Logischer Datenbankentwurf

Die Grundlage aller Datenbankanwendungen 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 Entity-Relationship-Modells, eines Modells, das Tabellen (Entity) und deren Beziehungen (Relationship) zueinander darstellt. Tabellen können Personen, Orte, Gegenstände oder Begriffe abbilden. Jede Tabelle wird anhand einer Reihe von Attributen beschrieben. In der Beispieldatenbank ist ein Kursleiter durch eine Anzahl von Attributen, wie z.B. Instructor_ID (Kursleiter-Identifikation), Last_Name (Nachname), First_Name (Vorname) und Department (Fachbereich) abgebildet. Der Administrator eines Fachbereichs benötigt eine Reihe weiterer Attribute für denselben Kursleiter, z.B. seine derzeitige Position. Wie Sie sehen, stellen die Anforderungen des Unternehmens den Motor für das Datenmodell dar. Tabellen existieren nicht im leeren Raum. Zwischen den Tabellen gibt es Beziehungen. Diese Beziehungen werden u.a. verwendet, um Integritätsbedingungen darzustellen. 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 der Implementierung 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 die Implementierung.

3.1 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.

Das Konzept des **NULL**-Werts

Ein großer Unterschied zwischen einer relationalen Datenbank und der Technologie älterer Datenbanken liegt im Konzept des **NULL**-Werts. In einer nichtrelationalen Datenbank gibt ein bestimmter Wert das Fehlen eines Werts in einem Attribut an.

Die Kennzeichnung **NULL** gibt an, dass der Wert einer Spalte oder eines Ausdrucks nicht verfügbar ist oder nicht zugewiesen wurde.

In einer relationalen Datenbank stellt der **NULL**-Wert einer Spalte verschiedene Konzepte dar:

- Ein Wert für diese Spalte ist in der betreffenden 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.

Es folgt eine kurze Zusammenfassung weiterer Begriffe aus dem Bereich relationaler Datenbanken:

- Jede Tabelle verfügt über eine Reihe von Attributen, welche die Tabelle beschreiben. Eine Tabelle namens `course` (Kurs) würde die in einem College angebotenen Kurse beschreiben.
- Ein Tupel ist eine Einzelinstanz von Attributwerten. Ein Tupel der Tabelle `course` beschreibt z.B. einen einzelnen vom College angebotenen Kurs.
- Einige Attribute identifizieren die einzelnen Tupel einer Tabelle eindeutig. Diese Attribute bezeichnet man als *Primärschlüssel*. Für die Tabelle, welche die Studenten beschreibt, ist das Attribut `Student_ID` (Studenten-Identifikation) der Primärschlüssel, da es die einzelnen Studenten eindeutig identifiziert.
- Keines der Attribute, die den Primärschlüssel bilden, kann einen **NULL**-Wert haben.
- Ein Fremdschlüssel ist eine Attributgruppe (ein oder mehrere Attribute) einer Tabelle, deren Werte als Primärschlüssel in einer anderen Tabelle vorkommen müssen.
- Tabellen werden zueinander in Beziehung gesetzt.
- Die Reihenfolge der Tupel innerhalb einer Tabelle ist beliebig.
- Die Reihenfolge der Attribute einer Tabelle ist beliebig.

- Unter *Schema* versteht man eine Gruppe von Datenbankobjekten wie z.B. Tabellen, Spalten, Primärschlüssel, Fremdschlüssel, die ein logisches Datenmodell implementieren.

Eine Tabelle besteht aus Zeilen und Spalten

Aus der Sicht eines Praktikers wird eine Tabelle im Entity-Relationship-Modell als konkretes Datenbankobjekt vom Typ `TABLE` in einem RDBMS implementiert. Die Attribute einer Tabelle werden als Tabellenspalten implementiert. Tupel entsprechen einem einzelnen Satz Attribute oder Spaltenwerte und werden als *Zeile* bezeichnet. Die Begriffe *Zeile* und *Datensatz* haben dabei dieselbe Bedeutung.

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

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

Ein *Datensatz* ist dasselbe wie eine Zeile – es handelt sich um eine einzelne Gruppe von Attributen 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 steuern, in der die Zeilen einer Tabelle abgerufen werden, ist es, diese Reihenfolge explizit in der Abfrage vorzugeben. Das Konzept der Unabhängigkeit vom Speicherort eines Tupels 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, die Kapazität einer 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 relationalen Datenbank-Managementsystems (RDBMS) von Oracle. Eine relationale Datenbank soll für *logische Datenunabhängigkeit* sorgen, da die Definitionen der einzelnen Spalten voneinander unabhängig sind.

3.2 Datenintegrität

Nach der relationalen Theorie verfügt jede Tabelle über eine Reihe von Attributen, welche die einzelnen Tupel dieser Tabelle eindeutig identifizieren. Die relationale Theorie sagt auch aus, dass in einer Tabelle keine Tupel 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 Tabelle verfügt über eine Gruppe von Attributen, die einen Datensatz eindeutig identifizieren. Diese Attributgruppe wird als Primärschlüssel bezeichnet. Der Primärschlüssel kann aus einem einzelnen Attribut – ein Student wird durch Student ID (Studenten-Bezeichnung) eindeutig identifiziert – oder mehreren Attributen bestehen. Manchmal ist es offensichtlich, welche Attribute den Primärschlüssel bilden, manchmal 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 einem oder mehreren Attributen, die eine Zeile eindeutig identifizieren.

Kein Teil des Primärschlüssels hat einen NULL-Wert

Ein Grundsatz der relationalen Theorie ist, dass kein Teil des Primärschlüssels einen **NULL**-Wert haben kann. Wenn Sie eine Weile darüber nachdenken, erscheint diese Tatsache intuitiv klar. Der Primärschlüssel muss die einzelnen Tupel einer Tabelle eindeutig identifizieren. Wenn der Primärschlüssel (oder ein Teil davon) einen **NULL**-Wert 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 Beziehung 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 **NULL**-Wert eingeschränkt werden. Hat der Datenbankentwickler Primär- und Fremdschlüssel erst einmal deklariert, ist die Gewährleistung 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 Tabellen wird als Beziehung bezeichnet. In einer Beziehung wird eine Tabelle als übergeordnet und die andere als untergeordnet bezeichnet. Eine Beziehung zwischen den Tabellen A und B kann aus der Sicht der Tabelle A oder aus der Sicht der Tabelle B betrachtet werden. Es ist denkbar, dass je nach Betrachtungsrichtung die eine bzw. die andere Tabelle als untergeordnet gilt. Ein Beispiel für eine Beziehung mit je nach Betrachtungsrichtung unterschiedlicher untergeordneter Tabelle ist die Beziehung zwischen Auftrag und Auftragsposition. Eine Auftragsposition kann nicht ohne Auftrag existieren und gilt folglich als untergeordnet. Ein Auftrag ohne Auftragspositionen ist nicht sinnvoll, daher kann der Auftrag seinerseits als untergeordnet gelten.

Eine Beziehung wird durch folgende Merkmale definiert:

- **Identifizierend oder nichtidentifizierend:** Bei einer nichtidentifizierenden Beziehung bildet der Primärschlüssel der übergeordneten Tabelle einen Teil des Primärschlüssels der untergeordneten Tabelle. In einer identifizierenden Beziehung ist der Primärschlüssel der übergeordneten Tabelle nicht Teil des Primärschlüssels der untergeordneten Tabelle.

- **Kardinalität:** Die *Kardinalität* einer Beziehung bezeichnet die Anzahl der Zeilen einer untergeordneten Tabelle, die sich auf eine einzelne Zeile einer übergeordneten Tabelle beziehen. Die Kardinalität kann die Werte null (0), eins (1) oder mehrere (n) annehmen. Zum Beispiel kann ein Kursleiter, der sich mit Grundlagenforschung beschäftigt, keine oder mehrere Klassen unterrichten. Eine Fachabteilung muss jedoch einen oder mehrere Kurse anbieten, andernfalls wäre sie keine Fachabteilung.
- **Erforderlich oder optional:** Eine Beziehung ist erforderlich, wenn sich eine Zeile in einer untergeordneten Tabelle auf eine Zeile in einer übergeordneten Tabelle beziehen muss oder umgekehrt. Die Beziehung ist optional, wenn eine Zeile in einer untergeordneten Tabelle ohne Bezug zu einer Zeile in einer übergeordneten Tabelle auskommt oder umgekehrt. Bei einer optionalen Beziehung darf der Fremdschlüssel der untergeordneten Tabelle den **NULL**-Wert annehmen.

  Die oben erwähnte Beziehung zwischen Auftrag und Auftragsposition ist ein Beispiel für eine Beziehung, die von beiden Seiten als erforderlich modelliert werden kann.
- **Integritätsregeln:** Diese Regeln definieren, welche Maßnahmen ergriffen werden müssen, wenn eine Zeile in einer übergeordneten oder untergeordneten Tabelle auf irgendeine Weise geändert wird. Es gibt sechs Möglichkeiten: Eine Zeile der übergeordneten Tabelle wird hinzugefügt, geändert oder entfernt, oder eine Zeile der untergeordneten Tabelle wird hinzugefügt, geändert oder entfernt. Für jede dieser Möglichkeiten gibt es mehrere anwendbare Maßnahmen – die Aktion zu verhindern, den Fremdschlüsselwert in der untergeordneten Tabelle auf den **NULL**-Wert zu setzen, die Operation in Bezug auf die untergeordnete Tabelle zu kaskadieren oder den Schlüsselwert der über- oder untergeordneten Tabelle auf einen Standardwert zu setzen. Am Beispiel des Löschens in der übergeordneten Tabelle bedeutet Kaskadieren, dass in allen untergeordneten Tabellen diejenigen Datensätze, die sich auf den zu löschenden Datensatz in der übergeordneten Tabelle beziehen, ebenfalls gelöscht werden.

Normalisierung

Es ist sehr einfach, eine Tabelle in einer Oracle-Datenbank zu erstellen. Sie können dazu aus einer Vielzahl von Werkzeugen auswählen – Enterprise Manager, SQL*Plus, SQL*Plus Worksheet u.a. Am Tag 4, »Implementierung des 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 Tabellen, Attributen (Spalten) und der Abhängigkeit der Attribute untereinander. Die Ziele der Normalisierung lauten wie folgt:

- Minimieren redundanter Daten
- Vermeiden von Änderungsanomalien
- Verhindern 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 viele verschiedene Begriffe. Wenn Sie diese Lektion lesen, werden Sie feststellen, dass einige der Begriffe synonym 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.

Theoretiker	Analytiker	Entwickler
Relation	Entity	Tabelle
Attribut	Attribut	Spalte
Tupel	Zeile	Zeile/Datensatz

Tabelle 3.1: Vergleich der Datenbankterminologie

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 Relation (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 Relation besteht. Wenn zum Beispiel eine Relation zum Speichern von Informationen über Mitarbeiter bestimmt ist, würden Sie kein Einzelattribut, Dependents (Untergebene), verwenden, um die Namen der Untergebenen eines Mitarbeiters zu speichern. Einige Mitarbeiter könnten keine, andere dagegen viele Untergebene haben.

Normalisierungsregel 2: Alle Tupel hängen nur vom Primärschlüssel ab

Bei einer Relation 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 der Tabelle Class ist die 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. Angenommen, der Primärschlüssel bestünde aus der Department ID und der Course ID, weil die Course ID nur innerhalb eines Fachbereiches eindeutig ist. Beide Spalten wären notwendig, um eine Zeile in der Tabelle eindeutig zu identifizieren. 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 würde der Fachbereichsleiter nur von einem Teil des Primärschlüssels, jedoch nicht vollständig vom Primärschlüssel abhängen.

Manchmal wird dieses Konzept auch als *abgeleitete Spalte* bezeichnet. Im obigen Beispiel kann Department Chairperson (Fachbereichsleiter) 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	Was nicht
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.	
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.

3.3 Werkzeuge für die Erstellung von Entity-Relationship-Diagrammen

Um Ihnen bei der Entwicklung eines logischen Datenmodells behilflich zu sein, sollten Sie ernsthaft den Einsatz eines Werkzeugs für die Erstellung von Entity-Relationship-Diagrammen, wie z.B. Oracle Designer, ERwin von LogicWorks oder Sybase S-Designer, in Erwägung ziehen. Alle diese Werkzeuge sind für Microsoft Windows Systeme erhältlich. Kurz gesagt ermöglichen Ihnen diese Werkzeuge, ein Entity-Relationship-Diagramm zu erstellen, indem Sie Symbole aus einer Werkzeugleiste auswählen und Beziehungen zwischen diesen Symbolen zeichnen.

Diese Werkzeuge haben einige Vorzüge:

- Sie vereinfachen den Prozess der Diagrammerstellung.
- Sie erzeugen automatisch die SQL-Anweisungen für die Erstellung von Datenbankobjekten wie Tabellen, Indizes und Integritätsbedingungen.

- Sie erlauben Ihnen, die einzelnen Tabellen, Attribute, Beziehungen und Integritätsbedingungen zu dokumentieren. Diese Werkzeuge erzeugen außerdem Berichte, die von anderen Entwicklern und Benutzern verwendet werden können, damit Sie Rückmeldungen über die Genauigkeit des Datenmodells erhalten.

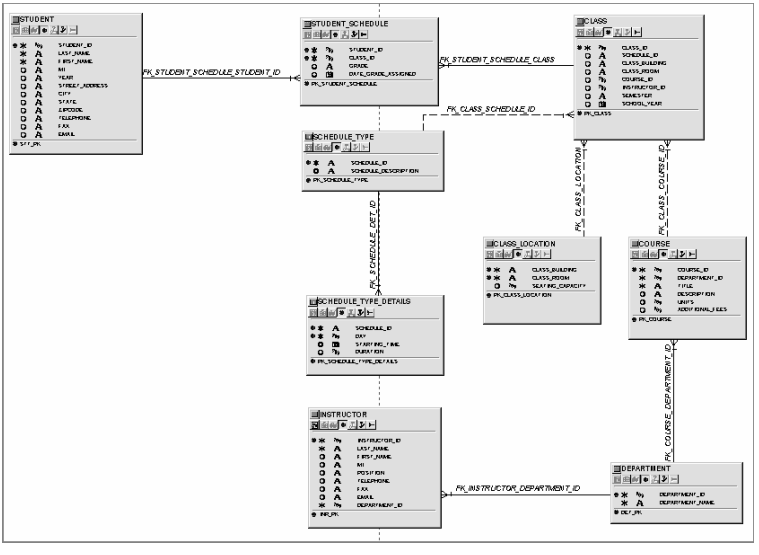

Abbildung 3.1: Ein mit Oracle Designer erstelltes Entity-Relationship-Diagramm

3.4 Star Schema

Der Datenbankentwurf für ein Data Warehouse ist gewöhnlich ein so genanntes Star Schema (sternförmiges Entity-Relationship-Modell). Das Star Schema hat seinen Namen von der sternförmigen Anordnung mehrerer Dimensionstabellen rund um eine Faktentabelle. Eine Faktentabelle enthält Maßzahlen für das Geschäftsergebnis eines Unternehmens, z.B. Verkäufe. Jede Maßzahl ist über Fremdschlüsselbeziehungen in Bezug zu Dimensionen gesetzt, welche eine Maßzahl genau charakterisieren. Ein Verkauf ereignet sich z.B. in einem bestimmten Quartal (Zeitraum) in einer bestimmten Region und wird mit einem Kunden getätigt. Zeitraum, Region und Kunde sind in diesem Beispiel Dimensionstabellen. Zwischen einer Dimension und der Faktentabelle besteht jeweils eine 1:n Beziehung, d.h. ein Fremdschlüssel in der Faktentabelle bezieht sich auf einen Primärschlüssel in einer Dimensionstabelle.

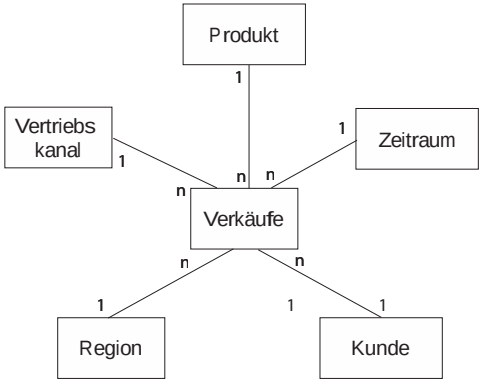

Abbildung 3.2:
Star Schema Datenmodell eines Data Warehouse

3.5 Das Beispielschema

Dieses Buch verwendet ein Datenbankschema für ein kleines, fiktives College als Beispiel. 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 Papier. Der Dekan des College hat Ihnen den Auftrag für die Entwicklung eines computergestützten Informationssystems auf der Basis einer Client-Server-Architektur erteilt.

Anforderungen

Die Anforderungen an das Informationssystem des Flugle-College 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 Anforderungen werden folgende Tabellen benötigt, um das Informationsbedürfnis des Flugle-College zu befriedigen:

- Die Student-Tabelle enthält Informationen über die einzelnen Studenten dieses College.
- Die Department-Tabelle umfasst Informationen über die einzelnen Fachbereiche des College.
- 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 wie viel 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.

Student
Student ID Last Name First Name Middle Initial (Anfangsbuchstabe des zweiten Vornamens) Street Address (Straße) City (Ort) State (Bundesland) Zipcode (PLZ) Telephone Fax E-Mail Year in School (Semester) **Primärschlüssel**: 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.

Department
Department ID Description (Beschreibung) **Primärschlüssel:** 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.

Instructor
Instructor ID Department ID (Fremdschlüssel zur Department-Tabelle)

Instructor
Last Name First Name Middle Initial Position Telephone Fax E-Mail **Primärschlüssel:** Instructor ID

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

Die Course-Tabelle

Jeder Fachbereich bietet eine Reihe von Kursen an. Jeder Kurs hat eine eindeutige Course ID. Weitere Attribute eines Kurses sind Titel, Beschreibung, Anzahl der Lerneinheiten und eventuell erforderliche Zusatzgebühren.

Course
Department ID (Fremdschlüssel zur Department-Tabelle) Course ID Title (Titel) Description Units (Lerneinheiten) Additional Fees (zusätzliche Gebühren) **Primärschlüssel:** Course ID

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.

Class Location (**Veranstaltungsort**)
Class Building (Gebäude) Class Room (Klassenzimmer) Seating Capacity (Anzahl der Sitzplätze) **Primärschlüssel:** 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.

Schedule Type (**Stundenplantyp**)
Schedule ID Description **Primärschlüssel:** 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.

Schedule Type Details (**Einzelheiten zum Stundenplantyp**)
Schedule ID (Fremdschlüssel zur Schedule-Type-Tabelle) Day (Wochentag) Starting Time (Anfangszeit) Duration (Dauer) **Primärschlüssel:** 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.

Class
Class ID 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) **Primärschlüssel:** 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.

Student Schedule (**Stundenplan des Studenten**)
Student ID (Fremdschlüssel zur Student-Tabelle) Class ID (Fremdschlüssel zur Class-Tabelle) Grade (Zensur) Date Grade Assigned (Datum der Zensurvergabe) **Primärschlüssel:** Student ID, Class ID

3.6 Terminologie zu Oracle

Bevor Sie am Tag 4 lernen, wie man eine Tabelle erstellt, müssen Sie sich mit der verwendeten Terminologie vertraut machen. Tabellen werden von Oracle in so genannten Tablespaces abgelegt.

Ein *Tablespace* ist ein Behälter für Datenbankobjekte. Ein *Tablespace* besteht aus bis zu 1022 Datendateien. Oracle kann Dateien in einem Dateisystem oder nicht mit einem Dateisystem formatierte Plattenbereiche, so genannte Raw Devices, als Datendatei nutzen.

Oracle9*i* kann auf den UNIX und Windows Plattformen Datendateien mit einer Größe von mehr als 4 GB verwalten. Wenn Sie Tablespaces nach dem Muster in Listing 3.1 anlegen, verwendet Oracle Bitmaps, um eine performante Freispeicherverwaltung auf dem Tablespace selbst (Locally Managed Tablespace) und den enthaltenen Datenbankobjekten (Automatic Segment Space Management) zu implementieren.

Listing 3.1: Anlegen eines Tablespace

```
CREATE TABLESPACE tablespace_name DATAFILE 'file_spec' SIZE nM [AUTOEXTEND ON
[NEXT kM] [MAXSIZE mM]] EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;
```

Die Bedeutungen der Platzhalter sind:

- *tablespace_name* ist der Name des Tablespace.
- *file_spec* ist der absolute Pfad einer Datei oder eines Raw Device, z.B. `d:\oradata\users.dbf` unter Windows oder `/oradata/users.dbf` unter UNIX.
- *n* ist die Größe der Datei bzw. des Raw Device (abzüglich eines Oracle Blocks) in MB.

Die Autoextend-Klausel kann für Datendateien in Dateisystemen verwendet werden und bewirkt, dass die Datei bei Platzmangel automatisch um *k* MB vergrößert wird, bis zu einer maximalen Größe von *m* MB.

Einige der wichtigsten Begriffe sind *Datenbank, Instanz, Benutzer und Datenbanksitzung*. Eine Oracle-Datenbank setzt sich aus Tablespaces und weiteren Dateien für besondere Aufgaben zusammen. Die Änderungen an der Datenbank werden in Redo Logs protokolliert. Um Änderungen rückgängig machen zu können, hebt Oracle vormals gültige Daten in Undo Segmenten in einem Undo Tablespace auf. Die Kontrolldatei enthält Verwaltungsinformation für die Datenbank. Diese Verwaltungsinformation beinhaltet z.B. welche Datendateien und Tablespaces zur Datenbank gehören.

Eine Oracle-*Datenbank* besteht aus dem Datenbankkatalog (Data Dictionary) im SYSTEM Tablespace, Kontrolldatei(en) (Controlfile), Änderungsprotokolldateien (Online Redo Logs) und weiteren Tablespaces für Temporärsegmente (z.B. TEMP), Undo Segmente (z.B. UNDOTBS) und Benutzerspeicherbereiche (z.B. Tablespace USERS).

Die maximale Größe einer Oracle-Datenbank liegt bei mehr als einem Petabyte (1024 Terabyte). Online Redo Logs werden periodisch überschrieben. Um die Wiederherstellbarkeit der Datenbank z.B. nach einem Festplattendefekt zu gewährleisten, sollte eine Oracle-Datenbank im ARCHIVELOG-Modus betrieben werden. In diesem Modus werden Offline Redo Logs, das sind exakte Kopien der Online Redo Logs, erstellt. Offline Redo Logs werden nicht von Oracle überschrieben. Die gesamte Änderungshistorie einer Datenbank kann so über Monate und Jahre aufbewahrt werden.

Eine *Oracle-Instanz* besteht aus Prozessen und Speicherstrukturen. Eine Instanz öffnet genau eine Datenbank. Jeder Zugriff auf eine Oracle-Datenbank wird durch eine Oracle-Instanz durchgeführt. Jede Oracle-Instanz ist durch die Umgebungsvariable ORACLE_SID (**O**racle **S**ystem **ID**entifier) eindeutig identifiziert.

Eine Oracle-Instanz ist auf Windows Systemen durch das Programm `oracle.exe` und auf UNIX Systemen durch das Programm `oracle` implementiert. Die Programme oracle.exe bzw. oracle liegen in einem Verzeichnisbaum, der bei der Installation der Oracle Distribution vom Oracle Installer aufgebaut wird. Die Wurzel dieses Verzeichnisbaums wird als `ORACLE_HOME` bezeichnet. Auf Windows Systemen wird das `ORACLE_HOME` in die Registry eingetragen. Auf UNIX Systemen ist `ORACLE_HOME` eine Umgebungsvariable und das Programm `oracle` befindet sich in `$ORACLE_HOME/bin`.

Eine Oracle-Instanz kann sich in Bezug auf die Datenbank in verschiedenen Zuständen befinden. Man unterscheidet drei Phasen, die durchlaufen werden müssen, um Änderungen an einer Datenbank durchzuführen. Die drei Phasen und die Befehle, um in die jeweilige Phase zu gelangen, sind:

1. `STARTUP NOMOUNT` – die Initialisierungsparameter werden gelesen und die Instanz wird gestartet.
2. `ALTER DATABASE MOUNT` – die Kontrolldatei wird geöffnet und es wird ermittelt, welche Datei den Datenbankkatalog enthält sowie welche weiteren Dateien zur Datenbank gehören.
3. `ALTER DATABASE OPEN` – die Datendateien der Tablespaces werden geöffnet. Eine Redo Log Datei (Current Redo Log) wird geöffnet, um Änderungen an der Datenbank zu protokollieren.

Sobald Phase 3 durchlaufen wurde, können Benutzer (z.B. FLUGLE) mit den Tabellen arbeiten. Zwei weitere Befehle zum Starten einer Instanz sind bedeutsam. Mit `STARTUP MOUNT` erreichen Sie Phase 2, mit `STARTUP` unmittelbar Phase 3. Normalerweise wird die Datenbank in Phase 3 zum Schreiben und Lesen geöffnet (OPEN READ WRITE). Es gibt aber auch die Möglichkeit, mit `ALTER DATABASE OPEN READ ONLY` die Datenbank ausschließlich zum Lesen zu öffnen. Sie sind somit in der Lage, Tablespaces einer Oracle-Datenbank auf nicht beschreibbaren Medien wie z.B. CD-ROM zu legen. Lediglich die Kontrolldatei muss immer auf einem beschreibbaren Medium liegen.

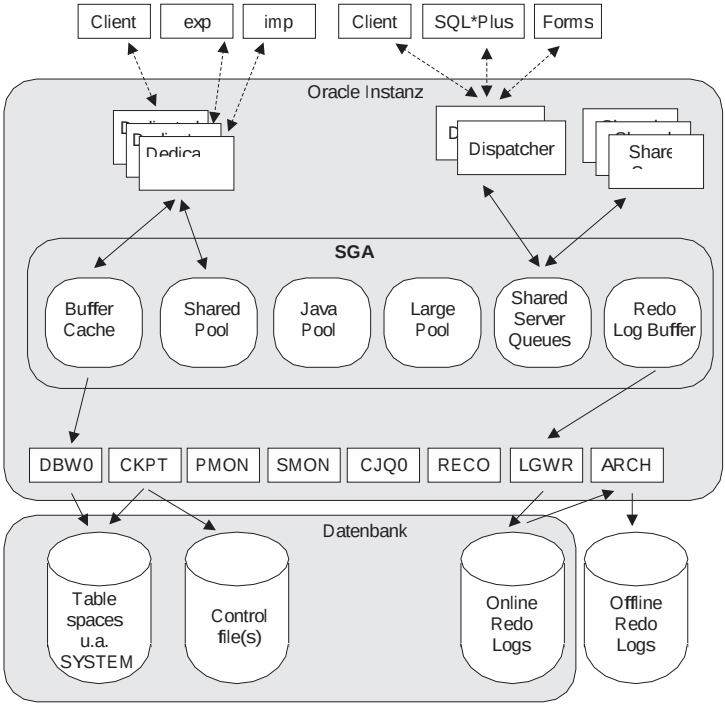

Abbildung 3.3: Architektur des Oracle RDBMS mit Anwendungsprogrammen (oben), der Oracle-Instanz (Mitte) und der Datenbank (unten). Pfeile stellen einen kleinen Ausschnitt der funktionalen Beziehungen zwischen den Komponenten dar

Die Prozesse einer Oracle-Instanz werden weiter in die Kategorien Hintergrundprozess, Serverprozess und Dispatcher untergliedert. Anwendungsprogramme kommunizieren mit einem Dispatcher oder einem dedizierten Serverprozess. Zu den Hintergrundprozessen zählt z.B. der Database Writer (DBW0), der veränderte Datenbankblöcke aus dem als System Global Area (SGA) bezeichneten Speicherbereich der Instanz auf einen Festplattenspeicher schreibt. Der Hintergrundprozess Log Writer (LGWR) schreibt Änderungsprotokolle aus dem Log Buffer in der SGA in die aktuelle Online Redo Log Datei.

Zum Starten einer Oracle-Instanz wird eine *Initialisierungsdatei* benötigt. Der Name der Datei auf Windows ist `<ORACLE_HOME>\database\init<ORACLE_SID>.ora`. Auf UNIX Systemen wird die Initialisierungsdatei `$ORACLE_HOME/dbs/init$ORACLE_SID.ora` verwendet.

Zumindest folgende Initialisierungsparameter müssen in der init<ORACLE_SID>.ora Datei enthalten sein, damit die Instanz eine Datenbank öffnen kann:

- db_name – der Name der Oracle-Datenbank
- control_files – absoluter Pfad einer oder mehrerer Kontrolldateien. Aus den Kontrolldateien geht hervor, welche weiteren Dateien zur Datenbank gehören.
- remote_login_passwordfile – gibt an, ob eine Passwortdatei und auf welche Art die Passwortdatei verwendet wird. Auf Windows sollte der Parameter auf `exclusive` stehen. Auf UNIX ist der Parameter optional und eine Passwortdatei nur dann erforderlich, wenn eine Fernverbindung zur Instanz als Benutzer `SYS` aufgebaut werden soll.

Bedeutsam ist der Parameter `db_block_size`, denn es handelt sich um den einzigen Parameter, der nach dem Erzeugen der Datenbank nicht mehr verändert werden kann. Er gibt die Größe eines Datenbankblocks an und kann die Werte 2048, 4096, 8192 und 16384 annehmen. Oracle9*i* gestattet Ihnen die Blockgröße jedes Tablespace individuell beim CREATE TABLESPACE anzugeben.

Ein *Datenbankbenutzer* hat einen Namen, ein Kennwort, ein Schema und Berechtigungen, die im Oracle-Datenbankkatalog vermerkt sind. Legt ein Datenbankbenutzer Datenbankobjekte an, werden diese standardmäßig in dem Datenbankschema erstellt, das denselben Namen hat wie der Benutzer.

Eine *Datenbanksitzung* ist ein Kommunikationskanal zwischen einem Anwendungsprogramm und einer Oracle-Instanz. Eine Datenbanksitzung hat bestimmte Attribute. Einige der Attribute beeinflussen die Ausführung von SQL-Anweisungen.

Um eine Datenbanksitzung aufzubauen, werden Benutzernamen und Kennwort an eine Oracle-Instanz gesendet. Über den Kommunikationskanal werden der Oracle-Instanz SQL-Anweisungen übermittelt und Ergebnisse empfangen.

Attribute einer Datenbanksitzung sind z.B. die Sitzungsnummer, Einstellungen für die Sprachumgebung (`NLS_LANG`), das Datumsformat (`nls_date_format`) und Nummernformat (`nls_numeric_characters`) sowie Parameter, die festlegen, welcher SQL-Optimierer (`optimizer_mode`) eingesetzt werden soll oder wie viel Speicherplatz für Sortiervorgänge (`sort_area_size`) verwendet werden soll. Der Anwender ist nicht mit diesen Einzelheiten konfrontiert, denn für jede mögliche Einstellung gelten Standardwerte, die vom Datenbankadministrator festgelegt werden.

Datenbanksitzung

Eine *Datenbanksitzung* mit Oracle herzustellen bedeutet, einen gültigen Benutzernamen und ein Kennwort anzugeben, die von der Oracle-Instanz akzeptiert werden. Mit Hilfe eines beliebigen Oracle-Werkzeugs, das mit einer der Oracle-Programmierschnittstellen erstellt wurde, kann ein Benutzer eine Sitzung zu einer Oracle-Instanz herstellen. Auch mit einem Drittwerkzeug wie PowerBuilder kann sich ein Benutzer bei einer Oracle-Instanz anmelden. Die Begriffe *Oracle-Sitzung*, *Oracle-Verbindung* und *Datenbanksitzung* werden oft gleichbedeutend verwendet.

Es ist möglich, eine Datenbanksitzung zu einer Oracle-Instanz aufzubauen, die auf demselben Rechner läuft wie das Anwendungsprogramm oder auf einem anderen Rechner in einem Netzwerk. Eine *lokale Verbindung* zu einer Oracle-Instanz herzustellen bedeutet, dass das vom Benutzer verwendete Programm auf demselben Computer ausgeführt wird wie die Oracle-Instanz. Der Begriff *Fernverbindung* bedeutet, dass sich die Oracle-Instanz auf einem anderen Rechner im Netzwerk befindet. Der Kommunikationskanal zwischen dem Anwendungsprogramm und der Oracle-Instanz wird immer von Oracle Net Services realisiert, unabhängig davon, ob mit einer lokalen Verbindung oder einer Fernverbindung gearbeitet wird.

Ein *Servicename* definiert auf Ebene der Oracle Net Services, wie der Kommunikationskanal für eine Datenbanksitzung mit einer Oracle-Instanz aufgebaut wird.

Die Verwendung eines Servicenamens ist zwingend, um eine Fernverbindung aufzubauen. Eine Servicename für eine Fernverbindung definiert z.B., dass das Protokoll TCP/IP verwendet werden soll, um einen Kommunikationskanal zu einer Oracle-Instanz aufzubauen.

Ein Listener ist ein Prozess, der Anforderungen für den Aufbau von Datenbanksitzungen entgegennimmt und den Verbindungsaufbau zwischen dem Anwendungsprogramm und der Oracle-Instanz bewerkstelligt.

Listener warten typischerweise auf dem Port 1521 darauf, dass ein Anwendungsprogramm über TCP/IP die Verbindung zu einer Oracle-Instanz nachfragt. Um welche Instanz es sich handelt, wird dem Servicenamen entnommen. Der Listener kennt die Instanz entweder, weil diese in der Listener Konfigurationsdatei listener.ora bekannt gegeben wurde, oder weil die Instanz beim Listener einen Service registriert hat. Der registrierte Service setzt sich aus den Werten der Initialisierungsparameter `db_name` und `db_domain` zusammen, oder wird dem Parameter `service_names` entnommen, falls dieser gesetzt ist. Einen Überblick über die Services und Instanzen, die einem Listener bekannt sind, erhalten Sie mit dem Kommando `lsnrctl services`.

Lokale Verbindungen benötigen keinen Listener. Eine lokale Verbindung wird hergestellt, wenn beim Verbindungsaufbau, z.B. mit dem Befehl `connect` in SQL*Plus, kein Servicename angegeben wird. In diesem Fall wird durch Auswertung der Variable `ORACLE_SID` ermittelt, zu welcher Oracle-Instanz eine Datenbanksitzung aufgebaut werden soll.

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 `sjensen`. Sally Jensen hat Zugang zu allen Tabellen des Schemas Flugle, besitzt aber keine eigenen Tabellen.

Jede in einer Oracle-Datenbank befindliche Tabelle gehört einem Eigentümer. 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 verwenden.

Externe Prozeduren

Oracle9*i* gibt einem Entwickler die Möglichkeit, Routinen, die in anderen Programmiersprachen geschrieben wurden, von einem PL/SQL-Unterprogramm aus aufzurufen. Diese Routinen, die in einer Programmiersprache der dritten Generation (3GL, z.B. C) geschrieben sind und sich in einer gemeinsam nutzbaren Bibliothek (shared library) befinden, werden als *externe Prozeduren* bezeichnet. Externe Prozeduren sind sinnvoll, wenn eine bestimmte Aufgabe nicht mit PL/SQL oder der Java Virtual Machine in der Oracle-Instanz realisiert werden kann oder eine externe Prozedur einen deutlichen Geschwindigkeitsvorteil einbringt.

3.7 Oracle Enterprise Manager

Oracle Enterprise Manager ist der Oberbegriff für eine Systemmanagementumgebung mit einer Vielzahl von Werkzeugen für die grafische Administration von Oracle-Produkten. Enterprise Manager unterstützt die Oracle9*i* Datenbank, Oracle9*i* Application Server, Oracle Applications und SAP/R3. Enterprise Manager setzt sich aus der Enterprise Manager Console und mehreren zum Teil zusätzlich zu lizenzierenden Packs zusammen. Einige der Packs sind Diagnostics Pack, Tuning Pack und Change Management Pack.

Enterprise Manager fällt in die Kategorie der Systemmanagement-Werkzeuge. Enterprise Manager ist als mehrbenutzerfähige Drei-Schichten-Architektur realisiert, kann aber mit Einschränkungen auch im Client-Server-Betrieb oder als Web-Anwendung eingesetzt werden. Alle für das Systemmanagement relevanten Informationen werden in einem Repository gehalten., das in einer Oracle-Datenbank abgelegt wird. Das Repository enthält Informationen über Administratoren und deren Arbeitszeiten, die verwalteten Rechnerknoten, Oracle-Instanzen u.v.a.m.

Die Anwendungsschicht bildet die Enterprise Manager Console oder eines der Packs. Die mittlere Schicht bildet der Oracle Management Server, der auf das Repository zugreift. Die dritte Schicht bilden so genannte Intelligent Agents, die auf den überwachten Rechnern laufen. Intelligent Agents liefern Statusinformationen über die Oracle-Instanzen an den Management Server, führen Aufgaben (Jobs) auf den Rechnern durch oder alarmieren bei bestimmten Ereignissen über den Management Server einen Administrator.

Weiterführende Informationen über Enterprise Manager finden Sie in der Oracle-Dokumentation, z.B. im »Oracle Enterprise Manager Concepts Guide«. Für unsere Zwecke reicht es aus, die Enterprise Manager Console im Client Server Betrieb zu nutzen. Management Server und Intelligen Agent müssen nicht gestartet werden.

Erzeugen eines neuen Benutzers

Als ersten Schritt bei der Erstellung der Beispieldatenbank werden Sie einen Oracle-Benutzer namens Flugle erzeugen, dem alle Datenbankobjekte gehören werden. Bevor Sie einen neuen Oracle-Benutzer erzeugen, müssen Sie eine Verbindung zu einer Oracle-Instanz herstellen. Die meisten Oracle-Werkzeuge stellen ein Dialogfenster bereit, um die Verbindung zu einer Oracle-Instanz herzustellen. Im Dialogfenster werden Sie aufgefordert, drei Eingaben vorzunehmen:

- den Oracle-Datenbankbenutzer, den Sie für Ihre Verbindung verwenden werden
- ein Benutzerkennwort
- einen Servicenamen (z.T. als 'Host String' gekennzeichnet)

Wenn Sie eine Verbindung zu einer Oracle-Instanz auf einem Server im Netzwerk herstellen möchten, müssen Sie einen Servicenamen angeben. Die Definition von Servicenamen führen Sie mit Oracle Net Manager durch.

Erzeugung eines neuen Benutzers mit der Enterprise Manager Console

Rufen Sie unter Windows über das Start Menü die Enterprise Manager Console auf. Auf UNIX Systemen erreichen Sie die Konsole, indem Sie den Befehl `oemapp console` in einer Shell eingeben. Den Befehl `oemapp` können Sie auch am MS DOS Command Prompt unter Windows verwenden. Akzeptieren Sie die Standardauswahl LAUNCH STANDALONE mit OK. Die Enterprise Manager Console erscheint (siehe Abbildung 3.4).

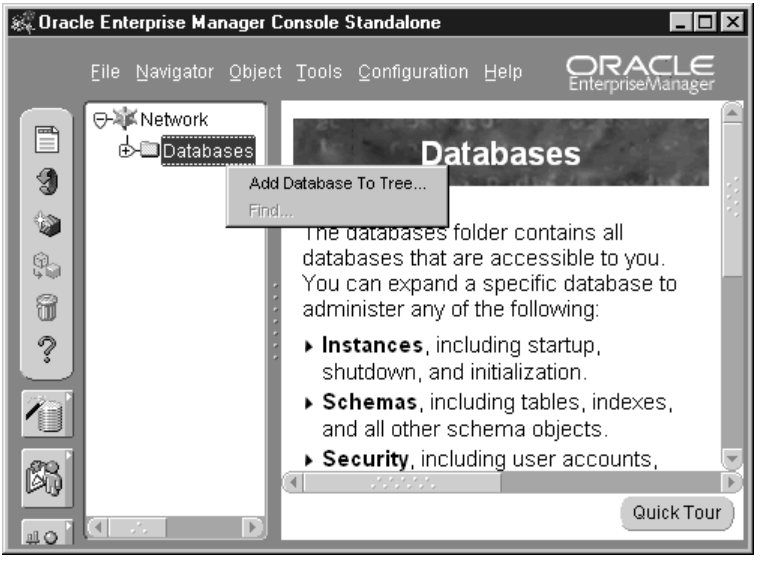

Abbildung 3.4: Hinzufügen einer Datenbank in der Enterprise Manager Console

Im linken Bereich des Fensters sehen Sie innerhalb der hierarchischen Darstellung des Netzwerks (NETWORK) den Knoten DATABASES. Klicken Sie mit der rechten Maustaste auf DATABASES und wählen Sie ADD DATABASE TO TREE. Ein Dialogfenster erscheint. Wählen Sie unter ADD SELECTED DATABASES... einen der Servicenamen aus der Konfigurationsdatei tnsnames.ora aus und bestätigen Sie mit OK.

Im Ast DATABASES erscheint jetzt der ausgewählte Servicename. Klicken Sie mit der rechten Maustaste auf den Servicenamen und wählen Sie CONNECT. Geben Sie den Benutzernamen 'system' ein. Wenn Sie das Standardpasswort nicht verändert haben, geben Sie 'manager' als Passwort ein. Bestätigen Sie mit OK. Nach erfolgreichem Aufbau einer Datenbanksitzung werden unter dem Servicename die Knoten INSTANCE, SCHEMA, SECURITY, usw. aufgeblendet.

Wählen Sie den Menüpunkt OBJECT und anschließend CREATE. Aus der Liste der Datenbankobjekte wählen Sie nun USER und bestätigen mit OK. Das Dialogfenster CREATE USER öffnet sich. Unter dem Kartenreiter GENERAL geben Sie Benutzernamen und Passwort ein. Unter TABLESPACES können Sie den Standard-Tablespace (DEFAULT) und den Tablespace für temporäre Segmente einstellen. Der Standard-Tablespace ist meistens 'USERS' und der Tablespace für temporäre Segmente entweder 'TEMP' oder <SYSTEM ASSIGNED>. System Assigned bedeutet, dass der Standard-Tablespace für temporäre Segmente, der beim Anlegen der Datenbank oder anschließend mit `alter database set default temporary tablespace` festgelegt wurde, für den Benutzer verwendet werden soll. Wenn eine Oracle9*i* Datenbank einen Standard-Tablespace für temporäre Segmente besitzt, kann es nicht mehr wie bei den Vorgängerversionen passieren, dass ein Benutzer versehentlich im SYSTEM Tablespace temporäre Segmente anlegt, weil vergessen wurde, der Benutzerdefinition den tatsächlichen Tablespace für temporäre Segmente zuzuweisen.

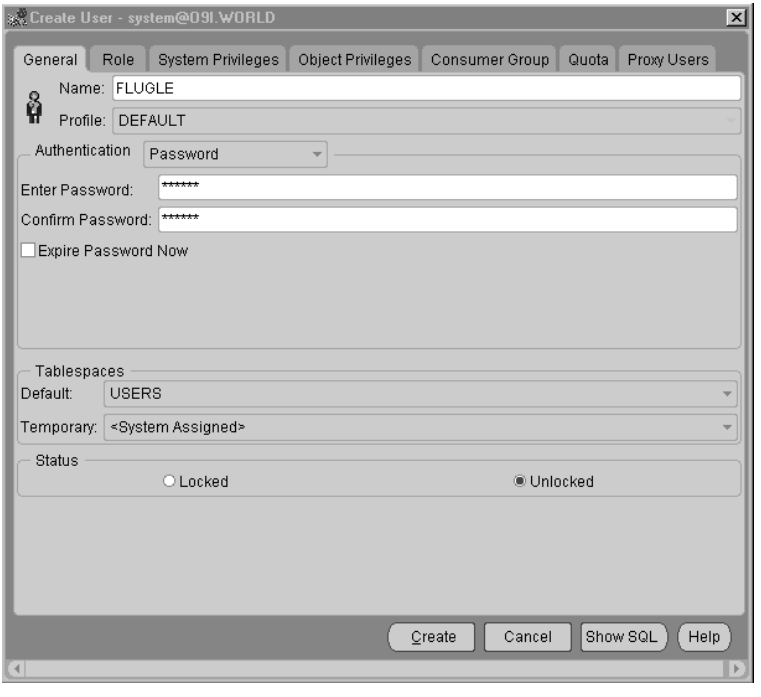

Abbildung 3.5: Eingabe des Benutzernamens und Passworts

Überzeugen Sie sich davon, dass der Benutzer unter dem Kartenreiter ROLES die Rolle 'CONNECT' erhält. Hinter der Rolle 'CONNECT' verbergen sich die Systemberechtigungen CREATE SESSION, ALTER SESSION, CREATE DATABASE LINK, CREATE TABLE, CREATE CLUSTER, CREATE SYNONYM, CREATE VIEW und CREATE SEQUENCE. Die Berechtigung CREATE SESSION ist erforderlich, um eine Datenbanksitzung aufzubauen.

Wechseln Sie zu dem Kartenreiter SYSTEM PRIVILEGES. Geben Sie dem Benutzer Flugle weitere Berechtigungen, indem Sie auf die Berechtigung klicken und diese mit einem Klick auf den nach unten gerichteten Pfeil in den mit GRANTED beschrifteten Bereich verschieben.

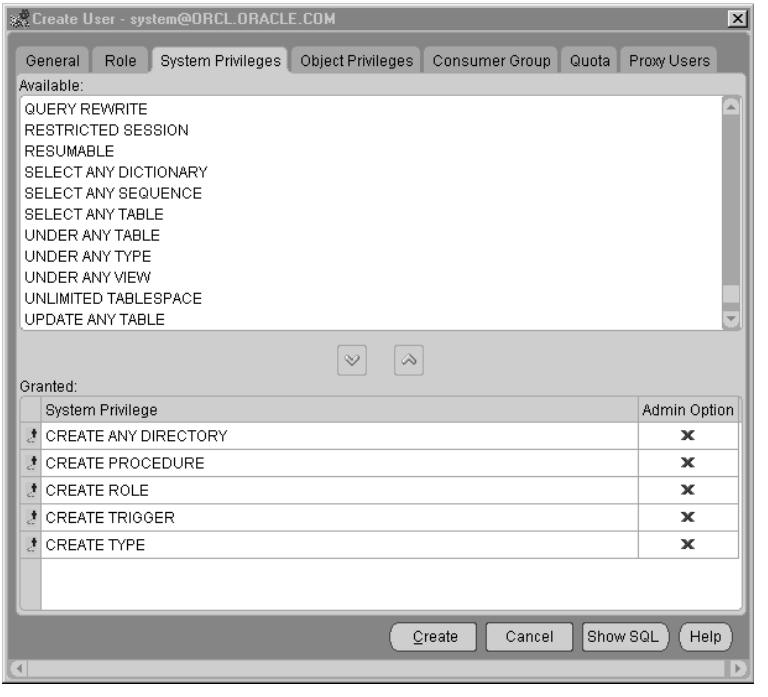

Abbildung 3.6: Vergabe von Systemberechtigungen

Wechseln Sie nun zu dem Kartenreiter QUOTA. Klicken Sie auf den zuvor unter GENERAL ausgewählten Standard-Tablespace und wählen Sie im unteren Bereich des Fensters UNLIMITED. Sie gestatten hiermit dem Benutzer Flugle unbeschränkt Speicherplatz in seinem Standard-Tablespace zu verbrauchen. Alternativ könnten Sie VALUE statt UNLIMITED wählen und dem Benutzer z.B. eine Quote von 50MB an diesem Tablespace einräumen.

Wenn Sie auf SHOW SQL klicken, zeigt Ihnen die Console, welche SQL-Anweisungen generiert werden, um Ihre Eingaben zu den einzelnen Kartenreitern umzusetzen. Sie erkennen die Elemente der Syntax von `create user`.

Die vereinfachte Syntax von `create user` ist:

```
CREATE USER user_name IDENTIFIED BY password
[DEFAULT TABLESPACE tablespace_name]
[TEMPORARY TABLESPACE temp_tablespace_name]
[QUOTA {UNLIMITED | amount M} ON tablespace_name]
```

```
[PASSWORD EXPIRE]
[ACCOUNT {UNLOCK | LOCK}];
```

Die Bedeutungen der Platzhalter sind:

- *user_name* ist der Name des neuen Benutzers.
- *password* ist das Passwort des neuen Benutzers.
- *tablespace_name* ist der Name eines vorhandenen Tablespace, der als Standard-Tablespace des neuen Benutzers verwendet wird.
- *temp_tablespace_name* ist der Name eines Tablespace für temporäre Segmente.
- *amount* ist die Anzahl Megabytes, die der Benutzer im Standard-Tablespace belegen darf.

Beim Anlegen eines Benutzers besteht die Möglichkeit, das Passwort mit `password expire` als abgelaufen zu kennzeichnen. Der Benutzer wird dann beim ersten Anmelden aufgefordert, das Passwort zu verändern. `ACCOUNT LOCK` bewirkt, dass der Zugang zu dem Account gesperrt wird. Nachdem Sie sich mit der Syntax von `create user` vertraut gemacht haben, klicken Sie abschließend auf 'Create', um den Benutzer Flugle anzulegen.

Sie können den Benutzer Flugle auch mit dem SQL-Skript sql\create_user_flugle.sql auf der CD-ROM anlegen.

3.8 Datentypen

Als Überleitung zu der Lektion des vierten Tags sollten wir nun einige Grundlagen in Bezug auf die in einer Oracle-Datenbank verfügbaren Datentypen erarbeiten. Jede Spalte in einer Oracle-Datenbank muss durch einen dieser Datentypen definiert werden. Mit Oracle Version 8.0 oder höher 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
- LOBs (**L**arge **OB**jects)

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 ANSI SQL und anderen relationalen Datenbanken – besonders SQL/DS und DB2. `Dec` und `numeric` sind gleichbedeutend mit `decimal`.
- `float` speichert Gleitkommazahlen und gewährleistet die Kompatibilität von Oracle zum ANSI-Datentyp `float`. Alternativen sind `double precision` und `real`.

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 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 einer 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 darzustellen.

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 3.2 dargestellt.

Listing 3.2: Oracle erzwingt numerische Genauigkeit

```
SQL> insert into number_digits_demo
(record_no, value)
values
(101, 9999);
1 row created.
SQL> insert into number_digits_demo
(record_no, value)
values
(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 `value` 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 3.3 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 3.3: 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

Zeichenfolgen werden grundsätzlich entsprechend der Vorgaben eines bestimmten Zeichensatzes verarbeitet. Zeichensätze sind international standardisiert und legen ein Repertoire an darstellbaren Schriftzeichen fest. Der Zeichensatz, mit dem eine Datenbanksitzung arbeitet, kann auf Windows Systemen dem Registry Value `NLS_LANG` des Registry Key HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE entnommen werden. Plattformübergreifend wird die Umgebungsvariable `NLS_LANG` ausgewertet. `NLS_LANG` hat drei Komponenten: Sprache, Land und Zeichensatz. Die Syntax von `NLS_LANG` ist: *<Sprache>_<Land>.<Zeichensatz>*. Eine mögliche Einstellung für den deutschen Sprachraum unter Verwendung eines Zeichensatzes, der das Euro-Symbol beinhaltet, ist `NLS_LANG=German_Germany.WE8ISO8859P15`. Eine umfangreiche Liste der von Oracle9*i* unterstützten Sprach-, Länder- und Zeichensatzeinstellungen finden Sie im »Oracle9*i* Globalization Support Guide«.

Eine Oracle-Datenbank kennt zwei Zeichensätze – den **Datenbank-Zeichensatz** (database character set) und den **nationalen Zeichensatz** (national character set). Der Datenbankzeichensatz ist meist ein 8bit Zeichensatz, der eine Obermenge das bekannten 7bit ASCII Zeichensatzes darstellt. Durch die Erweiterung auf 8bit ist es z.B. mit dem Zeichensatz WE8ISO8859P15 möglich, alle deutschen Umlaute und viele weitere Zeichen darzustellen, die nicht Teil des ASCII Zeichensatzes für englische Texte sind.

Unicode ist ein standardisierter Zeichensatz, der alle gebräuchlichen Sprachen beinhaltet.

Der Unicode Standard ist im Internet unter dem URL *http://www.unicode.org* dokumentiert. Der Datenbankzeichensatz kann auch die Unicode-Implementierung UTF8 sein. Dies ist zwingend, wenn Sie die Datenbank für den LDAP Verzeichnisdienst Oracle Internet Direc-

tory nutzen wollen und wünschenswert, wenn die Datenbank-Zeichensätze aus verschiedensten Sprachräumen unterstützen soll. Der Datenbank-Zeichensatz WE8ISO8859P15 bildet z.B. nur westeuropäische Zeichen ab. Kyrillische Zeichen könnten nicht in der Datenbank gespeichert werden. Wenn Sie jedoch UTF8 als Datenbank-Zeichensatz wählen, können westeuropäische und kyrillische Zeichen zusammen mit vielen anderen Zeichensätzen wie Arabisch oder Hebräisch gemeinsam in der Datenbank abgelegt werden.

Der Datenbank-Zeichensatz kann sich vom Zeichensatz der Datenbanksitzung (`NLS_LANG`) unterscheiden. Oracle9*i* konvertiert dann automatisch zwischen den Zeichensätzen. Zum Speichern von Zeichenfolgen können Sie aus mehreren Datentypen auswählen:

- `char`
- `nchar`
- `varchar2`
- `nvarchar2`
- `clob`
- `nclob`

Spalten vom Typ `char`, `varchar2` und `clob` werden im Datenbank-Zeichensatz kodiert. Spalten vom Typ `nchar`, `nvarchar2` und `nclob` werden im nationalen Zeichensatz kodiert. Wenn Sie mit Daten, die im nationalen Zeichensatz kodiert sind arbeiten, müssen Sie die Daten kennzeichnen. In SQL*Plus wird eine Zeichenkette im nationalen Zeichensatz durch Voranstellen des Buchstabens `N` vor die Zeichenkette (z.B. `N'Testdaten'`) gekennzeichnet.

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

Der Datentyp `char` speichert Zeichenketten fester Länge von bis zu 2000 Zeichen. Wenn Sie keine Länge vorgeben, wird in einer `char`-Spalte ein Zeichen gespeichert. Wenn die gespeicherte Zeichenkette kürzer als die definierte Länge ist, wird mit Leerzeichen aufgefüllt.

Der Datentyp `varchar2` speichert bis zu 4000 Zeichen in einer einzigen Spalte. Wenn Sie eine größere Zeichenmenge speichern müssen, sollten Sie die Verwendung eines LOB-Datentyps in Erwägung ziehen.

Sie können 4 GB in einer `clob`-Spalte speichern. Der Datentyp `clob` fällt in die Kategorie der Large Objects. Programmierschnittstellen für `clobs` liegen in den Sprachen PL/SQL, Java, C (OCI) und C++ (OCCI) vor. Es wäre möglich, lange Zeichenketten bis zu 2 GB in einer Spalte vom Datentyp `long` zu speichern, jedoch ist der Datentyp `long` veraltet und wesentlich schlechter handhabbar. Oracle empfiehlt den Datentyp `long` nicht mehr einzusetzen. Spalten des Datentyps `long` können mit der `alter table modify` Anweisung von Oracle9*i* auf `clob` Spalten umgestellt werden.

Folgen Sie diesen Richtlinien, wenn Sie den geeigneten Datentyp für eine Spalte auswählen, in der Zeichenketten gespeichert werden sollen.

- Sie können `char` verwenden, um Spalten zu definieren, in denen ein einziges Zeichen gespeichert wird.
- Verwenden Sie `varchar2` zum Speichern von Zeichenfolgen variabler Länge, die bis zu 4000 Zeichen enthalten.
- Verwenden Sie `clob` oder `nclob`, um mehr als 4000 Zeichen in einer Spalte zu speichern.

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 varchar2 verwenden

Da hiermit Zeichenfolgen variabler Länge gespeichert werden, sollte der Datentyp `varchar2` zum Speichern von Zeichenfolgen bevorzugt eingesetzt werden. Dieser Datentyp kann bis zu 4000 Zeichen speichern. Das Oracle-RDBMS weist nur den für die Speicherung der einzelnen Spaltenwerte benötigten Speicherplatz zu. Am Tag 8, »SQL-Funktionen«, werden Sie die Vielzahl der Möglichkeiten kennen lernen, wie Sie SQL-Funktionen und Operatoren zur Verarbeitung von Zeichenketten und anderer Daten verwenden können.

Datums- und Zeitinformationen

Oracle9*i* bietet mehrere Möglichkeiten Datum, Uhrzeit und Zeitintervalle zu speichern. Die verfügbaren Datentypen sind `date`, `interval` und verschiedene Varianten von `timestamp`. Oracle verfügt über integrierte Funktionen, die speziell zum Bearbeiten von Datums- und Zeitinformationen dienen. Am Tag 8 werden Sie einige Beispiele für diese Funktionen kennen lernen.

Der Datentyp date

Der Datentyp `date` ermöglicht es Ihnen, Daten im Zeitraum vom 1. Januar 4712 v. Chr. bis zum 31. Dezember 9999 n. Chr. zu speichern. Oracle weist einer `date`-Spalte einen Speicherplatz mit einer Länge von 7 Bytes zu. Oracle verwendet zur Datumseingabe und

Datumsanzeige standardmäßig das Format DD-Mon-YY. In diesem Format steht DD für den Tag des Monats, Mon für den Monatsnamen in englischer Sprache und YY für das zweistellige Jahr.

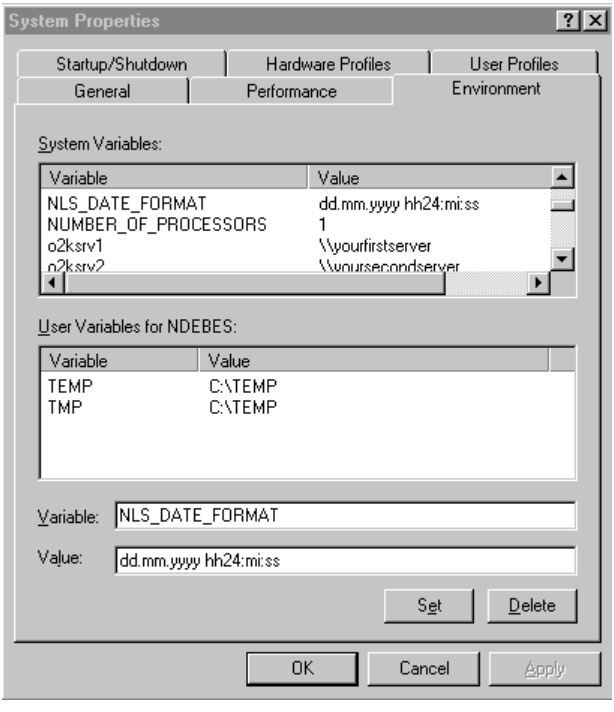

Abbildung 3.7: Festlegung des Datums- und Zeitformats als Umgebungsvariable über das Windows CONTROL PANEL

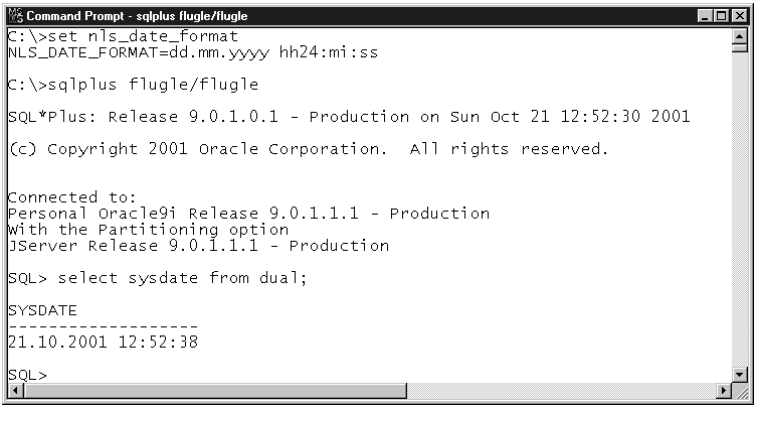

Abbildung 3.8: Einstellung des Formats der Datums- und Uhrzeitausgabe mit der Umgebungsvariablen NLS_DATE_FORMAT

Wenn dieses Format für Ihre Zwecke nicht geeignet ist, können Sie die Umgebungsvariable NLS_DATE_FORMAT auf ein anderes Format setzen (siehe Abbildung 3.7). Auf Windows Systemen verwenden Sie das START Menü, um über SETTINGS und CONTROL PANEL das Piktogramm SYSTEM auszuwählen. Dort wechseln Sie zu dem Kartenreiter ENVIRONMENT und geben unter VARIABLE NLS_DATE_FORMAT ein. Unter VALUE geben Sie z.B. dd.mm.yyyy hh24:mi:ss ein, um die Uhrzeit sekundengenau auszugeben. Klicken Sie auf SET und APPLY, um die Änderung wirksam zu machen. Wenn Sie jetzt an einem MS DOS Command Prompt oder über das START Menü SQL*PLUS starten, ist die Änderung wirksam.

Der Datentyp timestamp

Timestamp ist eine Erweiterung von date, in dem Sinne, dass der Datentyp timestamp zusätzlich Sekundenbruchteile und Zeitzoneninformation speichern kann.

Die Syntax von timestamp ist:

```
TIMESTAMP [(fractional_seconds_precision)] [WITH [LOCAL] TIME ZONE]
```

Der Platzhalter *fractional_seconds_precision* steht für die Anzahl der Ziffern, die Sekundenbruchteile darstellen.

Eine Oracle9*i* Instanz läuft im Kontext einer bestimmten Zeitzone. Die Zeitzoneneinstellung des Betriebssystems wird von der Oracle-Instanz für diesen Zweck genutzt und als Datenbankzeitzone bezeichnet. Wenn mit local timezone gearbeitet wird, werden alle Zeitinformationen normalisiert auf die Datenbankzeitzone gespeichert. Falls erforderlich, konvertiert Oracle zwischen der Zeitzone, in der das Anwendungsprogramm läuft, und der Datenbankzeitzone.

Bei Verwendung von timestamp with time zone wird in der Spalte zusätzlich die Abweichung von UTC (Universal Time Coordinated, früher Greenwich Mean Time, GMT) gespeichert. Datumswerte werden dann inklusive der Abweichung von UTC angegeben. Zum Beispiel stellt die Zeichenkette '01.10.2001 09:26:56.66 +02:00' den ersten Oktober 2001 und eine Abweichung von +2 Stunden von UTC dar.

Der Datentyp interval

Aufgabe des Datentyps interval ist die Speicherung von Zeitintervallen und die Durchführung von Berechnungen mit Zeitpunkten und Zeitintervallen. Zwei Varianten stehen zur Verfügung – interval year to month und interval day to second.

Die Syntax des Datentyps `interval` lautet:

```
INTERVAL YEAR [(year_precision)] TO MONTH
INTERVAL DAY [(day_precision)] TO
SECOND(fractional_seconds_precision)]
```

Die Bedeutungen der Platzhalter sind wie folgt:

- *year_precision* steht für die Anzahl der Ziffern, die für die Dauer des Intervalls in Jahren verwendet werden können.
- *day_precision* steht für die Anzahl der Ziffern, die für die Dauer des Intervalls in Tagen verwendet werden können.
- *fractional_seconds_precision* steht für die Anzahl der Ziffern, die für die Dauer des Intervalls in Sekundenbruchteilen verwendet werden können.

Oracle stellt mehrere SQL-Funktionen für den Umgang mit Intervallen zur Verfügung. Am Tag 8, »SQL-Funktionen«, werden Sie sich näher mit diesen Funktionen beschäftigen.

Große Objekte

Wie Sie wahrscheinlich wissen, erlauben die meisten Datenbanken das Speichern großer Objekte (LOB, Large Object). LOBs umfassen Dokumente, Grafiken, Klänge, Videos – eigentlich alle Arten von Binärdateien, die Sie sich vorstellen können.

LOBs werden in vier Kategorien eingeteilt:

- BLOBs, die unstrukturierte binäre Daten in der Datenbank speichern
- CLOBs für Dokumente im Datenbankzeichensatz
- NCLOBs für Dokumente im nationalen Zeichensatz
- temporäre LOBs, d.h. nicht persistente BLOBs, CLOBS oder NCLOBS
- 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 veralteten `long raw`-Spalte aufnehmen. Zweitens, eine Tabelle kann mehr als eine LOB-Spalte enthalten, jedoch nur eine `long raw`-Spalte. Außerdem können die in einer LOB-Spalte gespeicherten Daten in einem eigenen Segment gespeichert werden, was bei Zugriffscharakteristiken, die selten auf den LOB zugreifen, zu einer besseren Gesamt-Performance führt. LOBs, die größer als 4000 Bytes sind, werden grundsätzlich in einem eigenen LOB-Segment gespeichert.

LOB Locator

Die Handhabung eines LOBs erfolgt indirekt über einen so genannten **LOB-Locator** und das PL/SQL Paket DBMS_LOB. LOB-Spalten enthalten nicht den LOB selbst, sondern einen LOB-Locator. Der LOB-Locator dient als Parameter für die Prozeduren und Funktionen des Pakets DBMS_LOB. Eine Anwendung, die mit einem LOB arbeiten möchte, muss zunächst den LOB-Locator aus der Tabelle lesen. Anschließend kann durch Aufruf von DBMS_LOB und Übergabe des LOB-Locator eine der Funktionalitäten aus Tabelle 3.2 verwendet werden. Weitergehende Informationen zum Umgang mit LOBs erhalten Sie in den Handbüchern »Oracle9*i* Application Developer's Guide – Large Objects (LOBs)« und » Oracle9*i* Supplied PL/SQL Packages and Types Reference«.

Funktion	Prozedur/Funktion des Pakets DBMS_LOB
Schreiben	DBMS_LOB.WRITE
Daten am Ende des LOB einfügen	DBMS_LOB.WRITEAPPEND
Lesen	DBMS_LOB.READ
Ermitteln der Länge des LOB	DBMS_LOB.GETLENGTH
Daten des LOB ganz oder teilweise löschen	DBMS_LOB.ERASE
Kopieren	DBMS_LOB.COPY
Vergleichen von LOBs	DBMS_LOB.COMPARE
Suchen	DBMS_LOB.INSTR

Tabelle 3.2: Häufig genutzte Funktionalitäten des PL/SQL Pakets DBMS_LOB

Verwenden von BLOBs

Sie können sich den BLOB-Datentyp als eine neue, verbesserte Version des long raw-Datentyps vorstellen. Anders als eine long raw-Spalte kann eine INSERT-Anweisung mit einer Unterabfrage eine BLOB-Spalte enthalten. 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 SQL-Funktionen verwenden. Oracle empfiehlt long raw-Spalten zu Gunsten von BLOB-Spalten nicht mehr zu verwenden. Listing 3.4 ist ein Beispiel für eine Tabelle, die eine BLOB-Spalte enthält. Diese wird benutzt, um den eingescannten Lebenslaufs eines Dozenten im Pixelformat TIFF zu speichern.

Listing 3.4: Erstellen einer Tabelle, die eine Spalte vom Typ BLOB enthält

```
SQL> create table Instructor_Application (
  Application_Number number,
  Last_Name          varchar2(30),
  First_Name         varchar2(30),
  MI                 varchar2(1),
  Resume_TIFF             blob,
  Status             varchar2(30));
Table created.
```

Initialisierung einer LOB-Spalte

Es gibt zwei integrierte Funktionen, die benutzt werden, um einen LOB-Locator zu initialisieren: EMPTY_BLOB für BLOBs und EMPTY_CLOB für CLOBs und NCLOBs. Beide Funktionen besitzen keine Argumente. Listing 3.5 veranschaulicht die Benutzung von EMPTY_BLOB in einer INSERT-Anweisung.

Listing 3.5: 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_TIFF, Status)
  3  values
  4  (1001, 'Deevers', 'James', NULL, EMPTY_BLOB(), 'REJECTED');

1 row created.
```

Der Datentyp BFILE

Eine BFILE-Spalte wird benutzt, um eine binäre Datei zu referenzieren, die in einem Dateisystem, das für eine Oracle-Instanz erreichbar ist, gespeichert ist. Die Binärdatei befindet sich außerhalb der Datenbank. Als Folge dessen stehen die Mechanismen von Oracle für atomare Transaktionen, Schutz vor gleichzeitigen Änderungen, konsistentes Lesen, Ändern der Binärdaten (DBMS_LOB) sowie Datensicherung und Wiederherstellung (Backup und Recovery) nicht zur Verfügung.

Bevor Sie sich den BFILEs zuwenden können, benötigen Sie noch Vorkenntnisse über Aliasnamen für Verzeichnisse.

Die create directory-Anweisung

Die CREATE DIRECTORY-Anweisung ist für die Arbeit mit BFILE-Spalten gedacht. Die Ausführung dieser Anweisung erfordert die Systemberechtigung CREATE ANY DIRECTORY. 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. Der Eigentümer des Alias hat automatisch Leserechte.

Die Syntax der CREATE DIRECTORY-Anweisung lautet:

```
CREATE DIRECTORY oracle_directory_name AS 'absolute_file_system_path';
```

Die Bedeutungen der Platzhalter sind:

- *oracle_directory_name* ist der Name des Directory Objekts in der Datenbank. *Oracle_directory_name* ist der Name, den Oracle-Benutzer zur Referenzierung des Verzeichnisses benutzen müssen. Wie bei allen Namen von Datenbankobjekten, die nicht in doppelten Anführungszeichen angegeben werden, wird auch der *oracle_directory_name* in Großbuchstaben im Datenbankkatalog gespeichert. Achten Sie also darauf, den *oracle_directory_name* in Großbuchstaben zu schreiben, wenn Sie ihn als Argument an Funktionen wie z.B. BFILENAME verwenden.
- *absolute_file_system_path* ist der absolute Pfad eines Verzeichnisses, das mit den Betriebssystem-Zugriffsrechten der Oracle-Instanz erreichbar ist. Auf UNIX Systemen beginnen absolute Pfade mit dem Schrägstrich / (z.B. /home/oracle), auf Windows Systemen mit dem Laufwerksbuchstaben (z.B. C:\home\oracle). Wenn das Betriebssystem Groß-/Kleinschreibung unterscheidet, muss dies beachtet werden.

Listing 3.6 veranschaulicht, wie ein Verzeichnis erzeugt wird. Die Berechtigung zum Lesen von Dateien aus diesem Verzeichnis kann einer Datenbankrolle gewährt werden.

Listing 3.6: Erstellen eines Verzeichnisses

```
SQL> create directory PHOTO_DIR as 'C:\Flugle\Photos';

Directory created.
```

Das Gegenstück zur CREATE DIRECTORY-Anweisung ist die DROP DIRECTORY-Anweisung, mit der ein Verzeichnis-Alias gelöscht wird.

Als Beispiel demonstriert Listing 3.7, wie eine Tabelle erzeugt wird, die eine BFILE-Spalte enthält.

Listing 3.7: Definieren und Verwenden einer Spalte vom Typ BFILE

```
SQL> create table Instructor_Photo
    (Instructor_ID  number(5),
    Recent_Photo   bfile);

Table created.

SQL> insert into Instructor_Photo
    (Instructor_ID, Recent_Photo)
    values
    (251, BFILENAME ('PHOTO_DIR', '251.jpg'));

1 row created.
```

Die Instructor_Photo-Tabelle wird mit einer `BFILE`-Spalte namens Recent_Photo erzeugt. Der Zweck dieser Spalte ist es, das Verzeichnis und den Namen der `JPEG`-Datei zu bestimmen, die ein Foto des Dozenten enthält. Eine Zeile wird in die Instructor_Photo-Tabelle eingefügt. Wie Sie sehen können, wird die `BFILENAME`-Funktion zur Bereitstellung eines LOB-Locator für die `BFILE`-Spalte benutzt. Der Verzeichnisalias **PHOTO_DIR** und der Dateiname werden in Hochkommata eingeschlossen. Beachten Sie, dass der Verzeichnisalias beim Anlegen mit `CREATE DIRECTORY` nicht in doppelte Anführungszeichen eingeschlossen war. Daher wurde das Datenbankobjekt **PHOTO_DIR** in Großbuchstaben angelegt und muss bei der Verwendung ebenfalls groß geschrieben werden. Der Inhalt der mittels der `BFILE`-Spalte referenzierten binären Datei kann mit einer SQL-Anweisung nicht geändert werden. Das Verzeichnis und der Dateiname, die in der BFILE-Spalte gespeichert sind, jedoch schon.

Sie finden ein vollständiges Beispiel für die Verwendung von BFILEs in der Datei sql\bfile.sql auf der Begleit-CD zum Buch.

Lange Zeichenfolgen

Wie bereits zuvor erwähnt, sollten Sie den Oracle-Datentyp `clob` verwenden, um mehr als 4000 Zeichen in einer einzelnen Spalte zu speichern. Eine `clob`-Spalte fasst eine Zeichenmenge von bis zu 4 GB. Wie beim Datentyp `varchar2` weist das Oracle-RDBMS bei der Verwendung des Datentyps `clob` lediglich den für die Speicherung der einzelnen Spaltenwerte erforderlichen Speicherplatz zu. Sie werden allerdings bei der Verwendung von `clob`-Spalten in SQL auf eine Reihe von Einschränkungen stoßen. Sie können keine SQL-

Funktionen oder Operatoren verwenden, um den Inhalt einer clob-Spalte zu durchsuchen oder zu ändern. Sie müssen eine der Programmierschnittstellen für LOBs, z.B. das PL/SQL Paket dbms_lob verwenden.

Verwenden von CLOBs

In Listing 3.8 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 gewonnen wurde.

Listing 3.8: 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
     set OCR_Resume = 'Resume of James R. Deevers - Born in San Jose in 1972';
1 row updated.
SQL> select OCR_Resume from Instructor_Application;
OCR_RESUME
--------------------------------------------------
Resume of James R. Deevers - Born in San Jose in 1972
```

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.

Der Datentyp raw

Oracle stellt den Datentyp raw bereit, der bis zu 2000 Byte an Binärdaten fassen kann. Aufgrund der Speicherbegrenzung ist eine raw-Spalte weniger nützlich als eine BLOB-Spalte. Der Einsatz von raw-Spalten kann sinnvoll sein, wenn Sie kleine Binärobjekte speichern möchten. Da ein LOB 88 Byte Verwaltungsinformation benötigt, sparen Sie Speicherplatz, wenn Sie Binärobjekte mit weniger als 2000 Bytes in raw-Spalten ablegen.

3.9 Zusammenfassung

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

1. Identifizieren Sie jede Tabelle im zu modellierenden System.
2. Benennen Sie einen Primärschlüssel für jede Tabelle.
3. Bestimmen Sie die zu erstellenden Fremdschlüssel.

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

- Jede Spalte in einer Tabelle muss einen Datentyp erhalten.
- Verwenden Sie `number` zum Speichern numerischer Daten.
- Verwenden Sie `char` zum Speichern einzelner Zeichen.
- Verwenden Sie `varchar2` zum Speichern von Zeichenketten, die maximal 4000 Bytes enthalten. In Oracle integrierte Funktionen und Operatoren sind mit `varchar2` Spalten verwendbar.
- Verwenden Sie `clob`, wenn Sie lange Zeichenfolgen mit mehr als 4000 Bytes, wie z.B. Texte oder HTML Seiten speichern müssen.
- Verwenden Sie `date` zum Speichern aller Datums- und Zeitinformationen.
- Verwenden Sie `timestamp` zum Speichern von Datum, Uhrzeit und Sekundenbruchteilen. Der Datentyp `timestamp` kann auch Informationen über Zeitzonen beinhalten.
- Verwenden Sie `BLOB`-Spalten zum Speichern von bis zu 4 GB Binärdaten in der Datenbank.
- Verwenden Sie BFILE-Spalten, um Binärdaten außerhalb der Datenbank zu referenzieren und über Programmierschnittstellen von Oracle zu Lesen.

3.10 Wie geht es weiter?

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

3.11 Fragen und Antworten

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

A 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.

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

A Eine Tabelle kann bis zu 1000 Spalten enthalten.

F *Ist es möglich eine Binärdatei, die von einer BFILE-Spalte referenziert wird, mit Mechanismen von Oracle zu ändern?*

A Nein, das Ändern von Binärdateien, die über BFILE-Spalten referenziert werden, ist nicht möglich.

3.12 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. Richtig oder falsch? Wenn ein Attribut, das nicht Teil des Primärschlüssels ist, einen Fremdschlüssel darstellt, ist es unbedingt erforderlich; es lässt keinen **NULL**-Wert 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. Es gibt 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?