

# 1 Einführung

Führende Datenbanksystemhersteller wie Oracle und IBM setzen auf objektrelationale Datenbanktechnologie. Objektrelationale Datenbanken sind eine evolutionäre Erweiterung relationaler Datenbanken um objektorientierte Konzepte. Sie bieten eine anpassungsfähige Plattform, um moderne Anwendungen, wie sie etwa in Multimedia- und Web-Informationssystemen vorkommen, besser zu unterstützen.

Mit SQL:1999 gibt es einen ersten Standard für *objektrelationale* Datenbanken, der neben neuen Datentypen und Typkonstruktoren auch Objekttypen sowie Typ- und Tabellenhierarchien unterstützt. Für die Interaktion zwischen objektrelationalen Datenbanken und Java-Anwendungen stehen mit JDBC und SQLJ zwei Standardschnittstellen bereit. Der zukünftige Standard SQL:2003 sieht die Integration von XML in SQL vor.

Dieses Kapitel gibt einen kompakten Überblick über die Geschichte und Inhalte von SQL:1999 und dessen bereits designierten Nachfolger SQL:2003. Die zugrunde liegenden Konzepte werden kurz zur Übersicht eingeführt. Eine detaillierte Abhandlung der einzelnen Sprachkonstrukte folgt in den nächsten Kapiteln.

## 1.1 Historie von SQL

Die Grundlagen von SQL beruhen auf dem relationalen Datenmodell, das um 1970 zusammen mit der Relationenalgebra von E. F. Codd eingeführt wurde. Auf dieser Basis entwickelte IBM 1974 die Datenbanksprache *SEQUEL* (Structured English QUery Language), die heute als Urvater von SQL gilt. 1976 entstand mit *System R* der erste Prototyp eines relationalen Datenbanksystems, das *SEQUEL* bzw. dessen Nachfolger *SEQUEL2* implementierte [CAE<sup>+</sup>76].

Anfang der achtziger Jahre waren die ersten relationalen Datenbanksysteme kommerziell erhältlich. Die prominentesten unter ihnen sind aus heutiger Sicht die Vorgänger von Oracle, DB2 und Informix. Diese Systeme setzten auf einer Untermenge von SEQUEL2 auf und entwickelten ihre eigenen Dialekte unter dem Akronym *SQL* (Structured Query Language).

Das American National Standards Institute (ANSI) nahm 1982 die Normierung von SQL in Angriff. Die erste genormte Version von SQL wurde 1986 veröffentlicht. Sie wird daher als SQL-86 bezeichnet. Eine revidierte bzw. ergänzte Fassung dieser Norm wurde 1989 von der International Standards Organization (ISO) herausgegeben. Diese Version heißt SQL-89. Die erste gemeinsame ISO/ANSI-Norm aus dem Jahre 1992, die unter dem Namen SQL-92 bzw. SQL2 bekannt wurde, beruhte überwiegend auf dem relationalen Datenmodell.

Durch die massiv zunehmende Popularität objektorientierter Programmiersprachen angetrieben, entstanden Mitte der achtziger Jahre die ersten Prototypen objektorientierter Datenbanksysteme. Diese Systeme hatten das Ziel, die Vorzüge der Objektorientierung auch auf Datenbanken zu übertragen. Die kommerziellen Produkte, die ab 1990 verfügbar waren, konnten sich jedoch aus mehreren Gründen nicht gegenüber den etablierten relationalen SQL-Datenbanksystemen wie Oracle oder IBM DB2 durchsetzen. Das Problem lag nicht an der Idee objektorientierter Datenbanken, sondern vielmehr an der fehlenden Reife der Produkte, die grundlegende Datenbankfunktionalität wie Anfrageoptimierung, Transaktionsverarbeitung und Zugriffskontrolle nicht bzw. nur sehr rudimentär unterstützten. Mit der »Objektorientierung« der SQL-Datenbanksystemhersteller, die nach und nach objektorientierte Konzepte in ihre ehemals relationalen Systeme integrierten, wurde die Ära der *objektrelationalen* Datenbanksysteme eingeleitet. UNISQL war 1996 das erste kommerziell erhältliche objektrelationale Datenbanksystem. Die objektrelationalen Versionen von Informix, Oracle und DB2 folgten unmittelbar danach im Jahre 1997.

Zu diesem Zeitpunkt war das Normierungsprojekt SQL3 bereits fünf Jahre alt und umfasste eine Vielzahl von objektorientierten Konzepten. Im September 1999 wurde eine abgespeckte Fassung von SQL3 als die aktuelle Norm SQL:1999 (oft auch SQL-99 genannt) veröffentlicht. Einige der unberücksichtigten Konzepte sind für die zukünftigen SQL-Normen vorgesehen. Die nächste SQL-Norm wird für Ende 2003 erwartet. Sie wird SQL:2003 heißen.

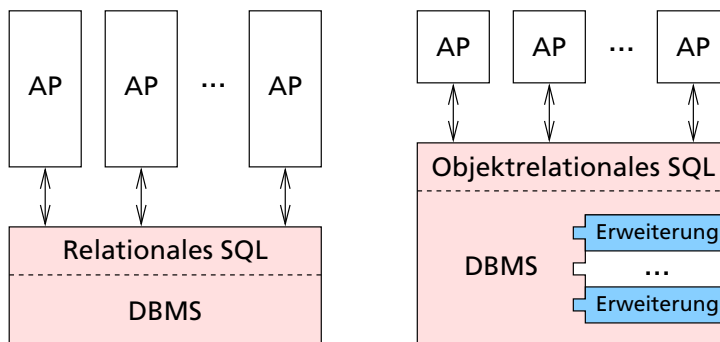
## 1.2 Aufgaben einer Datenbank

Eine *relationale Datenbank* ist eine strukturierte, persistente Sammlung von Daten, welche Fakten und Regeln über eine modellierte Miniwelt in Form von Relationen verwaltet. Ein *Datenbankmanagementsystem (DBMS)* ist eine Ansammlung von anwendungsunabhängigen Programmen, die das Erzeugen, Ändern und Löschen einer Datenbank ermöglicht. Die Kombination eines DBMS mit einer oder mehreren Datenbanken wird als *Datenbanksystem* bezeichnet.

Ein DBMS hat die folgende Funktionalität zu erbringen [HS00]:

- ❑ *Integration* und einheitliche Verwaltung *aller* von den Anwendungen benötigten Daten.
- ❑ *Speicherung (Persistenz)* der Daten über die Lebensdauer einer Anwendungsausführung hinaus.
- ❑ *Operationen* zum anwendungsunabhängigen Erzeugen, Ändern, Speichern, Suchen und Entfernen von Daten.
- ❑ *Schemakatalog* verwaltet und bietet Zugriff auf die Datenbeschreibungen (Metadaten).
- ❑ *Sichten* stellen benutzerspezifische Ausschnitte der Datenbank bereit.
- ❑ *Integritätssicherung* überwacht die semantische Korrektheit der Datenbank, das heißt stellt sicher, dass alle Fakten den modellierten Regeln entsprechen.
- ❑ *Zugriffskontrolle* verhindert unberechtigte Zugriffe auf die Datenbank.
- ❑ *Transaktionen* sind elementare Ausführungseinheiten, die aus einer Folge von Operationen bestehen, deren Effekt bei Erfolg persistent in der Datenbank gespeichert wird.
- ❑ *Synchronisation* der Transaktionsausführungen im Mehrbenutzerbetrieb, um einen inkorrekten Informationsfluss beim nebenläufigen Arbeiten auf gemeinsam benötigten Datenbeständen auszuschließen.
- ❑ *Datensicherung* verhindert den Verlust von Daten auch nach System- und Mediafehlern.

Ein DBMS bietet eine Schnittstelle, welche die Anwendungsprogrammierer davon befreit, die oben aufgeführten Aufgaben in jedem Anwendungsprogramm neu implementieren zu müssen. Je »höher« die Datenbankschnittstelle ist, desto kompakter werden die *Anwendungsprogramme (AP)*. Abbildung 1.1 veranschaulicht dies grafisch.



**Abbildung 1.1:** Architektur von Datenbanksystemen

Die Datenbankschnittstelle hängt unmittelbar vom zugrunde liegenden Datenmodell ab. Das *Datenmodell* definiert die möglichen Strukturen und Funktionen, die zur Beschreibung und Manipulation einer Datenbank eingesetzt werden können.

Eine relationale Datenbankschnittstelle bietet generische Operationen zum Erzeugen, Manipulieren und Entfernen von Relationen an. Objektrelationale Datenbanken sehen eine höhere Schnittstelle vor, die neben Operationen für Relationen auch benutzerdefinierte Prozeduren und Funktionen für objektorientierte Strukturen unterstützt.

Aus dieser Abbildung geht ein weiterer markanter Unterschied zwischen relationalen und objektrelationalen Datenbanken hervor. Letztere sind *erweiterbar* in dem Sinne, dass benutzerdefinierte Funktionalität *transparent* für alle Anwendungsprogramme vom DBMS unterstützt wird.

### 1.3 Basiskonzepte von SQL

SQL ist eine Datenbanksprache, die ursprünglich eine relationale Schnittstelle definierte. Seit SQL:1999 ist die Schnittstelle objektrelational. Das zentrale Konzept von SQL sind Tabellen. Eine *Tabelle* speichert die Fakten der modellierten Miniwelt. Logisch zusammengehörende Tabellen werden unter einem gemeinsamen Schema verwaltet. Ein *Schema* definiert einen Namensraum für alle Objekte des Schemas. Schemata sind ihrerseits Bestandteil eines *Katalogs*. Eine SQL-Datenbank besteht aus einem oder mehreren Katalogen. Mit anderen Worten ist eine SQL-Datenbank »hierarchisch« organisiert:

Katalog.Schema.Tabelle

Das Erzeugen und Löschen von Katalogen ist implementierungsabhängig. Das heißt, die SQL-Norm sieht hierfür keine Sprachkonstrukte vor. Das Erzeugen, Ändern und Löschen von Schemata und Tabellen erfolgt über die *Datendefinitionssprache (DDL)* von SQL. Das Erzeugen, Ändern und Löschen von Tabelleninhalten geschieht mit der *Datenmanipulationssprache (DML)* von SQL. Für die Abfrage der Tabelleninhalte wird die *Anfragesprache* von SQL genutzt.

#### Relationale Grundlagen von SQL

Im »relationalen« Datenmodell von SQL-92 [MS93, DD93] umfasst ein Schema im Wesentlichen folgende Objekte:

- ❑ *Tabellen und Sichten*
- ❑ *Basisdatentypen und Domänen*
- ❑ *Integritätsbedingungen und Assertions*

Eine *Tabelle* ist die einzige Datenstruktur zur Speicherung von Daten in einer SQL-Datenbank. Wie in Abbildung 1.2 dargestellt, besteht eine Tabelle aus *Spalten* und *Zeilen*. Im Gegensatz zu einer Relation im Relationenmodell entspricht eine SQL-Tabelle einer Multimenge von Tupeln. Die echte Mengeneigenschaft, das heißt der Ausschluss von Tupelduplikaten, kann durch die Definition von Eindeutigkeitsbedingungen erzwungen werden.

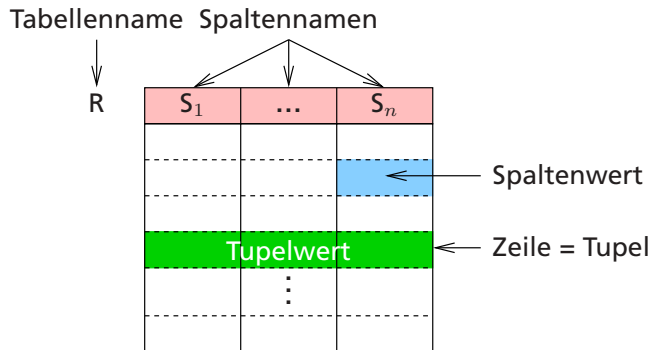


Abbildung 1.2: SQL-Tabelle

Die Felder einer Tabelle enthalten Werte aus Wertebereichen, die durch *Basisdatentypen* oder *Domänen* bereitgestellt werden. Das Typsystem von SQL bietet Basisdatentypen für numerische, alphanumerische, binäre und Datum-Zeit-Werte. Das Konzept der *Domänen* ermöglicht die Definition von benutzerdefinierten Wertebereichen durch Einschränkung existierender Datentypen. Eine *Sicht* ist eine virtuelle Tabelle, die genau genommen eine gespeicherte Anfrage repräsentiert.

*Benutzerdefinierte Prozeduren* und *Funktionen* sowie *Trigger* sind datenmodellunabhängige Erweiterungen von SQL, die im weitesten Sinne »Programme« in der Datenbank speicher- und ausführbar machen und damit die Wiederverwendung von Programmcode unterstützen.

### Objektorientierte Grundlagen von SQL

Das wesentliche Ziel der objektrelationalen Erweiterungen von SQL ist es, mehr Anwendungssemantik in die Datenbank einzubringen und damit Datenbankfunktionalität für anwendungsspezifische Datentypen und Funktionen verfügbar zu machen. Das Datenbankmanagementsystem soll das Wissen über die Semantik der Datentypen und Funktionen beispielsweise zur Optimierung von Anfragen und Synchronisation von Transaktionen ausnutzen.

Mit der objektrelationalen Erweiterung von SQL:1999 kommen unter anderem folgende Schemaobjekte hinzu:

- ❑ *Konstruierte und benutzerdefinierte Datentypen*
- ❑ *Typisierte Tabellen und Sichten*
- ❑ *Typ-, Tabellen- und Sichtenhierarchien*

### Konstruierte und benutzerdefinierte Datentypen

*Typkonstruktoren* wie *Tuple*, *Set*, *Multiset* und *List* ermöglichen die Definition von *konstruierten* Datentypen. Durch die geschachtelte Anwendung von Typkonstruktoren entstehen beliebig *komplexe* Datentypen. Damit wird eine »natürlichere« Abbildung der Realweltobjekte auf Datenbankobjekte unterstützt [SST97].

*Benutzerdefinierte* Datentypen repräsentieren *benannte* Typkonstruktoren. Sie ermöglichen die *Wiederverwendbarkeit* von konstruierten Datentypen und damit die *Erweiterbarkeit* des Typsystems. Die *strenge Typisierung* erzwingt eine exaktere Abbildung der Realwelt auf die Datenbankwelt.

Das Konzept der *strukturierten Typen* ist eine der zentralen objektrelationalen Erweiterungen von SQL. Ein strukturierter Typ repräsentiert einen *Objekttyp*, *Objekt-!TypTyp!Objekt-* welcher die *Eigenschaften (Attribute)* und das *Verhalten (Methoden)* von gleichartigen Datenbankobjekten beschreibt. Den strukturierten Typen liegen folgende Prinzipien der Objektorientierung zugrunde:

- ❑ *Strukturierung bzw. Modularisierung*: Komplexe Datenstrukturen bilden mit den assoziierten Funktionen semantische Einheiten, welche zu einer besseren Strukturierung und Wartbarkeit der Datenbank führen. Fehler können lokalisiert und Funktionsänderungen isoliert vorgenommen werden.
- ❑ *Kapselungsprinzip*: Der Zugriff auf ein Objekt erfolgt über die Methoden ihrer Schnittstelle. Die Implementierung der Methoden bleibt nach außen hin verborgen. Die klare Trennung zwischen Schnittstelle und Implementierung von Objektmethoden schafft die Basis für eine transparente Evolution der Objektfunktionalität.
- ❑ *Prinzip der zustandsunabhängigen Objektidentifikation*: Die Entkopplung der Identifikation der Objekte von den zugrunde liegenden Objektwerten ermöglicht zum einen eine eindeutige, *unveränderliche* Objektreferenz (OID) und zum anderen die Unterscheidung zwischen *identischen* und *gleichen* Datenbankobjekten.
- ❑ *Vererbungsprinzip*: Ein *Subtyp* erbt die Attribute und Methoden eines Supertyps. Die geerbten Methoden können *überschrieben* werden. Das *späte Binden* der Methoden ermöglicht das dynamische, objektspezifische Auswählen der Methodenimplementierungen zur Laufzeit.

- ❑ *Substituierbarkeitsprinzip*: Ein Objekt eines Subtyp kann überall dort verwendet werden, wo ein Objekt eines zugehörigen Supertyps erwartet wird. Damit können heterogene Kollektionen aufgebaut werden.

Mit der Erweiterung des Typsystems kann eine gewöhnliche SQL-Tabelle, die wir im Folgenden als *Tupeltabelle* bezeichnen wollen, Spalten enthalten, die auf konstruierten, komplexen Datentypen beruhen. Eine Spalte kann

- ❑ *atomar*,
- ❑ *tupelwertig*,
- ❑ *kollektionswertig*,
- ❑ *objektwertig* oder
- ❑ *referenzwertig* sein.

Eine Tupeltabelle ist damit nicht mehr auf atomare Spalten, das heißt die erste Normalform [HS00], beschränkt.

### Typisierte Tabellen und Sichten

*Typisierte Tabellen* entsprechen in etwa dem Konzept einer *Klasse* in objekt-orientierten Datenbanken. Eine typisierte Tabelle basiert auf einem strukturierten Typ. Die Zeilen einer typisierten Tabelle repräsentieren Objekte, auf denen Methoden aufgerufen werden können. Typisierte Tabellen werden daher oft auch als *Objekttabellen* bezeichnet. Die erste Spalte einer typisierten Tabelle in SQL ist immer die *OID-Spalte* (siehe Abbildung 1.3), über welche die Zeilen mittels eines streng typisierten *Referenztyps* referenzierbar werden.

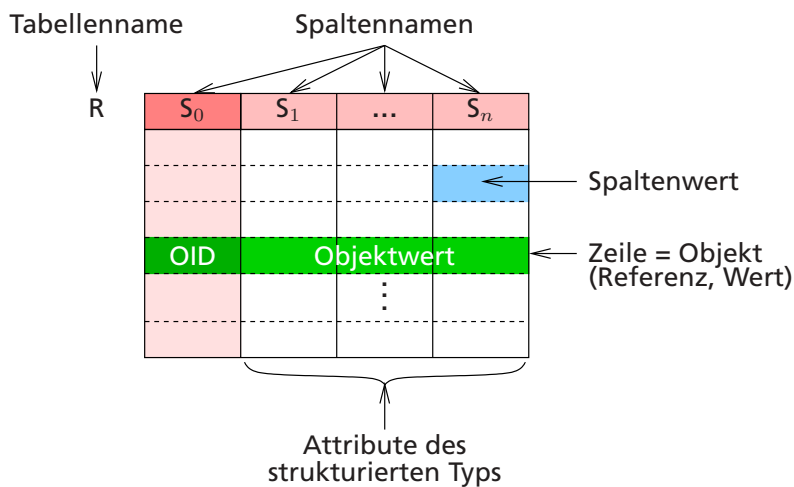


Abbildung 1.3: Typisierte Tabelle

Ähnlich zum *Spezialisierungskonzept* für Klassen in objektorientierten Datenbanken [Heu97, SST97] ermöglicht SQL:1999 die Organisation von Tabellen in Tabellenhierarchien. Eine typisierte Tabelle, die als *Subtabelle* einer anderen typisierten Tabelle definiert wird, legt eine Untermengenbeziehung zu der Supertabelle fest. Damit sind alle Zeilen der Subtabelle auch in der Supertabelle enthalten. Dies impliziert, dass die Zeilen einer Subtabelle speziellere Objekte repräsentieren. Der Typ der Objekte einer Subtabelle ist ein Subtyp des Typs der Objekte der Supertabelle (siehe Abbildung 1.4).

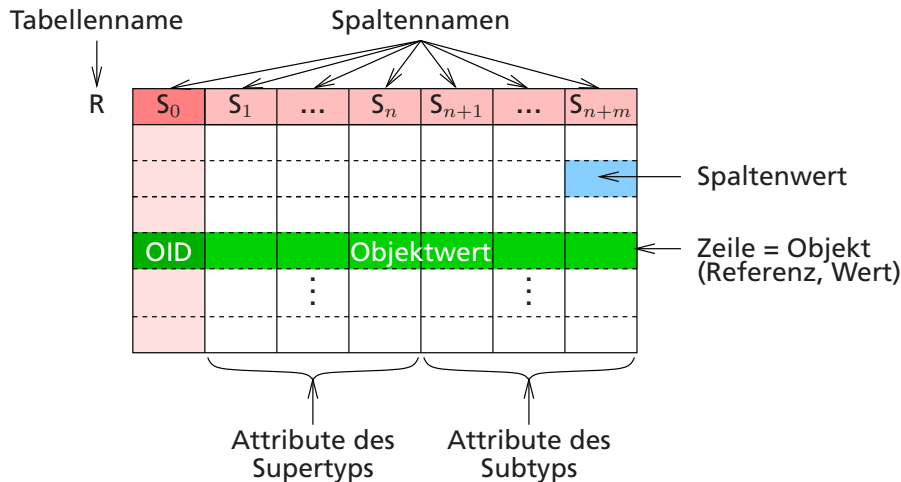


Abbildung 1.4: Subtabelle

*Typisierte Sichten* sind virtuelle, typisierte Tabellen. Eine typisierte Sicht in SQL entspricht einer Objektsicht, auf deren Zeilen die Methoden des zugrunde liegenden strukturierten Typs aufgerufen werden können. Analog zu Tabellenhierarchien können typisierte Sichten in Sichtenhierarchien organisiert werden. Eine typisierte Sicht, die als *Subsicht* einer anderen typisierten Sicht definiert ist, erweitert die »Sicht« der Supersicht. Das heißt, die Supersicht enthält implizit alle Zeilen der Subsicht.

## 1.4 Überblick über die Spezifikation von SQL:1999

Die Spezifikation von SQL:1999 war bei ihrer Publikation im September 1999 auf fünf Dokumente verteilt:

- *SQL – Part 1: Framework (SQL/Framework)* [ANS99c] gibt einen Überblick über den kompletten Standard und stellt insbesondere die verschiedenen Sprachebenen vor.

- ❑ *SQL – Part 2: Foundation (SQL/Foundation)* [ANS99d] beschreibt das Datenmodell von SQL und definiert die dazugehörigen Sprachen zur Datendefinition, Datenabfrage und Datenmanipulation.
- ❑ *SQL – Part 3: Call-Level Interfaces (SQL/CLI)* [ANS99e] beinhaltet die Definition von niederen Schnittstellen für den Zugriff von Anwendungen auf Datenbanken über Funktionsaufrufe.
- ❑ *SQL – Part 4: Persistent Stored Modules (SQL/PSM)* [ANS99f] normiert die prozedurale Erweiterung von SQL.
- ❑ *SQL – Part 5: Host Language Bindings (SQL/Bindings)* [ANS99g] spezifiziert die Einbettung von SQL-Anweisungen in Anwendungsprogrammen.

Nachträglich wurden weitere Teile von SQL:1999 veröffentlicht:

- ❑ *SQL – Part 9: Management of External Data (SQL/MED)* [ANS01b] definiert Datentypen und Funktionen, um auf Daten externer Datenquellen zugreifen zu können.
- ❑ *SQL – Part 10: Object Language Bindings (SQL/OLB)* [ANS00c] normiert die Java-Anbindung an SQL-Datenbanken.
- ❑ *SQL – Part 13: SQL Routines and Types Using the Java Programming Language (SQL/JRT)* [ANS02d] spezifiziert die Registrierung von externen Routinen und Datentypen, die in Java definiert wurden.

Die Spezifikation von SQL:1999 umfasst mehr als 3000 Seiten. Der im Vergleich zu den knapp 600 Seiten von SQL-92 stark angestiegene Umfang der Spezifikation lässt die Komplexität von SQL:1999 erahnen. Diese Komplexität wird in zukünftigen Versionen der SQL-Norm schon allein wegen der Forderung nach *Aufwärtskompatibilität* eher zu- als abnehmen.

### Überblick über die objektrelationalen Erweiterungen

SQL:1999 erweitert das »relationale« Datenmodell von SQL-92 um

- ❑ neue Basisdatentypen (**BOOLEAN**, **BLOB** und **CLOB**),
- ❑ unbenannte Typkonstruktoren (**ROW**, **ARRAY** und **REF**),
- ❑ benannte Typkonstruktoren zur Definition von benutzerdefinierten Datentypen in Form von Distinct-Typen (streng typisierte Kopien von Basisdatentypen) und strukturierten Typen (Objekttypen) mit Subtypenbildung (Typhierarchien),
- ❑ benutzerdefinierte Ordnungen für benutzerdefinierte Typen,
- ❑ benutzerdefinierte Cast-Funktionen für explizite oder implizite Typumwandlungen zwischen benutzerdefinierten Datentypen,
- ❑ benutzerdefinierte Transformationen für automatische Typumwandlungen zwischen SQL und externen Routinen,

- ❑ typisierte Tabellen (Objekttabellen) mit Subtabellenbildung (Tabellenhierarchien) und
- ❑ typisierte Sichten (Objektsichten) mit Subsichtenbildung (Sichtenhierarchien).

### Überblick über datenmodellunabhängige Erweiterungen

SQL:1999 enthält etliche Erweiterungen, die unabhängig vom objektrelationalen Datenmodell sind. Dies sind

- ❑ benutzerdefinierte Routinen (Prozeduren und Funktionen) zur Speicherung und Ausführung von Programmen im Datenbankserver,
- ❑ Trigger zur automatisierten Reaktion auf vordefinierte Datenbankereignisse,
- ❑ rekursive Anfragen und Sichten zur Berechnung von transitiven Hüllen und Erreichbarkeitsrelationen,
- ❑ OLAP-Anfragekonstrukte **CUBE** und **ROLLUP** für Anwendungen aus dem Bereich Data Warehousing,
- ❑ Ausdrücke in der **ORDER BY**-Klausel, die ein Sortierkriterium berechnen,
- ❑ neue Anfrageprädikate, etwa **FOR ALL**, **FOR SOME** und **SIMILAR TO**,
- ❑ **RESTRICT** als weitere Option für referenzielle Integritätsbedingungen,
- ❑ Rollenkonzept zur besseren Zugriffskontrolle,
- ❑ Savepoints zur Verbesserung der Recovery,
- ❑ Änderbarkeit von Vereinigungs- und Verbundsichten unter bestimmten Voraussetzungen und
- ❑ scroll- und änderbare Cursors zur komfortableren Anwendungsprogrammierung.

### Überblick über SQL/Multimedia

Auf der Basis der objektrelationalen Erweiterungen von SQL:1999 entstand mit SQL/Multimedia ein separater Standard, der Erweiterungspakete für die Verarbeitung von Multimediatdaten definiert. Dieser Standard setzt sich aktuell aus folgenden Teilen zusammen:

- ❑ *SQL Multimedia and Application Packages – Part 1: Framework* gibt einen Überblick über den SQL/Multimedia-Standard [ANS02a].
- ❑ *SQL Multimedia and Application Packages – Part 2: Full-Text* definiert ein Anwendungspaket für die Verarbeitung von Volltextdokumenten [ANS00b].

- ❑ *SQL Multimedia and Application Packages – Part 3: Spatial* spezifiziert ein Anwendungspaket für die Verarbeitung von räumlichen Objekten [ANS99b].
- ❑ *SQL Multimedia and Application Packages – Part 5: Still Image* normiert ein Anwendungspaket für die Verarbeitung von Bildobjekten [ANS01a].
- ❑ *SQL Multimedia and Application Packages – Part 6: Data Mining* stellt ein Anwendungspaket für Data Mining bereit [ANS02b].

In naher Zukunft soll noch ein Erweiterungspaket für die Verarbeitung von Zeitreihendaten folgen.

### Überblick über den Kern und die Zusatzpakete von SQL

Die Funktionalität von SQL:1999 besteht aus einem *Kern* und mehreren *Zusatzpaketen*. Der Kern enthält wenig neue Sprachkonstrukte im Vergleich zu SQL-92. Die meisten der neuen, objektrelationalen Spracherweiterungen sind auf die Zusatzpakete aufgeteilt. SQL:1999 sieht folgende Zusatzpakete vor:

- ❑ PKG001: *Erweiterte Datetime-Funktionalität*
- ❑ PKG002: *Erweiterte Integritätsbedingungen*
- ❑ PKG003: *OLAP-Erweiterungen*
- ❑ PKG004: *Prozedurale Erweiterungen von SQL*
- ❑ PKG005: *Funktionen des Call-Level-Interface*
- ❑ PKG006: *Grundlegende Objekt-Funktionalität*
- ❑ PKG007: *Erweiterte Objekt-Funktionalität*
- ❑ PKG008: *Aktive DB-Funktionalität*
- ❑ PKG009: *Multimedia-Funktionalität*

SQL-Dialekte werden als SQL-konform bezeichnet, wenn sie mindestens den SQL-Kern unterstützen. Je nach Unterstützung der Zusatzpakete befindet sich ein SQL-Dialekt auf einer bestimmten Konformitätsebene.

## 1.5 Überblick über die Spezifikation von SQL:2003

Mitte bis Ende 2003 soll die neue Norm SQL:2003 mit folgenden Neuerungen publiziert werden:

- ❑ *SQL – Part 2: Foundation (SQL/Foundation)* [ANS02f] sieht geringfügige Erweiterungen des SQL-Datenmodells und der damit verbundenen Sprachen vor. Neu hinzukommen werden insbesondere
  - ❑ der Basisdatentyp **BIGINT**,
  - ❑ der unbenannte Typkonstruktor **MULTISET**,

- ❑ generierte Spalten, deren Wert von anderen Spaltenwerten abgeleitet wird,
  - ❑ Sequenzgeneratoren zum inkrementellen Erzeugen von Werten,
  - ❑ Identitätsspalten zur automatischen Generierung künstlicher Schlüsselwerte,
  - ❑ tabellenwertige Funktionen und
  - ❑ eine Anweisung zum Kombinieren mehrerer Einfüge- und Änderungsanweisungen.
- ❑ Part 5 von SQL:1999 wird in den Part 2 von SQL:2003 verschoben. Die Sprachanbindungen werden um Konstrukte für die Handhabung der Datenmodellerweiterungen ergänzt. Die Java-Anbindung bleibt weiterhin in Part 10 definiert.
- ❑ *SQL – Part 11: Information and Database Schemata (SQL/Schemata)* [ANS02c] enthält die Spezifikation der Metatabellen, die aus Part 2 von SQL:1999 ausgelagert und gemäß den Datenmodellerweiterungen von SQL:2003 um neue Metatabellen erweitert wurden.
- ❑ *SQL – Part 14: XML-related Specifications (SQL/XML)* [ANS02e] normiert die Verknüpfung von SQL und XML. Er definiert einen Basisdatentyp **XML** mit dazugehörenden Funktionen und Abbildungen zwischen SQL und XML.

## 1.6 Fokus und Gliederung des Buches

Der Hauptfokus dieses Buches liegt auf der Vorstellung und Diskussion der objektrelationalen Konzepte und Sprachkonstrukte von Standard-SQL und den SQL-Dialekten führender objektrelationaler Datenbanksysteme.

Dieses Buch behandelt die zentralen Sprachkonstrukte aus den Parts 2, 4, 10, 13 und 14 der SQL-Spezifikation. Es stellt die DDL, DML und Anfragesprache von SQL vor (Part 2), zeigt die prozeduralen Erweiterungen von SQL auf (Part 4), diskutiert die Java-Call-Level-Schnittstelle und die Einbettung von SQL in Java (Part 10), präsentiert die Möglichkeiten zur Erweiterung der Datenbank durch externe Java-Datentypen und -Methoden (Part 13) und skizziert die Integration von SQL und XML (Part 14). Die sonstigen Call-Level-Schnittstellen von SQL (Part 3), die Einbettung von SQL in sonstige Programmiersprachen (Part 5), die Anbindung an externe Datenbanken (Part 9) und das Informationsschema (Part 11) werden nicht betrachtet.

Eine Spezialität dieses Buches ist es, SQL:1999 und SQL:2003 in Bezug zu den aktuell relevanten objektrelationalen SQL-Dialekten zu setzen. Wir betrachten die objektrelationalen Features folgender SQL-Dialekte:

- Oracle9i Release 2,
- IBM DB2 Version 8.1,
- IBM Informix Version 9.3.1 und
- PostgreSQL Version 7.3.

Im weiteren Verlauf dieses Buch werden diese SQL-Dialekte durchgehend als *Oracle*, *DB2*, *Informix* bzw. *Postgres* bezeichnet.

Die Gliederung des Buches ist wie folgt: Kapitel 2 stellt die Sprachkonstrukte von Standard-SQL zur objektrelationalen Datendefinition vor. Kapitel 3 geht auf die objektrelationale Datendefinition in Oracle, DB2, Informix und Postgres ein. Die Kapitel 4 und 5 stellen die Sprachkonstrukte für die Datenabfrage in Standard-SQL bzw. in den SQL-Dialekten vor. Die Datenmanipulation in Standard-SQL und in den SQL-Dialekten wird in den Kapiteln 6 bzw. 7 behandelt. Die objektrelationalen SQL-Schnittstellen JDBC und SQLJ werden in Kapitel 8 im Zusammenhang mit Standard-SQL präsentiert. Kapitel 9 geht auf die Spezialität der konkreten Implementierungen dieser Schnittstellen in den objektrelationalen Datenbanksystemen Oracle und Informix ein. Kapitel 10 skizziert den zukünftigen SQL/XML-Standard. Ein kurzes Resümee in Kapitel 11 beschließt das Buch.

## 1.7 Literaturhinweise

Die relationalen Grundlagen von SQL:1999 werden ausführlich in [MS01, PT00] beschrieben. Die objektrelationalen Erweiterungen von SQL:1999 werden in [Mel02] behandelt. Die Standardisierungsdokumente sind aufgrund der zum Teil unübersichtlichen Definitionen bestehend aus vielen Klauseln und Ausnahmen sowie einigen Inkonsistenzen nur bedingt als »Lektüre« zu empfehlen.

Die Tutorials [Mel99, Mat00] zeigen die wesentlichen Erweiterungen von SQL:1999 anhand von konkreten Beispielen auf. Einen umfassenden Überblick über die Unterstützung von Integritätsbedingungen in SQL:1999 und SQL-Dialekten gibt [TG01]. Eine Übersicht über die in SQL:1999 und SQL-Dialekten angebotenen Operatoren der Schemaevolution bietet [Tür01].

Das Buch [GP99] war kurz vor dem Verabschieden von SQL:1999 erschienen. Dadurch berücksichtigt es nicht alle kurzfristigen Änderungen der veröffentlichten Norm. Gleiches gilt für das etwas früher erschienene Buch [For99].

Die Sprachkonstrukte der SQL-Dialekte sind den Handbüchern zu entnehmen: Oracle [Ora02a, Ora02d], DB2 [IBM02a], Informix [IBM01a, IBM01b] und Postgres [Pos02b, Pos02a]. Die Bücher [Käh99, Bro00, LK00, BM00, Mom01, HP02, Gep02, Boe03] behandeln die objektrelationalen

Aspekte einzelner SQL-Dialekte. Das Buch [Bro00] bietet dabei einen vertieften Blick hinter die objektrelationalen Erweiterungen von Informix.

Die Webseite <http://java.sun.com/jdbc/> enthält die Spezifikation der niederen Java-SQL-Schnittstelle JDBC [Sun98, Sun99, Sun01]. Die höhere Java-SQL-Schnittstelle SQLJ ist Teil der SQL-Norm [ANS00c, ANS02d]. Die Bücher [SS00, ME00] stellen die grundlegenden Aspekte von JDBC und SQLJ gut aufbereitet vor. Die Handbücher der jeweiligen Datenbankhersteller enthalten umfassendes Material über die objektrelationalen Datenbankschnittstellen JDBC und SQLJ.

SQL/XML [ANS02e] ist Teil der zukünftigen SQL-Norm. Die Spezifikation verweist auf die XML-Anfragesprache *XQuery*, deren Spezifikation zurzeit ebenfalls als Draft [W3C02] vorliegt. Das Buch »Advanced SQL:1999« [Mel02] enthält eine Übersicht über SQL/XML. Die Bücher [KM03, RV03] gehen allgemeiner auf das Zusammenspiel zwischen Datenbanken und XML ein.