

Kapitel 6: Objektrelationale Datenbankmodelle

	Relationales Datenmodell (RDM)	Netzwerk- und Hierarchisches Datenmodell (NDM, HDM)	Objekt-orientierte Datenmodelle (OODM)	Objekt- relationale Datenmodelle
Überblick über die Konzepte	3.1	4.1 4.2	5.1	6.1
Darstellung von Assoziationen				
Datendefinition				
Anfragen				
Aktualisierungsoperationen				
Spezifika	3.2 SQL		5.2 ODMG	6.2, 6.3

Einführung in Datenbanksysteme Objektrelationale Datenbankmodelle 6.1.1

Kapitel 6: Objektrelationale Datenbankmodelle

Lernziele und Überblick

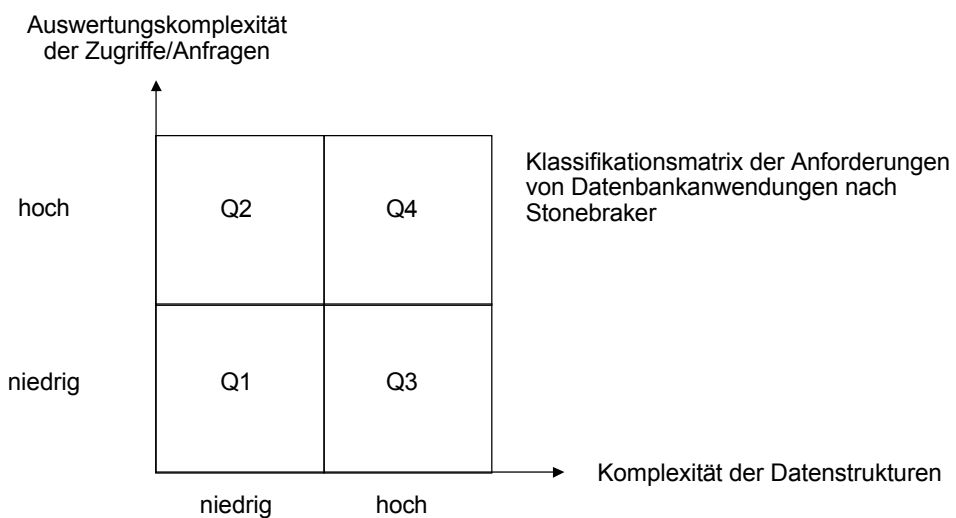
- 6.1 Klassifikation von Datenbanksystemen
 - Anforderungen an Datenbanksysteme
 - Einordnung *objektrelationaler* Datenbanksysteme
- 6.2 Erweiterte relationale Datenbanksysteme (Beispiel: Oracle 8)
- 6.3 Objektrelationale Middleware

Einführung in Datenbanksysteme Objektrelationale Datenbankmodelle 6.1.2

Literatur zu 6.1

- ❑ Michael Stonebraker, Dorothy Moore:
Object-Relational DBMSs - The Next Great Wave
The Morgan Kaufmann Series in Data Management Systems, 1996
- ❑ C. J. Date, Hugh Darwen
Foundation for Object/Relational Databases - The Third Manifesto
Addison-Wesley, 1998
- ❑ Thomas M. Connolly, Carolyn E. Bragg:
Database Systems (Chapter 23)
A Practical Approach to Design, Implementation, and Management
Addison-Wesley, 2nd edition, 1998

6.1 Klassifikation von Datenbanksystemen



Quadrant 1: Einfache Daten, keine Anfragen

- ❑ Typisches „Datenbankmanagementsystem“ dieser Klasse: Dateiverwaltungssystem des Betriebssystems (z.B. NTFS, FAT, NFS).
- ❑ Dateien werden nicht durch Anfragen bearbeitet, sondern
 - vom Anwendungsprogramm vollständig eingelesen,
 - innerhalb des Anwendungsprogramms bearbeitet und
 - nach Abschluß der Änderungen vollständig ins Dateisystem zurückgeschrieben.
- ❑ Das Dateiverwaltungssystem sorgt nur für die physikalische Speicherung der Daten.
- ❑ Einzige Datenstruktur: Die Datei - ein byteadressierbarer Adreßraum.
- ❑ Für die korrekte Interpretation und Benutzung der Inhalte einer Datei ist das jeweilige Anwendungsprogramm (z.B. Texteditor, Grafikprogramm) selbst verantwortlich.
- ❑ NDM und HDM würden auch in Q1 eingeordnet werden.

Quadrant 2: Einfache Daten mit Anfragen

- ❑ Das DBMS verwaltet Daten mit Hilfe einfacher, vordefinierter Datenstrukturen-
- ❑ Gespeicherte Daten können mit Hilfe einer deklarativen Anfragesprache untersucht und manipuliert werden.
- ❑ Typische Vertreter dieser Klasse: Relationale Datenbanksysteme (z.B. Oracle, Adabas D, CA-Ingres)

- Basisdatenstrukturen: Tabellen, Tabellenzeilen/Tupel
- Anfragesprache: SQL (SQL-92)

– Manipulation von Daten:

```
update angestellte
set gehalt = gehalt * 1,03
```

– Lesen von Daten:

```
select nachname, vorname
from angestellte
where abteilung = 'Verkauf'
```

Beispiel zu Q2: Relationale Datenbanksysteme

- ❑ Tupel innerhalb einer Relation setzen sich aus Werten weniger vordefinierter Wertebereiche (*Domains*) zusammen, z.B. INTEGER, FLOAT, DATE, VARCHAR.
- ❑ Tupel sind flach, die Verwendung komplexer Datenstrukturen innerhalb eines Tupels ist nicht möglich.
- ❑ Anwendungsprogramme beschreiben durch Anfragen Eigenschaften der gesuchten Tupelmenge.
- ❑ Die Bestimmung der Anfrageergebnisse übernimmt das Datenbanksystem (Auswahl von Zugriffspfaden, Optimierung der Anfrage).
- ❑ Das Datenbanksystem
 - garantiert die Dauerhaftigkeit und Integrität des von ihm verwalteten Datenbestandes,
 - löst Zugriffskonflikte, wie sie im Mehrbenutzerbetrieb auftreten können und
 - sichert die Daten gegen unbefugten Zugriff.

Quadrant 3: Komplexe Daten ohne Anfragen

- ❑ Daten werden in Form komplexer Objekte gespeichert.
- ❑ Vorhandene Objekte lassen sich beliebig zu neuen Objekten aggregieren.
- ❑ Zwischen verschiedenen Objekten können komplexe Beziehungen bestehen, z.B. Nachbarschaftsbeziehung zwischen Polygonen in einer CAD-Anwendung.
- ❑ Typische Systeme dieser Klasse: persistente, objektorientierte Programmiersprachen (z.B. Smalltalk, Tycoon) und kommerzielle Objektspeichersysteme (z.B. O₂, Objectivity).
- ❑ Objekte eines Anwendungsprogramms werden mit all ihren Attributen und Beziehungen dauerhaft gespeichert.
- ❑ Explizite Konvertierungen aus oder in eine externe Darstellung, wie bei der Speicherung in Dateisystemen oder relationalen Datenbanken sind nicht erforderlich.
- ❑ Außer der unterschiedlichen Lebensdauer „sieht“ der Anwendungsentwickler keinen Unterschied zwischen persistenten und transienten/temporären Objekten.

Quadrant 4: Anfragen auf komplexen Daten

Beispiele

- Suche in geographischen Daten
(„Alle Bushaltestellen im Umkreis von 500 m“)
- Volltextsuche
(„In welchen Texten steht das Wort *Objekt* in der Nähe von *relational* ?“)
- Bilddatenauswertung (z.B. Warburg Electronic Library): „Alle Bilder, auf denen Konrad Adenauer zu sehen ist“
- Audiodaten („Alle Vorlesungsmitschnitte, in denen das Wort *äh* weniger als zwanzig Mal ausgesprochen wird“)

Die zu speichernden Daten sind komplex strukturiert, z.B.

- Nahverkehrsnetz (räumliche Lage von Haltestellen, Fahrpläne)
- Bilder: abgebildete Objekte und Personen, Beziehungen (Person *x* steht vor, neben, hinter Person *y*)
- Audio: Name des Sprechers, Stimmuster, Sprechgeschwindigkeit, Aufnahmeverfahren

Bewertung der RDBs und Objektspeicher (nach Stonebraker)

Die Struktur der Anwendungsdaten und ihre Beziehungen sind zu komplex, als daß herkömmliche Datenbanksysteme sie effizient bearbeiten könnten:

Relationale Datenbanken

- können den Zugriff über einzelne, einfache Attribute einer Tabelle beschleunigen, z.B. durch einen B-Baum oder eine Hashtabelle,
- erlauben die Definition solcher Zugriffsstrukturen aber nur für die Werte ihrer Basisdatentypen,
- können keine Indizes erzeugen, die Strukturen *innerhalb* eines einzelnen Wertes berücksichtigen (z.B. die Wörter innerhalb eines Strings) und
- bieten nur einfache Anfrageprädikate (z.B. =, >, <).

Objektspeichersysteme

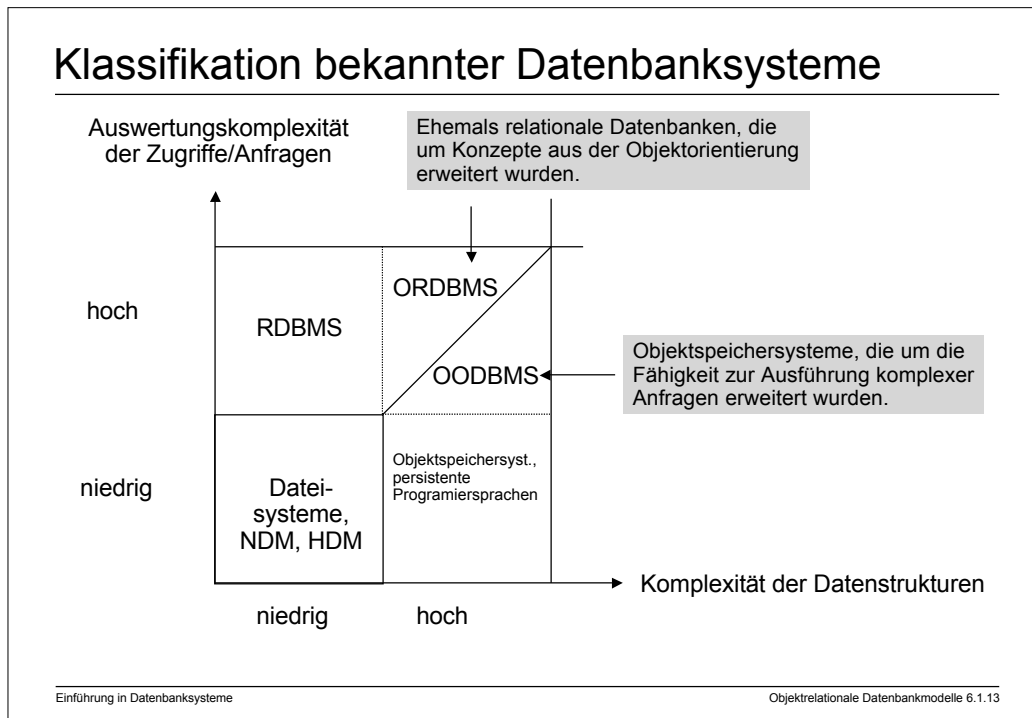
- bieten komplexe Strukturen und komplexe Methoden,
- unterstützen aber den navigierenden Objektzugriff besser als den anfrageorientierten Zugriff.

Anforderungen an ORDBMS (nach Stonebraker)

- ❑ Aus der relationalen Welt übernommene Konzepte:
 - deklarative Anfragesprache
 - Sicherung von Konsistenz, Integrität und Dauerhaftigkeit der Daten
- ❑ Aus der objektorientierten Welt übernommene Konzepte:
 - beliebige, orthogonale Kombinierbarkeit existierender Datenstrukturen zu neuen, komplexeren Strukturen
 - Vererbung
 - Objektreferenzen
 - Objekte kapseln nicht nur Daten, sondern auch Verhalten
- ❑ Erweiterbarkeit der Anfragesprache um
 - Funktionen und Operatoren für die Nutzung benutzerdefinierter Datenstrukturen

Diskussion

- ❑ Stonebrakers Klassifikation für Datenbanksysteme ist nur sehr grob:
 - Objektorientierte Datenbanksysteme erfüllen z.B. mit der *Object Query Language* (OQL) bereits viele Anforderungen, die Stonebraker an eine mächtige objektrelationale Anfragesprache stellt und
 - ehemals rein relationale Datenbanken erweitern ihren Funktionsumfang um objektorientierte Konzepte, wie Referenzen, Vererbung oder benutzerdefinierte Typen/Klassen (SQL 99).
- ❑ Bisher existiert *keine* geschlossene und allgemein akzeptierte objektrelationale Theorie oder entsprechende Standards (vgl. OQL vs. SQL 99).
- ❑ Bestehende „objektrelationale“ Datenbanken erweitern existierende relationale Datenbanksysteme oder Objektspeichersysteme um „nützliche“ Konzepte der jeweils anderen Systemklasse.



- ## Notwendigkeit objektrelationaler Systeme
- ❑ Objektorientierte und relationale Systeme unterscheiden sich hinsichtlich
 - Terminologie,
 - theoretischen Grundlagen und
 - typischen Anwendungsbereichen.
 - ❑ Diese historisch bedingte Unterscheidung ist bei der Entwicklung vieler datenintensiver Anwendungen nicht (mehr) möglich.
 - ❑ Verbilligung von Massenspeichern und Beschleunigung der Rechnerkommunikation erlauben neben einfachen, strukturierten Daten auch die persistente Speicherung und den Transport großer, heterogen strukturierter Datenmengen.
 - ❑ Anwendungsentwickler müssen sich in zwei verschiedenen Systemwelten auskennen, wenn sie sowohl auf die Konzepte relationaler wie objektorientierter Technik angewiesen sind.
 - **Beispiel:** Zugriff auf eine historisch gewachsene relationale Datenbank mit Hilfe eines (objektorientierten) Java-Programms.
- Einführung in Datenbanksysteme
- Objektrelationale Datenbankmodelle 6.1.14

Aktuelle Entwicklung objektrelationaler Software

Ziel: Softwaresysteme, welche

- die Vorteile von relationaler und objektorientierter Technik in sich vereinigen,
- Brüche zwischen den ehemals getrennten Datenmodellen überwinden,
- den Entwurf komplexer Anwendungen und Datenstrukturen innerhalb eines einheitlichen theoretischen und terminologischen Rahmens erlauben.

Entwicklungslinien objektrelationaler Systeme:

- Alternative 1: Erweiterung relationaler Datenbanken um objektorientierte Konzepte
 - Oracle (Oracle 8 mit „*Objects Option*“, s. Abschnitt 6.2)
 - Informix (nach dem Kauf von Illustra), Postgres
 - UniSQL
- Alternative 2: Nutzung vollständig objektorientierter Datenbanksysteme, die ohnehin relationale Funktionalität enthalten (vgl. SQL \square OQL, s. Abschnitt 5).
- Alternative 3: Integration relationaler Datenbanken in objektorientierte Anwendungen (s. Abschnitt 6.3)