CS411: Database Systems **Spring 2017**

Assignment 3 Solutions

1 SQL queries involving a single relation

Consider a relation with just one attribute, $R(A)$. Express the following queries using SQL. Your queries will be graded based on their simplicity, and correctness.

1. Write a query to compute the largest value of A in R. You are not allowed to use the `Max` operator. For this question, you may assume that there are no duplicates in R.
2. Write a query to compute the median value of A in R. For this question, you may assume that there are no duplicates in R, and that R has odd number of tuples.
3. Write a query to compute the mode of A in R. If multiple distinct values of A are candidates, then all of them should be returned. For this question, R may contain duplicates tuples; please don't assume otherwise.

Answers:

1. Largest value in R:

```
SELECT A
FROM R
WHERE A >= ALL (SELECT A FROM R)
```

2. Median value in R:

```
SELECT A
FROM R
WHERE (SELECT COUNT(*) FROM R AS R2
                WHERE R2.A < R.A) =
        (SELECT COUNT(*) FROM R AS R2
                WHERE R2.A > R.A)
```

3. Mode value/s in R:

```
SELECT A
FROM R
GROUP BY A
HAVING COUNT(*) >= ALL (SELECT COUNT(*)
        FROM R GROUP BY A)
```

2 SQL queries involving multiple relations

Consider the following relations:

```
Courses(cid, cname, room, pid)
Professors(pid, pname)
Students(sid, sname)
Enrolled(sid,cid)
```

Express the following queries using SQL. Your queries will be graded based on their simplicity, and correctness.

1. Find the names of students who are not enrolled in any class.
2. Find the names of students who are enrolled in the highest number of classes, i.e., all other students (not outputted) have taken fewer classes.
3. Find the names of professors who teach in every room in which some course is taught.

Answers:

1.
```
SELECT DISTINCT S.name
FROM Students S
WHERE S.sid NOT IN
        (SELECT E.sid FROM Enrolled E)
```

2.
```
SELECT DISTINCT S.sname
FROM Students S
WHERE S.sid IN (SELECT E.sid
        FROM Enrolled E
        GROUP BY E.sid
        HAVING COUNT(*) >= ALL (
                SELECT COUNT(*)
                FROM Enrolled E2
                GROUP BY E2.sid))
```

3.
```
SELECT DISTINCT P.pname
FROM Professors P
WHERE NOT EXISTS (
        (SELECT C1.room
                FROM Courses C1)
        EXCEPT
        (SELECT C2.room
                FROM Courses C2
                WHERE C2.pid = P.pid))
```

3 Database Manipulation and Views

Consider the relations `Courses, Professors, Students, Enrolled` as described in question 3.

1. Insert the following new tuple in `Students`: John with `sid` as `22222`.
2. Remove all professors from `Professors`, who are not teaching any course.
3. Update the name of the student with `sid` as `22222`, to Doe.
4. Create a materialized view `BusyProfessors`, containing the names of all professors for whom the number of students they teach across their courses is greater than hundred.

   Note: if a student is enrolled in two different courses of the same professor, we count the student only once.

Answers:

1.
   ```
   INSERT INTO Students(sid, sname)
           VALUES (‘22222’, ‘John’)
   ```

2.
   ```
   DELETE FROM Professors P
   WHERE P.pid NOT IN
           (SELECT C.pid FROM Courses C)
   ```

3.
   ```
   UPDATE Students
   SET sname = ‘Doe’
   WHERE sid = ‘22222’
   ```

4.
   ```
   CREATE MATERIALIZED VIEW BusyProfessors AS
           SELECT DISTINCT P.pname
           FROM Professors P
           Where 100 < (SELECT COUNT (DISTINCT E.sid)
                   FROM Courses C, Enrolled E
                   WHERE C.cid = E.cid
                   AND C.pid = P.pid)
   ```

4 Constraints and Triggers

Consider the relations `Courses, Professors, Students, Enrolled` as described in question 3.

1. Create the tables for `Courses, Professors, Students, Enrolled` using the SQL `CREATE` statement, with the following constraints:

- For `Enrolled`, all the student and course ids should be present in the corresponding tables
- For `Courses`, the professor id must be present in the corresponding table

2. A deletion or update to a `sid` should be handled appropriately in `Enrolled`, due to the constraint as explained in the previous question.

   (a) Please rewrite the SQL `CREATE` statement such that when an `sid` gets updated/deleted in `Students`, its value is propagated accordingly to `Enrolled`.

   (b) Also, explain why we can not set `Enrolled.sid` to `NULL` when we delete that `sid` from `Students`.

3. Create a trigger so that an insertion into `Courses` creates an tuple in `Professors` if the `pid` corresponding to the insertion doesn't exist in `Professors`.

Answers:

1.
```
CREATE TABLE Professors (
        pid INT PRIMARY KEY,
        pname VARCHAR(100)
);

CREATE TABLE Students (
        sid INT PRIMARY KEY,
        sname VARCHAR(100)
);

CREATE TABLE Courses (
        cid INT PRIMARY KEY,
        cname VARCHAR(100),
        room VARCHAR(100),
        pid INT REFERENCES Professors(pid)
);

CREATE TABLE Enrolled (
        sid INT REFERENCES Students(sid),
        cid INT REFERENCES Courses(cid),
        PRIMARY KEY(sid, cid)
);
```

2. (a)
```
CREATE TABLE Enrolled (
        sid INT REFERENCES Students(sid)
                ON UPDATE CASCADE
                ON DELETE CASCADE,
        cid INT REFERENCES Courses(cid),
        PRIMARY KEY(sid, cid)
);
```

(b) We can not set `Enrolled.sid` to `NULL` when we delete that `sid` from `Students` because primary keys can not be `NULL`.

3.
```
CREATE TRIGGER CoursesTrig
        AFTER INSERT ON Courses
        REFERENCING NEW ROW AS NewCourse
        FOR EACH ROW
        WHEN (NewCourse.pid NOT IN
                (SELECT pid FROM Professors))
        INSERT INTO Professors(pid, pname)
                VALUES(pid, NULL);
```

5 Closure of Functional Dependencies

Consider a relation $R(A_1, A_2, \ldots, A_n)$, with $n \geq 3$. R satisfies the following functional dependency: $A_1 \rightarrow A_2$.

1. How many logically implied functional dependencies are of the form, $P \rightarrow A_2A_n$, where $P \subseteq \{A_1, A_2, \ldots, A_n\}$.[1]

Answer: Consider a valid P. We first make the following observation: $A_n \in P$. This is because $P \rightarrow A_n$, and the only functional dependency of R provided does not contain A_n. Now, there are two possibilities:

i. $A_2 \in P$: Since $A_2 \in P$ and $A_n \in P$, we can have any subset of the remaining attributes—$\{A_1, A_3, \ldots, A_{n-1}\}$—in P, in addition to A_2 and A_n, and $P \rightarrow A_2A_n$ would hold. Number of such valid functional dependencies equals 2^{n-2}.

ii. $A_2 \notin P$: Since $A_2 \notin P$, we can ascertain that $A_1 \in P$. Now, we can have any subset of $\{A_3, \ldots, A_{n-1}\}$ in P, in addition to A_1 and A_n. Number of valid functional dependencies in this case equals 2^{n-3}.

Therefore, overall answer $= 2^{n-3} + 2^{n-2} = \boxed{3 \cdot 2^{n-3}}$

[1]based on a question asked on Piazza.