DM534 — Øvelser Uge 47

Introduktion til Datalogi, Efterår 2021

Jonas Vistrup

---

1 I

1.1

Given the following relation schema:

Band(name: CHAR(20), formed in: INTEGER)

Which of the following are valid tuples of the Band relation?

- ('Foo Fighters', 1994). **SVAR:** Valid.
- (1991, 'Incubus'). **SVAR:** Invalid, 1991 is not a string. 'Incubus' is not an int.
- ('Massive Attack'). **SVAR:** Invalid, no INTEGER.
- ('Disturbed', '1996'). **SVAR:** Invalid, '1996' is not an int.

1.2

The relation schema of task 1 together with the valid tuples define a relation instance. Visualize this relation instance as a table.

SVAR: Band

name	formed in
'Foo Fighters'	1994

1.3

Given the following relation instance:

Movies

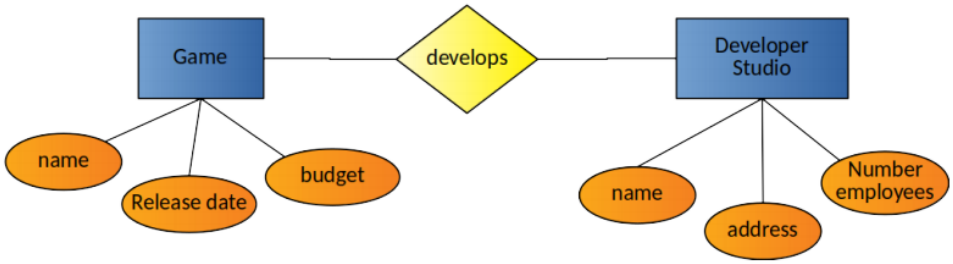
mid	title	director	production_year	budget_usd
0	'Matrix'	'The Wachowskis'	1999	63000000
1	'Raiders of the Lost Ark'	'Steven Spielberg'	1981	20000000
2	'The Shawshank Redemption'	'Frank Darabont'	1994	25000000
3	'Twilight'	'Robert Benton'	1998	20000000
4	'Dead Poets Society'	'Peter Weir'	1989	16400000
5	'Django Unchained'	'Quentin Tarantino'	2012	100000000
6	'Pulp Fiction'	'Quentin Tarantino'	1994	8500000
7	'Twilight'	'Catherine Hardwicke'	2008	37000000

Which of the following attribute sets are possible primary keys (i.e., the data instance above is consistent with choosing that set of attributes as the primary key)?

- {mid}. **SVAR:** Yes.
- {title}. **SVAR:** No, there are two Twilight.
- {director}. **SVAR:** No, there are two movies by Tarantino.
- {title, director}. **SVAR:** Yes.
- {director, production_year}. **SVAR:** Yes.

1.4

Given the following Entity-Relationship diagram:



How could this ER diagram be modeled in the relational model? Provide the relation schemas. What would change if the relationship "develops" had an attribute "start date"? Note that there is no designation of keys for the entities. It is part of the task to consider what is a good choice of keys (this is a data modeling choice, for which there is no single "right answer").

SVAR:

Let's first consider the question of keys. If we assume that developer studios would fight legally if there are name clashes among studios or among games, we could meaningfully designate "name" as the primary key in both entities. Otherwise, {name, release date} for entity Game, and {name, address} for entity Developer Studio could be reasonable suggestions.

For the corresponding schemas in the relational model: entity Game gives rise to a relation with schema (name, release date, budget), entitiy Developer Studio gives rise to a relation with schema (name, address, number employees), and relationship Develops gives rise to a relation whose scheme consists of the primary key (possibly renamed to avoid name clashes) from Game, the primary key (possibly renamed to avoid name clashes) from Developer Studio, and any attributes that it has itself. For example, it could have the schema (gid, dsid, start date). Appropiate types (string, integer, date,…) could be added to the attributes in these schemas (if the relations are later expressed in SQL, this will be needed).

1.5

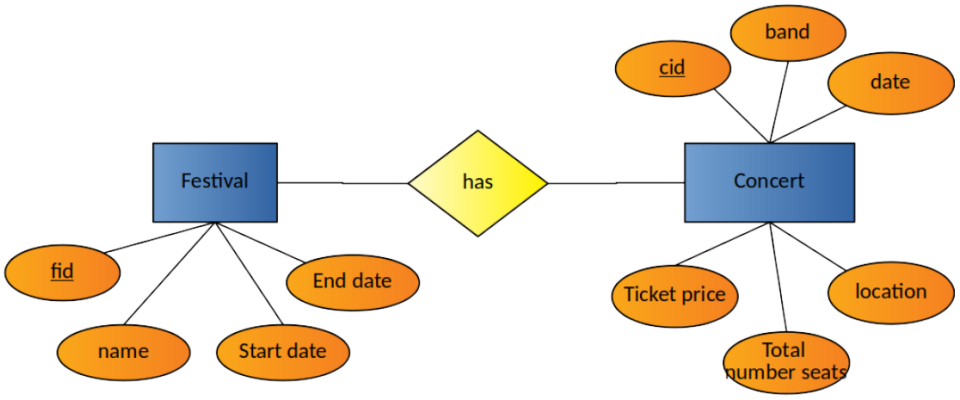
You are given the relation instance defined in task 3.

How many tuples do the relations resulting from the following relational operations contain? [Recall that in the relational model, relations are sets, so there can be no duplicates among (entire) tuples/rows.]

- $\sigma_{\text{production\_year}>1994}(\text{Movies})$. **SVAR:** 4.
- $\pi_{\text{mid,title,director}}(\text{Movies})$. **SVAR:** 8.
- $\pi_{\text{director,production\_year}}(\text{Movies})$. **SVAR:** 8.
- $\pi_{\text{director}}(\text{Movies})$. **SVAR:** 7.

1.6

You are given the following ER-diagram:



and the corresponding relations:

- Concert(cid: INTEGER , band: CHAR(20), date: CHAR(20), location: CHAR(20), total_number_seats: INTEGER, ticket_price: FLOAT)
- Festival(fid: INTEGER , name: CHAR(20), start_date: CHAR(20), end_date: CHAR(20))
- FestivalHasConcert(festival: INTEGER , concert: INTEGER)

Here, underlined attributes are primary keys of the relation and dashed underlined attributes are foreign keys.

Specify the SQL command that creates the table corresponding to the Concert relation of task.

Also specify the SQL command that creates a corresponding table for the FestivalHasConcert relation.

[Recall that FestivalHasConcert's festival and concert attributes are foreign keys referencing the fid in the festival and cid in the concert relation, respectively.]

SVAR:

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

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

1.7

Specify an SQL command that deletes from the Movies table all movies that were not produced in 1994.

[You can solve this task with the comparison operators that you saw in the lecture. Alternatively, use the not-equal-to operator != which is also defined in SQL.]

SVAR:

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

or

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

1.8

You want to retrieve the titles of all movies from the Movies table of task 3 that were produced in 1994 and directed by Quentin Tarantino. Express this query in both of these two ways:

- As an expression in relational algebra.

  SVAR: $\pi_{title}(\sigma_{\text{director='Quentin Tarantino' \& production\_year = 1994}}(Movies))$

- As an SQL command.

  SVAR: SELECT title FROM Movies where director = 'Quentin Tarantino' AND production_year = 1994.

1.9

Consider the Movies table of task 3. Provide in SQL one INSERT and one DELETE command that can to be executed to change it into the following table.

Movies

mid	title	director	production_year	budget_usd
0	'Matrix'	'The Wachowskis'	1999	63000000
1	'Raiders of the Lost Ark'	'Steven Spielberg'	1981	20000000
5	'Django Unchained'	'Quentin Tarantino'	2012	100000000
7	'Twilight'	'Catherine Hardwicke'	2008	37000000
8	'The Lord of the Rings'	'Peter Jackson'	2001	93000000

SVAR:

```
INSERT INTO Movies
    (9, 'The Lord of the Rings', 'Peter Jackson', 2001, 93000000);

DELETE FROM Movies M where M.production_year<1999
    AND M.production_year>1981;
```

1.10

Specify an SQL command without set operations (UNION or EXCEPT) that retrieves all movies from the Movie table, that have been produced before 1990 or that have a budget of at least 30 million USD. State the same query as an relational algebra expression.

SVAR:

```
SELECT * FROM Movies M where M.produced_year<1990 OR M.budget_usd>=30000000;
```

$$\sigma_{\text{produced\_year}<1990 \mid \text{budget\_usd}>=30000000}(Movies)$$

1.11

Solve task 10 with SQL set operations (UNION or EXCEPT). State the same query as an relational algebra expression.

SVAR:

```
SELECT * FROM Movies M
    where M.produced_year<1990
UNION
SELECT * FROM Movies M
    where M.budget_usd>=30000000;
```

$$\sigma_{\text{produced\_year}<1990}(Movies) \cup \sigma_{\text{budget\_usd}\geq 30000000}(Movies)$$

1.12

Specify an SQL command without set operations (UNION or EXCEPT) that retrieves all movies from the Movie table of task 3 that either

- have been produced before 1990 with a budget of at least 30 million USD
- or have been produced after 2010

SVAR:

```
SELECT * FROM Movies M
    where (M.produced_year<1990 AND M.budget_usd>=30000000)
    OR M.produced_year>2010;
```

State the same query as an relational algebra expression.

SVAR:

$$\sigma_{(\text{produced\_year}<1990 \ \& \ \text{budget\_usd}>=30000000) \mid \text{produced\_year}>2010}(Movies)$$

1.13

Specify an SQL command that retrieves all pairs of movies from the Movies table of task 3 that have been directed by the same director.

For instance, if two movies 'movie1' and 'movie2' were directed by the same director, the result relation should, amongst others, contain the following tuples:

- (movie1, movie2)
- (movie2, movie1)

SVAR:

```
SELECT * FROM Movies M1, Movies M2
  WHERE M1.director = M2.director AND M1.mid != M2.mid;
```

State the same query as an relational algebra expression.

SVAR:

$$\sigma_{\text{mid1!=mid\&director1=director}}($$
$$\rho(C(1 \to \text{ mid1}, 2 \to \text{ title1}, 3 \to \text{ director1}, 4 \to \text{ production\_year1}, 5 \to \text{ budget\_usd1})$$
$$, \text{Movies} \times \text{Movies}))$$

(These answers make the (natural) assumption that we are not interested in pairs of movies (movie1, movie2) where move1 and movie2 are the same movie).

1.14

Specify an SQL command that retrieves all pairs of movies (movie1, movie2) from the Movies table of task 3, with movie1's budget exceeding the budget of movie2.

For instance, if a movie 'movie5' has a larger budget than 'movie8', the result relation should contain, amongst others, the following tuple:

- (movie5, movie8)

SVAR:

```
SELECT * FROM Movies M1, Movies M2
  WHERE M1.budget_uds > M2.budget_usd;
```

State the same query as an relational algebra expression.

$$\sigma_{\text{budget\_usd1>budget}}($$
$$\rho(C(1 \to \text{ mid1}, 2 \to \text{ title1}, 3 \to \text{ director1}, 4 \to \text{ production\_year1}, 5 \to \text{ budget\_usd1})$$
$$, \text{Movies} \times \text{Movies}))$$

2 II

2.1

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. **SVAR:** TRUE.
- Entities of the ER-diagram can not be described by relations in the data model. **SVAR:** FALSE, we transfer our data models in the ER-model to the relational model, and in this transfer, we map entities very directly to relations.

- A relation instance needs to contain at least one tuple. **SVAR:** FALSE.
- Integrity constraints are specified when querying the database. **SVAR:** FALSE.
- Primary keys and foreign keys are types of integrity constraints. **SVAR:** TRUE.
- The relational selection operator always returns a relation instance with fewer tuples. **SVAR:** FALSE, could be equal amount if the selection criteria does not filter out any.
- The relational projection operator may return a relation instance with fewer tuples. **SVAR:** TRUE, in case of duplicate tuples arising, which in the relational model (where relations are sets of tuples) will be removed. [Note: for efficiency reasons, real world databases do not perform this removal unless explicitly specified in the SQL query.]
- The SQL UNION operator can be applied to two relation instances, if they have the same number of attributes. **SVAR:** FALSE, they also have to have the same attribute types at corresponding attribute positions.

2.2

Joins are compound operators which we did not cover in the lecture. They are useful for joining (combining) the data contained in multiple relations together. In relational algebra, the *conditional join* operator $\bowtie_C$ (also called the θ-join operator) is defined by

$$R1 \bowtie_C R2 = \sigma_C(R1 \times R2)$$

In words: A conditional join can be calculated by first computing the cross product of the two relations $R1$ and $R2$, followed by a selection using the condition C.

We now look at the following conditional join on relations from task I.6:

$$\text{Festival} \bowtie_{\text{fid=festival}} \text{FestivalHasConcert}$$

This join combines the two relations Festival and FestivalHasConcert using the attribute fid of the Festival relation, and the festival attribute of the FestivalHasConcert relation. Specifically, it combines those tuples of the two relations for which the equality condition holds.

What is the relation schema of the result relation?

SVAR: The schema is (fid, name, start date, end date, festival, concert), which is the same as the cross product of the two relations.

2.3

Specify an SQL command that calculates the conditional join given in task II.2.

SVAR:

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

2.4

Specify an SQL command that calculates the following nested relational operation involving two conditional joins:

$$\text{Festival} \bowtie_{\text{fid=festival}} \text{FestivalHasConcert} \bowtie_{\text{concert=cid}} \text{Concert}$$

Hint 1: This compound operation can be expressed using basic relational operators as

$$\sigma_{\text{fid=festival}}(\sigma_{\text{concert=cid}}(\text{Festival} \times \text{FestivalHasConcert} \times \text{Concert}))$$

Hint 2: You can start with and extend the SQL command of task II.3.

SVAR:

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