[4131658] Jan Germann
[4054962] Christian Ratz
Übungsgruppe 1

SQL Praktikum
7. Januar 2013

Three degrees of Heath Ledger 4

4.1 Aufgabe

a. SELECT-Anfrage auf den *database catalog* um alle IMDB-Schema Tabellen auszugeben:

```
SELECT TABNAME as Tabellen
FROM SYSCAT.TABLES
WHERE TABSCHEMA = 'IMDB'
```

b. SELECT-Anfrage auf den *database catalog* um die Struktur der IMDB.MOVIES-Tabelle auszugeben:

```
SELECT COLNAME, TYPENAME, LENGTH
FROM SYSCAT.COLUMNS
WHERE TABSCHEMA = 'IMDB'
AND TABNAME = 'MOVIES' ORDER BY COLNO
```

c. **IMDB.MOVIES**-Struktur manuell als **tv_movies_tmp** kopiert:

```
CREATE TABLE tv_movies_tmp (
TITLE_ID varchar(400),
TITLE_TYPE varchar(100),
TITLE_TITLE varchar(400),
TITLE_YEAR integer,
TITLE_NUMERAL integer,
TITLE_SUSPENDED decimal,
TITLE_ATTRIBUTES varchar(400),
EPISODE_SERIES varchar(400),
EPISODE_TITLE varchar(400),
EPISODE_TITLE_NUMERAL integer,
EPISODE_SEASON integer,
EPISODE_EPISODE integer,
EPISODE_DATE date,
YEAR_RANGE varchar(100));
```

4.2 Aufgabe

Struktur einer Tabelle mit

```
CREATE TABLE LIKE
```

kopieren: (in diesem Fall die Struktur von **IMDB.ACTORS**)

```
CREATE TABLE tv_actors_tmp LIKE IMDB.ACTORS
```

4.3 Aufgabe

a. Tabellen die überhaupt keine NULL-Werte enthalten:

IMDB.MOVIES:
TITLE_ID
TITLE_TYPE
TITLE_SUSPENDED

IMDB.ACTORS:
NAME_NAME
TITLE_ID
NAME_ID

b. NOT NULL CONSTRAINT zu **tv_movies_tmp** und zu **tv_actors_tmp** hinzufügen:
tv_movies_tmp

```
1  ALTER TABLE tv_movies_tmp ALTER COLUMN title_id SET NOT NULL
2  ALTER TABLE tv_movies_tmp ALTER COLUMN title_type SET NOT NULL
3  ALTER TABLE tv_movies_tmp ALTER COLUMN title_suspended SET NOT NULL
```

tv_actors_tmp

```
1  ALTER TABLE tv_actors_tmp ALTER COLUMN name_id SET NOT NULL
2  ALTER TABLE tv_actors_tmp ALTER COLUMN name_name SET NOT NULL
3  ALTER TABLE tv_actors_tmp ALTER COLUMN title_id SET NOT NULL
```

4.4 Aufgabe

Struktur von **tv_movies_tmp** und **tv_actors_tmp** kopieren und zwei neue Tabellen **tv_movies** und **tv_actors** mit derselben Struktur erstellen:

```
1  CREATE TABLE tv_movies LIKE tv_movies_tmp
2  CREATE TABLE tv_actors LIKE tv_actors_tmp
```

4.5 Aufgabe

Passende PRIMARY KEYS und FOREIGN KEYS einfügen:

```
1  ALTER TABLE TV_MOVIES ADD PRIMARY KEY (TITLE_ID, TITLE_TYPE);
2  ALTER TABLE TV_ACTORS ADD PRIMARY KEY (NAME_ID, NAME_NAME);
3  ALTER TABLE TV_ACTORS ADD UNIQUE (TITLE_ID);
4  ALTER TABLE TV_MOVIES ADD UNIQUE (TITLE_ID);
5  ALTER TABLE TV_ACTORS ADD FOREIGN KEY (TITLE_ID) REFERENCES TV_MOVIES (TITLE_ID);
6  ALTER TABLE TV_MOVIES ADD FOREIGN KEY (TITLE_ID) REFERENCES TV_ACTORS (TITLE_ID);
```

4.6 Aufgabe

tv_movies und **tv_actors** hit entsprechenden Daten füllen: Nur *TV movies* aus dem Jahr *2008* und alle ihre *actors* und *actresses* eintragen:

```
INSERT INTO DBLAB01.TV_MOVIES (TITLE_ID, TITLE_TYPE, TITLE_TITLE, TITLE_YEAR,
TITLE_NUMERAL, TITLE_NUMERAL, TITLE_SUSPENDED, TITLE_ATTRIBUTES,
EPISODE_SERIES, EPISODE_TITLE, EPISODE_TITLE_NUMERAL, EPISODE_SEASON,
EPISODE_EPISODE, EPISODE_DATE, YEAR_RANGE)
 (SELECT DISTINCT TITLE_ID, TITLE_TYPE, TITLE_TITLE, TITLE_YEAR,
 TITLE_NUMERAL, TITLE_NUMERAL, TITLE_SUSPENDED, TITLE_ATTRIBUTES,
 EPISODE_SERIES, EPISODE_TITLE, EPISODE_TITLE_NUMERAL, EPISODE_SEASON,
 EPISODE_EPISODE, EPISODE_DATE, YEAR_RANGE
 FROM IMDB.MOVIES AS m, IMDB.ACTORS AS a
 WHERE m.TITLE_ID = a.TITLE_ID
 AND TITLE_TYPE = 'TV movie'
 AND TITLE_YEAR = '2008');

INSERT INTO DBLAB01.TV_MOVIES (TITLE_ID, TITLE_TYPE, TITLE_TITLE, TITLE_YEAR,
TITLE_NUMERAL, TITLE_NUMERAL, TITLE_SUSPENDED, TITLE_ATTRIBUTES,
EPISODE_SERIES, EPISODE_TITLE, EPISODE_TITLE_NUMERAL, EPISODE_SEASON,
EPISODE_EPISODE, EPISODE_DATE, YEAR_RANGE)
 (SELECT DISTINCT TITLE_ID, TITLE_TYPE, TITLE_TITLE, TITLE_YEAR,
 TITLE_NUMERAL, TITLE_NUMERAL, TITLE_SUSPENDED, TITLE_ATTRIBUTES,
 EPISODE_SERIES, EPISODE_TITLE, EPISODE_TITLE_NUMERAL, EPISODE_SEASON,
 EPISODE_EPISODE, EPISODE_DATE, YEAR_RANGE
 FROM IMDB.MOVIES AS m, IMDB.ACTRESSES AS a
 WHERE m.TITLE_ID = a.TITLE_ID
 AND TITLE_TYPE = 'TV movie'
 AND TITLE_YEAR = '2008');

 INSERT INTO DBLAB01.TV_ACTORS (NAME_ID, NAME_NAME, NAME_NUMERAL, TITLE_ID,
 CHARACTERS, CREDIT_ORDER_NUMBER, CREDIT_ATTRIBUTES)
 (SELECT DISTINCT NAME_ID, NAME_NAME, NAME_NUMERAL, TITLE_ID,
 CHARACTERS, CREDIT_ORDER_NUMBER, CREDIT_ATTRIBUTES
 FROM IMDB.ACTORS AS a, IMDB.MOVIES AS m
 WHERE a.TITLE_ID = m.TITLE_ID
 AND TITLE_TYPE = 'TV movie'
 AND TITLE_YEAR = '2008');

  INSERT INTO DBLAB01.TV_ACTORS (NAME_ID, NAME_NAME, NAME_NUMERAL, TITLE_ID,
  CHARACTERS, CREDIT_ORDER_NUMBER, CREDIT_ATTRIBUTES)
 (SELECT DISTINCT NAME_ID, NAME_NAME, NAME_NUMERAL, TITLE_ID,
 CHARACTERS, CREDIT_ORDER_NUMBER, CREDIT_ATTRIBUTES
 FROM IMDB.ACTRESSES AS a, IMDB.MOVIES AS m
 WHERE a.TITLE_ID = m.TITLE_ID
 AND TITLE_TYPE = 'TV movie'
 AND TITLE_YEAR = '2008');
```

4.7 Aufgabe

materialized view namens *actor_co_occurrences* erstellen, die *tv_movies* und *tv_actors* basiert. Folgende Signatur:

```
1  actor_co_occurrences(actor1, actor2, movie)
```

4.8 Aufgabe

Für *actor1*, *actor2* und *movie* aus der *actor_co_occurrences* einen eigenen Index anlegen.

4.9 Aufgabe

Alle Schauspieler, die eine *Heath Ledger Zahl* von maximal 3 haben. Gib ihre Namen und ihre *Heath Ledger Zahl* aus. Das benutzte SELECT Statement soll **nicht** rekursiv sein!

4.10 Aufgabe

SELECT-Anfrage aus Aufgabe 9 (rekursive Version)