Institut for Matematik og Datalogi
Syddansk Universitet, Odense

11. oktober 2017
CW

Exercise Solutions DM534

Uge 43

1. ☑ ('Foo Fighters', 1994)
   ☒ (1991, 'Incubus')
   ☒ ('Massive Attack')
   ☒ ('Disturbed', '1996')

2. **Band**

name	**formed_in**
'Foo Fighters'	1994

3. ☑ {mid}
   ☑ {title}
   ☒ {director}
   ☑ {title, director}
   ☑ {director, production_year}

4. Multiple solutions possible.

   In a first possible solution, the two relations Game and DeveloperStudio have multi-attribute primary keys. As a consequence, the foreign keys of DeveloperStudioDevelopsGame requires these attributes in their foreign keys to the two tables.

   - Game(name: CHAR(20), release_date: CHAR(20) , budget: FLOAT)
   - DeveloperStudio(name: CHAR(20), address: CHAR(20) , number_employees: INTEGER)
   - DeveloperStudioDevelopsGame(name_studio: CHAR(20), address: CHAR(20) , name_game: CHAR(20), release_date: CHAR(20))

   A second solution is to introduce the dedicated primary key attributes *gid* ("game id") and *sid* ("studio id") into the two Game and DeveloperStudio relations. Then the foreign keys of DeveloperStudioDevelopsGame point to these two attributes instead.

- Game(gid: INTEGER , name: CHAR(20), release_date: CHAR(20), budget: FLOAT)
- DeveloperStudio(sid: INTEGER , name: CHAR(20), address: CHAR(20), number_employees: INTEGER)
- DeveloperStudioDevelopsGame(sid: INTEGER , gid: INTEGER)

The second solution is considered cleaner in practice, but it requires additional attributes which increases the required disk storage space of the relations.

5. FestivalJoinFestivalHasConcert(fid: INTEGER, name: CHAR(20), start_date: CHAR(20), end_date: CHAR(20), festival: INTEGER, concert: INTEGER)

6.
   - 6
   - 6
   - 6
   - 5

7.
```
CREATE TABLE Concert (cid INTEGER PRIMARY KEY, band CHAR(20),
        date CHAR(20), location CHAR(20),
        total_number_seats INTEGER, ticket_price FLOAT);
```

8.
```
CREATE TABLE FestivalHasConcert (festival INTEGER,
        concert INTEGER,
        FOREIGN KEY (festival) REFERENCES Festival (fid),
        FOREIGN KEY (concert) REFERENCES Concert (cid));
```

9.
```
DELETE FROM Movies WHERE
        production_year < 1994 OR production_year > 1994;
```

   or

```
DELETE FROM Movies WHERE production_year != 1994;
```

10.
```
SELECT * FROM Movies WHERE
        production_year = 1994 AND
        director = 'Quentin Tarantino';
```

11. Formulate the query of task 10 with nested relational operators.

$$\sigma_{production\_year=1994 \wedge director='QuentinTarantino'}(Movies)$$

12. Multiple solutions possible. One of them:

```
DELETE FROM Movies WHERE
        mid >= 2 AND mid <= 4;
INSERT INTO Movies VALUES (6, 'The Lord of the Rings',
        'Peter Jackson', 2001, 93000000);
```

13.
```
SELECT * FROM Movies WHERE production_year < 1990
    AND budget >= 30000000;
```

14.
```
SELECT * FROM Movies WHERE production_year < 1990
EXCEPT
SELECT * FROM Movies WHERE budget < 30000000;
```

15.
```
SELECT * FROM Movies WHERE
        (production_year < 1990 AND budget >= 30000000)
        OR production_year > 2010;
```

16.
```
SELECT M1.title, M2.title FROM Movies M1, Movies M2 WHERE
        M1.director = M2.director;
```

17. Formulate the query of task 16 with nested relational operators.

$$\pi_{M1.title,M2.title}(\sigma_{M1.director=M2.director}(Movies\,M1 \times Movies\,M2))$$

18.
```
SELECT M1.title, M2.title FROM Movies M1, Movies M2 WHERE
        M1.budget > M2.budget;
```

19.
```
SELECT * FROM Festival, FestivalHasConcert WHERE
        fid = festival;
```

20.
```
SELECT * FROM Festival, FestivalHasConcert, Concert WHERE
        fid = festival AND cid = concert;
```

21. Which of the following statements are true (multiple possible)?

- ☑ The result of applying a relational algebra operator to a relation instance is another relation instance.
- ☒ A relation in a data model is the equivalent concept of a relationship in an ER-diagram.
- ☒ Entities of the ER-diagram can not be described by relations in the data model.
- ☒ A relation instance needs to contain at least one tuple.
- ☒ Integrity constraints are specified when querying the database.

☑ Primary keys and foreign keys are types of integrity constraints.

☒ A foreign key can reference arbitrary attributes of other tables.

☑ A primary key can be used to look up tuples in a table.

☒ The relational selection operator always returns a relation instance with fewer tuples.

☑ The relational projection operator may return a relation instance with fewer tuples. 

☒ The SQL UNION operator can be applied to two relation instances, if they have the same number of attributes.