

















Datenbanken 1 für Medieninformatiker WS 06/07

SOLD	ONHAND	RATING	ARTIST	TITLE	COVER	VIDEO	MUSIC	SCRIPT
234	59	PG-13	Arnold	The Exterminator				
13	45	R	Kevin	Dancing with Bulls				
1295	209	G	Glenn	101 Doll Imitations				
379	112	G	Buzz	Toy Glory				

5. Objekt-relationale Systeme

5.1. Erweiterungen des Relationalen Modells

5.2. SQL-Erweiterungen

5.3. MM-DB

Beschränkungen des Relationalen Modells

- NF1: Atomare Attribute
- nicht erweiterbares Typkonzept
- keine typspezifischen Operationen
- keine Vererbung

Als Einschränkung besonders empfunden bei

- Mengenwertigen Attributen
- sicherheitsrelevanten komplexen Operationen auf mehreren Attributen eines Datensatzes
- Nicht-atomaren Daten mit such-relevanter Struktur (MM-Daten)

Objektrelationale Erweiterungen

- **NF²** – *Non First Normal Form*
 - Mengenwertige Attribute
 - LOBs
 - Tabellenwertigen Attribute
 - Typdefinition
- **ORDB** – *Objekt-relationale Datenbank*
 - Typhierarchie und Vererbung
 - Objektidentität
 - Operationsdefinition
 - (Polymorphie)

Wieso objekt-*relational*?

- Wichtige Konzepte bleiben erhalten:
- Datenunabhängigkeit
- Abfragekalkül
- Tabelle als Grundkonzept
- Stonebraker-Matrix (1996) – Versuch einer Kategorisierung

	atomare Daten	komplexe Daten
Anfragen	<i>RDB</i>	<i>ORDB</i>
keine Anfragen	<i>Filesystem</i>	<i>OODB</i>

- so **nicht zutreffend**: in der Praxis starke Durchdringung OODB-ORDB

State of the Art

- Normungsprojekt SQL-3 zu ambitioniert
- SQL-1999 große Teilmenge von SQL-3
- enthält mehrere OR-Stufen
- faktische Standardisierung verzögert sich durch viele proprietäre Erweiterungen von SQL-92
- (SQL-4 ist in Arbeit)

SQL-Erweiterungen (DDL)

- Große Objekte (LOBs: Large Objects)
- Mengenswertige Attribute
- Geschachtelte Relationen
- Typdeklarationen (UDTs: User-defined types)
- Referenzen
- Objektidentität
- Pfadausdrücke
- Vererbung
- Operationen

Beispiele im Folgenden aus

<http://www.dbai.tuwien.ac.at/education/dbs/fohlen/Kapitel14.pdf>

LOBs

- Idee:
 - Datentypen für sehr große Attributwerte, z.B.: Multimedia-Daten.
 - Größe bis zu einigen Giga-Byte.
 - DBMS stellt (optional) spezielle Optimierungen für LOBs zur Verfügung, z.B.: kein Logging, Komprimierung, ...
- Verschiedene Arten von LOBs:
 - CLOB (= Character Large Object): für lange Texte
 - BLOB (= Binary Large Objects): vom DBMS nicht interpretierbare binäre Anwendungsdaten
 - NCLOB (= National Character Large Objects): für Unicode

LOB-Beispiel

```
create table Professoren
(  PersNr      integer primary key,
   Name        varchar(30) not null,
   Rang        character(2)
              check (Rang in ('C2', 'C3', 'C4')),
   Raum        integer unique,
   Foto        BLOB(2M),
   CV          CLOB(75K) );
```

```
LOB (Lebenslauf) store as
(  tablespace Lebenslaeufe
   storage (initial 50M next 50M) );
```

UDT (User Defined Types)

- Idee:
 - Zusätzlich zu den SQL-Standard-Typen kann der Benutzer selbst Datentypen definieren.
- Unterscheidung:
 - wert-basierte Typen (attribute types)
 - Objekt-Typen (row types).

Beispiel wertbasierter Typ

```
CREATE DISTINCT TYPE NotenTyp AS DECIMAL (3,2);
```

```
Create Table Pruefen (  
    MatrNr    INT,  
    VorlNr    INT,  
    PersNr    INT,  
    Note      NotenTyp);
```

Listen-Typen (für nicht-atomare Attribute)

Mengenwertige Attribute:

- Einem Tupel (Objekt) wird in einem Attribut eine Menge von Werten zugeordnet.
- Beispiel:
 - Mengenwertiges Attribut ProgrSprachenKenntnisse in der Relation Studenten.

Objekt-Typen (für nicht-atomare Attribute)

Geschachtelte Relationen (Strukturierte Attribute):

- Attribute dürfen selbst wiederum Relationen sein.
- Beispiel:
 - ein Attribut absolviertePrüfungen in der Relation Studenten
 - enthält Menge von Prüfungen-Tupeln
 - Jedes Prüfungen-Tupel besteht selbst wieder aus Attributen, wie z.B. Note und Prüfer.
- Anforderung an Anfragesprache:
 - Schachtelung / Entschachtelung

Beispiel Listen- und Objekttypen

```
CREATE OR REPLACE TYPE PruefungenTyp AS OBJECT (  
    Inhalt      REF VorlesungenTyp,  
    Pruefer     REF ProfessorenTyp,  
    Note        DECIMAL(3,2),  
    Datum Date);
```

```
CREATE OR REPLACE TYPE PruefungsListenTyp  
    AS TABLE OF PruefungenTyp;
```

```
CREATE OR REPLACE TYPE StudentenTyp AS OBJECT (  
    MatrNr      NUMBER,  
    Name        VARCHAR(20),  
    Semester    NUMBER,  
    absolviertePruefungen PruefungsListenTyp);
```

Referenzen, Pfadausdrücke

Referenzen

- Mögliche Attribut-Werte: Referenzen auf Tupel/Objekte
- Zur Realisierung von Beziehungen ohne Fremdschlüssel.
- Verallgemeinerung: Menge von Referenzen als Attributwert
- → N:M-Beziehungen direkt darstellbar
- Beispiel: `Studenten.hört` als Menge von Vorlesungs-Referenzen

Objektidentität:

- Referenzen setzen eindeutige, unveränderliche Objektidentität der Objekte (Tupel) voraus.

Pfadausdrücke (in der Anfragesprache):

- mit Referenzattributen, z.B.: `s.hoert.gelesenVon.Name`

Vererbung

Idee:

- Ein komplexer Typ kann von einem Obertyp erben:
- Der Untertyp enthält alle Attribute und Operationen des Obertyps.

Beispiel Vererbung

```
CREATE TYPE AngestelltenTyp AS OBJECT  
(  
  PersNr      INT,  
  Name        VARCHAR(20))  
INSTANTIABLE ... ;
```



ProfessorenTyp
extends
AngestelltenTyp

```
CREATE TYPE ProfessorenTyp UNDER AngestelltenTyp AS  
(  
  Rang        CHAR(2),  
  Raum        INT) ... ;
```

```
CREATE TYPE AssistentenTyp UNDER AngestelltenTyp AS  
(  
  Fachgebiet VARCHAR(20),  
  Boss        REF(ProfessorenTyp)) ... ;
```

Operationen

Idee:

- Stored Procedures in SQL oder proz. Erweiterung
- Einem Typ zugeordnet → METHOD

Vorteil:

- Objekt des Typs per Abfrage auffindbar
- Typsetifischer Methodenaufruf direkt in SQL anwendbar
- Ergebnis in SQL verwendbar

Methoden-Beispiel

```
CREATE OR REPLACE TYPE ProfessorenTyp AS OBJECT (  
    PersNr      NUMBER,  
    Name       VARCHAR(20),  
    Rang       CHAR(2),  
    Raum       Number,  
    METHOD Notenschnitt() RETURNS NUMBER,  
    METHOD FUNCTION Gehalt() RETURNS NUMBER);  
)
```

```
CREATE INSTANCE METHOD Gehalt() RETURNS NUMBER  
FOR ProfessorenTyp  
RETURN SELECT MAX(Gehalt) FROM Gehaltsliste  
WHERE PersNr = SELF.PersNr;
```

Methoden-Beispiel

```
CREATE OR REPLACE TYPE AngestelltenListenTyp  
  AS TABLE OF AngestelltenTyp;
```

```
CREATE TABLE Forschungsgruppe (  
  Name          VARCHAR          PRIMARY KEY,  
  LeiterProfessorenTyp,  
  Referent      AssistentenTyp,  
  Mitarbeiter   AngestelltenListenTyp );
```

```
SELECT AVG(Leiter.Gehalt()) FROM Forschungsgruppen;
```

Anmerkung: Leere Parameterklammern können nach SQL-1999 weggelassen werden

Typisierte Tabellen

Idee

- Tabellentyp definierbar
 - Mehrere Tabellen dieses Typs erzeugbar (OF)
- Tabellenhierarchie
 - Tabelle aus Tabelle ableitbar (UNDER)
- Typenhierarchie
 - Untertypen ableitbar (UNDER)

Problem

- Tabellenhierarchie und Datentyphierarchie parallel

Typisierte Tabellen - Beispiel

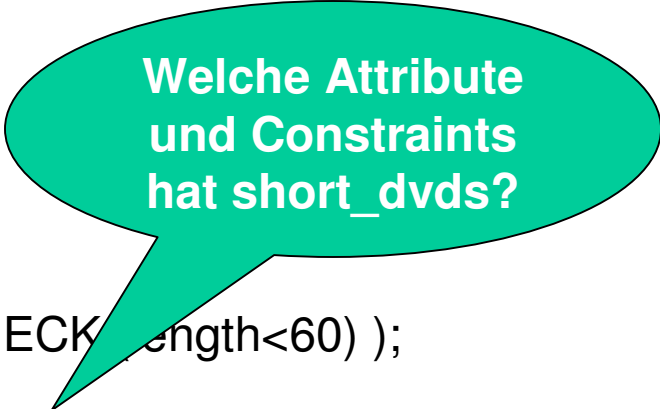
```
CREATE TYPE movie AS (  
  title      VARCHAR,  
  genre      VARCHAR,  
  length     NUMBER)  
INSTANTIABLE  
REF IS SYSTEM GENERATED;
```

```
CREATE TYPE dvd UNDER movie AS (  
  rental_price  NUMBER,  
  extras        FeatureListType)  
INSTANTIABLE
```

```
CREATE TABLE short_movies OF movie (  
  length WITH OPTIONS CONSTRAINT c_short CHECK (length < 60) );
```

```
CREATE TABLE short_dvds OF dvd UNDER short_movies (  
  rental_price WITH OPTIONS CHECK c_short_price  
  (CHECK (rental_price < 2.00) );
```

Beispiel aus Melton, Advances SQL:1999, San Francisco 2003



Welche Attribute
und Constraints
hat short_dvds?

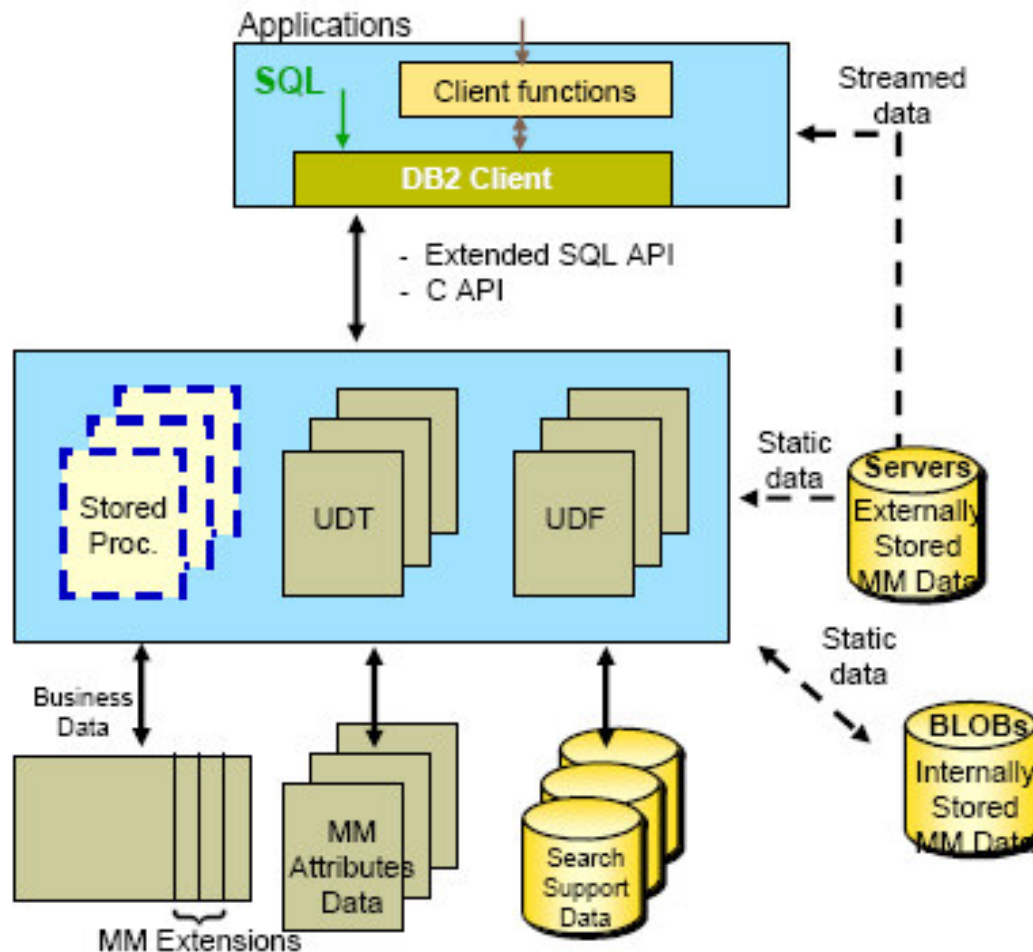
Externe Daten / Datalinks

- Große Datenmenge liegen oft auf zentralen Servern
- Daten unter die Kontrolle eines DBMS stellen: Beschreibungsdaten hinzufügen, als Suchkriterien nutzen
- Daten auf dem externen Server belassen, dort verwalten, Struktur nutzen
- → Typ **DATALINK**
- enthält Kontrolloptionen wie Lese-Schreibrechte, Recovery-vorgaben etc

Externe Typen / Data Extenders

- Große Dateneinheiten sind oft strukturiert
 - LOB wird dem nicht gerecht
- Typ / Struktur **außerhalb von SQL wohldefiniert**
 - Beispiel Bildformate
- **Daten und Struktur** unter die Kontrolle eines DBMS stellen
 - oft verbunden mit externer Speicherung
- **"Plugin"-Technik** für Datentypen (nicht SQL-1999):
 - DB2 – Data Extenders
 - Oracle – Data Cartridges
 - Illustra – Data Blades
- Besonders wichtig für **Multi-Media-Datenbanken**
 - Typen wie image, video, audio, graphics, text...
 - mit der Fähigkeit, externe Standardformate zu lesen (Headerinfo) und zu speichern

Extern → Heterogene System



Es gibt noch viel Interessantes zu entdecken:

Rechtevergabe, Transaktionen,
Multi-User-Betrieb, DB-Administration...

Aber für dieses Semester reicht es!



Nächstes Mal:
Klausurvorbereitung