

## Praktikum Datenbanken / DB2 Woche 10: Objektrelationale Konzepte

Raum: LF 230

Nächste Sitzung: 19./22. Januar 2004

Die Dokumentation zu DB2 steht online zur Verfügung. Eine lokale Installation der Dokumentation findet sich unter der Adresse [http://salz.is.informatik.uni-duisburg.de/db2doc/de\\_DE/index.htm](http://salz.is.informatik.uni-duisburg.de/db2doc/de_DE/index.htm).

Die englischsprachigen Handbücher für IBM DB2 V8.1 liegen als PDF Dateien lokal im Verzeichnis `/usr/projects/db2doc/`. Diese Handbücher sind sehr umfangreich und sollten nur zum Betrachten am Bildschirm herangezogen und nicht ausgedruckt werden. Die Referenzen in diesem Arbeitsblatt beziehen sich auf diese Handbücher. Die gleichen Informationen sind jedoch auch in der Online-Dokumentation zu finden.

### Einführung

IBM DB2 V8.1 ist ein objektrelationales Datenbankmanagementsystem. Obwohl das relationale Datenbankkonzept ausgereift, leistungsfähig und theoretisch fundiert ist, ist es für viele Anwendungsfälle unbefriedigend, da es die Modellierung komplexerer Aufgabenstellungen nicht unterstützt. Genau hierzu sind aber objektorientierte Datenbanken in der Lage, die allerdings keinen vollständigen Ersatz für relationale Datenbanken darstellen, da sie in den klassischen Anwendungen für relationale Systeme, also der Verwaltung großer, einfach strukturierte Datenbestände konzeptionelle Nachteile haben. Objektrelationale Datenbanksysteme wurden entwickelt, um die Vorteile des objektorientierten und des relationalen Konzeptes zu vereinen und dabei deren jeweilige Nachteile auszuschalten.

Wir können die objektrelationalen Konzepte hier nur kurz anreißen. Mehr Informationen (auch zahlreiche Beispiele) gab es in der Vorlesung. Das entsprechende Material steht zum Nachlesen im Web zur Verfügung: [http://www.is.informatik.uni-duisburg.de/teaching/lectures/db\\_ws02/fohlen/ORDMS.pdf](http://www.is.informatik.uni-duisburg.de/teaching/lectures/db_ws02/fohlen/ORDMS.pdf).

### Strukturierte Typen

Mit strukturierten Typen (UST) lassen sich in objektrelationalen Datenbanken komplexere Objekte modellieren. Diese können auf das in rein relationalen Datenbanken fehlende Konzept der Vererbung (siehe Arbeitsblatt 2 und 3) zurückgreifen. Ein strukturierter Typ kann Attribut und Methoden an seine Subtypen vererben. Man kann strukturierte Typen auch verschachteln.

Speichern kann man Instanzen eines UST als Tupel in einer *typisierten Tabelle*, deren Attribute durch Namen und Attribute des UST bestimmt werden – oder aber als Werte eines Attributes in einer Tabelle. Typisierte Tabellen identifizieren Instanzen eines UST über den Objektidentifikator (OID). Tupel können so ganz unabhängig von ihren Werten referenziert werden, anders als mit Fremdschlüsseln in rein relationalen DBMS.

Erstellt werden strukturierte Typen mit dem Befehl `CREATE TYPE`. Man unterscheidet zwischen instantiierbaren und nicht instantiierbaren Typen. Nicht instantiierbare Typen können nicht in einer Tabelle verwendet werden, nur als Attribute einer Tabelle. Dieses Attribut darf als Werte dann nur entweder `NULL` oder aber Instanzen

*db2s2e81.pdf,  
S. 427*

von instantiierbaren Subtypen dieses Typs annehmen. Im Normalfall sind USTs aber instantiierbar.

Man löscht einen UST mit `DROP TYPE typename`, verändern kann man einen bereits definierten UST mit dem Befehl `ALTER TYPE`. Man kann nachträglich Attribute zum Typ hinzufügen oder löschen, bzw. Methoden hinzufügen oder löschen (siehe unten).

*db2s2e81.pdf,*  
*S. 512*  
*db2s2e81.pdf,*  
*S. 83*

Beispiel:

```
CREATE TYPE adresse_typ AS (  
    Strasse    VARCHAR(25),  
    HNr        SMALLINT,  
    PLZ        INT,  
    Stadt      VARCHAR(25)  
) MODE DB2SQL;  
  
CREATE TYPE telefon_typ AS (  
    Vorwahl    SMALLINT,  
    Nummer     INT,  
    Art        VARCHAR(10)  
) MODE DB2SQL;  
  
CREATE TYPE person_typ AS (  
    Name       VARCHAR(50),  
    GebDatum   DATE  
) NOT INSTANTIABLE  
MODE DB2SQL;  
  
CREATE TYPE kunde_typ UNDER person_typ AS (  
    KundenNr   INT,  
    Adresse    adresse_typ,  
    TelNr      telefon_typ,  
    HandyNr    telefon_typ  
) MODE DB2SQL;
```

## Typisierte Tabellen

Eine typisierte Tabelle wird mit einer syntaktischen Variante des `CREATE TABLE` erzeugt. Sie kann nur Instanzen des festgelegten Typs aufnehmen. Um das obige Beispiel aufzunehmen, könnte man eine Tabelle Kunden erstellen, deren Tupel Instanzen des USTs `kunde_typ` sind:

*db2s2e81.pdf,*  
*S. 332*

```
CREATE TABLE kunden OF kunde_typ  
    (REF IS Oid USER GENERATED,  
     kundennr WITH OPTIONS NOT NULL);
```

Über `REF IS` wird ein OID-Attribut festgelegt. Dieses ist für jedes Tupel einzigartig und muß beim Eingeben des Tupels vom Benutzer angegeben werden. Es kann nachträglich nicht verändert werden. Man kann für einzelne Attribute des USTs zusätzliche Optionen festlegen, hier zum Beispiel die Bedingung, dass die Kundennr. keinen NULL-Wert annehmen darf.

Auch für Tabellen kann es Hierarchien geben, d.h. eine Tabelle kann Untertabelle (Subtabelle) einer anderen sein. Der UST der Untertabelle muß ein direkter Subtyp

des UST der Obertabelle sein. Die Subtabellen erben dann das OID-Attribut und alle Attribute der Obertabelle. Vorhandene SELECT-Rechte werden übernommen.

Angenommen es existieren bereits USTypen *angestellter\_typ*, *bedienung\_typ* und *manager\_typ*, und letztere beiden Typen sind Subtypen von *angestellter\_typ* dann könnte eine beispielhafte (und sehr kleine) Hierarchie so aussehen:

```
CREATE TABLE angestellte OF angestellter_typ
  (REF IS Oid USER GENERATED);
```

```
CREATE TABLE bedienungen OF bedienung_typ
  INHERIT SELECT PRIVILEGES;
```

```
CREATE TABLE manager OF manager_typ
  INHERIT SELECT PRIVILEGES;
```

Gelöscht wird eine typisierte Tabelle über den `DROP TABLE` Befehl, aber nur, falls keine Untertabellen zu dieser existieren. Mit `DROP HIERARCHY Wurzeltabelle` läßt sich auch eine ganze Tabellenhierarchie löschen. *db2s2e81.pdf*,  
S. 512

## Methoden

Wie aus der objektorientierten Programmierung bekannt, können auch strukturierte Typen Methoden haben. Diese können beim Anlegen eines neuen USTs (`CREATE TYPE`) oder nachträglich mit `ALTER TYPE` spezifiziert werden. Bei der Spezifikation verfährt man wie vom Erstellen benutzerdefinierter Funktionen bekannt. Insbesondere ist anzugeben, ob die Methode DETERMINISTIC oder NOT DETERMINISTIC ist, und ob sie andere Tabellen ausliest.

Man erstellt die Methode entweder als extern oder intern implementierte Methode mit dem Befehl `CREATE METHOD`. Im folgenden Beispiel wurde eine interne Implementierung gewählt. Um eine Methode wieder zu löschen benutzt man `DROP METHOD methode FOR typ`. *db2s2e81.pdf*,  
S. 288  
*db2s2e81.pdf*,  
S. 512

```
CREATE TABLE geliehene_dvds (
  kunde      REF(kunde_typ) SCOPE kunden,
  titel      VARCHAR(50),
  rueckgabe  DATE
)
```

```
ALTER TYPE kunde_typ
  ADD METHOD ist_ueberfaellig ()
  RETURNS CHAR(1)
  LANGUAGE SQL
  READS SQL DATA
  NO EXTERNAL ACTION
  NOT DETERMINISTIC
```

```
CREATE METHOD ist_ueberfaellig()
  FOR kunde_typ
  BEGIN ATOMIC
    IF EXISTS (SELECT titel
              FROM geliehene_dvds
              WHERE rueckgabe < current_date
```

```
                AND SELF..kundenr =
                  Deref(geliehene_dvds.kunde)..kundenr)
THEN RETURN 'y';
ELSE RETURN 'n';
END IF;
END
```

## Anwendung der Konzepte

### Einfügen von Tupeln in typisierte Tabellen

```
INSERT INTO kunden (Oid, name, gebdatum,
                   kundenr, adresse,
                   telnr, handynr)
VALUES (
  kunde_typ('a'),
  'Musterfrau, Hanna',
  '1981-11-02',
  123456789,
  adresse_typ()
    ..strasse('Musterweg')
    ..hnr(2)
    ..plz(12345)
    ..stadt('Musterstadt'),
  telefon_typ()
    ..vorwahl(555)
    ..nummer(1234567),
  telefon_typ()
    ..vorwahl(777)
    ..nummer(7654321)),
(kunde_typ('b'),
  'Mustermann, Peter',
  '1969-10-17',
  987654321,
  adresse_typ()
    ..strasse('Musterstrasse')
    ..hnr(18)
    ..plz(12543)
    ..stadt('Musterdorf'),
  telefon_typ()
    ..vorwahl(555)
    ..nummer(3456789),
  telefon_typ()
    ..vorwahl(777)
    ..nummer(9876543));

INSERT INTO geliehene_dvds
VALUES (kunde_typ('a'),'Terminator','2004-01-01');

INSERT INTO geliehene_dvds
VALUES (kunde_typ('b'),'Titanic','2004-02-01');
```

## Anfragen mit strukturierten Datentypen

```
SELECT name, CHAR(adresse..plz)
       CONCAT ' '
       CONCAT adresse..stadt AS stadt
FROM kunden
WHERE adresse..stadt='Musterstadt';
```

NAME	STADT
Musterfrau, Hanna	12345 Musterstadt

```
SELECT name, kundennr
FROM kunden
WHERE OID->ist_ueberfaellig()='y';
```

NAME	KUNDENNR
Musterfrau, Hanna	123456789

## Aufgaben

- (a) Erstellt die Typen und Tabellen aus den Beispielen des Arbeitsblatt in Eurer Datenbank. Die Tabellen sollen Kunden- und Angestelltendaten eines DVD-Verleihs fassen. Ergänzt sie, wenn nötig.
- (b) Benutzt die Modellierung der Miniwelt 'Filme und Serien', um daran die objektrelationalen Konzepte zu testen.
  - (i) Erstellt eine Typhierarchie für Personen, SchauspielerInnen, Regisseure, Komponisten, Autoren.
  - (ii) Macht das gleiche für Produktionen, Filme und Serien.
  - (iii) Legt Tabellen zu den unter (a) und (b) erstellten USTs an, und füllt sie mit einigen Daten.
  - (iv) In der rein relationalen Beispieldatenbank IMDB23 gibt es eine Relation *spielt\_in*. Jedes Tupel beschreibt eine Rolle, die ein Schauspieler (eine Schauspielerin) in einer Produktion gespielt hat. Dabei sind Produktion und Jahr Fremdschlüssel auf die Tabellen Serie bzw. Film. Name ist ein Fremdschlüssel auf die Tabelle Person.

Wie könnte diese Tabelle aussehen, wenn man objektrelationale Konzepte benutzt? Erstellt sie in Eurer Datenbank.
  - (v) Formuliert einige Anfragen, um die objektrelationalen Konzepte zu testen.
- (c) Der DVD-Verleih hat einen Bestand an DVDs. Diese sollen in einer Tabelle festgehalten werden. Eine DVD hat eine eindeutige ID, einen Titel, eine Laufzeit und einen Regionalcode; außerdem ist jeder DVD ein Film bzw. eine Serie aus der Filmdatenbank zugeordnet. Es kann mehrere verschiedene DVDs zu einem Film geben (extended versions, directors cuts, etc.), ausserdem hat der DVD-Verleih von populären Titeln mehrere DVDs auf Lager.

Erstellt eine Tabelle *DVDs*, die Referenzen auf die zuvor erstellte Tabelle für Filme enthält.