

# CSE 414 Midterm Exam

Autumn 2019

October 30, 2019

- Please read all instructions (including these) carefully.
- Write your name and UW student number below.
- **This is a closed-book exam. You are allowed one page of note sheets that you can write on both sides.**
- No electronic devices are allowed, including **cell phones** used merely as watches. Silence your cell phones and place them in your bag.
- Solutions will be graded on correctness and **clarity**. Each problem has a relatively simple and straightforward solution. Partial solutions will be graded for partial credit.
- Please write your answers in the boxed space provided on the exam, and clearly mark your solutions. You may use the blank sides as scratch paper. **Do not use any additional scratch paper.**
- *Relax. You are here to learn. Good luck!*

Relational algebra operators:

Union  $\cup$           Difference  $-$           Selection  $\sigma$           Projection  $\pi$           Join  $\bowtie$   
Rename  $\rho$           Duplicate elimination  $\delta$           Grouping and aggregation  $\gamma$           Sorting  $\tau$

Database constraints:

Primary key, Foreign key, Unique, Not null

By writing your name below, you certify that you have not received any unpermitted aid for this exam, and that you will not disclose the contents of the exam to anyone in the class who has not taken it.

NAME: \_\_\_\_\_

STUDENT NUMBER: \_\_\_\_\_

Problem	Points	Problem	Points
1	10	4	9
2	32	5	12
3	12	<b>Total</b>	75

## Problem 1: True/False (10 points total)

Select either True or False for each of the following questions.

- |   |      |       |
|---|------|-------|
| a) SQL and Datalog are declarative query languages.   | True | False |
| b) The value of a Primary Key for a relation uniquely identifies a tuple within that relation.  | True | False |
| c) In the relational model, the field of a tuple can itself be a relation.  | True | False |
| d) When using the aggregate <code>COUNT(attr)</code> , tuples with NULL values in <code>attr</code> field will be added to the total count. | True | False |
| e) In the relational model, the schema defines both the name and type of each attribute.  | True | False |
| f) In a single table's schema, a Primary Key cannot also be a Foreign Key.  | True | False |
| g) The SQL we have learned this quarter cannot express recursive queries.   | True | False |
| h) Expressing recursive queries in Datalog requires negation.   | True | False |
| i) The following Datalog rule is safe:<br>$Q(a, b) \text{ :- } R(a, \_), ! S(b, a)$   | True | False |
| j) The following Datalog rule is safe:<br>$Q(z) \text{ :- } R(a, b), S(b, c), T(c, z), ! R(a, z)$   | True | False |

## Problem 2: SQL (32 points total)

We will work with the following schema for a database of movies in this exam.

**ACTOR** (pid, fname, lname, gender)

**MOVIE** (mid, name, year, revenue)

**DIRECTORS** (did, fname, lname)

**CASTS** (pid, mid, role)

**MOVIE\_DIRECTORS** (did, mid)

A tuple in the *Casts* relation represents that a person from the *Actor* table with *pid*, starred (was cast) in a *Movie* with *mid*. *Movie\_Directors* represents a person from *Directors* with *did*, who directed a *Movie* with *mid*.

**ACTOR**.pid, **MOVIE**.mid, **DIRECTOR**.did = primary keys of the corresponding tables

**CASTS**.pid = foreign key to **ACTOR**.pid

**MOVIE\_DIRECTORS**.did = foreign key to **DIRECTORS**.did

**CASTS**.mid, **MOVIE\_DIRECTORS**.mid= foreign keys to **MOVIE**.mid

You can assume none of the tables contain NULL values. You may use subqueries for these questions. When writing your queries you can assume that the system is case-insensitive.

a) (8 points) Write a query that returns all the roles played by actors named “Kevin Bacon” (first name Kevin, last name Bacon.) Your query should return the *role* field. Do not include duplicates roles.

Schema repeated here for your reference:

**ACTOR** (pid, fname, lname, gender)

**MOVIE** (mid, name, year, revenue)

**DIRECTORS** (did, fname, lname)

**CASTS** (pid, mid, role)

**MOVIE\_DIRECTORS** (did, mid)

b) (8 points) Write a query that returns the first name and last name of all actors who starred in both the movies 'In the Mood for Love' and 'Chungking Express'. Only return actors who were in both movies, not just one of the movies. Duplicate names are allowed.

c) (8 points) Find the total number of movies made in each year, and return the results sorted by the number of movies, in descending order. Return the year numbers and count of movies made in that year.



Schema repeated here for your reference:

**ACTOR** (pid, fname, lname, gender)

**MOVIE** (mid, name, year, revenue)

**DIRECTORS** (did, fname, lname)

**CASTS** (pid, mid, role)

**MOVIE\_DIRECTORS** (did, mid)

d) (8 points) Write a query that returns all directors who only directed movies alone (i.e. who never directed a movie together with another director.) For any directors with no movies at all, include them in the output. Return the did field for each director.

### Problem 3: Relational Algebra (12 points total)

We will use the same schema as problem 2, repeated here for your reference:

**ACTOR** (pid, fname, lname, gender)

**MOVIE** (mid, name, year, revenue)

**DIRECTORS** (did, fname, lname)

**CASTS** (pid, mid, role)

**MOVIE\_DIRECTORS** (did, mid)

Note: you may use aliases in the query plan to avoid renaming, e.g.

$\bowtie_{x.pid=y.pid}$   
/ \  
WorksOn x Project y

a) (12 points) The following query lists all directors who directed at least 200 movies, in descending order of the number of movies they directed

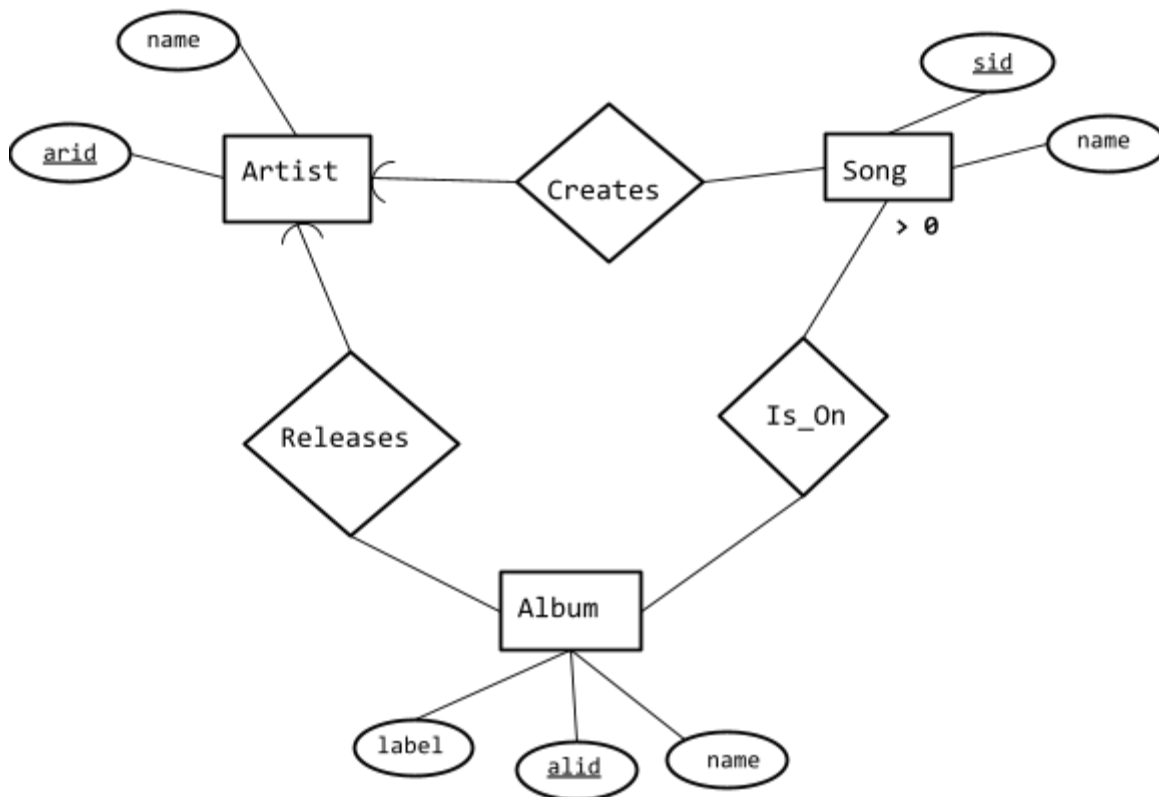
```
SELECT d.fname, d.lname, COUNT(*) AS c
FROM Directors d, Movie_Directors md
WHERE d.did = md.did
GROUP BY d.did, d.fname, d.lname
HAVING COUNT(*) >= 200
ORDER BY c DESC;
```

Write a Relational Algebra expression in the form of a logical query plan (you may draw a tree) that is equivalent to the above SQL query.

## Problem 4: Entity Relationship Diagrams (9 points total)

For this problem consider a database of musical artists, albums, and songs. Artists can be individuals like Adele, or bands like The Beatles. We have three entities and relationships between them:

- A **Song** is created by a single **Artist**
- **Songs** are published on albums
- An **Album** is released by a single **Artist**



- **Is\_On** is a many-to-many relationship. Albums can have multiple songs, and songs can be on multiple albums (The Essential Beatles, The Beatles' Greatest Hits, etc.) An Album must have at least one song.
- **Creates** is a many-to-exactly-one relationship from Song to Artist.
- Similarly, **Releases** is a many-to-exactly-one relationship from Album to Artist.



- (a) Assume we are trying to **minimize the number of tables** we use to represent the above entities.

Fill in the create table statements for the **Song** and **Album** relations, making sure to preserve the constraints in the diagram above and trying to minimize the number of tables we will need in total. The underlined attributes above are the primary keys of the respective entity sets. The types of arid, sid, and alid are INT, and all other attributes are VARCHAR(100).

(8 points)

```
CREATE TABLE Song (
```

```
);
```

```
CREATE TABLE Album (
```

```
);
```

- (b) Do we need a table for the “Is\_On” relationship? Write a sentence explaining why/why not. (1 point)

## Problem 5: Functional Dependencies (12 points total)

Assume we have a relation  $R$  and its attributes in parentheses, and we know that it satisfies the following functional dependencies:

$R(A, B, C, D, E)$

$A \rightarrow BC$

$D \rightarrow E$

$B \rightarrow E$

$CD \rightarrow A$

For the functional dependencies below, you will decide if we can derive them from the information above. Circle YES if we can guarantee that  $R$  satisfies the functional dependency given the knowledge we have. If not, circle NO. (2 points each)

- |    |                       |     |    |
|----|-----------------------|-----|----|
| a) | $A \rightarrow A$     | YES | NO |
| b) | $A \rightarrow AD$    | YES | NO |
| c) | $AD \rightarrow A$    | YES | NO |
| d) | $CD \rightarrow ABCD$ | YES | NO |
| e) | $DE \rightarrow BE$   | YES | NO |
| f) | $C \rightarrow ABCDE$ | YES | NO |