CMPS182 Midterm Examination

Name:

Question 1:
Question 2:
Question 3:
Question 4:
Question 5:
Question 6:
Warm Up – Short Questions (10 points)

• What is the difference between a view and a normal table?

   The contents of the view are defined in terms of a query and are computed on-the-fly when the view is referenced.

• Which SQL clause allows us to apply conditions on groups in a GROUP BY query?

   The HAVING clause.

• Is the Join a core operator of Relational Algebra? Briefly justify your answer.

   It is not a core operator, as the join is a cross product followed by a selection operator.

• Assume that we examine the instance of a relation R and discover that its attribute X takes unique values in this instance. Can we deduce that X is a key? Why or why not?

   No, because the property of a key is attached to the schema of a relation. The uniqueness of X may hold for the particular instance of R, but not for every instance.

• Which Relational Algebra operator corresponds to the SELECT clause of SQL?

   The projection operator.
Relational Model and Database Design (20 points)

1. Assume that X and Y are subsets of the attributes of a relation R. Suppose that X is a strict superset of Y (this means that all the attributes in Y appear in X, and X contains attributes that do not appear in Y).

   • If Y is a key, then is X a key? Justify briefly.

   *If Y takes unique values then so does X, since X contains Y. Hence, X is a key.*

   • What of the inverse, i.e., if X is a key, is Y a key as well? Justify briefly.

   *Not necessarily. Y may be a subset of attributes that do not take unique values in X.*

2. Consider the following schema that encodes information on movies and moviegoers:

   Movies(mid: integer, title: string, year: integer)
   Person(pid: integer, name: string, address: string)
   Saw(pid: integer, mid: integer, when: date, theaterAddress: string)

   All strings have a maximum length of 20.

   • Write the SQL command to create table Saw. Make sure to declare any primary keys and foreign keys to other tables.

   ```
   CREATE TABLE Saw(
       pid INTEGER,
       mid INTEGER,
       when DATE,
       theaterAddress VARCHAR(20),
       FOREIGN KEY (mid) REFERENCES Movies,
       FOREIGN KEY (pid) REFERENCES Person,
       PRIMARY KEY (pid,mid)
   );
   ```
• What is the relationship between Movies and Person in this particular schema (1-1, 1-many, or many-many)? Briefly justify your answer.

*It is many-to-many, since a movie can be associated to many persons through relation Saw and vice versa.*

• Assume that the key to “Saw” comprises solely of attribute pid instead of (pid,mid) as in the current schema. Does your previous answer change? Explain briefly.

*If the key is solely pid, then each pid value can appear at most once in Saw. This implies that each person can be associated once with a movie, hence the relationship between Person and Movie becomes many-to-one.*

3. Consider a relation \( R(X,Y,Z) \) where attribute \( X \) has a domain of 10 possible values, \( Y \) has a domain of 5 possible values, and \( Z \) has a domain of 100 possible values. What is the maximum number of tuples in any instance of \( R \) in each of the following cases? Briefly justify your answer.

• \( Z \) is a key.

\( Z \) must take unique values and there are 100 such values. Hence, \( R \) can have at most 100 tuples.

• \( (X,Y) \) is a key.

\( (X,Y) \) must take unique values and there are \( 10 \times 5 \) such combinations. Hence, \( R \) can have at most 50 tuples.
Relational Algebra (30 points)

1. The symmetric difference between two tables R and S contains the tuples of R and S that do not appear in their intersection. Write the relational algebra expression that computes the symmetric difference.

\[ R - S \cup S - R \]

2. Consider the five relational algebra operators union, intersection, projection, selection, and cross product. Which of these can generate more output than their input? Explain briefly for each case.

**Cannot generate more output:**

- **Projection:** Each output tuple corresponds to at least one input tuple.
- **Selection:** The output contains the subset of input tuples that satisfy a condition.
- **Union:** Each output tuple corresponds to at least one input tuple.
- **Intersection:** The output is a subset of the tuples in the union. Since the union cannot generate more tuples, the same holds for intersection.

**May generate more output:**

- **Cross Product:** The output contains \( n \times m \) tuples, which can be larger than \( n + m \) if both \( n \) and \( m \) are greater than 1.
3. Consider the following schema that describes information on movies:

\[
\begin{align*}
\text{Actors} & : \text{aid, name, yob} \\
\text{Movies} & : \text{mid, title, year} \\
\text{StarsIn} & : \text{aid, mid, order}
\end{align*}
\]

Assume that StarsIn.order specifies the order in which an actor appears in a movie. Write the following queries in relational algebra:

- Retrieve the names of actors who starred in the 2006 edition of ‘Pride and Prejudice’.

\[
\pi_{\text{name}} \left( \sigma_{\text{title}=\text{‘Pride and Prejudice’} \land \text{year}=2006} \left( M \bowtie S \bowtie A \right) \right)
\]

- Retrieve the movies in which the first actor was younger than 20 years old at the time the movie was released.

\[
\sigma_{\text{year} - \text{yob}<20} \left( \sigma_{\text{order}=1} \left( S \bowtie Movies \bowtie A \right) \right)
\]

- Retrieve the names of actors who have appeared in all movies released in 2005.

\[
\pi_{\text{name}, \text{mid}} \left( A \bowtie S \bowtie \sigma_{\text{year}=2005} \left( M \right) \right) / \pi_{\text{mid}} \left( \sigma_{\text{year}=2005} \left( M \right) \right)
\]
SQL (30 points)
Consider the following relational schema about Parts and Merchants who supply them:

Part( pid, pname, pcolor )
Merchant(mid, mname, maddress)
Supplies(mid, pid, price)

Write a single SQL query for each one of the following questions.

• Retrieve the names of merchants who supply a 'red' or a 'green' part at a price less than $20.

```sql
SELECT m.mname
FROM Merchant m
WHERE m.mid IN ( 
    SELECT s.mid
    FROM Part p, Supplies s
    WHERE p.pid = s.pid AND (p.pcolor = 'red' OR p.pcolor = 'green') AND s.price < 20)
```

• Retrieve pairs of merchant names who supply at least one common part.

```sql
SELECT m1.mname, m2.mname
FROM Merchant m1, Merchant m2, Supplies s1, Supplies s2
WHERE m1.mid = s1.mid AND m2.mid = s2.mid AND s1.pid = s2.pid;
```

• Retrieve the ids of parts that can be supplied both by merchants in 'Santa Cruz' and by merchants in 'Capitola'.

```sql
(SELECT s.pid
FROM Supplies s, Merchant m
WHERE s.mid = m.mid AND m.maddress = 'Santa Cruz')
INTERSECT
(SELECT s.pid
FROM Supplies s, Merchant m
WHERE s.mid = m.mid AND m.address = 'Capitola');
```
• For each part, retrieve its name, its average price, and the total number of merchants that supply it.

SELECT p.pname, t.avgPrice, t.mcount
FROM Part p, (SELECT pid,
    AVG(price) AS avgPrice,
    COUNT(DISTINCT mid) AS mcount
    FROM Supplies
    GROUP BY pid) AS t
WHERE p.pid = t.pid;

• Retrieve the part names that are supplied by more than 10 merchants.

SELECT p.pname
FROM Part p, (SELECT pid
    FROM Supplies
    GROUP BY pid
    HAVING COUNT(DISTINCT mid) >10) AS t
WHERE p.pid = t.pid

• For each merchant, retrieve his/her name and the parts that are more expensive than the average price of parts offered by the same merchant.

SELECT m.mname, p.pname
FROM Merchant m, Supplies s, Part p
WHERE m.mid = s.mid AND s.pid = p.pid AND s.price >
  (SELECT AVG(s.price)
    FROM Supplies sn
    WHERE sn.mid = m.mid)
Views (10 points)
Consider the following schema that records information on students and courses.

Student( sid, name, level )
Course( cid, title, dept )
Enrolled( sid, cid, quarter)

- Create a view “CurrentEnrollment” that records the total number of enrolled students per department and per quarter. The view has exactly three attributes: the department name, the quarter, and the total number of students that were enrolled in courses offered by the department in that quarter.

CREATE VIEW CurrentEnrollment(dname, quarter, totNumber) AS
SELECT c.dept, e.quarter, COUNT(*)
FROM Enrolled e, Course c
WHERE e.cid = c.cid
GROUP BY c.dept, e.quarter

- Create a view “MostPopular” that records the most popular courses. A course is the most popular if its total enrollment count (across all quarters) is the highest among all courses.

**Hint:** The following query returns the maximum total enrollment count. You can use it as the building block for the view.

```
SELECT MAX(t.enrollmentCount)
FROM ( SELECT COUNT(*) AS enrollmentCount
        FROM Enrolled
        GROUP BY cid
    ) AS t
```

CREATE VIEW MostPopular AS
SELECT e.cid, c.title, c.dept
FROM Enrolled e, Course c
WHERE e.cid = c.cid
GROUP BY e.cid, c.title, c.dept
HAVING COUNT(*) >= ( SELECT MAX(t.ec)
                      FROM (SELECT COUNT(*) AS ec
                             FROM Enrolled
                             GROUP BY cid) AS t )
**Trivia**

He was historically the first to challenge the geocentric worldview and propose a heliocentric hypothesis. He also used his mathematical skills and his astronomical observations to estimate the diameter of the Moon and the distance between the Earth and the Sun. Name him.