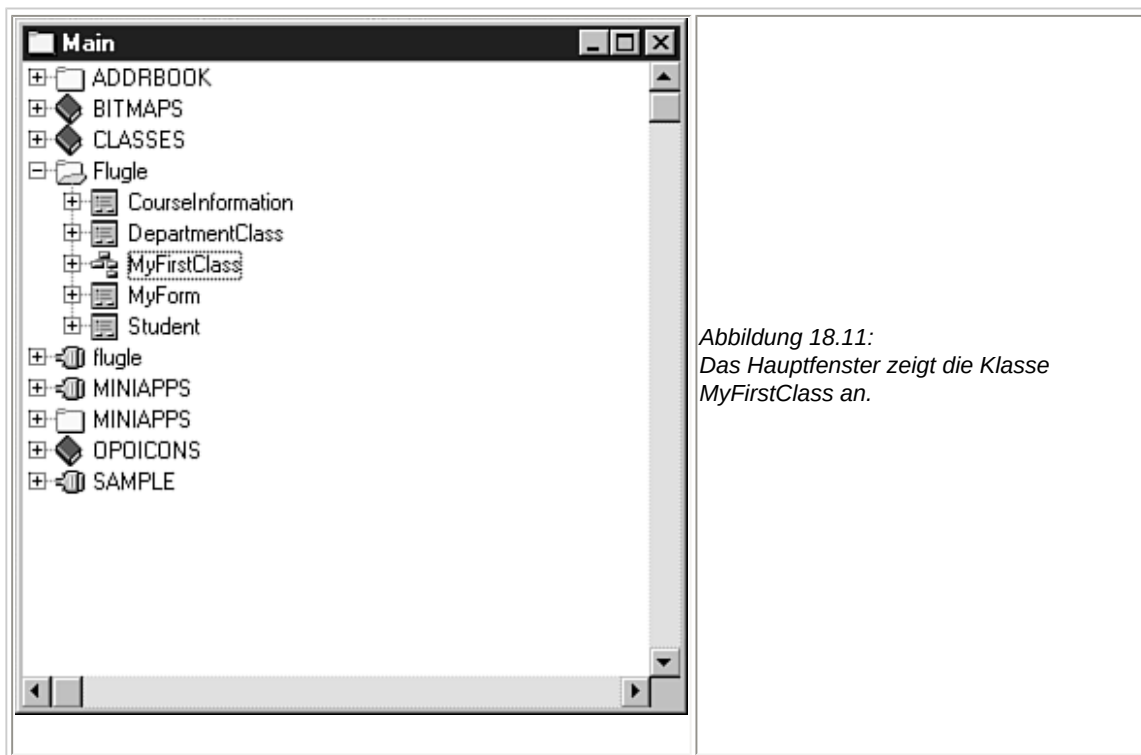
Label **Close**

Name **btnClose**

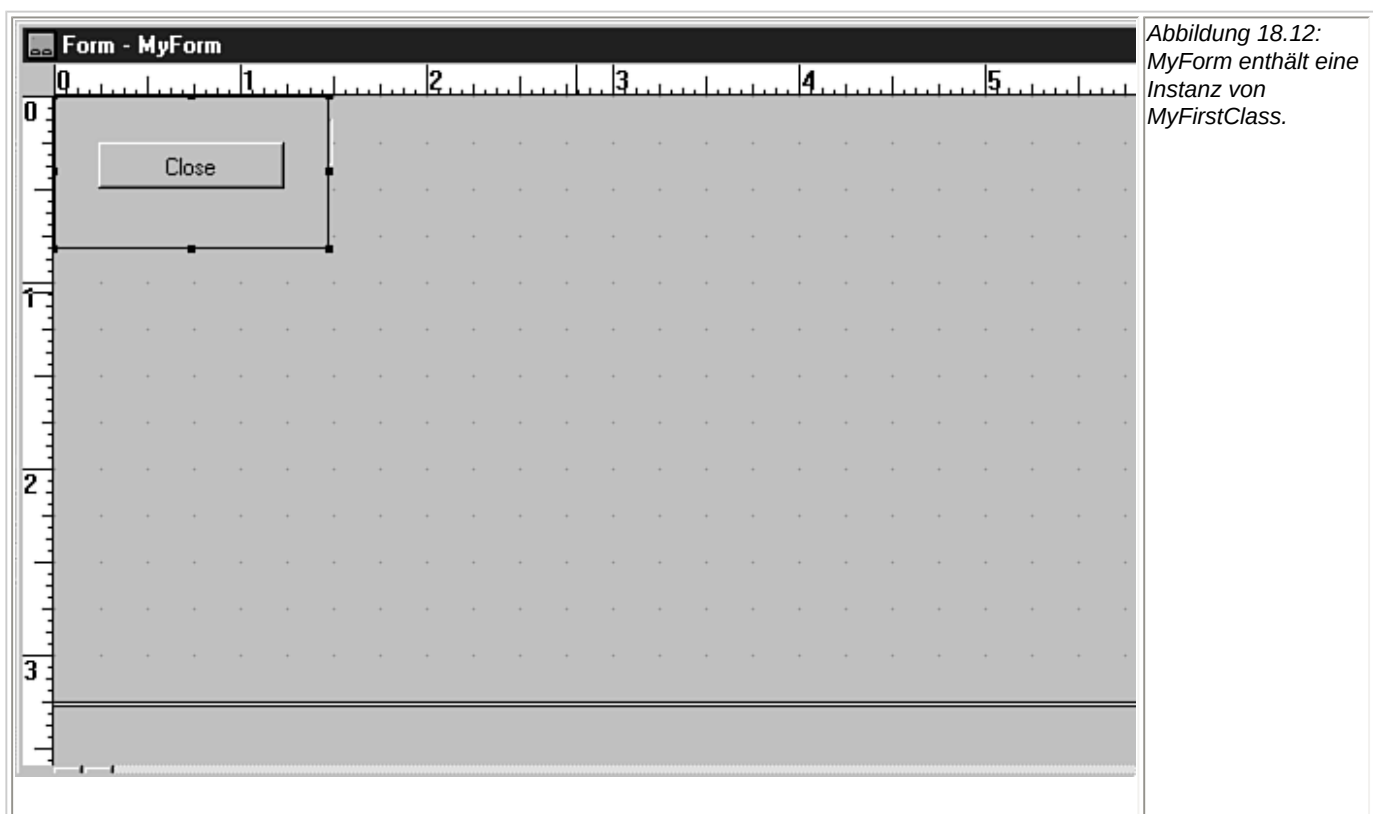
Schließen Sie das Attributblatt. Bevor Sie einem bestehenden Formular eine Instanz der Klasse hinzufügen, werden Sie ihr einen Namen geben wollen. Wählen Sie zu diesem Zweck die Klasse durch Anklicken innerhalb des Definitionsrahmens mit Ausnahme der Befehlsschaltfläche aus, und klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen. Setzen Sie die Eigenschaft Name für die Klasse auf MyFirstClass, und verlassen Sie das Attributblatt. Klicken Sie die Befehlsschaltfläche Save auf der Symbolleiste an, um die Änderungen der Klasse zu speichern. Betätigen Sie (Strg) + (W), um die Klasse zu schließen.

## Eine Instanz einer Klasse zu einem Formular hinzufügen

Wenn Sie einen Blick auf das Hauptfenster werfen, sollten Sie jetzt die Klasse, die Sie erstellt haben - MyFirstClass - in der Auflistung von Objekten sehen, die zur Flugle-Anwendung gehören (siehe Abbildung 18.11).



Fügen Sie als nächstes eine Instanz von MyFirstClass zu MyForm hinzu. Dazu wählen Sie das zu MyFirstClass nächstgelegene Symbol und halten die linke Maustaste gedrückt. Ziehen Sie das Symbol auf dasjenige von MyForm; Sie sollten ein + nahe beim Zeiger sehen. Lassen Sie die Maustaste los. Power Objects wird MyForm mit der Instanz der Klasse in der oberen linken Ecke anzeigen (siehe Abbildung 18.12).



Sie können MyFirstClass auf MyForm neu positionieren, indem Sie sie auswählen und nach rechts verschieben, wobei Sie die beiden Textfelder XPos und YPos zum Vorschein bringen, die Sie bereits erstellt haben. Versuchen Sie, das Formular durch die Wahl von Run | Run Form auszuführen. Versuchen Sie, auf Close zu klicken. Haben Sie irgend etwas erwartet? Nichts sollte passieren, weil Sie der Click-Methode der Befehlsschaltfläche keinerlei Code zur Verfügung stellen. Klicken Sie Stop, um in den Entwurfsmodus zurückzukehren.

### Methodencode zu einer Klasse hinzufügen

Um der Befehlsschaltfläche die Fähigkeit zu geben, das Formular zu schließen, auf welchem es plaziert ist, öffnen Sie MyFirstClass, wählen Sie die Schaltfläche und klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen. Blättern Sie bis zur Click-Methode herunter, und klicken Sie den Titel der Methode an. Geben Sie, wie in Abbildung 18.13 gezeigt, zwei Oracle-Basic-Zeilen ein:

```
MSGBOX("Closing " + GetTopContainer.Name)
GetTopContainer.CloseWindow()
```

Die erste Zeile ruft die MSGBOX-Funktion auf. Das Argument für MSGBOX ist eine Verkettung von Closing mit einem anderen Eintrag. Die zweite Anweisung benutzt GetTopContainer, um eine Objektreferenz auf den obersten Container - Formular, Bericht oder Klasse - zu erhalten, in welchem das Objekt enthalten ist. GetTopContainer.Name gibt den Namen des obersten Objekts zurück, das die Befehlsschaltfläche enthält. Der Nebeneffekt dieser Zeile ist der Aufruf eines Meldungsfeldes, das den Namen jedes Formulars anzeigen wird, das eine Instanz von MyFirstClass enthält. Die zweite Zeile schließt das Formular, welches diese Befehlsschaltfläche enthält.

Um MyForm erneut zu testen, doppelklicken Sie MyForm im Hauptfenster. Klicken Sie auf Close. Wie Sie in Abbildung 18.14 sehen können, erscheint ein Meldungsfeld, welches Sie darüber informiert, daß die Befehlsschaltfläche MyForm schließen wird. Klicken Sie OK auf dem Meldungsfeld an. Klicken Sie auf Stop, um in den Entwurfsmodus zurückzukehren.

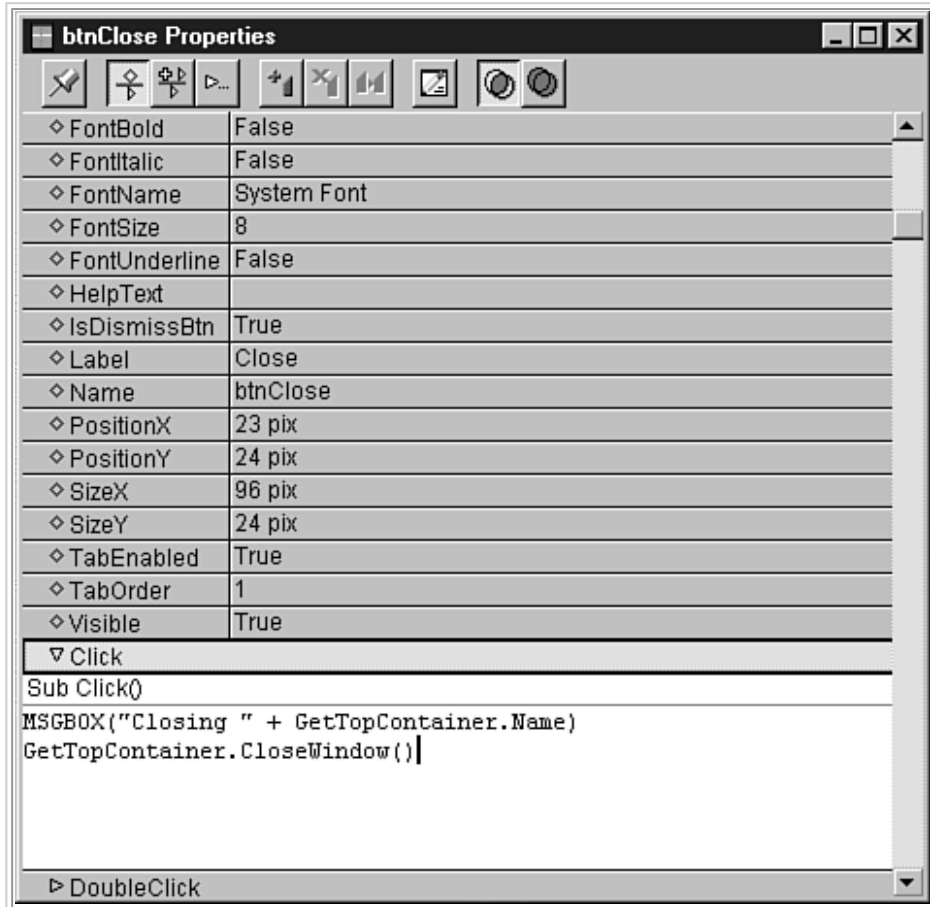
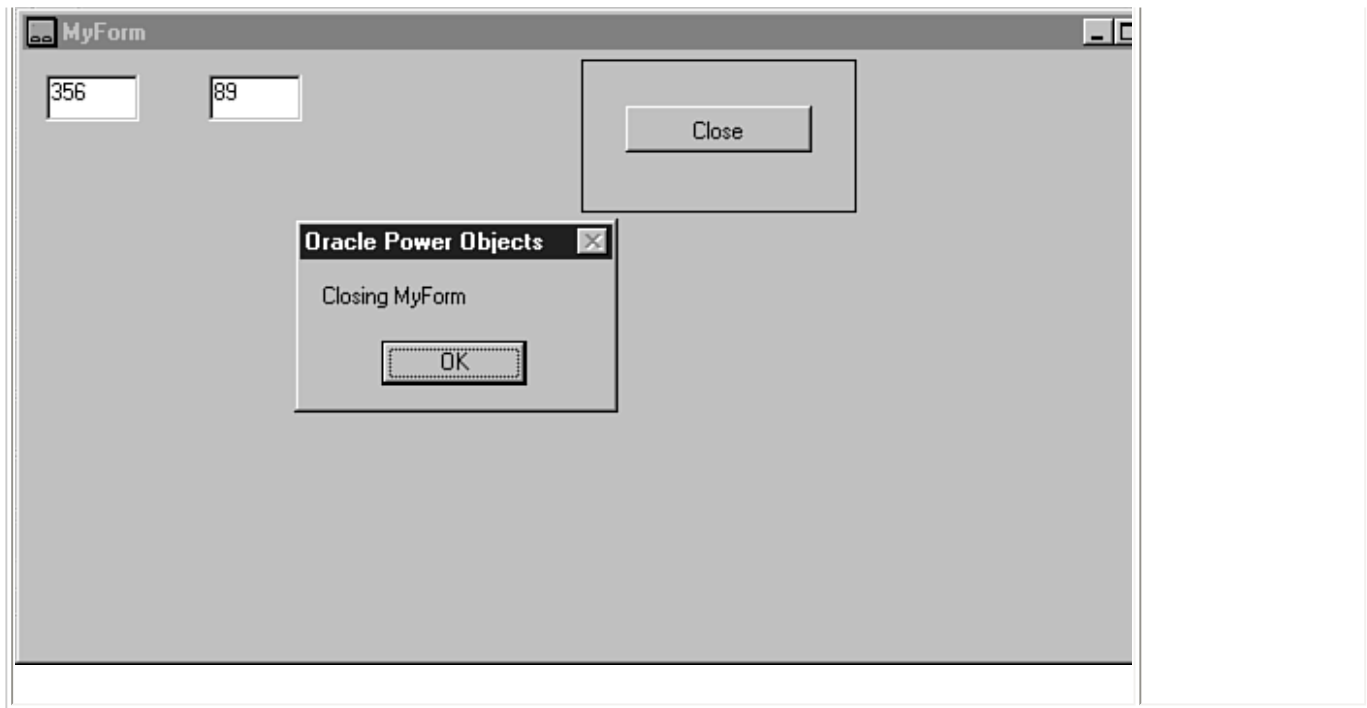


Abbildung 18.13:  
Festlegen des  
Methodencodes für  
die Methode Click in  
btnClose.

Abbildung 18.14:  
Testen der Methode  
Click in btnClose.



### Eine Eigenschaft in einem Klassenobjekt ändern

Sie haben jetzt ein einfaches Beispiel zur Anwendung von Klassen gesehen. Indem Sie den Methodencode für ein Objekt in der Klasse ändern, beeinflussen Sie das Verhalten von anderen Objekten, die eine Instanz der geänderten Klasse enthalten.

Lassen Sie uns einmal die Wirkung betrachten, die eine Änderung der Eigenschaft eines Objekts in einer Klasse hat. Wie Sie vielleicht schon in MyForm bemerkt haben, gibt es einen Rahmen, der die Close-Schaltfläche umgibt. Der Rahmen wird in der Klasse definiert. Rufen Sie MyFirstClass durch einen Doppelklick auf das entsprechende Symbol im Hauptfenster auf. Wählen Sie das Fenster MyFirstClass und klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen. Blättern Sie herunter bis zur Eigenschaft HasBorder, und klicken Sie darauf, so daß sie sich in False ändert. Schließen Sie das Attributblatt.

Rufen Sie MyForm im Hauptfenster durch einen Doppelklick auf das entsprechende Symbol auf. Beachten Sie, daß es den umschließenden Rahmen um die Close-Schaltfläche herum nicht mehr gibt.

### Eine geerbte Klasseneigenschaft überschreiben

Im letzten Beispiel haben Sie die Eigenschaft HasBorder für MyFirstClass durch Setzen auf False geändert; als Ergebnis davon erbt eine Instanz von MyFirstClass auf einem anderen Formular die Eigenschaft, keinen Rahmen zu besitzen. Sie können eine Eigenschaft oder Methode jedoch in einer Klasseninstanz überschreiben. Rufen Sie zum Beispiel MyForm auf, und wählen Sie die Instanz von MyFirstClass, die Sie auf ihr platziert haben. Klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen. Die Eigenschaft HasBorder ist auf False gesetzt, der Wert, den Sie für die Eigenschaft der Klasse gesetzt haben. Sie können jedoch diese Eigenschaft für diese Instanz von MyFirstClass durch Klicken auf False überschreiben; der Wert wird sich in True ändern. Der Diamant links vom Namen der Eigenschaft wird schwarz dargestellt, was besagt, daß die geerbte Eigenschaft überschrieben wurde (siehe Abbildung 18.15).

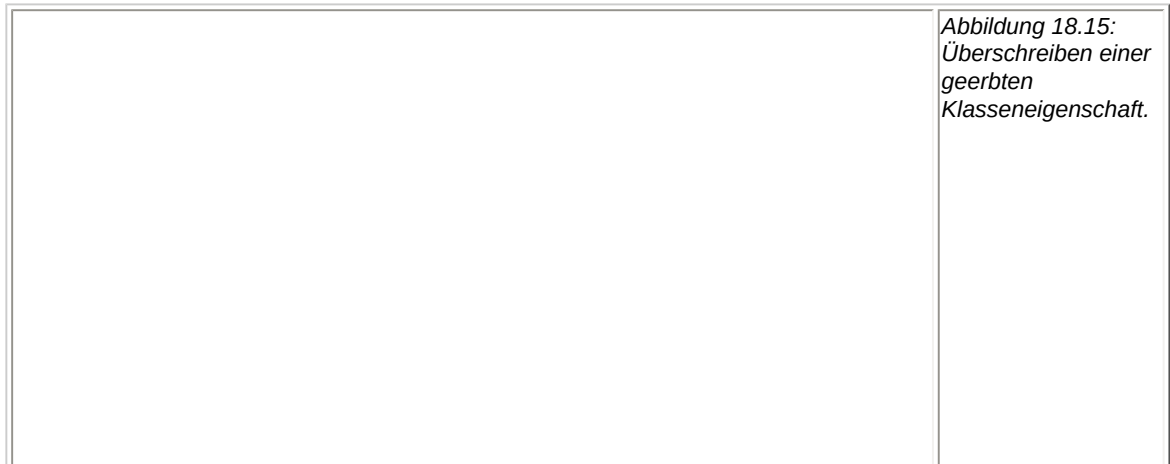
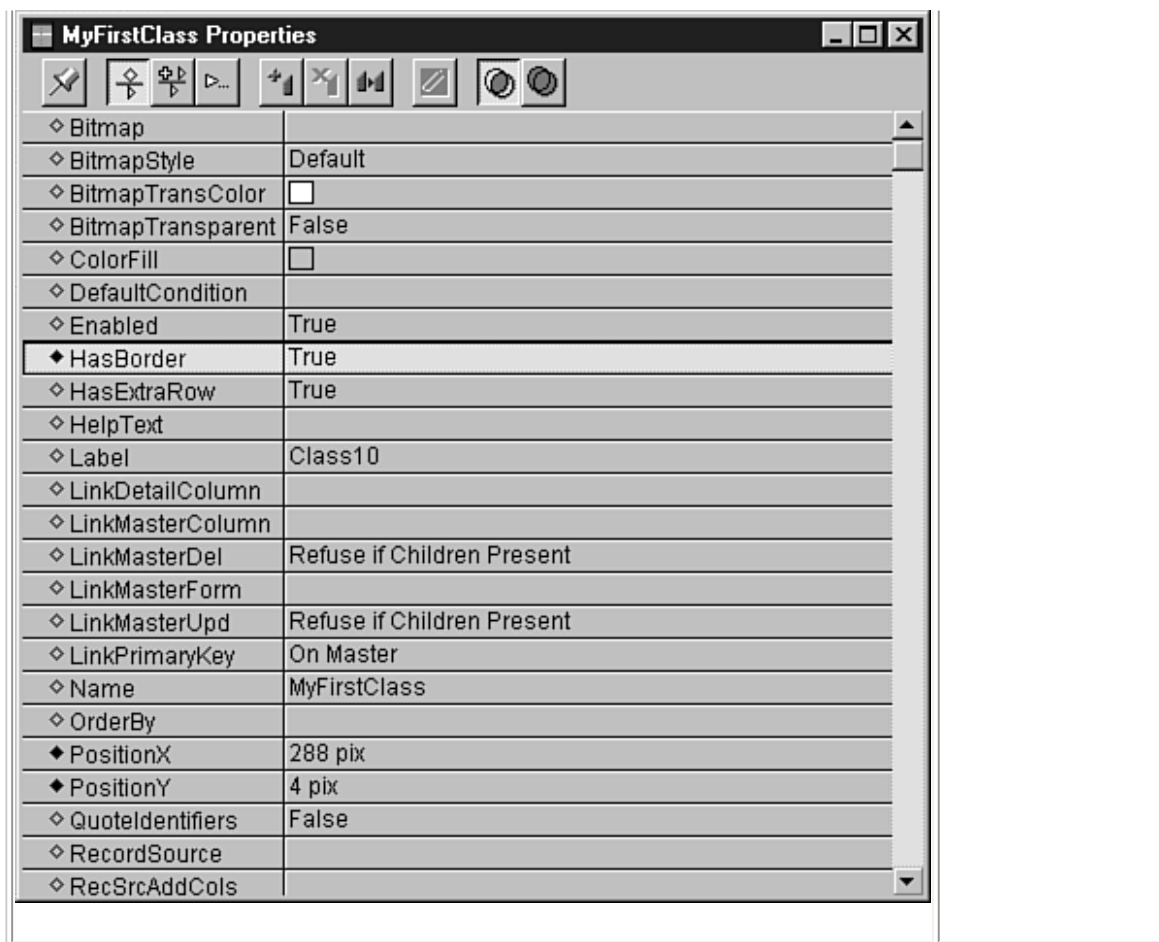


Abbildung 18.15:  
Überschreiben einer  
geerbten  
Klasseneigenschaft.



Wenn Sie die Eigenschaften *PositionX* und *PositionY* für die Instanz von *MyFirstClass* auf *MyForm* betrachten, werden Sie bemerken, daß der schwarze Diamant ein Überschreiben der geerbten Werte anzeigt. Dies geschieht automatisch, wenn Sie *MyFirstClass* auf *MyForm* nach rechts verschieben.

Sie können eine Eigenschaft oder Methode, die Sie überschrieben haben, durch Wählen der Eigenschaft (oder Methode) und Betätigen der Befehlsschaltfläche *Reinherit* im oberen Bereich des Attributblatts (sie befindet sich zwei Schaltflächen nach derjenigen mit der Bezeichnung *+*) neu vererben. Wenn Sie dies tun, wird die Eigenschaft (oder Methode) auf ihren ererbten Wert zurückgesetzt und der Diamant wird wieder grau dargestellt.

### Eine geerbte Klassenmethode überschreiben

Als objektorientierte Programmierumgebung erlaubt Power Objects es Ihnen, eine Methode zu überschreiben, die vererbt wurde. Sie können sehen, wie dies geschieht, indem Sie *MyForm* und *MyFirstClass* betrachten. Wie Sie sich erinnern, enthält *MyFirstClass* eine Befehlsschaltfläche namens *btnClose*, welche Methodencode für die Click-Methode festlegt.

Wenn Sie *MyForm* eine Instanz von *MyFirstClass* hinzufügen, erbt diese Instanz der Klasse den Code der Click-Methode. Lassen Sie uns sehen, wie man den Ablauf dessen, was in *btnClose* festgelegt wurde, überschreiben kann.

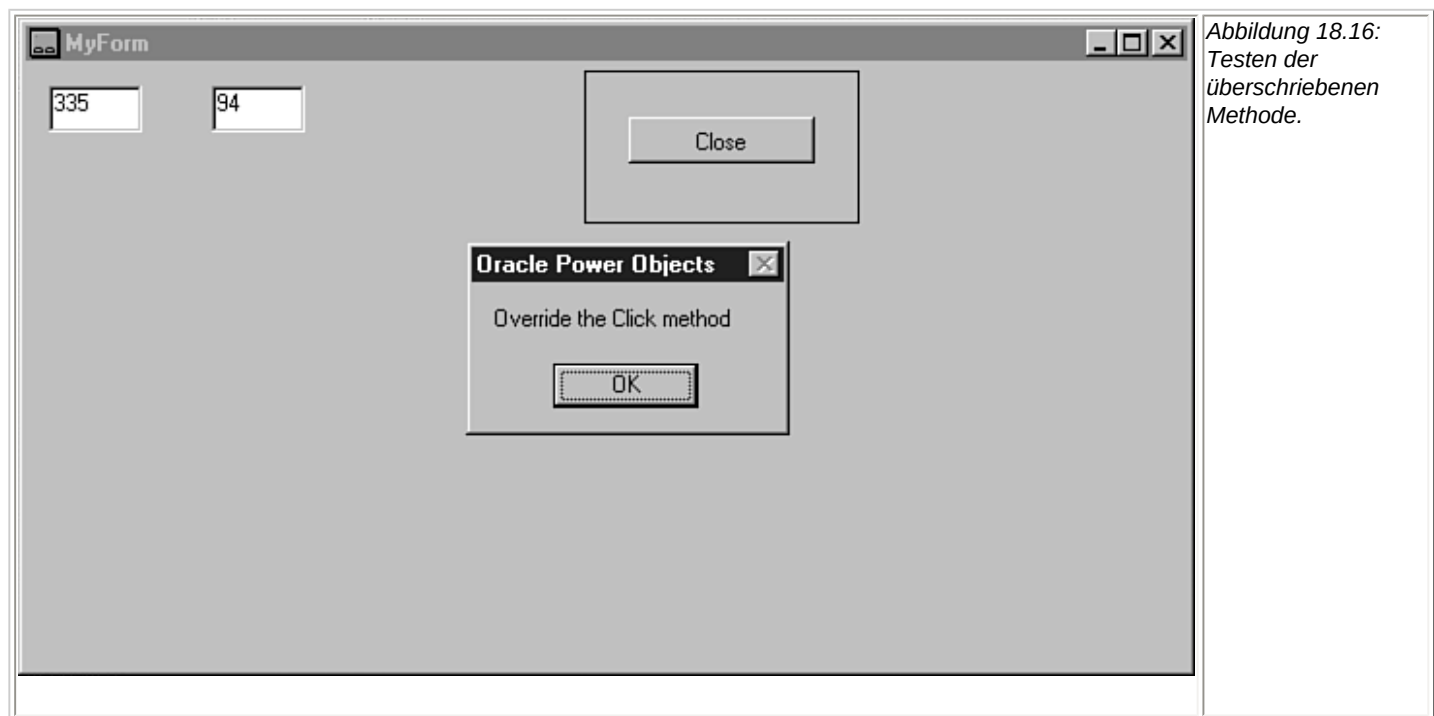
1. öffnen Sie *MyForm* durch einen Doppelklick auf das entsprechende Symbol im Hauptfenster.
2. Wählen Sie *btnClose* und klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen.
3. Blättern Sie herunter bis zur Click-Methode, und geben Sie eine einzelne Zeile ein:

**MSGBOX ("Override the Click method")**

4. Klicken Sie auf den Titel der Methode. Beachten Sie, daß die Methode als ausgefülltes schwarzes Dreieck angezeigt wird, was besagt, daß die ererbte Methode überschrieben wurde.

5. Schließen Sie das Attributblatt, und testen Sie MyForm. Wenn Sie auf Close klicken, erscheint ein Meldungsfeld, das Sie darüber informiert, daß die Click-Methode überschrieben wurde (siehe Abbildung 18.16).

6. Wenn Sie OK im Meldungsfeld anklicken, geschieht nichts weiter. Wie Sie wohl erwartet haben, wird der vererbte Code nicht ausgeführt.



Power Objects erlaubt es Ihnen außerdem, den Methodencode zu überschreiben *und* den vererbten Code aufzurufen. Die Anweisung `Inherited.Method_name()` wird benutzt, um den standardmäßigen Ablauf einer Methode auszuführen. Stellen Sie sich anhand des Beispiels mit MyForm und MyFirstClass vor, Sie haben sich zur Anzeige eines Meldungsfeldes beim Betätigen von Close entschieden, wollen aber ebenfalls den vererbten Methodencode ausführen. Um dies zu erreichen, wählen Sie `btnClose` auf MyForm und klicken mit der rechten Maustaste, um das Attributblatt zu öffnen. Blättern Sie zur Click-Methode und geben Sie diese zwei Oracle-Basic-Zeilen ein:

```
MSGBOX("Override the Click method")  
Inherited.Click()
```

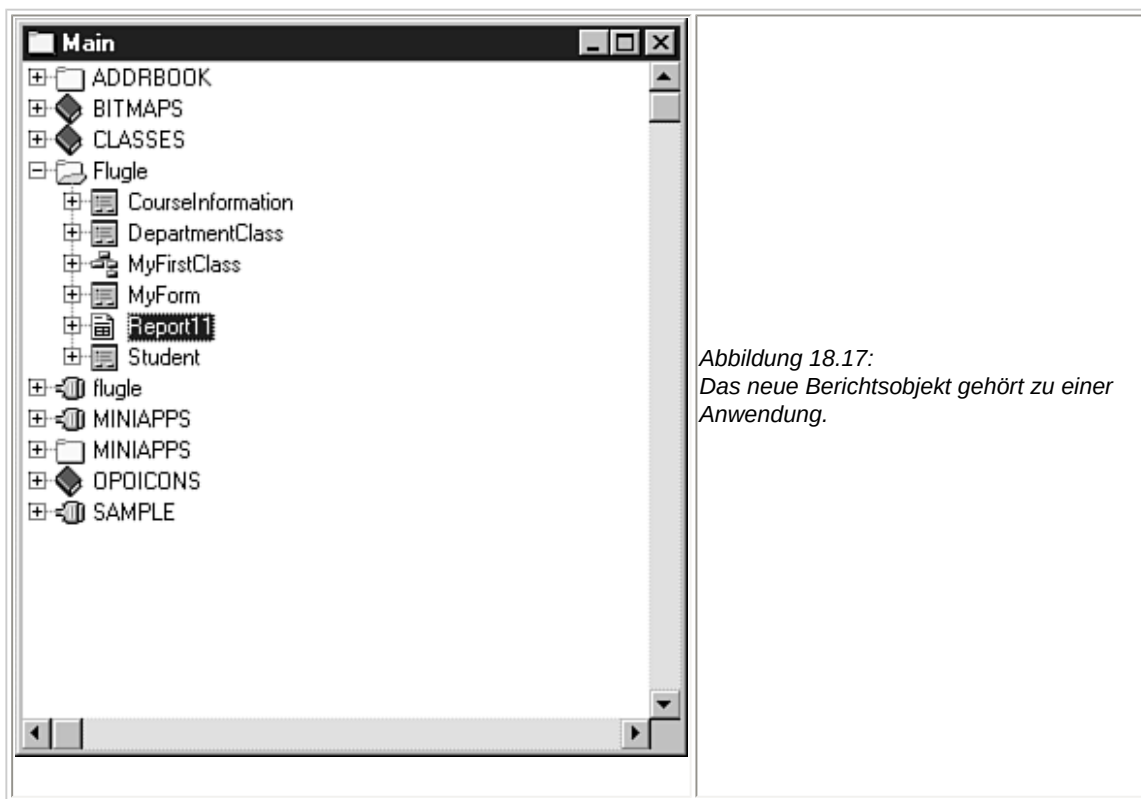
Die erste Zeile zeigt ein Meldungsfeld mit der angegebenen Nachricht an. Die zweite Zeile ruft den Methodencode für `btnClose` auf, der innerhalb der Klasse MyFirstClass festgelegt ist. Verlassen Sie das Attributblatt, und testen Sie das Formular. Wenn Sie auf Close klicken, zeigt das Meldungsfeld die Nachricht des Überschreibens der Click-Methode an. Wenn Sie auf OK klicken, erscheint ein anderes Meldungsfeld, welches zeigt, daß der vererbte Code ausgeführt wird.

Wenn Sie mehr über die Anwendung von Klassen und Vererbung gelernt haben, werden Sie entdecken, daß Sie auf verschiedenen Wegen von ihrer Benutzung profitieren:

- Mittels Vererbung werden Sie weniger Code schreiben müssen.
- Sie werden in der Lage sein, Ihre Klassen in einer großen Bandbreite von Anwendungen einsetzen zu können.
- Wenn Sie einen vernünftigen Satz von Klassen erstellt und diese in passender Weise vererbt haben, wird sich die Wartung Ihrer Anwendungen spürbar vereinfachen; Sie haben eine geringere Menge an Code zu ändern, als wenn Sie keine Klassen verwenden.

## Erstellen eines Oracle-Power-Objects-Berichts

Das letzte Objekt, das Sie in Power Objects erforschen werden, ist das Berichtsobjekt. Eine Anwendung kann keine oder mehrere Berichte enthalten. Um einer Anwendung einen Bericht hinzuzufügen, öffnen Sie das Hauptfenster, wählen die Anwendung, in welcher Sie den Bericht erstellen wollen, und betätigen Sie die Befehlsschaltfläche `New Report` auf der Symbolleiste. Alternativ dazu können Sie `Object | New Report` wählen. Ein Fenster wird erscheinen, das den neuen Bericht enthält. Wenn Sie das Hauptfenster betrachten, werden Sie sehen, daß der neue Bericht zur Anwendung gehört (siehe Abbildung 18.17).



## Abschnitte eines Berichts

Wenn Sie die Größe des Fensters maximieren, welches den neuen Bericht enthält, werden Sie sehen, daß der Bericht fünf Abschnitte aufweist (siehe Abbildung 18.18):

**ReportHeader** Die Objekte, die Sie in diesem Abschnitt platzieren, werden einmalig zu Beginn des Berichts angezeigt.

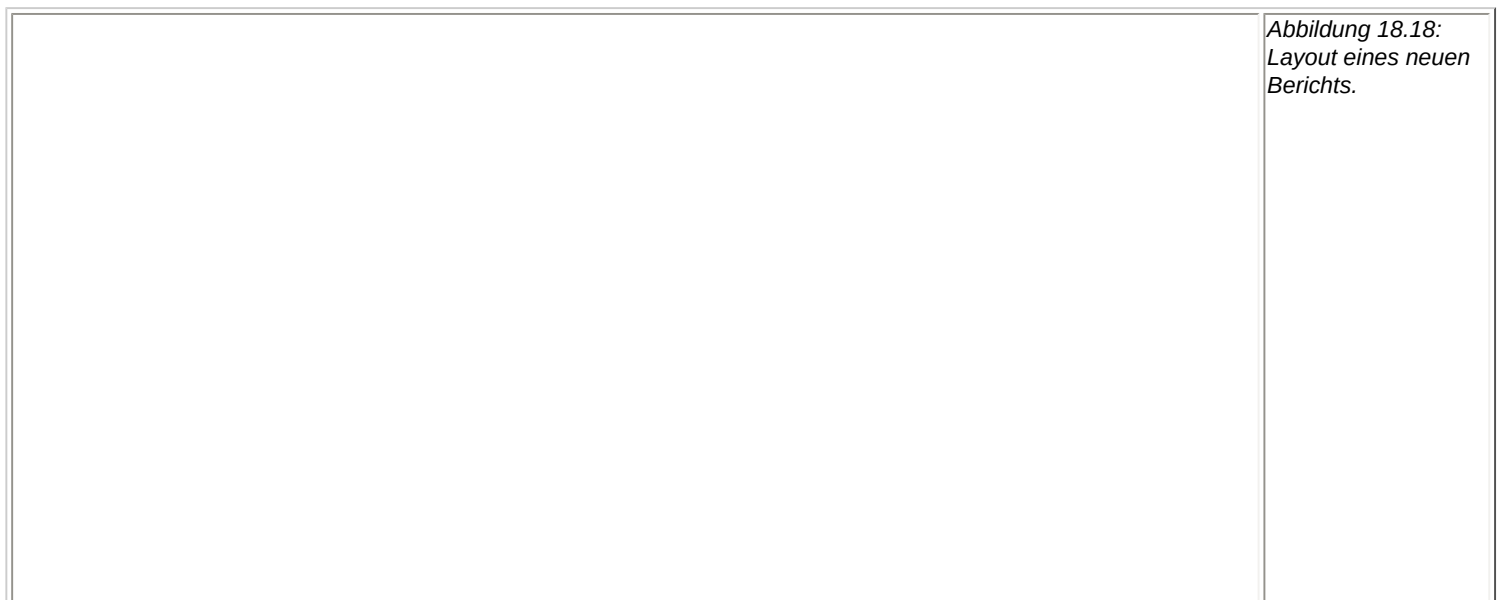
**PageHeader** Die Objekte, die Sie in diesem Abschnitt platzieren, werden zu Beginn jeder Seite angezeigt.

**Detail** Dieser Abschnitt enthält die Objekte, die den Rumpf des Berichts ausmachen. Diese Objekte werden einmal für jeden Datensatz wiederholt, der vom Bericht erhalten wird.

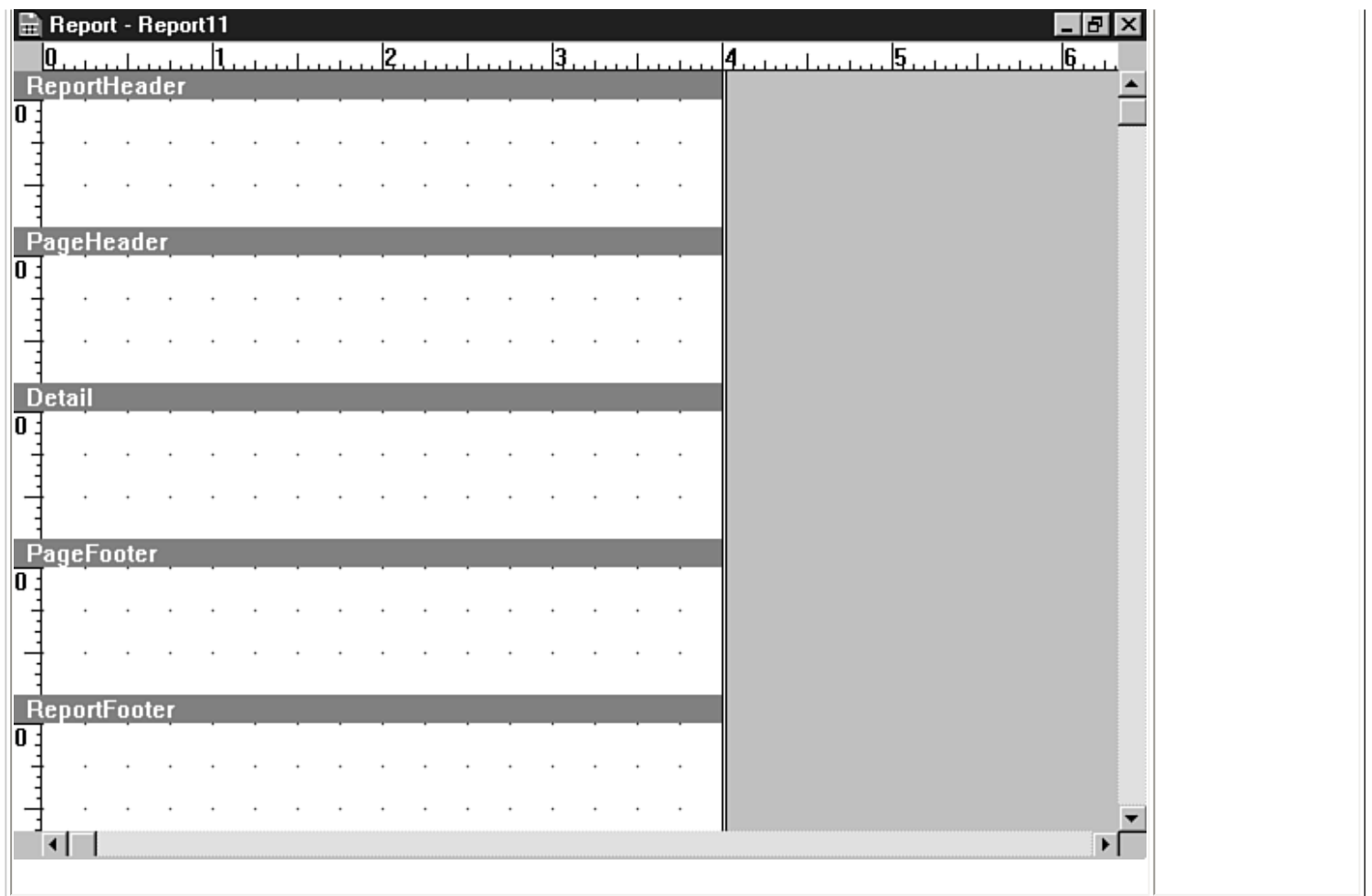
**PageFooter** Die Objekte, die Sie in diesem Abschnitt platzieren, werden am Ende jeder Seite angezeigt.

**ReportFooter** Die Objekte, die Sie in diesem Abschnitt platzieren, werden einmalig am Schluß des Berichts angezeigt.

Vergrößern Sie die Breite des Berichts, indem Sie die rechte Ecke des Berichts nach rechts ziehen.







## Festlegen von Berichtsobjekten

Lassen Sie uns einige Eigenschaften des neuen Berichts setzen:

1. Klicken Sie vom Berichtsfenster aus mit der rechten Maustaste, um das Attributblatt zu öffnen, und legen Sie diese Eigenschaften fest:

Label **Course Catalog**

Name **rptCourseCatalog**

RecordSource **Course**

RecSrcSession **Flugle** (dies ist die Datenbank-Arbeitssitzung, die vom Bericht verwendet wird)

2. Schließen Sie das Attributblatt.

3. Lassen Sie uns dem Bericht einen Gruppenbereich hinzufügen. Klicken Sie in der Objektpalette auf die Befehlsschaltfläche Berichtsbereich (ähnelt einer Seite, die rote Rechtecke enthält), und klicken Sie innerhalb des Detailbereichs auf den Bericht. Sie werden bemerken, daß der Bericht jetzt zwei zusätzliche Bereiche enthält: einen Bereich GroupHeader und einen Bereich GroupFooter (siehe Abbildung 18.19).

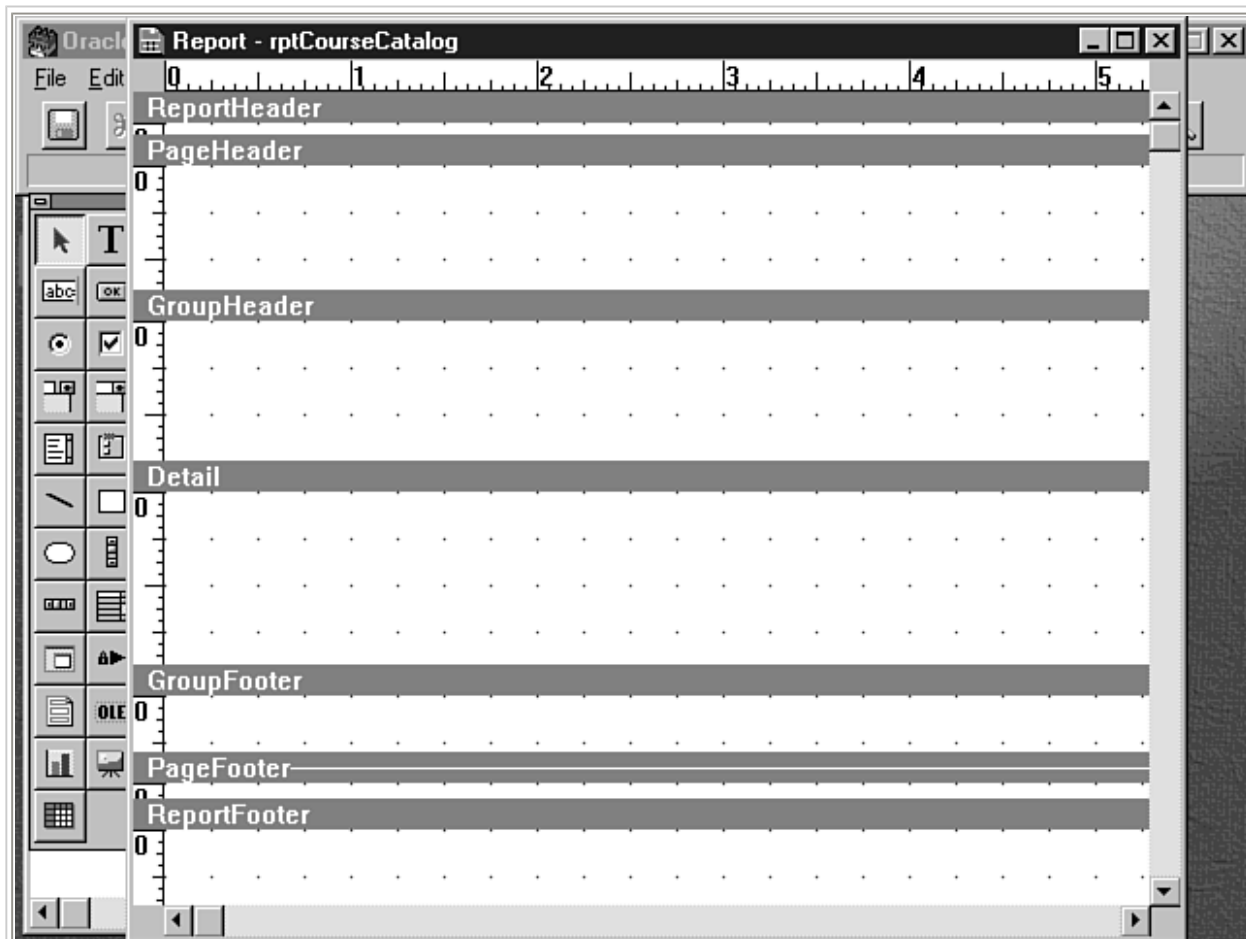
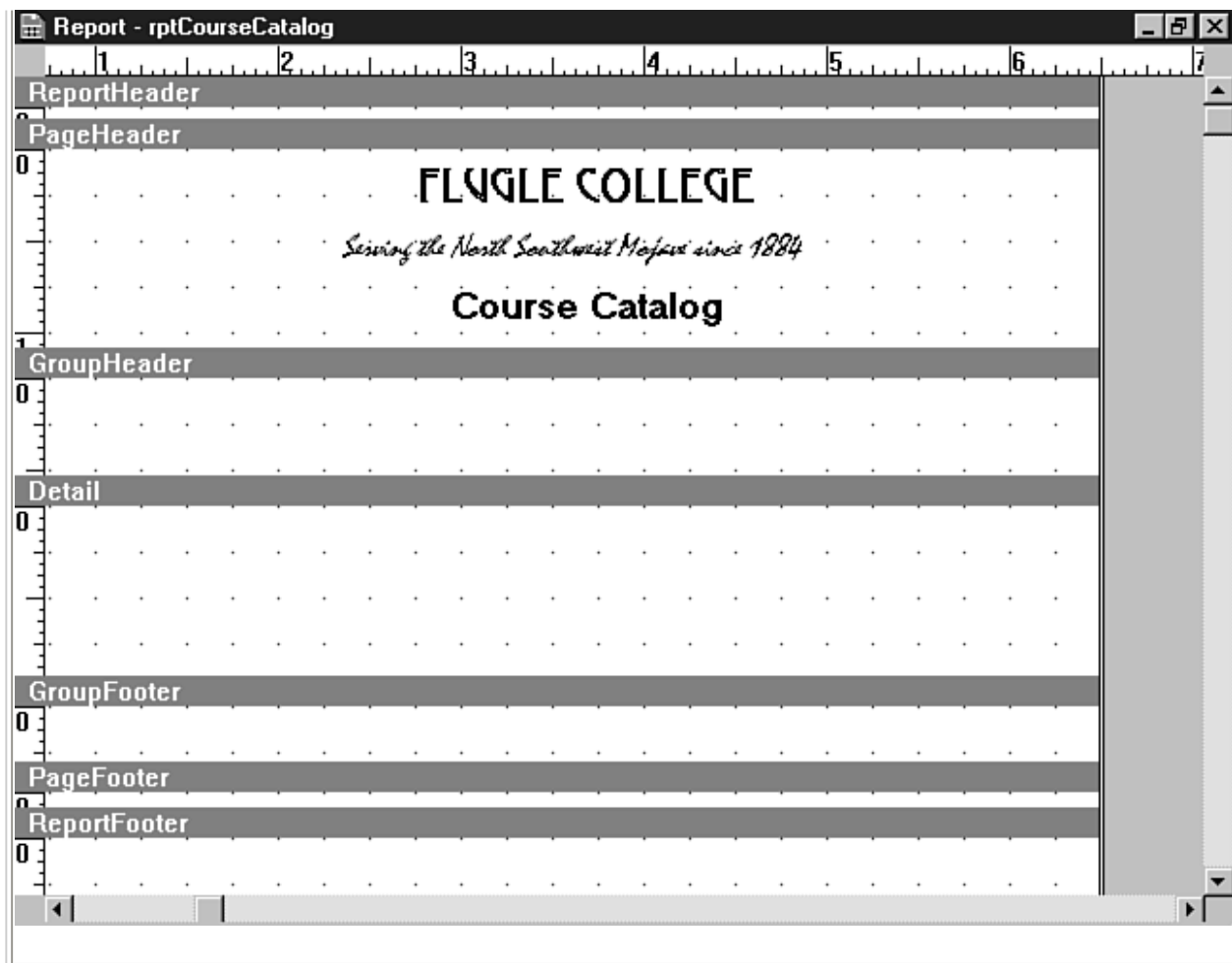


Abbildung 18.19:  
Zum Layout eines  
Berichts  
hinzugefügter  
GroupHeader und  
GroupFooter.

## Festlegen des PageHeader-Bereichs

Die Objekte im PageHeader-Bereich werden als Kopf einer jeden Berichtsseite ausgedruckt. Deshalb ist dies der Abschnitt, in dem Sie den Titel des Berichts und beliebige andere Informationen festlegen, von denen Sie wünschen, daß sie auf jeder Berichtsseite erscheinen sollen. Wählen Sie den PageHeader durch Klicken auf den dunkelgrauen Streifen, und klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen. Der PageHeader besitzt als Standard eine Eigenschaft namens FirstPgHdr, die auf False gesetzt ist, was bewirkt, daß die Objekte im PageHeader der ersten Seite unterdrückt werden. Setzen Sie diese Eigenschaft auf True, und schließen Sie das Attributblatt. Wie in Abbildung 18.20 gezeigt wird, fügen Sie dem Bericht einen Titel und einen Untertitel hinzu. Klicken Sie die Befehlsschaltfläche Static Text auf der Objektpalette und anschließend den PageHeader an der Stelle an, an welcher der Titel erscheinen soll. Sie werden ein markiertes Feld mit dem Titel static1 sehen; geben Sie den Titel ein, den Sie sich für den Bericht wünschen. Wiederholen Sie dieselben Schritte, um den Untertitel hinzuzufügen.

Abbildung 18.20:  
Zum Bericht  
hinzugefügter  
Seitenkopf



### Festlegen des GroupHeader-Bereichs

Ein Group-Bereich wird benutzt, um die Berichtsausgabe zu »unterbrechen«. Für diesen Beispielbericht soll der Bericht unterbrochen werden, wann immer das Department wechselt, so daß eine neue Spaltenüberschrift für dieses Department gedruckt wird.

1. Wählen Sie die Befehlsschaltfläche Text Field aus der Objektpalette und klicken Sie in den GroupHeader-Bereich. Klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen, und setzen Sie diese Eigenschaften für das Textfeld (siehe Abbildung 18.21):

DataSource **Department\_ID**

Datatype **String**

DrawStyle **Standard Control** (in der Annahme, daß Sie keinen vertieft erscheinenden Berichtseintrag wünschen)

HasBorder **False** (in der Annahme, daß Sie keinen Rahmen um die einzelnen Berichtseinträge wünschen)

Name **Department\_ID**

2. Schließen Sie das Attributblatt.

3. Fügen Sie dem Department\_ID-Feld auf dem Bericht eine Bechriftung hinzu, indem Sie die Befehlsschaltfläche Static Text auf der Objektpalette und anschließend links vom Department\_ID-Textfeld klicken.

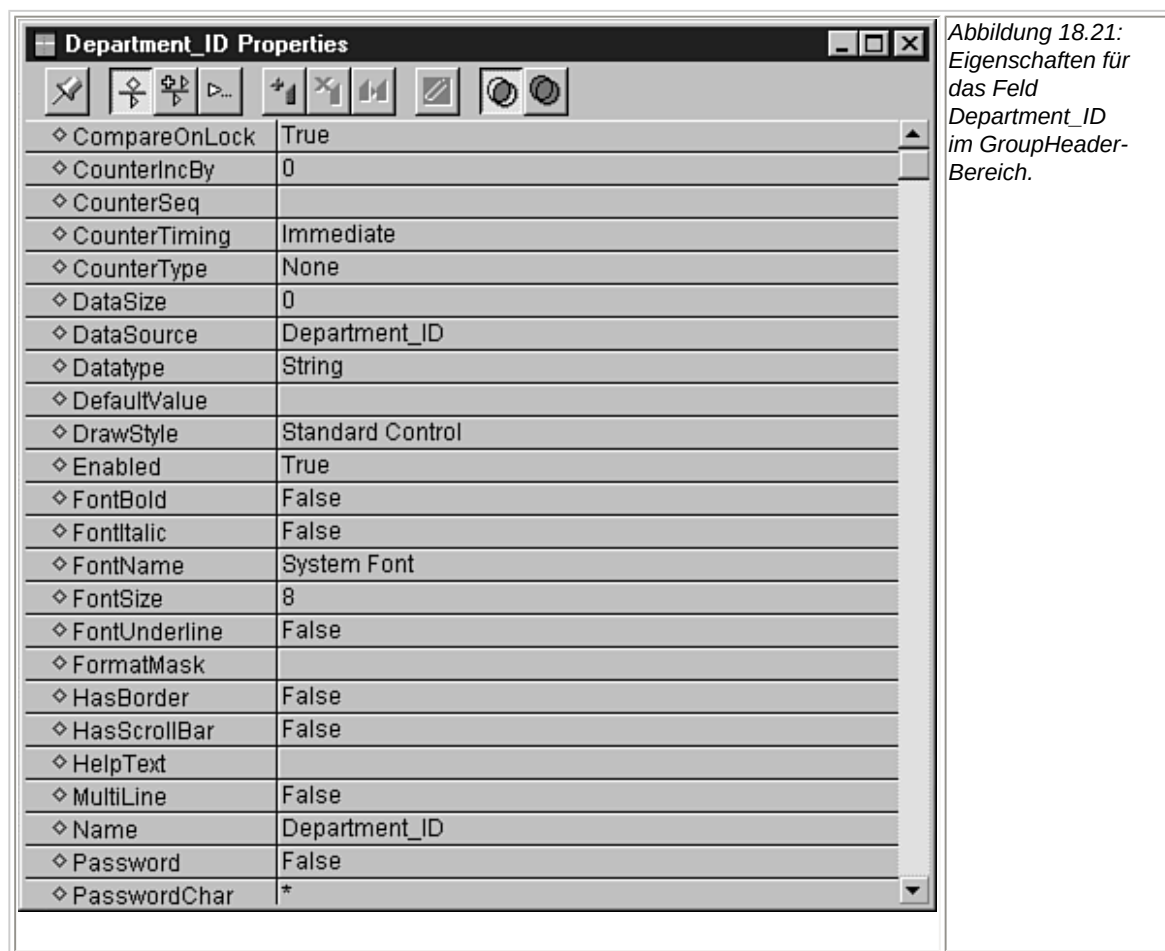
4. Setzen Sie die Eigenschaften für den statischen Text, wie z.B. Name und FontBold.

5. Sie müssen außerdem die Eigenschaft für den GroupHeader-Bereich selbst setzen.

6. Wählen Sie den GroupHeader-Bereich und klicken Sie mit der rechten Maustaste, um das Attributblatt zu öffnen.

7. Es gibt eine einzelne Eigenschaft, die Sie setzen müssen - GroupCol -, die auf Department\_ID gesetzt werden sollte.

Es gibt noch eine weitere Eigenschaft, die Sie vielleicht für einige Berichte setzen wollen: PageOnBreak. Diese Eigenschaft ist standardmäßig auf False gesetzt, was bedeutet, daß Oracle Objects *nicht* zur nächsten Seite wechseln wird, wenn es einen neuen Wert für Department\_ID gibt. Wenn Sie dieses Verhalten wünschen, setzen Sie PageOnBreak auf True.



## Festlegen des Detailbereichs

Sie müssen dem Detailbereich immer noch drei Objekte hinzufügen: die Kurs-ID, den Namen des Kurses und dessen Beschreibung. Dies werden alles Textfelder sein. Um diese drei Objekte im Detailbereich zu platzieren, wählen Sie die Befehlsschaltfläche Text Field aus der Objektpalette und fügen Sie drei Textfelder in den Bereich ein.

Wählen Sie das erste Textfeld, rufen Sie sein Attributblatt auf, und setzen Sie diese Eigenschaften:

DataSource **Course\_ID**  
 Datatype **String**  
 DrawStyle **Standard Control**  
 HasBorder **False**  
 Name **Course\_ID**

Wählen Sie das zweite Textfeld, rufen Sie sein Attributblatt auf, und setzen Sie diese Eigenschaften:

DataSource **Title**  
 Datatype **String**  
 DrawStyle **Standard Control**  
 HasBorder **False**  
 Name **Title**

Wählen Sie das dritte Textfeld, rufen Sie sein Attributblatt auf, und setzen Sie diese Eigenschaften:

DataSource **Description**  
 Datatype **String**  
 DrawStyle **Standard Control**  
 HasBorder **False**

Name **Description**

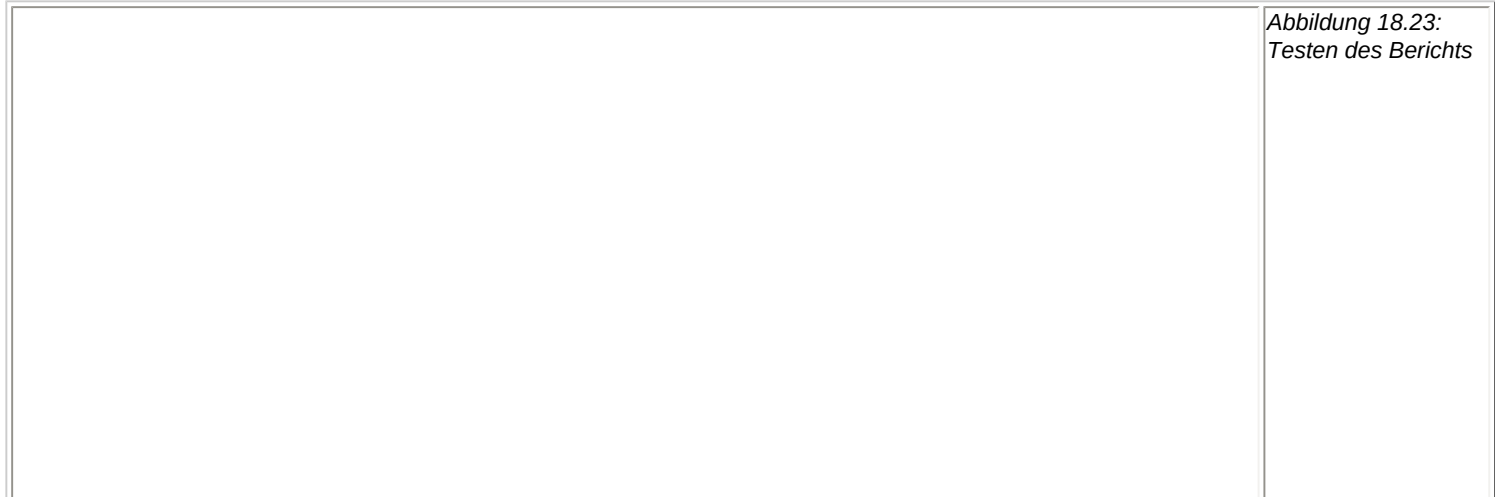
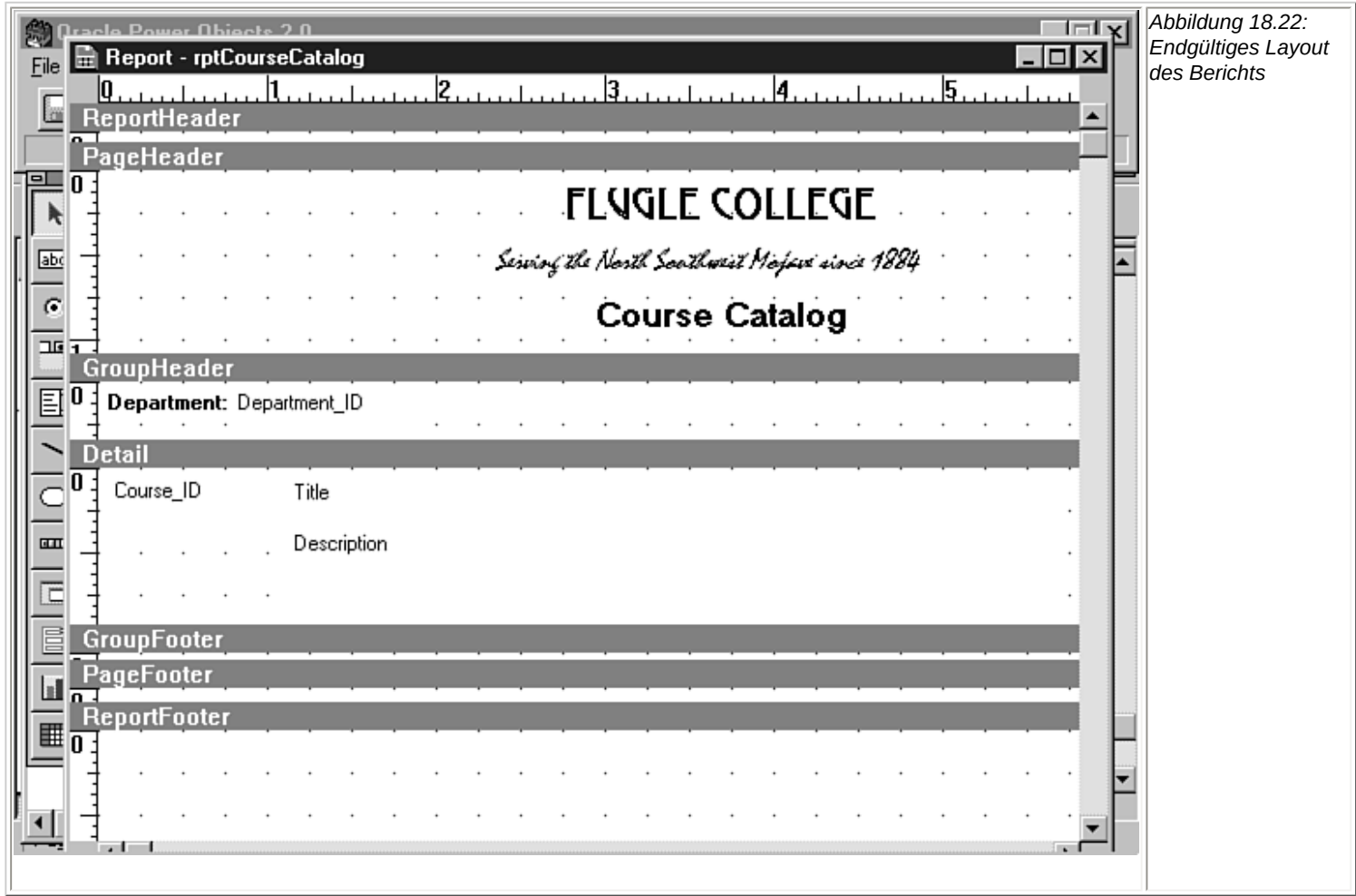
Es gibt jedoch zwei weitere Eigenschaften, die Sie für das Description-Feld festlegen müssen:

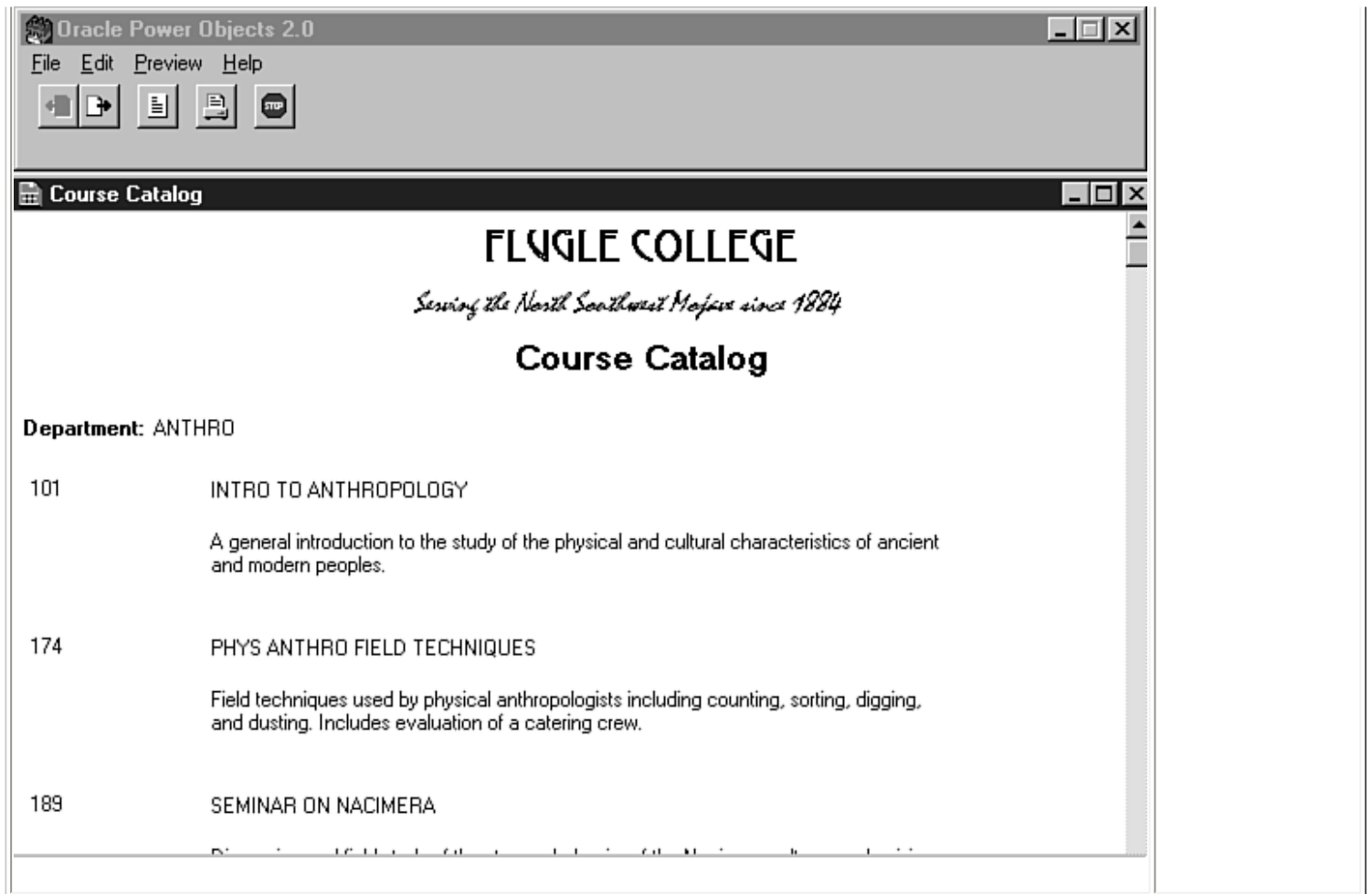
DataSetSize **2000** (dies erlaubt es dem Feld, sich den Inhalten der Description-Spalte anzupassen, die bis zu 2000 Zeichen lang sein können)

MultiLine **True** (dies erlaubt dem Description-Feld bei Bedarf das Einfügen von Zeilenumbrüchen für eine mehrzeilige Ausgabe)

Schließen Sie alle Attributblätter. Zu diesem Zeitpunkt sollte Ihr Bericht wie der in Abbildung 18.22 gezeigte aussehen. Versuchen Sie einen Test des Berichts durch einen Klick mit der rechten Maustaste auf Run Form. Abbildung 18.23 zeigt, wie der abgeschlossene Bericht aussehen sollte.

Wie Sie sehen können, erscheinen die Kurse unterhalb des Departments, das sie anbietet. Beachten Sie außerdem, daß die Symbolleiste für den Laufzeitmodus sich von der für ein Formular unterscheidet. In der oberen linken Ecke gibt es zwei Befehlsschaltflächen, die einen Seitenwechsel auf die vorhergehende und folgende Seite auslösen. Um den Laufzeitmodus zu beenden, klicken Sie auf Stop in der Symbolleiste.





## Hinzufügen eines Suchfeldes zu einem Bericht

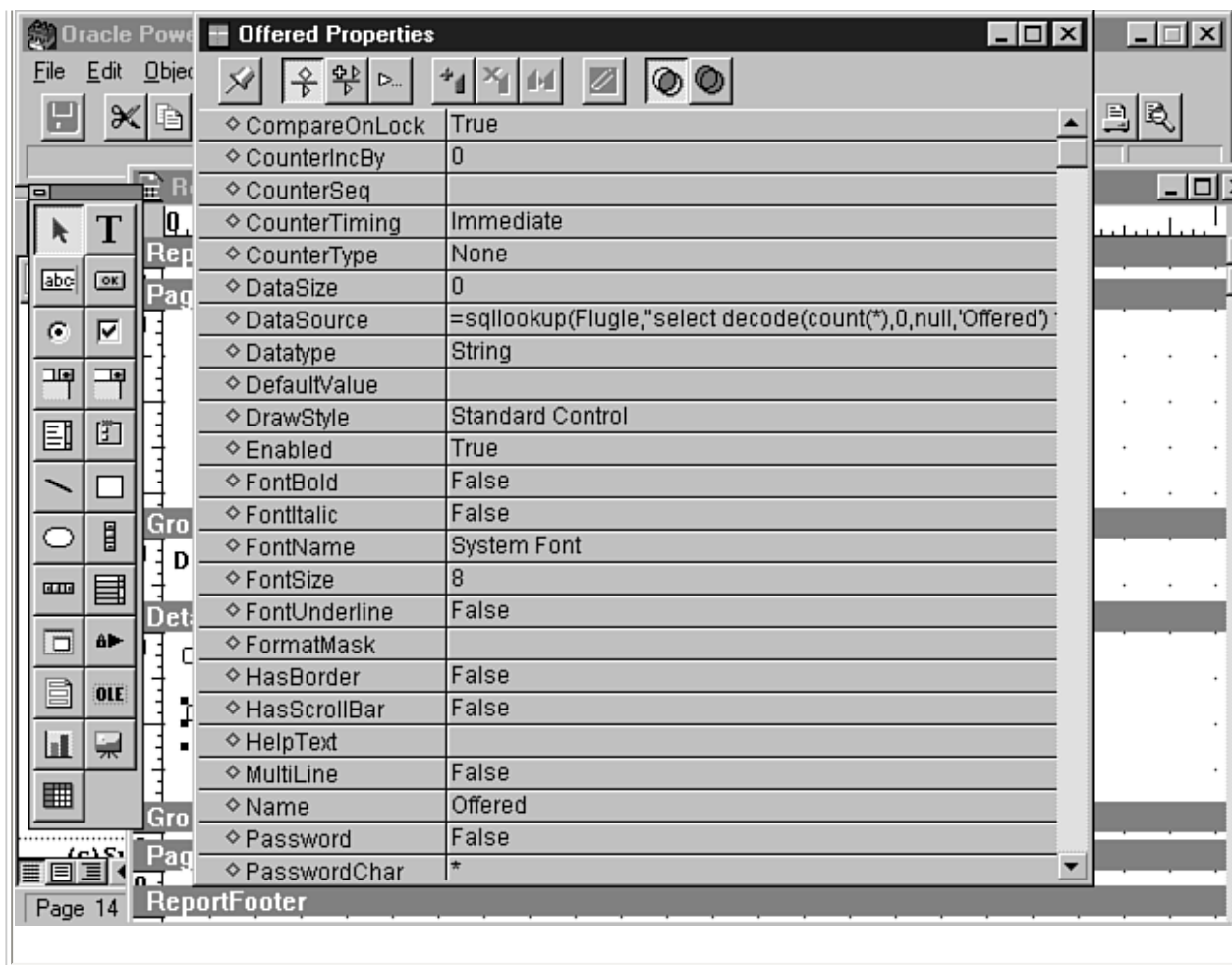
Genauso wie bei Formularen können Sie einem Bericht ein Suchfeld hinzufügen. Nehmen Sie beispielsweise an, Sie wollen dem Flugle College Kurskatalog ein Feld hinzufügen, das festlegt, ob ein Kurs angeboten wurde oder nicht - in anderen Worten, ob er in der Klassen-Tabelle erschienen ist oder nicht. Dazu fügen Sie ein Textfeld in den Detailbereich des Berichts ein. Rufen Sie das Attributblatt für dieses neue Feld auf. Setzen Sie diese Eigenschaften:

Datatype **String**  
 DrawStyle **Standard Control**  
 HasBorder **False**  
 Name **Offered**

Wie in Abbildung 18.24 gezeigt wird, setzen Sie für die DataSource-Eigenschaft folgendes:

```
=sqllookup(Flugle,"select decode(count(*),0,null,'Offered') from Class
where Department_ID = '" + Department_ID.value + "'
and Course_ID = '" + Course_ID.value + "' ")
```

Abbildung 18.24:  
Setzen der  
Eigenschaften für ein  
Suchfeld in einem  
Bericht.



Lassen Sie uns diese Anweisung analysieren. Die SQLLOOKUP-Funktion wird benutzt, um eine SELECT-Anweisung gegenüber der Flugle-Datenbank-Arbeitssitzung auszuführen. Die SELECT-Anweisung verwendet die DECODE-Funktion, um den Wert count(\*) zu dekodieren, welcher der Zahl der Klassen entspricht, die denselben Wert haben wie die Department-ID- und Course-ID-Felder. Falls count(\*) gleich 0 sein sollte, gibt die DECODE-Funktion Null zurück; im Bericht wird nichts passieren. Wenn count(\*) ungleich 0 ist, bedeutet dies, dass für dieses Department und diesen Kurs Klassen angeboten werden, und die DECODE-Funktion die Zeichenfolge Offered zurückgeben wird.

Lassen Sie uns diesen überarbeiteten Bericht testen. Klicken Sie auf Run Form in der Symbolleiste. Blättern Sie auf der ersten Seite des Berichts herunter bis zu den Klassen, die vom Department Biologie angeboten werden. Wie Sie in Abbildung 18.25 sehen können, wird BIO 101 dieses Semester angeboten.

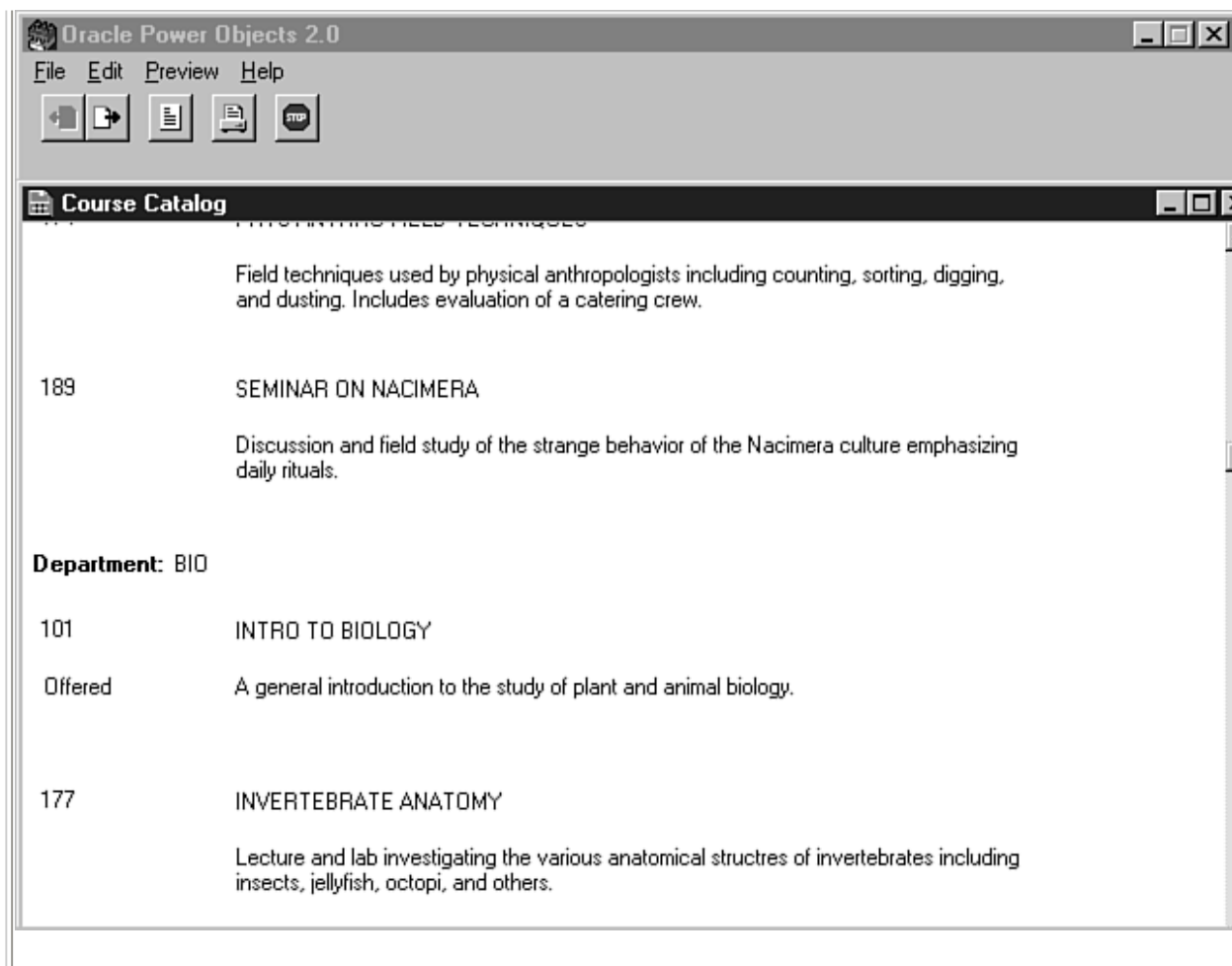


Abbildung 18.25:  
Testen des Suchfeldes in  
dem Bericht.

## Zusammenfassung

In dieser Lektion über Power Objects haben Sie diese wichtigen Konzepte kennengelernt:

- Sie können in Power Objects eine Klasse erstellen. Eine Klasse besteht aus einer Auflistung von einem oder mehreren Objekten, in der Sie die Eigenschaften und Methoden jedes Objekts definieren.
- Um eine Klasse zu benutzen, fügen Sie einem Formular, Bericht oder sogar einer anderen Klasse eine Instanz der Klasse zu. Diese Instanz erbt die Eigenschaften und Methoden von der Klasse, von der sie abgeleitet wurde.
- Sie können eine oder mehrere geerbte Eigenschaften der Instanz einer Klasse in einem Formular, Bericht oder einer anderen Klasse überschreiben.
- Sie können eine oder mehrere geerbte Methoden der Instanz einer Klasse in einem Formular, Bericht oder einer anderen Klasse überschreiben.
- Im Methodencode für die Instanz einer Klasse können Sie gleichzeitig den geerbten Code überschreiben und ihn mit der Anweisung `Inherited. Method_Name()` aufrufen.
- Eine Anwendung kann keine oder mehrere Berichte enthalten.
- Ein Power-Objects-Bericht enthält fünf Bereiche, in denen Sie Objekte wie Statischer Text oder Textfelder platzieren können: ReportHeader, PageHeader, Detail, PageFooter und ReportFooter. Im allgemeinen werden Sie dem Detailbereich Objekte hinzufügen, aber Sie können auch Objekte in den anderen Berichtsabschnitten platzieren.
- Wenn Sie dem Bericht eine Gruppe hinzufügen, werden dem Bericht ebenfalls zwei weitere Bereiche hinzugefügt: GroupHeader und GroupFooter.

## Wie geht es weiter?

An Tag 19, »Eine Übersicht über die Datenbanksicherheit und die Datenbankoptimierung unter Oracle«, werden Sie lernen, wie man Datenbank-Rollen und -Benutzer definiert, Zugriffsrechte vergibt, Synonyme benutzt, den Ausführungsplan für eine SQL-Anweisung bestimmt, und weitere Themen.



## Fragen und Antworten

Frage:

Kann man den geerbten Code für eine Methode in der Instanz einer Klasse einsehen, die zu einem Formular oder Bericht hinzugefügt wurde?

Antwort:

*Unglücklicherweise, nein. Sie können den Methodencode einer Klasse nur einsehen, wenn Sie das Attributblatt der Klasse aufrufen, welches den Code enthält, den Sie sehen wollen.*

Frage:

Kann ein Power-Objects-Bericht so gestaltet werden, daß das Format eines Feldes von der Größe seines Inhalts abhängt?

Antwort:

*Ja. Sie können eine Methode wie z.B. **Validate benutzen, um den Wert eines Feldes zu prüfen und dann eine andere Feld-Eigenschaft ändern, wie z.B. FontBold.***

Frage:

Enthält die Oracle-Power-Objects-2.0-CD-ROM noch andere Produkte?

Antwort:

*Ja! Die Oracle-Power-Objects-2.0-CD-ROM enthält eine Vollversion von Crystal Reports Professional Version 5.0. Sie können Power Objects mit Hilfe Ihres Web-Browser beim Oracle Store unter **www.oracle.com bestellen; diese Produkte sind nicht in der CD-ROM enthalten, die diesem Buch beigelegt ist.***

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

### Test

1. Richtig oder falsch? Die Eigenschaft RecordSource eines Berichts muß eine einzelne Tabelle oder Datensicht referenzieren; sie kann nicht mit zwei oder mehr Tabellen arbeiten.
2. Richtig oder falsch? Power Objects unterstützt ein Maximum von drei Ebenen von Vererbung. Mit anderen Worten, Sie können eine Klasse A erstellen, einer Klasse B eine Instanz von Klasse A hinzufügen, und einer Klasse C eine Instanz von B hinzufügen.
3. Wenn Sie in einer Power-Objects-Anwendung mit Hilfe der Validate-Methode eine Gültigkeitsprüfung von Benutzereingaben durchführen können, warum müssen Sie dann Datenbank-Integritätsregeln und Trigger benutzen, um die Datenintegrität sicherzustellen?

### Übungen

1. Power Objects enthält eine Anzahl von vordefinierten Klassen. Wenn Sie den Inhalt des Verzeichnisses Classes untersuchen, werden Sie eine Klasse namens clsMeter finden. Fügen Sie eine Instanz davon zu MyForm hinzu. Sehen Sie zu, ob Sie herausbekommen, wie man die X-Position des Mauszeigers auf dem Formular mit dieser Anzeige verknüpfen kann.
2. Benutzen Sie Power Objects, um einen Bericht aufzubauen, der eine Liste der Dozenten enthält, die zu jedem Department gehören. Fügen Sie ein Suchfeld zu jeder Dozenten-Zeile hinzu, das anzeigt, ob ein Dozent in dem Semester keinerlei Vorlesungen hält.

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖Kapitel❖ [Top](#) [Inhalt](#) [Index](#) **SAMS** Kapitel❖

# 19

# Datenbanksicherheit und Datenbankoptimierung unter Oracle

[Benutzer und Rollen](#)

[Systemrechte und Objektrechte](#)

[Erstellen und Verwenden einer Rolle](#)

[Einschränken der Ressourcen mit Hilfe eines Profils](#)

[Hindernisse für eine optimale Systemleistung](#)

[Synonyme](#)

[Synonyme, der Navigator und der Schema Manager](#)

[Wo befindet sich der Engpaß?](#)

[Ermitteln des Ausführungsplans für eine SQL-Anweisung mit Hilfe von explain plan](#)

[Zusammenfassung](#)

[Wie geht es weiter?](#)

[Fragen und Antworten](#)

[Workshop](#)

---

Die heutige Lektion behandelt zwei Themen, die während der Entwicklung von Oracle-Datenbankanwendungen nicht ignoriert werden dürfen - Sicherheit und Optimierung. Viele der Aufgaben, die in dieser Lektion zur Sprache kommen, werden üblicherweise vom zuständigen Datenbankadministrator durchgeführt. Auch wenn Sie niemals für die Sicherheit und die Optimierung einer Oracle-Datenbank verantwortlich sein sollten, ist es eine gute Idee, einige der damit verbundenen Sachverhalte verstanden zu haben. Diese Lektion beginnt mit einer Beschreibung von Benutzern, Rollen und der an sie zu vergebenden Berechtigungen.

## Benutzer und Rollen

Während des dritten Tages, »Logischer Datenbankentwurf«, haben Sie die Schritte des Hinzufügens eines neuen Benutzers zu einer Oracle-Datenbank absolviert. Zusätzlich zum Benutzer erlaubt Oracle Ihnen das Erstellen einer Datenbank-Rolle - einem Satz von Oracle-Berechtigungen, die mit der funktionalen Rolle in einem Unternehmen korrespondieren können. Sie können zum Beispiel einem Datenerfasser die Aufgabe des Einfügens und Erneuerns der Datensätze in einer bestimmten Tabelle übertragen, ihn oder sie jedoch am Löschen irgendwelcher Daten hindern. Um dieses Sicherheitsschema zu erzwingen, würden Sie eine Datenbankrolle namens DATA\_ENTRY erstellen, welche diese Berechtigungen vergibt.

Auf der anderen Seite sollte der Dateneingabe-Manager in der Lage sein, Datensätze in die Tabellen einzufügen, sie zu erneuern oder zu löschen. Aus diesem Grund würden Sie eine separate Rolle namens DATA\_ENTRY\_MANAGER erstellen, welche diesen Sicherheitsgrundsatz erfüllt. Ist eine Datenbankrolle erstellt, kann Sie bei Bedarf einer bestimmten Person zugewiesen werden. Zusätzlich kann eine Rolle auch einer anderen Rolle zugewiesen werden. Datenbankrollen helfen nicht nur, die entsprechende Sicherheit zu bieten, sie vereinfachen auch die Datenbankadministration.

Für eine Anwendungsumgebung mit vielen Tabellen, Benutzern und funktional klaren Rollen ist die Einführung und Wartung eines Sicherheitsschemas eine Hauptaufgabe. Wenn der Datenbankadministrator Berechtigungen direkt an einzelne Benutzer vergibt, ist es sehr leicht möglich, daß einige Dinge im Laufe der Zeit übersehen werden. Der DBA könnte vergessen, einigen Mitarbeitern die Berechtigung zu erteilen, die sie für ihre Arbeit benötigen, oder er könnte anderen eine überflüssige Berechtigung erteilen. Falls eine Änderung der Datenbank-Anwendung ebenso eine Änderung der Benutzerberechtigungen notwendig macht, wird der DBA ohne die Verwendung von Rollen weit mehr Arbeit zu erledigen haben.

Indem Sie eine Datenbankrolle für jede funktionale Rolle definieren, die in Ihrem Unternehmen existiert, reduzieren Sie die Aufgabe des Verwaltens von Benutzern und deren Berechtigungen deutlich. Nehmen Sie beispielsweise an, es gibt zehn Datenbankbenutzer - drei gehören zur Marketing-Abteilung und die anderen sieben zur Entwicklungsabteilung. Sie würden zwei Datenbankrollen erstellen, Marketer und Engineer, und die entsprechenden Berechtigungen jeder Rolle zuweisen; die Marketer-Rolle benötigt den Zugriff auf einen Satz von Tabellen, während die Engineers-Rolle den Zugriff auf einen anderen Satz benötigt. Anschließend würden Sie jedem Benutzer die entsprechende Rolle zuweisen. Wenn sich die Datenbank ändert, z.B. durch Hinzufügen einer Tabelle, erneuern Sie einfach nur die Berechtigungen der Rolle, nicht die der Benutzer.

### Vordefinierte Benutzer: sys und system

Jede Oracle-Datenbank besitzt zwei vordefinierte Accounts, die einen speziellen Zweck erfüllen - SYS und SYSTEM. Der SYS-Account umfaßt die Oracle Data Dictionary-Tabellen und die damit verbundenen Datenbankobjekte. Der SYSTEM-Account umfaßt Tabellen, die von den Oracle Anwendungsentwicklungs-Werkzeugen wie Oracle Forms und Reports benutzt werden. Sie sollten keinerlei Datenbankobjekte wie Tabellen oder Indizes erzeugen, solange Sie als SYS oder SYSTEM angemeldet sind, mit Ausnahme für die von Oracle unterstützten SQL-Skripte, die speziell festlegen, daß sie als SYS oder SYSTEM installiert werden sollten.

### Vordefinierte Rollen

Wenn eine Oracle-Datenbank installiert ist, sind diese fünf Datenbankrollen bereits erstellt:

- CONNECT besitzt die Möglichkeit zur Verbindung mit der Personal-Oracle-Datenbank (mittels der Systemberechtigung CREATE SESSION).
- RESOURCE besitzt die Möglichkeit zur Erstellung von Tabellen, Indizes, Datenansichten und anderer Oracle-Objekte.
- DBA besteht aus allen Systemberechtigungen, die zur Erstellung, Änderung oder dem Ausschluß von Benutzern und der Verwaltung der zugehörigen Datenbankobjekte benötigt werden.
- IMP\_FULL\_DATABASE besteht aus allen Systemberechtigungen, die zur Durchführung eines vollständigen Datenbankimports benötigt werden.
- EXP\_FULL\_DATABASE besteht aus allen Systemberechtigungen, die zur Durchführung eines vollständigen Datenbankexports benötigt werden.



*Der historische Grund für diese vordefinierten Rollen ist, daß Oracle in der Version 6 nur drei Berechtigungen besaß - connect, resource und DBA. Weil Oracle diese Berechtigungen in Oracle 7 als Rollen neu definiert hat, können für Oracle Version 6 entwickelte SQL\*Plus-Skripte in Oracle 8 ausgeführt werden. Diese Rollen vereinfachen außerdem die Zuweisung von Systemberechtigungen an andere Rollen und Benutzer.*

## Systemrechte und Objektrechte

Eine Rolle ist nicht von Nutzen, solange sie nicht mindestens eine Berechtigung erteilt. Oracle-Berechtigungen gehören zu zwei Kategorien:

- *System-Berechtigungen* erlauben einem Oracle-Benutzer die Ausführung von Anweisungen wie CREATE TABLE, CREATE USER oder ALTER INDEX.
- *Object-Berechtigungen* stehen in Zusammenhang mit einer bestimmten Operation (so wie SELECT oder UPDATE) an einem festgelegten Datenbankobjekt (so wie einer Tabelle oder einem Index).

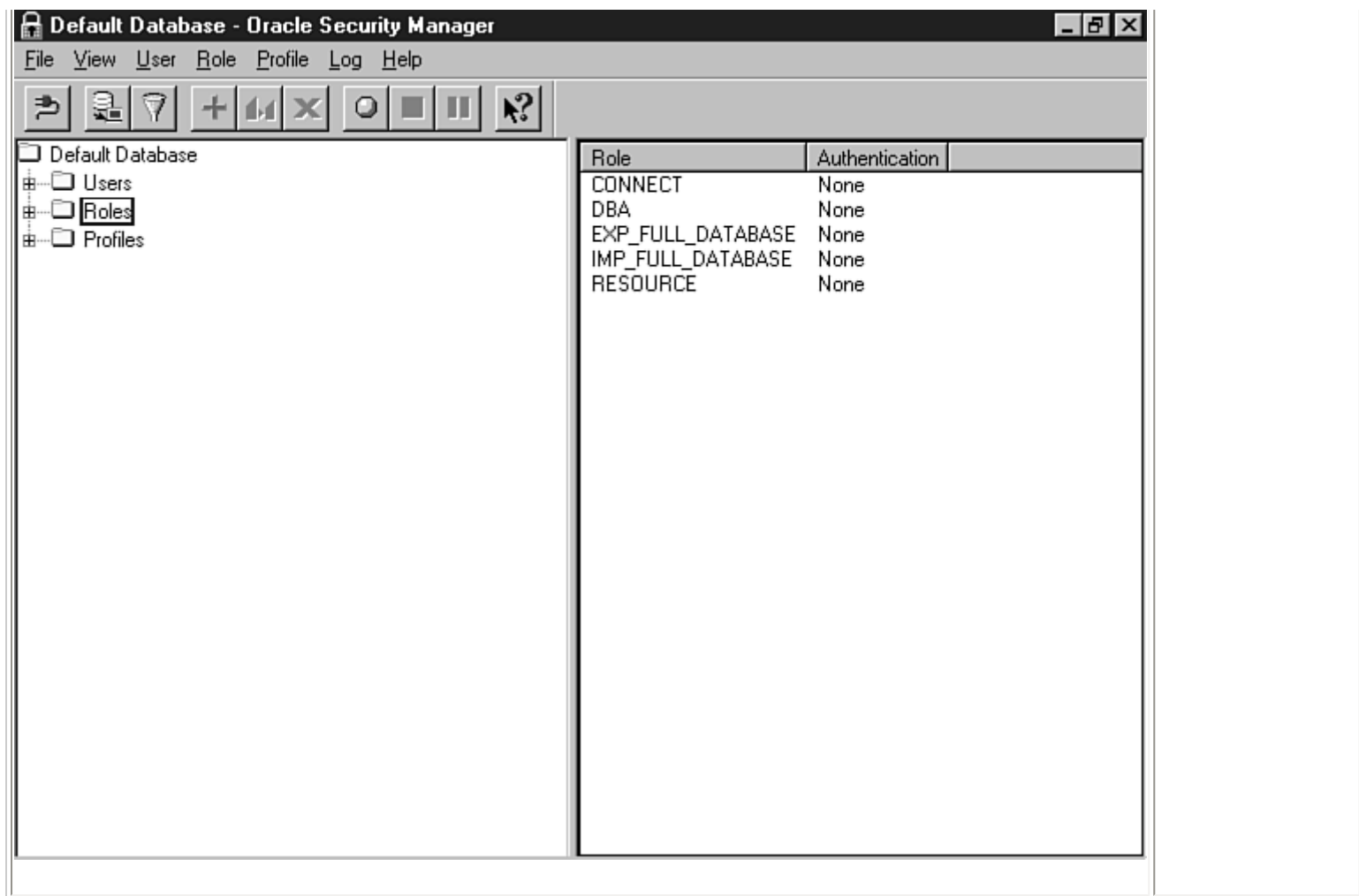
Sie können einer Rolle oder einem Benutzer viele verschiedene Berechtigungen zuweisen - tatsächlich besitzt Oracle mehr als 80 Systemberechtigungen. Eine Systemberechtigung, die nahezu jeder Oracle-Benutzer braucht, ist CREATE SESSION - die Möglichkeit, eine Oracle-Arbeitssitzung aufzubauen.

## Erstellen und Verwenden einer Rolle

Um diese Konzepte zu konkretisieren, geht dieser Abschnitt durch den Prozeß des Erstellens einer Rolle und ihrer Zuweisung an eine bestimmte Person. Sie werden der Rolle außerdem Systemberechtigungen zuweisen. Dazu werden Sie einen Account benötigen, dem die Rolle DBA erteilt wurde. Bitte seien Sie sich bewußt, daß das, was Sie auf Ihrem Bildschirm sehen, sich von den Abbildungen in dieser Lektion unterscheiden kann, weil Sie eventuell bereits Benutzer und Rollen definiert haben.

1. Um zu beginnen, rufen Sie den Security Manager aus der Oracle Enterprise Manager-Programmgruppe auf.
2. Geben Sie den Benutzernamen, das Kennwort und den Service in entsprechender Weise ein.
3. Nachdem der Security Manager die Verbindung zur Oracle-Datenbank hergestellt hat, klicken Sie das Verzeichnis Roles im linken Fenster an.
4. Im rechten Fenster sollten Sie die bereits existierenden Datenbankrollen sehen (siehe Abbildung 19.1).

*Abbildung 19.1:  
Betrachtung  
vorhandener  
Datenbankrollen mit  
dem Security  
Manager*



5. Um eine neue Rolle zu erstellen, können Sie die Befehlsschaltfläche + auf der Symbolleiste betätigen oder den Menüpunkt Role | Create wählen. Ein Dialogfenster wird angezeigt, das den Namen der neuen Rolle erfragt (siehe Abbildung 19.2).

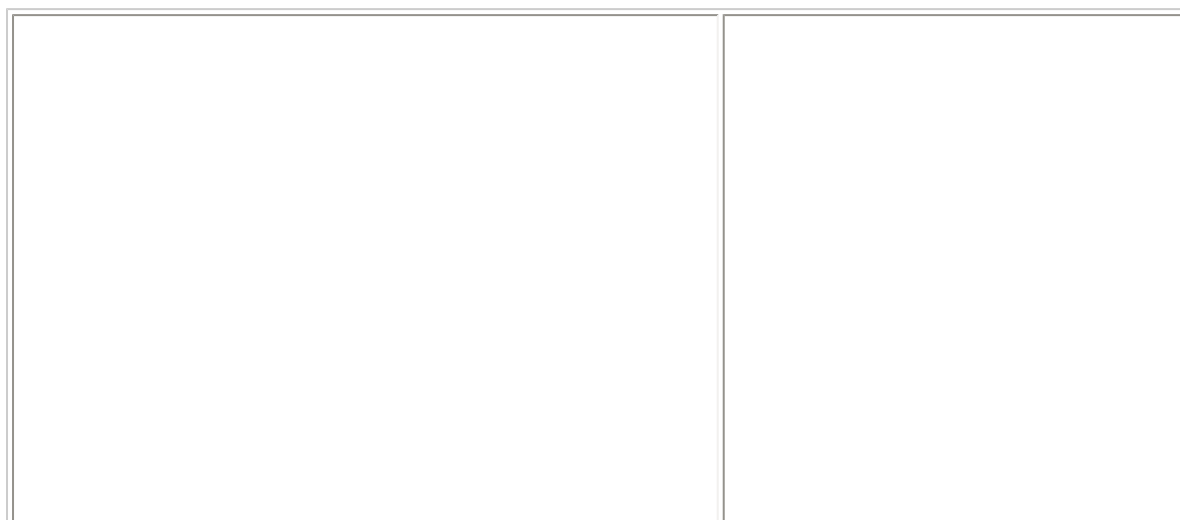
6. Geben Sie **Department\_Administrator** ein.

7. Belassen Sie die Authentication-Optionsschaltfläche in der Stellung None.

Vergeben Sie als nächstes die Rollen CONNECT und RESOURCE an die neue Datenbankrolle.

1. Betätigen Sie die Registermarke namens Privileges. Standardmäßig zeigt der Security Manager diejenigen Rollen an, die der neuen Datenbankrolle gegenwärtig zugeteilt sind. Am unteren Ende des Dialogfensters sehen Sie die bestehenden Rollen.

2. Halten Sie die Strg-Taste gedrückt, wählen Sie CONNECT und RESOURCE, und klicken Sie auf Add (siehe Abbildung 19.3). Sie sollten jetzt die Rollen CONNECT und RESOURCE im oberen Bereich des Dialogfensters sehen.



**Create Role - Default Database**

General | Privileges

Role:

Authentication

☒ None

☐ External

☐ Password

Confirm Password

Create Cancel Show SQL Help

Abbildung 19.2:  
Erzeugen einer neuen Rolle mit dem  
Security Manager

**Create Role - Default Database**

General | Privileges

Privilege Type:

| Current  | Admin Option |
|----------|--------------|
| CONNECT  | NO           |
| RESOURCE | NO           |

Delete

Roles:

CONNECT  
DBA  
EXP\_FULL\_DATABASE  
IMP\_FULL\_DATABASE  
RESOURCE

☐ With Admin Option Add

Create Cancel Show SQL Help

Abbildung 19.3:  
Vergabe der Rollen connect und resource  
an die neue Rolle

3. Haben Sie die Befehlsschaltfläche Create angeklickt, ist damit die Rolle Department\_Administrator erzeugt, und Sie sollten sie in der Liste der Datenbankrollen auf der rechten Seite des Fensters sehen.
4. Fügen Sie jetzt der Datenbank einen Benutzer hinzu - Terri Greenglass, den Administrator der englischen Abteilung.
5. Klicken Sie das Verzeichnis User und anschließend die Befehlsschaltfläche + an.
6. Geben Sie in das Feld User **tgreenglass** ein.
7. Geben Sie in die Felder Password und Confirm das Kennwort ein, welches Sie dem neuen Benutzer zu geben wünschen (siehe Abbildung 19.4).

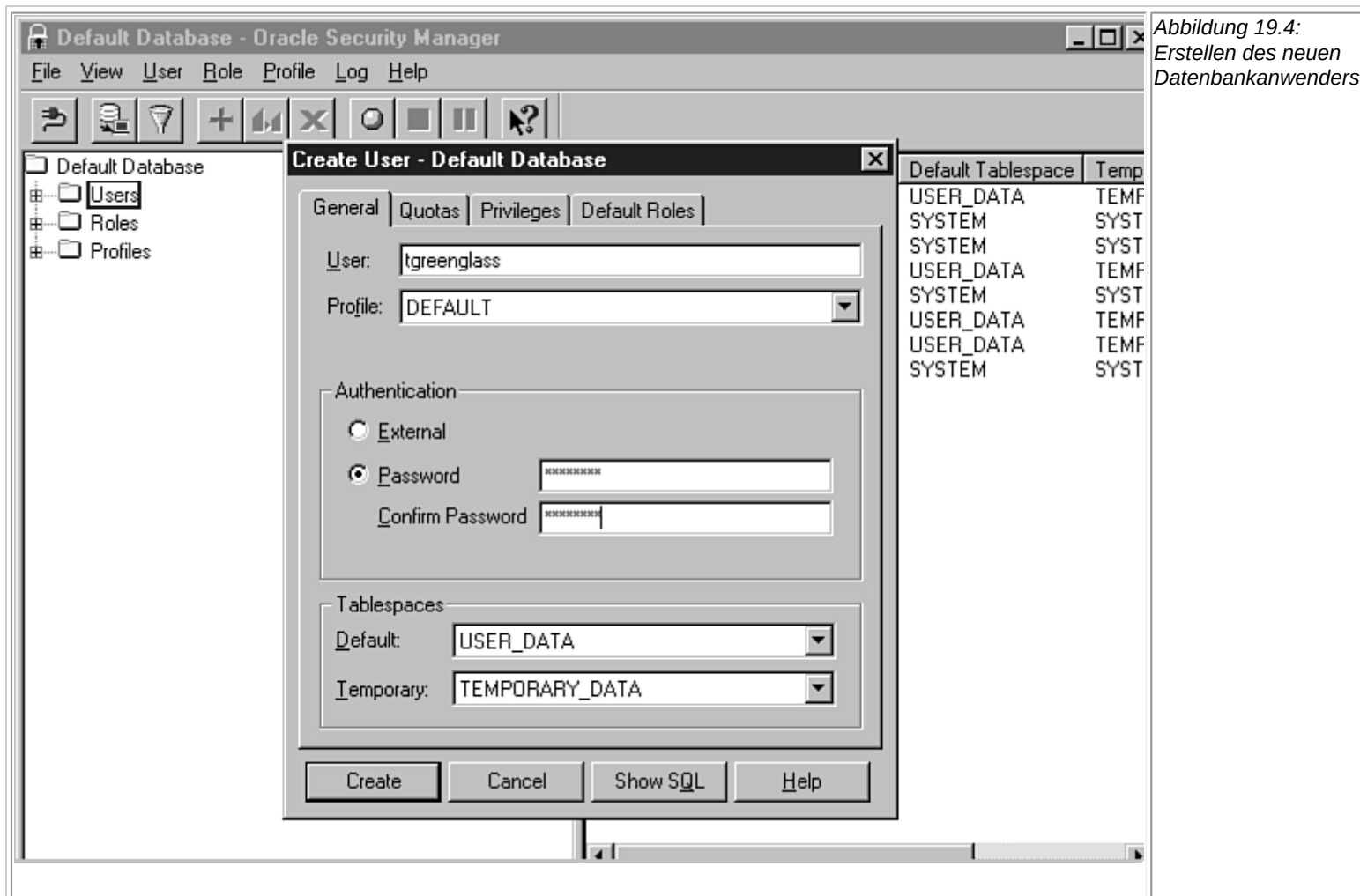


Abbildung 19.4:  
Erstellen des neuen  
Datenbankanwenders

8. In diesem Beispiel können Sie den Abschnitt Tablespaces des Fensters ignorieren.

9. Um die Rolle Department\_Administrator Teri Greenglass zu erteilen, wählen Sie die Registerkarte Privileges.

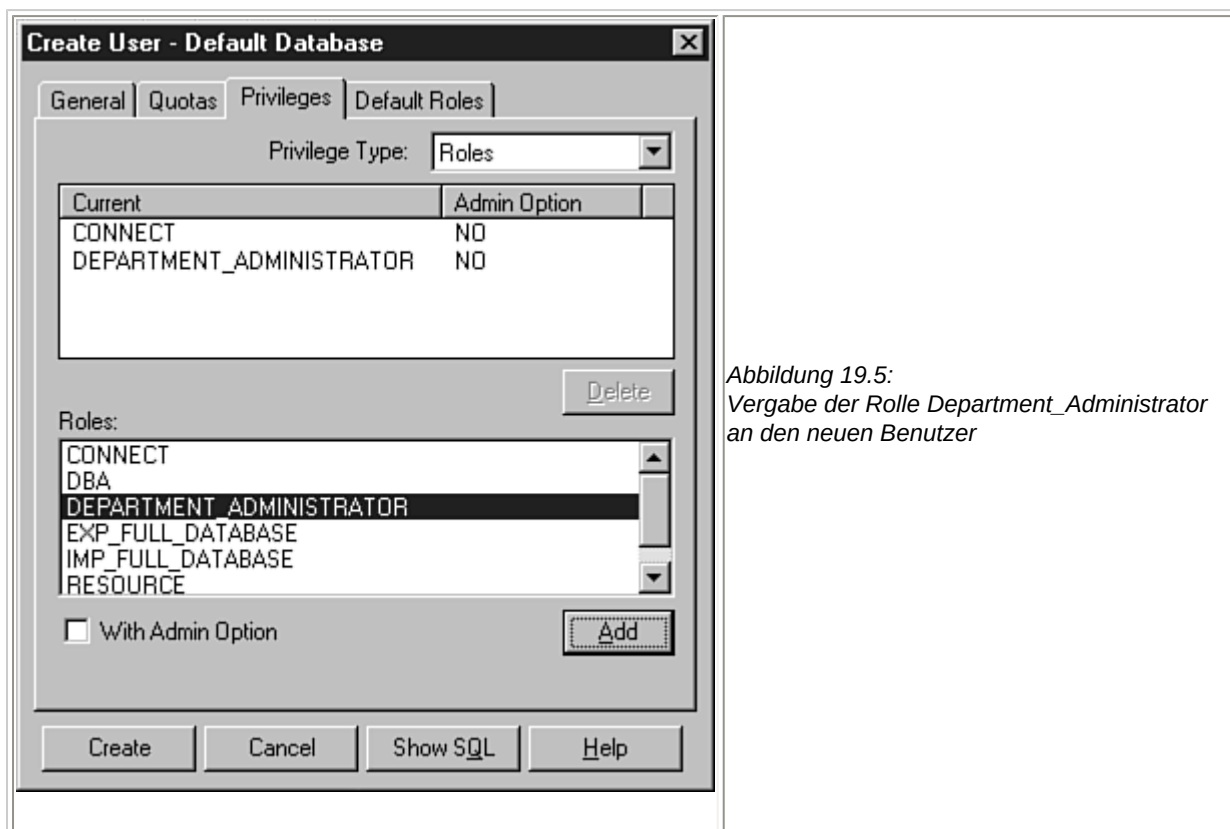


Abbildung 19.5:  
Vergabe der Rolle Department\_Administrator  
an den neuen Benutzer



10. Wählen Sie mit Hilfe der Maus die Rolle `Department_Administrator`, und klicken Sie die Add-Befehlsschaltfläche an (siehe Abbildung 19.5).

11. Um die Erstellung des Oracle-Accounts für Terri Greenglass abzuschließen, klicken Sie die Create-Befehlsschaltfläche. Der Security Manager übermittelt die SQL-Anweisungen an die Oracle-Datenbank, um den neuen Benutzer zu erstellen und die Rolle zu erteilen.

## Einschränken der Ressourcen mit Hilfe eines Profils

Zusätzlich zur Freigabe eines Speicherkontingents in einem Tablespace bietet Oracle auch andere Systemlevel-Kontingente, die einem Benutzer zur Verfügung gestellt werden können. Diese Quotas schließen folgendes ein:

- `CONNECT_TIME` ist die Zahl der abgelaufenen Minuten, seit der Benutzer die Verbindung zur Oracle-Datenbank aufgebaut hat.
- `IDLE_TIME` ist die Zahl der abgelaufenen Minuten, seit der Benutzer zum letzten Mal eine Aktion ausgeführt hat, die zu einem Oracle-Aufruf geführt hat.
- `LOGICAL_READS_PER_CALL` ist die Zahl der Oracle-Blöcke, die aus dem Speicher oder von einem Laufwerk gelesen wurden, um einen Oracle-Aufruf zu erfüllen.

Der Mechanismus, den Sie für die Spezifizierung dieser Kontingente festlegen, ist das *Profil*. Sie könnten beispielsweise für einen Gelegenheitsanwender ein Profil erstellen, das seine oder ihre Verbindungszeit auf zwei Stunden oder weniger begrenzt. Darüber hinaus könnten Sie außerdem befürchten, daß ein Gelegenheitsbenutzer - insbesondere jemand, der Zugriff zu einem ad-hoc-Abfragewerkzeug hat - versehentlich Abfragen konstruieren kann, die ein kartesisches Produkt aus verschiedenen Tabellen zur Folge haben. Indem Sie das Limit von `LOGICAL_READS_PER_CALL` auf einen ziemlich hohen Wert begrenzen, können Sie die Benutzerabfrage von der Inanspruchnahme von Speicher und Rechenzeit abhalten, die von anderen Benutzern benötigt wird.

Um die in einem Benutzerprofil festgelegten Ressourcen-Beschränkungen zu erzwingen, muß der DBA eine der folgenden Aufgaben ausführen:

- Setzen Sie den Initialisierungsparameter für `resource_limit` auf `TRUE`. Der neue Wert für `resource_limit` wird erst nach einem Neustart der Datenbank wirksam.
- Passen Sie die Datenbank an, indem Sie den folgenden Befehl in SQL\*Plus eingeben: `ALTER SYSTEM SET RESOURCE LIMIT = TRUE`; oder indem Sie in der Datei `init.ora` `RESOURCE_LIMIT=TRUE` setzen.

## Hindernisse für eine optimale Systemleistung

Eine der Ziele dieser Lektion ist die Benennung von Performance-Merkmalen im Zusammenhang mit einer Oracle-Datenbank. Das Material dieser Lektion ist eine Einführung in einige der Punkte, die Ihnen während des Datenbank-Entwicklungsprozesses begegnen werden. Hier sind einige allgemeine Vorschläge.

### Was Sie tun sollen

### Was nicht

**Kaufen Sie** so viele Laufwerke und Speicher, wie Sie sich leisten können.

**Wählen Sie** die geeignete Plattform - Hardware und Betriebssystem - für die Anforderungen der Anwendungen.

**Verbessern Sie** die Performance, indem Sie die entsprechende Hardware kaufen, anstelle des Designs und des Schreibens komplexerer Client-Server-Logik.

**Denormalisieren Sie** ein logisches Datenbankdesign nicht, ohne durch Prototyping und Tests sicherzustellen, daß die Abfrage-Performance ausreichend ist.

Behalten Sie diese Richtlinien im Gedächtnis, wenn Sie etwas über die Performance-Charakteristiken lesen, die in einem Oracle-Client-Server-Umfeld auftreten können.

## Synonyme

Ein Synonym ist ein anderer Name für eine Tabelle. Eigentlich ist der Begriff *Tabellen-Synonym* genauer. Synonyme gibt es in zwei Gültigkeitsbereichen: privat und öffentlich. Ein privates Synonym ist nur für den Oracle-Benutzer sichtbar, der es erstellt hat. Ein öffentliches Synonym ist für alle Oracle-Benutzer sichtbar. Jeder Oracle-Benutzer, dem die RESOURCE-Rolle erteilt wurde, kann ein privates Synonym erzeugen. Auf der anderen Seite können nur Oracle-Benutzer, denen die DBA-Rolle erteilt wurde - oder die Systemberechtigung Create Public Synonym -, ein öffentliches Synonym erstellen.



*Ein Synonym ist ein anderer Name für eine Tabelle oder eine Datenansicht. Ein Synonym, das nur für seinen Erzeuger sichtbar ist, wird als privates Synonym bezeichnet. Ein Synonym, das für alle Oracle-Benutzer sichtbar ist, wird als öffentliches Synonym bezeichnet.*

In einer Hinsicht ist ein Synonym einer Datenansicht ähnlich: beide Objekte erlauben es einer Tabelle, unter einem anderen Namen referenziert zu werden. Ein Synonym ermöglicht es Ihnen jedoch nicht, Spalten einzuschränken oder sie neu zu benennen.

Ein Synonym bietet einen zusätzlichen Namen für die Referenzierung einer Tabelle. Sie können zum Beispiel eine Tabelle nicht neu benennen, weil existierende Anwendungen sich auf den gegenwärtigen Tabellennamen beziehen. Ein Synonym, das einen etwas intuitiveren und sinnvollen Namen bietet, könnte ideal für die Nutzung in einem ad-hoc-Abfragewerkzeug sein.

### Die Syntax eines Synonyms

Sie sollten sich ansehen, wie Sie Synonyme mit dem Personal Oracle Navigator oder mit dem Schema Manager verwalten können. Es gibt jedoch einen Vorteil bei der Verwendung von SQL-Skripten für das Erstellen und Entfernen von Synonymen.



*Werfen Sie deshalb einen Blick auf die Syntax für das Erzeugen eines Synonyms:*

```
CREATE [PUBLIC] SYNONYM synonym-name  
FOR owner.object-name;
```

Die Variablen sind wie folgt definiert:

- *synonym-name* ist der Name des Synonyms und muß den Anforderungen der Namensvergabe an Objekte einer Oracle-Datenbank genügen.
- *owner* ist der Name des Oracle-Accounts, der die referenzierte Tabelle oder Datenansicht besitzt.
- *object-name* ist der Name der Tabelle oder Datenansicht, die von dem Synonym referenziert wird.

Sie könnten ein Synonym erstellen wollen, das eine Tabelle referenziert, deren Name unpassend ist oder der sich schwer merken läßt. In diesem Beispiel wird das Synonym p\_artifact benutzt, um auf die Tabelle parthaginian\_artifacts zu zeigen, die zu demselben Oracle-Benutzer gehört:

```
SQL> create synonym p_artifact for parthaginian_artifacts;
```

## Synonym created

Ein privates Synonym kann auch auf eine Tabelle verweisen, die im Besitz eines anderen Oracle-Benutzers ist. Nehmen Sie einmal an, Oracle-Benutzer RJOHNSON hat die Projekttabelle erstellt und will es dem Benutzer KCHOW ermöglichen, diese Tabelle ebenfalls zu lesen. Als erstes erteilt RJOHNSON die Select-Berechtigung für die Projekttabelle an KCHOW. Jedesmal jedoch, wenn KCHOW sich die Tabelle ansehen will, muß sie daran denken, sich für die Tabelle mit dem Namen RJOHNSON zu qualifizieren - dem Besitzer der Tabelle. Sie erstellt deshalb konsequenterweise ein privates Synonym, das es ihr erlaubt, die Tabelle allein mit dem Projektnamen zu referenzieren, wie in Listing 19.1 gezeigt wird.

### *Listing 19.1: Verwendung eines privaten Synonyms*

```
SQL> select Project_Number
2 from Project;
from Project
*
ERROR at line 2:
ORA-00942: table or view does not exist

SQL> create synonym Project for RJOHNSON.Project;

Synonym created.

SQL> select Project_Number
2 from Project;

PROJECT_NUMBER
-----
1201
2143
4310
```

## Löschen von Synonymen

Oracle bietet eine Anweisung für das Löschen eines Synonyms - die DROP SYNONYM-Anweisung.



*Die Syntax lautet wie folgt:*

---

```
DROP [PUBLIC] SYNONYM synonym-name;
```

Die Variable ist wie folgt definiert:

- *synonym-name* ist der Name des existierenden Synonyms, das Sie löschen wollen.

## Verbergen des Besitzes einer Tabelle mit Hilfe öffentlicher Synonyme

Betrachten Sie das folgende Beispiel. Sie haben eine projektbezogene Anwendung in Ihrem Unternehmen entwickelt. Sie müssen jetzt zwei Gruppen von Benutzern unterstützen: diejenigen, die Version 1.0 benutzen, und diejenigen mit Version 2.0. Die Version 2.0 erfordert jedoch einige Änderungen an der Datenbank; die von Version 1.0 benutzte Datenbank kann nicht von der Version 2.0 verwendet werden. Die von beiden Versionen benutzten Tabellen können in derselben Oracle-Datenbank gespeichert werden, unter der Voraussetzung, daß sie im Besitz von verschiedenen Oracle-Accounts sind.

Angenommen, daß die Tabellen der Version 1.0 im Besitz eines Oracle-Accounts namens PAV10 sind. Die Tabellen der Version 2.0 sind im Besitz eines anderen Oracle-Accounts mit Namen PAV20. Falls Sie eine Gruppe von Software-Testern unterstützen wollen, die zwischen den beiden Versionen hin- und herwechseln müssen, können Sie zwei SQL\*Plus-Skripte erstellen. Das erste Skript löscht die existierenden Synonyme und erstellt neue, die auf die Tabellen der Version 1.0 verweisen, wie in Listing 19.2 gezeigt wird.

**Listing 19.2: Verweis auf die erste Version der Tabellen mit Hilfe von Synonymen**

```
drop synonym Account_Number;  
...  
drop synonym Task_Header;  
drop synonym Task_Detail;  
...  
...  
create synonym Account_Number for PAV10.Account_Number;  
...  
create synonym Task_Header for PAV10.Task_Header;  
create synonym Task_Detail for PAV10.Task_Detail;
```

Das zweite Skript, gezeigt in Listing 19.3, löscht ebenfalls die existierenden Synonyme, die neu erstellten verweisen jedoch auf die Tabellen der Version 2.0.

**Listing 19.3: Verweis auf die zweite Version der Tabellen mit Hilfe von Synonymen**

```
drop synonym Account_Number;  
...  
drop synonym Task_Header;  
drop synonym Task_Detail;  
...  
...  
create synonym Account_Number for PAV20.Account_Number;  
...  
create synonym Task_Header for PAV20.Task_Header;  
create synonym Task_Detail for PAV20.Task_Detail;
```

Mit diesen Skripten kann ein Benutzer zwischen den beiden projektbezogenen Versionen hin- und herwechseln.

## Synonyme, der Navigator und der Schema Manager

Personal Oracle für Windows 95 ermöglicht Ihnen die Manipulation von Synonymen mit dem Navigator. Um eine Liste der in Ihrer lokalen Datenbank existierenden Synonyme einzusehen, doppelklicken Sie auf das Synonym-Verzeichnis. Eine Liste aller Synonyme wird auf der rechten Seite des Hauptfensters angezeigt (siehe Abbildung 19.6)

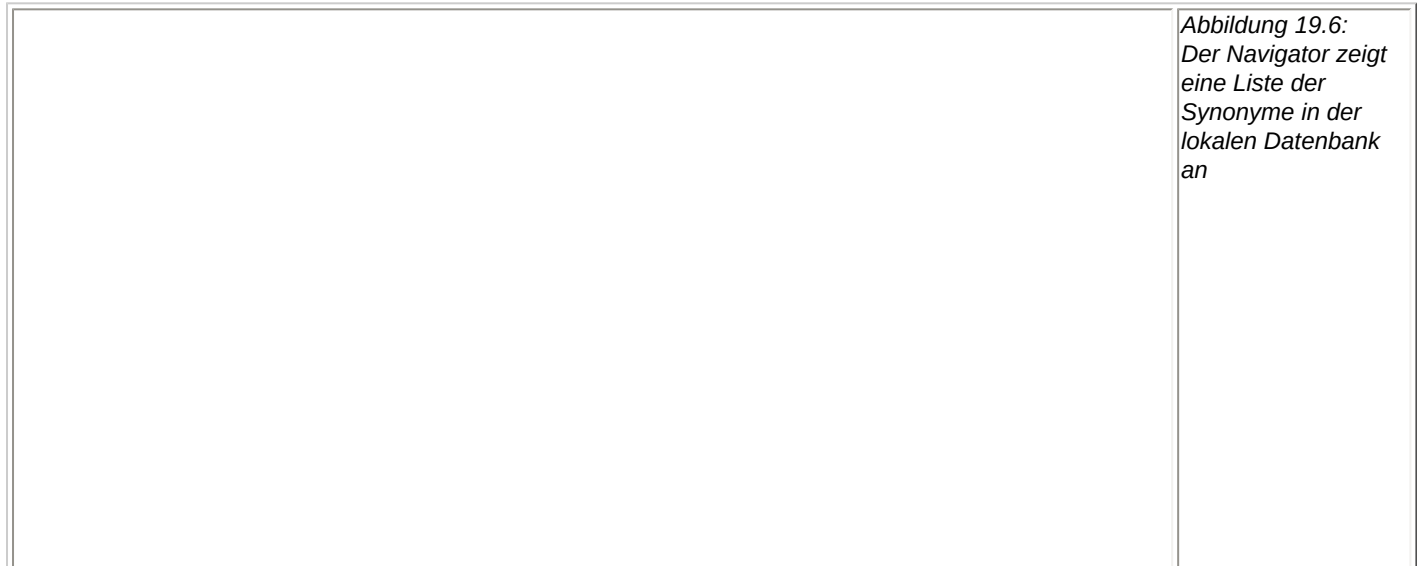
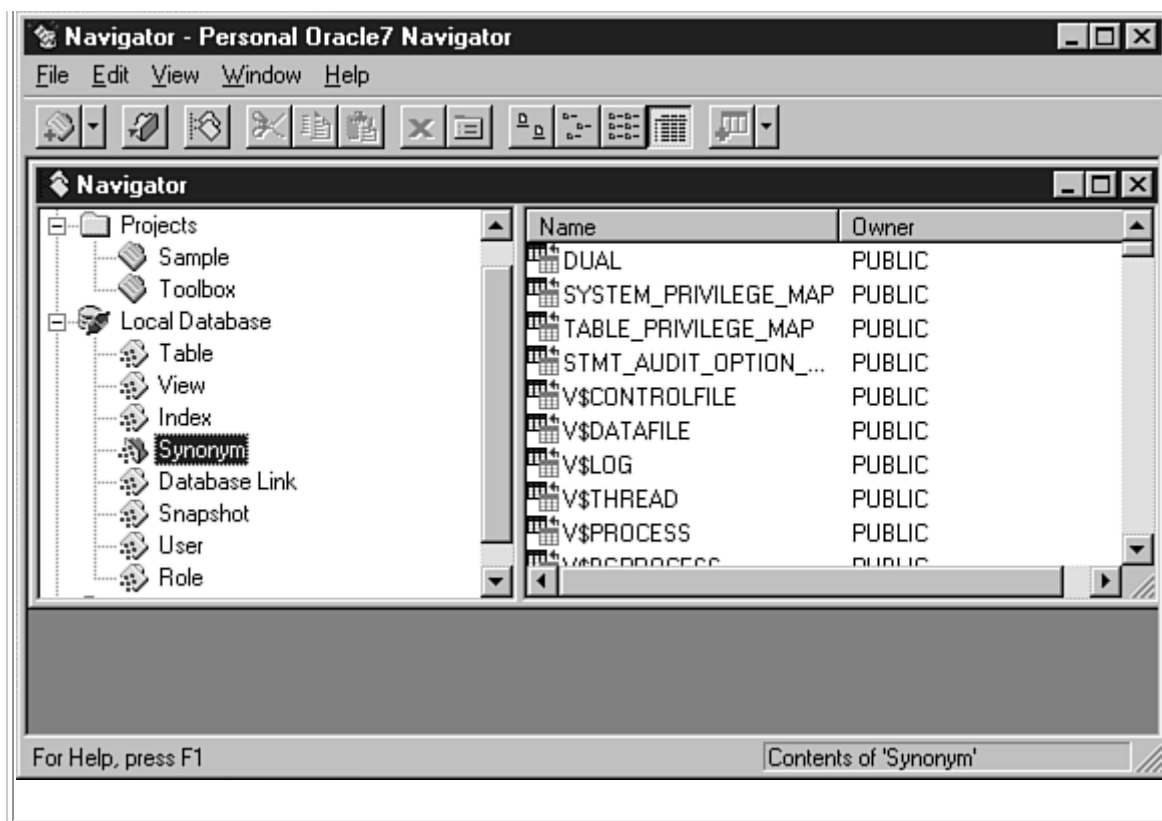


*Abbildung 19.6:  
Der Navigator zeigt  
eine Liste der  
Synonyme in der  
lokalen Datenbank  
an*



Wenn Sie eine Oracle-Server benutzen, können Sie den Schema Manager verwenden, um private und öffentliche Synonyme zu verwalten. Tatsächlich können Sie den Schema Manager auch mit einer Personal Oracle-Datenbank benutzen.

### Ein Synonym mit dem Navigator untersuchen

Um die Definition eines Synonyms zu untersuchen, wählen Sie das Synonym und klicken mit der rechten Maustaste, um das Attributblatt zu öffnen. Der Navigator zeigt ein Fenster, das den Namen des Synonyms, den Besitzer und die referenzierte Tabelle (oder Datenansicht) identifiziert (siehe Abbildung 19.7). Allerdings können Sie bei sehr langen Besitzer- und Tabellennamen eventuell den vollständigen Tabellennamen nicht lesen.

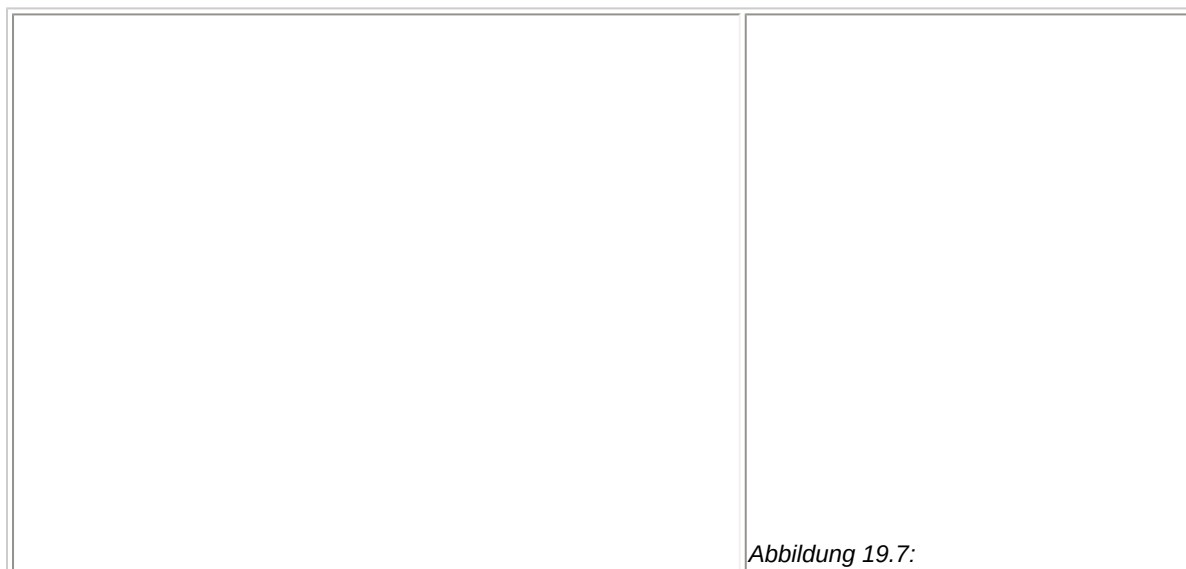
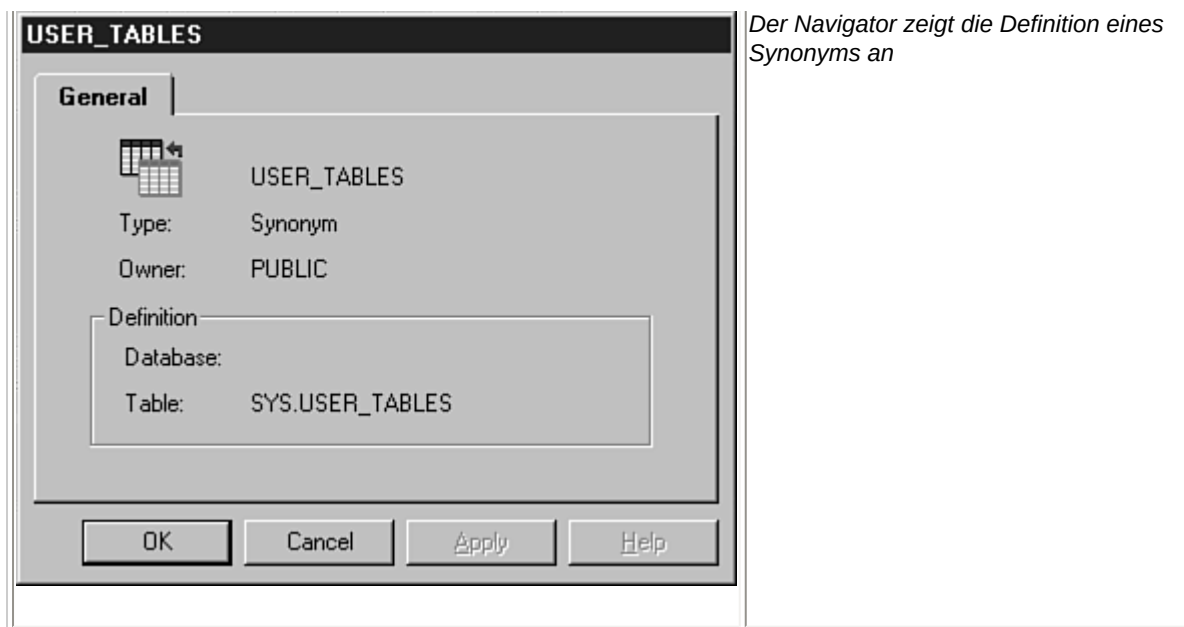


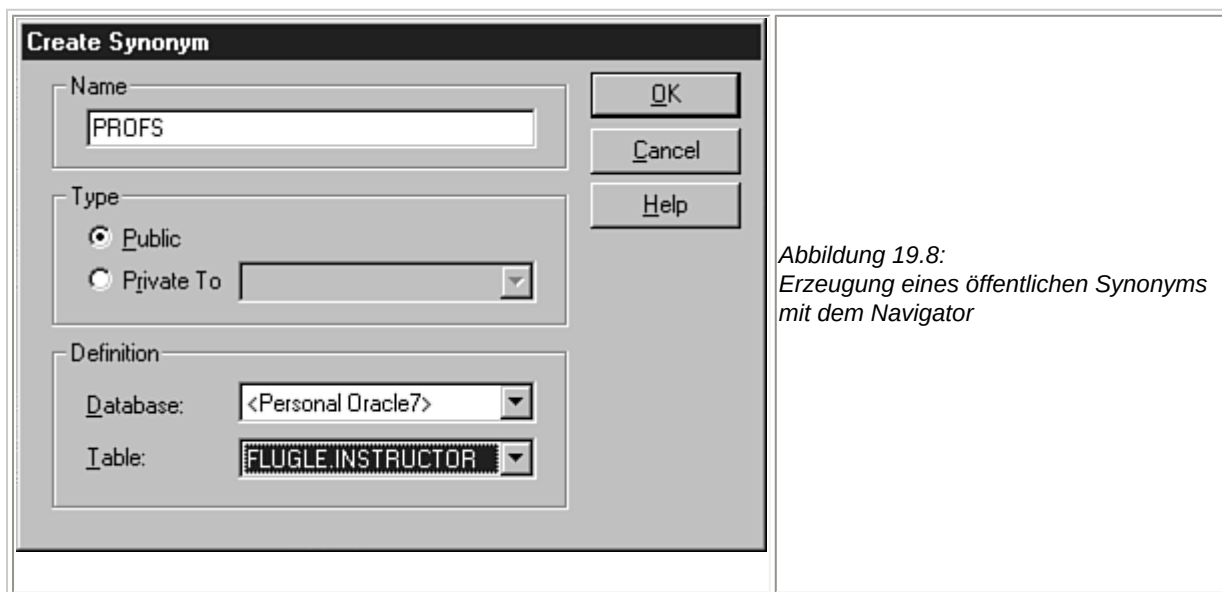
Abbildung 19.7:



## Ein Synonym mit dem Navigator erzeugen

Folgen Sie diesen Schritten, um mit dem Navigator ein Synonym zu erstellen:

1. Wählen Sie das Synonym-Verzeichnis, klicken Sie mit der rechten Maustaste und anschließend New.
2. Geben Sie im Create Synonym-Fenster einen Namen für das neue Synonym in das Feld Name ein.
3. Sie haben die Wahl für den Typ des Synonyms - es kann entweder öffentlich oder privat sein.
4. Wählen Sie Personal Oracle im Database-Feld und die Tabelle aus dem Dropdown-Listenfeld im Feld Table. Klicken Sie auf OK, um das Synonym zu erzeugen. Abbildung 19.8 zeigt, wie ein öffentliches Synonym erstellt wurde.
5. Wenn Sie ein privates Synonym erzeugen wollen, müssen Sie die Optionsschaltfläche Private To wählen und den Benutzer angeben, der das Synonym in der Dropdown-Liste rechts davon besitzen wird.



## Ein Synonym mit dem Navigator löschen

Um ein Synonym zu löschen, wählen Sie es aus, klicken mit der rechten Maustaste und anschließend Delete. Der Navigator wird Sie nach Ihrer Zustimmung fragen, ob das Synonym tatsächlich gelöscht werden soll.

## Wo befindet sich der Engpaß?

Benutzer sind die besten Richter über die Performance eines Informationssystems. Sie messen die Leistung in Begriffen wie Antwortzeit, Bearbeitungszeit eines Berichts und Datenübertragungsgeschwindigkeit. Ein sehr ineffizientes System kann von Benutzern als ein System mit exzellenter Bearbeitungszeit wahrgenommen werden. Umgekehrt könnte eine Gruppe von Benutzern die Performance eines sehr effizienten Informationssystems für unzureichend halten. Ignorieren Sie die Wahrnehmungen der Benutzer auf eigene Gefahr.

Nehmen Sie an, Sie müssen ein System analysieren, welches das Objekt von Beschwerden über mangelhafte Leistung ist. Sie werden vier Elemente einer Client-Server-Architektur untersuchen müssen:

- Das Netzwerk
- Den Server
- Die Oracle Datenbank
- Die Anwendung

Das Ziel einer Leistungssteigerung sollte darin bestehen, die größtmögliche, unmittelbare Verbesserung mit den geringsten Störungen für die vorhandene Software, Hardware und Prozeduren zu erreichen. Die folgenden Abschnitte behandeln jedes der vier Elemente etwas ausführlicher.

### Netzwerk

Das in einer Client-Server-Architektur benutzte Netzwerk kann ein LAN, WAN oder eine Kombination aus beiden sein. Suchen Sie die Hilfe eines Netzwerk-Spezialisten, um den Auslastungsgrad des Netzwerks zu bestimmen. Charakterisieren Sie die Netzwerkanforderungen eines einzelnen Clients. Falls sich das Netzwerk als der Leistungsflaschenhals herausstellen sollte, untersuchen Sie die Client-Anwendungssoftware, um zu erforschen, ob man die Zahl der SQL-Anforderungen an den Server reduzieren kann. Eine Änderung der Client-Software hat weitreichende Auswirkungen; diese Strategie sollte Ihr letzter Ausweg sein!

Wenn Sie gerade mit dem Design einer Anwendung beginnen, können Sie den Datenverkehr auf dem Netzwerk reduzieren, indem Sie Funktionalitäten ermitteln, die mittels in der Datenbank gespeicherter Prozeduren und Funktionen realisiert werden können.

### Ermitteln von Anwendungsempässen

Oracle bietet zwei Optimizer - den regelbasierten und den kostenbasierten Optimizer. Beide sind im Hinblick auf Leistungsmaximierung bei Abfragen ausgelegt. Der regelbasierte Optimizer betrachtet die Struktur der SQL-Anweisung, die es zu optimieren gilt, ermittelt, welche Indizes für eine Tabelle existieren, und erstellt einen Ausführungsplan; es werden keine Informationen über die Inhalte der abzufragenden Tabelle oder ihre Indexwerte verwendet.



*Der regelbasierte Optimizer ist ein Mechanismus innerhalb des Oracle-RDBMS, der die von einer SQL-Anweisung referenzierten Datenbankobjekte betrachtet, um den effizientesten Plan zur Durchführung der in der SQL-Anweisung festgelegten Aufgaben zu ermitteln. Der kostenbasierte Optimizer ist eine Alternative zum regelbasierten; er benutzt die Charakteristiken der Tabellen- und Indexinhalte, um den effizientesten Plan für die Ausführung der Aufgaben zu bestimmen, die von der SQL-Anweisung festgelegt wurden.*

---

Der kostenbasierte Optimizer betrachtet Statistiken, die sich auf die Tabelle, ihre Spalten und ihre Indizes beziehen, um daraus einen Ausführungsplan mit größtmöglicher Verringerung des hauptsächlichen Kostenfaktors zu berechnen - der Zahl der Datenbankblöcke, die gelesen werden müssen, um die Abfrageergebnisse abzurufen. Oracle speichert diese Statistiken in den Data-Dictionary-Tabellen, wann immer eine Analyse der Tabellen oder Indizes ausgeführt wird.

Ein Initialisierungsparameter namens `OPTIMIZER_MODE` steuert die Wahl zwischen den regelbasierten und den kostenbasierten Optimizern; dieser Parameter besitzt drei mögliche Werte:

**CHOOSE** Falls das Data Dictionary keinerlei Tabellen- oder Index-Statistiken enthält, verwendet Oracle den regelbasierten Optimizer. Existieren jedoch Tabellen- und Index-Statistiken, benutzt Oracle den kostenbasierten Optimizer. Der Standardwert für `OPTIMIZER_MODE` ist `CHOOSE`.

**RULE** Oracle verwendet immer den regelbasierten Optimizer.

**COST** Oracle verwendet immer den kostenbasierten Optimizer.



*Die Messung der Performance einer Abfrage ist nicht so einfach, wie es sich anhört. Wenn Sie dieselbe Abfrage zweimal auf eine lange Tabelle anwenden, wird die zweite Abfrage die Ergebnisse wahrscheinlich schneller liefern, weil die SGA bereits die Datenbankblöcke enthält, welche die Abfrage lesen muß. Es werden weniger Laufwerkszugriffe benötigt, so daß die zweite Abfrage weniger Zeit zur Durchführung benötigt. Sie können dies durch die Benutzung der folgenden SQL-Anweisung erreichen:*

```
SQL> alter system flush shared_pool;
```

## Ermitteln des Ausführungsplans für eine SQL-Anweisung mit Hilfe von `explain plan`

Wenn Sie keine Tabellen und Indizes analysieren, wird Oracle den regelbasierten Optimizer verwenden, um den besten Ausführungsplan für jede Abfrage zu ermitteln. Sie können die `EXPLAIN PLAN`-Anweisung benutzen, um den Ausführungsplan für eine Abfrage zu erhalten.



*Die Syntax für `EXPLAIN PLAN` lautet wie folgt:*

```
EXPLAIN PLAN FOR sql-statement
```

Die Variable ist wie folgt definiert:

- *sql-statement* ist die SQL-Anweisung, für die ein Ausführungsplan generiert werden soll.

Bevor Sie die `EXPLAIN PLAN`-Anweisung benutzen, müssen Sie ein SQL\*Plus-Skript ausführen, das die `PLAN_TABLE`-Tabelle in Ihrem Oracle-Account generiert. Dies ist in Listing 19.4 dargestellt.

### **Listing 19.4: Erstellen der `plan_table`-Tabelle**

```
SQL> @c:\orawin95\rdbms73\admin\utlxplan.sql
```

Table created.

```
SQL> desc plan_table
```

```
Name Null? Type
```

```
-----  
STATEMENT_ID VARCHAR2(30)
```



```
TIMESTAMP DATE
REMARKS VARCHAR2(80)
OPERATION VARCHAR2(30)
OPTIONS VARCHAR2(30)
OBJECT_NODE VARCHAR2(128)
OBJECT_OWNER VARCHAR2(30)
OBJECT_NAME VARCHAR2(30)
OBJECT_INSTANCE NUMBER(38)
OBJECT_TYPE VARCHAR2(30)
OPTIMIZER VARCHAR2(255)
SEARCH_COLUMNS NUMBER(38)
ID NUMBER(38)
PARENT_ID NUMBER(38)
POSITION NUMBER(38)
OTHER LONG
```

Nachdem die PLAN\_TABLE-Tabelle erzeugt ist, können Sie mit der Benutzung der EXPLAIN PLAN-Anweisung beginnen. Wann immer diese Anweisung ausgeführt wird, fügt Oracle Zeilen in die PLAN\_TABLE-Tabelle ein; deshalb müssen Sie den Inhalt der Tabelle vor jeder Benutzung von EXPLAIN PLAN löschen. Nehmen Sie beispielsweise an, Sie erstellen eine Tabelle, die den jeweiligen Tag des Jahres und die maximale Temperatur in Grad Fahrenheit für jeden Tag aufzeichnet. Als Entwickler wollen Sie die Effizienz eines Abfragenausführungsplans bestimmen. Listing 19.5 enthält die Schritte, welche Sie befolgen müßten, um den Ausführungsplan einer Anweisung zu ermitteln.

***Listing 19.5: Verwendung der explain plan-Anweisung***

```
SQL> delete from plan_table;
```

0 rows deleted.

```
SQL> explain plan for
2 select day_number, temp_deg_f
3 from day_temp
4 where day_number = 100;
```

Explained.

```
SQL> select operation, options, object_name, id, parent_id, position
2 from plan_table
3 order by id;
```

```
OPERATION OPTIONS OBJECT_NAME ID PARENT_ID POSITION
-----
SELECT STATEMENT 0
TABLE ACCESS FULL DAY_TEMP 1 0 1
```

EXPLAIN\_PLAN sucht nach dem Wort FULL, das in der Spalte OPTIONS im Zusammenhang mit der TABLE ACCESS-Operation angezeigt wird. FULL bedeutet, daß die Abfrage einen vollständigen Tabellen-Scan ausführt, um die Daten abzurufen. Falls eine Abfrage unter Einbeziehung von Day\_Number eine ziemlich häufige Operation ist, sollten Sie sich überlegen, der Day\_Number-Spalte einen Index hinzuzufügen. Listing 19.6 zeigt, wie man den Index erzeugt und EXPLAIN\_PLAN erneut ausführt.

***Listing 19.6: Festlegen des Ausführungsplans nach Erstellung eines Index***

```
SQL> create index day_temp_day_number_ck
2 on day_temp (day_number);
```

Index created.

```
SQL> delete from plan_table;
```

2 rows deleted.

```
SQL> explain plan for
```

```
2 select day_number, temp_deg_f
3 from day_temp
4 where day_number = 100;
```

Explained.

```
SQL> select operation, options, object_name, id, parent_id, position
2 from plan_table
3 order by id;
```

```
OPERATION OPTIONS OBJECT_NAME ID PARENT_ID POSITION
-----
SELECT STATEMENT 0
TABLE ACCESS BY ROWID DAY_TEMP 1 0 1
INDEX RANGE SCAN DAY_TEMP_ 2 1 1
DAY_NUMBER_CK
```

Wie Sie sehen können, haben Sie durch Erstellen des Index den Ausführungsplan des Optimizers für die Abfrage geändert. Anstatt einen vollständigen Tabellen-Scan auszuführen, führt der Optimizer einen Scan des Indexbereiches durch, was eigentlich immer die effizientere Operation ist (allerdings nicht für eine Tabelle mit einer kleinen Anzahl von Reihen). Obwohl die EXPLAIN PLAN-Anweisung nützliche Informationen über die vom Optimizer benutzten Methoden liefert, bietet sie keine exakten Leistungszahlen. Um Leistungsdaten abzufragen, sollten Sie das tkprof-Werkzeug benutzen.

## Analyse einer Anweisung mit Hilfe von tkprof

Falls Sie Personal Oracle benutzen, werden Sie eine ausführbare Datei c:\orawin95\bin\tkprof73.exe (das Verzeichnis und der Dateiname hängen von der Oracle-Version ab, die Sie verwenden) finden. Wenn Sie einen Oracle-Server benutzen, kann eine Version von tkprof im Verzeichnis bin unterhalb des Oracle-Hauptverzeichnisses gefunden werden. Die Befehlszeilenargumente von tkprof sind für jede Version von Oracle 7 oder höher dieselben. tkprof erstellt Oracle-Protokolldateien, die eine Textdatei erzeugen, welche die SQL-Aktivität beschreibt, die während einer bestimmten Oracle-Arbeitssitzung anfällt. Eine Protokolldatei ist extrem nützlich für Leistungsanalysen und Leistungssteigerungen aus diesen Gründen:

- Sie enthält die exakten SQL-Anweisungen, die von einem bestimmten Benutzer während einer Oracle-Arbeitssitzung ausgeführt wurden.
- Wenn der Initialisierungsparameter TIMED\_STATISTICS gesetzt ist, wird die Protokolldatei statistische Informationen über die Ausführung jeder SQL-Anweisung enthalten.

Standardmäßig wird eine Oracle-Datenbank keine Protokolldateien generieren. Um eine solche Datei mit Leistungsdaten zu erstellen, brauchen Sie Ihren DBA, der einen Initialisierungsparameter namens TIMED\_STATISTICS auf TRUE setzt und die Oracle-Instanz neu startet. Sie müssen die Datenbank schließen und neu starten, damit dieser Parameter wirksam wird.

Falls Sie die Leistung einer Gruppe von mit Hilfe von SQL\*Plus ausgeführten SQL-Anweisungen analysieren wollen, können Sie eine Protokolldatei für die Oracle-Arbeitssitzung auf folgende Weise erhalten:

```
SQL> alter session set sql_trace true;
```

Session altered.

Sie können dann jede SQL-Anweisung ausführen, deren Analyse Sie interessiert. Wenn Sie Personal Oracle verwenden, finden Sie die Protokolldateien in c:\orawin95\rdbms73\trace. Falls Sie einen Oracle-Server benutzen, können die Dateien im Verzeichnis mit dem Namen rdbms73\trace unterhalb des Oracle-Hauptverzeichnisses gefunden werden. Sie sollten die Oracle-Arbeitssitzung beenden, bevor Sie versuchen, die Protokolldateien zu analysieren.

Öffnen Sie als nächstes ein MS-DOS-Fenster. Wenn Sie tkprof73 an der Eingabeaufforderung eingeben, zeigt das Programm eine Liste seiner Befehlszeilenargumente. Als Minimum sollten Sie den Namen der Protokolldatei, den Namen der Ausgabedatei, den Oracle-Benutzer und das Kennwort angeben, dessen PLAN\_TABLE-Tabelle benutzt wird, um den Anweisungsausführungsplan zu bestimmen.

Als Standardeinstellung besitzt die Ausgabedatei die Erweiterung .prf. Sie können anschließend einen Editor verwenden, um den Inhalt der Datei anzuzeigen und auszudrucken. In dem folgenden Code-Abschnitt wird der zusammengefaßte Teil einer tkprof-Beispieldatei

gezeigt.

## OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

```
call count cpu elapsed disk query current rows
-----
Parse 7 0.00 1.59 2 0 4 0
Execute 8 0.00 0.49 2 13 38 10
Fetch 5 0,00 0.04 2 3 9 35
-----
total 20 0.00 2.12 6 16 51 45
```

Misses in library cache during parse: 4  
Misses in library cache during execute: 1

8 user SQL statements in session.  
26 internal SQL statements in session.  
34 SQL statements in session.  
4 statements EXPLAINED in this session.  
\*\*\*\*\*

Trace file: c:\orawin95\rdbms73\trace\ora63171.trc  
Trace file compatibility: 7.02.01  
Sort options: execpu  
1 session in tracefile.  
8 user SQL statements in trace file.  
26 internal SQL statements in trace file.  
34 SQL statements in trace file.  
18 unique SQL statements in trace file.  
4 SQL statements EXPLAINED using schema:  
FLUGLE.prof\$plan\_table  
Default table was used.  
Table was created.  
Table was dropped.  
382 lines in trace file.

Wie Sie sehen können, bietet die Protokolldatei eine Fülle an statistischen Informationen über die SQL-Anweisungen, die während der Oracle-Arbeitssitzung ausgeführt wurden. Dieses Werkzeug ist insbesondere bei der Lösung sogenannter 80-20 Probleme eine große Hilfe: Die Identifizierung und Verbesserung von 20 % der SQL-Anweisungen macht 80 % der Arbeit aus.

## Die Leistungsfähigkeit durch Erstellen von Indizes verbessern

Ein wesentlicher Teil der Datenbankaktivität besteht aus SELECT-Anweisungen. Entsprechend führt eine Verbesserung der Abfrageleistung zu einer Steigerung der Gesamtleistung. Eine Abfrage wird im allgemeinen schneller abgearbeitet, wenn ein Index für den Zugriff auf die gewünschten Zeilen benutzt wird. Ein *vollständiger Tabellen-Scan* ist eine Abfrage, in der alle Zeilen der Tabelle gelesen werden, um die gewünschten Zeilen zu finden. Um zu ermitteln, ob eine Abfrage einen vollständigen Tabellen-Scan durchführt, müssen Sie den *Ausführungsplan* der Abfrage einsehen. Wenn ein Ausführungsplan anzeigt, daß ein vollständiger Tabellen-Scan durchgeführt wird, erwägen Sie die Erstellung eines Index, welchen die Abfrage benutzen kann.



*Ein Ausführungsplan ist diejenige Methode, welche das Oracle-RDBMS benutzt, um die Arbeiten für eine SQL-Anweisung auszuführen. Falls eine Tabelle zum Beispiel keine Indizes besitzt, wird der Ausführungsplan der Tabelle es erfordern, daß jede Zeile der Tabelle gelesen wird; dies ist der sogenannte vollständige Tabellen-Scan.*

Seien Sie vor der Möglichkeit einer *Überindizierung* einer Tabelle gewarnt. Erinnern Sie sich, daß Oracle automatisch die Indizes einer

Tabelle bearbeitet, wann immer sich die Inhalte der Tabelle aufgrund einer Anweisung wie INSERT, UPDATE oder DELETE ändern. Ihr Ziel sollte es sein, die häufigsten Abfragen zu optimieren, ohne Oracle zu veranlassen, eine unmäßige Zahl von Indizes zu verwalten.

## Optimierung einer Oracle-Datenbank

Bemühungen um eine Oracle-Leistungssteigerung können in drei Gruppen klassifiziert werden:

- Maßnahmen, die anwendungsunabhängig sind. Sie müssen zum Beispiel keinerlei Software modifizieren oder neu aufbauen, um den globalen Systembereich zu optimieren.
- Maßnahmen, die weitgehend anwendungsunabhängig sind. Sie könnten beispielsweise bei der Inspektion der Abfragen einer Anwendung herausfinden, daß Sie Indizes zur Verbesserung der Abfrageleistung erstellen sollten.
- Maßnahmen, welche die Anwendungssoftware mit einbeziehen. Sie könnten herausfinden, daß das grundlegende Datenbankdesign uneffizient ist oder daß die Software unnötige SQL-Abfragen ausführt. Die Durchführung solcher Änderungen ist sehr teuer in bezug auf das Budget und den Zeitplan. Der Versuch, ein System zu restrukturieren, nachdem es konstruiert wurde, ist sehr schwierig!

Der beste Tip für Sie ist, sich auf Optimierungsmaßnahmen zu konzentrieren, die minimale Auswirkungen auf die bestehende Anwendung haben.

## Den Umfang des SGA vergrößern

Neben ihren anderen Funktionen arbeitet die SGA als Cache für Oracle. Wenn Sie die Zahl der Datenblockpuffer in der SGA vergrößern, erhöhen Sie gleichzeitig die Wahrscheinlichkeit, daß eine SQL-Anweisung den benötigten Block im Speicher findet - wodurch auch die Zahl der Laufwerkszugriffe zur Lokalisierung eines Blocks reduziert werden. Versichern Sie sich jedoch, daß die SGA nicht so groß wird, daß sie den maximalen Speicher des Betriebssystems überschreitet. Die Zahl der von einer Oracle-Datenbank benutzten Datenbankpuffer ist einer der Initialisierungsparameter, die benutzt werden, wenn eine Oracle-Datenbank gestartet wird. Der neue Wert für einen Initialisierungsparameter wird erst wirksam, wenn die Datenbank das nächste Mal gestartet wird.

## Konflikte beim Zugriff auf Plattenspeicher verringern

Um I/O-Operationen von Laufwerken zu optimieren, müssen Sie zuerst herausfinden, ob diese Operationen gleichmäßig auf alle Laufwerke des Servers verteilt sind. Benutzen Sie Befehle und Werkzeuge des Betriebssystems, um die durchschnittliche Zahl der I/O-Anforderungen an jedes Laufwerk zu ermitteln. Ihre Zielsetzungen sollten folgendes einschließen:

- Verringern Sie die Gesamtzahl von Laufwerks-I/O-Anforderungen, indem Sie dem Server mehr Arbeitsspeicher zur Verfügung stellen. Durch Hinzufügen von mehr RAM werden Sie, unter der Voraussetzung, daß keine weiteren Änderungen an der Datenbank durchgeführt wurden, in der Lage sein, die Größe des Oracle-SGA zu vergrößern und damit die Gesamtzahl der I/O-Laufwerkszugriffe zu reduzieren,.
- Verringern Sie die durchschnittliche Zahl der I/O-Anforderungen für jedes Laufwerk, indem Sie dem Server zusätzliche Laufwerke zur Verfügung stellen. Sie können anschließend die Datendateien eines Tablespace auf ein neues Laufwerk verschieben, indem Sie die Anweisung ALTER TABLESPACE mit der Option RENAME benutzen.
- Verringern Sie die durchschnittliche Anzahl von I/O-Anforderungen für jeden Laufwerkscontroller, indem Sie dem Server einen weiteren Controller hinzufügen.
- Verteilen Sie die I/O-Anforderungen gleichmäßig zwischen den Laufwerken, so daß jedes Laufwerk dieselbe Zahl von I/O-Zugriffen verarbeitet. Diese Optimierungsmethode benötigt gleichermaßen Analysen und Experimente, um die optimale Verteilung von Oracle-Datendateien und Log-Dateien zu ermitteln.

## Zusammenfassung

Diese Lektion behandelte die folgenden Prinzipien in bezug auf das Management von Oracle-Benutzern, Datenbankrollen und Berechtigungen und einer Leistungssteigerung:

- Sie können Benutzer, Rollen und Berechtigungen mit dem Point-and-Click-Interface des Personal Oracle Navigators und des Security Managers verwalten. Sie können außerdem SQL-Befehle in SQL-Worksheet oder SQL\*Plus benutzen, um Benutzer, Rollen und Berechtigungen zu erstellen und anzupassen.
- Ein Profil ist ein Satz von Kontingenten auf Systemebene, die einem Benutzer oder einer Rolle zur Verfügung gestellt werden können. Diese Kontingente schließen den Aufwand an Leerlaufzeit und die Zahl der logischen Lesezugriffe mit ein, die für eine einzelne Abfrage ausgeführt werden.
- Um die Netzwerkleistung vor Inbetriebnahme einer Anwendung zu maximieren, planen Sie die Verwendung von in der Datenbank gespeicherten Prozeduren und Funktionen, um die Zahl der SQL-Anforderungen auf dem Netzwerk zu reduzieren und um überflüssige SQL-Anfragen von der Client-Anwendung zu minimieren.
- Um die Performance der Oracle-Datenbank zu verbessern, optimieren Sie die SGA, reduzieren Sie die Fragmentierung von Datensegmenten, und fügen Sie nicht einmalige Indizes dort ein, wo es geeignet erscheint.
- Um die Performance des Servers zu verbessern, fügen Sie Arbeitsspeicher hinzu, so daß der SGA mehr Datenblockpuffer zur Verfügung stehen; fügen Sie außerdem mehr Laufwerke hinzu, um die durchschnittliche Anzahl von I/O-Anforderungen pro Laufwerk zu reduzieren.
- Oracle benutzt zwei Optimizer: den regelbasierten und den kostenbasierten Optimizer.
- Der regelbasierte Optimizer betrachtet die Syntax einer Abfrage und das Vorhandensein von Indizes, um die effizienteste Methode zur Abfrage der gewünschten Zeilen zu ermitteln.
- Der kostenbasierte Optimizer betrachtet die Syntax einer Abfrage, das Vorhandensein von Indizes und die Statistiken von Tabellen und Indizes, um die effizienteste Methode zum Abruf der gewünschten Zeilen zu ermitteln.
- Benutzen Sie die EXPLAIN\_PLAN-Anweisung, um den Ausführungsplan einer Abfrage einzusehen.
- Sie können für eine bestimmte Oracle-Arbeitssitzung eine Protokolldatei generieren, indem Sie ALTER SESSION SET SQL\_TRACE TRUE zu Beginn der Arbeitssitzung ausführen.
- tkprof ist ein Werkzeug, das eine binäre Oracle-Protokolldatei in eine Textdatei konvertiert, die jede SQL-Anweisung enthält, welche während einer Oracle-Arbeitssitzung ausgeführt wurde.

## Wie geht es weiter?

An Tag 20, »Einsatz von Oracle Database Designer und Oracle Designer/2000 in der Anwendungsentwicklung«, werden Sie etwas über die Anwendung von zwei Datenbank-Designwerkzeugen lernen. Oracle Database Designer wird benutzt, um ein logisches Datenmodell zu entwickeln und die physische Datenbank automatisch zu implementieren, indem SQL-DDL-Anweisungen generiert werden, die Tabellen, Indizes und andere Objekte erzeugen. Oracle Designer/2000 ist ein umfassendes Mehrbenutzer-Design-Werkzeug, das es Ihnen erlaubt, Datenbanken und Anwendungen für ein großes, komplexes Unternehmen zu formen und zu entwickeln.

## Fragen und Antworten

Frage:

Kann eine Datenbankrolle einer anderen Datenbankrolle gewährt werden?

Antwort:

*Ja. Sie können eine Rolle einer anderen Rolle oder einem Oracle-Benutzer zuteilen. Sie sollten es jedoch vermeiden, einen unnötig komplexen Satz von Rollen zu erstellen; die Benutzung von Rollen sollte das Verwalten von Datenbankberechtigungen vereinfachen und es nicht komplizieren.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Richtig oder falsch? Eine Datenbankrolle kann Benutzern und anderen Datenbankrollen zugeteilt werden.
2. Richtig oder falsch? Das Hinzufügen eines Index verbessert immer die Leistung einer Anwendung.
3. Richtig oder falsch? Jeder Oracle-Benutzer kann ein öffentliches Synonym erstellen, solange dieses auf eine Tabelle verweist, die sich im Besitz des Benutzers befindet.

## Übungen

Sie können diese Fragen allgemein beantworten, ohne die Verwendung von SQL-Anweisungen: Wenn Sie der DBA am Flugle College wären, welche Datenbankrolle würden Sie erzeugen? Welche Berechtigungen würden Sie dieser Rolle gewähren?

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖Kapitel❖ Top Inhalt Index **SAMS** Kapitel❖

# 20

# Database Designer und Designer/2000 in der Anwendungsentwicklung

[Erstellen eines Entity-Relationship-Diagramms mit dem Database Designer](#)

[Eine Tabelle zu dem Diagramm hinzufügen](#)

[Festlegen der Spalten für eine Tabelle](#)

[Festlegen der Integritätsregeln für eine Tabelle](#)

[Festlegen eines Fremdschlüssels](#)

[Die DDL-Anweisungen für das Diagramm erstellen](#)

[Eine Oracle-Datenbank mit Hilfe des Database Designers nach der Umkehrtechnik entwickeln](#)

[Die Grundlagen von Designer/2000](#)

[Zusammenfassung](#)

[Fragen und Antworten](#)

[Workshop](#)

Obwohl diese Lektion ziemlich am Schluß dieses Buches erscheint, könnten Sie sie als den Beginn der Entwicklung einer Oracle-Datenbankanwendung betrachten. In dieser Lektion untersuchen Sie die Benutzung von Database Designer und Designer/2000.

Database Designer wurde 1997 von Oracle angekündigt. Es ist ein Werkzeug zur Erstellung von ER-Modellen (ER, Entity-Relationship), entworfen, um mit ERwin von Logic Works und S-Designor von Sybase zu konkurrieren. Alle drei Werkzeuge bieten ein grafisches Interface für die Erstellung eines Entity-Relationship-Diagramms, das die DDL-Anweisungen zur Erzeugung einer funktionierenden Datenbank generiert. Database Designer und seine Mitbewerber sind für Projekte geeignet, bei denen eine Person das Datenmodell entwirft und wartet. Außerdem konzentrieren sich diese Werkzeuge nur auf die Entwicklung eines logischen Datenmodells und seine Realisierung als physische Datenbank; sie helfen dem Anwendungsentwickler nicht bei der Prozeß- und Datenflußmodellierung sowie anderen Facetten eines umfangreichen Anwendungsdesigns.

Im Gegensatz dazu ist Designer/2000 eine umfassende Suite von Werkzeugen, welche die Analyse, das Design und die Realisierungsphase eines großen Projekts unterstützt. Sie können sich Designer/2000 als eine »von der Wiege bis zur Bahre«-Entwicklungsmethode vorstellen.

Diese Werkzeuge können einem ganzen Team helfen, die Anforderungen für eine Anwendung auf vielen Ebenen zusammenzufassen, zu dokumentieren und zu verifizieren - Datenmodell, Prozeßmodell und Datenfluß, um nur einige zu nennen. Diese Werkzeuge arbeiten so zusammen, daß die Ausgabedaten des einen die Eingabewerte eines anderen Werkzeugs sind. Das Herz des Designer/2000 ist ein Oracle-basiertes Repository, das eine Beschreibung der Elemente der Anwendung enthält, die erstellt werden soll.

Sie können Database Designer und Database/2000 direkt bei der Oracle Corporation oder einem Oracle-Händler bestellen. Sie können auch eine Testversion des Database Designers von der Oracle-Website mit der Adresse [www.oracle.com](http://www.oracle.com) herunterladen. Um von dieser Lektion profitieren zu können, sollten Sie eine Kopie des Database Designers besitzen, damit Sie die einzelnen Schritte jeder Übung nachvollziehen können.



*Ein Repository ist ein Satz von Datenbankobjekten, welche Metadaten enthalten - das bedeutet Daten über Daten. Die Designer/2000-Repository zum Beispiel enthält Informationen über das Datenmodell, das Prozeßmodell und viele andere Design-Aspekte einer Anwendung.*



*Der Database Designer ist eine 16-Bit-Anwendung. Als Konsequenz daraus müssen Sie die Windows-Version von SQL\*Net, ebenfalls eine 16-Bit-Version, installieren, wenn Sie das Produkt unter Windows 95 oder Windows NT benutzen wollen.*

Eine natürliche Frage ist die: Welches Werkzeug ist besser - eines wie der Database Designer (einschließlich ERwin oder S-Designor) oder Designer/2000? Dies hängt davon ab, »wohin Sie gehen wollen«. Um eine Analogie zu benutzen, wenn Sie ein langes Wochenende in Ihrer Berghütte planen, würde es ausreichen, nur leichtes Gepäck zum Übernachten mitzunehmen. Wenn Sie jedoch eine sechsmonatige Expedition in eine ferne Region in Zentralasien planen, kann leichtes Gepäck ein schwerwiegender Fehlgriff sein; umsichtiger wäre es, ein Dutzend große Zelte und Zubehör einzupacken, um unvorhersehbaren Bedingungen begegnen zu können.

Ähnlich dazu sind der Database Designer, ERwin oder S-Designor ein gute Wahl für ein Projekt, das

- für ein kleines Unternehmen gedacht ist;
- nicht zu ambitioniert im Hinblick auf die Möglichkeiten ist, welche die Anwendung bieten soll;
- von einer Handvoll von Entwicklern betreut wird, die Kontakt zu den Endanwendern haben.

Database Designer, ERwin oder S-Designor würden jedoch für ein Projekt unpassend sein, das

- für ein großes Unternehmen mit vielen Abteilungen und Bereichen eingeführt werden soll;
- von einflußreichen Entscheidungsträgern in verschiedenen Abteilungen einer Organisation als Bedrohung angesehen würde;
- im Hinblick auf die Möglichkeiten, welche die Anwendung bieten soll, und der zu entwickelnden Technologien sehr ambitioniert ist,
- von vielen Einzelpersonen betreut wird - Analytikern, Designern, Entwicklern, Testern, technischen Dokumentaren und Schulenden.

Wenn Sie noch nicht mit Designer/2000 gearbeitet haben, erwarten Sie nicht, die Software am Montag zu erhalten und mit der Erstellung vollständiger Anwendungen am Dienstag beginnen zu können. Dies ist eine komplexe Suite von Werkzeugen, deren Nutzung einen bedeutenden Aufwand an Training erfordert. Erwarten Sie ebenfalls nicht von einer Einzelperson, daß sie bis ins Detail mit jeder Möglichkeit in jeder Komponente vertraut werden kann; das ist schlicht unrealistisch. Statt dessen macht es für jedes Mitglied des Entwicklungsteams viel mehr Sinn, sich auf die Komponenten zu konzentrieren, die mit seiner oder ihrer Rolle innerhalb des Entwicklungsprozesses korrespondieren. In dieser Lektion erhalten Sie nur einen flüchtigen Blick auf die Komponenten des Designer/2000; eine umfassende Beschäftigung mit Designer/2000 würde ein eigenes Buch erfordern.

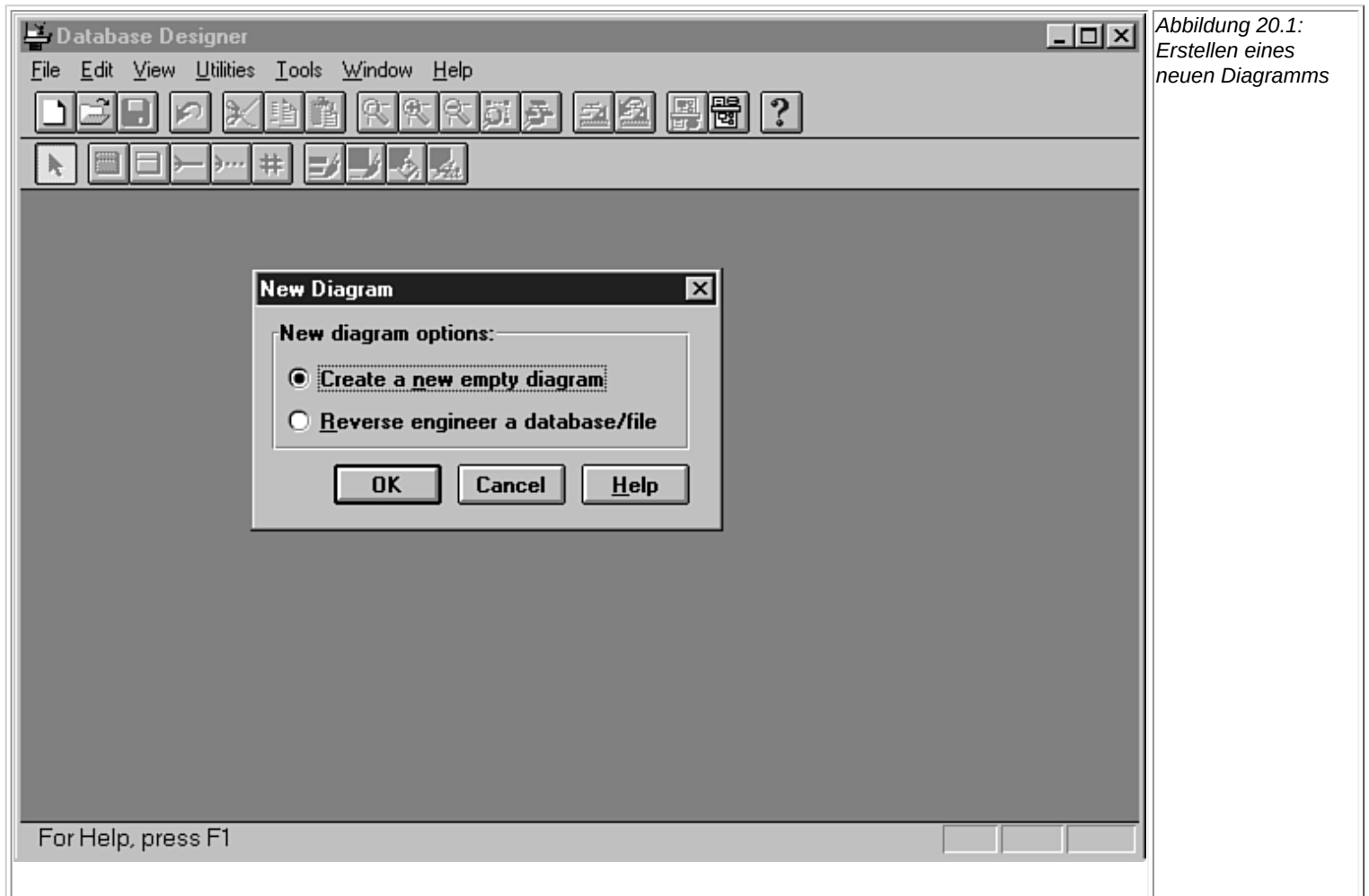
## Erstellen eines Entity-Relationship-Diagramms mit dem Database Designer

Es gibt viele Vorteile bei der Anwendung eines Werkzeugs wie des Database Designers anstelle der Wartung eines Skripts mit SQL-



Anweisungen. Es ist viel unwahrscheinlicher, mit dem Database Designer typografische Fehler zu machen; er prüft beispielsweise, ob die von einem Fremdschlüssel referenzierte Spalte selbst ein Primärschlüssel ist. Sie können das Entity-Relationship-Diagramm ausdrucken und in einer Konferenz präsentieren; es ist viel schwieriger, das von einem Skript repräsentierte Datenmodell zu visualisieren.

Lassen Sie uns die Schritte betrachten, die notwendig sind, um ein neues ER-Diagramm anzufertigen. Um den Database Designer in Windows 95 aufzurufen, betätigen Sie Start | Programs | Database Designer | Database Designer. Wenn das Programm erstmals gestartet wird, werden Sie ein Menü und eine Symbolleiste im oberen Teil des Bildschirms sehen. Um ein neues Diagramm zu erstellen, wählen Sie den Menüeintrag File | New. Es wird ein Fenster erscheinen, in dem Sie zwei Wahlmöglichkeiten haben: Sie können entweder ein neues Diagramm erstellen oder eine bereits bestehende Datenbank rückübersetzen. Die Standardeinstellung ist die Erzeugung eines neuen Diagramms (siehe Abbildung 20.1). Klicken Sie OK an. Sie sollten ein Fenster mit dem Titel ODD1 sehen - der Standardname für das erste zu erstellende Diagramm.

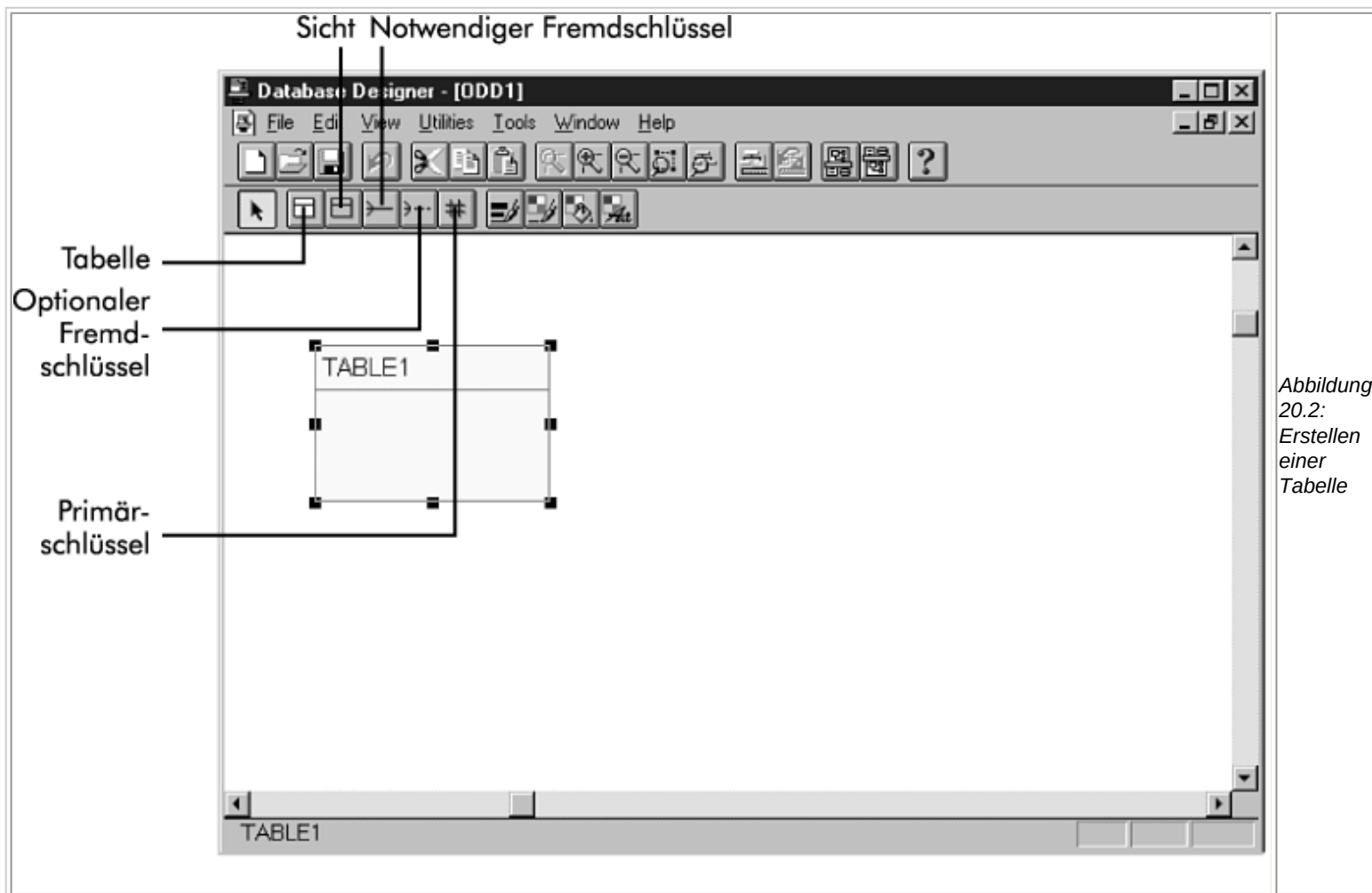


## Eine Tabelle zu dem Diagramm hinzufügen

Lassen Sie uns zu Beginn dem Diagramm die Student-Tabelle hinzufügen. In der unteren Zeile der Symbolleiste befinden sich fünf Befehlsschaltflächen, die mit diesen fünf Elementen in einem ER-Diagramm korrespondieren:

- Table (Tabelle)
- View (Datenansicht)
- Mandatory foreign key (obligatorischer Fremdschlüssel)
- Optional foreign key (optionaler Fremdschlüssel)
- Primary key (Primärschlüssel)

Um zum Beispiel eine Tabelle zu erstellen, würden Sie die Schaltfläche Table auf der Symbolleiste anklicken, den Mauszeiger auf dem Diagramm dort positionieren, wo Sie die Tabelle plazieren wollen, und mit der linken Maustaste klicken (siehe Abbildung 20.2).



Um Details der neuen Tabelle festzulegen, bewegen Sie den Mauszeiger über die Tabelle und klicken doppelt. Ein Fenster mit dem Titel Edit Table wird erscheinen und sechs Registerkarten enthalten:

- Table
- Column Definition
- Column Display (wird optional mit Designer/2000-Anwendungsgeneratoren benutzt, um die Anzeigecharakteristiken der Tabellenspalten zu bestimmen)
- Constraints (Integritätsregeln)
- Validation (Gültigkeitsprüfung)
- Index

Wie in Abbildung 20.3 gezeigt wird, füllen Sie die Felder im Tabellenregister folgendermaßen aus:

|                           |                                                                                                                                                                                                                                                                                   |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name                      | STUDENT                                                                                                                                                                                                                                                                           |
| Display Title             | Student                                                                                                                                                                                                                                                                           |
| Number of Rows, Start/End | Eine Schätzung der Zahl der Zeilen, die zu Beginn bzw. letztlich in der Tabelle gespeichert werden sollen (wird zur Abschätzung der Startlänge für die Tabelle benutzt).                                                                                                          |
| Comment                   | Enthält Basisinformationen über die Studenten.                                                                                                                                                                                                                                    |
| Journal                   | Markiert. Wenn es mit Designer/2000 benutzt wird, zeigt es an, daß eine Journaltabelle für diese Tabelle erstellt werden sollte (eine Journaltabelle wird zur Speicherung aller Änderungen einer Tabelle benutzt, einschließlich der Datumsänderung und der ausführenden Person). |

**Edit Table STUDENT**

**Table** | Column Defn | Col Display | Constraints | Validation | Index

**Name**  
STUDENT

**Alias**

**Display Title**  
Student

**Number of Rows**  
Start: 500 End: 600

☒ Create ? ☒ Journal ?

**Storage Definition**

**Tablespace**

**Init Trans**

**Max Trans**

**Pct Free**

**Pct Used**

**Comment**  
Contains basic information about students.

OK Cancel Help Text...

Abbildung 20.3:  
Festlegen der  
Tabellenmerkmale

## Festlegen der Spalten für eine Tabelle

Um die Spalten in der Student-Tabelle festzulegen, wählen Sie die Registerkarte mit der Bezeichnung Column Defn. Sie werden fünf Spaltenüberschriften sehen: Column Name, Seq, Domain, Datatype und Avg Len. Es gibt allerdings rechts von Avg Len noch weitere Felder, die mit Hilfe der horizontalen Bildlaufleiste angezeigt werden können. In diesem Formular repräsentiert jede Zeile Informationen über eine einzelne Spalte (siehe Abbildung 20.4). Um der Student-Tabelle eine Zeile hinzuzufügen, legen Sie einen Wert für diese Felder fest:

|             |                                                                                                                                        |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------|
| Column Name | Name der Spalte (zum Beispiel Student_ID)                                                                                              |
| Seq         | Die Reihenfolge, in der die Spalten in der von Database Designer generierten CREATE TABLE-Anweisung festgelegt werden                  |
| Domain      | Kann leer gelassen werden (wird mit Database Designer benutzt, um jede Domäne und die damit zusammenhängenden Attribute zu definieren) |
| Datatype    | Der Datentyp der Spalte kann aus der Popup-Liste gewählt werden                                                                        |
| Avg Len     | Mittlere Länge der Spalte (dies wird zur Abschätzung der Startlänge der Tabelle benutzt)                                               |

Wenn Sie den Bildlauf nach rechts verschieben, werden Sie einige weitere Felder sehen (siehe Abbildung 20.5):

|         |                                                                                                                |
|---------|----------------------------------------------------------------------------------------------------------------|
| Max Len | Die maximale Länge für die Spalte (die Sie festlegen können oder auch nicht, abhängig vom Datentyp der Spalte) |
|---------|----------------------------------------------------------------------------------------------------------------|

Opt

Wenn markiert, ist die Spalte optional (die Standardeinstellung); falls nicht markiert, ist die Spalte obligatorisch (NOT NULL)

| Column Name    | Seq | Domain | Datatype | Avg Len |
|----------------|-----|--------|----------|---------|
| STUDENT_ID     | 1   |        | Varchar2 | 20      |
| LAST_NAME      | 2   |        | Varchar2 | 10      |
| FIRST_NAME     | 3   |        | Varchar2 | 7       |
| MI             | 4   |        | Varchar2 | 1       |
| YEAR           | 5   |        | Varchar2 | 10      |
| STREET_ADDRESS | 6   |        | Varchar2 | 12      |
| CITY           | 7   |        | Varchar2 | 7       |
| STATE          | 8   |        | Varchar2 | 2       |
| ZIPCODE        | 9   |        | Varchar2 | 5       |
| TELEPHONE      | 10  |        | Varchar2 | 10      |

Abbildung 20.4:  
Definition einer  
Spalte

| Column Name    | Max Len | Dec Pl | Opt                                 | Init Vol | Final Vol | Uppercase                           | Create?                             |
|----------------|---------|--------|-------------------------------------|----------|-----------|-------------------------------------|-------------------------------------|
| STUDENT_ID     | 20      |        | <input type="checkbox"/>            |          |           | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| LAST_NAME      | 25      |        | <input type="checkbox"/>            |          |           | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| FIRST_NAME     | 25      |        | <input type="checkbox"/>            |          |           | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| MI             | 1       |        | <input checked="" type="checkbox"/> |          |           | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| YEAR           | 25      |        | <input checked="" type="checkbox"/> |          |           | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| STREET_ADDRESS | 25      |        | <input checked="" type="checkbox"/> |          |           | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| CITY           | 25      |        | <input checked="" type="checkbox"/> |          |           | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| STATE          | 2       |        | <input checked="" type="checkbox"/> |          |           | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| ZIPCODE        | 9       |        | <input checked="" type="checkbox"/> |          |           | <input type="checkbox"/>            | <input checked="" type="checkbox"/> |
| TELEPHONE      | 10      |        | <input checked="" type="checkbox"/> |          |           | <input type="checkbox"/>            | <input checked="" type="checkbox"/> |

Abbildung 20.5:  
Zusätzliche Felder für  
die Definition einer  
Spalte

## Festlegen der Integritätsregeln für eine Tabelle

Um die Integritätsregeln für eine Tabelle festzulegen, wählen Sie die Registerkarte Constraints. Innerhalb dieser werden Sie vier weitere Registerkarten sehen, die mit den vier Typen von Integritätsregeln korrespondieren: Primary, Foreign, Unique und Check. Um den Primärschlüssel für die Student-Tabelle festzulegen, geben Sie den Regelnamen ein und wählen die Student\_ID-Spalte aus dem Popup-Listefeld mit der Bezeichnung Column Name aus (siehe Abbildung 20.6).

**Edit Table STUDENT**

Table Column Defn Col Display **Constraints** Validation Index

**Primary** Foreign Unique Check

**PK\_STUDENT**

Constraint Name: **PK\_STUDENT**

Error Message:

Exceptions Table:

Validate in: Server ☒ Create ☒ Enable

Storage Definition:

Tablespace:

Init Trans:

Max Trans:

| Seq | Column Name | Datatype | Len | Opt                      | Comment |
|-----|-------------|----------|-----|--------------------------|---------|
| 1   | STUDENT_ID  | Varchar2 | 20  | <input type="checkbox"/> |         |
|     |             |          |     | <input type="checkbox"/> |         |
|     |             |          |     | <input type="checkbox"/> |         |

Insert Constraint Delete Constraint Insert Key Column Delete Key Column

OK Cancel Help Text...

Abbildung 20.6:  
Festlegen des  
Primärschlüssels

Die Student-Tabelle besitzt eine andere Integritätsregel: eine vom Typ CHECK auf das Studienjahr, um festzustellen, ob es sich um einen FRESHMAN (Erstsemester), SOPHOMORE (Student im zweiten Semester), JUNIOR (Student im vorletzten Semester) oder SENIOR (Student im letzten Semester) handelt. Um diese Integritätsregel zu erstellen, wählen Sie die Registerkarte Validation. Wählen Sie die Spalte, deren Werte Sie einer Gültigkeitsprüfung unterziehen wollen - in diesem Falle YEAR. Geben Sie jeden gültigen Wert für die Spalte ein, wie in Abbildung 20.7 gezeigt wird.

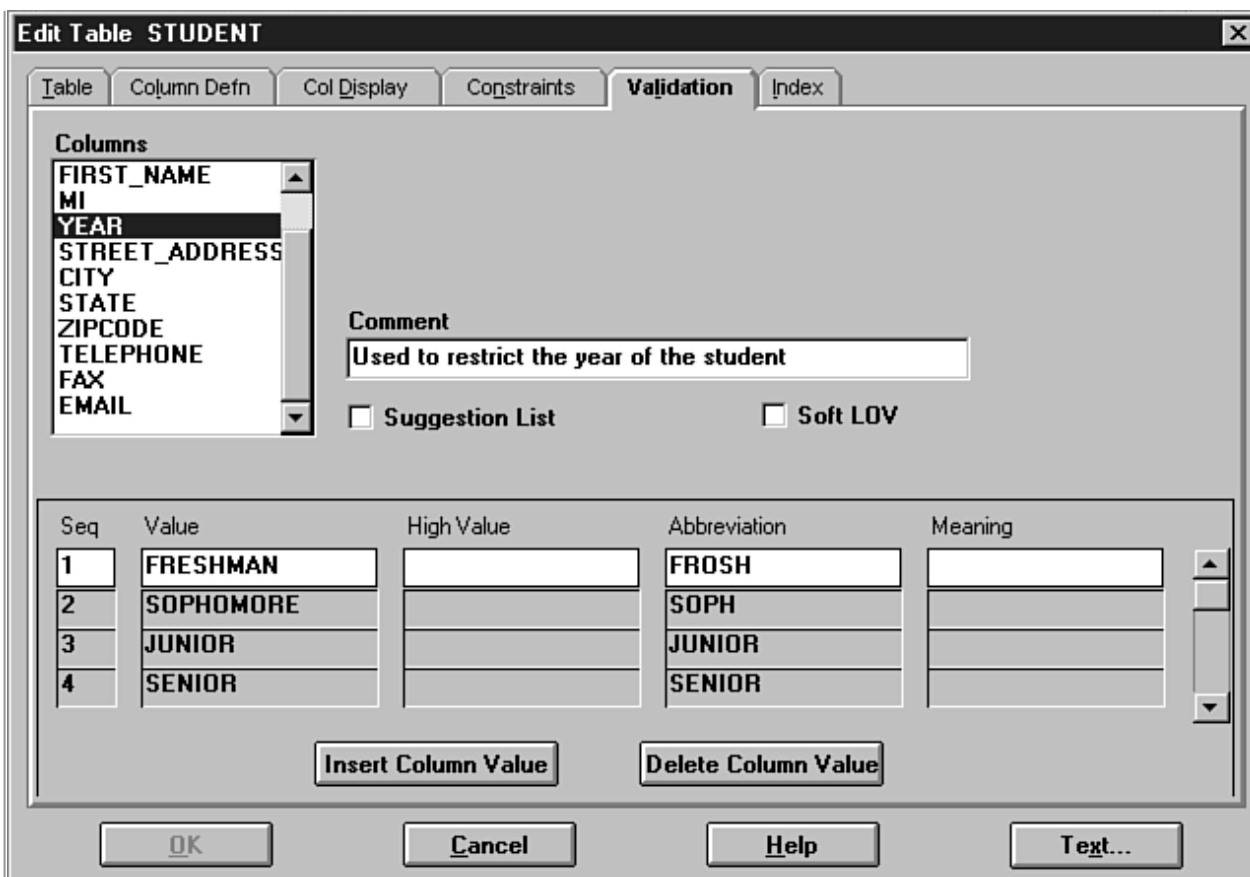


Abbildung 20.7:  
Festlegen der  
Gültigkeitsüberprüfung  
für eine Spalte

Wenn Sie das Festlegen der Charakteristika der Tabelle beendet haben, klicken Sie auf OK. Die Tabelle wird mit den Eigenschaften im Diagramm angezeigt, die Sie bestimmt haben. Wie Sie in Abbildung 20.8 sehen können, wird die Raute # links von der Student\_ID angezeigt, was die Spalte als Primärschlüssel kennzeichnet. Ebenso wird links von allen obligatorischen Spalten ein Stern \* angezeigt.

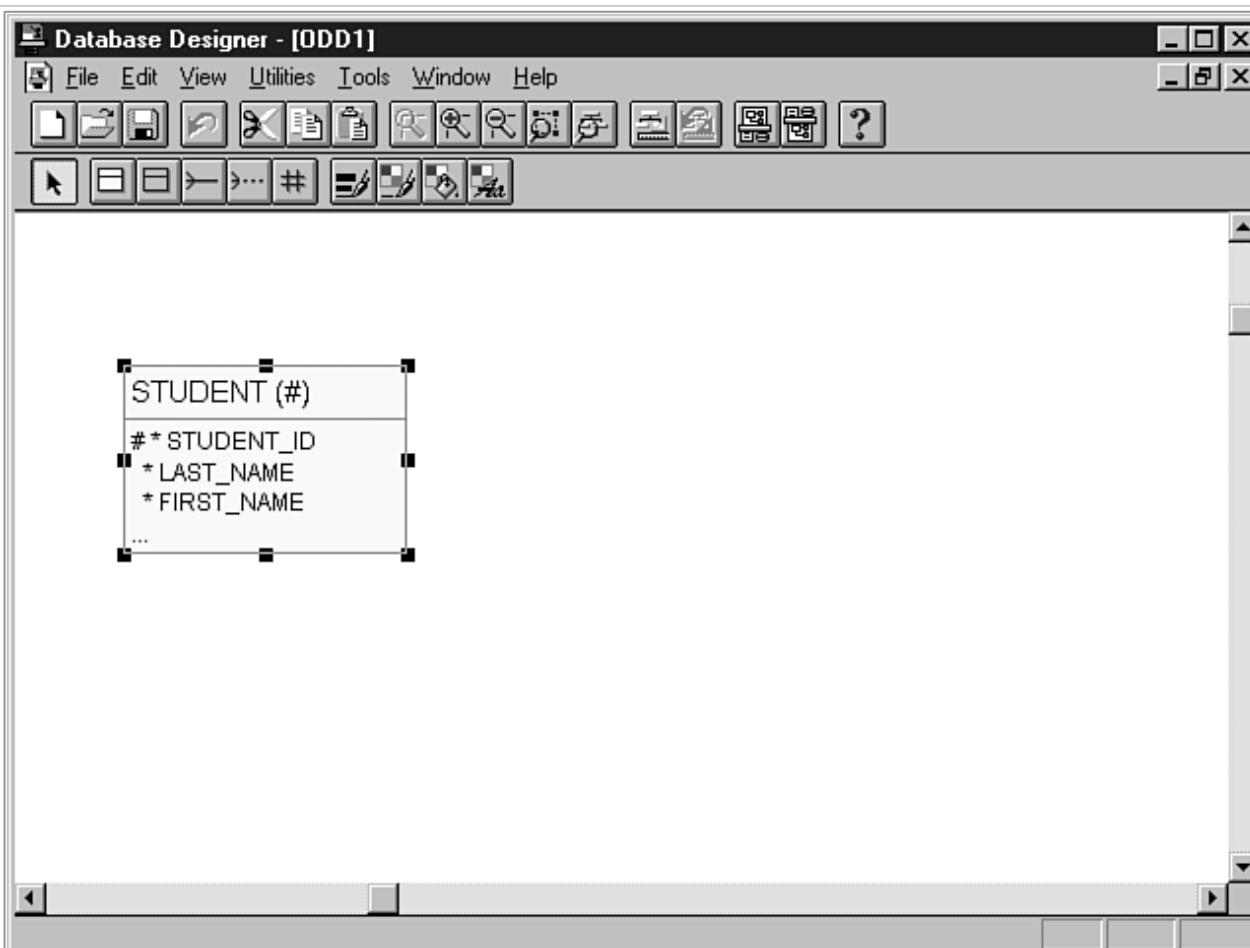


Abbildung 20.8:  
Die erstellte Tabelle  
wird in einem Diagramm  
angezeigt

## Festlegen eines Fremdschlüssels

Um den Schritten zu folgen, die zur Erzeugung eines Fremdschlüssels benutzt werden, lassen Sie uns ein bereits existierendes Diagramm betrachten, das zwei Tabellen besitzt: Department und Instructor. Wie Sie in Abbildung 20.9 sehen können, enthält die Tabelle Instructor nicht die Department\_ID, welche den Fachbereich anzeigt, zu dem der Dozent gehört.

Um einen obligatorischen Fremdschlüssel in der Instructor-Tabelle zu erzeugen, wählen Sie die Befehlsschaltfläche Mandatory Foreign Key auf der Symbolleiste (wird im allgemeinen als *Krähenfuß* bezeichnet). Beachten Sie, daß der Mauszeiger sich jetzt in einen Krähenfuß verwandelt hat. Bewegen Sie ihn über die Instructor-Tabelle. Klicken Sie mit der linken Maustaste auf die Tabelle; ziehen Sie den Krähenfuß über die Department-Tabelle, und klicken Sie erneut mit der linken Maustaste. Sie sollten jetzt zwei Dinge bemerken (wie in Abbildung 20.10 gezeigt wird):

- Ein obligatorischer Fremdschlüssel mit der Bezeichnung Department\_FK verbindet die Tabellen Department und Instructor.
- Die Department\_ID wurde der Instructor-Tabelle hinzugefügt.

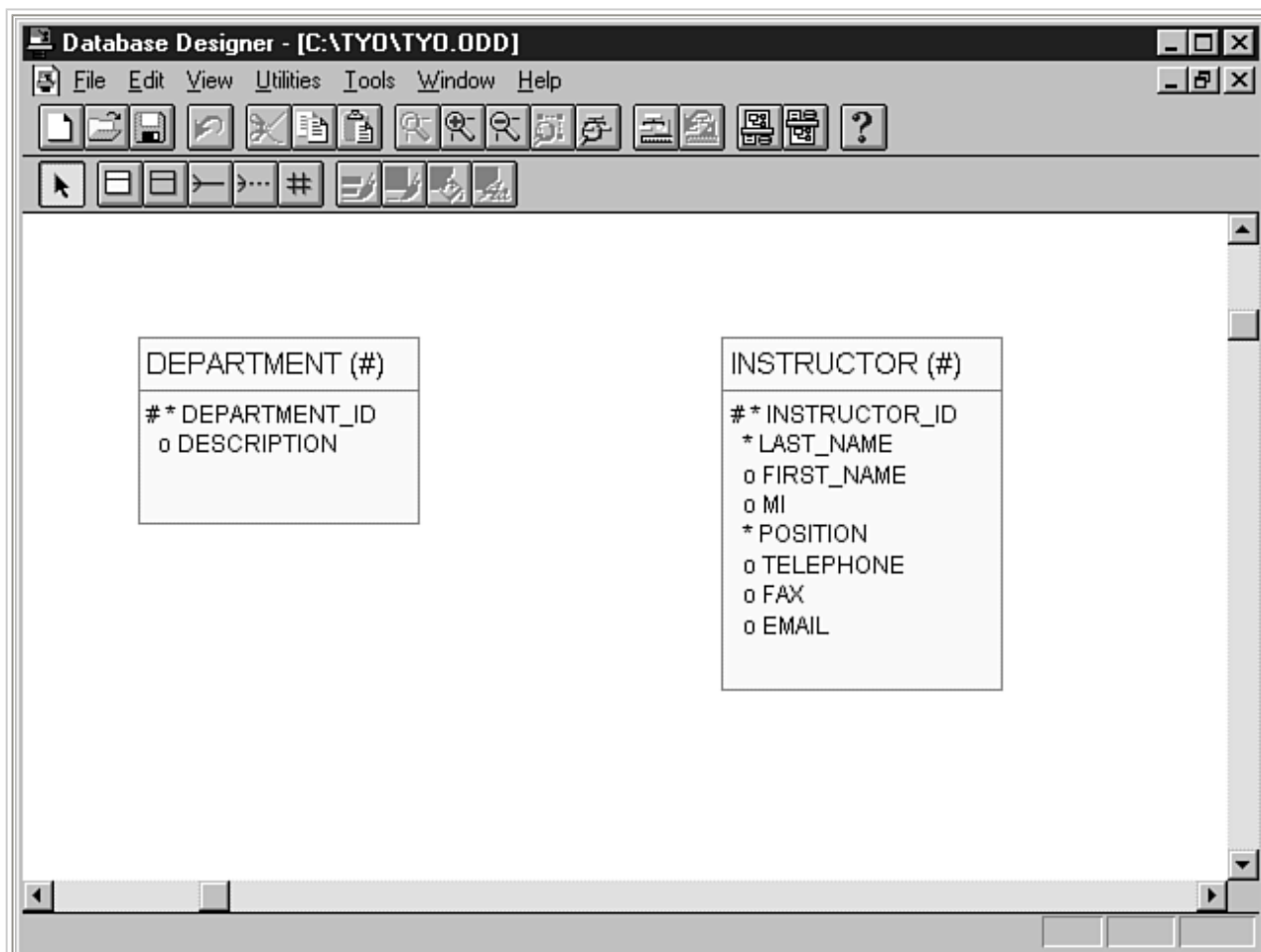
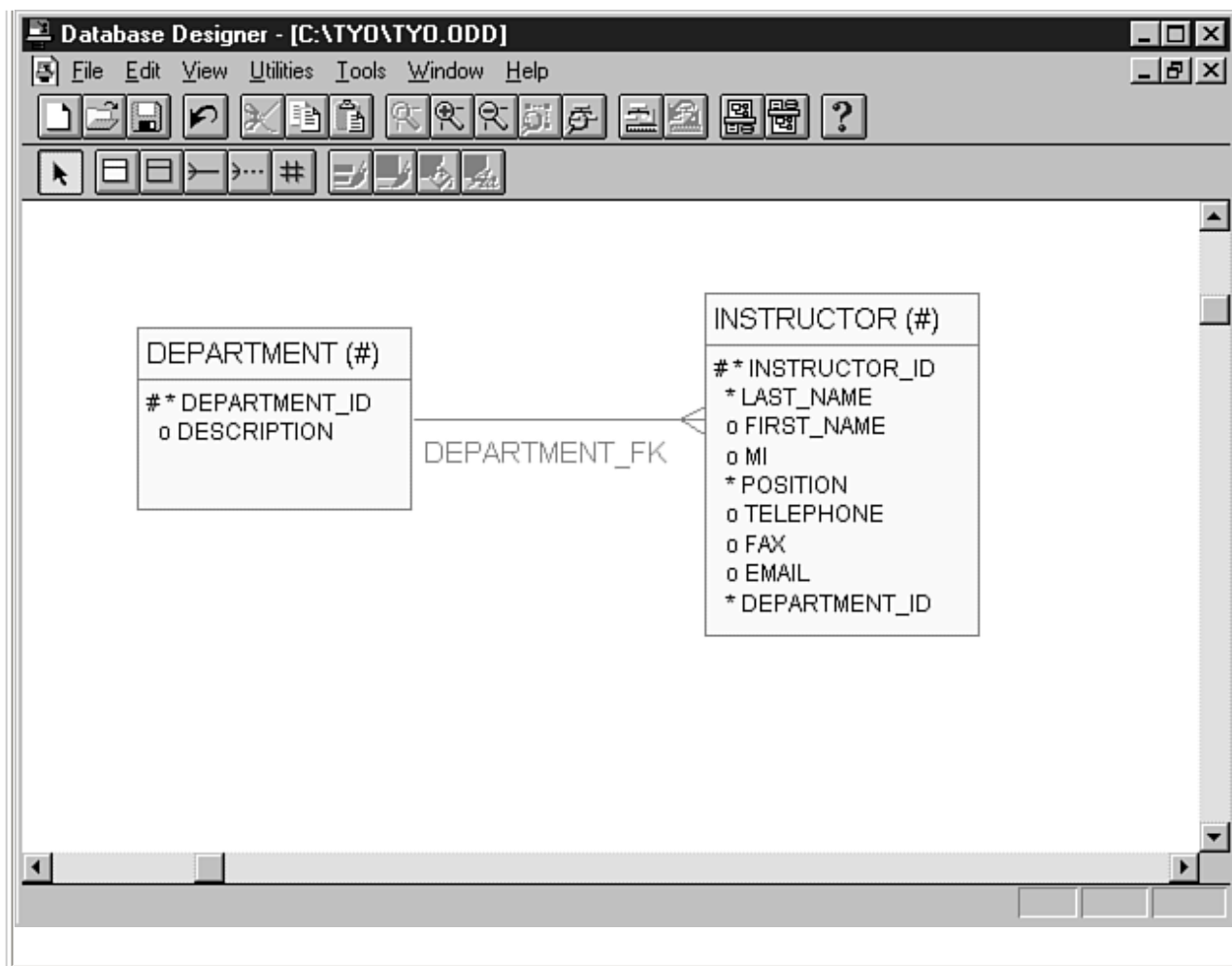


Abbildung 20.9:  
Ein vorhandenes  
Diagramm mit zwei  
Tabellen ohne  
Beziehung

Abbildung 20.10:  
Der Fremdschlüssel  
wird erzeugt



Um den Fremdschlüssel zu bearbeiten, doppelklicken Sie auf die Zeile, welche den Fremdschlüssel definiert. Ein Fenster mit dem Titel Edit Foreign Key wird erscheinen (siehe Abbildung 20.11). Für diesen Fremdschlüssel müssen die Standardwerte nicht geändert werden.

The 'Edit Foreign Key DEPARTMENT\_FK' dialog box is shown. It has two tabs: 'Foreign Key' (selected) and 'Key Columns'. The 'Foreign Key' tab contains the following fields and options:

- Constraint Name:** DEPARTMENT\_FK
- Join Table:** DEPARTMENT
- Error Message:** (empty text box)
- Exceptions Table:** (empty text box)
- Validate in:** Server (dropdown menu)
- Create ?** ☒
- Enable ?** ☒
- Mandatory ?** ☒
- Transferable ?** ☒
- Delete Rule:** Restricted (dropdown menu)
- Update Rule:** Restricted (dropdown menu)

Buttons at the bottom: OK, Cancel, Help, Text...

Abbildung 20.11:  
Bearbeitung des  
Fremdschlüssels



## Die DDL-Anweisungen für das Diagramm erstellen

Database Designer bietet einen Generation Wizard, der Sie durch den Prozeß der Generierung eines Datenbankschemas für Ihr ER-Diagramm führt. Um den Generation Wizard aufzurufen, können Sie den Menüeintrag Tools | Generation Wizard wählen. Das erste vom Generation Wizard gezeigte Fenster informiert Sie einfach darüber, daß Sie Datenbankobjekte in eine Datei oder eine Datenbank generieren können (siehe Abbildung 20.12). Klicken Sie auf Next, um zum nächsten Fenster zu wechseln.

In diesem Fenster müssen Sie festlegen, wohin Sie die Datenbankobjekte erzeugen wollen - in eine Datei, welche die DDL-Anweisungen enthält, die wiederum Ihre Datenbank erstellen werden, oder direkt in eine Datenbank. Als Standard werden die DDL-Anweisungen in eine SQL-Datei geleitet (siehe Abbildung 20.13). Klicken Sie auf Next, um zum nächsten Fenster zu wechseln.

Im diesem Fenster können Sie die Erzeugung von Datenbankobjekten auf bestimmte Objekttypen beschränken. Standardmäßig werden jedoch alle Objekttypen generiert (siehe Abbildung 20.14).

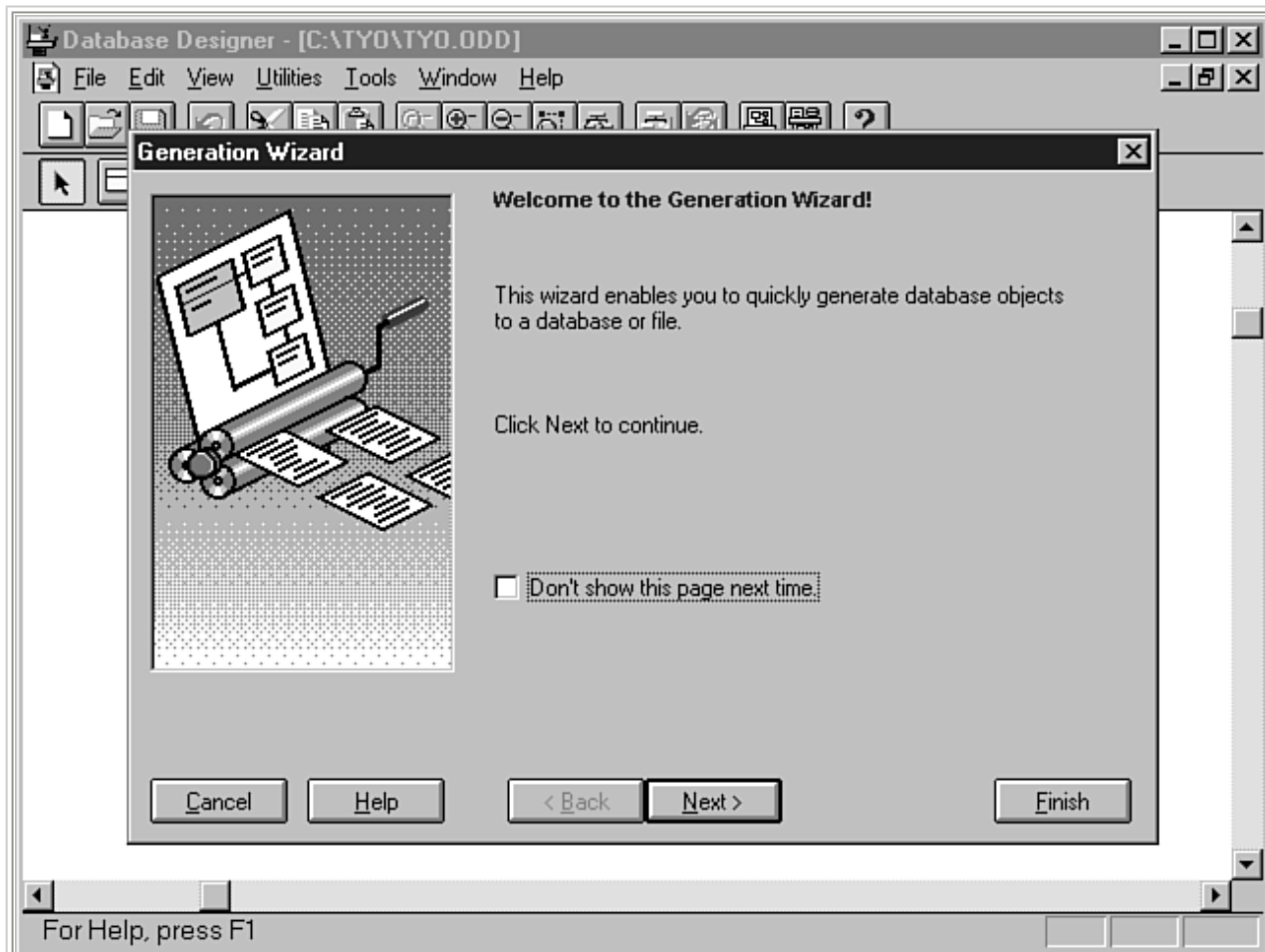
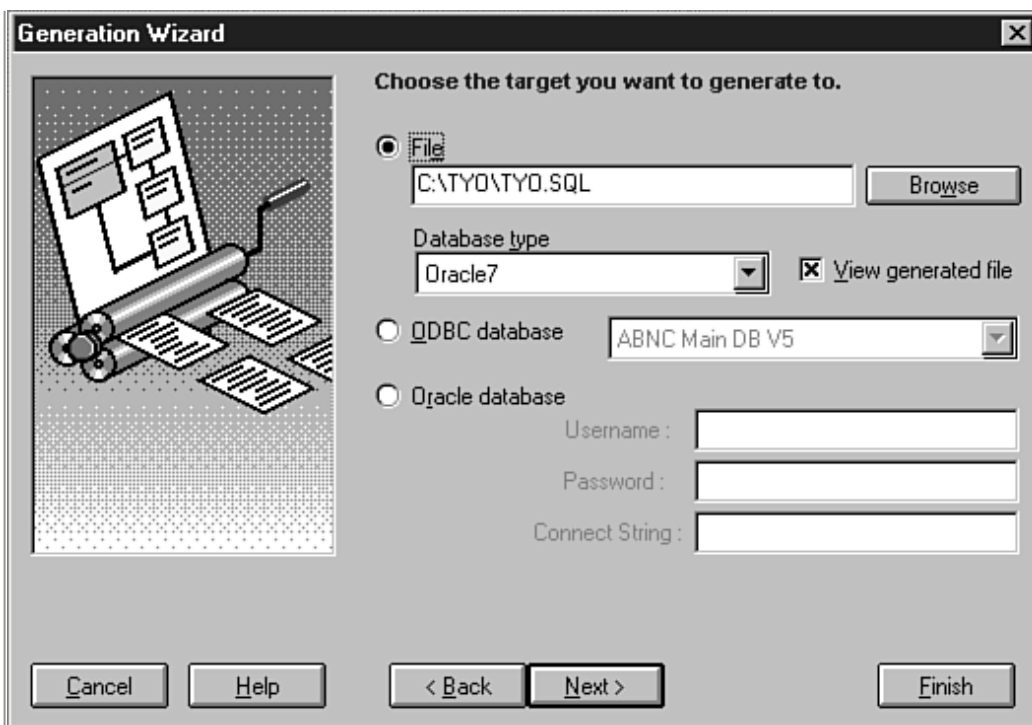


Abbildung 20.12:  
Erstes vom  
Generation Wizard  
angezeigtes Fenster

Abbildung 20.13:  
Senden von DLL-  
Anweisungen  
an eine Datei



An diesem Punkt können Sie auf Finish klicken, um die SQL-Datei zu generieren. Wenn dabei keinerlei Fehler auftreten, wird ein Dialogfenster dies anzeigen. Klicken Sie Done auf der Fortschrittsanzeige an, um fortzufahren (siehe Abbildung 20.15).

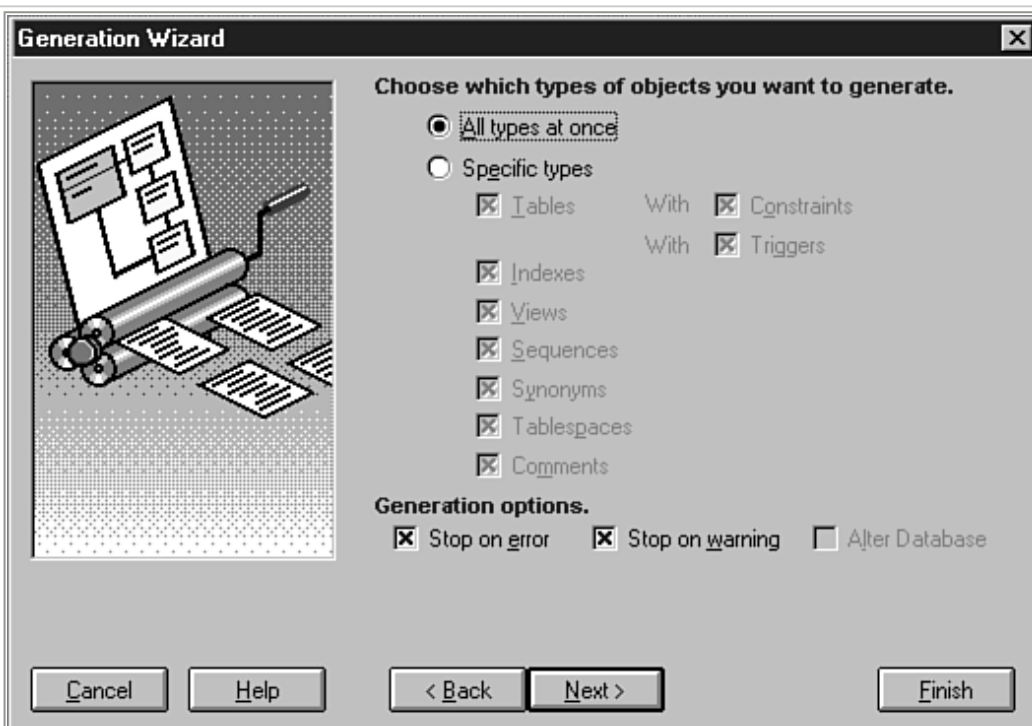


Abbildung 20.14:  
Festlegen zu  
erzeugender  
Datenbankobjekte

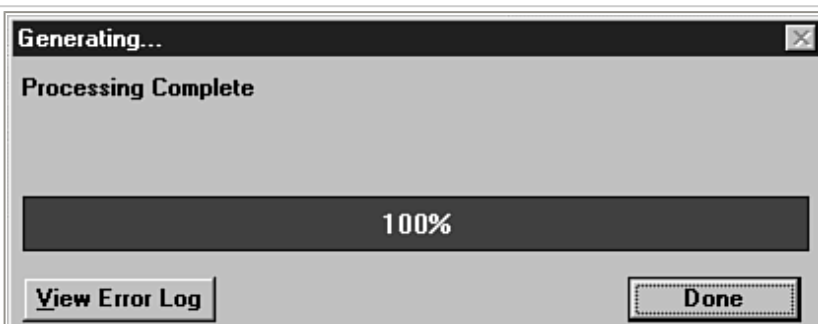


Abbildung 20.15:  
Vom Generation Wizard  
eingebblendete Anzeige für den  
Fortschritt des  
Erzeugungsprozesses

Wenn Sie auf die Taskleiste klicken, werden Sie sehen, daß der Database Designer das Notepad mit der generierten SQL-Datei geöffnet hat (siehe Listing 20.1).

***Listing 20.1: Vom Generation Wizard erstellte SQL-Datei***

```
- C:\TYO\TYO.SQL
-
- Generated for Oracle 7 on 12/5/1997 17:47:07 by Oracle Database Designer v1.00
/* Source Diagram Details:
* Name C:\TYO\TYO.ODD created 6/4/1997 18:55:25
* Last modified 11/5/1997 22:51:13
*/

CREATE TABLE DEPARTMENT
(DEPARTMENT_ID VARCHAR2(20) NOT NULL
,DESCRIPTION VARCHAR2(25)
);

CREATE TABLE INSTRUCTOR
(INSTRUCTOR_ID VARCHAR2(20) NOT NULL
, LAST_NAME VARCHAR2(25) NOT NULL
, FIRST_NAME VARCHAR2(25)
, MI VARCHAR2(1)
, POSITION VARCHAR2(25) NOT NULL
, TELEPHONE VARCHAR2(10)
, FAX VARCHAR2(10)
, EMAIL VARCHAR2(100)
, DEPARTMENT_ID VARCHAR2(20) NOT NULL
);

ALTER TABLE DEPARTMENT
ADD CONSTRAINT DEPARTMENT_PK PRIMARY KEY
(DEPARTMENT_ID) ;

ALTER TABLE INSTRUCTOR
ADD CONSTRAINT INSTRUCTOR_PK PRIMARY KEY
(INSTRUCTOR_ID) ;

ALTER TABLE INSTRUCTOR
ADD CONSTRAINT DEPARTMENT_FK
FOREIGN KEY
(DEPARTMENT_ID)
REFERENCES DEPARTMENT
(DEPARTMENT_ID)
;
```

## Eine Oracle-Datenbank mit Hilfe des Database Designers nach der Umkehrtechnik entwickeln

Database Designer kann außerdem ein Datenmodell nach der Umkehrtechnik aus verschiedenen Quellen entwickeln - einer SQL-Datei, die DDL-Anweisungen enthält, einer ODBC-Datenquelle oder einer Oracle-Datenbank. Die Möglichkeit, ein Datenmodell rückzuübersetzen, ist manchmal notwendig. Als Entwickler können Sie an eine bestehende Datenbank ohne begleitende Dokumentation geraten. Indem Sie diese Datenbank mit Hilfe der Umkehrtechnik rückübersetzen, kann der Database Designer ein Diagramm konstruieren, das Ihnen beim Verständnis des Datenmodells helfen wird. Sie sind dann in der Lage, das Modell umzukehren - vielleicht unter Hinzufügung von Dateneinheiten und Attributen -, indem Sie das bestehende Datenmodell als Grundlage benutzen. Lassen Sie uns die einzelnen Schritte durchgehen, die zur Rückübersetzung eines Datenmodells aus einer SQL-Datei notwendig sind.

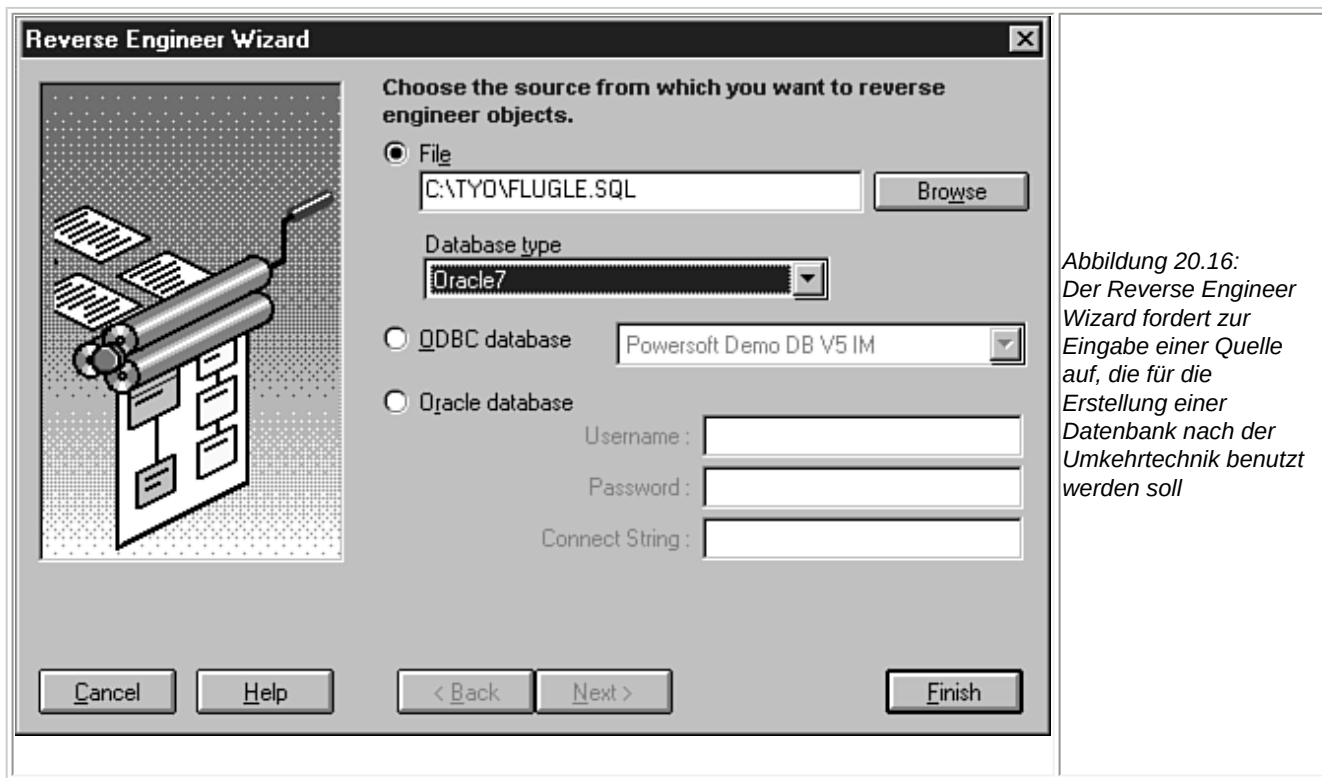
1. Um den Reverse Engineer Wizard aufzurufen, wählen Sie den Menüeintrag Tools | Reverse Engineer Wizard.

2. Auf dem dann erscheinenden Fenster können Sie die Quelle für die Rückübersetzung wählen:

- Eine SQL-Datei
- Eine ODBC-Datenquelle
- Eine Oracle-Datenbank

3. Standardmäßig vermutet der Reverse Engineer Wizard, daß Sie ein Datenmodell unter Verwendung einer SQL-Datei als Quelle rückübersetzen wollen. Um eine bestimmte SQL-Datei auszuwählen, können Sie den Dateinamen direkt eingeben oder Browse benutzen.

4. Wie Sie in Abbildung 20.16 sehen können, vermutet der Reverse Engineer Wizard ebenfalls, daß die Quelle eine Oracle7-Datenbank darstellt - mit anderen Worten, die DDL-Anweisungen in der SQL-Datei sind für die Verwendung mit einer Oracle7-Datenbank gedacht.



5. Nachdem Sie eine SQL-Datei bestimmt haben, klicken Sie auf Finish, um mit dem Umkehrungsprozeß fortzufahren.

6. Eine Fortschrittsanzeige wird erscheinen, die Meldungen über die aus der SQL-Datei erzeugten Objekte anzeigt. Wenn während des Umkehrungsprozesses keine Fehler auftreten, wird ein Fenster Sie darauf hinweisen, daß dies der Fall ist.

7. Um das Datenmodell anzusehen, klicken Sie Done auf der Fortschrittsanzeige an.



Wie Sie in Abbildung 20.17 sehen können, wird der Reverse Engineer Wizard die Objekte nach den in der SQL-Datei referenzierten Objekten konstruieren. Sie müssen vielleicht einige der Objekte neu positionieren; da sie überlappen können, was das Lesen des Diagramms erschwert. Sie können das Diagramm ausdrucken, um sich alle Objekte und ihre Beziehungen anzusehen.

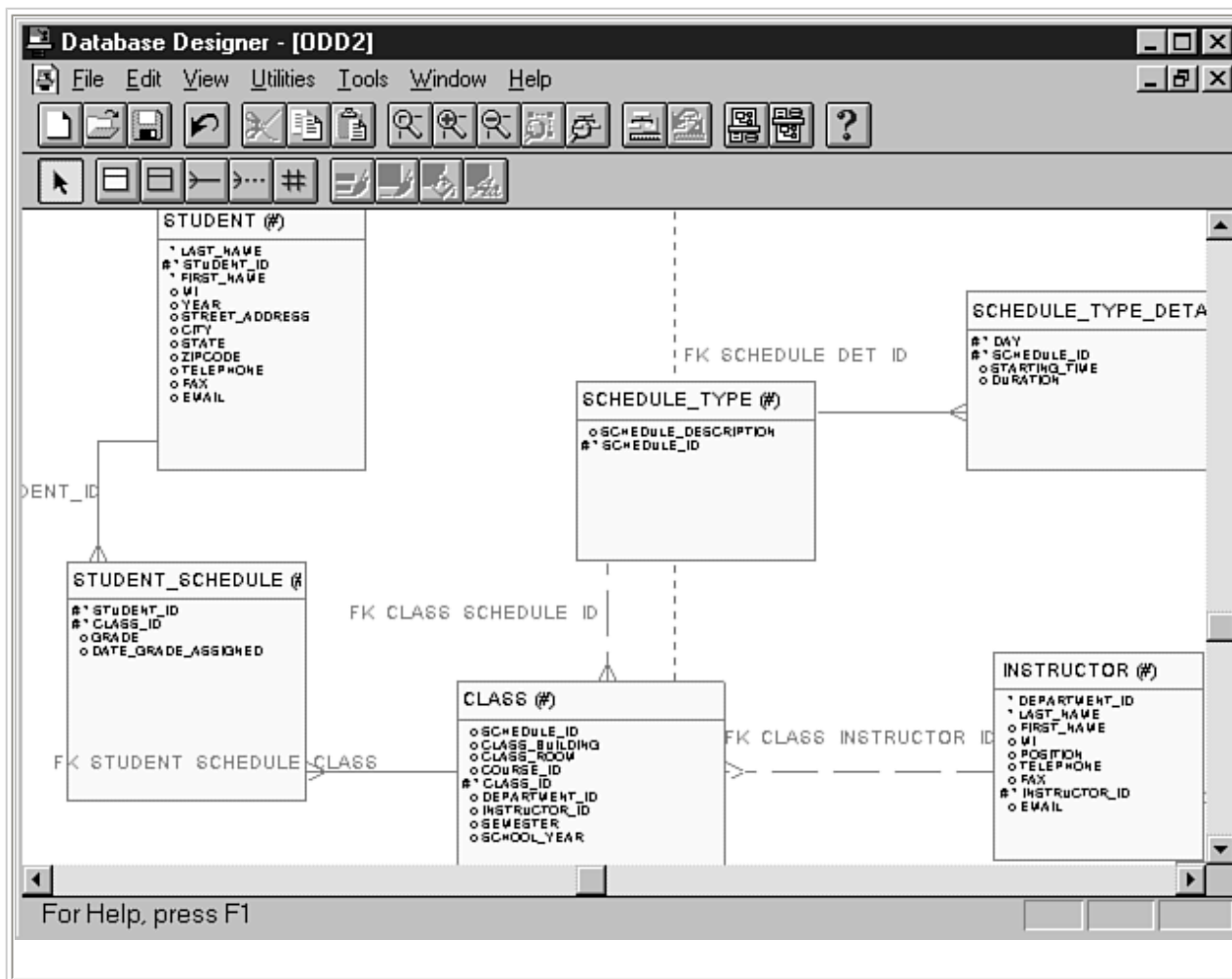


Abbildung 20.17:  
Vom Reverse  
Engineer Wizard  
erstelltes Diagramm

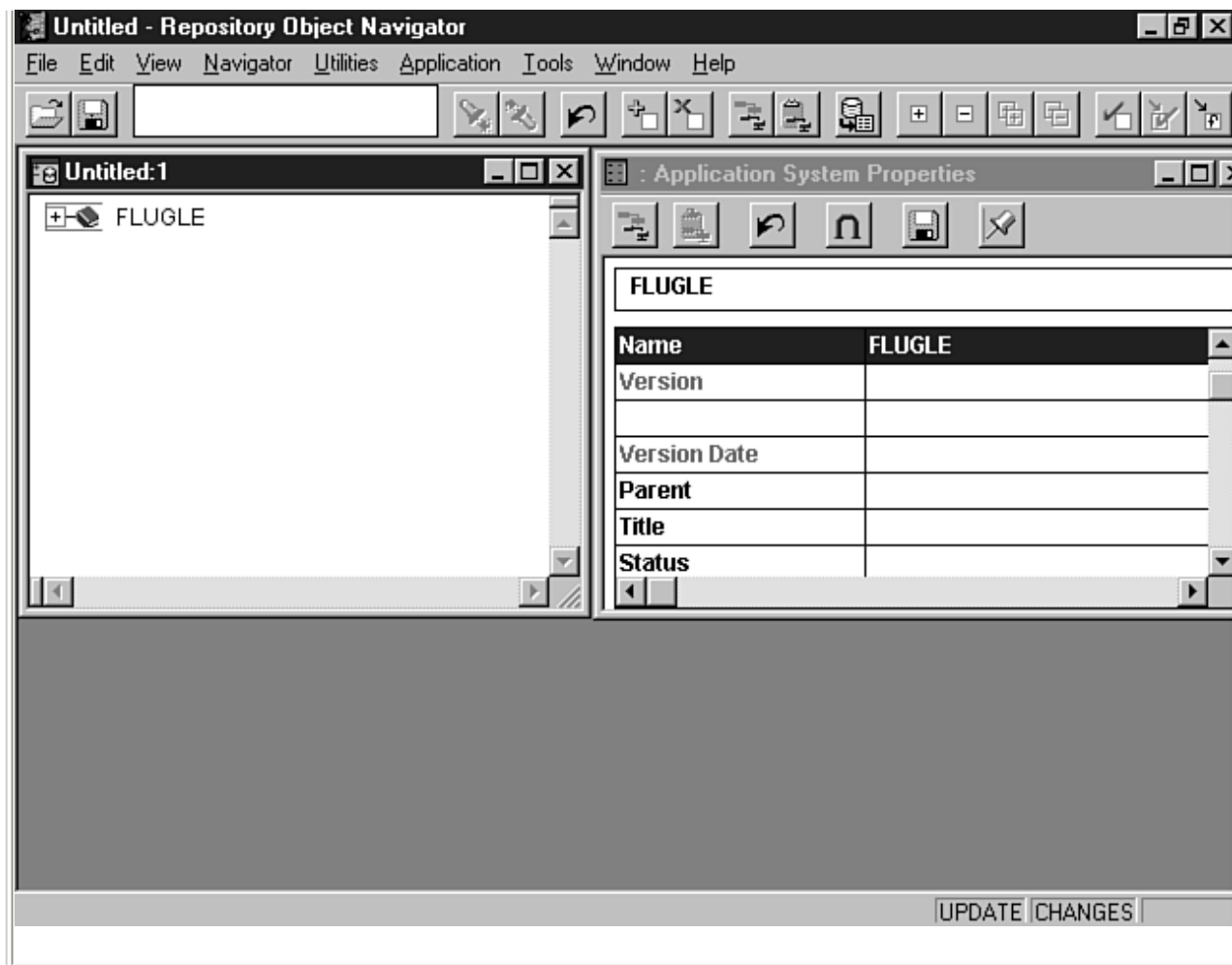
## Die Grundlagen von Designer/2000

Die Komponenten von Designer/2000 können in diese Kategorien eingeteilt werden:

- **Prozeßmodellierung:** Besteht aus dem Process Modeller
- **Systemmodellierung:** Umfaßt den Entity Relationship Diagrammer, Function Hierarchy Diagrammer und Dataflow Diagrammer
- **Design Wizards:** Umfaßt Datenbank-Design- und Anwendungs-Design-Wizards
- **System Design:** Umfaßt Data Diagrammer, Modul Structure Diagrammer, Module Data Diagrammer, Preferences Navigator und Modul Logic Navigator
- **Generierung:** Umfaßt den Server-, Forms-, Reports-, Graphics-, Visual Basic-, Webserver-, MS-Help- und C++-Object-Layer-Generator
- **Repository-Administration:** Besteht aus der Repository Administration Utility

Lassen Sie uns die Rolle diskutieren, die einige dieser Komponenten innerhalb des Entwicklungsprozesses einer Anwendung spielen. Um eine neue Anwendung zu erstellen, benutzen Sie den Repository Object Navigator, und wählen Sie den Menüeintrag File | New. Eine neue Anwendung wird erzeugt, die den Namen Flugle besitzt (siehe Abbildung 20.18).

Abbildung 20.18:  
Erzeugung einer  
neuen Anwendung  
mit dem Repository  
Object Navigator



## Process Modeller

Der Process Modeller ist ein Werkzeug, das Ihnen die grafische Beschreibung der Geschäftsprozesse ermöglicht, die Sie modellieren wollen. Der Process Modeller wird in erster Linie benutzt, um die Prozesse in ihrer gegenwärtig existierenden Form zu verstehen. Das Werkzeug ist in der Lage, einen Prozeß zu animieren, was die Dynamik leichter verständlich macht. Ausgehend vom Verständnis der gegenwärtigen Prozesse können diese neu geformt werden, um Zykluszeiten zu reduzieren oder um unnötige Arbeitsschritte zu eliminieren.

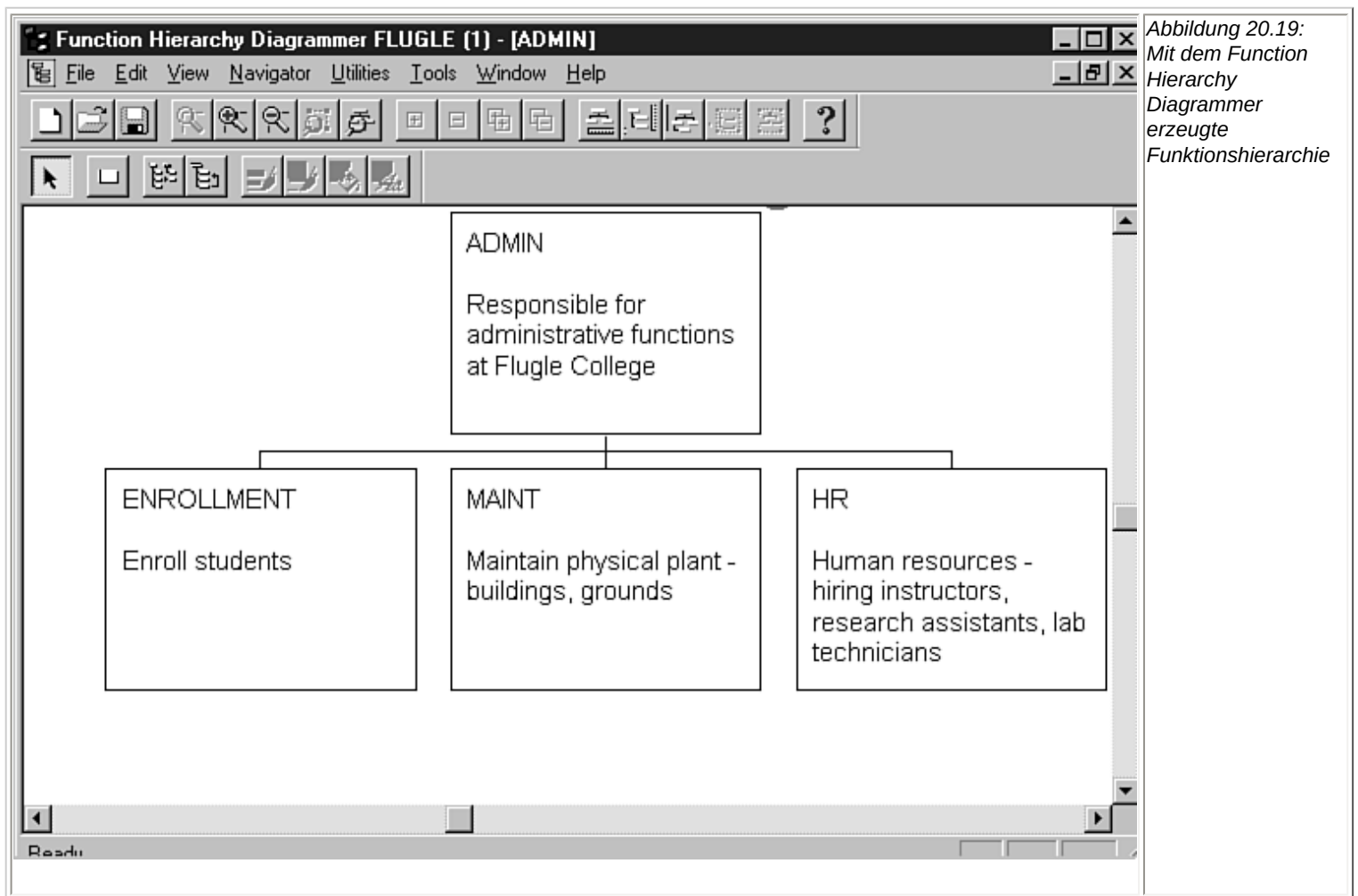
## Entity Relationship Diagrammer

Der Entity Relationship Diagrammer ist dem Database Designer ähnlich, bietet aber mehr Möglichkeiten. Der Entity Relationship Diagrammer unterscheidet speziell zwischen dem logischen Datenmodell - zusammengesetzt aus Dateneinheiten, Attributen, Domänen und Beziehungen - und dem physischen Datenbank-Design - bestehend aus Tabellen, Spalten, Datenansichten, Integritätsregeln und anderen Objekten. Wie der Database Designer, ermöglicht es auch der Entity Relationship Diagrammer dem Designer, das Datenmodell grafisch darzustellen. Wie die anderen Komponenten von Designer/2000 benutzt auch der Entity Relationship Diagrammer die Designer/2000 Repository zur Speicherung des Datenmodells für jede Anwendung.

## Function Hierarchy Diagrammer

Sie verwenden den Function Hierarchy Diagrammer, um die Hierarchie von Geschäftsabläufen, die von einem Unternehmen ausgeführt werden, grafisch zu beschreiben. Sie könnten zum Beispiel auf dem obersten Level eine Funktion mit dem Namen Manage Human Resources beschreiben. Innerhalb der Funktion gibt es auf niedrigerem Level weitere Funktionen, wie Recruit New Instructors, Hire New Instructors und so weiter. Mit dem Function Hierarchy Diagrammer können Sie die Verwendung von Dateneinheiten und Attributen in jeder Funktion identifizieren. Abbildung 20.19 veranschaulicht als ein Beispiel eine einfache Funktionshierarchie für die administrative Funktion am Flugle College.





## Datenbankentwicklung und Assistenten für die Anwendungsentwicklung

Der Database Design Wizard ist ein Werkzeug, das SQL-Anweisungen aus einem Entity-Relationship-Diagramm generiert und damit die physische Datenbank aus dem logischen Modell erzeugt, das der Designer/2000 in seiner Repository enthält. Ähnlich dazu generiert der Application Design Wizard ein Modul oder ein Menü aus einer Funktion, die in einer der Developer/2000-Komponenten enthalten ist - so wie Formulare oder Berichte.

## Module Structure Diagrammer

Der Module Structure Diagrammer wird benutzt, um Module und ihre Details zu erstellen und zu modifizieren. Dieses Werkzeug erlaubt es Ihnen ebenfalls, zu beschreiben, in welcher Beziehung die Module zueinander stehen. Ein Modul kann eines der folgenden sein:

- Bildschirm
- Bericht
- Menü
- PL/SQL-Unterprogramm (Paket, Prozedur oder Funktion)
- Diagramm

Abbildung 20.20 zeigt zum Beispiel eine Registerkarte für die Festlegung des Flugle-College-Kurskatalogs, bei dem es sich um einen Bericht handelt.

**Edit Module : COURSE\_CATALOG**

Definition | Implements | Parameters | Program Data | Datastructure | Access

Module | Client Details | Client Help Details | Server Details

Short Name: COURSE\_CATALOG      Name: COURSE\_CATALOG      ☒ Owned?

Purpose: To print the Flugle College Course Catalog.

Language: Oracle Reports      Type: Report

Complexity: Average      Size: 50      ☐ Complete?

OK      Cancel      Help      Text

Abbildung 20.20:  
Festlegen der  
Details für einen  
Bericht im Module  
Structure  
Diagrammer

## Zusammenfassung

Die Schlüsselkonzepte in dieser Lektion, die man sich merken sollte, waren die folgenden:

- Database Designer ist ein Datenbank-Diagrammwerkzeug, das Ihnen bei der Entwicklung eines Datenbank-Designs helfen und dieses für eine Reihe von Datenbanken erstellen kann - z.B. für Oracle, RDB, DB2 und SQL-Server.
- Mit dem Database Designer beschreiben Sie Ihr Datenbank-Design auf grafischem Wege: Tabellen, Spalten, Integritätsregeln und Indizes.
- Database Designer kann eine physische Datenbank aus dem Design generieren, das Sie festlegen.
- Database Designer ermöglicht Ihnen die Dokumentation jeder Tabelle und Spalte, wobei ein vollständiges Datenbank-Design aus einer Quelle erfaßt wird.
- Database Designer speichert ein Datenbank-Design in einer Datei mit der Erweiterung .ODD.

## Fragen und Antworten

Frage:

Können Database Designer und Designer/2000 Informationen austauschen?

Antwort:

*Ja. Natürlich benutzt Database Designer nur eine Untermenge der Informationen, die Designer/2000 in seiner Repository enthält. Database Designer ermöglicht Ihnen beispielsweise die Festlegung der Anzeigecharakteristiken einer Spalte. Diese Information kann an den Designer/2000 übergeben werden, der sie bei der Erzeugung von Bildschirmdarstellungen und Berichten benutzen wird.*

Frage:

Wenn ich eine kleine Anwendung entwickle, brauche ich dann wirklich ein Werkzeug wie den Database Designer oder ERwin?

Antwort:

*Es ist in der Tat eine gute Idee, ein ER-Diagrammwerkzeug wie den Database Designer oder ERwin zu benutzen. Wie Sie wahrscheinlich schon wissen, sind kleine Projekte oft ein typischer Fall von »schleichender Aufgabenerweiterung«; dies bedeutet, daß der*



*Aufgabenbereich des Projektes von vielen der daran Teilhabenden erweitert wird. Database Designer und ERwin helfen Ihnen wirklich bei der Verwaltung einer wachsenden Zahl von Dateneinheiten, Attributen und Beziehungen.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

### Test

1. Richtig oder falsch? Ein Nachteil vom Database Designer ist, daß er nur Oracle-Datenbanken unterstützt.
2. Nennen Sie drei Vorteile der Verwendung des Database Designer anstelle eines Texteditors zur Erstellung der DDL-Anweisungen, die eine Datenbank erzeugen.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖Kapitel❖ [Top](#) [Inhalt](#) [Index](#) **SAMS** Kapitel❖

# 21

## Oracle - Die nächste Generation

[Anbindung einer Oracle-Datenbank an das Web: Oracle WebServer verwenden](#)

[Elemente des Oracle WebServer](#)

[Developer/2000 for the Web](#)

[Die Network Computing Architecture von Oracle](#)

[Zusammenfassung](#)

[Fragen und Antworten](#)

[Workshop](#)

Der Zweck dieser letzten Lektion ist die Untersuchung folgender drei Themen:

- Der Oracle Web Application Server, Version 3.0 - die neueste Version des Oracle Web Servers, welche die Zugriffsmöglichkeiten einer Web-Anwendung auf eine Oracle-Datenbank durch die Verwendung von Cartridges deutlich erweitert.
- Developer/2000 for the Web, Version 1.4W - eine neue Ausgabe von Developer/2000, welche den Einsatz von Formularen und Berichten in Anwendungen unterstützt, Client-Server- ebenso wie Web-Anwendungen.
- Oracles Network Computing Architecture - eine bedeutende Initiative, welche die Rolle des Internet und des World Wide Web, die wachsenden Bedeutung von verteilten Objekten und die Unterstützung für offene Standards hervorhebt.

Außerdem sollten Sie vielleicht den Bonustag 23, »Oracle8: Neue Merkmale«, lesen, der neue Möglichkeiten des Oracle8-Servers vorstellt, die für Datenbank- und Anwendungsentwickler von Interesse sind.

### Anbindung einer Oracle-Datenbank an das Web: Oracle WebServer verwenden

Im Jahre 1995 führte die Oracle Corporation ein Produkt ein, das den Zugriff auf eine Oracle-Datenbank von einem Web-Browser aus ermöglicht: Oracle WebServer. Die erste Version des WebServer bestand aus verschiedenen Komponenten:

- Ein oder mehrere Web-Listener, die auf HTTP-Verbindungsanrufe warten
- Web-Agenten, die mit einer lokalen oder entfernten Oracle-Datenbank mittels eines bestimmten Oracle-Accounts verbunden sind
- Zwei eingebaute PL/SQL-Pakete, die Prozeduren und Funktionen enthalten, welche HTML-Einträge erzeugen



*Die in dieser Lektion gezeigten Beispiele basieren auf der Erstellung einer WebServer-Anwendung, die eine gespeicherte Prozedur*

*aufruft, welche in PL/SQL geschrieben ist. Diese gespeicherte Prozedur generiert dynamisch den HTML-Code, der zur Anzeige der Inhalte der Student-Tabelle notwendig ist.*

---

Seitdem wurde Oracle WebServer deutlich erweitert. Die gegenwärtige Version 3.0 ist für die Unterstützung der Network Computing Architecture entwickelt worden, Oracles großer Initiative zur Bereitstellung einer offenen, plattformunabhängigen, objektbasierten, Web-basierten und mit verteilten Objekten arbeitenden Architektur.

Heute werden Sie sich die Konfiguration von WebServer 3.0 ansehen. Außerdem werden Sie durch die Anwendung von gespeicherten Prozeduren einige dynamische Web-Seiten erzeugen, die Komponenten aus den http- und htf-Paketen aufrufen.

## Elemente des Oracle WebServer

Der Oracle WebServer besteht aus mehreren Hauptkomponenten:

- **Web Listener:** Ein Web Listener namens admin wird während der Installation des Produktes erzeugt. Als Standardeinstellung überwacht der Listener admin den Port 8888. Ein zweiter Web Listener namens www wird ebenfalls während der WebServer-Installation erzeugt. Dieser überwacht als Listener den Port 80. Sie können jedoch die Konfiguration verändern, einschließlich der Ports, die sie bedienen.
- **Web Request Broker:** Dieses Programm bildet das Herz des WebServer-Produkts. Es dispatcht HTTP-Anfragen an andere WebServer-Programme, abhängig vom Typ der Anfrage.
- **Database Access Descriptor:** Eine Datei, die beschreibt, wie ein PL/SQL-Agent eine Verbindung zu einer Oracle-Datenbank aufbaut. Der Database Access Descriptor schließt den Namen des Oracle-Accounts mit ein, der für die Verbindung benutzt wird, und die Oracle SID, welche die Datenbank identifiziert, mit welcher der Agent die Verbindung aufbauen will. Alternativ dazu kann der Database Access Descriptor einen SQL\*Net2-Datenbank-Alias beschreiben, der für die Oracle-Verbindung benutzt wird.
- **Cartridges:** WebServer enthält verschiedene Cartridges, von denen jede eine andere Programmiersprache unterstützt. Zusammen mit dem Produkt werden Cartridges für PL/SQL, Java, Perl und ODBC ausgeliefert (was Ihnen den Zugriff auf Nicht-Oracle-Datenbanken mittels des Oracle WebServer ermöglicht).
- **Web Agent:** Ein gegebener Cartridge-Typ kann eine oder mehrere Web-Agenten unterstützen. Jeder Web-Agent steht in Verbindung mit einem bestimmten Database Access Descriptor, der identifiziert, wie eine Verbindung zu einer Oracle-Datenbank aufgebaut werden soll.

Der Oracle WebServer muß auf einem Server wie Solaris, HP-UX oder Windows NT installiert sein. Nachdem Sie die grundlegende Installation ausgeführt haben, können Sie die Installation und die Konfiguration mit einem Web-Browser abschließen. Dazu sollten Sie die URL <http://server-name:8888> benutzen, wobei server-name der Name des Servers ist, einschließlich seiner Domäne. Wie Sie in Abbildung 21.1 sehen können, wird die Startseite neun Links anzeigen, zwischen denen Sie wählen können.

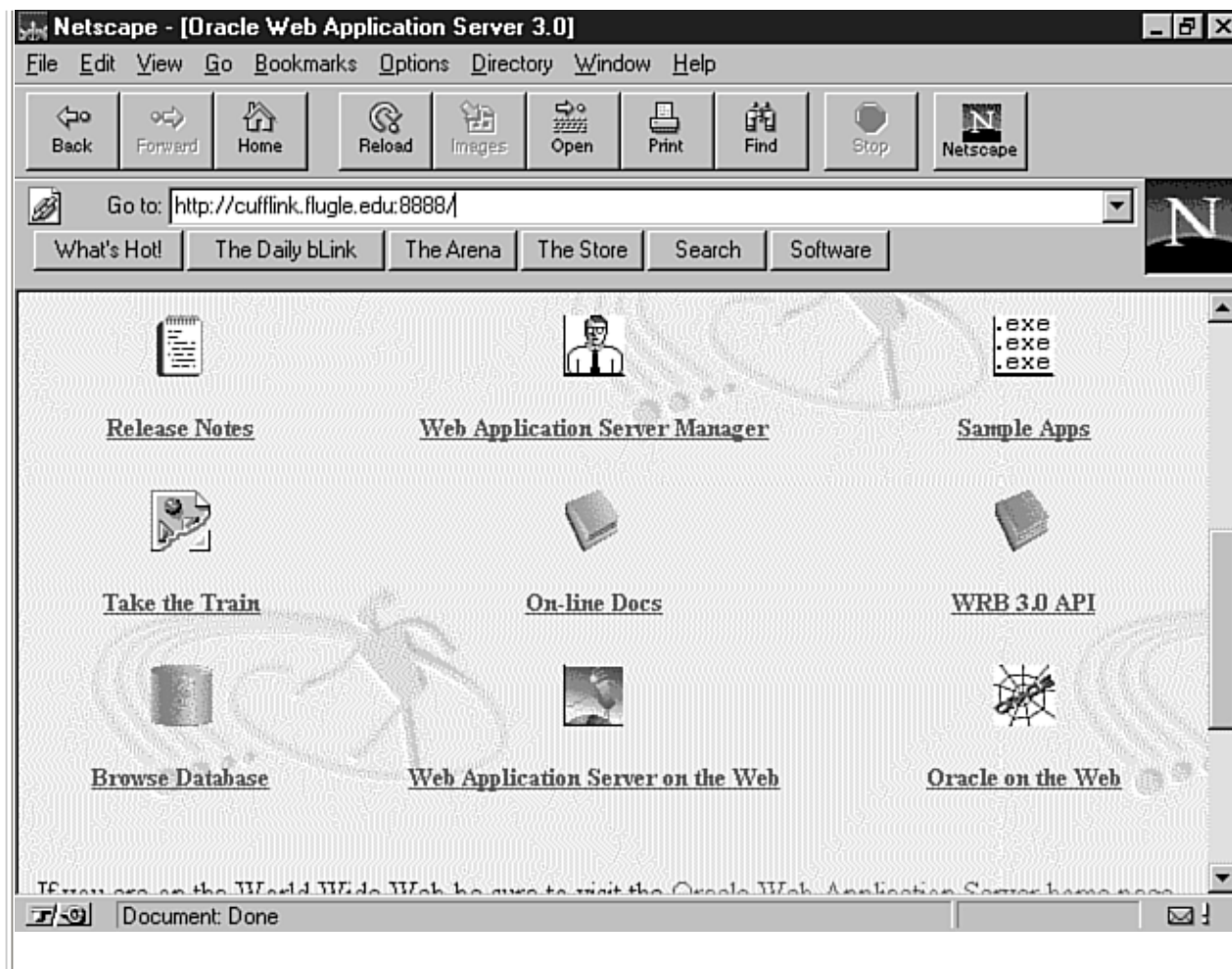


Abbildung 21.1:  
Die zu Beginn vom  
Oracle WebServer  
erstellte Web-Seite



Um auf die in dieser Lektion gezeigten Web-Seiten zugreifen zu können, müssen Sie das admin-Kennwort wissen, das während der Installation von WebServer festgelegt wurde.

Wenn Sie den Web Application Server Manager-Link wählen, wird eine andere Web-Seite erscheinen, die drei Links enthält (siehe Abbildung 21.2):

- Oracle-Database (ermöglicht es Ihnen, eine Oracle-Datenbank auf dem Server, auf dem Oracle WebServer installiert ist, zu beginnen oder zu beenden)
- Oracle Web Listener
- Oracle Web Application Server

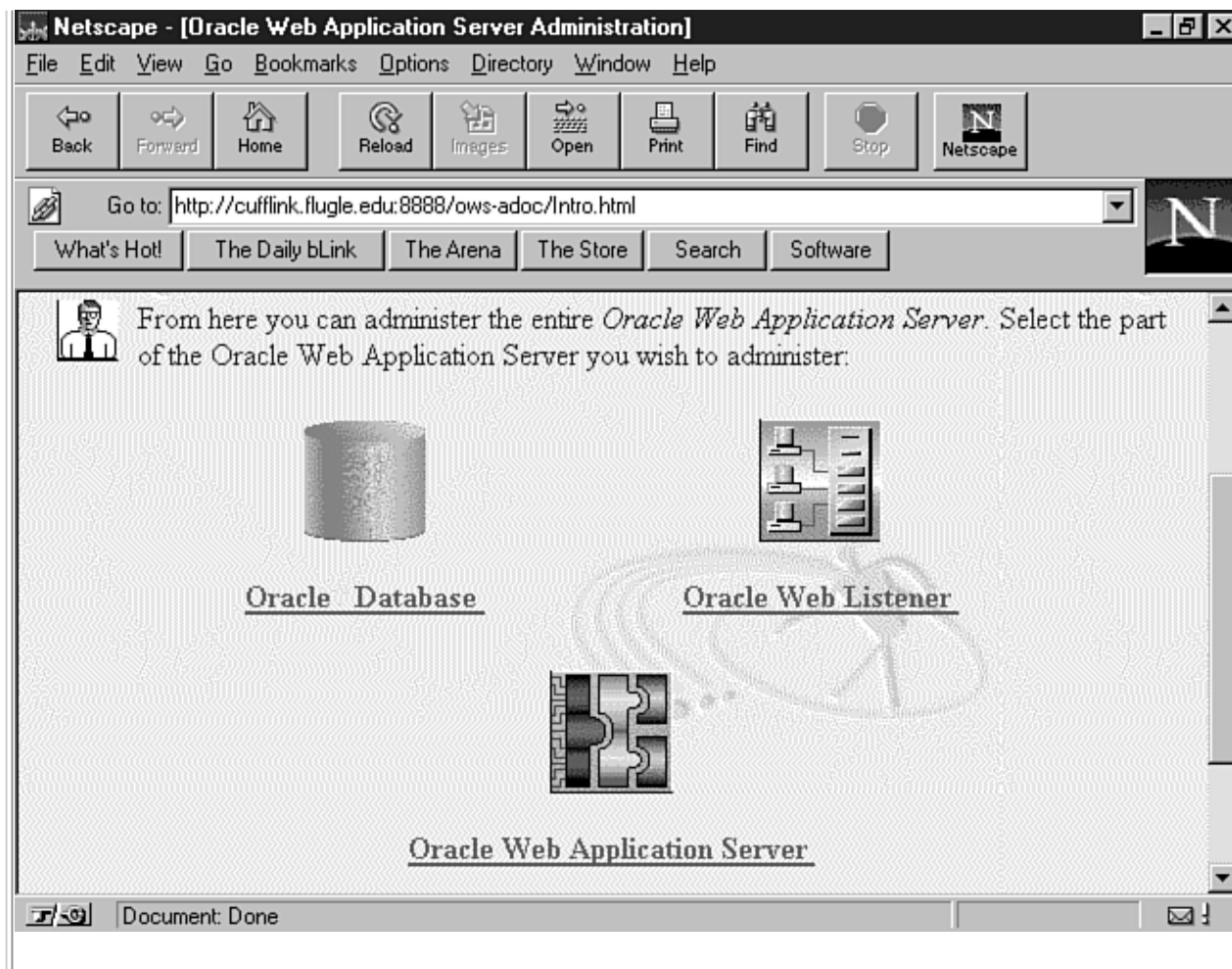


Abbildung 21.2:  
Betrachtung des Web-  
Application-Server-  
Managers

## Web Listener

Um die Web Listener zu konfigurieren, wählen Sie den Oracle Web Listener Administration-Link, wie er in der Abbildung 21.2 angezeigt wird. Die Seite, die erscheint, trägt den Titel Oracle Web Listener Administration. Wenn Sie die Seite herunterblättern, werden Sie zwei Web Listener angezeigt sehen: admin und www. Beide wurden während des Installationsprozesses erzeugt. Der admin-Web Listener muß jedoch laufen, um eine beliebige andere administrative Aufgabe ausführen zu können.

Sie können den www-Web Listener neu konfigurieren, indem Sie den CONFIGURE-Link wählen. Eine Seite mit dem Titel Oracle Web Listener Advanced Configuration for Listener www wird erscheinen. Wenn Sie eine andere Verzeichniszuordnung hinzufügen wollen, können Sie den Rahmen rechts herunterblättern, bis Sie die Überschrift Directory Mappings sehen. Vielleicht wollen Sie zu dieser Liste eine Verzeichniszuordnung für jeden Web-Agenten hinzufügen, den Sie erstellen wollen. Abbildung 21.3 zum Beispiel zeigt eine Verzeichniszuordnung /flugle/owa für den Web-Agenten des Flugle College, der in Kürze erzeugt wird. Blättern Sie herunter bis zur Schaltfläche Modify Listener, und klicken Sie darauf. Um zur vorherigen Seite zurückzukehren, können Sie die Schaltfläche Admin im unteren Rahmen anklicken.



Nachdem Sie einen Web Listener verändert haben, müssen Sie ihn neu starten, damit die Änderungen wirksam werden.

## Web Application Server Administration

Um auf die Web-Application-Server-Administration-Seite zugreifen zu können, wählen Sie den Web Application Server-Link (wie in der letzten Abbildung 21.2 gezeigt wurde). Eine andere Seite mit dem Titel Oracle Web Application Server Administration wird erscheinen. Wie Sie in Abbildung 21.4 sehen können, werden auf dieser Seite neun Links angezeigt:



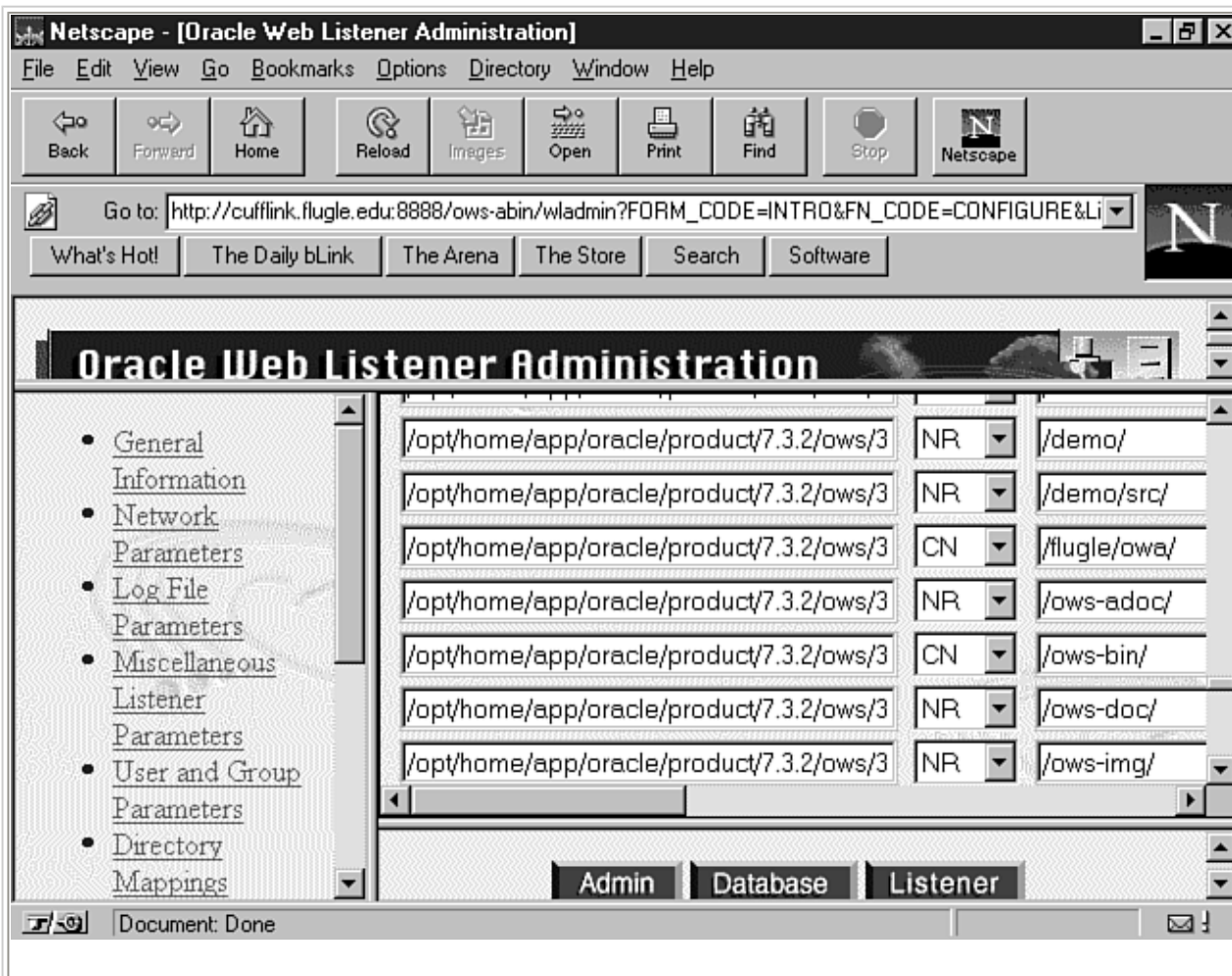


Abbildung 21.3:  
Erstellen einer  
Verzeichniszuordnung  
für den Web Listener

#### Authorization Server

Mit dieser Komponente können Sie Authentifizierungs-Schemata definieren, die mit dem Web Application Server benutzt werden sollen, einschließlich Basic, Digest und IP-basiert.

#### Logger

Diese Komponente sammelt automatisch statistische Angaben in Bezug auf HTTP-Anfragen. Sie können den Logger auch so konfigurieren, daß er die Ausführung anderer WebServer-Komponenten verfolgt.

#### Log Analyzer

Mit dieser Komponente können Sie HTTP-Anfrage-Log-Eintragungen einsehen und verschiedene Berichte erzeugen, welche die Ausführung von Oracle WebServer aufsummieren.

#### Migration

Zusätzlich zur Migration von WebServer 2.0 auf WebServer 3.0 ermöglicht Ihnen diese Komponente die Konfiguration des Netscape Servers für die Benutzung mit dem Oracle Web Broker.

#### DAD Administration

Diese Komponente ermöglicht Ihnen das Erstellen und Verändern von Datenbank-Zugriffs-Deskriptoren.

#### Web Request Broker

Dies ist ein Link auf die Web Request Broker Administration-Seite.

|                          |                                                                                                                                                                                                                                    |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Default Configuration    | Wenn Sie diesen Link wählen, können Sie einen Satz von Standardbenutzern und PL/SQL-Paketen in einer Oracle-Datenbank installieren.                                                                                                |
| Support                  | Dieser Link erlaubt Ihnen das Ausfüllen eines Oracle-Support-Formulars.                                                                                                                                                            |
| Cartridge Administration | Mit diesem Link können Sie einen Cartridge-Typ wie PL/SQL, Java oder Perl auswählen. Haben Sie einmal einen Cartridge-Typ gewählt, können Sie einen Web-Agenten für diesen Typ ansehen, erstellen, neu konfigurieren oder löschen. |

Die folgenden Abschnitte zeigen einen näheren Blick auf einige dieser Links.

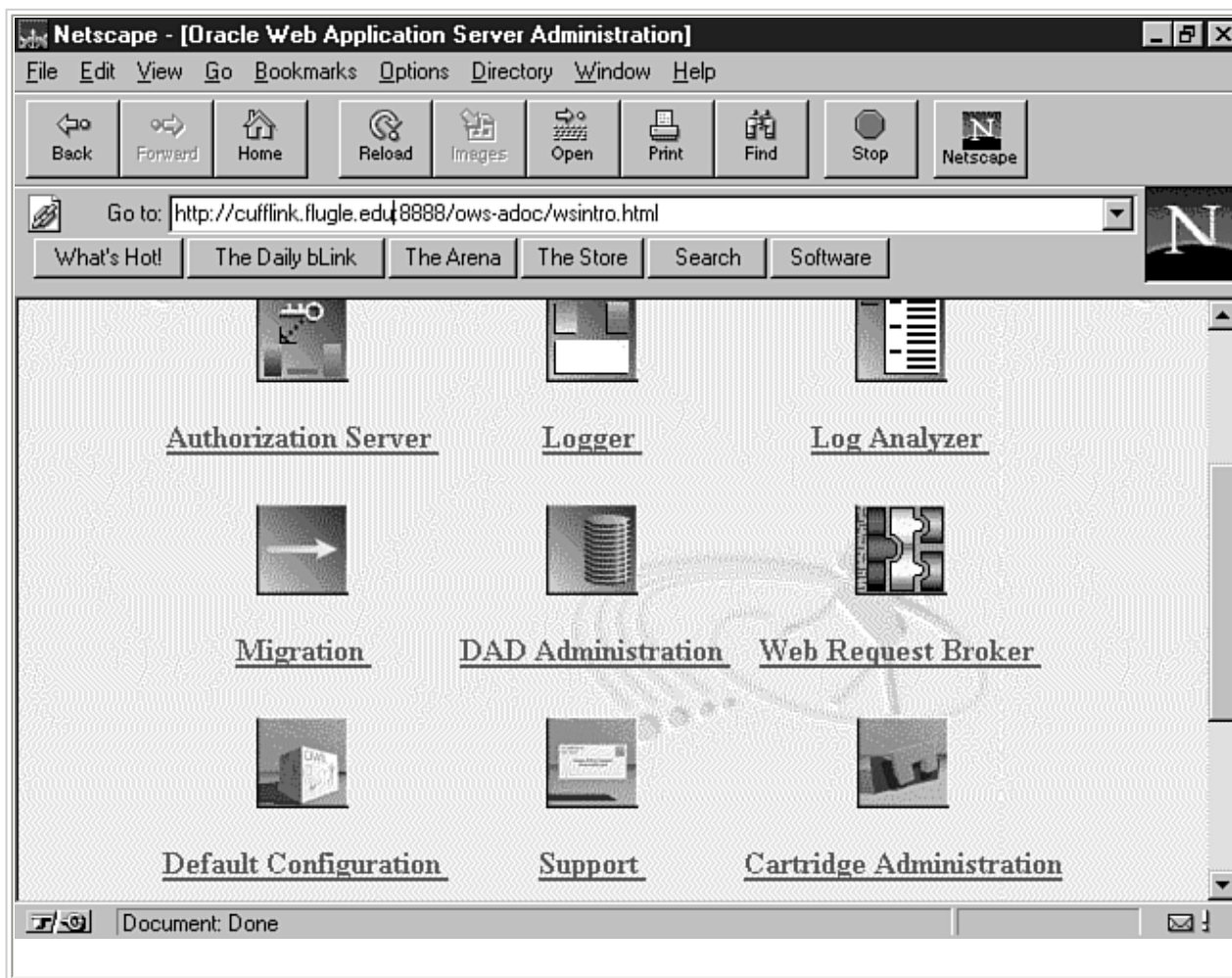


Abbildung 21.4:  
Auswahlmöglichkeiten  
für die Konfiguration  
im Web Application  
Server

## Einen Database Access Descriptor (DAD) verwalten

Bevor Sie einen neuen Web-Agenten basierend auf einem bestimmten Cartridge-Typ (so wie PL/SQL) erzeugen können, müssen Sie eventuell einen neuen DAD erstellen. Dazu wählen Sie den DAD Administration-Link auf der Seite Web Application Server Administration. Auf der Seite DAD Administration werden Sie einen weiteren Link mit dem Titel Create New DAD sehen. Wählen Sie diesen Link. Eine andere Seite mit dem Titel Database Access Descriptor Creation wird erscheinen (siehe Abbildung 21.5). Auf dieser Seite gibt es mehrere Felder, in die Sie Informationen eingeben müssen:

|          |                                                            |
|----------|------------------------------------------------------------|
| DAD Name | Der Name des neuen Datenbank-Zugriffsdeskriptors (flugle). |
|----------|------------------------------------------------------------|

|                        |                                                                                                                                                                                                                                                                                  |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Database user          | Der Name des Oracle-Accounts, den der neue DAD benutzen sollte, wenn er eine Verbindung zur Oracle-Datenbank aufbaut (flugle).                                                                                                                                                   |
| Database User Password | Das Kennwort für den Datenbankbenutzer (flugle)                                                                                                                                                                                                                                  |
| Confirm Password       | Eine Bestätigung des Benutzerkennworts (flugle)                                                                                                                                                                                                                                  |
| ORACLE_HOME            | Das Server-Verzeichnis, das als ORACLE_HOME für die ORACLE_SID definiert ist (der nächste Parameter in diesem Formular). Fragen Sie Ihren DBA nach dem korrekten Wert für ORACLE_HOME.                                                                                           |
| ORACLE_SID             | Der Name der Oracle-Datenbank, mit welcher der neue DAD verbunden werden soll. Dies setzt voraus, daß sich die Oracle-Datenbank auf demselben Rechner wie der Oracle WebServer befindet. Ist dies nicht der Fall, müssen Sie einen Wert in das Feld SQL*Net V2 Service eingeben. |

Abbildung 21.5:  
Erstellen eines  
neuen Deskriptors  
für den  
Datenbankzugriff.

Wenn Sie in alle erforderlichen Felder Werte eingegeben haben, klicken Sie die Schaltfläche Submit New DAD im unteren Bereich der Seite an.

## Einen Web Request Broker verwalten

Sie werden den Web Request Broker aus wenigstens einem Grund neu konfigurieren müssen: Um ein virtuelles Verzeichnis für Ihre Anwendung festzulegen.



1. Wählen Sie den Web Request Broker-Link auf der Seite Web Application Server Administration.
2. Auf der Seite, die erscheint - Web Request Broker Administration - wählen Sie Applications and Directories im linken Laufrahmen.
3. Blättern Sie den Rahmen auf der rechten Seite herunter bis zum ersten leeren Eintrag in der Tabelle, und geben Sie diese Werte ein (siehe Abbildung 21.6):

Virtual Directory **/flugle/owa**  
 Application **PLSQL**  
 Physical Path **%ORACLE\_HOME%/bin**

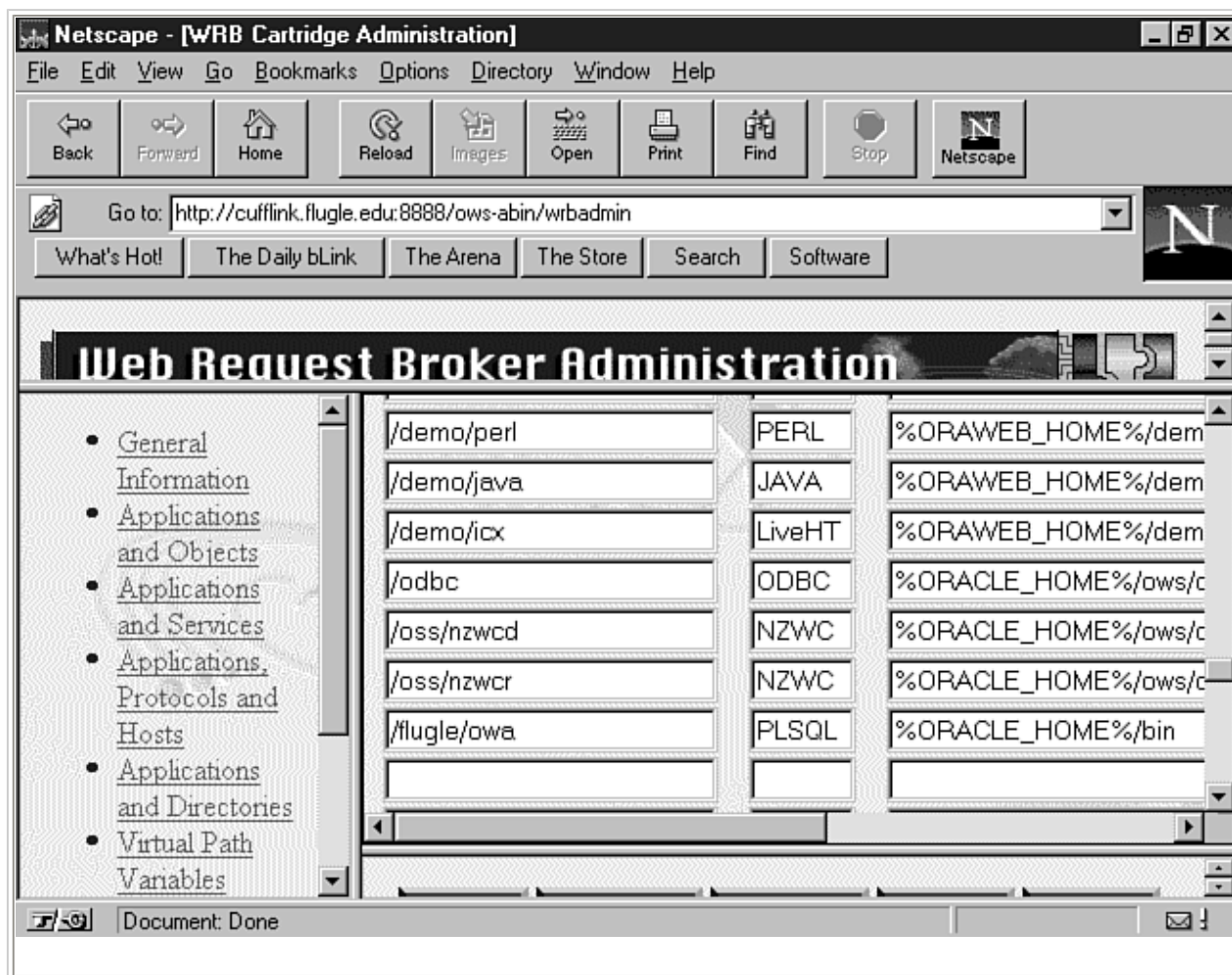


Abbildung 21.6:  
Festlegen eines  
virtuellen  
Verzeichnisses  
für eine  
Anwendung für  
den Web  
Request Broker

4. Blättern Sie herunter, bis Sie die Schaltfläche Modify WRB Configuration sehen. Klicken Sie darauf, damit die Änderungen wirksam werden.

Für eine aktuelle Anwendung wollen Sie vielleicht andere WRB-Parameter, wie den Schutz von Anwendungen, verändern.

## Cartridge-Verwaltung

Sie können den Cartridge Administration-Link auf der Seite Web Application Server Administration wählen. Wie Sie in Abbildung 21.7 sehen können, werden verschiedene Cartridge-Typen angezeigt, einschließlich:

- PLSQL
- JAVA
- PERL
- ODBC

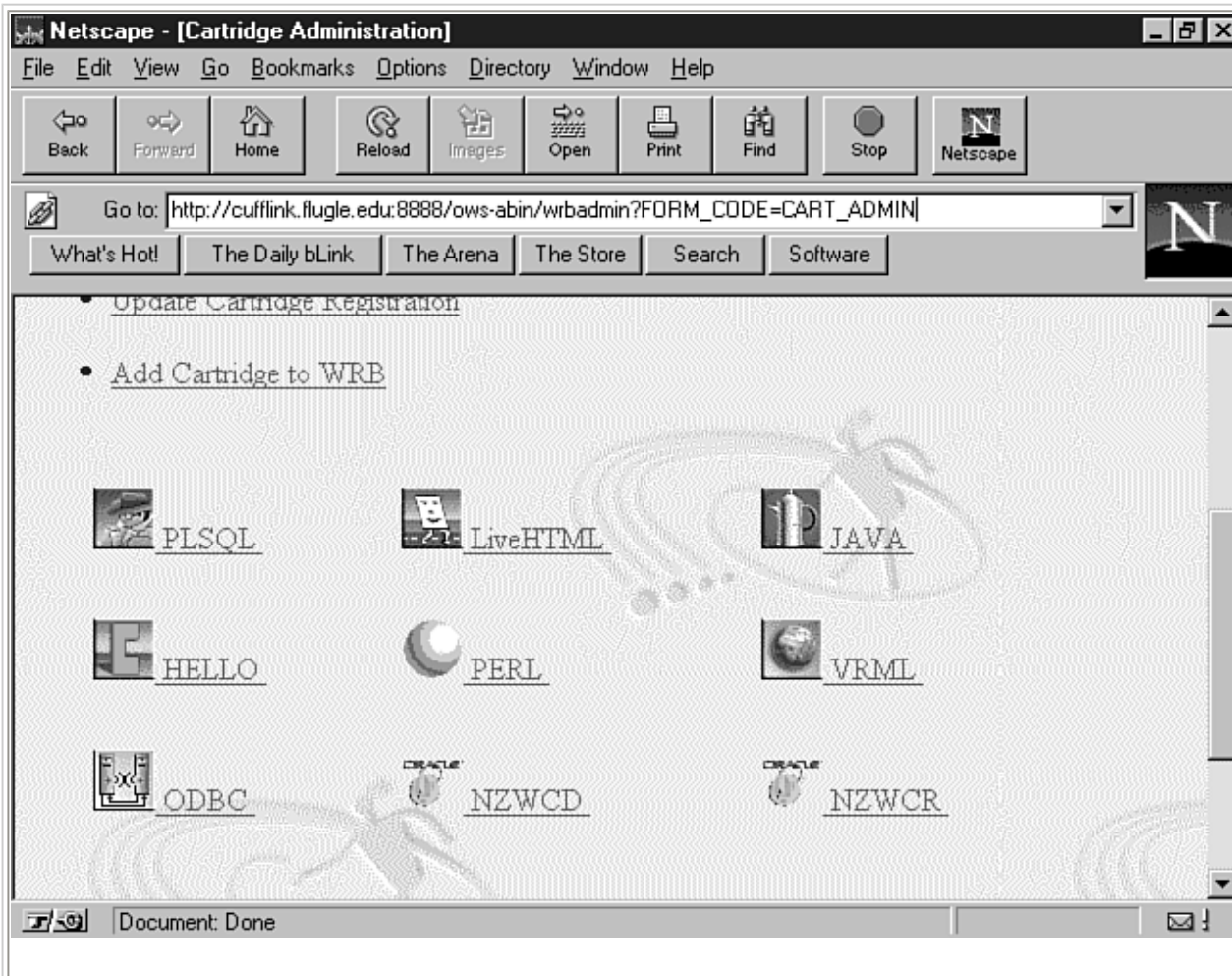


Abbildung 21.7:  
Auswahl eines Cartridge-  
Typs

Wählen Sie den Cartridge-Typ PLSQL. Eine andere Seite mit dem Titel PL/SQL Agent Administration wird angezeigt (siehe Abbildung 21.8).

Auf dieser Seite können Sie den Link namens Create New PL/SQL Agent wählen. In der nächsten Seite, die erscheint - PL/SQL Agent Creation - geben Sie diese Werte in die Felder ein:

Name of PL/SQL Agent **flugle**  
Name of DAD to be used **flugle**

Blättern Sie herunter, und klicken Sie die Befehlsschaltfläche mit der Bezeichnung Submit New PL/SQL Agent an. Oracle WebServer sollte jetzt den neuen PL/SQL-Agenten erzeugen.

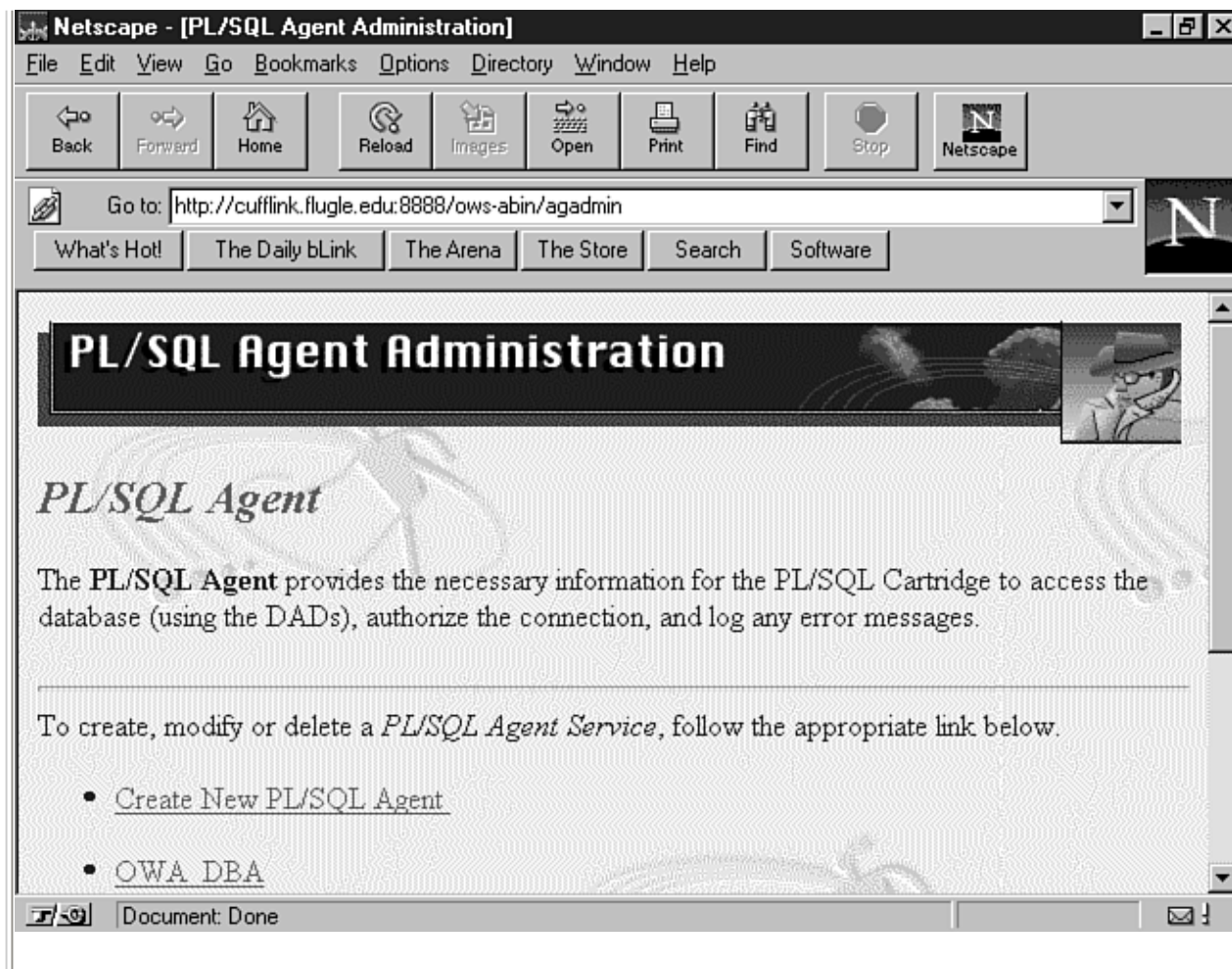


Abbildung 21.8:  
Verwaltung eines  
PL/SQL-Agenten

## Die besonderen PL/SQL-Pakete http und htf

Beiliegend zum Oracle WebServer gibt es verschiedene PL/SQL-Pakete, die standardmäßig installiert werden. Ein Paket, http, enthält eine große Anzahl von Prozeduren, die HTML-Einträge generieren. Ein anderes Paket, htf, umfaßt eine große Menge an Funktionen, die als Argumente für viele der Prozeduren in dem http-Paket benutzt werden.

## Erzeugen von Dynamic HTML mit Hilfe einer gespeicherten Prozedur

Lassen Sie uns einmal ein sehr elementares Beispiel dazu betrachten, wie eine dynamische Web-Seite von einer gespeicherten Prozedur generiert wird. Der folgende Code ist eine gespeicherte Prozedur namens StudentList, die eine Untermenge der Information über die Studenten in einem Web-Browser anzeigt, ein Student pro Zeile.

### Listing 21.1: Die Prozedur Studentlist

```
PROCEDURE StudentList IS
cursor GetStudents is
select Student_ID, Last_Name, First_Name
from Student
order by Student_ID;
S_ID Student.Student_ID%type;
LName Student.Last_Name%type;
FName Student.First_Name%type;

begin

open GetStudents;

http.HTMLOpen;
http.BodyOpen;
```

```
loop
fetch GetStudents into S_ID, LName, FName;
exit when GetStudents%NOTFOUND;

http.print ('Student ID: ' || S_ID);
http.print ('Last Name: ' || L_Name);
http.print ('First Name: ' || F_Name);

http para;

end loop;

http.BodyClose;
http.HTMLClose;

END;
```



Ein Cursor mit dem Namen *GetStudents* wird deklariert, der die *Student ID*, den *Nach-* und den *Vornamen* aus der *Student-Tabelle* auswählt. Drei andere Variablen werden deklariert - *S\_ID*, *LName* und *FName* - welche die Werte aus jeder *Cursorspalte* enthalten werden, wenn die jeweilige Zeile ausgewählt ist.

Zu Beginn der Prozedur ist der Cursor geöffnet. Zwei Prozeduren aus dem *http-Paket* werden aufgerufen: *HTMLOpen* und *BodyOpen*. Diese Pakete generieren jeweils die Einträge *<HTML>* und *<BODY>*.

Wenn der Cursor die Schleife ausführt, wird jede Zeile vom Cursor ausgewählt. Eine Prozedur namens *http.print* wird benutzt, um jeden Zeilenwert für die *Student ID* und den *Nach-* und *Vornamen* auszugeben. Ein Aufruf von *http para* wird benutzt, um den Eintrag *<P>* zu erzeugen, der einen Absatz darstellt. Zuletzt, wenn die Schleife geschlossen wird, werden *http.BodyClose* und *http.HTMLClose* aufgerufen, welche jeweils die Einträge *</BODY>* und *</HTML>* generieren.

Abbildung 21.9 zeigt die Ausgabe der gespeicherten Prozedur *StudentList*.

|  |  |
|--|--|
|  |  |
|--|--|



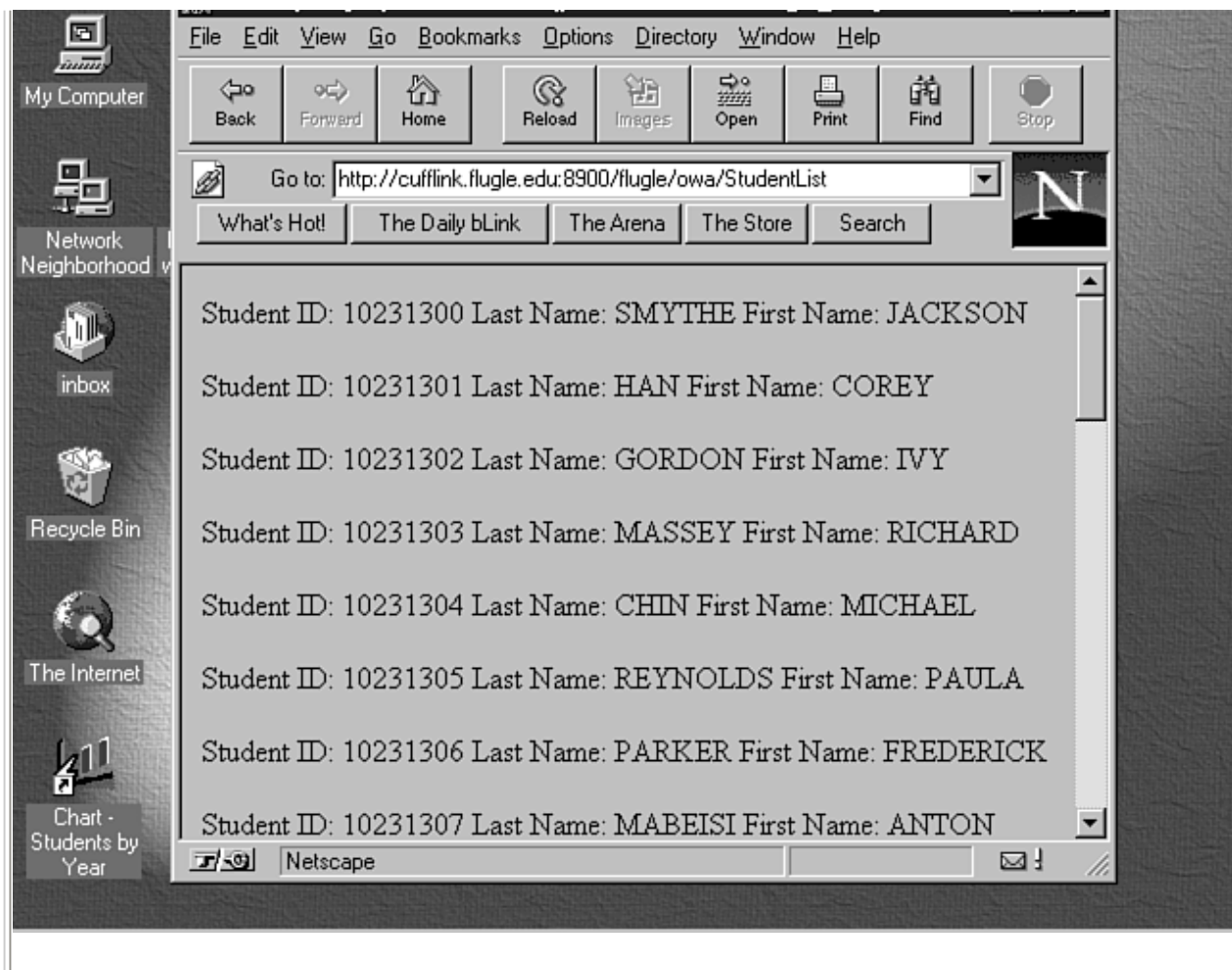


Abbildung 21.9: Beispiel für eine gespeicherte PL/SQL-Prozedur, die Dynamic HTML erzeugt

## Developer/2000 for the Web

Im Mai 1997 kündigte die Oracle Corporation die Auslieferung von Developer/2000 for the Web, Version 1.4W an. Zum Zeitpunkt der Veröffentlichung dieses Buches kann von der Oracle Website eine Testversion von Developer/2000 for the Web, Version 1.4W heruntergeladen werden. Seien Sie jedoch gewarnt - die Datei zum Herunterladen hat einen Umfang von 159 MB und ist nach dem Dekomprimieren nahezu 500 MB groß. Dieser Abschnitt konzentriert sich auf die Windows-NT-Version von Developer/2000 for the Web (es gibt auch eine Version für das Solaris-Betriebssystem).

Developer/2000 for the Web unterstützt den Einsatz von Oracle-Formularen und -Berichten als Web-Anwendungen. Developer/2000 erfordert entweder die Benutzung des Oracle WebServers 2.1 oder Oracle Web Application Servers 3.0. Developer/2000 for the Web enthält eine neue Komponente, den Forms Server Listener, der sich zusammen mit dem Web Application Server auf dem gleichen Server befinden muß. Lassen Sie uns die Architektur in Abbildung 21.10 etwas näher betrachten.

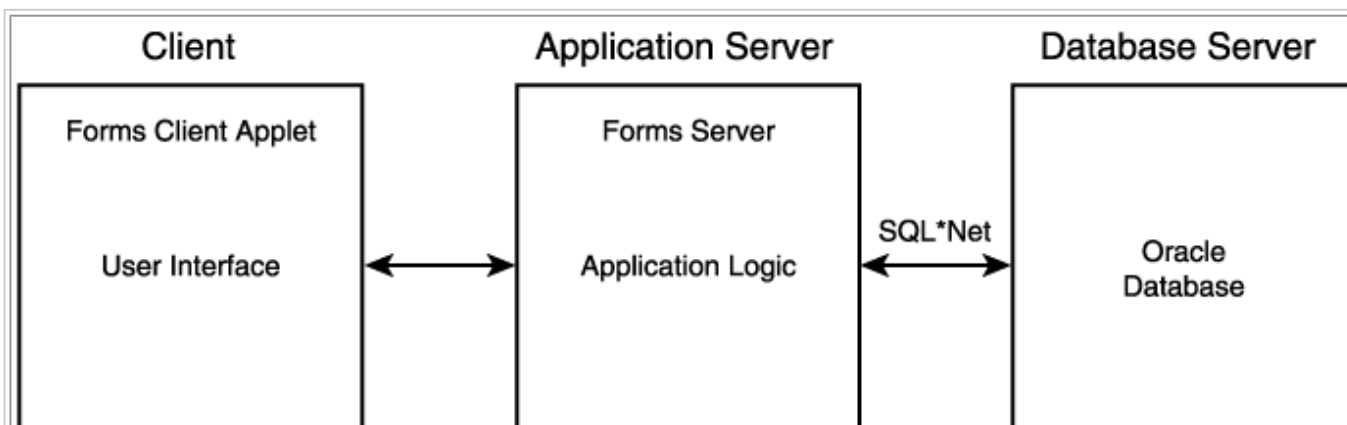


Abbildung 21.10: Architektur, die den Developer/2000 for the Web unterstützt

Wie Sie in Abbildung 21.10 sehen können, besteht die Architektur aus drei Schichten:

- dem Formular-Client, der aus einem Web-Browser besteht, der die Formular-Anwendung als Java-Applet herunterlädt,
- dem Applikations-Server, auf dem sich der WebServer und der Forms Server Listener befinden,
- dem Datenbank-Server, auf dem sich die Oracle-Datenbank befindet, welche die Formularanwendung unterstützt.

Der Client-Rechner benötigt nur einen Web-Browser, der für Java-Unterstützung eingerichtet ist (genaugenommen muß das Java Development Kit 1.1.2 oder höher unterstützt werden). Der Benutzer gibt einen URL (Uniform Resource Locator) ein, der folgendes enthält:

- den Namen des Applikations-Servers,
- den Port, den der Web Listener beobachtet,
- den Namen der Web-Anwendung,
- den Namen des Moduls,
- die Benutzer ID und den Datenbank-Alias.



*Die gegenwärtige Version von Developer/2000 for the Web, Version 1.4W, hat die Bezeichnung Limited Production. Diese Ausgabe unterstützt voll den Einsatz von Client-Server-Anwendungen, die in Oracle Forms, Reports und Graphics geschrieben sind. Developer/2000 for the Web hat jedoch begrenzte Produktionsfähigkeiten für den Einsatz von Web-Anwendungen. Zum gegenwärtigen Zeitpunkt erfordert diese Ausgabe die Benutzung eines Web-Browsers, der das Java Development Kit (JDK), Version 1.1.2 oder höher unterstützt; zum gegenwärtigen Zeitpunkt gibt es keine käuflich zu erwerbenden Browser, die diese Unterstützung bieten. Als Konsequenz daraus schließt die Installation von Developer/2000 for the Web einen Appletviewer mit ein, der verwendet werden muß, um das Java-Applet zu starten, welches die Oracle Forms- oder Reports-Anwendung ausführt. Zusätzlich gibt es bekannte Einschränkungen und Fehler bei der Benutzung des Appletviewers. Lesen Sie in den Hinweisen zu Developer/2000 for the Web weitere Einzelheiten.*

Der Applikations-Server (Solaris und Windows NT werden gegenwärtig unterstützt) enthält diese Komponenten:

- Oracle WebServer 2.1 oder Oracle Web Application Server 3.0
- Forms Server Listener
- Java Development Kit (JDK) 1.1.2 oder höher
- Oracle Formulardateien .FMX und Oracle Berichtsdateien .RDF

Der Forms Listener Server ist für das Lesen der .FMX-Datei, die in der URL angegeben ist, und das Herunterladen der erforderlichen Java-Klassen und des Applets verantwortlich, um die Funktionalität nachzuahmen, welche das Formular in einer Client-Server-Umgebung bietet.

Zuletzt unterstützt der Datenbank-Server die Oracle-Datenbank, auf die von der Formular-Web-Anwendung aus zugegriffen wird. Natürlich kann sich die Oracle Datenbank auch auf dem Applikations-Server befinden.

## Oracle Forms für das Web konfigurieren

Um Oracle-Formulare für das Web zu konfigurieren, müssen Sie drei Arbeiten ausführen:

- Generieren und Einsetzen der Oracle-Formulardateien .FMX auf dem Applikations-Server.
- Der Forms Listener Server muß auf dem Applikations-Server gestartet werden.
- Der WebServer muß so konfiguriert sein, daß die Formularanwendung dem Client-Rechner zur Verfügung steht. Lesen Sie die weiteren Details im Developer/2000 for the Web-Installationsführer.

Lassen Sie uns die ausführlichen Schritte betrachten, die für die Durchführung jeder der Aufgaben notwendig sind.

## FMX-Dateien von Oracle Forms erzeugen und einsetzen

Wenn Sie Developer/2000 for the Web, Version 1.4W installieren, wird Oracle Forms 4.5.8 ebenfalls installiert. Für jede .FMB-Datei, die Sie als Web-Anwendung ausführen wollen, müssen Sie die Datei .FMX mit Hilfe von Oracle Forms 4.5.8 neu erzeugen. Wenn während der Generierung der .FMX-Datei keine Fehler auftreten, platzieren Sie die Datei auf dem Applikations-Server. Sie können die .FMX-Dateien in

jedem Verzeichnis auf dem Applikations-Server unterbringen, solange Sie den Verzeichnispfad und den Dateinamen im Cartridge oder der HTML-Datei festlegen, die der Benutzer referenziert, um die Web-Anwendung auszuführen.

## Aufruf des Forms Server Listeners

Der Forms Listener Server muß gestartet sein, bevor ein Benutzer ein Oracle-Formular als Web-Anwendung starten kann. Es gibt zwei Wege, um den Forms Listener Server unter Windows NT aufzurufen:

- Sie können Start | Programs | Developer/2000 for NT | Forms Listener Server wählen.
- Sie können Ausführen wählen, indem Sie C:\ORANT\bin\f45srv32 port=5555 eingeben und auf OK klicken.

Um festzustellen, ob der Forms Server Listener gestartet ist, bewegen Sie die Maus, bis der Zeiger sich über der freien Fläche der Task-Leiste befindet, und klicken Sie mit der rechten Maustaste, um den Taskmanager aus dem Popup-Menü zu wählen. Wählen Sie die Registerkarte Processes, und blättern Sie durch die Liste der Prozesse. Wenn der Forms Server Listener gestartet ist, werden Sie einen Prozeß namens f45srv32.exe in der Liste sehen.

## Ein Formular aus dem Applet Viewer heraus aufrufen

Um den Appletviewer auf dem Client-Rechner verfügbar zu machen, müssen Sie das JDK 1.1.2 installieren, das sich auf der Developer/2000-for-the-Web-CD-ROM befindet. Standardmäßig wird das JDK 1.1.2 unterhalb des Oracle-Hauptverzeichnisses als JDK1.1 installiert.

Sie sollten außerdem zwei Systemvariablen setzen:

- CLASSPATH, die auf C:\ORACLE\_HOME\jdk1.1\jdk\lib\classes.zip gesetzt sein sollte (wobei *ORACLE\_HOME* dasjenige Verzeichnis ist, in dem die Oracle-Produkte auf dem Applikations-Server installiert werden)
- PATH, welche den Pfad C:\ORACLE\_HOME\jdk1.1\jdk\bin einschließen sollte (wobei *ORACLE\_HOME* dasjenige Verzeichnis ist, in dem die Oracle-Produkte auf dem Applikations-Server installiert werden)

Um den Appletviewer auszuführen, müssen Sie auf dem Client-Rechner ein DOS-Fenster öffnen.

## Entwicklungskriterien hinsichtlich Developer/2000 for the Web

Falls Sie oder Ihr Unternehmen bereits Developer/2000 benutzen, um Client-Server-Anwendungen zu entwickeln, bietet Developer/2000 for the Web einen direkten Weg zum Einsatz Ihrer Oracle-Forms- und Reports-Anwendungen als Web-Anwendungen (war Gegenstand der Vorbehalte, die im letzten Hinweis diskutiert wurden). Auf der positiven Seite ermöglicht Ihnen Developer/2000 for the Web, die Last der Programmierung eines Java-Applets zu vermeiden, das eine äquivalente Funktionalität wie Ihre Forms- oder Reports-Anwendungen besitzt. Auf der anderen Seite haben Sie mit Developer/2000 for the Web nur wenig Kontrolle über den Java-Code, der erzeugt wird; es kann ziemlich schwierig werden, eine Forms-Anwendung zusammen mit anderen Java-Applets zu integrieren. Falls Sie jedoch zum gegenwärtigen Zeitpunkt Developer/2000 nicht benutzen und dies auch nicht planen, stehen Ihnen andere Alternativen zur Verfügung:

- Entwickeln Sie PL/SQL-Unterprogramme unter Zuhilfenahme der PL/SQL-Pakete, die mit dem Web Application Server geliefert werden, um HTML dynamisch zu generieren.
- Entwickeln Sie ein Java-Applet mit Hilfe einer Java-Entwicklungsumgebung wie z.B. Symantec Cafe.
- Benutzen Sie Software-Pakete wie Cold Fusion oder Tango von Drittanbietern, die Ihnen den Zugriff auf eine Oracle-Datenbank mittels ODBC ermöglichen.

Lassen Sie uns einen Blick auf die Vor- und Nachteile dieser Alternativen werfen.

## Entwickeln von Web-Anwendungen mit PL/SQL

Oracles Web Application Server umfaßt mehrere Pakete, einschließlich http und htf, welche Sie zur Erstellung von Web-Anwendungen benutzen können. Es gibt verschiedene Vorteile für diesen Ansatz:

- Weil Sie PL/SQL zum Schreiben Ihrer Anwendung benutzen, haben Sie Zugriff auf alle integrierten Pakete und die weiteren Möglichkeiten der Oracle-Datenbank.
- Indem Sie für die Erstellung jedes Web-Formulars selbst verantwortlich sind, haben Sie auch die Kontrolle über das Erscheinungsbild und das Verhalten Ihrer Web-Anwendung.
- Durch die Anwendung der Web Application Server 3.0 PL/SQL Cartridge können Sie sicherstellen, daß Transaktionen korrekt rückgängig gemacht oder bestätigt werden.

Es gibt auch einige Nachteile beim Schreiben Ihrer Web-Anwendung in PL/SQL:

- Anders als bei einem Java-Applet ist ein PL/SQL-Unterprogramm, das HTML dynamisch erzeugt, auf die in HTML zur Verfügung stehende Benutzerschnittstelle begrenzt. Mit anderen Worten, eine Benutzereingabe wird nicht auf Ihre Gültigkeit geprüft, bis der Benutzer das Formular durch Betätigen einer Befehlsschaltfläche bestätigt.
- PL/SQL bietet nicht die objektorientierten Merkmale, die in einer Programmiersprache wie Java existieren. Dies bleibt auch nach den in Oracle8 Server und PL/SQL 3.0 eingeführten objektrelationalen Merkmalen eine Tatsache.
- Mit Ausnahme des Designer/2000 gibt es keine Entwicklungsumgebungen, die Ihnen das grafische Erstellen eines Web-Formulars und das Erzeugen des PL/SQL-Codes zu seiner Realisierung erlauben. Und Designer/2000 ist auf die Generierung von Web-Formularen begrenzt, die einzig eine Oracle-Datenbank abfragen können.

## Entwickeln von Web-Anwendungen mit Java

Es gibt verschiedene Java-Entwicklungsumgebungen, so wie Java Cafe, die für die Entwicklung eines Java-Applets benutzt werden können, das von einem Web-Browser aufgerufen werden kann. Diese Produkte verwenden typischerweise einen JDBC-Treiber - ähnlich einem ODBC-Treiber -, um auf eine relationale Datenbank zugreifen zu können. Dieser Entwicklungsansatz besitzt einige Vorteile:

- Sie haben die Kontrolle über das Aussehen und die Handhabung des Applet-Benutzerinterface.
- Indem Sie eine Gültigkeitsprüfung auf Ebene der Felder einfügen, wird das Benutzerinterface für den Benutzer weniger frustrierend. Das Applet kann den Benutzer davor bewahren, zum nächsten Feld überzugehen, wenn das aktuelle Feld einen ungültigen Wert enthält.
- Das Applet kann unterschiedliche Entwicklungsumgebungen unterstützen. Mit einem JDBC-Treiber kann das Applet gleichzeitig mit verschiedenen Datenbanken anderer Hersteller Verbindung aufnehmen und einen geschlossenen Überblick auf zusammenhängende Informationen präsentieren.

Lassen Sie uns auch die negativen Aspekte dieses Entwicklungsansatzes überdenken:

- Durch die Benutzung eines JDBC-Treibers ist das Applet nicht so gut in die Oracle-Datenbank integriert wie Developer/2000 for the Web oder PL/SQL. Sie werden nicht in der Lage sein, so viele der Merkmale zu benutzen, wie durch die Oracle-Werkzeuge zur Verfügung gestellt werden.
- Weil Sie die Verantwortung für die Benutzerschnittstelle haben, müssen Sie eine beachtliche Menge an Java-Code schreiben.

## Entwickeln von Web-Anwendungen mit anderen Software-Paketen

Es gibt verschiedene Software-Pakete, so wie Cold Fusion und Tango, die ein grafisches Interface für die Entwicklung von Web-Anwendungen bieten. Im allgemeinen benötigen diese Produkte die Installation eines Applikations-Servers auf einem Sun-Solaris- oder Windows-NT-Server. Der Applikations-Server greift auf eine Oracle-Datenbank mittels eines ODBC-Treibers zu. Es gibt einige klare Vorteile dieser Pakete:

- Sie sind relativ leicht zu benutzen. Ein ungeübter Entwickler kann eine Anwendung innerhalb von ein paar Tagen schreiben.
- Weil der generierte HTML-Code Aufrufe an den Applikations-Server einschließt, kann die resultierende Anwendung Gültigkeitsprüfungen auf Feld-Ebene enthalten.

Es gibt ebenfalls einige Aspekte, deren Sie sich bewußt sein sollten:

- Weil diese Pakete typischerweise einen ODBC-Treiber verwenden, um auf eine Oracle-Datenbank zuzugreifen, werden Sie nicht in der Lage sein, viele der zur Verfügung stehenden Merkmale des Oracle-Servers zu benutzen.
- Abhängig von dem Produkt, das Sie bevorzugen, wird die sich ergebende Web-Anwendung nicht in der Lage sein, zwei Benutzer von der Änderung desselben Datensatzes zur gleichen Zeit abzuhalten. Wenn Sie eine Anwendung erstellen, die nur Abfragen enthält, ist das nicht von Bedeutung. Falls Sie jedoch eine Web-Anwendung entwickeln, die es einem Benutzer ermöglichen soll, Datensätze hinzuzufügen, zu ändern und zu löschen, müssen Sie die Möglichkeiten untersuchen, die das Software-Paket für das



Sperren von Datensätzen bietet.

## Die Network Computing Architecture von Oracle

Im September 1996 führte die Oracle Corporation eine Strategie für zukünftige Software-Entwicklung ein: die Network Computing Architecture (NCA). Die Schlüsselfaktoren, welche die Erzeugung von NCA beeinflussten, waren folgende:

- explosionsartiges Wachstum des Internet und des World Wide Web
- wachsende Komplexität und Verteuerung bei der Verwaltung von Client-Server-Anwendungen auf einer Reihe verschiedener Plattformen, so wie PCs, Macs und Unix-Workstations
- Unterstützung von Objekten
- Unterstützung von mehreren Programmiersprachen
- Unterstützung für verteilte Anwendungen und Datenbank-Server
- fortgesetzte Unterstützung für bereits bestehende Anwendungen (z.B. Client-Server)



*Verwechseln Sie die NCA nicht mit der Entwicklung eines Network Computers durch die Network Computer, Inc., einer Tochtergesellschaft der Oracle Corporation. Die Network Computing Architecture ist eine Software-Architektur, während der Network Computer eine Hardware/Software-Plattform ist, die in der NCA eine Rolle spielt.*

---

## Überblick über die Network Computing Architecture

Es ist wichtig zu verstehen, daß die NCA eine Strategie für kommerzielle Software-Entwicklung ist, die im Laufe der Zeit von bestimmten Oracle-Produkten eingeführt wird. Nach einer Aussage der Oracle Corporation schließen die Ziele der Strategie folgendes ein:

- **Skalierbarkeit:** Ein Hauptziel der NCA ist das Auflösen vieler der Beschränkungen, die mit Client-Server-Anwendungen im Zusammenhang stehen, wie z.B. Performance, Administration und Einführung. Mit der Einführung von Web-basierten Anwendungen reduziert die NCA die Administration der Client-Rechner; ein Netzwerk-Administrator muß sich keine Sorgen mehr über die Logistik der Installation der aktuellen Version eines ausführbaren Programms und der zugehörigen Bibliotheken auf Hunderten von PCs machen.
- **Portierbarkeit:** Ein Ziel der NCA ist die Unterstützung von unternehmensweiter Software-Entwicklung, unabhängig von Programmiersprachen, Rechnerplattformen oder Betriebssystemen. Die Benutzung von Java ist von zentraler Bedeutung für dieses Ziel. Java bietet Entwicklern die Möglichkeit, Software einmal zu schreiben und sie dann auf verschiedenen Rechnerumgebungen einsetzen zu können.
- **Interoperabilität:** Die NCA nimmt an, daß der Kunde sich die Flexibilität wünscht, unabhängig vom Hersteller verschiedene Komponenten wählen und diese integrieren zu können; ein Kunde könnte beispielsweise den Oracle Datenbank-Server, einen Netscape-WebServer und eine Kombination von Microsoft-Windows-95- und Mac-Clients verwenden wollen. Wenn man zwei Standards einhält - CORBA 2.0 und HTTP/HTML - unterstützt NCA verschiedenartige Umgebungen.
- **Unterstützung von komponentenbasierter Software-Entwicklung:** Die Oracle Corporation hat angekündigt, daß die NCA die CORBA 2.0-Spezifikation einhalten wird, welche die Benutzung von verteilten Objekten definiert, unabhängig von ihrer Platzierung im Netzwerk - Client, Applikations-Server oder Datenbank-Server. Diese Unterstützung wird in künftige Versionen von Oracle-Produkten integriert werden.

Lassen Sie uns einige der von der CORBA-2.0-Spezifikation gebotenen Vorteile untersuchen.

## CORBA

CORBA ist die Abkürzung für Common Object Request Broker Architecture, wobei es sich um eine Architektur und Spezifikation der Object Management Group handelt, einem Industrie-Konsortium, das 1989 gegründet wurde. Die aktuelle Version von CORBA ist 2.0, die im Juli 1995 erstmalig ausgegeben und im Juli 1996 verbessert wurde. Das Hauptziel von CORBA ist die Förderung der Benutzung und Integration von Objekten, unabhängig von der Programmiersprache, die zur Erstellung der Objekte benutzt wurde, und vom Betriebssystem,

in dem die Objekte abgelegt wurden. CORBA erreicht dieses Ziel durch die Verwendung eines Object Request Brokers (ORB), wobei es sich um eine Software handelt, die Anfragen für den Service von Objekten bearbeitet. Sie können sich ein ORB als einen Objekt-Bus vorstellen. Es gibt eine Reihe von ORB-Herstellern, einschließlich Iona, Visigenic, Sun, HP und IBM.

Ein anderer wichtiger Teil von CORBA 2.0 ist die Spezifikation der Interface Definition Language (IDL), die eine sprachenunabhängige Beschreibung des Service bietet, den ein Objekt zur Verfügung stellt. Die ORB-Hersteller bieten Software an, die sprachenspezifische Bindings von IDL aus herstellt. Die Absicht von CORBA 2.0 ist es, die externe Spezifikation eines Objekts zu seiner IDL zu reduzieren. Ein Entwickler muß nichts über die Programmiersprache wissen, mit der es geschrieben wurde, oder über das Betriebssystem, unter dem es entwickelt wurde; alles, was ein Entwickler über ein Objekt wissen muß, ist in dessen IDL-Spezifikation enthalten. IDL ist eine Untermenge des vorgeschlagenen C++-ANSI-Standards mit Erweiterungen für die Einführung von Objekt-Services.

CORBA 2.0 schließt außerdem eine Spezifikation für ein Internet Inter-ORB Protocol (IIOP) mit ein. Dieses Protokoll legt die Kommunikationsmechanismen mittels TCP/IP fest, die einem ORB die Kommunikation mit einem anderen ORB ermöglicht. Im wesentlichen ist IIOP ein Mechanismus, mittels dem ein Client auf einer Rechnerplattform eine Anfrage für einen Service von Objekt abc starten kann, die von einem ORB bearbeitet wird. Indem es als Client auftritt, startet Objekt abc eine mittels TCP/IP gesendete Anfrage für einen Service an Objekt xyz, das von einem ORB auf einer anderen Rechnerplattform verwaltet wird.

### Komponenten der Network Computing Architecture

Weil ein Bild mehr sagt als tausend Worte, schauen Sie bitte auf Abbildung 21.11 mit einem Diagramm der Network Computing Architecture. Wie Sie sehen können, gibt es folgende Komponenten in der NCA:

- den Oracle Universal Server,
- den Oracle Universal Application Server,
- Cartridges, die als Interface für den Client, den Applikations- oder den Universal-Server dienen können,
- den Inter-Cartridge Exchange, welcher Cartridges die Kommunikation untereinander ermöglicht,
- ein Client kann ein PC, ein Mac oder ein Netzwerk-Computer sein.

### Cartridges

Wie von der NCA vorgeschlagen, ist eine Cartridge eine Software-Komponente, die als Interface zwischen dem Client, dem Applikations-Server oder der Datenbank selbst dient. Der Zweck einer Cartridge ist es, eine spezielle Fähigkeit für andere Komponenten und Cartridges in der NCA verfügbar zu machen. Eine Organisation könnte beispielsweise eine Cartridge entwickeln, um die Berechnung einiger externer Daten, wie z.B. von Rohstoffpreisen, zu ermöglichen. Eine Cartridge kann den Service benutzen, den die NCA bietet, wie die Installation, die Registration, die Instanzenbildung, die Administration und die Sicherheit. Sie können eine Cartridge in vielen verschiedenen Sprachen schreiben, wie z.B. Java, C++ oder PL/SQL.

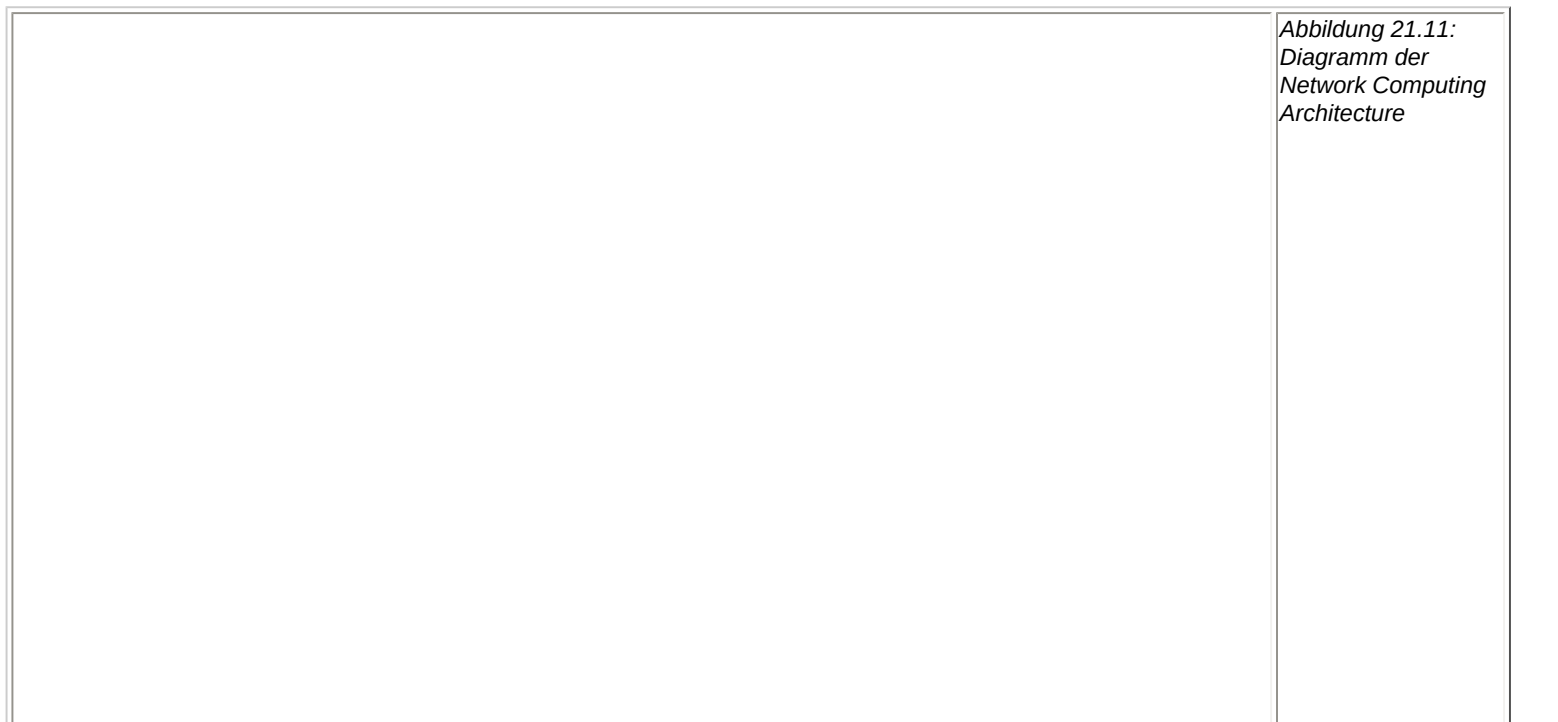
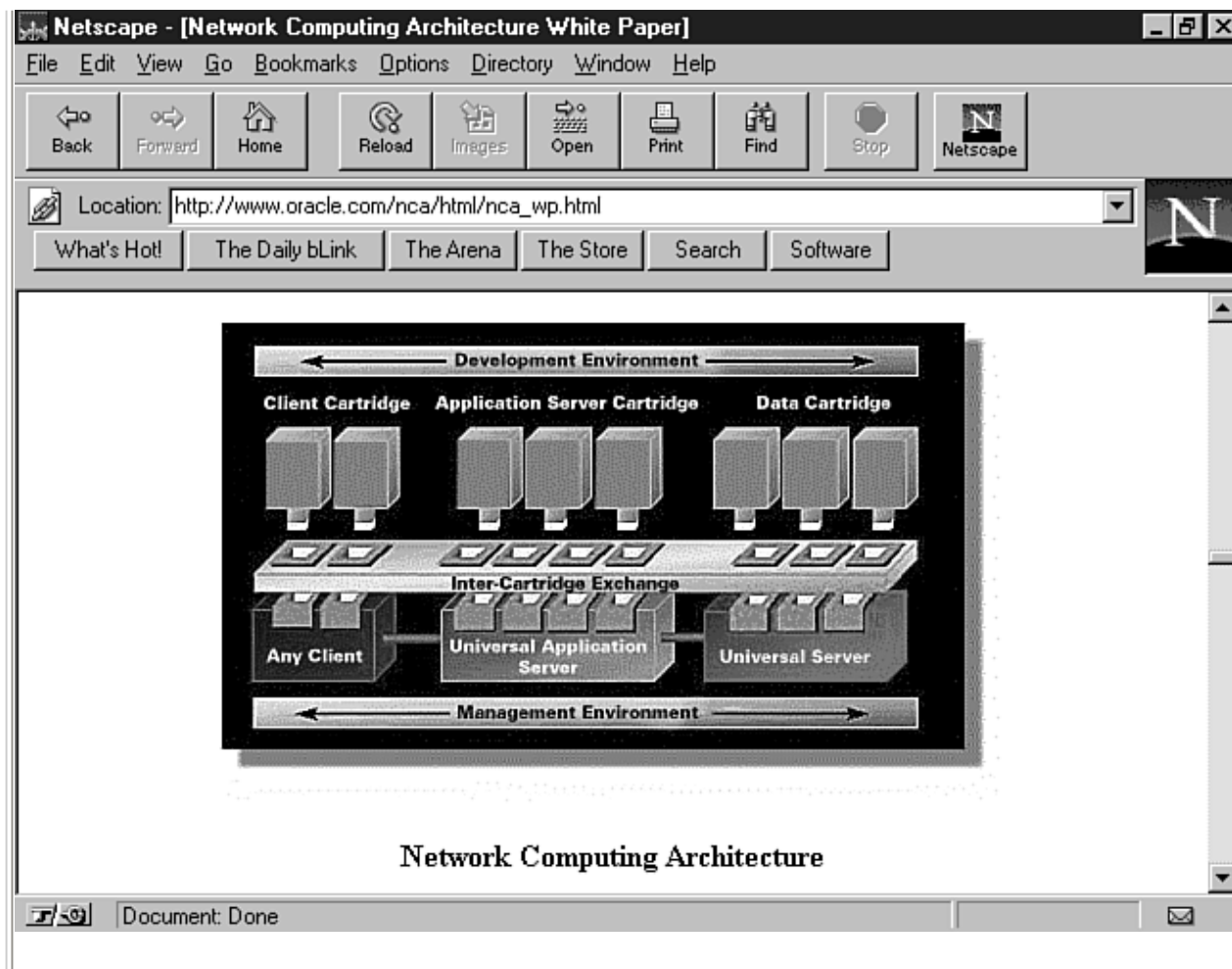


Abbildung 21.11:  
Diagramm der  
Network Computing  
Architecture



## Inter-Cartridge Exchange

Die NCA schließt ein Element mit der Bezeichnung Inter-Cartridge Exchange (ICX) ein, das als eine Clearing-Stelle für den von Cartridges zur Verfügung gestellten Service funktioniert, wo immer sich diese im Netzwerk befinden - auf dem Client, dem Applikations-Server oder dem Datenbank-Server. Zusätzlich zu seiner Funktion als ORB verbindet das ICX NCA mit dem Microsoft Object Model (COM), so daß ActiveX-Clients mit den NCA Cartridges kommunizieren können. Das ICX bietet außerdem die Verbindung zu Java-Komponenten und bestehenden System-Interfaces. Wenn Sie den Oracle-Server in der Version 7.3.3 verwenden, können Sie mit Hilfe der PL/SQL-Cartridge mit dem ICX kommunizieren.

## Zusammenfassung

Die Schlüsselkonzepte dieser abschließenden Lektion, die Sie sich merken sollten, waren folgende:

- Der Oracle Web Server kann verwendet werden, um den Zugriff auf eine Oracle-Datenbank mittels eines Web-Browsers zu ermöglichen.
- Die Komponenten des Web Application Server sind der Web Listener, der Database Access Descriptor, der Web Request Broker und Cartridges für verschiedene Sprachen.
- Developer/2000 for the Web erlaubt einer Oracle-Forms- oder Reports-Anwendung den Einsatz als Client-Server- oder Web-Anwendung.
- Developer/2000 for the Web benutzt einen Formular-Server, der die Inhalte von .FMX- oder .RDF-Dateien in Aufrufe von verschiedenen Java-Komponenten übersetzt.
- Die Network Computing Architecture ist Oracles Strategie für die zukünftige kommerzielle Software-Entwicklung. Sie betont die Unterstützung offener Standards, so wie CORBA 2.0 und HTTP/HTML.

## Fragen und Antworten

Frage:

Kann der Web Application Server für den Zugriff auf eine Nicht-Oracle-Datenbank benutzt werden?

Antwort:

*Ja. Dies kann durch die Verwendung eines Gateways erreicht werden. Oracle bietet Gateway-Produkte für verbreitete Datenbanken, wie z.B. DB2 und Sybase. Eine andere Alternative ist die Benutzung der ODBC-Cartridge, um auf eine Nicht-Oracle-Datenbank zugreifen zu können, unter der Voraussetzung, daß für diese Datenbank ein ODBC-Treiber vorhanden ist.*

## Workshop

Der Zweck dieses Workshop ist es, Ihnen die Möglichkeit zu geben, Ihr Wissen über das in dieser Lektion behandelte Thema zu testen. Schauen Sie, ob Sie die Fragen des Tests richtig beantworten können, und machen Sie die Übungen, bevor Sie mit der morgigen Lektion fortfahren.

## Test

1. Welche Aufgabe haben die http- und http-PL/SQL-Pakete?
2. Richtig oder falsch? Die Network Computing Architecture steht nur für Sun Solaris und Microsoft Windows NT zur Verfügung.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖ Kapitel ❖ Top Inhalt Index **SAMS** Kapitel ❖

# 22

## Bonustag: Power-Builder für eine Oracle-Datenbank verwenden

[Definieren eines Datenbankprofils](#)

[Ein DataWindow erstellen](#)

[Fehlerbehandlung mit Oracle](#)

[Zusammenfassung](#)

Dieser Bonustag behandelt die Benutzung von PowerBuilder und Oracle. PowerBuilder ist eine Entwicklungsumgebung für Anwendungen, die mit Oracle Power Objects verglichen werden kann. Es ist das Hauptprodukt von Powersoft, einer Tochtergesellschaft im vollständigen Besitz von Sybase.

Die Merkmale von PowerBuilder sind folgende:

- Das Datenfenster, ein flexibler und intuitiver Mechanismus für das Abbilden einer Datenbanktabelle in ein Objekt, das in einer Anwendung manipuliert werden kann
- Unterstützung für die meisten Desktop- und Server-Datenbanken, einschließlich Oracle, Sybase, SQL Server, Informix und SQL Anywhere
- Exzellente Integration in Windows mittels DDE und OLE
- Unterstützung für verteilte Objekte
- Ein Pipeline-Objekt, das die Datenübertragung zwischen verschiedenen Datenbank-Plattformen unterstützt (zum Beispiel von Sybase nach Oracle)
- Unterstützung für Klassen, Vererbung und Polymorphie



*Die Beispiele in dieser Lektion basieren auf der PowerBuilder Enterprise Edition. Um mehr über PowerBuilder zu erfahren, besuchen Sie die PowerSoft-Website unter [www.powersoft.com](http://www.powersoft.com).*

Wie Developer/2000 und Power Objects benutzt auch PowerBuilder eine Skriptsprache namens PowerScript für eine ereignisgesteuerte Programmierung. Jedes Objekt besitzt einen Satz von vordefinierten Ereignissen, für die Sie ein Skript schreiben können. Dieses Skript wird ausgeführt, wenn das Ereignis durch ein externes Ereignis - ein Benutzer betätigt zum Beispiel die Schaltfläche Entf - oder durch das Skript eines anderen Objekts ausgelöst wird. Zusätzlich dazu können Sie weitere Ereignisse für ein Objekt definieren und damit eine aus dem umfassenden Satz der in PowerBuilder integrierten Funktionen von einem ereignisgesteuerten Skript aus aufrufen.

Heute werden Sie drei bestimmte Bereiche betrachten:

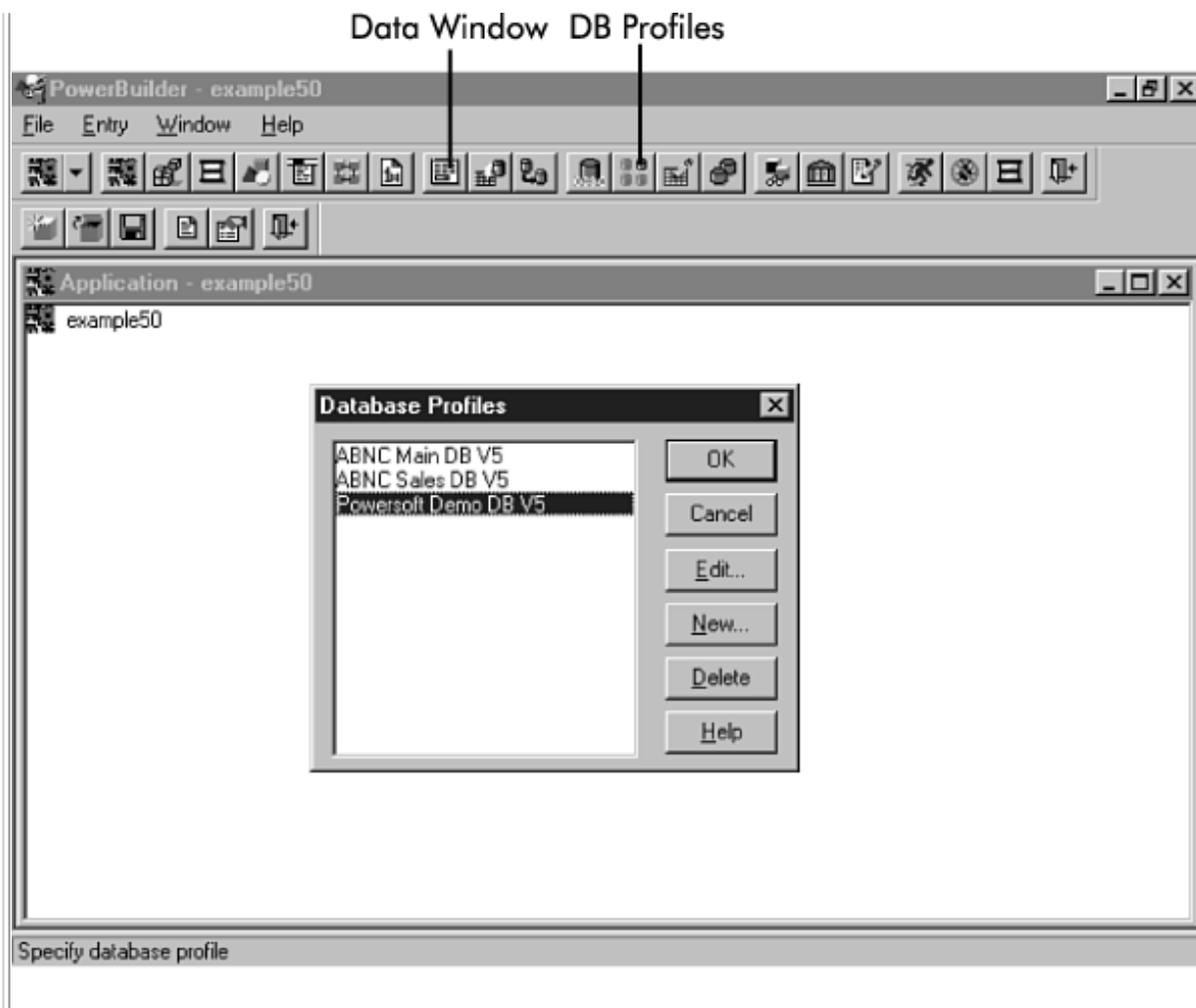
- Definieren eines Datenbankprofils für eine Oracle-Datenbank
- Erstellen eines Datenfensters für eine Oracle-Tabelle oder -Join
- Verarbeiten von Oracle-Fehlern

## Definieren eines Datenbankprofils

Die erste Phase bei der Benutzung von PowerBuilder mit einer Oracle-Datenbank ist das Aufstellen eines Datenbankprofils.

1. Öffnen Sie die PowerBuilder-Anwendung, der Sie das Datenbankprofil hinzufügen wollen.
2. Dazu klicken Sie auf das Symbol DB Profile auf der Symbolleiste (sieht aus wie vier grüne Zylinder). Ein Fenster erscheint, das die bereits existierenden Datenbankprofile anzeigt (siehe Abbildung 22.1).
3. Klicken Sie auf New, um ein neues Datenbankprofil zu erstellen. Das Fenster Database Profile Setup wird erscheinen.
4. Geben Sie Flugle in das Feld Profile Name ein (siehe Abbildung 22.2).
5. Wählen Sie im DBMS-Feld die aktuelle Version Ihrer Datenbank aus. Dieses Beispiel zeigt O72 ORACLE v7.2 aus der Dropdown-Liste (siehe Abbildung 22.2).
6. Klicken Sie auf More, um die anderen Felder für das Oracle-Datenbankprofil festzulegen.

Abbildung 22.1:



Das Dialogfeld Database Profiles zeigt vorhandene Datenbankprofile an



Falls Sie die aktuelle Version nicht in der Dropdown-Liste sehen, müssen Sie einige Komponenten neu installieren. Während der Installation der PowerBuilder Enterprise Edition können Sie zwischen den Optionen Standard und benutzerdefinierte Installation wählen. Die Standardoption wird die aktuelle Version des originalen Datenbanktreibers nicht installieren. Falls Sie Windows 95 oder Windows NT benutzen, müssen Sie SQL\*Net V2.2 oder höher verwenden, und Sie müssen den originalen Oracle-7.x- oder einen neueren Datenbanktreiber von PowerBuilder installieren.

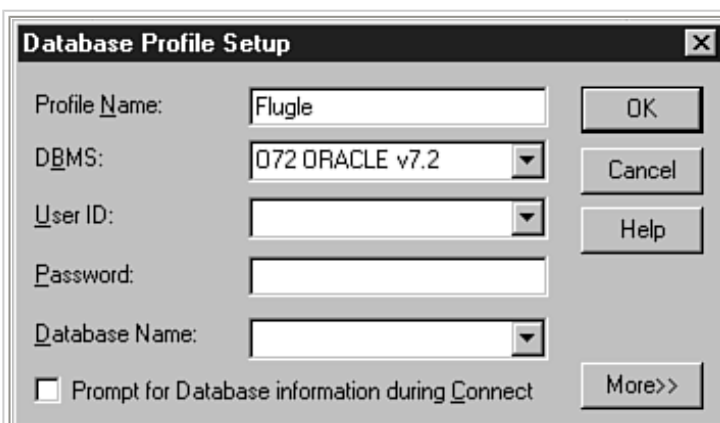
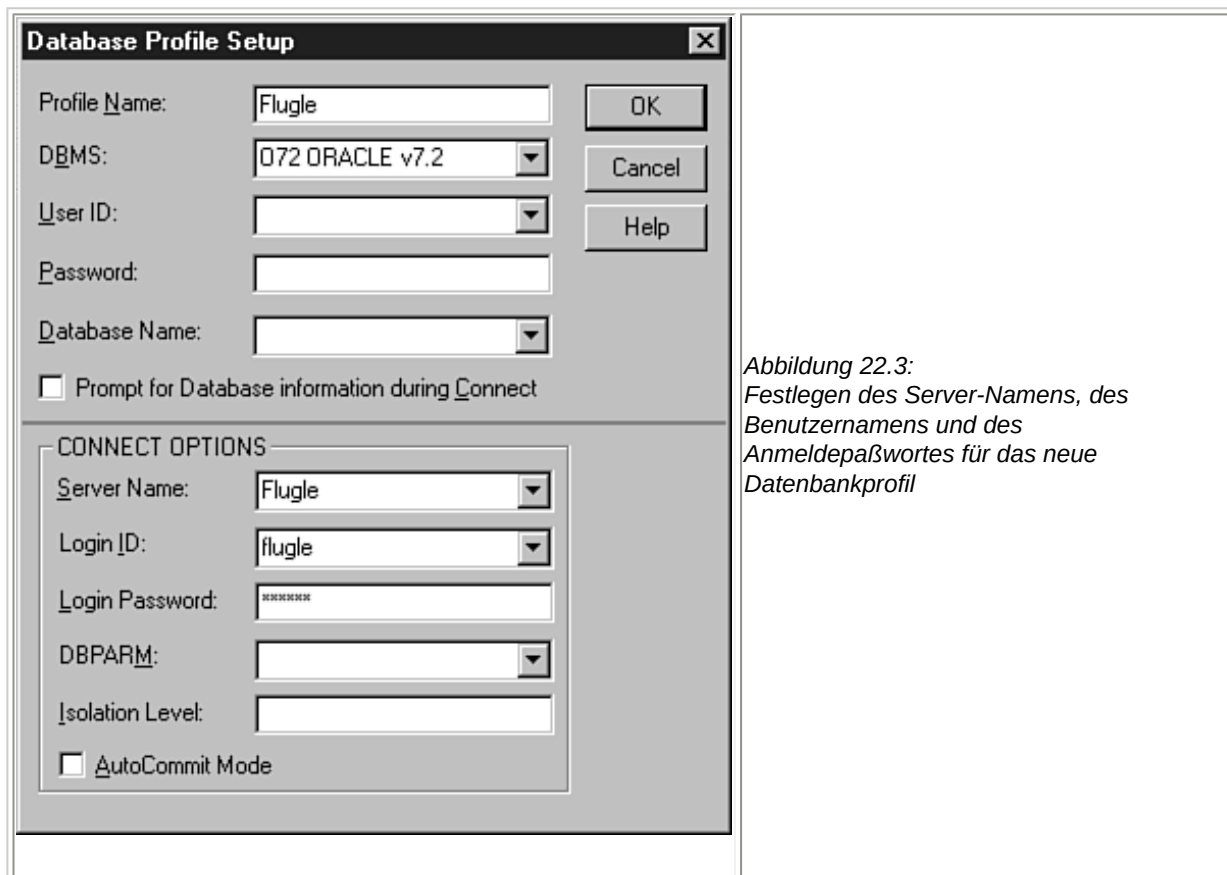


Abbildung 22.2:  
Festlegen des Server-Namens, des Benutzernamens und des Anmeldepaßwortes für das neue Datenbankprofil

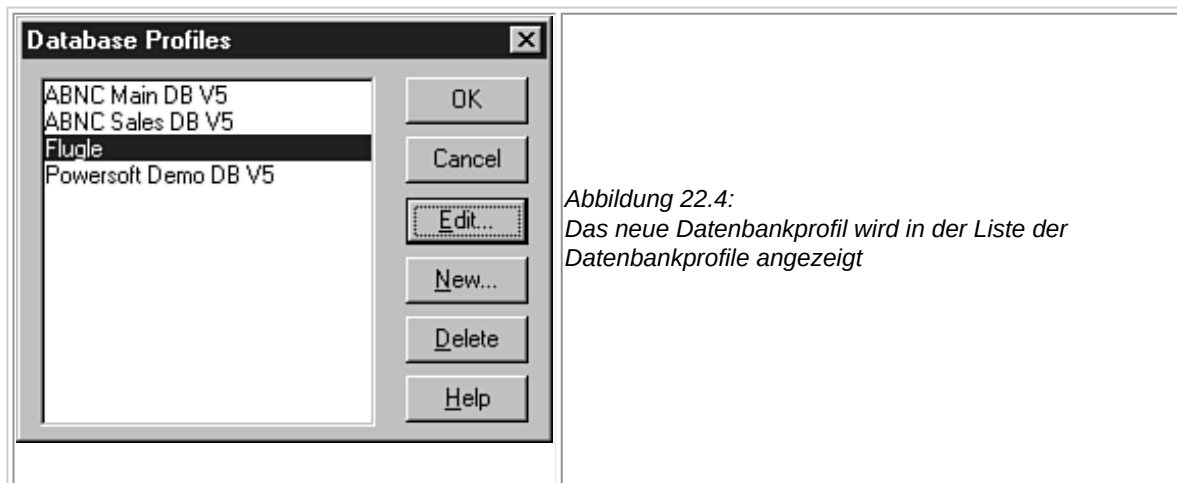
7. Wenn Sie die Verbindung zu einem Oracle-Server aufbauen, geben Sie den Datenbank-Alias, den Sie verwenden, in das Feld Server Name ein. Wenn Sie Personal Oracle verwenden, lassen Sie dieses Feld leer.

8. Geben Sie in Login ID den Oracle-Benutzernamen ein, den Sie für den Zugriff auf die Datenbank verwenden wollen.

9. Geben Sie das Kennwort des Oracle-Benutzers in Login Password ein, wie in Abbildung 22.3 gezeigt wird.



10. Klicken Sie auf OK. PowerBuilder wird das Datenbankprofil erzeugen und in der Liste der zur Verfügung stehenden Profile anzeigen (siehe Abbildung 22.4).



11. Nachdem Sie das neue Datenbankprofil erstellt haben, werden Sie es in der Liste der bereits existierenden Datenbankprofile sehen. Um es zu benutzen, wählen Sie es aus und klicken auf OK.



Wenn Sie erstmalig ein Datenbankprofil für eine Oracle-Datenbank erstellen, wird PowerBuilder versuchen, ein Repository zu erzeugen - einen Satz von fünf Tabellen, die benutzt werden, um zusätzliche Informationen über jede Tabellenspalte zu speichern, so wie das Format oder eine Gültigkeitsprüfung. Die Namen der Tabellen sind PBCATCOL, PBCATEDT, PBVCATFMT, PBCATTBL und PBCATVLD.



PowerBuilder versucht standardmäßig, diese Tabellen im Oracle SYSTEM-Account zu erstellen. Normalerweise wird PowerBuilder einen Oracle-Fehler ORA-01031, insufficient privileges zurückgeben, wenn der in Login ID festgelegte Oracle-Benutzer nicht die Berechtigung besitzt, eine Tabelle im SYSTEM-Account zu erzeugen.

Es gibt verschiedene Wege, die Repository zu erstellen, wenn dies noch nicht der Fall ist. Sie können das neue Datenbankprofil erzeugen und dabei für die Login ID und das Kennwort SYSTEM festlegen. Von SQL\*Plus oder SQL Worksheet aus können Sie die Berechtigungen SELECT, INSERT, UPDATE und DELETE für die fünf Repository-Tabellen dem Oracle-Benutzer zuteilen, den Sie in der Login ID bestimmt haben.

## Ein DataWindow erstellen

Um ein neues Datenfenster zu erstellen, folgen Sie diesen Schritten:

1. Klicken Sie die DataWindow-Schaltfläche auf der Symbolleiste an (sieht aus wie ein Formular und ist das neunte Symbol von links). Ein Fenster mit dem Titel Select DataWindow erscheint und zeigt die für die Anwendung existierenden Datenfenster (siehe Abbildung 22.5).
2. Klicken Sie auf New, um ein neues Datenfenster zu erzeugen. Daraufhin werden Sie ein Fenster New DataWindow sehen.
3. Sie müssen für das neue Datenfenster zweimal eine Auswahl treffen - die Datenquelle und den Präsentationsstil (siehe Abbildung 22.6).
4. Klicken Sie auf OK.

Ein anderes Fenster mit dem Titel Select Tables wird erscheinen. Dieses Fenster zeigt standardmäßig alle Tabellen an, für die der Oracle-Benutzer, der mit dem gegenwärtigen Datenbankprofil verknüpft ist, die Zugriffsrechte besitzt. Blättern Sie herunter bis zur STUDENT-Tabelle, und klicken Sie auf Open (siehe Abbildung 22.7).

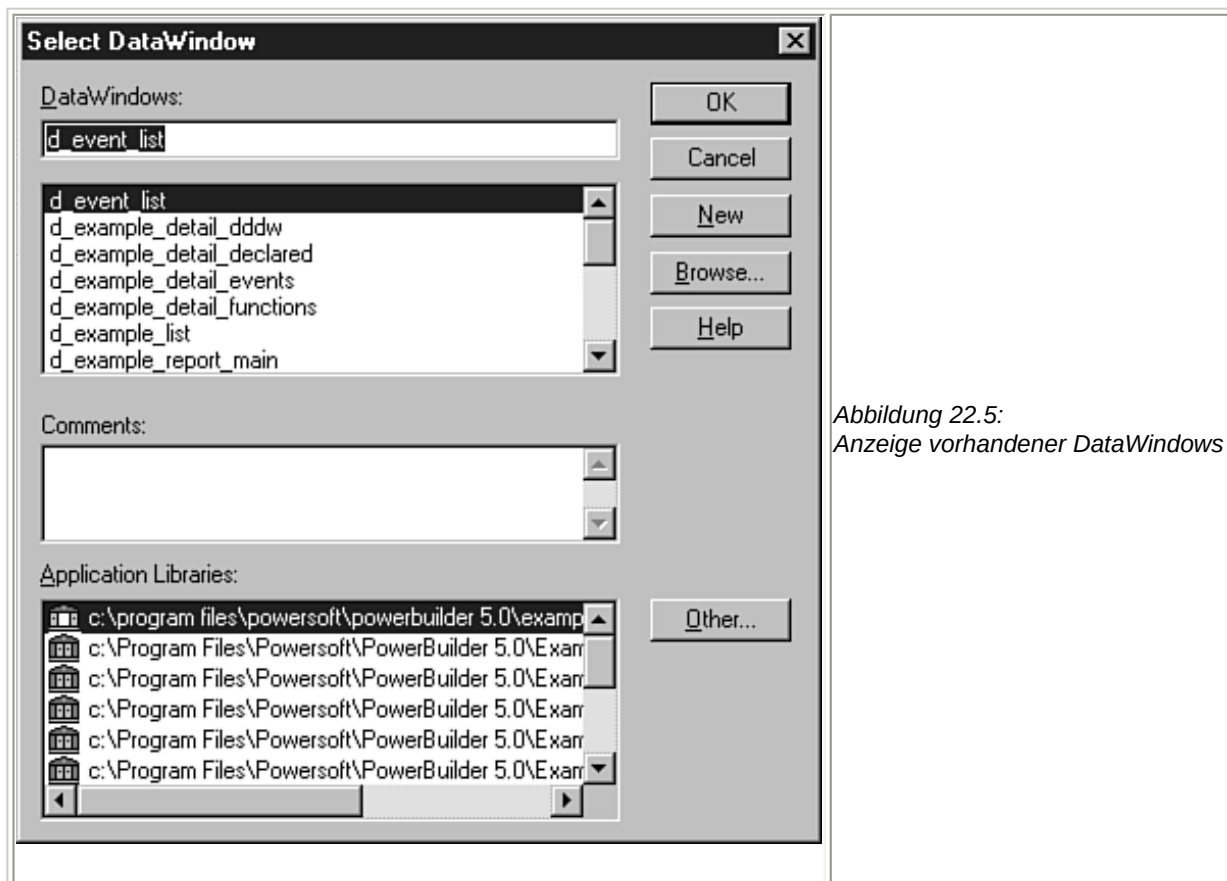




Abbildung 22.6:  
Festlegen der  
Datenquelle und der  
Darstellungsart für  
das neue  
DataWindow

Sie benutzen jetzt den DataWindow Painter. PowerBuilder bietet einen »Painter« für jeden Objekttyp. Im DataWindow Painter sehen Sie jede Spalte der Student-Tabelle zusammen mit ihrem Datentyp. Wählen Sie mit Hilfe der Maus jede Spalte aus, die Sie in das Datenfenster einfügen wollen. Wenn Sie jede gewünschte Spalte gewählt haben, sehen Sie diese Spalten in der Selection List nahe der oberen Kante des Datenfensters aufgereiht. Wählen Sie alle Spalten, wie in Abbildung 22.8 gezeigt wird.

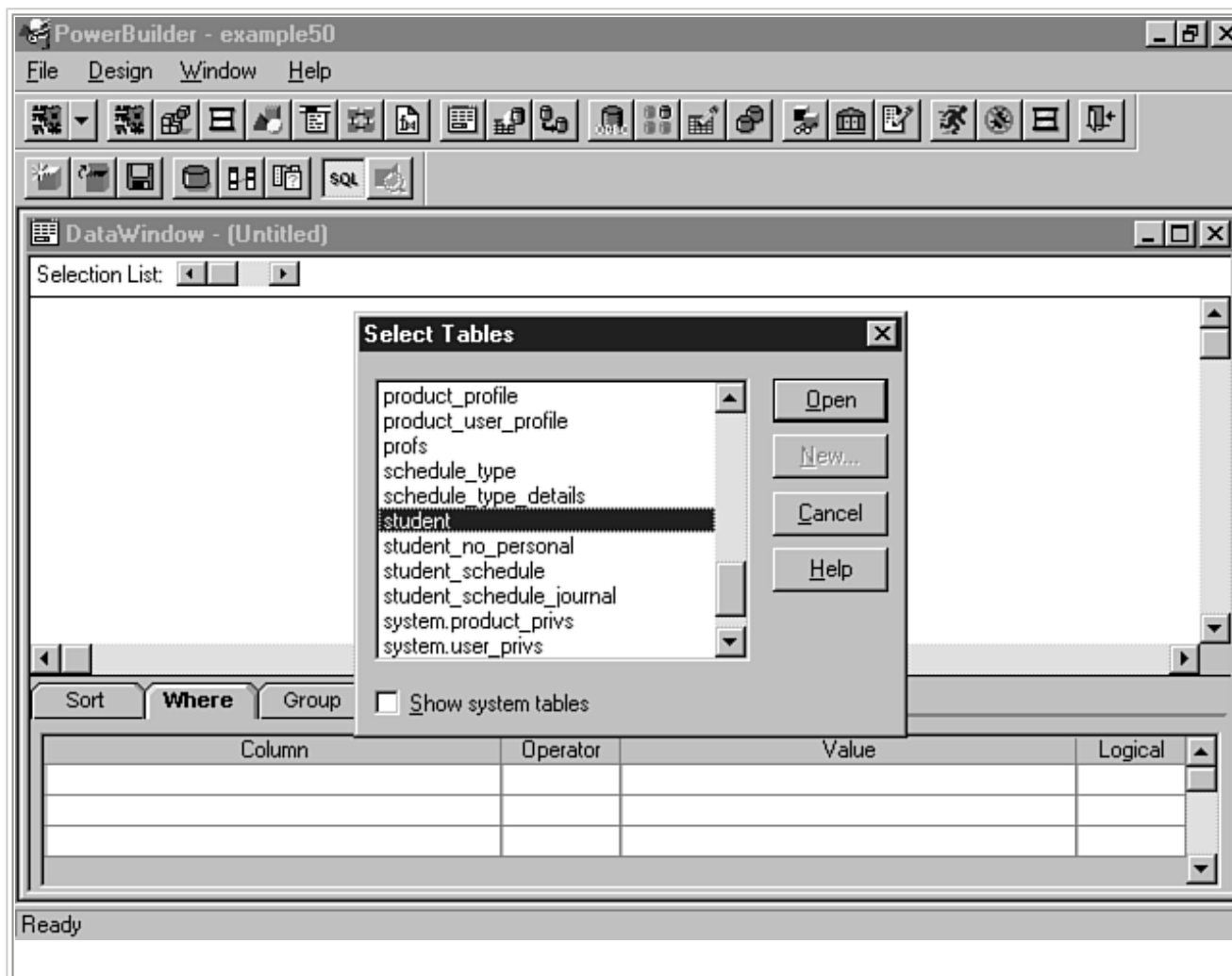


Abbildung 22.7:  
Anzeige einer Liste  
verfügbarer  
Tabellen

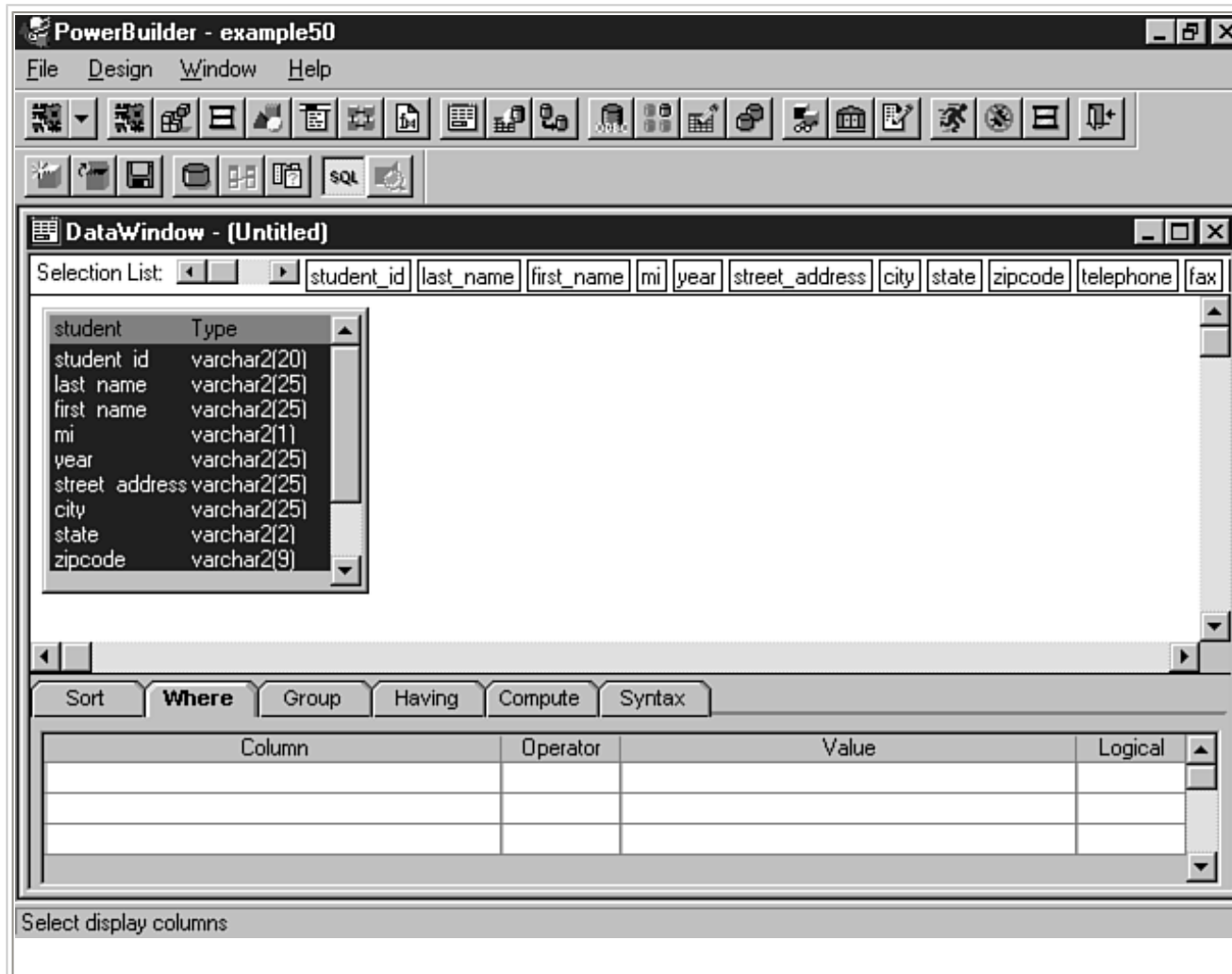


Abbildung 22.8:  
Auswahl der  
Spalten, die im  
neuen  
DataWindow  
enthalten sein  
sollen

Als nächstes sollten Sie die Reihenfolge festlegen, in der Sie die Datensätze im Datenfenster angezeigt sehen wollen. Nahe dem unteren Rand des DataWindow Painter sollten Sie sechs Karteikartenreiter sehen: Sort, Where, Group, Having, Compute und Syntax.

1. Wählen sie die Karteikarte Sort aus. Die Liste auf der linken Seite der Karte enthält die Spalten der Student-Tabelle.
2. Ziehen Sie jede Spalte in das Fenster auf der rechten Seite, und zwar in der Reihenfolge, in der die Datensätze sortiert sein sollen. Als Standard wird jede Spalte in aufsteigender Folge sortiert.
3. Wählen Sie Student\_ID in der Liste links, und ziehen Sie sie in die Liste rechts. Abbildung 22.9 veranschaulicht, was Sie im DataWindow Painter sehen sollten.

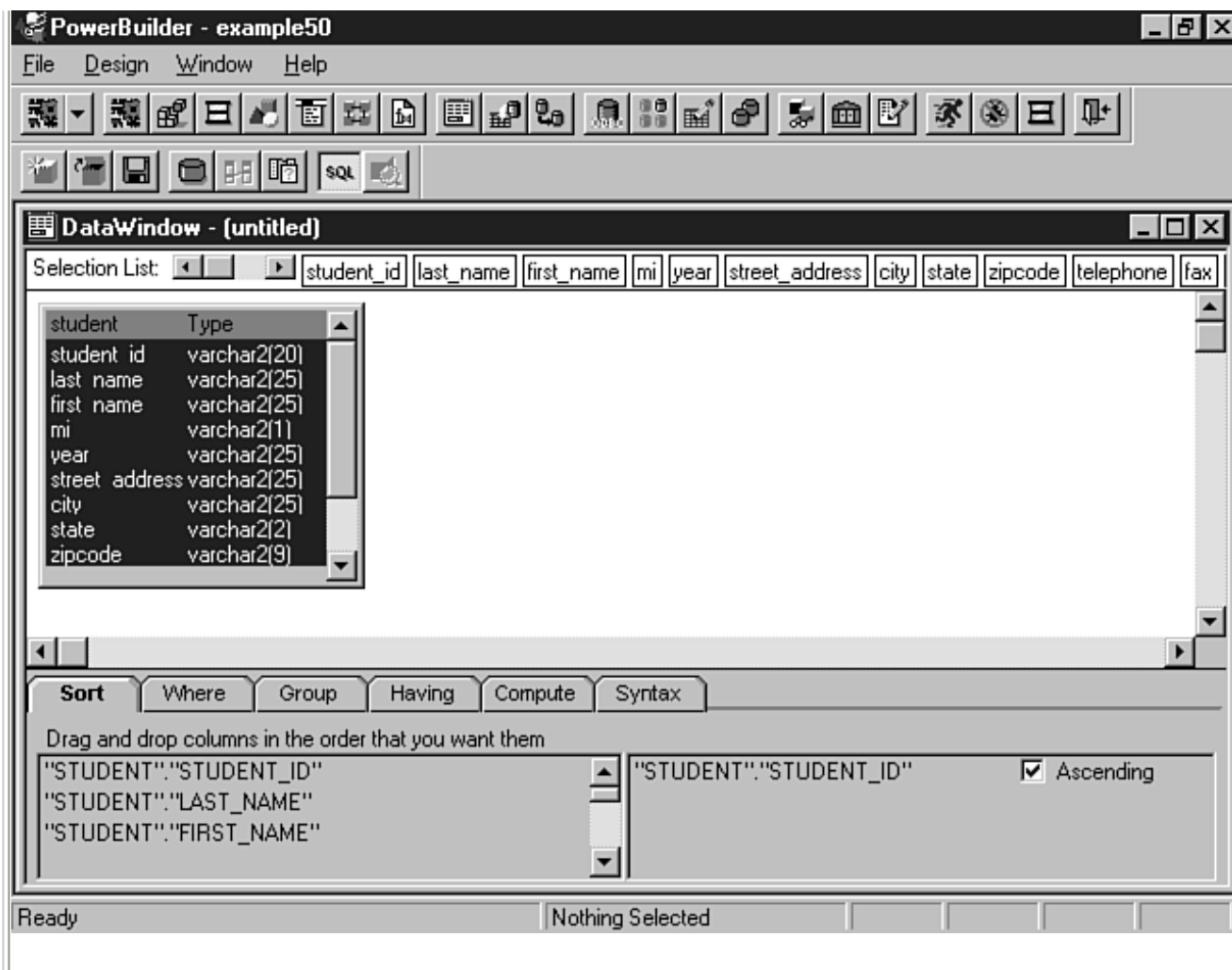


Abbildung 22.9:  
Festlegen der  
Sortierreihenfolge  
für das  
DataWindow

Wenn Sie die SQL-Anweisung sehen wollen, die vom DataWindow Painter erstellt wurde, wählen Sie die Karteikarte Syntax. Sie können mit Hilfe der Bildlaufleiste die SQL-Anweisungen durchblättern. Wie Sie in Abbildung 22.10 sehen können, setzt der DataWindow Painter als Standard die Tabellen- und Spaltennamen in Anführungszeichen. Außerdem enthält jede Spaltenreferenz die Tabelle, zu der sie gehört, auch wenn nur eine einzige Tabelle abgefragt wird.

Um das Datenfenster zu erstellen, wählen Sie den Menüeintrag Design | Data Source. Sie sehen jetzt das Standard-Datenfenster-Layout, das der DataWindow Painter erzeugt hat (siehe Abbildung 22.11). Der DataWindow Painter erlaubt Ihnen die Platzierung von Objekten in vier Bereichen: Header, Body, Summary und Footer. Wie Sie sehen können, wurde jede Spaltenüberschrift durch das Entfernen des Unterstrichs in jedem Spaltennamen und durch Großschreibung jedes Wortes erzeugt. Jede Spalte ist im Body-Bereich platziert.

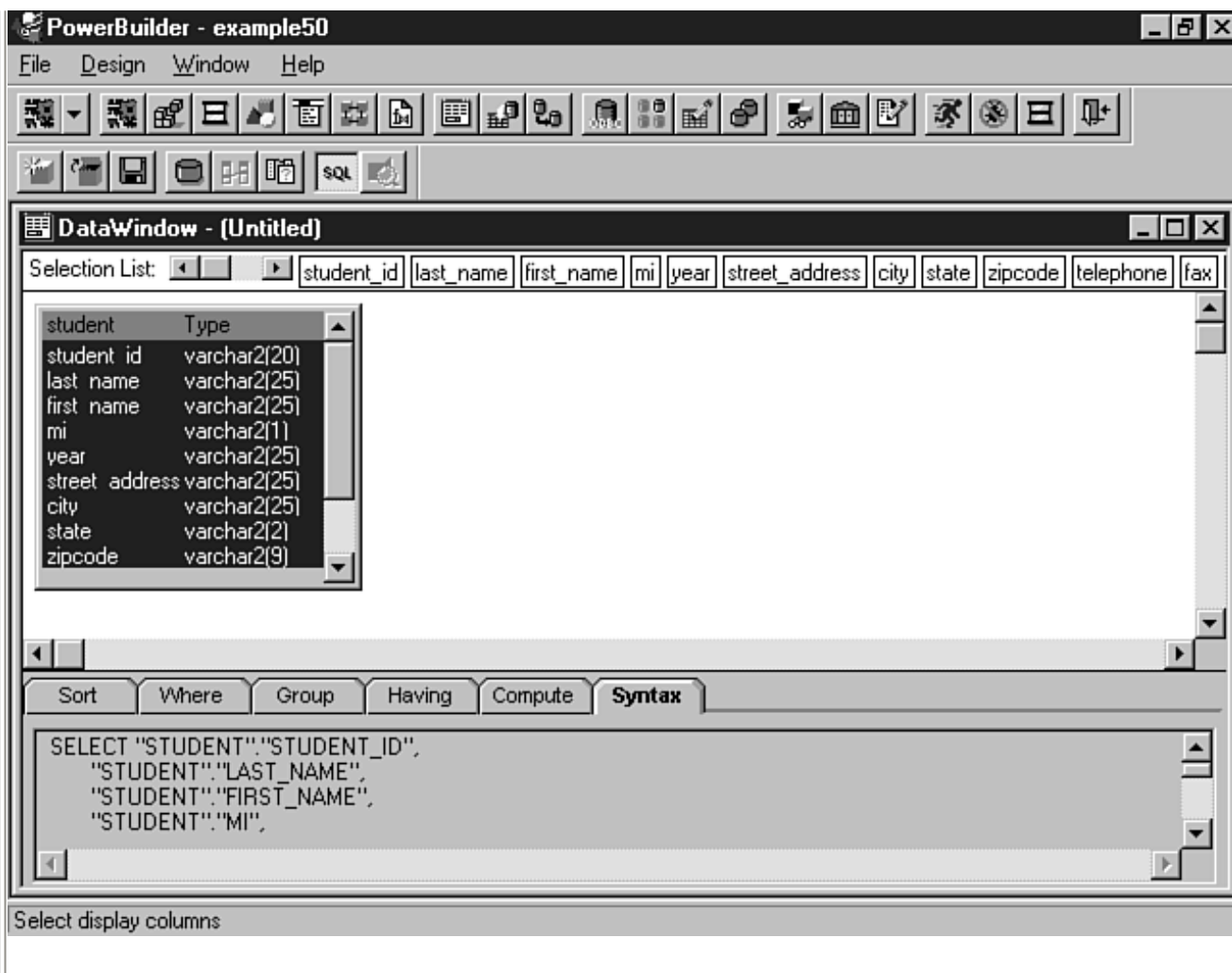


Abbildung 22.10:  
Der DataWindow  
Painter zeigt  
Tabellen- und  
Spaltennamen in  
Anführungszeichen an

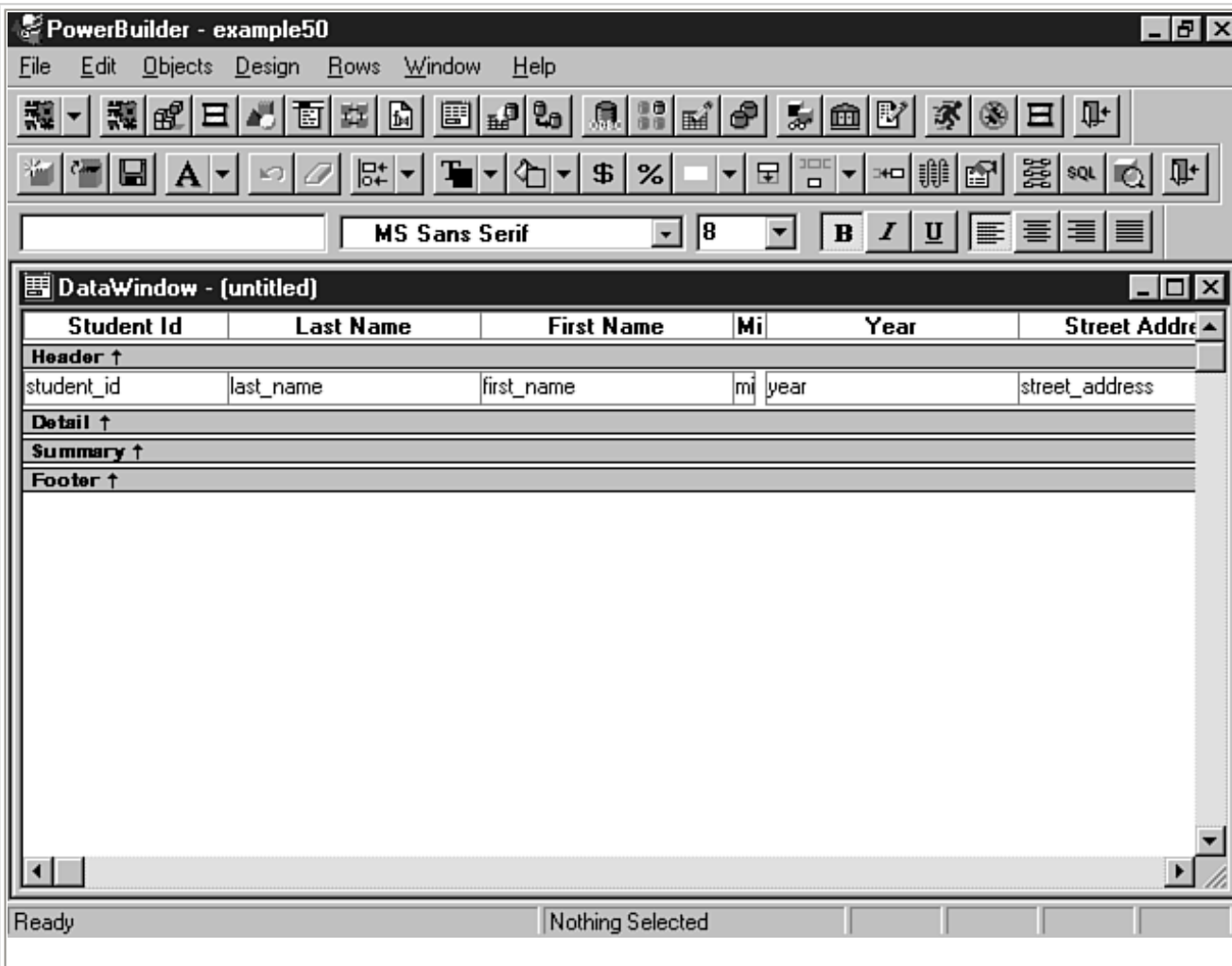
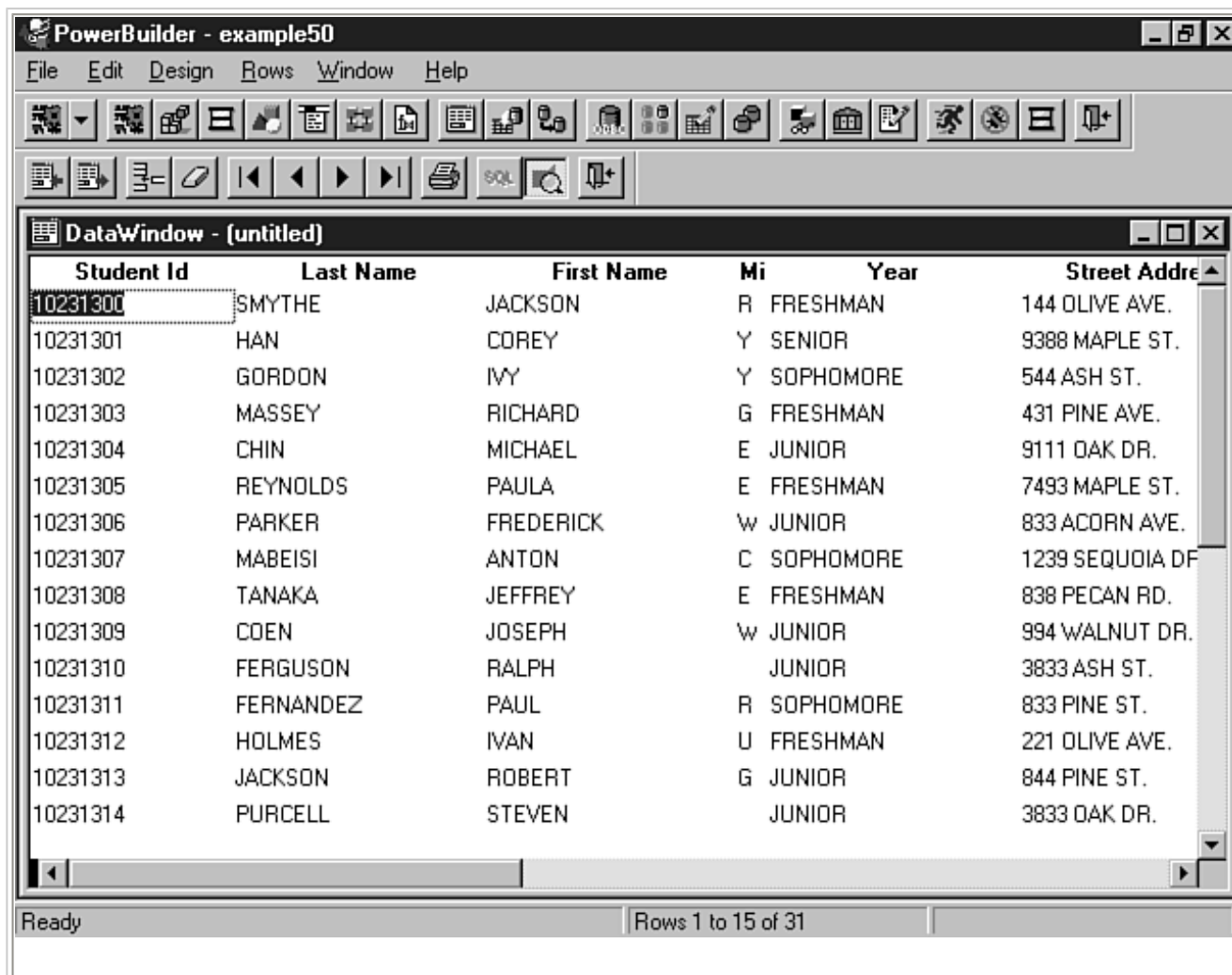


Abbildung 22.11:  
Standardlayout für  
das neue  
DataWindow

Um das Datenfenster in einer Vorschau zu sehen, wählen Sie den Menüeintrag Design | Preview. PowerBuilder wird die Abfrage ausführen, die das Datenfenster benutzt, und die Ergebnisse anzeigen (siehe Abbildung 22.12). Im Vorschau-Modus können Sie ebenfalls Zeilen im Datenfenster einfügen, löschen oder aktualisieren. Um den Vorschau-Modus zu beenden, wählen Sie erneut Design | Preview.



PowerBuilder - example50

File Edit Design Rows Window Help

DataWindow - (untitled)

| Student Id | Last Name | First Name | Mi | Year      | Street Address   |
|------------|-----------|------------|----|-----------|------------------|
| 10231300   | SMYTHE    | JACKSON    | R  | FRESHMAN  | 144 OLIVE AVE.   |
| 10231301   | HAN       | COREY      | Y  | SENIOR    | 9388 MAPLE ST.   |
| 10231302   | GORDON    | IVY        | Y  | SOPHOMORE | 544 ASH ST.      |
| 10231303   | MASSEY    | RICHARD    | G  | FRESHMAN  | 431 PINE AVE.    |
| 10231304   | CHIN      | MICHAEL    | E  | JUNIOR    | 9111 OAK DR.     |
| 10231305   | REYNOLDS  | PAULA      | E  | FRESHMAN  | 7493 MAPLE ST.   |
| 10231306   | PARKER    | FREDERICK  | W  | JUNIOR    | 833 ACORN AVE.   |
| 10231307   | MABEISI   | ANTON      | C  | SOPHOMORE | 1239 SEQUOIA DR. |
| 10231308   | TANAKA    | JEFFREY    | E  | FRESHMAN  | 838 PECAN RD.    |
| 10231309   | COEN      | JOSEPH     | W  | JUNIOR    | 994 WALNUT DR.   |
| 10231310   | FERGUSON  | RALPH      |    | JUNIOR    | 3833 ASH ST.     |
| 10231311   | FERNANDEZ | PAUL       | R  | SOPHOMORE | 833 PINE ST.     |
| 10231312   | HOLMES    | IVAN       | U  | FRESHMAN  | 221 OLIVE AVE.   |
| 10231313   | JACKSON   | ROBERT     | G  | JUNIOR    | 844 PINE ST.     |
| 10231314   | PURCELL   | STEVEN     |    | JUNIOR    | 3833 OAK DR.     |

Ready Rows 1 to 15 of 31

Abbildung 22.12: Vorschau für das neue DataWindow

## Fehlerbehandlung mit Oracle

Um von einer PowerBuilder-Anwendung aus mit einer Oracle-Datenbank zu kommunizieren, müssen Sie erfolgreich ein Transaktionsobjekt mit der Datenbank verbunden haben. PowerBuilder benutzt als Standard ein Transaktionsobjekt namens SQLCA, um mit der Datenbank zu kommunizieren. Das Transaktionsobjekt besitzt einen Satz von Eigenschaften, die auf die richtigen Werte gesetzt werden müssen, um eine Oracle-Verbindung aufzubauen. Diese Eigenschaften sind

**ServerName** Der Datenbank-Alias, der benutzt werden soll, wenn die Verbindung zu einer Oracle-Datenbank aufgebaut wird

**DBMS** O72 Oracle v7.2 (es ist wichtig, daß diese Zeichenfolge genau dem entspricht, was PowerBuilder im Fenster Database Profile anzeigt)

**LogID** Die Oracle-Benutzerbezeichnung, die für die Verbindung verwendet wird

**LogPass** Das Kennwort des zu LogID gehörenden Oracle-Benutzers

Sind die korrekten Transaktionseigenschaften gesetzt, können Sie die CONNECT-Anweisungen benutzen, um die Verbindung mit der Oracle-Datenbank aufzubauen. Weil es sich um eine eingebettete SQL-Anweisung handelt, wird die CONNECT-Anweisung mit einem Semikolon abgeschlossen. Nachdem die CONNECT-Anweisung ausgeführt wurde, werden Sie eine andere Eigenschaft eines Transaktionsobjektes auswerten müssen, um festzustellen, ob die Verbindung erfolgreich war. Es gibt verschiedene Eigenschaften von Transaktionsobjekten, deren Werte nach jeder SQL-Anweisung untersucht werden können.



**SQLCode** Dies ist ein DBMS-unabhängiger Fehlercode, dessen Wert Null ist, wenn es bei der Ausführung der letzten SQL-Anweisung keinen Fehler gab.

**SQLDBCode** Dies ist ein DBMS-spezifische Fehlercode, dessen Wert Null ist, wenn es bei der Ausführung der letzten SQL-Anweisung keinen Fehler gab. Falls **SQLDBCode** jedoch nicht gleich Null ist, enthält es einen herstellerspezifischen Fehlercode.

**SQLErrText** Dies ist eine DBMS-spezifische Fehlermeldung, die mit dem Wert von **SQLDBCode** korrespondiert.

**SQLNRows** Wenn Sie eine Verbindung zu einer Oracle-Datenbank aufbauen, zeigt dieser Wert die Zahl der Reihen an, die von der letzten SQL-Anweisung betroffen waren.

Listing 22.1 zeigt das Skript für das Open-Ereignis einer Anwendung, die eine Verbindung zu einem Oracle-Server aufbaut. Dieser erste Teil des Skripts weist den Eigenschaften von **SQLCA**, dem Standard-Transaktionsobjekt, einen Wert zu. Die **CONNECT**-Anweisung wird benutzt, um die Verbindung zur Oracle-Datenbank herzustellen. **SQLDBCode** wird untersucht, um herauszufinden, ob der Wert kleiner als Null ist; falls ja, ist während des Verbindungsversuchs ein Fehler aufgetreten.

### *Listing 22.1: Skript zum Öffnen eines Anwendungsobjekts*

```
SQLCA.ServerName = ""
SQLCA.DBMS = "072 Oracle v7.2"
SQLCA.LogID = "flugle"
SQLCA.LogPass = "flugle"

Connect;

if SQLCA.SQLDBCode < 0 then
messagebox("Unable to connect to the database", &
"SQLDBCode: " + string(SQLCA.SQLDBCode))
end if

open (w_genapp_frame)
```



*Das Open-Skript wird benutzt, um die Verwendung eines Transaktionsobjekts zu veranschaulichen; in einer realen Anwendung besteht keine Notwendigkeit, diese Werte in einem Skript fest einzutragen*

---

## Zusammenfassung

PowerBuilder ist eine exzellente Entwicklungsumgebung für Anwendungen, die gut an eine Oracle-Datenbank sowie an andere Datenbanken angepaßt ist. Die Newsgroup comp.soft.sys.powerbuilder im Usenet ist eine sehr gute Quelle für Informationen - Beispiele, Antworten und Tips - über PowerBuilder und seine Verwendung mit einer Oracle-Datenbank.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

# 23

## Bonustag: Oracle8 - Neue Merkmale

[Verbesserung vorhandener und Ergänzung um neue Datentypen](#)

[Neue Datentypen, die mit der Oracle Objects Option verfügbar werden](#)

[Die select-Anweisung betreffende Änderungen](#)

[Zugriff auf externe Prozeduren](#)

[Objektansichten](#)

[Betrachtungen zur weiteren Entwicklung](#)

[Zusammenfassung](#)

---

Am 24. Juni 1997 hat die Oracle Corporation das nächste größere Update des Oracle RDBMS formell angekündigt: Oracle8. Diese neue Version ist eine der Hauptkomponenten in Oracles Network Computing Architecture. Die Oracle Corporation hebt einige der neuen Merkmale von Oracle8 Server hervor, insbesondere:

- **Verbesserte Skalierbarkeit:** Die maximale Größe einer Oracle-Datenbank wurde vergrößert, um abhängig vom Betriebssystem, auf dem sie sich befindet, Hunderte von Terabytes zu unterstützen.



- **Verbesserte Sicherheits-Administration:** Der Oracle8 Server enthält jetzt beispielsweise ein Kennwort-Management, so daß ein Kennwort eine begrenzte Gültigkeitsdauer hat und verschiedene komplexe Kriterien erfüllen muß, wie z.B. eine minimale Länge. Ein Account kann nach einer vorgegebenen Zahl von fehlgeschlagenen Login-Versuchen gesperrt werden.
- **Verbesserte Performance durch Partitionierung:** Eine Tabelle oder ein Index können basierend auf dem Wert einer oder mehrerer Spalten in kleinere Abschnitte aufgeteilt werden, die man als Partitionen bezeichnet. Die Partitionen einer Tabelle können individuell verwaltet werden, so daß Operationen in einer Partition die Verfügbarkeit der Daten in einer anderen Partition nicht beeinflussen. Es können auch gleichzeitig Einfüge-, Erneuerungs- und Löschoperationen in partitionierten Tabellen ausgeführt werden; mit anderen Worten, der Oracle8 Server kann einen Teil der Ausführungsarbeit einer einzelnen DML-Anweisung verschiedenen Prozessen zuweisen, die dann vom Betriebssystem des Servers auf mehrere Prozessoren übertragen werden. Als Ergebnis wird eine parallele DML-Operation schneller abgeschlossen.
- **Erweiterter Support für Datenbankreplikation:** Die Leistungsfähigkeit und Handhabung von Datenbankreplikation wurde deutlich verbessert.
- **Die Fähigkeit, eine sehr viel größere Zahl gleichzeitig arbeitender Benutzer zu verwalten:** Indem ein Pooling für Datenbankverbindungen eingerichtet wurde, ist der Oracle8 Server in der Lage, eine deutlich größere Zahl von Benutzern gleichzeitig zu bedienen - bis zu 3000, abhängig vom Server-Betriebssystem und den Server-Hardware-Ressourcen.
- **Neue und verbesserte Datentypen:** Einige existierende Datentypen wurden erweitert, und es wurden neue Datentypen eingeführt.
- **Verbesserte SELECT-Anweisung:** ein neues Merkmal der SELECT-Anweisung erlaubt die Verwendung einer Unterabfrage anstelle einer Tabelle in einer FROM-Bedingung.
- **Objektrelationale Merkmale mittels der Oracle Objects Option:** Eine neue RDBMS-Option unterstützt die Definition und Verwendung von neuen Datentypen.



*Die Oracle Corporation hat angekündigt, daß das Oracle8-Migrationswerkzeug eine Oracle7-Datenbank (Version 7.1, 7.2 oder 7.3) in eine Oracle8-Datenbank umwandeln kann. Zusätzlich dazu kann eine Anwendung, die gegenwärtig mit einer Oracle7-Datenbank arbeitet, problemlos mit einer Oracle8-Datenbank arbeiten. Aus diesem Grund werden alle bisher in diesem Buch diskutierten Merkmale von SQL und PL/SQL auch in Oracle8 funktionieren.*

---

Viele dieser neuen Möglichkeiten sind von hauptsächlichem Interesse für Datenbank-Administratoren und System-Architekten. Diese Lektion wird sich jedoch auf solche Merkmale konzentrieren, die für Datenbank-Designer und Anwendungsentwickler von Interesse sind. Das Material dieser Lektion basiert auf Oracle8 für Windows NT, Version 8.0.2.0.1 Beta2.

## Verbesserung vorhandener und Ergänzung um neue Datentypen

Oracle8 hat die maximale Größe einiger bereits bestehender Oracle-Datentypen vergrößert:

- Die Kapazität einer RAW-Spalte wurde von 255 Byte auf 2000 Byte vergrößert.
- Die Kapazität einer VARCHAR2-Spalte wurde von 2000 Zeichen auf 4000 Zeichen vergrößert.

Außerdem kann eine einzelne Tabelle jetzt bis zu 1000 Spalten besitzen.

Zwei neue Datentypen, NCHAR und NCHAR VARYING, können mit Oracle8 benutzt werden. Diese Datentypen können für landesspezifische Zeichensätze verwendet werden, die mehr als ein Byte pro Zeichen erfordern; dies ist für viele asiatische Sprachen notwendig.



*Die Beispiele in dieser Lektion wurden mit SQL\*Plus und SQL Worksheet ausgeführt.*

---

## Große Objekte (lobs)

Oracle8 bietet viel mehr Unterstützung im Bereich großer Objekte. Tatsächlich wurde eine neue Kategorie von Datentypen, genannt LOB oder Large Objects, mit Oracle8 eingeführt.

LOBs können einer von vier Datentypen sein:

- BLOBs, die unstrukturierte binäre Daten speichern
- CLOBs, die Zeichenfolgedaten speichern
- NCLOBs, die Zeichenfolgedaten unter Verwendung eines landesspezifischen Zeichensatzes speichern (hauptsächlich zur Unterstützung asiatischer Sprachen)
- BFILEs, die binäre Dateien im Dateisystem referenzieren

Die Benutzung von LOBs besitzt mehrere Vorteile. Erstens, ein LOB kann 4 Gbyte oder die doppelte Kapazität der LONG- oder LONG RAW-Spalte aufnehmen. Zweitens, eine Tabelle kann mehr als eine LOB-Spalte enthalten, jedoch nur eine LONG- oder LONG RAW-Spalte. Außerdem werden die in einer LOB-Spalte gespeicherten Daten in einem anderen Bereich gespeichert als die Tabelle, welche die LOB-Spalte enthält, was zu einer besseren Gesamt-Performance führt. Lassen Sie uns einige Beispiele betrachten, die zeigen, wie diese Datentypen benutzt werden können.

## Verwenden von blobs

Sie können sich den BLOB-Datentyp als eine neue, verbesserte Version des LONG RAW-Datentyps vorstellen. Eine BLOB-Spalte kann doppelt so viel aufnehmen wie eine LONG RAW-Spalte. Anders als eine LONG RAW-Spalte kann eine INSERT-Anweisung mit einer Unterabfrage eine BLOB-Spalte enthalten. Listing 23.1 ist ein Beispiel für eine Tabelle, die eine BLOB-Spalte enthält. Diese wird benutzt, um das eingescannte Bild des Lebenslaufs eines potentiellen Dozenten zu speichern.

### *Listing 23.1: Erstellen einer Tabelle, die eine Spalte vom Typ blob enthält*

```
SQL> create table Instructor_Application (
2 Application_Number number,
3 Last_Name varchar2(30),
4 First_Name varchar2(30),
5 MI varchar2(1),
6 Resume blob,
7 Status varchar2(30));
```

Table created.

## Verwenden von clobs

Der CLOB-Datentyp ist dem Datentyp LONG ähnlich. Wie dieser, wird der CLOB-Datentyp benutzt, um lange Zeichenfolgen zu speichern. Der CLOB-Datentyp besitzt jedoch die doppelte Kapazität des Typs LONG - 4 Gbyte anstelle von 2 Gbyte.

In Listing 23.2 können Sie an einem Beispiel sehen, wie eine CLOB-Spalte namens OCR\_RESUME zu der Tabelle Instructor\_Application hinzugefügt wird. Der Zweck der OCR\_Resume-Spalte ist die Speicherung des Textes, der von der Scanner-Software aus dem eingescannten Lebenslauf extrahiert wurde.

### *Listing 23.2: Erweiterung einer Tabelle um eine Spalte vom Typ clob*

```
SQL> alter table Instructor_Application add
2 (OCR_Resume clob);
```

Table altered.

```
SQL> update Instructor_Application
2 set OCR_Resume = 'Resume of A. Einstein - Born in Ulm, Germany in
1879'
3 ;
```

1 row updated.

```
SQL> select OCR_Resume from Instructor_Application;
```

```
OCR_RESUME
```

```
-----
Resume of A. Einstein - Born in Ulm, Germany in 1879
```

---



*Nachdem die CLOB-Spalte zur Tabelle hinzugefügt wurde, wird eine UPDATE-Anweisung benutzt, um den Wert von OCR\_Resume zu setzen. Eine SELECT-Anweisung veranschaulicht, daß die Inhalte von OCR\_Resume aus der Tabelle abgefragt werden können.*

---

Ein weiterer wichtiger Unterschied zwischen den Datentypen CLOB und LONG ist, daß es für die Referenzierung einer CLOB-Spalte in einer SQL-Anweisung weniger Beschränkungen gibt. Schauen Sie sich das Beispiel in Listing 23.3 an.

***Listing 23.3: Einfügen einer Zeile in eine Tabelle mit einer Spalte vom Typ long***

```
SQL> create table New_Instructor_Application
2 as
3 select * from Instructor_Application;
```

Table created.

```
SQL> create table Old_Instructor_Application (
2 Application_Number number,
3 Last_Name varchar2(30),
4 First_Name varchar2(30),
5 MI varchar2(1),
6 Resume long,
7 Status varchar2(30));
```

Table created.

```
SQL> create table Another_Instructor_Application
2 as
3 select * from Old_Instructor_Application;
select * from Old_Instructor_Application
*
```

ERROR at line 3:

## ORA-00997: illegal use of LONG datatype



Wie Sie in Listing 23.3 sehen können, wird eine Tabelle namens *New\_Instructor\_Application* mit einer Unterabfrage erstellt, die alle Spalten in *Instructor\_Application* abfragt, einschließlich *OCR\_Resume*. Die nächste Anweisung erstellt eine Tabelle namens *Resume* mit dem Datentyp **LONG**. Falls Sie versuchen, eine andere Tabelle namens *Another\_Instructor\_Application* mittels einer Unterabfrage von *Old\_Instructor\_Application* zu erstellen, weist Oracle diese Anweisung zurück.

## Festlegen eines Nullwerts für ein blob oder ein clob

Es gibt zwei neu integrierte Funktionen, die benutzt werden, um einem LOB den Wert **NULL** zuzuweisen: **EMPTY\_BLOB** und **EMPTY\_CLOB**. Beide Funktionen besitzen keine Argumente. Listing 23.4 veranschaulicht die Benutzung von **EMPTY\_BLOB** in einer **INSERT**-Anweisung.

### Listing 23.4: Einfügen einer Zeile in eine Tabelle mit einer Spalte vom Typ blob

```
SQL> insert into Instructor_Application
2 (Application_Number, Last_Name, First_Name, MI, Resume, Status)
3 values
4 (1001, 'EINSTEIN', 'ALBERT', NULL, EMPTY_BLOB(), 'REJECTED');

1 row created.
```

## Verwenden von bfiles

Eine **BFILE**-Spalte wird benutzt, um eine binäre Datei zu referenzieren, die in dem Dateisystem gespeichert ist, wo sich der Oracle8 Server befindet; die Binärdatei befindet sich außerhalb der Oracle8-Datenbank. Als Beispiel demonstriert Listing 23.5, wie eine Tabelle erzeugt wird, die eine **BFILE**-Spalte enthält.

### Listing 23.5: Definieren und Verwenden einer Spalte vom Typ bfile

```
SQL> create table Instructor_Photo
2 (Instructor_ID varchar2(5),
3 Recent_Photo bfile);
```

Table created.

```
SQL> insert into Instructor_Photo
2 (Instructor_ID, Recent_Photo)
3 values
4 ('E251', BFILENAME ('C:\Flugle\Photos', 'E251.jpg'));
```

1 row created.



*Die Instructor\_Photo-Tabelle wird mit einer BFILE-Spalte namens Recent\_Photo erzeugt. Der Zweck dieser Spalte ist es, den Ort und den Namen der .JPG-Datei zu bestimmen, die ein Foto des Dozenten, identifiziert durch die Instructor\_ID, enthält. Eine Zeile wird in die Instructor\_Photo-Tabelle eingefügt. Wie Sie sehen können, wird die BFILENAME-Funktion zur Bereitstellung eines Literalwertes für die BFILE-Spalte benutzt; das Verzeichnis und der Dateiname werden in Hochkommas eingeschlossen. Der Inhalt der mittels der BFILE-Spalte referenzierten binären Datei kann mit einer SQL-Anweisung nicht geändert werden; das Verzeichnis und der Dateiname, wie er in der BFILE-Spalte gespeichert ist, jedoch schon.*

## Die create directory-Anweisung

Eine mit Oracle8 neu eingeführte Anweisung ist CREATE DIRECTORY. Diese Anweisung ist für die Arbeit mit BFILE-Spalten gedacht. Der Zweck von CREATE DIRECTORY ist das Erstellen eines Alias für das Verzeichnis eines Dateisystems. Ist der Verzeichnis-Alias definiert, kann die Berechtigung zum Lesen der Inhalte einer Datei in diesem Verzeichnis einer Rolle oder einem Benutzer zugeteilt werden.



*Hier ist die Syntax für die CREATE DIRECTORY-Anweisung:*

```
CREATE DIRECTORY directory_alias as 'directory';
```

Die Variablen sind folgendermaßen definiert:

- *directory\_alias* ist der Name, den Oracle-Benutzer zur Referenzierung des Verzeichnisses benutzen müssen.
- *directory* ist der vollständige Name des Verzeichnisses, in dem die binären Dateien sich befinden, die von einer BFILE-Spalte referenziert werden.

Listing 23.6 veranschaulicht zum Beispiel, wie ein Verzeichnis erzeugt wird. Die Berechtigung zum Lesen von Dateien aus diesem Verzeichnis kann einer Datenbankrolle gewährt werden.

**Listing 23.6: Erstellen eines Verzeichnisses**

```
SQL> create directory instructor_photos as 'C:\Flugle\Photos';
```

Directory created.

Zusätzlich zur CREATE DIRECTORY-Anweisung wird eine andere Anweisung - DROP DIRECTORY - benutzt, um einen Verzeichnis-Alias zu löschen, so daß keine Berechtigung zum Lesen des Verzeichnisses mit einer Oracle8-Anwendung zugeteilt werden kann.

## Neue Datentypen, die mit der Oracle Objects Option verfügbar werden

Die Oracle Objects Option ist eine optionale Komponente des Oracle8-Servers. Sie müssen diese Option haben, um in der Lage zu sein, die folgenden Datentypen zu benutzen:

- Der Object-Datentyp
- Verschachtelte Tabellen
- VARRAY
- REF

Wir werden in diesem Abschnitt den Objekt-Datentyp, verschachtelte Tabellen und den VARRAY-Datentyp untersuchen. Um zu beginnen, lassen Sie uns den Objekt-Datentyp betrachten.



*Vererbung ist ein wichtiges Konzept in objektorientierten Programmiersprachen. Zum gegenwärtigen Zeitpunkt unterstützen weder SQL noch PL/SQL die Vererbung.*

---

## Der Datentyp object

Die Oracle8 Objects Option bietet eine neue Anweisung `CREATE TYPE`, mit der ein neuer Datentyp erzeugt werden kann, der dann benutzt wird, wenn eine neue Tabelle oder ein anderer Datentyp erstellt werden soll. Wenn Sie zum Beispiel eine Tabelle erstellen wollten, welche die Ausstattung beschreibt, die ein Dozent in einem Klassenraum verwenden möchte, können Sie einen neuen Datentyp mit Namen `Classroom_Equipment` für diesen Zweck erzeugen. Ist der Datentyp erstellt, können Sie eine Tabelle erzeugen, die diesen Datentyp benutzt (siehe Listing 23.7).

### *Listing 23.7: Erstellen eines neuen Datentyps*

```
SQL> create type Available_Room as object
2 (building varchar2(20),
3 room varchar2(6));
```

Type created.

Haben Sie den neuen Datentyp definiert, können Sie ihn für eine Spalte festlegen, wie in Listing 23.8 demonstriert wird.

### *Listing 23.8: Erstellen einer Tabelle, die einen Objektdatentyp enthält*

```
SQLWKS> create table Seminar (
2> Seminar_ID varchar2(6),
3> Instructor_ID varchar2(6),
4> Location Available_Room);
Statement processed.
```

Ein Objekt besteht aus Attributen (die dafür definierten Spalten) und Methoden (die Funktionen und Prozeduren, die das Objekt manipulieren und Informationen darüber zurückgeben). Als Standard erstellt Oracle eine Konstruktor-Methode für jeden Objekttyp. Diese Konstruktor-Methode erzeugt eine Instanz des Objekts. Wenn Sie einen Wert für die Objektspalte festlegen wollen, müssen Sie die Konstruktor-Methode benutzen, die aus dem Namen des Objekttyps besteht, gefolgt von den in runde Klammern eingeschlossenen Attributwerten. Listing 23.9 bietet dafür ein Beispiel.

### *Listing 23.9: Festlegen eines Wertes für einen Objektdatentyp mit Hilfe eines Konstruktors*

```
SQLWKS> insert into Seminar
2> (Seminar_ID, Instructor_ID, Location)
3> values
4> ('1001', 'A101', Available_Room('NARROW HALL', 'B200'));
1 row processed.
```

Wenn Sie die Tabelle erstellt haben, die eine Objektspalte enthält, können Sie eine `SELECT`-Anweisung benutzen, um die Werte aus der Objektspalte abzurufen. Sie müssen jedoch jedes



Objektattribut festlegen, das Sie abrufen wollen (siehe Listing 23.10).

***Listing 23.10: Kennzeichnung eines Attributs mit dem Namen der Objektspalte***

```
SQLWKS> select Location.Building, Location.Room from Seminar;
LOCATION.BUILDING LOCATI
-----
NARROW HALL B200
1 row selected.
```

## Verschachtelte Tabellen

Oracle8 unterstützt ebenfalls die Verwendung verschachtelter Tabellen. Nehmen Sie einmal an, Sie erzeugen einen Objekt-Datentyp mit der Bezeichnung `Test_Score`, der dafür gedacht ist, eine Studenten-ID und die Ergebnisliste eines bestimmten Tests zu speichern (siehe Listing 23.11). Sie können außerdem einen anderen Datentyp mit der Bezeichnung `Test_Score_Table` erstellen, bei dem es sich um eine Tabelle handelt, die auf dem `Test_Score`-Datentyp basiert. Abschließend können Sie den `Test_Score_Table`-Datentyp benutzen, um den Datentyp für eine Spalte festzulegen, wenn Sie eine Tabelle erzeugen oder ändern.

***Listing 23.11: Erstellen einer Tabelle, die eine Tabelle als Spalte enthält***

```
SQLWKS> create type Test_Score as object (
2> Student_ID varchar2(6),
3> Score number);
Statement processed.

SQLWKS> create type Test_Score_Table as table of Test_Score;
Statement processed.

SQLWKS> create table Test_Results (
2> Instructor_ID varchar2(6),
3> Class_ID varchar2(6),
4> Test_Name varchar2(30),
5> Scores Test_Score_Table)
6> nested table Scores store as Test_Scores;
Statement processed.
```



Die letzte SQL-Anweisung in Listing 23.11 ist eine `CREATE TABLE`-Anweisung. Wenn Sie

*eine Tabelle erstellen, die eine Spalte enthält, bei der es sich um eine verschachtelte Tabelle handelt, müssen Sie eine Speichertabelle festlegen, die für die verschachtelte Tabelle bestimmt ist.*

---

Wenn Sie einen Satz von Werten für eine verschachtelte Tabelle in einer DML-Anweisung bestimmen, müssen Sie die Konstruktor-Methode für den Datentyp festlegen, auf dem die verschachtelte Tabelle basiert. Listing 23.12 präsentiert ein Beispiel; eine einzelne Zeile wird in die Test\_Results-Tabelle eingefügt. Obwohl nur eine einzelne Zeile in die Test\_Results-Tabelle eingefügt wurde, werden drei Testergebnisse in der Scores-Spalte gespeichert, bei der es sich um eine verschachtelte Tabelle handelt.

***Listing 23.12: Einfügen einer Zeile in eine Tabelle, die eine Tabelle als Spalte enthält***

```
SQLWKS> insert into Test_Results
2> (Instructor_ID, Class_ID, Test_Name, Scores)
3> values
4> ('E101', '123456', 'Final exam',
5> Test_Score_Table
6> (Test_Score ('A12345', 98),
7> Test_Score ('E13111', 87),
8> Test_Score ('F13999', 84)));
1 row processed.
```

---



*In Zeile 5 wird die Konstruktor-Methode für die verschachtelte Tabelle vom Typ Test\_Score\_Table festgelegt. In den Zeilen 6, 7 und 8 wird die Konstruktor-Methode für Test\_Score mit jedem Satz von Attributwerten festgelegt.*

---

## Der Datentyp varray

Ein in Oracle8 neu zur Verfügung stehender Datentyp ist **VARYING ARRAY** oder **VARRAY**. Dieser Datentyp repräsentiert einen sortierten Satz von Elementen desselben Datentyps.

---



*Hier ist eine vereinfachte Version der VARRAY-Syntax:*

---

```
CREATE TYPE type_name as VARRAY (limit) of datatype;
```

Die Variablen sind wie folgt definiert:

- *type\_name* ist der Name des zu erstellenden VARRAY-Typs.
- *limit* ist die maximale Zahl von Elementen in dem Array.
- *datatype* ist ein entweder vordefinierter oder benutzerdefinierter Datentyp.

Das VARRAY-Limit muß ein Integer-Literal sein; Sie können für die Festlegung des Limits keinen Ausdruck verwenden. Wie Sie in Listing 23.13 sehen können, kann ein VARRAY-Datentyp als solcher für eine Spalte benutzt werden, sobald er deklariert ist.

*Listing 23.13: Erstellen einer Tabelle, die eine Spalte vom Typ varray enthält*

```
SQL> create type Available_Rooms as varray (100) of Available_Room;
```

Type created.

```
SQL> create table Class_Scheduling (  
2 Administrator_ID varchar2(6),  
3 Semester varchar2(6),  
4 Year number,  
5 Room_Assignments Available_Rooms);
```

Table created.

Lassen Sie uns einen Blick darauf werfen, wie Sie Werte für eine VARRAY-Spalte in einer INSERT-Anweisung festlegen können. Wenn Sie einen Nullwert für die VARRAY-Spalte festlegen wollen, können Sie die Konstruktor-Methode für den VARRAY-Datentyp mit einem NULL-Argument ausführen (siehe Listing 23.14).

*Listing 23.14: insert-Anweisung, in der für eine Spalte vom Typ varray ein Nullwert angegeben ist*

```
SQL> insert into Class_Scheduling  
2 (Administrator_ID, Semester, Year, Room_Assignments)  
3 values  
4 ('101', 'FALL', '1998', Available_Rooms(NULL));
```

1 row created.

Wenn Sie einen Wert für eine Objektspalte in einer DML-Anweisung festlegen wollen, müssen Sie die Konstruktor-Methode für den Basis-Objektyp `VARRAY`, eingeschlossen in die Konstruktor-Methode für den `VARRAY`-Typ, festlegen. Listing 23.15 enthält zum Beispiel eine `INSERT`-Anweisung, die versucht, einen zur Verfügung stehenden Unterrichtsraum in der Spalte `Room_Assignments` zu speichern. Oracle weist die erste `INSERT`-Anweisung ab, weil `Available_Rooms` ein `VARRAY` ist. Die zweite `INSERT`-Anweisung benutzt die Konstruktor-Methode für `Available_Room`, eingeschlossen in die Konstruktor-Methode für `Available_Rooms`.

***Listing 23.15: Festlegen eines Wertes für eine Spalte vom Typ varray mit Hilfe einer insert-Anweisung***

```
SQL> insert into Class_Scheduling
2 (Administrator_ID, Semester, Year, Room_Assignments)
3 values
4 ('101', 'FALL', '1998', Available_Rooms('FLUGLE HALL', 100));
('101', 'FALL', '1998', Available_Rooms('FLUGLE HALL', 100))
*
```

```
ERROR at line 4:
ORA-00932: inconsistent datatypes
```

```
SQL> insert into Class_Scheduling
2 (Administrator_ID, Semester, Year, Room_Assignments)
3 values
4 ('101', 'FALL', '1998',
5 Available_Rooms(Available_Room('FLUGLE_HALL', '100')));

1 row created.
```

## Data-Dictionary-Ansichten für benutzerdefinierte Datentypen

Um dem Benutzer zu helfen, benutzerdefinierte Typen auszuwerten, enthält Oracle8 einige neue Data-Dictionary-Ansichten:

- `USER_TYPES` präsentiert grundlegende Informationen über Datentypen, die vom Benutzer erstellt wurden - mit anderen Worten, über andere Datentypen als die vordefinierten.
- `USER_TYPE_ATTRS` bietet detaillierte Informationen über jeden der in `USER_TYPES` aufgelisteten Datentypen.

Listing 23.16 enthält eine Beschreibung der von `USER_TYPES` zurückgegebenen Spalten und einer Beispiel-Abfrage von `USER_TYPES`.



Viele der Oracle-Data-Dictionary-Ansichten sind als USER\_, ALL\_ und DBA\_ kategorisiert. Die Ansichten, die mit USER\_ beginnen, bieten Informationen über die angewendeten Objekte im Besitz des Benutzers, der die Data Dictionary-Ansicht abfragt. Die Ansichten, die mit ALL\_ beginnen, bieten Informationen über alle benutzten Objekte, für die der Benutzer, der die Data-Dictionary-Ansicht abfragt, die Berechtigung besitzt, sie einzusehen. Abschließend gibt es noch Ansichten, die mit DBA\_ beginnen und Informationen über alle in der Datenbank benutzten Objekte bieten. Obwohl dieser Abschnitt nur die Ansichten USER\_TYPES- und USER\_TYPEATTRS-Data Dictionary vorstellte, bietet der Oracle8 Server vier weitere verwandte Ansichten: ALL\_TYPES, ALL\_TYPE\_ATTRS, DBA\_TYPES und DBA\_TYPE\_ATTRS.

---

### **Listing 23.16: Betrachtung des Inhalts von user\_types**

```
SQLWKS> desc USER_TYPES
```

```
Column Name Null? Type
```

```
-----
```

```
TYPE_NAME NOT NULL VARCHAR2(30)
```

```
TYPE_OID NOT NULL RAW(16)
```

```
TYPECODE VARCHAR2(30)
```

```
ATTRIBUTES NUMBER
```

```
METHODS NUMBER
```

```
PREDEFINED VARCHAR2(3)
```

```
INCOMPLETE VARCHAR2(3)
```

```
SQLWKS> select type_name, typecode, attributes, methods from  
user_types;
```

```
TYPE_NAME TYPECODE ATTRIBUTES METHODS
```

```
-----
```

```
AVAILABLE_ROOM OBJECT 2 0
```

```
AVAILABLE_ROOMS COLLECTION 0 0
```

```
CLASS_PRESIDENTS COLLECTION 0 0
```

```
TEST_SCORE OBJECT 2 0
```

```
TEST_SCORE_TABLE COLLECTION 0 0
```

```
5 rows selected.
```

---



Wie Sie in der letzten Abfrage sehen konnten, gibt die TYPECODE-Spalte OBJECT zurück, wenn ein Typ als Objekt definiert ist, oder COLLECTION, wenn ein Typ als VARRAY oder

verschachtelte Tabelle definiert ist. AVAILABLE\_ROOM z.B. war als Objekt definiert, deshalb ist sein TYPECODE OBJECT, wohingegen AVAILABLE\_ROOMS als VARRAY von AVAILABLE\_ROOM definiert war, so daß der TYPECODE COLLECTION ist.

---

Wenn Sie die Attribute einsehen wollen, die für einen Typ definiert wurden, können Sie die Data-Dictionary-Ansicht USER\_TYPE\_ATTRS abfragen. Listing 23.17 bietet eine Beschreibung der Ansicht USER\_TYPE\_ATTRS und eine Beispiel-Abfrage, welche die Attribute zurückgibt, die für den Typ TEST\_SCORE definiert wurden.

***Listing 23.17: Betrachtung des Inhalts von user\_type\_attrs***

```
SQLWKS> desc user_type_attrs
```

```
Column Name Null? Type
```

```
-----
TYPE_NAME NOT NULL VARCHAR2(30)
ATTR_NAME NOT NULL VARCHAR2(30)
ATTR_TYPE_MOD VARCHAR2(7)
ATTR_TYPE_OWNER VARCHAR2(30)
ATTR_TYPE_NAME VARCHAR2(30)
LENGTH NUMBER
PRECISION NUMBER
SCALE NUMBER
CHARACTER_SET_NAME VARCHAR2(44)
```

```
SQLWKS> select attr_name, attr_type_name, length
```

```
2> from user_type_attrs
```

```
3> where
```

```
4> type_name = 'TEST_SCORE';
```

```
ATTR_NAME ATTR_TYPE_NAME LENGTH
```

```
-----
STUDENT_ID VARCHAR2 6
SCORE NUMBER
2 rows selected.
```

## Die select-Anweisung betreffende Änderungen

Ein neues Merkmal, das der Oracle8 Server bietet, ist die Fähigkeit, eine Unterabfrage in der FROM-Bedingung einer SELECT-Anweisung anstelle einer Tabelle oder Datenansicht einzusetzen. Listing 23.18 präsentiert ein Beispiel.

***Listing 23.18: Verwenden einer Unterabfrage in der from-Klausel***

```

SQLWKS> select Operating_Room, Surgeon_ID, Last_Name
2> from (select Patient_ID, Last_Name from Patient
3> where Status != 'MORGUE') P, Surgery_Schedule S
4> where
5> P.Patient_ID = S.Patient_ID;
OPERAT SURGEO LAST_NAME
-----
A10 G101 JOHNSON
1 row selected.

```

---



*In Zeile 2 von Listing 23.18 ist eine Unterabfrage der Patient-Tabelle in runde Klammern eingeschlossen; sie gibt Patient\_ID und Last\_Name nur für diejenigen Patienten zurück, deren Wert für Status nicht gleich 'Morgue' ist. In Zeile 3 wird die Unterabfrage mit der Surgery\_Schedule-Tabelle verbunden, basierend auf dem Verknüpfungskriterium in Zeile 5.*

---

## Zugriff auf externe Prozeduren

Oracle8 gibt einem Entwickler die Möglichkeit, Routinen von einem PL/SQL-Unterprogramm aus aufzurufen, die in anderen Programmiersprachen geschrieben wurden. Diese Routinen, die in 3GL geschrieben sind und sich in einer geteilt nutzbaren Bibliothek befinden, werden als *externe Prozeduren* bezeichnet. Um eine externe Prozedur zu registrieren, enthält Oracle8 eine neue Anweisung, die einem Entwickler die Referenzierung einer externen Bibliothek erlaubt, welche wiederum die *externen Prozeduren* enthält: CREATE LIBRARY (siehe Listing 23.19).

---



*Zum gegenwärtigen Zeitpunkt müssen alle externen Prozeduren in C geschrieben werden.*

---



*Hier ist die Syntax zur Erzeugung einer Bibliothek, die externe Prozeduren enthält:*

```
CREATE LIBRARY library_name as 'file_name';
```

Die Variablen sind wie folgt definiert:

- `library_name` ist der Name der Bibliothek, die von einem PL/SQL-Unterprogramm aus referenziert werden kann.
- `file_name` ist das Verzeichnis und der Name der geteilt nutzbaren Bibliothek.

### ***Listing 23.19: Erzeugung eines Bibliotheksalias für den Zugriff auf externe Prozeduren***

```
SQLWKS> create library flugle_lib as  
'/opt/flugle/lib/flugle_lib.so';  
Statement processed.
```

## **Objektansichten**

Oracle8 führt außerdem noch ein neues Merkmal mit der Bezeichnung *Objektansichten* ein. Eine Objektansicht ist eine Erweiterung der relationalen Datensicht, die Sie bereits kennen. Eines von Oracles Zielen im Zusammenhang mit Oracle8 ist, mit der Unterstützung objektrelationaler Anwendungen zu beginnen; die Objektansicht ist ein Teil dieser Strategie. Objektansichten können mit relationalen Datensichten derselben Tabellen koexistieren; deshalb können Benutzer weiterhin traditionelle relationale Anwendungen benutzen, die mit einem Satz von Tabellen arbeiten, während Entwickler mit objektorientierten Anwendungen experimentieren können, die auf Objektansichten derselben Tabelle beruhen.

Listing 23.20 präsentiert ein Beispiel einer Objektansicht. Zuerst wird die Patient-Tabelle erzeugt. Als nächstes wird der Datentyp `Patient_Type` erstellt, der die Patient-Tabelle spiegelt. Zuletzt wird die Objektansicht `Patient_View` erstellt, die `Patient_ID` als Objekt-ID benutzt.

### ***Listing 23.20: Beispiel für eine Objektansicht***

```
SQLWKS> create table Patient (  
2> Patient_ID varchar2(6),  
3> Last_Name varchar2(30),  
4> First_Name varchar2(20),  
5> MI varchar2(1),  
6> SS_Number varchar2(9),  
7> Location varchar2(6),  
8> Status varchar2(20),  
9> Admittance_Date date,
```



```

10> Discharge_Date date,
11> Carrier varchar2(30));
Statement processed.

```

```

SQLWKS> create type Patient_Type (
2> Patient_ID varchar2(6),
3> Last_Name varchar2(30),
4> First_Name varchar2(20),
5> MI varchar2(1),
6> SS_Number varchar2(9),
7> Location varchar2(6),
8> Status varchar2(20),
9> Admittance_Date date,
10> Discharge_Date date,
11> Carrier varchar2(30));
Statement processed.

```

```

SQLWKS> create view Patient_View of Patient_Type
2> with object oid (Patient_ID) as
3> select Patient_ID, Last_Name, First_Name, MI, SS_Number,
4> Location, Status, Admittance_Date, Discharge_Date, Carrier
5> from Patient;
Statement processed.

```

## Betrachtungen zur weiteren Entwicklung

Obwohl Oracle einen einfachen Migrationsweg zu Oracle8 bietet, ist es schwierig, vorherzusagen, wie schnell bisherige Oracle-Benutzer ihre Datenbanken von Oracle7 nach Oracle8 migrieren werden. Wie dem auch sei, wenn Sie die Entwicklung einer Anwendung planen, die mit Oracle7-Datenbanken arbeitet, sind Sie mit dieser Version der Anwendung nicht in der Lage, die in diesem Kapitel diskutierten neuen Merkmale zu nutzen. Natürlich können Sie eine separate Version für eine Oracle8-Datenbank schreiben, welche einige der neuen Merkmale einschließt. Zum gegenwärtigen Zeitpunkt ist die Unterstützung für die neuen Merkmale begrenzt auf die Verwendung in:

- PL/SQL gespeicherten Programmen,
- C/C++-Programmen, die das Oracle Call Interface (OCI) benutzen, eine Bibliothek von aufrufbaren Funktionen,
- C/C++-Programmen, die mit dem Oracle-Pro\*C/C++-Precompiler entwickelt wurden, einem Werkzeug, das es dem Benutzer erlaubt, SQL-Anweisungen in ein C/C++-Programm einzubetten, welches dann in die nativen C/C++-Anweisungen für einen bestimmten C/C++-Compiler übersetzt wird.

Die Oracle Corporation und andere Drittanbieter von Software-Tools werden in Kürze Werkzeuge auf den Markt bringen, die in der Lage sein werden, die Vorteile der neuen Datentypen und

Möglichkeiten zu nutzen, welche die Object Option bietet. Bis dahin haben Sie die Gelegenheit zu lernen, wie man diese Merkmale nutzbringend in einer Anwendung einsetzt.

## Zusammenfassung

Diese Lektion war eine kurze Einführung in einige der neuen Merkmale, die vom Oracle8 Server geboten werden, wie z.B. die folgenden:

- Die LOB-Datentypen, die aus den Datentypen BLOB, CLOB, NCLOB und BFILE bestehen. Der BLOB-Datentyp kann bis zu 4 Gbyte unstrukturierter Binärdaten speichern. Die CLOB- und NCLOB-Datentypen können bis zu 4 Gbyte an Zeichenfolgedaten speichern. Der BFILE-Datentyp wird benutzt, um eine externe Datei zu referenzieren, die Binärdaten enthält.
- Mit der optionalen Oracle Objects Option können Sie Ihre eigenen Datentypen erzeugen, wie z.B. einen Datentyp Object, ein variables Array aus Basis-Datentypen und eine verschachtelte Tabelle als Datentyp.
- Oracle8 unterstützt die Benutzung einer Unterabfrage in der FROM-Bedingung einer SELECT-Anweisung.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖ Kapitel ❖ [Top](#) [Inhalt](#) [Index](#) **SAMS** Kapitel ❖

# Anhang

# Antworten zu den Tests und Übungen

[Tag 1, »Die Welt relationaler Datenbanken erforschen«](#)

[Tag 2, »Richtlinien für die Entwicklung einer Oracle-Anwendung«](#)

[Tag 3, »Logischer Datenbankentwurf«](#)

[Tag 4, »Implementierung Ihres logischen Modells: physischer Datenbankentwurf«](#)

[Tag 5, »Einführung in SQL«](#)

[Tag 6, »Ändern von Daten mit Hilfe von SQL«](#)

[Tag 7, »Die in SQL integrierten Funktionen ausnutzen«](#)

[Tag 8, »Bilden fortgeschrittener Abfragen mit SQL«](#)

[Tag 9, »Programmierung einer Oracle-Datenbank mit PL/SQL«](#)

[Tag 10, »Programmentwicklung mit PL/SQL«](#)

[Tag 11, »Weitere Programmiertechniken für PL/SQL«](#)

[Tag 12, »Developer/2000: Einführung in Oracle Forms«](#)

[Tag 13, »Entwicklung einer Benutzeroberfläche mit Hilfe von Oracle Forms«](#)

[Tag 14, »Developer/2000: Anwendungsentwicklung mit Hilfe von Oracle Forms«](#)

[Tag 15, »Developer/2000: Entwicklung von Berichten mit Hilfe von Oracle Reports«](#)

[Tag 16, »Developer/2000: Oracle Graphics und den Procedure Builder verwenden«](#)

[Tag 17, »Einführung in Oracle Power Objects«](#)

[Tag 18, »Eine Anwendung mit Oracle Power Objects entwickeln«](#)

[Tag 19, »Eine Übersicht über die Datenbanksicherheit und die Datenbankoptimierung unter Oracle«](#)

[Tag 20, »Einsatz von Oracle Database Designer und Oracle Designer/2000 in der Anwendungsentwicklung«](#)

[Tag 21, »Oracle - Die nächste Generation«](#)

---

## Tag 1, »Die Welt relationaler Datenbanken erforschen«

### Test

1. Nennen Sie zwei Vorteile relationaler Datenbanken gegenüber Dateiverwaltungssystemen.

**Antwort:** Eine relationale Datenbank bietet deklarative Integrität. Der Begriff *deklarativ* bedeutet, daß die Integrität durch einfaches Deklarieren der Integritätsregeln erzwungen wird, wenn eine Tabelle erstellt oder verändert wird. Eine relationale Datenbank bietet außerdem eine ad-hoc-Abfragemöglichkeit, die ein Formulieren von komplexen Abfragen ohne Programmierung ermöglicht.

2. Nennen Sie drei Vorteile der Architektur für die Client-Server-Datenverarbeitung im Vergleich zu der Datenverarbeitungsarchitektur von Großrechnern.

**Antwort:** Vorteile der Client-Server-Architektur sind die Fähigkeit, unterschiedliche Client-Betriebssysteme zu unterstützen, unabhängig vom Netzwerkprotokoll und der Aufteilung von Prozessen zwischen Client und Server (der Client kontrolliert beispielsweise die

Benutzerschnittstelle, wohingegen der Server die Datenspeicherung und -verwaltung bietet).

3. Wie heißt das Middleware-Produkt von Oracle?

**Antwort:** Der Name des Oracle-Middleware-Produkts ist SQL\*Net.

4. Was ist ein »dicker« Client? Was ist ein »dünner« Client?

**Antwort:** Heutzutage wird eine traditionelle Client-Server-Architektur, in der ein Client-Rechner eine ziemlich große, ausführbare Datei, die notwendigen Bibliotheken und ein Middleware-Produkt beherbergt, als »fat«-Client bezeichnet. Ein »thin«-Client beschreibt einen Client-Rechner, in dem ein Web-Browser verwendet wird, um eine Anwendung auszuführen, entweder mittels HTML oder dem Herunterladen von Java-Applets nach Bedarf.

## Übung

Wie bereits erwähnt, wird in diesem Buch eine Beispieldatenbank eines kleinen Colleges verwendet. Denken Sie darüber nach, welche Art von Informationen Ihrer Meinung nach ein College speichern und abfragen möchte. Fragen Sie sich, welche Fragen ein Student, ein Professor oder ein Administrator an die Datenbank stellen könnte. Notieren Sie, wie Sie diese Informationen anordnen könnten. Sie werden am Tag 3, »Logischer Datenbankentwurf«, auf diese Informationen zurückkommen.

**Antwort:** Ein College würde bestimmt Informationen verwalten über die

- Studenten,
- Departments (Fachbereiche),
- Dozenten,
- Kurse,
- Einrichtungen sowie Gebäude und Ausrüstung.

Ein Student wird wahrscheinlich wissen wollen, welche Vorlesungen es gibt, wo sie gehalten werden und wer der Dozent ist; also die Kosten für jede Klasse.

Ein Dozent wird etwas über die Klassen wissen wollen, die ihm zum Unterrichten zugewiesen wurden. Ein Dozent muß in der Lage sein, jedem Studenten einer Klasse einen Benotungsgrad zuzuweisen.

Ein Administrator wird in der Lage sein wollen, jeder Klasse eine geeignete Räumlichkeit zur Verfügung zu stellen (in bezug auf die Zahl der Sitzplätze und der Laborressourcen). Ein Administrator wird sich außerdem die Möglichkeit wünschen, ein Department mit einem anderen auf verschiedene Arten vergleichen zu können - Lehrbelastung, Dienstalter der Dozenten oder die Durchschnittsnote in einer Klasse.

## Tag 2, »Richtlinien für die Entwicklung einer Oracle-Anwendung«

### Test

1. Richtig oder falsch? Wenn Sie ein Werkzeug für die Anwendungsentwicklung benutzen, welches eine objektorientierte Entwicklung unterstützt, ist die Chance größer, eine erfolgreiche Anwendung zu implementieren.

**Antwort:** Falsch. Ein objektorientiertes Entwicklungswerkzeug würde den Aufwand der Entwicklung und Wartung einer Anwendung reduzieren. Das bedeutet aber nicht, daß dies zu einer erfolgreichen Realisierung führt; es gibt noch viele andere Faktoren, die den Verlauf eines Projektes beeinflussen.

2. Nennen Sie drei Arten von Anforderungen, die bei der Entwicklung eines Systems erforderlich sind.

**Antwort:** Anforderungen an die Daten, die Funktionalität und die Leistung.

3. Richtig oder falsch? Sie sollten erst mit der Software-Entwicklung beginnen, wenn Ihnen die Anforderungen vollständig bekannt sind.

**Antwort:** Falsch. Sie *sollten* mit der Entwicklung von Software warten, bis Sie eine größere Anzahl von Anforderungen ermittelt haben. Wenn Sie jedoch warten, bis alle Anforderungen aufgestellt sind, warten Sie unter Umständen eine lange Zeit. Viele Anforderungen entstehen erst dann, wenn Endanwender einen Prototyp der Anwendung testen. Außerdem sind Anforderungen nicht statisch, sie verändern sich definitiv im Lauf der Zeit.

### Übungen

1. Welche Arten von Risiken könnten bei der Entwicklung einer Oracle-Anwendung auftreten?

2. Welche Faktoren könnten zu diesen Risikokategorien gehören?

3. Welche Schritte könnten Sie unternehmen, um diese Risiken zu mildern?

**Antwort:** Sie können die Risiken während der Entwicklung einer Oracle-Anwendung wie folgt kategorisieren:

- **Risiko Zeitplan:** Das Risiko, daß die Anwendung die Termine der

## Entwicklungsmeilensteine versäumt.

Ein offensichtlicher Grund, warum ein Projekt oftmals die Entwicklungsetappen nicht erreicht, ist die Tatsache, daß die Etappen von Beginn an unrealistisch waren. Oftmals ist der Druck auf das Projektmanagement schlicht zu hoch; einzelne Mitglieder werden gedrängt, die Zeit und die Ressourcen, welche für die Erledigung eines Arbeitsabschnitts notwendig sind, zu gering einzuschätzen.

Auch kann ein Projekt frühe Etappen erreichen, wie z.B. die Analyse der Anforderungen, scheitert aber bei der Erreichung von Etappen, welche die aktuelle Anwendung bietet. Was oftmals während solcher Projekte passiert, ist, daß die frühen Etappen erreicht zu werden scheinen, eine sorgfältige Untersuchung zeigt aber ungelöste Probleme. Aus diesem Grund ist es bei einem großen Projekt für den umsichtigen Kunden wichtig, eine unabhängige Überprüfung des Projekts für jede frühe Etappe einzusetzen.

- **Risiko Budget:** Das Risiko, daß die Anwendung das Budget, das dafür zur Verfügung steht, überschreitet.

Oftmals, wenn ein Projekt seinen Zeitplan überschreitet, geschieht dies auch in Bezug auf das Budget. Der Grund ist simpel: der größte Teil eines Projekt-Budgets ist üblicherweise für die Durchführung der Arbeiten gedacht. Wenn die Schätzung für den Arbeitsaufwand zur Absolvierung eines Arbeitsschrittes zu gering war, wird auch die korrespondierende Schätzung für das Budget zu gering ausfallen. Eine Ausnahme hierfür ist ein Projekt, für das die notwendigen Mitarbeiter verspätet eingebunden werden, in diesem Fall ist das Budget innerhalb seines Limits, aber der Zeitplan ist es nicht.

- **Risiko Technik:** Die technischen Risiken eines Projekts können viele Ursachen haben. Neue Produkte oder Technologien können ohne eine sorgfältige Analyse ausgewählt worden sein. Entwickler haben unter Umständen nicht die nötige Übung oder ausreichende Erfahrung, um eine zuverlässige Anwendung zu entwickeln. Die Projektentwickler könnten die Anforderungen an den Server-Rechner (zum Beispiel CPU-Geschwindigkeit, Speicheranforderungen) oder notwendige Netzwerkressourcen für die Produktionsumgebung unterschätzt haben. Jedes dieser Elemente sollte sorgfältig bedacht werden. Oftmals ist es Zeit- und Geldverschwendung, einen einfachen Benchmarktest durchzuführen, um die Performance der Anwendung in einer Produktionsumgebung zu charakterisieren.

## Tag 3, »Logischer Datenbankentwurf«

### Test

1. Richtig oder falsch? Wenn ein Attribut, das nicht Teil des Primärschlüssels ist, einen Fremdschlüssel darstellt, ist es unbedingt erforderlich; es läßt keinen Nullwert zu.

**Antwort:** Falsch. Manchmal ist ein Fremdschlüssel obligatorisch und manchmal nicht. Dies hängt von den Firmenregeln ab, die modelliert werden sollen.

2. Kann die Student-Schedule-Tabelle verwendet werden, wenn ein Student einen Kurs wiederholen muß? Warum bzw. warum nicht?

**Antwort:** Ja. Jede Klasse ist eindeutig identifiziert durch die Klassen-ID. Aber jedes erneute Anbieten desselben Kurses wird eine andere Klassen-ID haben. Nehmen Sie zum Beispiel an, ein Student belegt Biologie 101 während des Herbstsemesters 1996 und erreicht den Abschluß des Kurses nicht - dieser hat die Klassen-ID 109230. Im Frühjahr 1997 wiederholt er den Kurs - der jetzt die Klassen-ID 110330 hat. Weil zwei Klassen einmalige IDs haben, kann für beide Klassen eine Zeile in die Student-Schedule-Tabelle eingefügt werden.

3. Welche Eigenschaft einer relationalen Datenbank verhindert, daß eine Klasse aus der Class-Tabelle gelöscht wird, wenn in der Student-Schedule-Tabelle Zeilen existieren, welche die zu löschende Class ID enthalten?

**Antwort:** Referentielle Integrität.

4. Aus der Sicht der Entitätsbeziehung gibt es eine Beziehung zwischen der Class- und der Instructor-Tabelle. Handelt es sich hierbei um eine identifizierende oder nichtidentifizierende Beziehung?

Nichtidentifizierend. `Instructor ID` ist nicht Teil des Primärschlüssels in der Klassen-Tabelle.

5. Richtig oder falsch? Wenn Sie den entsprechenden Index für eine Tabelle erstellen, werden die Zeilen stets in aufsteigender Reihenfolge der indizierten Spalten abgerufen.

**Antwort:** Falsch. Es gibt keine festgelegte Reihenfolge für die Daten in einer Tabelle. Um zu garantieren, daß Zeilen in einer bestimmten Reihenfolge abgerufen werden, müssen Sie dies in einer SQL-Anweisung festlegen.

## Übungen

1. Angenommen, Sie möchten den Kursleiter identifizieren, der auch Bereichsleiter ist. Können Sie mindestens zwei Möglichkeiten nennen, dies herauszufinden? Wie lauten die Stärken und Schwächen der jeweiligen Ansätze?

**Antwort:** Es gibt offensichtlich zwei Möglichkeiten. Eine davon ist, zur Department-Entität ein Attribut hinzuzufügen, das den Dozenten identifiziert, der Leiter des Departments ist.



Natürlich würde dieses Attribut für die Dozenten-Entität ein Fremdschlüssel sein. Auf diese Weise würde die Department-Entität aus drei Attributen bestehen:

- Department ID
- Department Name
- Department Head

Die andere Möglichkeit ist, zur Dozenten-Entität ein Attribut hinzuzufügen, das festlegt, ob ein Dozent der Leiter eines Departments ist oder nicht. Sie können dieses Attribut ebenfalls Department Head nennen und ihm den Wert Y zuweisen, wenn der Dozent der Leiter eines Departments ist (der Wert kann auch Null sein, wenn der Dozent nicht der Leiter ist). Das Problem dieses Ansatzes ist, daß diese Information ein Charakteristikum eines Departments ist; aus diesem Grund macht es mehr Sinn, sie in die Department-Entität zu integrieren.

2. Die Student Schedule-Tabelle enthält zur Zeit sowohl die derzeitigen Klassen als auch die früheren Klassen, die ein Student besucht hat. Schlagen Sie einen alternativen Entwurf mit zwei Tabellen vor, wobei eine Tabelle den aktuellen Stundenplan und die zweite Tabelle die Klassen enthält, die der Student früher besucht hat. Nennen Sie alle Attribute in den zwei Tabellen. Welche Vorteile hat dieser Entwurf?

**Antwort:** Die gegenwärtige Definition der Student\_Schedule-Entität ist:

- Student-ID (Fremdschlüssel für die Student-Tabelle)
- Klassen-ID (Fremdschlüssel für die Class-Tabelle)
- Einstufung
- Datum der Zuweisung der Einstufung

Primärschlüssel: Student-ID, Class-ID

Als Alternative hierzu kann die Student\_Schedule-Entität in zwei Entitäten zerlegt werden: Current Student Schedule und Student Class History. Der gegenwärtige Stundenplan des Studenten wird in Current Student Schedule gespeichert, dies schließt die Einstufung des Studenten nicht mit ein. Sobald einem Studenten Einstufungen zugewiesen werden, würden der Student Class History die entsprechenden Datensätze hinzugefügt werden. Zu Beginn jedes Semesters werden die Inhalte der Current Student Schedule gelöscht.

Die Definition für die Current Student Schedule könnte diese sein:

- Student-ID (Fremdschlüssel für die Student-Tabelle)
- Klassen-ID (Fremdschlüssel für die Klassen-Tabelle)
- Die Definition für die Student Class History könnte diese sein:
- Student-ID (Fremdschlüssel für die Student-Tabelle)
- Klassen-ID (Fremdschlüssel für die Klassen-Tabelle)
- Einstufung
- Datum der Zuweisung der Einstufung

Für eine große Universität macht dieser Entwurf mehr Sinn. Nehmen Sie zum Beispiel an, eine Universität hat zwanzigtausend immatrikulierte Studenten. Nehmen Sie weiterhin an, daß jeder Student im Mittel 3,5 Klassen pro Semester belegt. Zu Beginn jedes Semesters würde die Current Student Schedule deshalb  $20.000 \times 3,5 = 70.000$  Datensätze enthalten. Am Ende jedes Semesters würden zur Student Class History weitere 70.000 Datensätze hinzugefügt. Wenn all diese Datensätze in einer einzelnen Entität, Student\_Schedule, gehalten sein würden, würde die Performance des Systems mit der Zeit zurückgehen. In vierzehn Jahren würde die Student\_Schedule-Entität nahezu eine Million Zeilen enthalten, was den Vorgang der Einschreibung für eine Klasse deutlich verlangsamen könnte.

## Tag 4, »Implementierung Ihres logischen Modells: physischer Datenbankentwurf«

### Test

1. Was ist an dieser Anweisung falsch?

```
CREATE TABLE new_table (
first_col number,
second_col date
third_col number default sysdate);
```

**Antwort:** Es gibt zwei Probleme mit dieser Anweisung. Zuerst einmal wird ein Komma am Ende der Definition für `second_col` benötigt. Zweitens ist `third_col` eine Zahl und kann keinen Standardwert - `sysdate` - aufnehmen, der vom Datentyp `Date` ist.

2. Beschreiben Sie eine SQL-Anweisung, welche die folgende Oracle-Fehlermeldung verursachen könnte.

```
ORA-02266: unique/primary keys in table referenced by enabled
foreign keys
```

**Antwort:** Sie werden diese Fehlermeldung sehen, wenn Sie versuchen, eine Tabelle zu löschen, deren Primärschlüssel vom Fremdschlüssel einer anderen Tabelle referenziert wird.

3. Worin besteht der Unterschied zwischen den `check`-Regeln für eine Spalte und für eine Tabelle?

**Antwort:** Eine Spalten-Integritätsregel vom Typ `CHECK` wird auf Spaltenebene definiert und kann keine anderen Spalten in der Tabelle referenzieren. Eine Tabellen-Integritätsregel vom Typ `CHECK` wird auf dem Tabellen-Level definiert und kann jede beliebige Spalte der Tabelle

referenzieren.

## Übung

Die Instructor-Tabelle besitzt eine Spalte namens Position. In dem derzeitigen Tabellenentwurf gibt es eine check-Regel für die Spalte Position, welche den Wert auf `assistant professor`, `associate professor` und `full professor` begrenzt. Ändern Sie den Datenbankentwurf so, daß eine zusätzliche Tabelle, `INSTRUCTOR_POSITION`, verwendet wird, um zulässige Werte für die Position des Kursleiters anzugeben.

**Antwort:** Erstellen Sie eine Tabelle namens `Instructor_Position`:

```
create table Instructor_Position
(Position varchar2(25),
constraint PK_Instructor_Position Primary Key (Position));
```

Definieren Sie die Dozenten-Tabelle wie folgt neu:

```
create table Instructor
(Instructor_ID varchar2(20),
Department_ID varchar2(20)
constraint NN_Instructor_Dept_ID NOT NULL,
Last_Name varchar2(25)
constraint NN_Instructor_Last_Name NOT NULL,
First_Name varchar2(25),
MI varchar2(1),
Position varchar2(25),
Telephone varchar2(10),
Fax varchar2(10),
Email varchar2(100),
constraint PK_Instructor Primary Key (Instructor_ID),
constraint FK_Instructor_Department_ID
Foreign Key (Department_ID) references Department (Department_ID),
constraint FK_Instructor_Position
Foreign Key (Position) references Instructor_Position (Position));
```

## Tag 5, »Einführung in SQL«

## Test

1. Richtig oder falsch? Sie müssen eine Spalte in die `select`-Liste aufnehmen, wenn Sie die von der `select`-Anweisung für diese Spalte zurückgegebenen Zeilen sortieren möchten.

**Antwort:** Falsch. Sie können eine Spalte in der `ORDER BY`-Bedingung festlegen, die sich nicht in der Auswahlliste befindet.

2. Was ist an dieser Anweisung falsch?

```
select First_Name
from Student
order by Last_Name
where
Last_Name like '%IN%';
```

**Antwort:** Die `ORDER BY`-Bedingung steht vor der `WHERE`-Bedingung.

3. Richtig oder falsch? Eine Spalte muß zuerst indiziert werden, bevor sie in der `order by`-Klausel angegeben werden kann.

**Antwort:** Falsch. Oracle und andere relationale Datenbanken benötigen keine Indizierung einer Spalte, bevor diese für die Sortierung der Reihen benutzt werden kann, die von einer Tabelle abgerufen werden.

## Übung

Konstruieren Sie unter Verwendung der in dieser Lektion behandelten `Course`-Tabelle eine `select`-Anweisung, welche `Department ID`, `Course ID` und `Course Title` sortiert nach `Department ID` und `Course ID` zurückgibt, und zwar für alle Kurse, deren Beschreibung das Wort `introduc` enthält, unabhängig von der Groß- und Kleinschreibung.

**Antwort:** Hier ist die `SELECT`-Anweisung, welche diese Aufgabe erfüllt. Beachten Sie, daß die Funktion `lower` benutzt wird, um sämtliche Kursbeschreibungen in Kleinschrift umzuwandeln, so daß ein einheitlicher Vergleich mit `introduc` durchgeführt werden kann.

```
SQL> select Department_ID, Course_ID, Title
2 from Course
3 where
4 lower(Description) like '%introduc%'
5 order by Department_ID, Course_ID;
```

```
DEPARTMENT_ID COURS TITLE
-----
```

```
ANTHRO 101 INTRO TO ANTHROPOLOGY
BIO 101 INTRO TO BIOLOGY
ECON 101 INTRO TO ECONOMICS
ENG 101 INTRO TO STRUCTURES
ENG 102 INTRO TO CIRCUIT THEORY
ENG 103 INTRO TO DYNAMICS
ENGL 101 INTRO TO ENGLISH LIT
ENGL 193 SEMINAR ON THEME ANALYSIS
MATH 101 GENERAL CALCULUS
MATH 189 NUMBER THEORY
PHILO 101 INTRO TO PHILOSOPHY
PSYCH 101 INTRO TO PSYCHOLOGY
```

12 rows selected

## Tag 6, »Ändern von Daten mit Hilfe von SQL«

### Test

1. Konstruieren Sie eine SQL-Anweisung, die einen Kurs mit den folgenden Eigenschaften hinzufügt: Department\_ID = BIO, Course\_ID = 137, Title = INSECT BEHAVIOR, Description = In-depth study of insect societies and their behavior patterns, Units = 3, keine zusätzlichen Gebühren.

**Antwort:** Hier ist die INSERT-Anweisung:

```
insert into Course
(Department_ID, Course_ID, Title, Description, Units)
values
('BIO', '137', 'INSECT BEHAVIOR',
'In-depth study of insect societies and their behavior patterns',
3);
```

2. Konstruieren Sie eine SQL-Anweisung, die zusätzliche Gebühren für alle Kurse in der Abteilung Philosophie einträgt.

**Antwort:** Hier ist die UPDATE-Anweisung:

```
update Course
set Additional_Fees = 50
where
Department_ID = 'PHILO';
```

3. Konstruieren Sie eine SQL-Anweisung, die eine geplante Klasse löscht, wenn diese von der Abteilung für Englisch angeboten wird oder in Flugle Hall abgehalten werden soll.

**Antwort:** Hier ist die DELETE-Anweisung:

```
delete from Class
where
Department_ID = 'ENG' or
Class_Building = 'FLUGLE HALL';
```

## Übung

Einige der Lehrer in Flugle College haben sich entschieden, eine neue Abteilung namens Integrated Studies einzurichten. Als Folge davon werden die Abteilungen für Englisch, Geschichte und Philosophie zusammengelegt, um die Abteilung für Integrated Studies zu bilden. Die Department ID für diese neue Abteilung wird `intstd` sein. Erzeugen Sie in der Datenbank eine Abteilung für Integrated Studies (ohne die existierenden Abteilungen zu löschen). Verändern Sie zudem die Tabelle Instructor, so daß die Lehrer in den Abteilungen für Englisch, Geschichte und Philosophie der Abteilung für Integrated Studies zugewiesen werden.

**Antwort:** Sie können mit dieser Abfrage ermitteln, daß sechs Dozenten mit den Departments English, History und Philosophy verbunden sind:

```
SQL> select Instructor_ID
2 from Instructor
3 where
4 Department_ID in ('ENGL', 'HIST', 'PHILO');
```

```
INSTRUCTOR_ID
```

```
-----
```

```
G331
```

```
L391
```

```
E491
```

```
T149
```

```
D944
```

```
B331
```

```
6 rows selected.
```

Erstellen Sie zuerst das neue Department:

```
SQL> insert into Department
2 (Department_ID, Department_Name)
```

```

3 values
4 ('INTSTD', 'Integrated Studies');

1 row created.

```

Erneuern Sie als nächstes die Dozenten-Tabelle dahingehend, daß jeder Dozent der English-, History- und Philosophy-Departments jetzt mit Integrated Studies verbunden wird:

```

SQL> update Instructor
2 set
3 Department_ID = 'INTSTD'
4 where
5 Department_ID in ('ENGL', 'HIST', 'PHILO');

6 rows updated

```

Wenn Sie die Dozenten-Tabelle abfragen, werden Sie sehen, daß dieselben sechs Dozenten jetzt zum Department Integrated Studies gehören:

```

SQL> select Instructor_ID
2 from Instructor
3 where
4 Department_ID = 'INTSTD';

```

```

INSTRUCTOR_ID
-----

```

```

G331
L391
E491
T149
D944
B331

```

```

6 rows selected.

```

Führen Sie abschließend ein ROLLBACK durch, so daß die Änderungen nicht von Dauer sind.

```

SQL> rollback;

```

```

Rollback complete.

```

## Tag 7, »Die in SQL integrierten Funktionen ausnutzen«

## Test

1. Erstellen Sie eine SQL-Anweisung, die die Zeilen der Tabelle Instructor wie im folgenden Beispiel gezeigt liefert:

Professor Parker

**Antwort:** Die SELECT-Anweisung ist

```
SQL> select initcap(position || ' ' || last_name)
2 from instructor;
```

2. Erstellen Sie eine SQL-Anweisung, die den Instructor liefert, dessen Nachname der erste in der alphabetischen Reihenfolge ist.

**Antwort:** Die SELECT-Anweisung ist

```
SQL> select min(Last_Name)
2 from Instructor;
```

```
Min(LAST_NAME)
-----
ANGELO
```

## Übung

Erzeugen Sie unter Verwendung der Anweisung `create table <xyz> as` eine Tabelle namens `NEW_CLASS`, die auf einem Verbund der drei Tabellen `Class`, `Schedule_Type` und `Schedule_Type_Details` basiert und die folgenden Spalten enthält:

- Class\_ID
- Department\_ID
- Course\_ID
- Den ausgeschriebenen Wochentag
- Die ausgeschriebene Zeit (z.B. 11:00 a.m.)

**Antwort:**

```
SQL> create table New_Class as
2 select Class_ID, Department_ID, Course_ID,
3 decode(day,1, 'Sunday',2, 'Monday',3, 'Tuesday',4, 'Wednesday',
4 5, 'Thursday',6, 'Friday',7, 'Saturday') Day_of_Week,
```



```

5 to_char(Starting_Time,'HH:MI PM') Starting_Time
6 from Class, Schedule_Type_Details
7 where
8 Class.Schedule_ID = Schedule_Type_Details.Schedule_ID;

```

Table created.

```

SQL> select Class_ID, Department_ID, Course_ID, Day_of_Week,
Starting_Time
2 from New_Class
3 order by Class_ID;

```

```

CLASS_ID DEPARTMENT_ID COURS DAY_OF_WE STARTING_TIME
-----

```

```

103400 PSYCH 183 Monday 01:00 PM
103400 PSYCH 183 Wednesday 01:00 PM
103400 PSYCH 183 Friday 01:00 PM
103600 MATH 50 Monday 03:00 PM
103600 MATH 50 Friday 03:00 PM
103600 MATH 50 Wednesday 03:00 PM
104200 PHILO 198 Friday 09:00 AM
104500 HIST 184 Monday 03:00 PM
104500 HIST 184 Wednesday 03:00 PM
104500 HIST 184 Friday 03:00 PM
108300 ENGL 101 Monday 10:00 AM
108300 ENGL 101 Wednesday 10:00 AM
108300 ENGL 101 Friday 10:00 AM
108400 HIST 115 Tuesday 09:00 AM
108400 HIST 115 Thursday 09:00 AM
108600 PSYCH 183 Monday 10:00 AM
108600 PSYCH 183 Wednesday 10:00 AM
108600 PSYCH 183 Friday 10:00 AM
109100 BIO 101 Monday 03:00 PM
109100 BIO 101 Wednesday 03:00 PM
109100 BIO 101 Friday 03:00 PM
110300 ENG 199 Friday 09:00 AM
120200 ECON 199 Tuesday 09:00 AM
120200 ECON 199 Thursday 09:00 AM

```

24 rows selected.

## Tag 8, »Bilden fortgeschrittener Abfragen mit SQL«

## Test

1. Erstellen Sie eine SQL-Anweisung, die den Nachnamen eines Lehrers liefert, der einen Kurs mit zusätzlichen Gebühren über DM 50,- unterrichtet.

**Antwort:** Hier ist ein Drei-Tabellen-Join, der die Information zurückgibt:

```
SQL> select Last_Name
2 from Class CL, Instructor I, Course CO
3 where
4 CL.Instructor_ID = I.Instructor_ID and
5 CL.Department_ID = CO.Department_ID and
6 CL.Course_ID = CO.Course_ID and
7 Additional_Fees > 50;
```

LAST\_NAME

-----

WEISS

2. Erstellen Sie eine SQL-Anweisung, die eine Liste von Städten, in denen Studenten wohnen, sowie die Zahl der Studenten in jeder Stadt liefert.

**Antwort:** Hier ist die SELECT-Anweisung, welche die gewünschten Resultate liefert:

```
SQL> select City, count(*)
2 from Student
3 group by City;
```

CITY COUNT( \*)

-----

DOVER 14

SPRINGFIELD 17

3. Erstellen Sie eine Datensicht, die jede Klasse auflistet - ihre Class\_ID, Department\_ID und Kurs\_ID -, die am Montag zusammentrifft.

**Antwort:** Hier ist die CREATE VIEW-Anweisung, welche die geforderte Aktion ausführt. Bevor Sie jetzt versuchen, die Datensicht zu erzeugen, sollten Sie experimentieren, bis Sie die SELECT-Anweisung definiert haben, auf der die Datenansicht basieren wird:

```
SQL> select Class_ID, Department_ID, Course_ID
2 from Class CL, Schedule_Type ST, Schedule_Type_Details STD
3 where
4 CL.Schedule_ID = ST.Schedule_ID and
5 ST.Schedule_ID = STD.Schedule_ID and
```

**6 STD.Day = 2;**

CLASS\_ID DEPARTMENT\_ID COURS

```
-----
104500 HIST 184
109100 BIO 101
108300 ENGL 101
108600 PSYCH 183
103400 PSYCH 183
103600 MATH 50
```

6 rows selected.

Jetzt können Sie die Datenansicht erzeugen:

```
SQL> create view Classes_on_Monday as
2 select Class_ID, Department_ID, Course_ID
3 from Class CL, Schedule_Type ST, Schedule_Type_Details STD
4 where
5 CL.Schedule_ID = ST.Schedule_ID and
6 ST.Schedule_ID = STD.Schedule_ID and
7 STD.Day = 2;
```

View created.

```
SQL> select Class_ID, Department_ID, Course_ID
2 from Classes_on_Monday;
```

CLASS\_ID DEPARTMENT\_ID COURS

```
-----
104500 HIST 184
109100 BIO 101
108300 ENGL 101
108600 PSYCH 183
103400 PSYCH 183
103600 MATH 50
```

6 rows selected.

## Übung

Die Anzahl der Lehrer ist 18, die Anzahl der angebotenen Klassen ist 10. Trotzdem ist die Zahl der eindeutigen Instructor\_IDs in der Tabelle Class gleich 9. Geben Sie unter Verwendung dieser Tabelle und von SQL eine vollständige Erklärung.

**Antwort:** SQL> **select count(\*) from Class;**

COUNT(\*)

-----

10

Und es gibt neun Dozenten für diese Klassen:

**SQL> select distinct Instructor\_ID from Class;**

INSTRUCTOR\_ID

-----

D944

E405

E491

G331

J505

R983

S131

T149

W490

9 rows selected.

Lassen Sie uns sehen, ob einer der Dozenten mehr als eine Klasse unterrichtet:

**SQL> select Instructor\_ID from Class**

**2 having count(\*) > 1**

**3 group by Instructor\_ID;**

INSTRUCTOR\_ID

-----

J505

Dozent J505, bei dem es sich um Jerrold Jason handelt, unterrichtet mehr als eine Klasse:

**SQL> select Class\_ID from Class**

**2 where Instructor\_ID = 'J505';**

CLASS\_ID

-----

108600

103400

## Tag 9, »Programmierung einer Oracle-Datenbank mit PL/SQL«

### Test

1. Nennen Sie die drei Teile eines PL/SQL-Unterprogramms.

**Antwort:** Die drei Abschnitte eines PL/SQL-Unterprogramms sind der Deklarations-, der ausführbare und der Zusatz-Teil.

2. Richtig oder falsch? Eine PL/SQL-Variable, die einen Spaltenwert speichert, muß denselben Namen wie die Spalte haben.

**Antwort:** Falsch. Eine PL/SQL-Variable, die einen Spaltenwert speichert, kann jeden gültigen Namen besitzen.

3. Warum ist es eine gute Angewohnheit, bei der Deklaration von Variablen %type zu verwenden?

**Antwort:** Es ist sinnvoll, bei der Variablendeklaration %TYPE zu benutzen, weil dies den benötigten Wartungsaufwand für den PL/SQL-Code reduziert. Wenn Sie eine Variable ohne die Verwendung von %TYPE deklarieren und die korrespondierende Spaltendefinition sich ändert, müssen Sie die Variablendeklaration anpassen. Wenn Sie %TYPE benutzen, müssen Sie die Variablendeklaration nicht ändern.

### Übung

Konstruieren Sie einen anonymen Block, der eine Prozedur deklariert, die für einen Kurs eine zusätzliche Gebühr von DM 50,- festlegt, wenn es keine laufenden Gebühren gibt. Die Prozedur sollte nur ein Argument haben - Department\_ID - und sollte diese Aktion nur ausführen, wenn der Wert von Department\_ID angegeben wird.

**Antwort:** Lassen Sie uns »bisher keine Zusatzgebühren« so interpretieren, daß diese entweder den Wert 0 oder Null haben. Betrachten wir eine mögliche Lösung:

Wie Sie in der folgenden Abfrage sehen können, gibt es im mathematischen Department drei Kurse ohne Zusatzgebühren: die Kurs-IDs 50, 101, 189.

```
SQL> select Course_ID, Additional_Fees
2 from Course
3 where
4 Department_ID = 'MATH';
```

COURS ADDITIONAL\_FEES

-----

50 0  
101 0  
189 0  
51 10

Hier ist der anonyme Block, der eine Prozedur (siehe Zeile 2) enthält, die in Zeile 13 aufgerufen wird. Wie festgelegt wurde, enthält die Prozedur ein einzelnes Argument: Department\_ID.

```
SQL> declare
2
2 procedure Set_Default_Course_Fees (arg_Department_ID varchar2) is
3
3 begin
4
4 update Course
5 set Additional_Fees = 50
6 where
7 Department_ID = arg_Department_ID and
8 Additional_Fees is null or
9 Additional_Fees = 0;
10
10 end;
11
11 - Main block.
12
12 begin
13
13 Set_Default_Course_Fees ('MATH');
14
14 end;
15 /
```

PL/SQL procedure successfully completed.

Sie können aus der folgenden Abfrage ersehen, daß die Prozedur die Zusatzgebühren für die drei Kurse auf DM 50,- gesetzt hat:

```
SQL> select Course_ID, Additional_Fees
2 from Course
3 where
4 Department_ID = 'MATH';
```

COURS ADDITIONAL\_FEES

-----

50 50  
101 50  
189 50  
51 10

## Tag 10, »Programmentwicklung mit PL/SQL«

### Test

1. Richtig oder falsch? Eine gespeicherte Prozedur kann eine gespeicherte Funktion aufrufen, eine gespeicherte Funktion jedoch keine gespeicherte Prozedur.

**Antwort:** Falsch. Eine gespeicherte Prozedur kann andere Prozeduren oder Funktionen aufrufen. Eine gespeicherte Funktion kann ebenfalls andere Prozeduren und Funktionen aufrufen.

2. Nennen Sie drei Gründe für das Speichern von Prozeduren, Funktionen und Paketen in einer Anwendung.

**Antwort:** Effizienz, Wiederverwendbarkeit und Portabilität.

3. Welche Werkzeuge können verwendet werden, um PL/SQL-Unterprogramme zu erstellen?

**Antwort:** SQL\*Plus, SQL Worksheet, Procedure Builder und andere Werkzeuge von Drittanbietern. Procedure Builder macht die Arbeit der PL/SQL-Entwicklung viel leichter als die Verwendung von SQL\*Plus oder SQL Worksheet.

4. Wenn »x > 32« einer PL/SQL-Variablen zugewiesen wird, wie ist dann der Datentyp der Variablen?

**Antwort:** Boolean. Einer Boolean-Variable kann nur ein Boolean-Ausdruck zugewiesen werden.

### Übungen

1. Erzeugen Sie einen anonymen PL/SQL-Block, der die Prozedur `Assign_Grade` aus dem Flugle-Paket aufruft und Anna Anastasia für Biologie 101 ein B zuweist.

**Antwort:** Wenn Sie die Student- und Klassen-Tabellen abfragen, werden Sie folgendes finden:

- Die Student\_ID von Anna Anastatia ist 10231324.
- Die Klassen\_ID für Biologie 101 ist 109100.

Wie Sie in der folgenden Abfrage sehen können, hat Anna bisher noch keinen Abschluß in Biologie 101 erreicht:

```
SQL> select * from Student_Schedule;
```

```
STUDENT_ID CLASS_ID GR DATE_GRAD
-----
10231311 104200 B 02-JUN-97
10231311 104500 B- 03-JUN-97
10231324 109100
10231311 109100
```

Hier ist ein anonymer Block, der die Prozedur Assign\_Grade im Flugle Paket aufrufen wird:

```
SQL> set serveroutput on
SQL>
SQL> declare
2
2 Status number;
3
3 - Main block.
4
4 begin
5
5 dbms_output.enable;
6
6 Flugle.Assign_Grade('10231324','109100','B', status);
7
7 dbms_output.put_line('status: ' || to_char(status));
8
8 end;
9 /
status: 0
```

```
PL/SQL procedure successfully completed.
```

Noch einmal: Wenn Sie die Student\_Schedule-Tabelle einsehen, werden Sie feststellen, daß Anna jetzt für die Klasse ein B zugewiesen bekommen hat.

```
SQL> select * from Student_Schedule;
```



STUDENT\_ID CLASS\_ID GR DATE\_GRAD

```
-----
10231311 104200 B 02-JUN-97
10231311 104500 B- 03-JUN-97
10231324 109100 B 04-JUN-97
10231311 109100
```

2. Ein Student Jackson Smythe hat eine gespeicherte Funktion namens `Change_My_Grade` mit zwei Argumenten erstellt: `Student ID` und `Class ID`. Die Funktion ändert die Note für jeden beliebigen Studenten und eine beliebige Klasse auf A+. Schreiben Sie diese Funktion.

Hier ist die Funktion, welche diese niederträchtige Tat ausführt:

```
SQL> create or replace function
2 change_My_Grade(arg_student_ID IN varchar2,
3 arg_class_ID IN varchar2)
4 return number is
5
5 counter number;
6 status number;
7
7 normal CONSTANT number := 0;
8 unsuccessful CONSTANT number := -1;
9
9 not_registered exception;
10
10 begin
11
11 status := normal;
12
12 - Determine if the student is registered for this class.
13
13 select count(*) into counter
14
14 from student_schedule
15 where
16 student_id = arg_student_id and
17 class_id = arg_class_id;
18
18 if counter = 0 then
19 -
20 - The student is not taking this class.
21 -
22 raise not_registered;
23 status := unsuccessful;
```

```

24 else
25 -
26 - Assign the grade for this class.
27 -
28 update student_schedule
29 set
30 grade = 'A+',
31 date_grade_assigned = sysdate
32 where
33 student_id = arg_student_id and
34 class_id = arg_class_id;
35 end if;
36
36 return status;
37
37 exception
38 when not_registered then
39 raise_application_error (-21003, 'Student not registered for
class');
40 when others then
41 null;
42
42 end;
43 /

```

Function created.

Um die Funktion zu testen, erstellen Sie einen anonymen Block, der Change\_My\_Grade für die Note von Paul Fernandez in History 184 aufruft.

```

SQL> declare
2
2 Status number;
3
3 - Main block.
4
4 begin
5
5 dbms_output.enable;
6
6 status := Change_My_Grade('10231311', '104500');
7
7 dbms_output.put_line('status: ' || to_char(status));
8
8 end;
9 /

```

status: 0

PL/SQL procedure successfully completed.

Wie Sie bei der folgenden Abfrage sehen können, arbeitet die Funktion wie angekündigt:

SQL> select \* from Student\_Schedule;

STUDENT\_ID CLASS\_ID GR DATE\_GRAD

```
-----
10231311 104200 B 02-JUN-97
10231311 104500 A+ 04-JUN-97
10231324 109100
10231311 109100
```

## Tag 11, »Weitere Programmier Techniken für PL/SQL«

### Test

1. Wie werden SQLCODE und SQLERRM in PL/SQL-Unterprogrammen verwendet?

**Antwort:** SQLCODE gibt den Oracle-Fehler zurück, der von der letzten SQL-Operation herrührt. Ein Fehlercode von 0 zeigt den normalen Abschluß der Operation an. SQLERRM gibt den Text der Oracle-Fehlermeldung zurück, die mit dem Wert in SQLCODE korrespondiert.

2. Richtig oder falsch? Ein PL/SQL-Unterprogramm kann gleichzeitig mehrere Cursor geöffnet haben.

**Antwort:** Richtig. In einem PL/SQL-Unterprogramm gibt es in bezug auf die Öffnung mehr als eines Cursors zur selben Zeit keine Beschränkung.

3. Welches sind die Vorteile bei der Verwendung von Datenbank-Triggern?

**Antwort:** Trigger bieten mehrere Möglichkeiten. Ein Trigger kann komplexe Gültigkeitsprüfungen ausführen, wenn eine Zeile hinzugefügt oder modifiziert wird. Ein Trigger kann einen Spaltenwert ändern, bevor er in einer Tabelle gespeichert wird. Ein Trigger in einer Tabelle kann benutzt werden, um die Inhalte einer anderen Tabelle zu ändern.

4. Welchen Datenbank-Trigger würden Sie verwenden, um den in einer Spalte zu speichernden Wert zu verändern, wenn der Tabelle eine neue Zeile hinzugefügt wird?

**Antwort:** Ein Before-Insert-Trigger, der bei jeder Zeile auslösen wird.

## Übungen

1. Erstellen Sie eine gespeicherte Funktion `Teaching_Load` mit einem einzigen Argument - einer Abteilungsnummer (`Department_ID`). `Teaching_Load` soll die durchschnittliche Anzahl an Stunden zurückgeben, die Ausbilder in den entsprechenden Abteilungen zur Zeit unterrichten.

**Antwort:** Zuerst müssen Sie die gespeicherte Funktion erzeugen:

```
SQL> create or replace function Teaching_Load(arg_Department_ID IN
varchar2)
2
2 return number is
3
3 number_of_instructors number;
4 number_of_classes number;
5 avg_teaching_load number;
6
6 begin
7
7 - Get the number of instructors for this department.
8
8 select count(*) into number_of_instructors
9 from Instructor
10 where
11 Department_ID = arg_Department_ID;
12
12
12 - Get the number of classes for this department.
13
13 select count(*) into number_of_classes
14 from Class
15 where
16 Department_ID = arg_Department_ID;
17
17 avg_teaching_load := number_of_classes / number_of_instructors;
18
18 return avg_teaching_load;
19
19 exception
20 when others then
21 null;
22
22 end;
```

23 /

Function created.

Um die Funktion zu testen, können Sie sie in einer SELECT-Anweisung aufrufen, welche die DUAL-Tabelle referenziert. Wie Sie sehen können, unterrichten die Dozenten im Department Biologie im Durchschnitt 0,5 Klassen:

```
SQL> select Teaching_Load('BIO') from dual;
```

```
TEACHING_LOAD( 'BIO' )
-----
.5
```

2. Erstellen Sie eine gespeicherte Prozedur `Suitable_Locations` mit einem einzigen Argument - Anzahl der Sitzplätze. `Suitable_Locations` soll die ersten drei Gebäude und Räume ausdrucken, die die angegebene Anzahl Sitzplätze überschreiten.

**Antwort:** Hier ist eine gespeicherte Prozedur, welche die Anforderungen erfüllt:

```
SQL> create or replace procedure Suitable_Locations
2 (arg_Seating_Capacity IN number) is
3
4 cursor Get_Locations is
5 select class_building, class_room, seating_capacity
6 from Class_Location
7 where
8 Seating_Capacity >= arg_Seating_Capacity
9 order by Seating_Capacity desc;
10
11 begin
12
13 -
14
15 dbms_output.enable;
16
17 for Get_Locations_Rec in Get_Locations loop
18
19 exit when Get_Locations%rowcount > 3;
20 dbms_output.put_line('Building: ' ||
21 Get_Locations_Rec.Class_Building ||
22 ' Room: ' || Get_Locations.Rec.Class_Room || ' Capacity: ' ||
23 to_char(Get_Locations_Rec.Seating_Capacity));
24
25 end loop;
```

```

18 exception
19 when others then
20 null;
21
21 end;
22 /

```

Procedure created.

Um die gespeicherte Prozedur zu testen, können Sie einen anonymen Block benutzen, der diese aufruft:

```

SQL> declare
2
2 Status number;
3
3 - Main block.
4
4 begin
5
5 dbms_output.enable;
6
6 Suitable_Locations (30);
7
7 end;
8 /
Building: FLUGLE HALL Room: 100 Capacity: 200
Building: FLUGLE HALL Room: 150 Capacity: 120
Building: NARROW HALL Room: 200 Capacity: 90

```

PL/SQL procedure successfully completed.

3. Schreiben Sie einen Before-Delete-Trigger für die Tabelle Instructor, der eine Exception auslöst, wenn ein Ausbilder laut Plan eine Klasse unterrichtet.

**Antwort:** Hier ist das Skript für die Erstellung des Triggers. In Zeile 17 wird eine Abfrage benutzt, um zu ermitteln, wie viele Klassen ein Dozent unterrichtet; dieser Wert wird in `counter` gespeichert. In Zeile 11 wird `counter` ausgewertet; falls er größer als Null ist, löst der Trigger einen Anwendungsfehler aus, was die Löschung des Dozenten verhindert:

```

SQL> create or replace trigger Instructor_BD
2 before delete on Instructor
3 for each row
4
4 declare
5

```

```

5 counter number;
6
6 begin
7
7 select count(*) into counter
8 from Class
9 where
10 Instructor_ID = :old.Instructor_ID;
11
11 if counter > 0 then
12 raise_application_error (-20800,
13 'Instructor is scheduled to teach a class');
14 end if;
15
15
15 end;
16 /

```

Trigger created.

Um den Trigger zu testen, können Sie versuchen, einen Dozenten zu löschen, der gegenwärtig eine Klasse unterrichtet. Wie Sie sehen können, löst der Trigger den Anwendungsfehler - 20800 - aus, was die erfolgreiche Ausführung der DELETE-Anweisung verhindert.

```

SQL> delete from Instructor
2 where
3 Instructor_ID = 'E405';
delete from Instructor
*
Error at line 1:
ORA-20800: Instructor is scheduled to teach a class
ORA-06512: at line 9
ORA-04088: error during execution of trigger 'FLUGLE.INSTRUCTOR_BD'

```

Wenn Sie jedoch versuchen, einen Dozenten zu löschen, der keine Klasse unterrichtet, wird der Trigger den Anwendungsfehler nicht auslösen:

```

SQL> delete from Instructor
2 where
3 INSTRUCTOR_ID = 'P331';

1 row deleted.

```

## Tag 12, »Developer/2000: Einführung in Oracle Forms«

## Test

1. Richtig oder falsch? Oracle Forms Runtime führt nur eine `.fmb`-Datei aus.

**Antwort:** Falsch. Das Laufzeitmodul von Oracle Forms führt nur `.fmx`-Dateien aus, keine `.fmb`-Dateien.

2. Welches sind die drei Modultypen, die mit dem Forms-Designer erzeugt werden können?

**Antwort:** Die drei Module sind Formulare, Menüs und Bibliotheken.

3. Richtig oder falsch? Der Forms Designer kann ein Formular nur in einer Oracle-Datenbank speichern, wenn dieses einen Basistabellenblock enthält.

**Antwort:** Falsch. Der Forms-Designer kann ein Formular entweder in einer Datenbank oder einem Dateisystem speichern, unabhängig vom Inhalt des Formulars.

4. Richtig oder falsch? Sie können einen Block erzeugen, der auf einer Tabelle basiert, auf die ein anderer Oracle-Benutzer die Rechte besitzt.

**Antwort:** Richtig. Sie können einen Block erstellen, der auf den Inhalten einer Tabelle basiert, die sich im Besitz eines anderen Oracle-Benutzers befindet, solange Ihnen die entsprechenden Berechtigungen für die Tabelle gewährt wurden.

## Übung

Erzeugen Sie für die Tabelle `Instructor` ein Formular. Geben Sie eine `ORDER BY`-Klausel ein, damit die abgerufenen Datensätze nach der Abteilungs-ID, dem Nachnamen, Vornamen und Anfangsbuchstaben des Mittelnamens des Unterrichtenden sortiert werden. Testen Sie das Formular.

**Antwort:** Sehen Sie sich bitte die `Instructor.fmb`-Datei im Verzeichnis `Day 13` der CD-ROM an.

## Tag 13, »Entwicklung einer Benutzeroberfläche mit Hilfe von Oracle Forms«



## Test

1. Richtig oder falsch? Ein Master-Block kann nicht auf einer Datensicht basieren, es muß eine Tabelle sein.

**Antwort:** Falsch. Ein Master-Block *kann* auf einer Datenansicht basieren.

2. Richtig oder falsch? Man kann das Erscheinungsbild eines Elements mit dem Layout-Editor oder durch Ändern der Eigenschaften des Elements verändern.

**Antwort:** Richtig. Sie können die Anzeigecharakteristiken eines Datenelements ändern - zum Beispiel seine Position oder die Schriftart -, indem Sie das Element mit dem Object Navigator wählen und mit der rechten Maustaste das Attributblatt öffnen.

3. Wie kann die Reihenfolge der Aktivierung in einer Gruppe von Elementen geändert werden?

**Antwort:** Sie können die Reihenfolge der Karteikarten einer Gruppe von Elementen ändern, indem Sie ihre Reihenfolge im Object Navigator ändern.

## Übungen

1. Erstellen Sie ein Master-/Detail-Fenster, um Schedule\_Type und Schedule\_Type\_Details einzugeben.

**Antwort:** Sehen Sie sich bitte die `Schedule_Type.fmb`-Datei im Verzeichnis Day 14 auf der CD-ROM an.

2. Erstellen Sie im Formular Department\_Class eine LOV für Schedule\_ID, die die Benutzereingabe überprüft.

**Antwort:** Sehen Sie sich bitte die `Department_Class.fmb`-Datei im Verzeichnis Tag 14 auf der CD-ROM an.

## Tag 14, »Developer/2000: Anwendungsentwicklung mit Hilfe von Oracle Forms«

## Test

1. Richtig oder falsch? Ein Trigger auf Elementebene eines Blocks kann nicht auf Elemente eines anderen Blocks verweisen.

**Antwort:** Falsch. Es gibt keine Beschränkungen für die Fähigkeit eines Item-Level-Triggers, Elemente in einem anderen Block zu referenzieren.

2. Wie wird in einem Trigger auf Blockelemente Bezug genommen?

**Antwort:** Um einen Block-Trigger in einem Block zu referenzieren, verwenden Sie die Schreibweise `:block-name.item-name`.

3. Welche Fehlermeldung sollte bei einer ungültigen Benutzereingabe hervorgerufen werden, wenn Sie einen Trigger auf Elementebene zur Gültigkeitsüberprüfung von Benutzereingaben erstellen?

**Antwort:** `FORM_TRIGGER_FAILURE`.

## Übungen

1. Ändern Sie das Formular `Instructor` dahingehend ab, daß es die gleiche Breite und Höhe wie das `MDI_Frame`-Formular besitzt.

**Antwort:** Sehen Sie sich bitte die `Instructor.fmb`-Datei im Verzeichnis Tag 15 auf der CD-ROM an.

2. Erstellen Sie ein Element für das Browse-Menü, das ein Formular aufruft, in dem das Formular `Instructor` sortiert nach dem Nachnamen, Vornamen und Mittelinitialen in einer Tabelle angezeigt wird.

**Antwort:** Sehen Sie sich bitte die `menu15.mmb`-Datei im Verzeichnis Tag 15 auf der CD-ROM an.

## Tag 15, »Developer/2000: Entwicklung von Berichten mit Hilfe von Oracle Reports«

### Test

1. Richtig oder falsch? Sie können eine RDF-Datei in einem Code-Steuerungssystem wie PVCS speichern.

**Antwort:** Falsch. Eine RDF-Datei ist eine binäre Datei. Anstatt zu versuchen, eine Versionskontrolle Ihrer RDF-Dateien zu unterstützen, sollten Sie REX-Dateien generieren, wobei es sich um Textdateien handelt, die unter Versionskontrolle gehalten werden können.

2. Nennen Sie die vier Elemente eines Berichtlayouts.

**Antwort:** Header, Footer, Body und Margin. Nahezu jeder Bericht besitzt ein Body-Layout.

3. Richtig oder falsch? Ein Bericht kann nicht mehr als Abfragen miteinander verknüpfen.

**Antwort:** Falsch. Sie können mehrere Abfragen integrieren, die zueinander in Beziehung stehen. Sie können beispielsweise einen Bericht erstellen, der auf drei Abfragen basiert: eine Abfrage der Departments, eine Abfrage der Dozenten jedes Departments und eine Abfrage der Klassen, die jeder Dozent unterrichtet.

## Übung

Erstellen Sie einen Master-/Detail-Bericht, der jeden Klassenraum und die für diesen Raum geplanten Vorlesungen auflistet, ohne Berücksichtigung der Unterrichtszeiten (z.B. alle Klassen, die in einem bestimmten Raum stattfinden sollen).

**Antwort:** Sehen Sie sich bitte die `Classroom.fmb`-Datei im Verzeichnis Tag 16 auf der CD-ROM an.

## Tag 16, »Developer/2000: Oracle Graphics und den Procedure Builder verwenden«

### Test

1. Welche beiden Dateitypen verwendet Oracle Graphics?

**Antwort:** Eine `.ogd`-Datei - was für Oracle Graphics Designer-Datei steht - enthält die Definition für ein Diagramm oder eine Anzeige. Wenn Sie den Graphics Designer benutzen, generieren Sie eine `.ogr`-Datei (Oracle Graphics Runtime) aus einer `.ogd`-Datei.

2. Nennen Sie drei Diagrammtypen, die mit dem Graphics-Designer festgelegt werden können.

**Antwort:** Linien-, Balken- und Tortengrafik-Diagramm.

3. Richtig oder falsch? Sie können Stored-Program-Unit-Editor zur Änderung der Merkmale eines Pakets, aber nicht des Pakettrumpfs verwenden.

**Antwort:** Falsch. Der Stored Unit Program Editor kann verwendet werden, um sowohl eine Package-Spezifikation als auch einen Package Body zu verändern.

## Übungen

1. Erstellen Sie ein Tortendiagramm, das die von den einzelnen Fachrichtungen des Flugle College angebotenen Kurse darstellt.

**Antwort:** Sehen Sie sich bitte die `Courses_by_Department.ogd`-Datei im Verzeichnis Tag 17 auf der CD-ROM an.

2. Ändern Sie den Trigger für die `Student_Schedule`-Tabelle in der Form, daß in der `Student_Schedule`-Tabelle Änderungen nur gespeichert werden, wenn der Wert in der Spalte `Grade` A, B oder C ist.

**Antwort:** Sehen Sie sich bitte die `Student_Schedule.sql`-Datei im Verzeichnis Tag 17 auf der CD-ROM an.

## Tag 17, »Einführung in Oracle Power Objects«

### Test

1. Richtig oder falsch? Eine Power-Objects-Anwendung verbindet sich mit einer Oracle-Datenbank über ein Transaktionsobjekt der Anwendung.

**Antwort:** Falsch. Eine Power-Objects-Anwendung stellt die Verbindung zu einer Oracle-Datenbank über die Festlegung eines Datenbank-Arbeitssitzungsobjekts her.

2. Nennen Sie drei Typen von Objekten, die Sie auf einem Formular plazieren können.

**Antwort:** Ein Textfeld, eine Optionsschaltfläche und eine Auswahlliste.

3. Geben Sie ein Beispiel für ein Container-Objekt.

**Antwort:** Ein Beispiel für ein Container-Objekt ist die Auswahlliste. Diese kann ein oder mehrere Textfelder enthalten.

## Übungen

1. Bauen Sie ein Formular für die Anzeige und Modifizierung von Informationen über jeden Dozenten auf. Zeigen Sie die Zahl der Klassen an, falls vorhanden, für die der Dozent zum Unterricht eingeteilt ist.
2. Ändern Sie das Student-Formular so ab, daß das GPA des zum gegenwärtigen Zeitpunkt gewählten Studenten angezeigt wird.

**Antwort:** Sehen Sie sich bitte die `Flugle.poa`-Datei im Verzeichnis Tag 18 auf der CD-ROM an.

## Tag 18, »Eine Anwendung mit Oracle Power Objects entwickeln«

### Test

1. Richtig oder falsch? Die Eigenschaft `RecordSource` eines Berichts muß eine einzelne Tabelle oder Datensicht referenzieren; sie kann nicht mit zwei oder mehr Tabellen arbeiten.

**Antwort:** Richtig. Die `RecordSource`-Eigenschaft ist auf eine einzelne Tabelle oder Datenansicht beschränkt. Um einen Master-/Detail-Bericht zu erstellen, müssen Sie entweder eine Datenansicht erzeugen oder dem Bericht ein Unterformular hinzufügen, das einen detaillierten Bericht enthält, der mit dem Master-Bericht verbunden werden kann.

2. Richtig oder falsch? Power Objects unterstützt ein Maximum von drei Ebenen von Vererbung. Mit anderen Worten, Sie können eine Klasse A erstellen, einer Klasse B eine Instanz von Klasse A und einer Klasse C eine Instanz von B hinzufügen.

**Antwort:** Falsch. Power-Objects unterstützt viele Ebenen von Vererbung. Sie sollten jedoch klugerweise beachten, wie viele Ebenen Sie konstruieren, weil zu viele davon einen nachteiligen Effekt auf die Performance haben können.

3. Wenn Sie in einer Power-Objects-Anwendung mit Hilfe der `Validate`-Methode eine Gültigkeitsprüfung von Benutzereingaben durchführen können, warum müssen Sie dann Datenbank-Integritätsregeln und Trigger benutzen, um die Datenintegrität sicherzustellen?

**Antwort:** Sie werden immer Datenbank-Integritätsregeln und Trigger verwenden müssen, um Datenintegrität zu gewährleisten, auch wenn dieselben Firmenregeln in einer Power-Objects-Anwendung erzwungen werden. Der Grund ist, daß Datenbank-Integritätsregeln und Trigger die erste Linie der Verteidigung von Datenintegrität sind. Weil eine einzelne Datenbank viele verschiedene Anwendungen unterstützen kann, wobei jede dieser Anwendungen mit einem anderen Werkzeug oder von einem anderen Entwickler erstellt wurde, können Sie nicht davon ausgehen, daß in alle Anwendungen der vollständige Satz von Firmenregeln integriert wurde.

## Übungen

1. Power Objects enthält eine Anzahl von vordefinierten Klassen. Wenn Sie den Inhalt des Verzeichnisses `Classes` untersuchen, werden Sie eine Klasse namens `clsMeter` finden. Fügen Sie eine Instanz davon zu `MyForm` hinzu. Sehen Sie zu, ob Sie herausbekommen, wie man die X-Position des Mauszeigers auf dem Formular mit dieser Anzeige verknüpfen kann.
2. Benutzen Sie Power Objects, um einen Bericht aufzubauen, der eine Liste der Dozenten enthält, die zu jedem Department gehören. Fügen Sie ein Suchfeld zu jeder Dozenten-Zeile hinzu, das anzeigt, ob ein Dozent in dem Semester keinerlei Vorlesungen hält.

**Antwort:** Sehen Sie sich bitte die `FlugLe.poa`-Datei im Verzeichnis Tag 19 auf der CD-ROM an.

## Tag 19, »Eine Übersicht über die Datenbanksicherheit und die Datenbankoptimierung unter Oracle«

### Test

1. Richtig oder falsch? Eine Datenbankrolle kann Benutzern und anderen Datenbankrollen zugeteilt werden.

**Antwort:** Richtig. Sie können bestimmte typische Rollen erzeugen, die einige grundlegenden Berechtigungen besitzen, und diese Rollen dann mehr spezialisierten Rollen zuteilen.

2. Richtig oder falsch? Das Hinzufügen eines Index verbessert immer die Leistung einer Anwendung.

**Antwort:** Falsch. Ein nicht-einmaliger Index sollte nur dann hinzugefügt werden, wenn die indizierte Spalte häufig in Abfragen der Tabelle benutzt wird. Außerdem kann der Index die Gesamtleistung herabsetzen, wenn die Inhalte der Tabelle häufig wechseln.

3. Richtig oder falsch? Jeder Oracle-Benutzer kann ein öffentliches Synonym erstellen, solange dieses auf eine Tabelle verweist, die sich im Besitz des Benutzers befindet.

**Antwort:** Falsch. Das Erzeugen eines öffentlichen Synonyms erfordert die DBA-Rolle oder das Systemprivileg `CREATE PUBLIC SYNONYM`.

## Übung

Sie können diese Fragen allgemein beantworten, ohne die Verwendung von SQL-Anweisungen: Wenn Sie der DBA am Flugle College wären, welche Datenbankrolle würden Sie erzeugen? Welche Berechtigungen würden Sie dieser Rolle gewähren?

**Antwort:** Sie würden wahrscheinlich diese Rollen erzeugen:

- Eine Rolle für Studenten
- Eine Rolle für Dozenten
- Eine Rolle für die Person des Vorsitzenden eines Departments
- Eine Rolle für den College-Administrator

Objektprivilegien werden im allgemeinen als Create(C), Read(R), Update(U), Delete(D) oder CRUD bezeichnet. Eine Matrix, die alle diese Berechtigungen für jede Rolle definiert, wird als CRUD-Matrix bezeichnet.

Berechtigungen für die Studenten-Rolle:

Class: R

Class\_Location: R

Course: R

Department: R

Instructor: R

Schedule\_Type: R

Schedule\_Type\_Details: R

Student: R (nur für die eigenen Daten)

Student\_Schedule: CRUD (nur für die eigenen Daten)

## Berechtigungen für die Dozenten-Rolle:

Class: R

Class\_Location: R

Course: R

Department: R

Instructor: RU (nur für die eigenen Daten)

Schedule\_Type: R

Schedule\_Type\_Details: R

Student: R (nur für die eigenen Daten)

Student\_Schedule: RU

## Berechtigungen für die Rolle des Department-Vorsitzenden:

Class: CRUD

Class\_Location: R

Course: CRUD

Department: R

Instructor: CRUD

Schedule\_Type: R

Schedule\_Type\_Details: R

Student: R

Student\_Schedule: RU

## Berechtigungen für die Rolle des College-Administrators:

Class: CRUD



Class\_Location: CRUD

Course: CRUD

Department: CRUD

Instructor: CRUD

Schedule\_Type: CRUD

Schedule\_Type\_Details: CRUD

Student: CRUD

## Tag 20, »Einsatz von Oracle Database Designer und Oracle Designer/2000 in der Anwendungsentwicklung«

### Test

1. Richtig oder falsch? Ein Nachteil vom Database Designer ist, daß er nur Oracle-Datenbanken unterstützt.

**Antwort:** Falsch. Database Designer erlaubt Ihnen die Generierung einer RDB-, DB2- oder SQL-Server-Datenbank, ausgehend von Ihrem Datenbankdesign.

2. Nennen Sie drei Vorteile der Verwendung des Database Designer anstelle eines Texteditors zur Erstellung der DDL-Anweisungen, die eine Datenbank erzeugen.

**Antwort:** Einige der Vorteile der Verwendung des Database Designer umfassen folgendes:

- Deutlich weniger Eingabefehler, wie z.B. das falsche Eingeben eines Spaltennamens.
- Die Fähigkeit, die Definitionen von Tabellen und Spalten in derselben Datei zu erfassen.
- Die Fähigkeit, Informationen an Designer/2000 weiterzugeben, wie z.B. Anzeigeformate für Spalten.
- Ein Diagramm ist viel leichter zu verstehen als ein Satz von SQL-Anweisungen und kann anderen Projektmitgliedern als Überblick/Rückmeldung präsentiert werden.

## Tag 21, »Oracle - Die nächste Generation«

### Test

1. Welche Aufgabe haben die `http`- und `http`-PL/SQL-Pakete?

**Antwort:** Die `http`- und `http`-PL/SQL-Pakete werden benutzt, um dynamischen HTML-Code zu generieren.

2. Richtig oder falsch? Die Network Computing Architecture steht nur für Sun Solaris und Microsoft Windows NT zur Verfügung.

**Antwort:** Falsch. Die Network Computing Architecture ist eine Strategie, die voraussichtlich eine breite Palette von Plattformen unterstützen wird.

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

---

❖ Kapitel ❖ Top | Inhalt | Index | **SAMS**

# Oracle in 21 Tagen Inhaltsverzeichnis

---

## Einleitung

1. Die Welt relationaler Datenbanken erforschen
  2. Richtlinien für die Entwicklung einer Oracle-Anwendung
  3. Logischer Datenbankentwurf
  4. Implementierung Ihres logischen Modells: Physischer Datenbankentwurf
  5. Einführung in SQL
  6. Mit SQL Daten modifizieren
  7. In SQL integrierte Funktionen nutzen
  8. Fortgeschrittene Abfragen mit SQL
  9. Programmierung einer Oracle-Datenbank mit PL/SQL
  10. Programmentwicklung mit PL/SQL
  11. Weitere Programmiertechniken mit PL/SQL
  12. Developer/2000: Einführung in Oracle Forms
  13. Developer/2000: Entwicklung einer Benutzeroberfläche
  14. Developer/2000: Anwendungsentwicklung mit Hilfe von Oracle Forms
  15. Developer/2000: Entwicklung von Berichten mit Hilfe von Oracle Reports
  16. Developer/2000: Oracle Graphics und den Procedure Builder verwenden
  17. Einführung in Oracle Power Objects
  18. Eine Anwendung mit Oracle Power Objects entwickeln
  19. Datenbanksicherheit und Datenbankoptimierung unter Oracle
  20. Database Designer und Designer/2000 in der Anwendungsentwicklung
  21. Oracle - Die nächste Generation
  22. Bonustag: Power-Builder für eine Oracle-Datenbank verwenden
  23. Bonustag: Oracle8 - Neue Merkmale
  24. Antworten zu den Tests und Übungen
- 

Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3





# Oracle in 21 Tagen Index

[%](#) | [4](#) | [Ä](#) | [A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [J](#) | [K](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#) | [R](#) | [S](#) | [T](#) | [U](#) | [V](#) | [W](#) | [Z](#)

---

## - % -

%found [1](#)

%notfound [1](#)

%rowcount [1](#)

%rowtype [1](#)

%rowtype : verwenden [1](#)

%type [1](#)

%type : verwenden [1](#)

## - 4 -

4GL [1](#)

## - Ä -

Änderung rückgängig machen [1](#)

## - A -

Abfrage, Ad-hoc- [1](#)

Abfragesprache, strukturierte [1](#)

Alias [1](#), [2](#)

Aliasname [1](#)

alter table [1](#)

alter table : Grundformen [1](#)

and [1](#)

Anforderungen sammeln [1](#)

ANSI [1](#)

Anweisung : alter table [1](#)

Anweisung : Analyse [1](#)

Anweisung : commit [1](#), [2](#)

Anweisung : create directory [1](#)

Anweisung : create table [1](#)

Anweisung : CREATE TRIGGER [1](#)

Anweisung : delete [1](#)

Anweisung : exit [1](#)

Anweisung : for-loop [1](#)

Anweisung : goto [1](#)

Anweisung : if [1](#)

Anweisung : insert [1](#)

Anweisung : loop [1](#)

Anweisung : null [1](#)

Anweisung : rollback [1](#), [2](#)

Anweisung : select [1](#), [2](#), [3](#)

Anweisung : truncate table [1](#)

Anweisung : update [1](#)

Anweisung : while-loop [1](#)

Anweisung : Zuweisungs- [1](#)

Anwendung [1](#)

Anwendung : Komplexität [1](#)

Anwendung : Pflege [1](#)

Anwendung : \ einsetzen [1](#)

Anwendung : \ erstellen [1](#)

Anwendung : \ installieren [1](#)

Anwendung : \ testen [1](#)

Anwendungsengpässe ermitteln [1](#)

Anwendungsentwicklung [1](#), [2](#)

Anwendungsentwicklung : Assistenten [1](#)

Anwendungsentwicklung : Schritte [1](#)

Anwendungsentwicklung : Umgebungen [1](#)

Anwendungsentwicklung : Werkzeuge [1](#)

Anwendungsprogramm [1](#)

Anwendungs-Trigger [1](#)

Applet Viewer [1](#)

Arbeit sichern [1](#)

Arbeitseinheit, logische [1](#)

Arbeitssitzung [1](#)

Arbeitssitzung : \ erzeugen [1](#)

Architektur [1](#)

Architektur : \ mit drei Ebenen [1](#)

Architektur : \ mit zwei Ebenen [1](#)

ascii [1](#)

ASCII-Wert [1](#)

atomar [1](#)

Attribut [1](#), [2](#)

Attributwert [1](#)

Ausführungsplan [1](#)

Ausführungsplan : \ ermitteln [1](#)

Ausführungsteil [1](#)

Auswahlliste [1](#)

Authorization Server [1](#)

Automatisierung [1](#)



avg [1](#), [2](#)

## **- B -**

B\*-Baumstruktur [1](#)

Basistabellenblock [1](#)

Bedingungen kombinieren [1](#)

Benutzer [1](#), [2](#), [3](#)

Benutzer : \ erzeugen [1](#)

Benutzer : \ vordefinierte [1](#)

Benutzerberechtigung [1](#)

Benutzereingaben prüfen [1](#)

Benutzeroberfläche [1](#)

Benutzeroberfläche : konsistente [1](#)

Benutzerrolle [1](#)

Benutzerschulung [1](#)

Bericht : Abschnitte [1](#)

Bericht : Ausführung aufzeichnen [1](#)

Bericht : Detail [1](#)

Bericht : Laufzeit-Version [1](#)

Bericht : Oracle-Power-Objects- [1](#)

Bericht : PageHeader [1](#)

Bericht : ReportFooter [1](#)

Bericht : ReportHeader [1](#)

Bericht : Suchfeld hinzufügen [1](#)

Bericht : tabellarischer [1](#)

Bericht : Verknüpfung erstellen [1](#)

Bericht : \ aufrufen [1](#)

Bericht : \ drucken [1](#)

Bericht : \ speichern [1](#)

Bericht : \ starten [1](#)

Bericht : \ testen [1](#), [2](#)

Berichtsobjekt [1](#)

Berichtsobjekt : \ festlegen [1](#)

Betriebssystem [1](#)

between [1](#)

Beziehung [1](#)

Beziehung : \ definieren [1](#)

BFILE [1](#)

BFILE : \ verwenden [1](#)

Bibliothek [1](#), [2](#)

Bildlaufleiste [1](#)

Bildlaufleiste : \ hinzufügen [1](#)

Binärobjekte, große [1](#)

binary integer [1](#), [2](#)

BLOB [1](#), [2](#)

BLOB : Festlegen eines Nullwerts [1](#)

BLOB : verwenden [1](#)

Block [1](#)

Block : anonymer [1](#)

Blockelement, Aktivierreihenfolge [1](#)

boolean [1](#), [2](#)

Budget [1](#)

## **- C -**

C [1](#), [2](#)

Canvas [1](#)

Canvas : Ansichten [1](#)

Canvas : \ erstellen [1](#)

Cartridge-Administration [1](#)

Cartridges [1](#), [2](#)

Cartridge-Verwaltung [1](#)

cascade [1](#)

ceil [1](#)

char [1](#)

check-Klausel [1](#)

check-Regel [1](#)

check-Regel : Vorteile [1](#)

check-Regel : \ erstellen [1](#)

check-Regel : \ festlegen [1](#)

Client [1](#)

Client : Thin [1](#)

Client/Server [1](#)

Client/Server-Architektur [1](#)

Client/Server-Architektur : Vorteile [1](#)

Client/Server-Datenverarbeitung, Ursprünge [1](#)

Client-System [1](#)

Client-Treiber [1](#)

CLOB [1](#)

CLOB : Festlegen eines Nullwerts [1](#)

CLOB : \ verwenden [1](#)

COBOL [1](#), [2](#)

Code dokumentieren [1](#)

commit [1](#), [2](#)

Common Facility [1](#)

Computer Associates [1](#)

CONNECT [1](#)

CONNECT\_TIME [1](#)

Connect-by-Klausel [1](#), [2](#)

constraint [1](#)

CORBA [1](#), [2](#)

CORBA 2.0 [1](#)

count [1](#)

create directory [1](#)

create table [1](#)

CREATE TRIGGER [1](#)

Cursor [1](#)

Cursor : lesen der Zeilen [1](#)

Cursor : lesen von Zeilen [1](#)

Cursor : \ deklarieren [1](#), [2](#)

Cursor : \ öffnen [1](#), [2](#)

Cursor : \ schließen [1](#), [2](#)

**- D -**

DAD [1](#)

DAD Administration [1](#)

Data Dictionary [1](#)

Data Dictionary : Ansichten [1](#)

Database Access Descriptor [1](#)

Database Access Descriptor : \ verwalten [1](#)

Database Designer [1](#), [2](#)

DataWindow erstellen [1](#)

date [1](#)

date : \ verwenden [1](#)

Daten : Anforderungen [1](#)

Daten : hierarchische [1](#)

Daten : Zugriff [1](#)

Daten : \ ändern [1](#), [2](#)

Daten : \ abrufen [1](#), [2](#)

Daten : \ löschen [1](#)

Daten : \ sortieren [1](#)

Datenbank : Arbeitssitzung [1](#)

Datenbank : Aufbau einer Verbindung [1](#)

Datenbank : Benutzer [1](#)

Datenbank : Entwicklung [1](#)

Datenbank : LAN- [1](#)

Datenbank : PC- [1](#)

Datenbank : relationale [1](#), [2](#)

Datenbank : Verbindung [1](#)

Datenbankentwurf : logischer [1](#)

Datenbankentwurf : physischer [1](#)

Datenbankobjekt : Definition [1](#)

Datenbankobjekt : Inhalt [1](#)

Datenbankprofil definieren [1](#)

Datenbankprogramm [1](#)

Datenbanktabelle erstellen [1](#)

Datenbank-Trigger [1](#), [2](#), [3](#), [4](#)

Datenbank-Trigger : Editor [1](#)

Datenbank-Trigger : \ erstellen [1](#), [2](#)

Datenbank-Trigger : \ testen [1](#)

Datenelement [1](#)

Datenintegrität [1](#)

Datenmodell [1](#), [2](#)

Datenmodell : \ entwickeln [1](#)

Datenmodell : \ erstellen [1](#), [2](#)

Datenmodell : \ logisches [1](#), [2](#)

Datensatz [1](#)

Datensatz : benutzerdefinierter [1](#)

Datensatz : \ ändern [1](#)

Datensatzgruppe [1](#)

Datensicht [1](#), [2](#)

Datensicht : einfache [1](#)

Datensicht : Komplexität verbergen [1](#)

Datensicht : Syntax [1](#)

Datensicht : Zugriff auf Daten einschränken [1](#)

Datentyp [1](#)

Datentyp : binary integer [1](#)

Datentyp : boolean [1](#)

Datentyp : char [1](#)

Datentyp : date [1](#)

Datentyp : numerischer [1](#)

Datentyp : OBJECT [1](#)

Datentyp : raw [1](#)

Datentyp : record [1](#)

Datentyp : sysdate [1](#)

Datentyp : table [1](#)



Datentyp : varchar [1](#)

Datentyp : varray [1](#)

Datenunabhängigkeit, logische [1](#)

Datenverarbeitung [1](#)

Datenverarbeitung : Client/Server [1](#)

Datenverarbeitungsarchitektur, frühere [1](#)

Datum [1](#), [2](#)

Datumsangabe [1](#)

Datumsangabe : Umwandlung in Zeichenfolgen [1](#)

Datumsangabe : \ bearbeiten [1](#)

Datumsberechnung [1](#)

Datumsformat [1](#)

Datumsformat : Elemente [1](#)

Datumsformat : Standard- [1](#)

Datumsinformation [1](#)

Datumswert : Fehlercodes [1](#)

DBA [1](#)

DBA\_OBJECTS [1](#)

dBASE II [1](#)

DBMS [1](#)

DCL, Data Control Language [1](#)

DDL, Data Definition Language [1](#), [2](#)

DDL-Anweisungen erstellen [1](#)

dec [1](#)

decimal [1](#), [2](#)

declare [1](#)

decode [1](#)

Default Configuration [1](#)

default-Klausel [1](#)

Deklarationsteil [1](#)

delete [1](#), [2](#)

delete : Syntax [1](#)

Design Wizards [1](#)

Designer/2000 [1](#), [2](#)

Designer/2000 : Grundlagen [1](#)

Detail [1](#)

Detail-Abfrage erstellen [1](#)

Detailbereich festlegen [1](#)

Detail-Block definieren [1](#)

Developer/2000 [1](#), [2](#), [3](#)

Developer/2000 : Entwicklungskriterien [1](#)

Developer/2000 for the Web [1](#)

Diagnosefähigkeit [1](#)

Diagramm : Tabelle hinzufügen [1](#)

Diagramm : \ aufrufen [1](#)

Diagramm : \ erstellen [1](#)

Diagramm : \ speichern [1](#)

Diagramm-Layout anpassen [1](#)

Diagrammtyp wählen [1](#)

distinct [1](#)

DML : Data Manipulation Language [1](#)

DML : Grundanweisungen [1](#)

drop-primary-key-Klausel [1](#)

DSS [1](#)

DUP\_VAL\_ON\_INDEX [1](#)

Dynamic HTML [1](#)

## **- E -**

Effizienz [1](#)

Entität [1](#), [2](#)

Entitätsattribut [1](#)

Entitätsbeziehungsdiagramm [1](#)

Entitätsbeziehungsdiagramm : Erstellung [1](#)

Entitätsbeziehungsmodell [1](#)

Entity Relationship Diagrammer [1](#)

Entity-Relationship-Diagramm erstellen [1](#)

Entwicklungsprozeß [1](#)

Entwicklungssoftware [1](#)

Entwicklungswerkzeug [1](#)

Ereignis : auslösen [1](#)

Ereignis : triggerauslösendes [1](#)

Exception [1](#), [2](#)

Exception : deklarieren [1](#)

Exception : DUP\_VAL\_ON\_INDEX [1](#)

Exception : INVALID\_NUMBER [1](#)

Exception : no\_data\_found [1](#)

Exception : too\_many\_rows [1](#)

Exception : value\_error [1](#)

Exception : vordefinierte [1](#)

Exception-Handler [1](#)

Exception-Teil [1](#), [2](#)

execute [1](#)

exists [1](#)

exit [1](#)

EXP\_FULL\_DATABASE [1](#)

explain plan [1](#)

## - F -

false [1](#)

Fehler, Rückgabe [1](#)

Fehlerbehandlung [1](#)

Fehlerbehandlung : \ unter PL/SQL [1](#)

Fehlermeldung bei Prozeduren [1](#)

Fenster [1](#)

Fernverbindung [1](#)

float [1](#)

floor [1](#)

FMX-Dateien erzeugen [1](#)

for-loop [1](#), [2](#)

Formatmaske, numerische [1](#)

Forms Server Listener [1](#)

Formular [1](#)

Formular : Abrufreihenfolge festlegen [1](#)

Formular : berechnete Werte anzeigen [1](#)

Formular : Elemente [1](#)

Formular : Instanz einer Klasse hinzufügen [1](#)

Formular : \ aufrufen [1](#)

Formular : \ benennen [1](#)

Formular : \ erstellen [1](#)

Formular : \ navigieren [1](#)

Formular : \ schließen [1](#)

Formular : \ speichern [1](#)

Formular : \ testen [1](#), [2](#)

Formular : \ verbessern [1](#)

Formular Instructor [1](#)

for-Schleife, Cursor-gesteuerte [1](#)

FORTRAN [1](#)

FoxPro [1](#)

Fremdinformation [1](#)

Fremdschlüssel [1](#), [2](#), [3](#)

Fremdschlüssel : Deklaration [1](#), [2](#)

Fremdschlüssel : \ festlegen [1](#), [2](#)

Fremdschlüssel : \ löschen [1](#), [2](#)

Fremdschlüssel : \ vereinbaren [1](#)

from-Klausel [1](#)

Function Hierarchy Diagrammer [1](#)

Funktion : Anforderungen [1](#)

Funktion : Argument [1](#)

Funktion : ascii [1](#)

Funktion : avg [1](#)

Funktion : ceil [1](#)

Funktion : count [1](#)

Funktion : decode [1](#)

Funktion : floor [1](#)

Funktion : gespeicherte [1](#), [2](#)

Funktion : Gruppen- [1](#)

Funktion : initcap [1](#)

Funktion : integrierte [1](#), [2](#)

Funktion : Konvertierungs- [1](#)

Funktion : length [1](#)

Funktion : lower [1](#)

Funktion : lpad [1](#)

Funktion : ltrim [1](#)

Funktion : max [1](#), [2](#)

Funktion : min [1](#), [2](#)

Funktion : numerische [1](#)

Funktion : nvl [1](#)

Funktion : replace [1](#)

Funktion : round [1](#)

Funktion : rpad [1](#)

Funktion : rtrim [1](#)

Funktion : substr [1](#)

Funktion : sum [1](#)

Funktion : Syntax [1](#)

Funktion : to\_char [1](#), [2](#)

Funktion : to\_date [1](#)

Funktion : to\_number [1](#)

Funktion : trunc [1](#)

Funktion : upper [1](#)

Funktion : Vorwärtsdeklaration [1](#)

Funktion : vsize [1](#)

Funktion : Zeichen- [1](#)

Funktion : \ aufrufen [1](#)

Funktion : \ deklarieren [1](#), [2](#)

Funktion : \ erstellen [1](#)



## - G -

Generierung [1](#)

Gleichzeitigkeit [1](#)

goto [1](#)

Großrechner [1](#)

Großschreibung [1](#)

Großschreibung : \ ändern [1](#)

group by-Klausel [1](#)

GroupHeader-Bereich festlegen [1](#)

Gruppenfunktion : integrierte [1](#)

Gruppenfunktion : \ kombinieren [1](#)

## - H -

Hauptfenster [1](#)

having-Klausel [1](#)

HP [1](#)

htf [1](#)

htp [1](#)

## - I -

IBM [1](#), [2](#)

identifizierend [1](#)

IDLE\_TIME [1](#)

if [1](#)

IMP\_FULL\_DATABASE [1](#)

Implementierung [1](#), [2](#)

in [1](#), [2](#)

in out [1](#)

Index [1](#)

Index : Anzahl [1](#)

Index : Syntax [1](#)

Index : \ erstellen [1](#), [2](#)

Informationen, hierarchische [1](#)

Informationssystem [1](#)

Informix [1](#), [2](#)

initcap [1](#)

insert [1](#), [2](#)

insert : Syntax [1](#), [2](#)

Integrität : Daten- [1](#)

Integrität : deklarative [1](#)

Integrität : referentielle [1](#), [2](#), [3](#), [4](#)

Integritätsregel [1](#), [2](#), [3](#), [4](#), [5](#)

Integritätsregel : für Eindeutigkeit [1](#)

Integritätsregel : Primary Key [1](#)

Integritätsregel : Unique [1](#)

Integritätsregel : \ aktivieren [1](#)

Integritätsregel : \ ansehen [1](#)

Integritätsregel : \ deaktivieren [1](#)

Integritätsregel : \ für Schlüsselwerte [1](#)

Inter-Cartridge Exchange [1](#)

Internet [1](#)

Interoperabilität [1](#)

intersect [1](#)

intersect : Anforderungen [1](#)

into-Klausel [1](#)

INVALID\_NUMBER [1](#)

ISO [1](#)

**- J -**

Java [1](#)

Java : Unterstützung [1](#)

Join [1](#)

Join : einfacher [1](#)

Join : Equi- [1](#)

Join : mit mehreren Tabellen [1](#)

Join : Outer- [1](#)

Join : Self- [1](#)

## - K -

Kardinalität [1](#)

Kartesisches Produkt [1](#)

Klasse : Eigenschaft überschreiben [1](#)

Klasse : Instanz [1](#)

Klasse : Methode überschreiben [1](#)

Klasse : Methodencode hinzufügen [1](#)

Klasse : \ erstellen [1](#)

Klasseneigenschaft, geerbte [1](#)

Klassenmethode, geerbte [1](#)

Klassenobjekt, Eigenschaft ändern [1](#)

Kleinschreibung [1](#)

Kleinschreibung : \ ändern [1](#)

Kommentar [1](#)

Kommentar : \ einfügen [1](#)

Kommunikation [1](#)

Kommunikation : zwischenmenschliche [1](#)

Komponenten, wiederverwendbare [1](#)

Konfigurationsverwaltung [1](#)

Konstante, deklarieren [1](#)

Kontextoption [1](#)

Kontrollstruktur [1](#)

Kopiertechnik [1](#)

## **- L -**

LAN [1](#)

Layout [1](#)

Layout-Editor [1](#)

Leerzeichen [1](#)

Leerzeichen : führendes [1](#)

Leerzeichen : nachfolgendes [1](#)

Leerzeichen : \ entfernen [1](#)

Leerzeichen : \ löschen [1](#)

Leistung, Anforderungen [1](#)

Leistungsfähigkeit verbessern [1](#)

Leistungssteigerung [1](#)

length [1](#)

Lesekonsistenz [1](#)

level [1](#)

like [1](#), [2](#), [3](#)

Liste : \ von Funktionen [1](#)

Liste : \ von Paketen [1](#)

Liste : \ von Prozeduren [1](#)

lob [1](#)

Local Area Network [1](#)

Log Analyzer [1](#)

Logger [1](#)

LOGICAL\_READS\_PER\_CALL [1](#)

long [1](#)

loop [1](#)

LOV [1](#)

lower [1](#)

lpad [1](#)

ltrim [1](#)

**- M -**

Management-System [1](#), [2](#)

Master-/Detail-Formular [1](#)

Master-/Detail-Formular : \ ausführen [1](#)

Master-/Detail-Formular : \ erstellen [1](#)

Master-/Detail-Formular : verstellen [1](#)

Master-Abfrage erstellen [1](#)

Master-Block definieren [1](#)

Master-Detail-Bericht erstellen [1](#)

max [1](#), [2](#)

Meldungszeile [1](#)

Menge, aktive [1](#)

Mengenoperator [1](#)

Mengenoperator : \ verwenden [1](#)

Menü [1](#)

Menü : Elemente [1](#)

Menü : Elemente hinzufügen [1](#)

Menü : \ entwickeln [1](#)

Menü : \ erzeugen [1](#)

Menü : \ integrieren [1](#)

Menü : \ speichern [1](#)

Menü : \ umbenennen [1](#)

Methode [1](#)

Methodencode [1](#)

Methodencode : \ testen [1](#)

Microsoft [1](#), [2](#)

Middleware [1](#), [2](#)

Migration [1](#)

min [1](#), [2](#)

minus [1](#)

Modell, logisches [1](#)

Module Structure Diagrammer [1](#)

Multiform-Anwendung entwickeln [1](#)

Multiplexer [1](#)

## **- N -**

natural [1](#), [2](#)

Navigator [1](#)

Navigator Manager [1](#)

NCA, Network Computing Architecture [1](#)

NCLOB [1](#)

NetWare [1](#)



Network Computer [1](#)

Network Computing Architecture [1](#)

Network Computing Architecture : Komponenten [1](#)

Netzwerk [1](#)

Netzwerkprotokoll [1](#)

nicht-identifizierend [1](#)

no\_data\_found [1](#)

Normalform : dritte (3NF) [1](#)

Normalform : erste (1NF) [1](#)

Normalform : zweite (2NF) [1](#)

Normalisierung [1](#)

Normalisierung : Ziele [1](#)

Normalisierung : \ anwenden [1](#)

Normalisierungsregel 1 [1](#)

Normalisierungsregel 2 [1](#)

Normalisierungsregel 3 [1](#)

Null [1](#), [2](#), [3](#)

Nullwert [1](#)

number [1](#)

numeric [1](#)

Nur-Lese-Transaktion [1](#)

nvl [1](#)

- **O** -

object [1](#)

Object Navigator [1](#), [2](#)

Object Service [1](#)

Object-Berechtigung [1](#)

Objekt : Abstand einstellen [1](#)

Objekt : Ausrichtung einstellen [1](#)

Objekt : Ausrichtung festlegen [1](#)

Objekt : Größe einstellen [1](#), [2](#)

Objekt : Größe festlegen [1](#)

Objekt : gro\_ es [1](#)

Objektansichten [1](#)

Objekteigenschaft [1](#)

OMG, Object Management Group [1](#)

Online-Hilfe [1](#)

Operator : exists [1](#)

Operator : intersect [1](#)

Operator : minus [1](#)

Operator : prior [1](#)

Operator : union [1](#)

Operatoren : arithmetische [1](#)

Operatoren : logische [1](#)

Operatoren : Zeichenfolgen- [1](#)

Optimizer : kostenbasierter [1](#)

Optimizer : regelbasierter [1](#)

Option : all [1](#)

Option : Context [1](#)

Option : distinct [1](#)

Option : Distributed [1](#)

Option : Replication [1](#)

Option : Spatial Data [1](#)

Option räumliche Daten [1](#)

Optionsschaltfläche verwenden [1](#)

or [1](#)

Oracle [1](#), [2](#)

Oracle : Personal [1](#)

Oracle : Account [1](#)

Oracle : Datentypen [1](#)

Oracle : Fehlerbehandlung [1](#)

Oracle : Forms [1](#)

Oracle : Navigator [1](#)

Oracle : Personal [1](#)

Oracle : Procedure Builder [1](#)

Oracle : Produktreihe [1](#)

Oracle : Reports [1](#)

Oracle : Terminologie [1](#)

Oracle : Universal Server [1](#), [2](#)

Oracle : WebServer [1](#)

Oracle : Workgroup Server [1](#), [2](#)

Oracle Forms : Abfrageverhalten [1](#)

Oracle Forms : Elemente [1](#)

Oracle Forms : \ für das Web konfigurieren [1](#)

Oracle Graphics [1](#)

Oracle Graphics : Elemente [1](#)

Oracle Objects Option [1](#)

Oracle Power Objects, Elemente [1](#)

Oracle Reports [1](#)

Oracle Reports : Programme [1](#)

Oracle Web Server [1](#)

Oracle Web Server : Elemente [1](#)

Oracle8 [1](#)

Oracle8 : neue Merkmale [1](#)

Oracle-Anwendung, Entwicklung [1](#)

Oracle-Bericht, Elemente [1](#)

Oracle-Datenbank : Anbindung an das Web [1](#)

Oracle-Datenbank : Optimierung [1](#)

Oracle-Datenbank : \ entwickeln [1](#)

Oracle-Power-Objects-Bericht erstellen [1](#)

Oracle-RDBMS [1](#)

ORB, Object Request Broker [1](#)

order-by-Klausel [1](#), [2](#)

or-replace-Klausel [1](#)

out [1](#)

## **- P -**

Package [1](#)

PageFooter [1](#)

PageHeader [1](#)

PageHeader-Bereich festlegen [1](#)

Paket [1](#)

Paket : \ deklarieren [1](#)

Pakettrumpf deklarieren [1](#)

Paradox [1](#)

Parameter [1](#)

Parameterformular [1](#)

PC [1](#)

PC : Geburtsstunde [1](#)

Personalcomputer [1](#)

PL/SQL [1](#), [2](#), [3](#)

PL/SQL : Unterprogramm [1](#)

PL/SQL-Datentypen, zusätzliche [1](#)

PL/SQL-Interpreter [1](#)

PL/SQL-Paket : htf [1](#)

PL/SQL-Paket : htp [1](#)

PL/SQL-Skript [1](#)

PL/SQL-Tabelle [1](#)

PL/SQL-Tabelle : Rückgabe [1](#)

PL/SQL-Tabelle : \ deklarieren [1](#)

Plattenspeicher, Konflikte beim Zugriff verringern [1](#)

Plattform [1](#)

Plattformunabhängigkeit [1](#)

Politik [1](#)

Portabilität [1](#)

Portierbarkeit [1](#)

positive [1](#), [2](#)

Power Objects [1](#)

PowerBuilder, Merkmale [1](#)

Precompiler [1](#)

Primärer Bereich [1](#)

Primärschlüssel [1](#), [2](#)

Primärschlüssel : \ ändern [1](#)

Primärschlüssel : \ definieren [1](#), [2](#)

Primärschlüssel : \ Deklaration [1](#)

Primärschlüssel : \ durchsetzen [1](#)

Primärschlüssel : \ festlegen [1](#)

Primärschlüssel : \ hinzufügen [1](#)

Primärschlüssel : \ löschen [1](#), [2](#)

Primärschlüssel : \ neu erstellen [1](#)

Primärschlüssel : \ verwenden [1](#)

prior [1](#)

Procedure Builder [1](#)

Procedure Builder : Elemente [1](#)

Process Modeller [1](#)

Profil [1](#)

Programmblock : \ bearbeiten [1](#)

Programmblock : \ betrachten [1](#)

Programmblock-Editor [1](#)

Programmentwicklung [1](#)

Programmiertechnik für PL/SQL [1](#)

Programmierung [1](#)

Prozedur : Argument [1](#)

Prozedur : externe [1](#)

Prozedur : gespeicherte [1](#)

Prozedur : permanente [1](#)

Prozedur : raise\_application\_error [1](#)

Prozedur : Syntax [1](#)

Prozedur : Vorwärtsdeklaration [1](#)

Prozedur : \ abrufen [1](#)

Prozedur : \ aufrufen [1](#)

Prozedur : \ deklarieren [1](#)

Prozedur : \ erstellen [1](#)

Prozedur : \ temporäre [1](#)



Prozeßmodellierung [1](#)

Prozessor, 8088 [1](#)

Pseudospalte [1](#), [2](#)

Pseudospalte : \ verketten [1](#)

Pyramid [1](#)

## - R -

Rahmen [1](#)

raise\_application\_error [1](#)

RDBMS, Relational Database Management System [1](#)

RDF-Datei [1](#)

record [1](#)

referentielle Integrität erzwingen [1](#)

Registrierungseditor [1](#)

Relation [1](#)

Relationale Datenbank : Entstehung [1](#)

Relationale Datenbank : Theorie [1](#)

replace [1](#)

Replikationsoption [1](#)

ReportFooter [1](#)

ReportHeader [1](#)

Repository [1](#)

Repository-Administration [1](#)

RESOURCE [1](#)

Ressourcen einschränken [1](#)

rollback [1](#), [2](#)

Rollback [1](#)

Rolle [1](#)

Rolle : vordefinierte [1](#)

Rolle : \ erstellen [1](#)

Rolle : \ verwenden [1](#)

round [1](#)

rpad [1](#)

rtrim [1](#)

Rückgabewerte speichern [1](#)

## - S -

Savepoint [1](#)

Savepoint : Vereinbarung [1](#)

Savepoint : \ verwenden [1](#)

Schema [1](#)

Schema Manager [1](#)

Schlüsselwort : cascade [1](#)

Schlüsselwort : distinct [1](#)

Schlüsselwort : optionales [1](#)

Schnittstelle [1](#)

SCO UNIX [1](#)

Security Manager [1](#)

Seitenzahlen [1](#)

Sekundärer Bereich [1](#)

select [1](#), [2](#)

select : Änderungen [1](#)

select : Liste [1](#)

Sequent [1](#)

Sequenz [1](#)

Server [1](#)

Server-Treiber [1](#)

SGA, Umfang vergrößern [1](#)

Shortcuts [1](#)

show errors [1](#)

Sichtbarkeit [1](#)

Skalierbarkeit [1](#)

Skript [1](#)

Software [1](#)

Software : Qualität [1](#)

Software-Entwicklung, komponentenbasierte [1](#)

Spalte [1](#)

Spalte : mehrdeutige [1](#)

Spalte : numerische [1](#)

Spalte : \ benennen [1](#)

Spalte : \ löschen [1](#)

Spalte : \ mit Fremdschlüssel [1](#)

Spalte : \ mit Primärschlüssel [1](#)

Spalte : \ umbenennen [1](#)

Spaltenbreite : \ erweitern [1](#)

Spaltenbreite : \ verringern [1](#)

Spaltendefinition, Änderung [1](#)

Spaltenüberschrift bearbeiten [1](#)

Spaltenwert : \ auf null setzen [1](#)

Spaltenwert : \ einschränken [1](#)

Sprache, blockstrukturierte [1](#)

SQL [1](#), [2](#), [3](#)

SQL : Anweisungen [1](#)

SQL : Grammatik [1](#)

SQL\*Net [1](#)

SQL\*Plus-Skript [1](#)

SQLCode [1](#)

sqlcode auswerten [1](#)

SQLDBCode [1](#)

sqlerrm auswerten [1](#)

SQLErrText [1](#)

SQLNRows [1](#)

Standard-Layout festlegen [1](#), [2](#)

Standardmethode [1](#)

Standardwert [1](#)

Standardwert : \ festlegen [1](#)

Standardwert : \ für Variablen [1](#)

start with-Klausel [1](#)

Statistikfunktion, integrierte [1](#)

Statuszeile [1](#)

stddev [1](#)

Steuerblock [1](#)

Steuerelement hinzufügen [1](#)

Stored Program Unit Editor [1](#)

Structured Query Language [1](#)

substr [1](#)

Suchfeld, Anzeige [1](#)

sum [1](#)

Sun [1](#)

Support [1](#)

Sybase [1](#), [2](#)

Synonym [1](#)

Synonym : öffentliches [1](#)

Synonym : privates [1](#)

Synonym : Syntax [1](#)

Synonym : \ erzeugen [1](#)

Synonym : \ löschen [1](#), [2](#)

Synonym : \ untersuchen [1](#)

Syntax [1](#)

sys [1](#)

sysdate [1](#)

system [1](#)

System Design [1](#)

System-Berechtigung [1](#)

Systemleistung, Hindernisse [1](#)

Systemmodellierung [1](#)

## - T -

Tabelle [1](#)

Tabelle : Auswahl aus einer anderen [1](#)

Tabelle : Integritätsregeln festlegen [1](#)

Tabelle : mögliche Trigger [1](#)

Tabelle : Rückgabewerte [1](#)

Tabelle : Spalten festlegen [1](#)

Tabelle : Synonym [1](#)

Tabelle : virtuelle [1](#)

Tabelle : \ ändern [1](#)

Tabelle : \ benennen [1](#)

Tabelle : \ erstellen [1](#), [2](#), [3](#)

Tabelle : \ verbergen [1](#)

Tabellen, verschachtelte [1](#)

Tabellendefinition [1](#)

Tabelleneigentümer [1](#)

Tabellenindex [1](#)

Tabellen-Scan, vollständiger [1](#)

table [1](#)

Teilzeichenfolge [1](#)

Terminal [1](#)

Terminalemulator [1](#)

Terminalkonzentrator [1](#)

Testdaten erzeugen [1](#)

Text : \ ändern [1](#)

Text : \ formatieren [1](#)

Titel [1](#)

tkprof [1](#)

to\_char [1](#), [2](#)

to\_date [1](#)

to\_number [1](#)

too\_many\_rows [1](#)

Transaktion [1](#)

Trigger [1](#), [2](#), [3](#)

Trigger : Abrufen von Werten [1](#)

Trigger : ausführen einer Abfrage [1](#)

Trigger : before [1](#)

Trigger : Benutzereingaben prüfen [1](#)



Trigger : Format- [1](#)

Trigger : kaskadierende [1](#)

Trigger : Prozeduren aufrufen [1](#)

Trigger : Rumpf [1](#)

Trigger : Setzen von Spaltenwerten [1](#)

Trigger : Sicherheit erzwingen [1](#)

Trigger : Spaltenwerte prüfen [1](#)

Trigger : Standard [1](#)

Trigger : Verwendungsmöglichkeiten [1](#)

Trigger : \ after [1](#)

Trigger : \ aktivieren [1](#)

Trigger : \ auf Anweisungsebene [1](#)

Trigger : \ auf Zeilenebene [1](#)

Trigger : \ deaktivieren [1](#)

Trigger : \ erstellen [1](#)

Trigger : \ für eine Tabelle [1](#)

Trigger : \ löschen [1](#)

true [1](#)

trunc [1](#)

truncate table [1](#)

## - U -

Uhrzeit [1](#)

Umkehrtechnik [1](#)

union [1](#)

Unique-Integritätsregeln [1](#)

Unique-Integritätsregeln : Deklaration [1](#)

UnixWare [1](#)

Unterabfrage [1](#), [2](#)

Unterabfrage : Syntax [1](#)

Unterabfrage : \ in einer update-Anweisung verwenden [1](#)

Unterblock verwenden [1](#)

update [1](#), [2](#)

update : Syntax [1](#)

upper [1](#)

USER\_OBJECTS [1](#)

## - V -

value\_error [1](#)

varchar [1](#)

varchar2 [1](#)

Variable : Gültigkeitsbereich [1](#)

Variable : \ deklarieren [1](#)

Variable : \ initialisieren [1](#)

Variablendeklaration [1](#)

variance [1](#)

varray [1](#)

Verbindung [1](#)

Verteilungsoption [1](#)

Vorwärtsdekларation [1](#)

vsize [1](#)

## **- W -**

Warnung [1](#)

Wartungsfreundlichkeit [1](#)

Web Agent [1](#)

Web Application Server Administration [1](#)

Web Listener [1](#), [2](#)

Web Request Broker [1](#), [2](#)

Web Request Broker : \ verwalten [1](#)

Web-Anwendungen : \ mit anderen Softwarepaketen [1](#)

Web-Anwendungen : \ mit Java [1](#)

Web-Anwendungen : \ mit PL/SQL [1](#)

Web-Browser [1](#)

Wert : boolscher [1](#)

Wert : \ einschränken [1](#)

Wert : \ ermitteln [1](#)

Werteliste [1](#)

Werteliste : \ erstellen [1](#)

where-Klausel [1](#)

where-Klausel : Kriterien festlegen [1](#)

while-loop [1](#)

Wiederverwendbarkeit [1](#)

Windows NT [1](#)

World Wide Web [1](#)

## **- Z -**

Zahl : Umwandeln in eine Zeichenfolge [1](#)

Zahl : \ abschneiden [1](#)

Zahl : \ runden [1](#)

Zahl : \ speichern [1](#)

Zahlen [1](#)

Zeichen umwandeln [1](#)

Zeichenfolge : Länge ermitteln [1](#)

Zeichenfolge : Muster [1](#)

Zeichenfolge : Rückgabe [1](#)

Zeichenfolge : Umwandlung in Datumsangaben [1](#)

Zeichenfolge : Umwandlung in eine Zahl [1](#)

Zeichenfolge : \ auffüllen [1](#)

Zeichenfolge : \ bearbeiten [1](#)

Zeichenfolge : \ ersetzen [1](#)

Zeichenfolgen [1](#)

Zeichenfolgen : lange [1](#)

Zeichenfolgen : \ einbetten [1](#)

Zeichenfolgen : \ kombinieren [1](#)

Zeile [1](#), [2](#)

Zeile : doppelte [1](#)

Zeile : eindeutige [1](#)

Zeilen : alle löschen [1](#)

Zeilen : \ ermitteln [1](#)

Zeilen : \ gruppieren [1](#)

Zeilen : \ hinzufügen [1](#)

Zeilen : \ suchen [1](#)

Zeilen : \ zählen [1](#)

Zeilenanzahl ermitteln [1](#)

Zeitangabe [1](#)

Zeitinformation [1](#)

Zeitinformation : \ speichern [1](#)

Zeitraum berechnen [1](#)

Zugriffsberechtigung [1](#)

Zuweisungsanweisung [1](#)

---

*Ein Imprint des Markt&Technik Buch- und Software-Verlag GmbH.  
Elektronische Fassung des Titels: Oracle in 21 Tagen, ISBN: 3-8272-2017-3*

 **Top** | **Inhalt**

# Wir sind an Ihrer Meinung interessiert!

Als Teil unseres Bestrebens, Bücher höchster Qualität zu produzieren, ist Pearson Education Deutschland an Ihrer Meinung interessiert.

Sie können Ihre Kommentare, Ideen oder Vorschläge zur Verbesserung zukünftiger Ausgaben an die unten angegebene Adresse oder per Fax an 089-46003-100 schicken. Online-Verbindung bekommen Sie über das Internet <http://www.pearsoneducation.de>.

Schon jetzt vielen Dank. Ihre Kommentare werden uns dabei helfen, weiterhin die besten Fachbücher herauszugeben.

## Markt+Technik

Pearson Education Deutschland  
[Information](#)  
Martin Kollar Str. 10-12  
80804 München

## Fragen zur HTML-Version?

Pearson Education Deutschland  
[Support](#)  
Martin Kollar Str. 10-12  
80804 München

## Wichtiger Hinweis:

Alle unsere Titel unterliegen dem Copyright. © 2001 by Pearson Education Deutschland

Alle Rechte vorbehalten - All rights reserved.

Die Vervielfältigung und / oder die Veröffentlichung ist nur mit der ausdrücklichen Genehmigung des Verlages gestattet.