Name

CWID

Final Exam

December 10th, 2014
8:00-10:00

CS425 - Database Organization
Results

Please leave this empty!

1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 | Sum

Instructions

- Try to answer all the questions using what you have learned in class. Keep hard questions until the end.
- **When writing a query, write the query in a way that it would work over all possible database instances and not just for the given example instance!**
- The exam is closed book and closed notes!

Consider the following database schema and example instance about music albums:

song

title	length	writtenBy	writtenYear
De Charybde en Scylla	5:35	Louis Sclavis	2008
Un Vent Noir	3:40	Louis Sclavis	2008
Standing Ovation	4:26	Louis Sclavis	1995
moto [ein weltbewusstsein.]	6:00	Michael Wertmueller	2000

songInAlbum

song	album	trackNumber
De Charybde en Scylla	Lost on the Way	1
Un Vent Noir	Lost on the Way	9
Standing Ovation	Carnet De Routes	1
moto [ein weltbewusstsein.]	Werthmueller	4

album

aTitle	releaseYear	producedBy	playedBy
Lost on the Way	2008	ECM	Louis Sclavis
Carnet de Routes	1995	Label Blue	Louis Sclavis
Werthmueller	2005	GROB	Steamboat Switzerland

artist

name	nationality
Louis Sclavis	french
Steamboat Switzerland	swiss
Michael Wertmueller	swiss

label

name	revenue
Sony	50,000,000
ECM	3,000,000
Label Bleu	150,000
GROB	10,000

Hints:

- Attributes with black background form the primary key of a relation (.e.g, *title* for relation *song*)
- The attribute *writtenBy* of relation *song* is a foreign key to relation *artist*. Note that the artist writing a song is not necessarily the artist playing the song.
- The attribute *album* of relation *songInAlbum* is a foreign key to relation *album*.
- The attribute *song* of relation *songInAlbum* is a foreign key to relation *song*.

- The attribute *producedBy* of relation *album* is a foreign key to relation *label*.
- The attribute *playedBy* of relation *album* is a foreign key to relation *artist*.
- All foreign keys have been created with the **CASCADE** option.

Part 1.1 Relational Algebra (Total: 20 Points)

The queries in this part should be written using the set semantics version of relational algebra.

Question 1.1.1 (4 Points)

Write a **relational algebra** expression that returns the titles and length of all songs written by french artists (*nationality*) in 2008.

Solution

$$\pi_{title,length}(\sigma_{writtenYear=2008}(song) \bowtie_{writtenBy=name} \sigma_{nationality='french'}(artist))$$

Question 1.1.2 (4 Points)

Write a **relational algebra** expression that returns the names of labels that produced at least one album in 1993.

Solution

$$\pi_{name}(\sigma_{releaseYear=1993}(album) \bowtie_{producedBy=name} label)$$

Question 1.1.3 (5 Points)

Write a **relational algebra** expression that returns the number of albums produced by the ECM label per year.

Solution

$$releaseYear\mathcal{G}_{count(*)}(\sigma_{producedBy='ECM'}(album))$$

Question 1.1.4 (7 Points)

Write a **relational algebra** expression that returns labels that have not released (*producedBy*) any albums by swiss artists.

Solution

$$swissL \leftarrow \pi_{l.name}(\rho_a(\sigma_{nationality='swiss'}(artist)) \bowtie_{l.playedBy=c.name} \rho_c(album) \bowtie_{c.producedBy=l.name} \rho_l(label)))$$
$$q \leftarrow \pi_{name}(label) - swissL$$

Part 1.2 SQL - DDL (Total: 7 Points)

Question 1.2.1 (7 Points)

Write an **SQL statement** that creates a new relation *reviews* that records information about reviews of albums written by critics. For each review we want to store the magazine the review was published in as well as the release year and issue. Furthermore, we store the name of the critic who has written the review, the album title, the rating (one out of "positive", "negative", "neutral"), and the actual text of the article. A review can be uniquely identified by the magazine and album. We require that every review has a critic, rating, and text.

Solution

```
CREATE TABLE review (
    magazine VARCHAR(40),
    relYear INT,
    issue INT,
    critic VARCHAR(60) NOT NULL,
    album VARCHAR(2000) REFERENCES album,
    rating VARCHAR(8)  NOT NULL,
    text CLOB  NOT NULL,
    PRIMARY KEY (magazine, album),
    CHECK(rating = 'positive' OR rating = 'negative' OR rating = 'neutral')
);
```

Part 1.3 SQL - Queries (Total: 20 Points)

Question 1.3.1 (3 Points)

Write an **SQL query** that returns the total length for each album, i.e., the total length of all tracks of each album. Return two attributes: album title and the total length.

Solution

```
SELECT album, sum(length) AS totalLen
FROM song s JOIN songInAlbum a ON (s.title = a.song)
GROUP BY album;
```

Question 1.3.2 (5 Points)

Write an **SQL query** that returns the names of all artists that have only played songs (have songs included in albums they have played on) that they have written themselves.

Solution

```
SELECT name
FROM artist
WHERE name NOT IN (SELECT a.name
                   FROM artist a, album m, songInAlbum sa, song s
                   WHERE a.name = m.playedBy
                         AND m.aTitle = sa.album
                         AND sa.song = s.title
                         AND s.writtenBy != a.name)
```

Question 1.3.3 (6 Points)

Write an **SQL query** that returns artists that have worked for all labels, i.e., for which there exists for every label at least one album published by this label and played by this artist.

Solution

```
SELECT name
FROM artist a
WHERE NOT EXISTS (SELECT *
                  FROM label l
                  WHERE NOT EXISTS (SELECT *
                                    FROM album m
                                    WHERE m.producedBy = l.name
                                          AND m.playedBy = a.name));
```

Question 1.3.4 (6 Points)

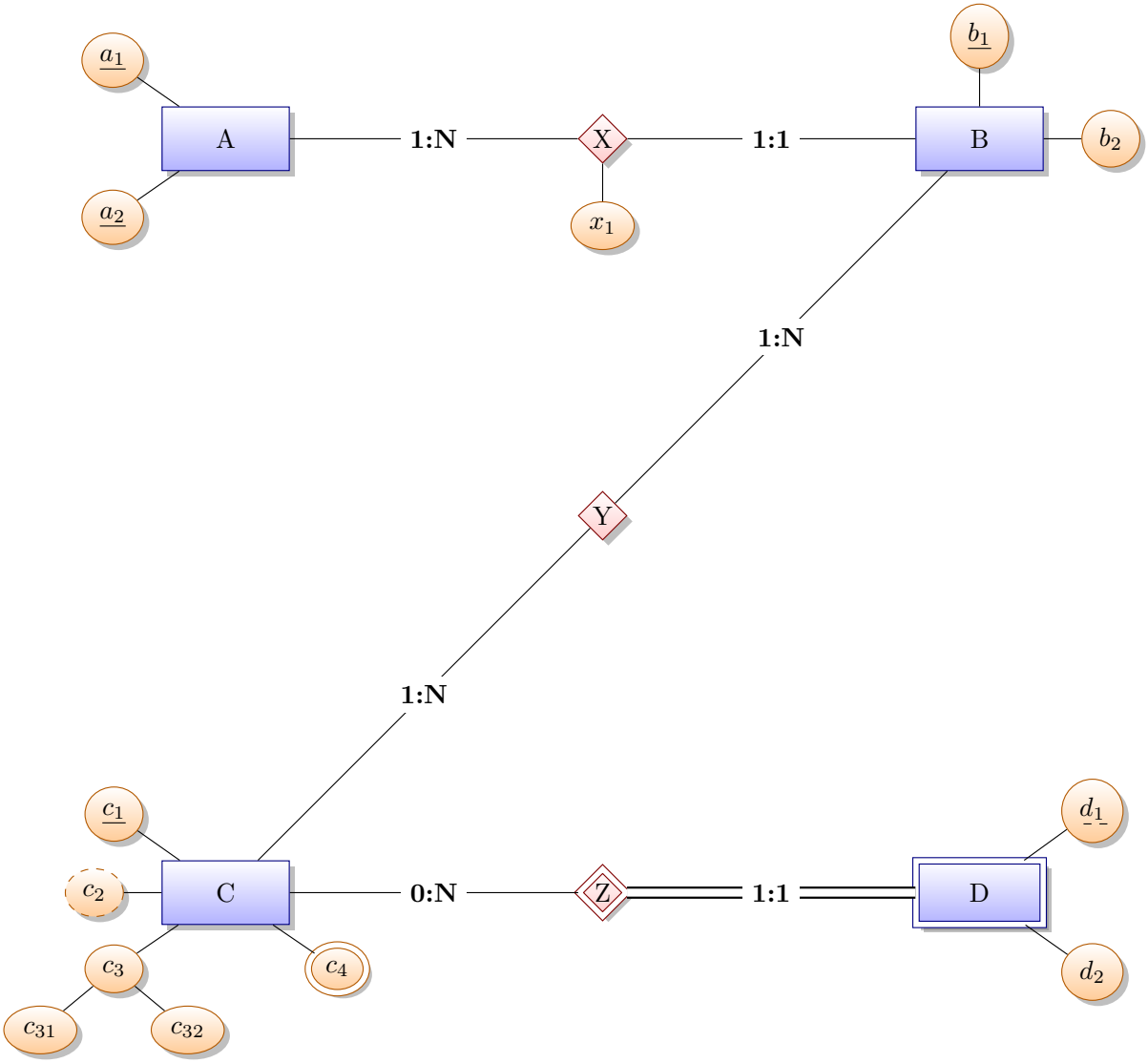
Write an **SQL query** that returns the year in which the maximum amount of music was produced. That is, the year with the maximal total length of all songs written in this year. **Solution**

```
SELECT writtenYear
FROM song
GROUP BY writtenYear
HAVING sum(length) = (SELECT max(ttl)
                      FROM (SELECT sum(length) AS ttl
                            FROM song
                            GROUP BY writtenYear) sLen);
```

Part 1.4 Translation of ER into Relational Model (Total: 22 Points)

Question 1.4.1 (22 Points)

Take the following ER-model and translate it into a relational schema using the rules presented in class. Present the relational schema using the notation from the slides. For example, a relation R with attributes a_1 and a_2 where a_2 is the primary key is written as $R(a_1, \underline{a_2})$. You do not need to specify foreign key constraints. You do not have to show intermediate results.



Solution

1st Step: strong entities

$A(\underline{a_1}, \underline{a_2})$
$B(\underline{b_1}, b_2)$
$C(\underline{c_1}, c_{31}, c_{32})$

2nd Step: weak entities

$A(\underline{a_1}, \underline{a_2})$
$B(\underline{b_1}, b_2)$
$C(\underline{c_1}, c_{31}, c_{32})$
$D(\underline{c_1}, \underline{d_1}, d_2)$

3rd Step: relationships

$A(\underline{a_1}, \underline{a_2})$
$B(\underline{b_1}, b_2, a_1, a_2, x_1)$
$C(\underline{c_1}, c_{31}, c_{32})$
$D(\underline{c_1}, \underline{d_1}, d_2)$
$Y(\underline{b_1}, \underline{c_1})$

4th Step: multi-valued attribues

$A(\underline{a_1}, \underline{a_2})$
$B(\underline{b_1}, b_2, a_1, a_2, x_1)$
$C(\underline{c_1}, c_{31}, c_{32})$
$C4(\underline{c_1}, \underline{c_4})$
$D(\underline{c_1}, \underline{d_1}, d_2)$
$Y(\underline{b_1}, \underline{c_1})$

Part 1.5 Normalization and Functional Dependencies (Total: 22 Points)

Consider the following relation $R(A, B, C, D, E)$ and functional dependencies F that hold over this relation.

$$\begin{aligned} F = & A, B \rightarrow C \\ & A \rightarrow E \\ & C \rightarrow D, E \\ & D \rightarrow B, E \end{aligned}$$

Question 1.5.1 (4 Points)

Determine all candidate keys of R.

Solution

$\{A, B\}$ $\{A, C\}$ $\{A, D\}$

Question 1.5.2 (8 Points)

Compute the canonical cover of F. Show each step of the generation according to the algorithm shown in class.

Solution

1th iteration: 1) Apply union rule to combine right-hand sides:
no changes
1th iteration: 2) Find extraneous attribute:
E is extraneous in $C \rightarrow D, E$

$$\begin{aligned} F_1 = & A, B \rightarrow C \\ & A \rightarrow E \\ & C \rightarrow D \\ & D \rightarrow B, E \end{aligned}$$

2nd iteration: 1) Apply union rule to combine right-hand sides:
no changes
2nd iteration: 2) Find extraneous attribute:
no changes

Question 1.5.3 (6 Points)

Apply the 3NF decomposition algorithm to relation R.

Solution

1st step: Create one relation for each FD in F_c:

$$R_1(A, B, C)$$
$$R_2(A, E)$$
$$R_3(C, D)$$
$$R_4(B, D, E)$$

2st step: Remove redundant relations
none are redundant

3st step: If no relation contains all attributes of a candidate key, then create a relation with all attributes of a candidate key
R_1 contains $\{A, B\}$

Question 1.5.4 (4 Points)

Which of the statements below are true and which ones are false? Put a check mark into the box to indicate that a statement is correct or an X to indicate that a statement is wrong. You also have the option to leave the check box empty if you are unsure. Incorrect answers will result in negative points (you will get at least 0 points for this question).

- ❑ In 2NF, for every non-trivial dependency $\alpha \rightarrow \beta$, α is a superkey
- ❑ Every superkey is a candidate key
- ■ Every candidate key is a superkey
- ❑ A relation can have at most one superkey
- ■ Let α and β be candidate keys. A functional dependency $\alpha \rightarrow \beta$ does not violate 3NF.
- ■ Testing whether a relation is in 3NF is harder than decomposing a relation into 3NF. This is because testing requires the generation of all candidate keys for a relation.
- ■ Every relation in BCNF is also in 2NF.
- ■ The difference between BCNF and 3NF is that 3NF allows dependencies between parts of candidate keys.

Part 1.6 Concurrency Control (Total: 9 Points)

Question 1.6.1 (9 Points)

For each of the following schedules determine which properties this schedule has. E.g., a schedule may be *recoverable* and *cascade-less* (*strict*). Consider the following notation for operations of transactions:

$w_1(A)$	transaction 1 wrote item A
$r_1(A)$	transaction 1 read item A
c_1	transaction 1 commits
a_1	transaction 1 aborts

$$S_1 = r_1(C), r_1(B), r_2(A), w_2(A), w_1(B), r_2(C), w_2(C), c_1, w_2(B), c_2$$

$$S_2 = r_2(C), w_2(A), w_1(A), w_2(B), c_2, r_3(B), c_1, c_3$$

$$S_3 = r_2(A), r_1(B), w_2(A), r_2(B), r_3(A), w_1(B), c_1, w_3(A), c_3, w_2(B), c_2$$

- ■ S_1 is recoverable
- ■ S_1 is cascade-less
- ■ S_1 is conflict-serializable
- ■ S_1 is 2PL
- ■ S_1 is S2PL
- ❑ S_1 is SS2PL

- ■ S_2 is recoverable
- ■ S_2 is cascade-less
- ■ S_2 is conflict-serializable
- ■ S_2 is 2PL
- ❑ S_2 is S2PL
- ❑ S_2 is SS2PL

- ❑ S_3 is recoverable
- ❑ S_3 is cascade-less
- ❑ S_3 is conflict-serializable
- ❑ S_3 is 2PL
- ❑ S_3 is S2PL
- ❑ S_3 is SS2PL