BSCCS2001: Graded Assignment with Solutions
Week 3

Answer questions 1 and 2 based on the relational schema given in Figure 1.

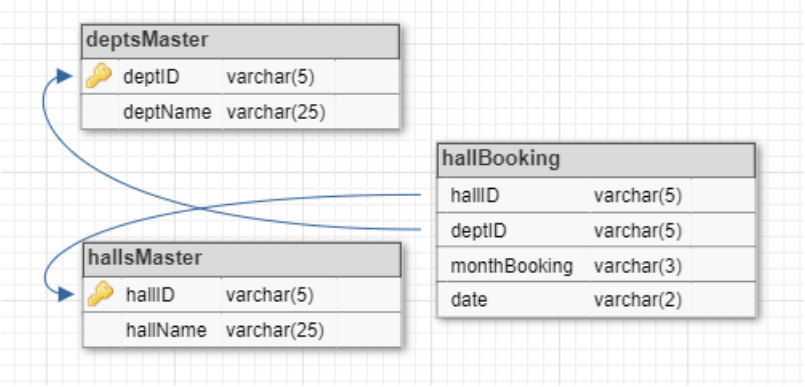

Figure 1: Hall Booking Relational Schema

1. What does the query below return? [MCQ: 2 points]

```
SELECT deptName FROM deptsMaster
WHERE deptID IN (SELECT deptID FROM hallBooking
                 WHERE monthBooking = 'Jan')
    AND deptID NOT in (SELECT deptID FROM hallBooking
                 WHERE monthBooking = 'Feb');
```

- ◯ The names of departments that have booked all halls in the month of January but not in February.
- √ The names of departments that have booked at least one hall in the month of January but never in February.
- ◯ The names of departments that have booked a hall in the month of January or February.
- ◯ The names of departments that have booked at least one hall in the months of January and February.

Solution: The `SELECT` query looks for those departments which has made a booking at least once in January, which is achieved by
`deptID IN (SELECT deptID FROM hallBooking WHERE monthBooking = 'Jan')`
and never in February, which is achieved by
`deptID NOT in (SELECT deptID FROM hallBooking WHERE monthBooking = 'Feb')`.
Since both of these conditions have to be satisfied, these two clauses are connected by `AND`.

2. Find the names of departments that have booked either hall H0001 or hall H0002 or both, in both the months of January and February. [MCQ: 2 points]

◯
```
SELECT deptName FROM deptsMaster
WHERE deptID IN (SELECT deptID FROM hallBooking
                 WHERE hallID IN ('H0001','H0002')
                 AND
                 monthBooking IN ('Jan', 'Feb'));
```

◯
```
SELECT deptName FROM deptsMaster
WHERE deptID IN (SELECT deptID FROM hallBooking
                 WHERE hallID IN ('H0001','H0002')
                 OR deptID IN
                 ((SELECT deptID FROM hallBooking
                 WHERE monthBooking='Jan')
                 UNION
                 (SELECT deptID FROM hallBooking
                 WHERE monthBooking='Feb')));
```

√
```
SELECT deptName FROM deptsMaster WHERE deptID IN
((SELECT deptID FROM hallBooking WHERE hallID IN
('H0001', 'H0002') AND monthBooking = 'Jan')
INTERSECT
(SELECT deptID FROM hallBooking WHERE hallID IN
('H0001', 'H0002') AND monthBooking = 'Feb') )
```

◯
```
SELECT deptName FROM deptsMaster
WHERE deptID IN (SELECT deptID FROM hallBooking
                 WHERE hallID IN ('H0001','H0002')
                 AND
                 SELECT deptID FROM hallBooking
                 WHERE monthBooking='Jan'
                 INTERSECT
                 SELECT deptID FROM hallBooking
                 WHERE monthBooking='Feb');
```

Solution: `monthBooking IN ('Jan', 'Feb')` will select all the rows that have either January or February as the month of booking.
But, what we need is those departments that have booked in both January and February and NOT either of them.
In Option 2, the first nested query and the second nested query (with two select statements connected by an `UNION` operator) are not looking at rows that satisfy the condition of having booked in both months. They are individually listing out rows that satisfy the two conditions, and then the OR is applied. This is not correct.
In Option 4, the same reasoning as that for Option 2 applies.

In Option 3, it will take common tuples to both -

```
SELECT deptID FROM hallBooking
WHERE hallID IN ('H0001', 'H0002') AND monthBooking='Jan'
```

INTERSECT

```
SELECT deptID FROM hallBooking
WHERE hallID IN ('H0001', 'H0002') AND monthBooking='Feb'
```

will give deptID of departments that have booked either hall H0001 or hall H0002 or both, in both the months of January and February. And outer query will give deptName from deptsMaster based on deptID, which is the required set of tuples.

3. Consider the relational schema given in Figure 2.

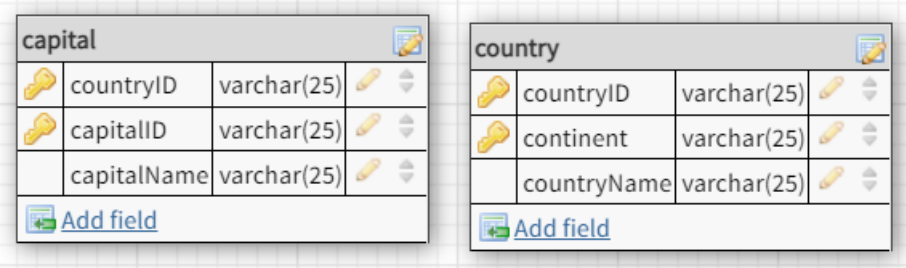

Figure 2: Country Capitals Relational Schema

Choose the SQL statement that returns the capitals of all countries that belong to Asia or Europe. [MSQ: 2 points]

√
```
SELECT capitalName FROM capital
WHERE countryID IN (SELECT countryID FROM country
                    WHERE continent ='Asia'
                    UNION
                    SELECT countryID FROM country
                    WHERE continent ='Europe');
```

◯
```
SELECT capitalName FROM capital
WHERE countryID IN (SELECT countryID FROM country
                    WHERE continent ='Asia'
                    INTERSECT
                    SELECT countryID FROM country
                    WHERE continent ='Europe');
```

◯
```
SELECT capitalName FROM capital
WHERE countryID NOT IN (SELECT countryID FROM country
                    WHERE continent ='Asia'
                    UNION
                    SELECT countryID FROM country
                    WHERE continent ='Europe');
```

√
```
SELECT capitalName FROM capital
WHERE countryID NOT IN (SELECT countryID FROM country
                    WHERE continent != 'Asia'
                    INTERSECT
                    SELECT countryID FROM country
                    WHERE continent != 'Europe');
```

Solution: `countryID IN` filters in and `countryID NOT IN` filters out those countries that satisfy the conditions that follow.

In the first option, two select statements are connected by `UNION` operator in the inner query. One select statement returns countries that belong to Europe, and the second select statement returns countries that belong to Asia. Together with union, the inner query returns all those countries that belong to either Europe or Asia.
In option 2, the connection operator is `INTERSECT`. This will return only those countries which belong to both Asia and Europe.
In option 3, `countryID NOT IN` filters out the countries that belong to either Europe or Asia.
In option 4, `countryID NOT IN` filters out the countries that does not belong to Asia and does not belong to Europe, which effectively filters in those countries that belong to either Europe or Asia.

Use the schema given in Figure 3 to answer the questions 4 to 6.

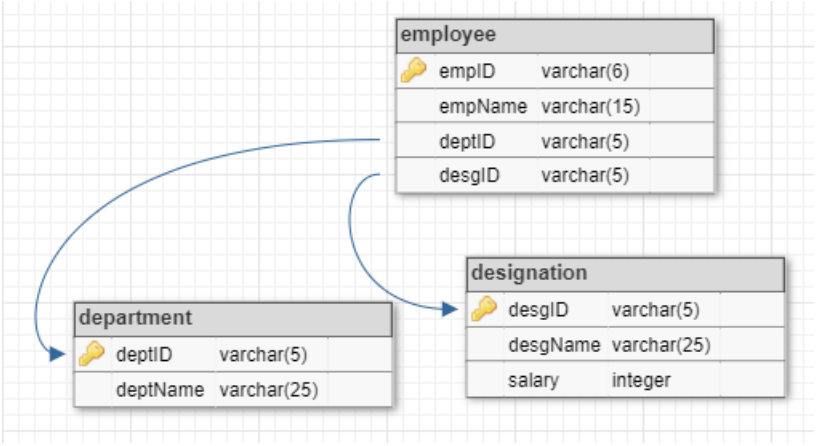

Figure 3: Employee Schema

4. Find the names of employees whose salary is greater than that of every employee in the department 'Sales'. [MCQ: 2 points]

◯
```
SELECT empName FROM employee E NATURAL JOIN designation D
WHERE E.deptID = (SELECT deptID from department, designation
WHERE deptName = 'Sales'
AND salary > ALL (SELECT salary
FROM designation D, department T, employee E
WHERE E.desgID = D.desgID AND T.deptID = E.deptID));
```

◯
```
SELECT empName FROM employee E NATURAL JOIN designation D
WHERE D.salary >= ALL (SELECT salary
FROM designation D, department T, employee E
WHERE E.desgID = D.desgID AND T.deptID = E.deptID
AND T.deptName = 'Sales');
```

◯
```
SELECT empName FROM employee E NATURAL JOIN designation D
WHERE D.salary > SOME (SELECT salary
FROM designation D, department T, employee E
WHERE E.desgID = D.desgID AND T.deptID = E.deptID
AND T.deptName = 'Sales');
```

√
```
SELECT empName FROM employee E NATURAL JOIN designation D
WHERE D.salary > ALL (SELECT salary
FROM designation D, department T, employee E
WHERE E.desgID = D.desgID AND T.deptID = E.deptID
AND T.deptName = 'Sales');
```

Solution: In option 1, the department name is mentioned in the first inner query. So, the salary comparison happens between employees of Sales only.
In option 2, $\geq$ ALL will filter in values that are at least as large as the salary values returned by the inner query.
In option 3, $>$ SOME will filter in all values that are larger than some salary values returned by the inner query.
In option 4, $>$ ALL returns the only value that is larger than all values returned by the inner query. Hence, option 4 is correct.

5. What should be filled in the blanks to find the designation of employees with lowest salary in the Purchase department? [MCQ: 2 points]

```
SELECT DISTINCT desgName FROM designation D1, department T1, employee E1
WHERE E1.desgID = D1.desgID
        AND E1.deptID = T1.deptID
        AND deptName = 'Purchase'
        AND salary ____ (SELECT ____(salary)
                FROM designation D2, department T2, employee E2
                WHERE E2.desgID = D2.desgID
                        AND E2.deptID = T2.deptID
                        AND T2.deptName = 'Purchase');
```

- ◯ $<$, MIN
- √ $=$, MIN
- ◯ $<$ ALL, MAX
- ◯ $\leq$ ALL, MAX

Solution: In order to obtain the least salary within a department, we use $=$ `MIN(salary)` within the department.

6. Based on the relations given in Figure 4, answer the question that follows.

employee

empID	empName	deptID	desgID
E00001	Akash	D0002	G0001
E00002	Akshay	D0002	----(a)----
E00003	Subha	D0003	G0003
E00004	Lavanya	---(b)---	G0002
E00005	Diya	D0001	G0001

department

deptID	deptName
D0001	Purchase
D0002	Sales
D0003	Accounts

designation

desgID	desgName	Salary
G0001	Clerk	5000
G0002	Supervisor	7000
G0003	Manager	10000

Figure 4: Employee instance

What should be filled in blank (b) in the table `employee` in Figure 4, if the query given below returns the value: Lavanya? (Use CAPS for alphabets in the answer) [NAT: 2 points]

```
SELECT empName FROM employee WHERE desgID IN
        (SELECT desgID FROM designation WHERE salary =
                (SELECT MAX(salary)
                FROM employee E, designation G, department D
                WHERE E.deptID = D.deptID
                AND E.desgID = G.desgID
                AND deptName = 'Purchase'))
        AND deptID IN
        (SELECT deptID FROM department WHERE deptName = 'Purchase');
```

Answer: D0001

Solution: The value returned is Lavanya. If we look at the query, it is clear that it is looking for an employee of department 'Purchase' with maximum salary. Clearly, the deptID should be filled by the deptID of the department 'Purchase', which is 'D0001' from the table `department`.

7. Consider the relation **multiple** that stores tuples of the form (a, b) where a is a multiple of b: [MCQ: 2 points]

```
CREATE TABLE multiple(
    first INTEGER,
    second INTEGER,
    PRIMARY KEY (first),
    FOREIGN KEY (second) REFERENCES multiple
            ON DELETE CASCADE);
```

If a tuple (a, b) is deleted from `multiple`, then which of the following is possible?

- ◯ a tuple (c, d) such that c is a factor of a is deleted.
- √ a tuple (c, d) such that c is a multiple of a is deleted.
- ◯ a tuple (c, d) such that d is a factor of a is deleted.
- ◯ a tuple (c, d) such that d is a factor of b is deleted.

Solution: The foreign key has been created with `ON DELETE CASCADE`. When a tuple (a, b) is deleted from `multiple`, any tuple of the form (e, a) will be deleted. This leads to any tuple of the form (d, e) to be deleted, and further any tuple of the form (c, d) to be deleted. By the property of being a multiple, observe that a is a multiple of b, e is a multiple of a and b, d is a multiple of e, a and b, and c is a multiple of d, e, a and b and so on. Thus, a tuple (c, d) such that c is a multiple of a is deleted.

8. Consider the relational schema given in Figure 5.

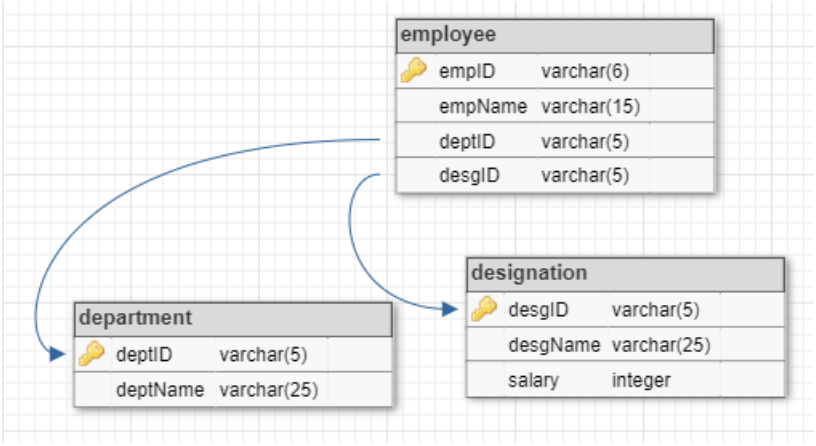

Figure 5: Employee Schema

If the relations **employee, designation** and **department** have 100, 6, 5 rows respectively, what is the maximum number of rows returned by the following query?

[NAT: 2 points]

```
SELECT * FROM employee NATURAL JOIN designation;
```

Answer: 100

Solution: `desgID` is the foreign key in table `employee` that references `designation` table. It follows that the `desgID` in any row of `employee` table will have a corresponding entry in the `designation` table. However the converse is not always true. That is, corresponding to each `desgID` in the `designation` table, there need not be an entry in the `employee` table. Hence there can only be as many rows in the natural join as the number of rows in the `employee` table (since natural join looks for same value of the common attribute). Thus, the maximum number of rows in the natural join in the given example is 100.

9. Consider the table **employee** and table **department** as shown in Figure 6 and answer the question that follows. [MCQ: 2 points]

employee

emp_name	emp_id	age	dept_id
WADE	1	23	10
MADDEN	4	54	10
HARM	6	34	13
TALLY	3	41	16
RODEY	2	46	14
JONES	7	38	14
MULE	5	49	16

department

dept_name	dept_id	dept_location
MATHS	10	Houston
ENGLISH	15	San Antonio
PHYSICS	14	Houston
COMPUTER	13	New York
CHEMISTRY	16	Chicago

Figure 6: employee & department

What will be the output of the following query?

```
SELECT emp_name, dept_name, dept_location FROM employee JOIN department
ON employee.dept_id = department.dept_id
WHERE emp_id IN (6,4,2,7,5) AND age<>54 ORDER BY age asc;
```

◯ Output:

emp_name	dept_name	dept_location
HARM	COMPUTER	New York
TALLY	CHEMISTRY	Chicago
RODEY	PHYSICS	Houston
MULE	CHEMISTRY	Chicago

◯ Output:

emp_name	dept_name	dept_location
HARM	COMPUTER	New York
RODEY	PHYSICS	Houston
MULE	CHEMISTRY	Chicago
JONES	PHYSICS	Houston

◯ Output:

emp_name	dept_name	dept_location
WADE	MATHS	Houston
TALLY	CHEMISTRY	Chicago
RODEY	PHYSICS	Houston
MULE	CHEMISTRY	Chicago

√ Output:

emp_name	dept_name	dept_location
HARM	COMPUTER	New York
JONES	PHYSICS	Houston
RODEY	PHYSICS	Houston
MULE	CHEMISTRY	Chicago

Solution: When table employee and table department are joined on employee.dept_id= department.dept_id, it will fetch the following result -

emp_name	dept_name	dept_location
WADE	MATHS	Houston
MADDEN	MATHS	Houston
HARM	COMPUTER	New York
TALLY	CHEMISTRY	Chicago
RODEY	PHYSICS	Houston
JONES	PHYSICS	Houston
MULE	CHEMISTRY	Chicago

Now as per the query, emp_id should be IN (6,4,2,7,5) AND age must not be 54 and it will fetch the following result -

emp_name	dept_name	dept_location
RODEY	PHYSICS	Houston
MULE	CHEMISTRY	Chicago
HARM	COMPUTER	New York
JONES	PHYSICS	Houston

Now as and when we put ORDER BY age in ascending order, it will fetch the following result -

emp_name	dept_name	dept_location
HARM	COMPUTER	New York
JONES	PHYSICS	Houston
RODEY	PHYSICS	Houston
MULE	CHEMISTRY	Chicago

10. Consider table **Employee**(*eid, edept, ename, esalary, ebonus*) and the code given below.

[MCQ:2 points]

```
CREATE OR REPLACE FUNCTION bonus_fun() RETURNS TRIGGER AS $$
BEGIN
      IF NEW.edept = 'R/D' THEN
             NEW.ebonus = NEW.esalary * .75;
      END IF;
      RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER bonus_trig
    BEFORE INSERT ON Employee
    FOR EACH ROW
    EXECUTE PROCEDURE bonus_fun();

INSERT INTO Employee VALUES (4,'R/D','Diksha',30000);
INSERT INTO Employee VALUES (2,'Accounts','Raj',40000);
SELECT ebonus FROM Employee;
```

What will be the output of the given code?

- ◯ 22500
  0
- √ 22500
  NULL
- ◯ 22500
  30000
- ◯ The code has errors.

Solution: The trigger is executed for the first insert statement since it satisfies the condition described in *bonus_fun*. Accordingly the *ebonus* attribute of the first entry is set to 75% of the *esalary*. For the second insert the trigger condition is not satisfied and hence its *ebonus* is not set therefore it will remain NULL.