UNIVERSITÄT ULM

Fakultät für Ingenieurswissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

6. Übung zur Vorlesung Datenbanksysteme

WS 08/09

Musterlösung

Aufgabe 6-1: Update- und Schema-Operationen, Indexe

a)
```
CREATE TABLE ueberblick(
    titel    CHAR(20)          NOT NULL,
    angnr    INTEGER           NOT NULL,
    datum    DATE,
    ort      CHAR(12));
```

```
DB20000I  The SQL command completed successfully.
```

LIST TABLES FOR USER;

```
Table/View                     Schema       Type  Creation time
------------------------------ ---------- -----  --------------------------
ANGEBOT                        MP10         T    2007-10-19-16.56.58.891001
……..
UEBERBLICK                     MP10         T    2007-10-20-13.42.34.058002
VORAUSS                        MP10         T    2007-10-19-16.56.59.262000

21 record(s) selected.
```

DESCRIBE TABLE ueberblick;

```
Column                         Type         Type
name                           schema       name                 Length   Scale  Nulls
------------------------------ --------- ------------------      -------- -----  ------
TITEL                          SYSIBM    CHARACTER            20       0      No
ANGNR                          SYSIBM    INTEGER              4        0      No
DATUM                          SYSIBM    DATE                 4        0      Yes
ORT                            SYSIBM    CHARACTER            12       0      Yes

 4 record(s) selected.
```

b)
```
INSERT INTO ueberblick
    SELECT k.titel, a.angnr, a.datum, a.ort
    FROM kurs AS k
            JOIN angebot AS a ON k.kursnr = a.kursnr;
```

```
DB20000I  The SQL command completed successfully.
```

c) **INSERT INTO ueberblick VALUES**
('Kursueberblick', 1, '02.01.2008', 'Berlin');

```
DB20000I  The SQL command completed successfully.
```

d) **CREATE UNIQUE INDEX titelindex**
ON ueberblick(titel, angnr);

```
DB20000I  The SQL command completed successfully.
```

e) **CREATE INDEX ortindex**
ON ueberblick(ort);

```
DB20000I  The SQL command completed successfully.
```

f) **ALTER TABLE ueberblick**
ADD zeit time;

```
DB20000I  The SQL command completed successfully.
```

DESCRIBE TABLE ueberblick;

```
Column      Type        Type
name        schema      name            Length   Scale    Nulls
----------- ----------- -------------- -------- -------- -----
TITEL       SYSIBM      CHARACTER            20        0 No
ANGNR       SYSIBM      INTEGER               4        0 No
DATUM       SYSIBM      DATE                  4        0 Yes
ORT         SYSIBM      CHARACTER            12        0 Yes
ZEIT        SYSIBM      TIME                  3        0 Yes

  5 record(s) selected.
```

g) **UPDATE ueberblick**
SET zeit='12.00'
WHERE titel='Kursueberblick';

```
DB20000I  The SQL command completed successfully.
```

UPDATE ueberblick
SET zeit='08.00'
WHERE titel<>'Kursueberblick';

```
DB20000I  The SQL command completed successfully.
```

h) **DELETE FROM ueberblick**
WHERE titel LIKE 'Grundlagen%';

```
DB20000I  The SQL command completed successfully.
```

i) **SELECT titel, angnr, CHAR(datum, EUR) AS DATUM, ort, CHAR(zeit, EUR) AS Zeit FROM ueberblick;**

```
TITEL                ANGNR       DATUM      ORT          ZEIT
-------------------- ----------- ---------- ------------ --------
C-Programmierung               1 28.05.2007 Ulm          08.00.00
C-Programmierung               2 01.07.2007 Essen        08.00.00
Datenbanken                    1 27.03.2007 Stuttgart    08.00.00
Datenbanken                    2 23.04.2007 Hamburg      08.00.00
Datenbanken                    3 29.05.2007 Muenchen     08.00.00
Kursueberblick                 1 02.01.2008 Berlin       12.00.00

  6 record(s) selected.
```

j) **DROP TABLE ueberblick;**

```
DB20000I  The SQL command completed successfully.
```

LIST TABLES FOR USER;

```
   Table/View                     Schema   Type  Creation time
------------------------------- -------------- ----- --------------------------
ANGEBOT                         MP10       T     2007-10-19-16.56.58.891001
…..
VORAUSS                         MP10       T     2007-10-19-16.56.59.262000

 20 record(s) selected.
```

Aufgabe 6-2: Sichten

a) **CREATE VIEW kurse AS**
SELECT k.titel, a.angnr, a.datum, a.ort, l.name
FROM kurs AS k
JOIN angebot AS a ON k.kursnr = a.kursnr
JOIN fuehrt_durch AS f ON (a.kursnr, a.angnr) = (f.kursnr, f.angnr)
JOIN kursleiter AS l ON f.persnr = l.persnr;

```
DB20000I  The SQL command completed successfully.
```

SELECT titel, angnr, CHAR(datum, EUR) AS DATUM, ort, name FROM kurse;

```
  TITEL                ANGNR       DATUM      ORT          NAME
-------------------- ----------- ---------- ------------ ------------
C-Programmierung               1 28.05.2007 Ulm          Meier, I.
C-Programmierung               2 01.07.2007 Essen        Meier, I.
Grundlagen II                  2 15.02.2007 Hamburg      Schulze, H.
Datenbanken                    3 29.05.2007 Muenchen     Schulze, H.
Datenbanken                    2 23.04.2007 Hamburg      Schulze, H.
Datenbanken                    1 27.03.2007 Stuttgart    Schulze, H.
Grundlagen I                   1 13.10.2007 Muenchen     Huber, L.
Grundlagen I                   2 24.11.2007 Bremen       Huber, L.
Grundlagen II                  1 01.12.2007 Muenchen     Mueller, K.

  9 record(s) selected.
```

b) **GRANT SELECT ON kurse TO PUBLIC;**

```
DB20000I  The SQL command completed successfully.
```

Ohne CHECK-OPTION:

c) **CREATE VIEW leiter AS**
SELECT persnr, name
FROM kursleiter
WHERE gehalt > 4000;

```
DB20000I  The SQL command completed successfully.
```

d) **INSERT INTO leiter VALUES**
(10101, 'Adam, J');

```
DB20000I  The SQL command completed successfully.
```

e) **DELETE FROM leiter WHERE persnr=43325;**

```
SQL0100W  No row was found for FETCH, UPDATE or DELETE; or the result of
a query is an empty table.  SQLSTATE=02000
```

f) **SELECT * FROM leiter;**

```
PERSNR      NAME
----------- ------------
      27183 Meier, I.
      38197 Huber, L.

  2 record(s) selected.
```

SELECT * FROM kursleiter;

```
PERSNR      NAME         GEHALT
----------- ------------ ------------------------
      27183 Meier, I.              +4.30050E+003
      29594 Schulze, H.            +3.89019E+003
      38197 Huber, L.              +4.20010E+003
      43325 Mueller, K.            +3.40080E+003
      10101 Adam, J                             -

  5 record(s) selected.
```

g) **DROP VIEW leiter;**

```
DB20000I  The SQL command completed successfully.
```

Mit CHECK-OPTION:

c) **CREATE VIEW leitercheck AS**
SELECT persnr, name
FROM kursleiter
WHERE gehalt > 4000
WITH CHECK OPTION;

```
DB20000I  The SQL command completed successfully.
```

d) **INSERT INTO leitercheck VALUES**
(10101, 'Adam, J');

```
DB21034E  The command was processed as an SQL statement because it was
not a valid Command Line Processor command.  During SQL processing it
returned:
SQL0161N  The resulting row of the INSERT or UPDATE does not conform to
the view definition.  SQLSTATE=44000
```

e) **DELETE FROM leitercheck**
WHERE persnr=43325;

```
SQL0100W  No row was found for FETCH, UPDATE or DELETE; or the result of
a query is an empty table.  SQLSTATE=02000
```

f) **SELECT * from leitercheck;**

```
PERSNR      NAME
----------- ------------
      27183 Meier, I.
      38197 Huber, L.

2 record(s) selected.
```

SELECT * from kursleiter;

```
PERSNR      NAME         GEHALT
----------- ------------ ------------------------
      27183 Meier, I.              +4.30050E+003
      29594 Schulze, H.            +3.89019E+003
      38197 Huber, L.              +4.20010E+003
      43325 Mueller, K.            +3.40080E+003

4 record(s) selected.
```

g) **DROP VIEW leitercheck;**

```
DB20000I  The SQL command completed successfully.
```

Aufgabe 6-3: Referentielle Integrität, Trigger, Anfragen

a)
```
CREATE TABLE abteilung(
  AbtNr INTEGER NOT NULL PRIMARY KEY,
  AbtName VARCHAR(12) NOT NULL);

CREATE TABLE mitarbeiter(
  PersNr INTEGER NOT NULL PRIMARY KEY,
  Name      VARCHAR (15) NOT NULL,
  Gehalt REAL,
  AbtNr INTEGER NOT NULL REFERENCES abteilung,
  VorgesPersNr INTEGER REFERENCES mitarbeiter);

CREATE TABLE projekt(
  ProjNr INTEGER NOT NULL PRIMARY KEY,
  ProjName VARCHAR(12) NOT NULL);

CREATE TABLE arbeitet_in(
  PersNr INTEGER NOT NULL REFERENCES mitarbeiter,
  ProjNr INTEGER NOT NULL REFERENCES projekt,
  PRIMARY KEY (PersNr, ProjNr));
```

Die Relationen werden mit den folgenden Daten gefüllt (Achtung: Reihenfolge des Einfügens wegen gegenseitiger Referenzierung beachten):

Abteilung:

ABTNR	ABTNAME
2	Fertigung
3	Montage
4	Verkauf

Mitarbeiter:

PERSNR	NAME	GEHALT	ABTNR	VORGESPERSNR
13	Hugo Boss	+4.02050E+003	2	-
9	Eva Petersen	+5.00000E+003	2	13
10	Peter Meier	+5.00000E+003	2	9
11	Petra Schulze	+4.00050E+003	2	13
15	Jochen Adam	+4.82050E+003	3	13
16	Klaus Lehmann	+3.00000E+003	3	13

Projekt:

PROJNR	PROJNAME
7	Proj 1
8	Proj 2
9	Proj 3

arbeitet_in:

PERSNR	PROJNR
10	7
10	8
11	8
13	8

b) **CREATE TABLE ehemalige_MA(**
PersNr INTEGER NOT NULL,
Name VARCHAR (15) NOT NULL,
Gehalt REAL,
AbtNr INTEGER NOT NULL,
VorgesPersNr INTEGER);

oder

CREATE TABLE ehemalige_MA LIKE mitarbeiter;

Erste Möglichkeit fuer den Trigger:

```
CREATE TRIGGER loesche_MA AFTER DELETE
ON mitarbeiter REFERENCING OLD_TABLE AS gel
FOR EACH STATEMENT MODE DB2SQL
  INSERT INTO ehemalige_MA
  SELECT * FROM gel;
```

Ausgangszustand wiederherstellen:

```
INSERT INTO mitarbeiter
SELECT * FROM ehemalige_ma;
DELETE FROM ehemalige_ma;

DROP TRIGGER loesche_MA;
```

Zweite Möglichkeit fuer den Trigger:

```
CREATE TRIGGER loesche_MA AFTER DELETE
ON mitarbeiter REFERENCING OLD AS gel
FOR EACH ROW MODE DB2SQL
      INSERT INTO ehemalige_ma VALUES
      (gel.PersNr, gel.Name, gel.Gehalt, gel.AbtNr, gel.VorgesPersNr);

DELETE FROM Mitarbeiter WHERE PersNr > 13;
```

Mitarbeiter:

PERSNR	NAME	GEHALT	ABTNR	VORGESPERSNR
13	Hugo Boss	+4.02050E+003	2	-
9	Eva Petersen	+5.00000E+003	2	13
10	Peter Meier	+5.00000E+003	2	9
11	Petra Schulze	+4.00050E+003	2	13

ehemalige_MA:

PERSNR	NAME	GEHALT	ABTNR	VORGESPERSNR
15	Jochen Adam	+4.82050E+003	3	13
16	Klaus Lehmann	+3.00000E+003	3	13

c) **SELECT COUNT(*) AS #Mitarbeiter**
FROM abteilung AS a
JOIN mitarbeiter AS m ON a.AbtNr = m.AbtNr
WHERE a.AbtName='Fertigung';

```
#MITARBEITER
------------
           4

  1 record(s) selected.
```

d) **SELECT m.Name, a.AbtName**
FROM mitarbeiter AS m
RIGHT OUTER JOIN abteilung AS a ON m.AbtNr = a.AbtNr;

```
NAME            ABTNAME
--------------- ------------
Hugo Boss       Fertigung
Eva Petersen    Fertigung
Peter Meier     Fertigung
Petra Schulze   Fertigung
Jochen Adam     Montage
-               Verkauf

  6 record(s) selected.
```

e1) **SELECT p.ProjName**
FROM projekt AS p
WHERE NOT EXISTS
(SELECT *
FROM arbeitet_in AS a
WHERE a.ProjNr = p.ProjNr);

e2) **SELECT ProjName**
FROM projekt
WHERE ProjNr NOT IN
(SELECT ProjNr
FROM arbeitet_in);

e3) **SELECT p.ProjName**
FROM projekt AS p
LEFT OUTER JOIN arbeitet_in AS a ON p.ProjNr = a.ProjNr
WHERE a.ProjNr IS NULL;

```
PROJNAME
------------
Proj 3

  1 record(s) selected.
```

f) **SELECT v.PersNr, v.Name, v.Gehalt, COUNT(*) AS #Untergebene**
FROM mitarbeiter AS v
JOIN mitarbeiter AS m ON v.PersNr = m.VorgesPersNr
GROUP BY v.PersNr, v.Name, v.Gehalt
HAVING COUNT(*) >= 3;

```
PERSNR      NAME            GEHALT                    #UNTERGEBENE
----------- --------------- ------------------------- ------------
         13 Hugo Boss                  +4.02050E+003             3

  1 record(s) selected.
```

g)
```
WITH recrel(Name, VorgesPersNr) AS
(
  SELECT v.Name, v.VorgesPersNr
  FROM mitarbeiter AS v, mitarbeiter AS m
  WHERE v.PersNr = m.VorgesPersNr AND m.Name = 'Peter Meier'
  UNION ALL
  SELECT v.Name, v.VorgesPersNr
  FROM mitarbeiter AS v, recrel AS rec
  WHERE v.PersNr = rec.VorgesPersNr
)
SELECT Name FROM recrel;
```

```
NAME
---------------
SQL0347W  The recursive common table expression "MP10.VORGESETZTE" may
contain an infinite loop.  SQLSTATE=01605

Eva Petersen
Hugo Boss

  2 record(s) selected with 1 warning messages printed.
```