SWE3003 Introduction to Database Systems - Midterm Fall 2024

Student ID	Name

For Instructor/TA only,

Q1	Q2	Q3	Q4	Q5	Q6	Total

Academic Honor Pledge

I affirm that I will **not** at any time be involved with **cheating** or **plagiarism** while enrolled as a student of **Introduction to Database Systems** class at Sungkyunkwan University. I understand that violation of this code will result in penalties as severe as indefinite suspension from the university.

Your signature: ______________________

1. [30 pts] For each of the following statements, indicate whether it is TRUE or FALSE. You will get 3 points for each correct answer, -3 points for each incorrect answer, and 0 points for each blank answer or both marked answers.

			T	F
(a)	Physical data independence is the ability to modify the logical schema without changing the physical schema.	T	☐	☐
(b)	The DELETE statement in SQL deletes a table.	F	☐	☐
(c)	Normalization improves query performance by reducing the number of joins required.	F	☐	☐
(d)	Weak entities do not have a primary key of their own.	T	☐	☐
(e)	A many-to-many relationship between two entities requires a relationship (mapping) table when implemented in a relational DBMS.	T	☐	☐
(f)	In a relational DBMS, all rows within a table must be unique.	F	☐	☐
(g)	If there are duplicate rows in a table, no superkey can exist.	T	☐	☐
(h)	Total participation of an entity in a relationship means that every instance of the entity must be related to at least one instance of another entity.	T	☐	☐
(i)	A 'view' is a virtual table that does not store data physically but is based on the result of a query.	T	☐	☐
(j)	The ON DELETE CASCADE option in a foreign key constraint ensures that when a referenced row is deleted, all related rows are also deleted.	T	☐	☐

2. [10 pts] Consider the following schema for a Music streaming service, such as Melon, Spotify, etc.

   - Artist(artist_id, artist_name, bio)
   - Play(artist_id, song_id)
   - Song(song_id, song_title, duration, genre)
   - Album(album_id, album_name, release_date)
   - Contain(album_id, song_id)

   (Constraint 1) Multiple artists can play the same song for their own albums.

   (Constriaint 2) The Contain relationship between Album and Song is many-to-many. That is, an album can contain multiple songs, and a song can belong to multiple albums.

   (Constriaint 3) Every song must be part of at least one album, i.e., total participation.

   Draw an ER diagram:

3. [20 pts] Consider the following schema with the same constraints described in problem 2.

- Artist(<u>artist_id</u>, artist_name, bio)
- Play(<u>artist_id</u>, <u>song_id</u>)
- Song(<u>song_id</u>, song_title, duration, genre)
- Album(<u>album_id</u>, album_name, release_date)
- Contain(<u>album_id</u>, <u>song_id</u>)

- (a) Write a relational algebra that retrieves the titles of all 'K-pop' genre songs in the DB.

answer:
$\pi_{song\_title}(\sigma_{genre='K-pop'}(\text{Song}))$

- (b) Write a relational algebra that retrieves all the songs played by 'Beatles'.

answer:
$\pi_{song\_title}\ (\sigma_{artist\_name='Beatles'}(\text{Artist}) \bowtie \text{Play} \bowtie \text{Song})$
or
$\pi_{song\_title}\ (\sigma_{artist\_name='Beatles'}(\text{Artist} \bowtie \text{Play}) \bowtie \text{Song})$
or
$\pi_{song\_title}\ (\sigma_{artist\_name='Beatles'}(\text{Artist} \bowtie \text{Play} \bowtie \text{Song}))$
or some other equivalent expressions.

- (c) Write a relational algebra that retrieves the names of all albums that contain both the song titled 'Parklife' and the song titled 'Song 2'.

answer:
$\pi_{album\_name}\ (\sigma_{song\_title='Parklife'}(\text{Song}) \bowtie \text{Contain} \bowtie \text{Album})$
$\cap$
$\pi_{album\_name}\ (\sigma_{song\_title='Song2'}(\text{Song}) \bowtie \text{Contain} \bowtie \text{Album})$

INCORRECT:
$\pi_{album\_name}\ (\sigma_{song\_title='Parklife'}(\text{Song}) \bowtie \text{Contain} \bowtie \text{Album}$
$\cap$
$\sigma_{song\_title='Song2'}(\text{Song}) \bowtie \text{Contain} \bowtie \text{Album}\ )$

- (d) Write a relational algebra that retrieves all the titles of songs played by multiple artists.

answer:
$\pi_{song\_title}(\sigma_{count(song\_id)>1}(_{song\_id}G_{count(artist\_id)}(Play)) \bowtie Song)$

Note: Unfortunately, I figured we didn't cover relational algebra for Group By operator in class. So, this question will not be graded. Sorry.

4. [30 pts] Write each of the following queries in SQL for the given relations with the same constraints described in problem 2.

- Artist(artist_id, artist_name, bio)
- Play(artist_id, song_id)
- Song(song_id, song_title, duration, genre)
- Album(album_id, album_name, release_date)
- Contain(album_id, song_id)

(a) Find the names of the albums that contain at least one song of the "Pop" genre.

```
SELECT DISTINCT album_name   -- DISTINCT can be ommitted
FROM Album NATURAL JOIN Contain NATURAL JOIN Song
WHERE genre = 'Pop'
```

;

,

```
SELECT DISTINCT album_name   -- DISTINCT can be ommitted
FROM Album JOIN Contain USING album_id
      JOIN Song USING song_id
WHERE S.genre = 'Pop'
```

;

, or alternatively, you may use ON construct or WHERE predicates to specify the join conditions.

(b) Find the total number of songs each artist has played.

```
SELECT artist_name, COUNT(song_id)
FROM Artist NATURAL JOIN Play
GROUP BY artist_name
```

;

(Cont'd)

- Artist(<u>artist_id</u>, artist_name, bio)
- Play(<u>artist_id</u>, <u>song_id</u>)
- Song(<u>song_id</u>, song_title, duration, genre)
- Album(<u>album_id</u>, album_name, release_date)
- Contain(<u>album_id</u>, <u>song_id</u>)

(c) Find the titles of songs that are contained in more than one album.

```
SELECT song_title
FROM Song NATURAL JOIN Contain
GROUP BY song_id
HAVING COUNT(album_id) > 1
```

;

(d) Find the names of artists who played songs on albums that contain 10 or more songs.

```
SELECT DISTINCT artist_name
FROM Artist NATURAL JOIN Play
WHERE song_id IN (
    SELECT song_id
    FROM Contain NATURAL JOIN (
        SELECT album_id
        FROM Contain
        GROUP BY album_id
        HAVING COUNT(song_id) >= 10
    ) AS subquery
);
```

;

The following query will get 2 points

```
SELECT DISTINCT artist_name
FROM Artist
     NATURAL JOIN Play
     NATURAL JOIN Contain
     NATURAL JOIN Album
GROUP BY album_id
HAVING COUNT(c.song_id) >= 10;
```

(Cont'd)

- Artist(artist_id, artist_name, bio)
- Play(artist_id, song_id)
- Song(song_id, song_title, duration, genre)
- Album(album_id, album_name, release_date)
- Contain(album_id, song_id)

(e) Write a SQL query using the 'WITH' clause to find the titles of songs that have a duration longer than the average duration of songs in their respective genre.

```
WITH AverageDurations AS (
    SELECT genre, AVG(duration) AS avg_duration
    FROM Song
    GROUP BY genre
)
SELECT s.song_title
FROM Song s
JOIN AverageDurations a ON s.genre = a.genre
WHERE s.duration > a.avg_duration;
```

(f) Write a SQL query using the Correlation Variable to find the titles of songs that have a duration longer than the average duration of songs in their respective genre.

```
SELECT song_title
FROM Song s1
WHERE duration > (
    SELECT AVG(duration)
    FROM Song s2
    WHERE s1.genre = s2.genre
);
```

5. [10 pts] Consider the following schema.

- Artist(<u>artist_id</u>, artist_name, bio)
- Play(<u>artist_id</u>, <u>song_id</u>)
- Song(<u>song_id</u>, song_title, duration, genre)

If an artist is deleted from the Artist table, all rows in the Play table that reference that artist need to be automatically deleted. Write DDL statements for the Artist and Play tables.

artist_id: INT
artist_name: VARCHAR(100)
bio: TEXT
song_id: INT

(a) Artist table:

```
CREATE TABLE Artist (
    artist_id INT PRIMARY KEY,
    artist_name VARCHAR(100) NOT NULL,
    bio TEXT
)

;
```

(b) Play table:

```
CREATE TABLE Play (
    artist_id INT,
    song_id INT,
    PRIMARY KEY (artist_id, song_id),
    FOREIGN KEY (artist_id) REFERENCES Artist(artist_id) ON DELETE CASCADE,
    FOREIGN KEY (song_id) REFERENCES Song(song_id)
)

;
```

6. [15 pts] Consider the following relation and the functional dependency set:

- R = (A, B, C, D, E, F)
- FD = {AB $\rightarrow$ DE, B $\rightarrow$ F, DF $\rightarrow$ C }

(a) Find a candidate key of R.
answer:
AB
AB $\rightarrow$ DE
(ABDE)
B $\rightarrow$ F
(ABDEF)
DF $\rightarrow$ C
(ABCDEF)

(b) Find functional dependencies that violate the BCNF, if any.
answer: B $\rightarrow$ F and DF $\rightarrow$ C violate BCNF.

(c) If R is not in BCNF, decompose it into multiple relations where each one becomes in BCNF. For each step, clearly identify which FD you use for the decomposition.
answer:
(answer) If you choose to decompose using DF $\rightarrow$ C first, (ABCDEF) will be decomposed into (ABDEF) and (CDF).
For (ABDEF), B$\rightarrow$F and AB$\rightarrow$DE are preserved. Still, (ABDEF) is not in BCNF because of B$\rightarrow$F.
(ABDEF) is now decomposed into (ABDE) and (BF).
(ABDE) is in BCNF, i.e., AB is the super key. AB$\rightarrow$DE.
(BF) is also in BCNF.

(another correct answer) If you choose to decompose using B $\rightarrow$ F first, (ABCDEF) will be decomposed into (ABCDE) and (BF).
What is the super key of (ABCDE)?
ABC is the new super key of (ABCDE) because we lost DF$\rightarrow$C.
Now, AB$\rightarrow$DE violates the BCNF.
So, (ABCDE) is decomposed into (ABDE) and (ABC).