Vorlesung Datenbanksysteme vom 25.11.2015

Objektrelationale Datenbanken

- Konzepte objektrelationaler DBs
- SQL:1999
- OO vs. OR

Konzepte objektrelationaler Datenbanken

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

Große Objekte (LOBs)

- Idee:
  - Datentypen, die es erlauben, auch sehr große Attributwerte zu speichern, z.B.: Multimedia-Daten.
  - Die Größe kann bis zu einigen Giga-Byte betragen.
  - 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): für Anwendungsdaten, die vom Datenbanksystem gar nicht interpretiert sondern nur gespeichert werden sollen.
  - NCLOB (= National Character Large Objects): für Unicode Texte (mit > 1 Byte pro Zeichen).

Große Objekte (LOBs)

- 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,
   Passfoto BLOB(2M),
   Lebenslauf CLOB(75K) );

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

UDTs: User-defined types

- Idee:
  - Zusätzlich zu den built-in types kann der Benutzer selbst Datentypen definieren.
  - Unterscheidung: wert-basierte Typen (attribute types) und Objekt-Typen (row types).
- Beispiel (attribute type):

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

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

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.
- **Geschachtelte Relationen:**
  - Attribute dürfen selbst wiederum Relationen sein.
  - Beispiel: In der Relation Studenten ein Attribut absolviertePrüfungen, unter dem die Menge von Prüfungen-Tupeln gespeichert ist.
  - Jedes Tupel dieser geschachtelten Relation besteht selbst wieder aus Attributen, wie z.B. Note und Prüfer.
- Anforderung an Anfragesprache:
  - Schachtelung / Entschachtelung möglich

Beispiel: Geschachtelte Relationen

```
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, Objektidentität, Pfadausdrücke

- **Referenzen:**
  - Mögliche Attribut-Werte: Referenzen auf Tupel/Objekte
  - Zur Realisierung von Beziehungen ohne Fremdschlüssel.
  - Verallgemeinerung: Menge von Referenzen als Attributwert => N:M-Beziehungen ohne separate Beziehungsrelation.
  - z.B.: Studenten.hört als Menge von Referenzen auf Vorlesungen
- **Objektidentität:**
  - Referenzen setzen eindeutige, unveränderliche Objekt-identität der Objekte (Tupel) voraus.
- **Pfadausdrücke** (in der Anfragesprache):
  - mit Referenzattributen, z.B.: s.hoert.gelesenVon.Name

Vererbung

Ein komplexer Typ kann von einem Obertyp erben:

- Der Untertyp enthält alle Attribute und Operationen des Obertyps.
- Beispiel:

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

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:
  - Den Objekttypen zugeordnet oder auch nicht
  - Einfache Operationen können direkt in SQL implementiert werden.
  - Komplexere Operationen können in einer DB-Programmier-sprache (z.B.: PL/SQL in Oracle) bzw. in einer Wirt-Sprache (z.B.: Java, C, C++, etc.) "extern" realisiert werden.

- Beispiel:

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

Standardisierung in SQL:1999

- SQL-92:
  - Derzeit realisierter Standard der kommerziellen relationalen Datenbanksysteme
  - verschiedene Stufen der Einhaltung:

    "Entry level" ist die schwächste Stufe
- SQL:1999
  - Objekt-relationale Erweiterungen
  - Weitere Erweiterungen, z.B.: Trigger, stored Procedures, erweiterte Anfragesprache (wie transitive Hülle)
- Problem: Viele Systeme haben schon ihre eigenen proprietären Erweiterungen von SQL-92 realisiert.
  - Anpassung an den Standard kann noch einige Zeit dauern

OO- vs. OR-Datenbanken

- Objektorientierte Datenbanken:
  - "revolutionärer" Weg.
  - Ausgangspunkt: OO-Grundkonzepte
  - Ergänzt um bewährte Konzepte der relationalen DBs.
- Objektrelationale Datenbanken:
  - "evolutionärer" Weg.
  - Ausgangspunkt: relationale DB-Konzepte
  - Ergänzt um bewährte OO-Konzepte
- Unterscheidung
  - gar nicht so klar
  - mögliches Unterscheidungsmerkmal: "Tabellen" in OO unbedeutend, in OR immer noch das zentrale Konstrukt.