CS451 Database Systems

School of Electrical Engineering and Computer Science
Washington State University

Sample Midterm 2
Solution

PART 1 – True/False and Multiple Choice Questions

(12pts) Question 1: TRUE/FALSE

For each of the following statements, indicate whether it is TRUE or FALSE by circling your choice.

1. In SQL, an INSERT statement always requires all column values to be specified.

   TRUE / **FALSE**

2. Given an SQL query, there are often multiple ways of writing it in relational algebra.

   TRUE / FALSE

3. Given a relation that is not in BCNF there is always a unique decomposition into relations that are in BCNF.

   TRUE / **FALSE**

4. If A → BC and B→ D hold on relation R(A,B,C,D) then A → CD also holds on R.

   TRUE / FALSE

(5pts) Multiple choice

Assume the schemas of the relations are R(a, b) and S(b, c). R's primary key is (a,b) and S's primary key is (b,c). We are using the bag semantics for query results, i.e., the result may have duplicates.

Q1:

```
SELECT a
FROM R, S
WHERE R.b = S.b;
```

Q2:

```
SELECT a
FROM R
WHERE b IN
     (SELECT b FROM S);
```

a) Q1 and Q2 produce the same answer.
b) The answer to Q1 is always contained in the answer to Q2.
c) The answer to Q2 is always contained in the answer to Q1.
d) Q1 and Q2 always produce different answers.

Answer: (c)

Both queries will match the R.b values to S.b values and return the R.a values in those matched tuples . If a R tuple is matched with more than one tuple in S, then the same R.a value will appear multiple times in the result of Q1 (as much as the number of matching S tuples). Since we are using bag semantics the duplicates won't be eliminated in the result.

However, the result of Q2 will not repeat the R.a values for each matching tuple in S.

(6pts) TRUE / FALSE

Consider the following relation R(A,B,C,D)
An instance of R is given below.

A	B	C	D
1	3	2	2
2	3	2	4
3	1	3	6
3	1	1	12

Consider the query:

```
SELECT  A, B
FROM  R
WHERE  C >
     (SELECT D from R where A = 3)
```

Claim: The above query will run successfully on R.

If TRUE, give output.
If FALSE, explain why.

Answer:

FALSE
The inner query (`select D from R where A = 3`) returns a bag of values. We cannot directly perform "less than" comparison between a bag of values and a single atomic value.

(6pts) TRUE / FALSE

Consider the following relations:

```
Emp (eid, ename, salary)
Project (pid, title, budget)
Works (eid, pid, year)
```

- The primary keys are underlined.
- `Works.eid` and `Works.pid` are foreign keys to Emp and Project, respectively
- An employee may work on multiple projects during the same year, and may also work on the same project during multiple years.
- Assume bag semantics, i.e., duplicates are not eliminated.

```sql
SELECT DISTINCT E.ename
FROM Emp as E, Works as W, Works as W2
WHERE E.eid = W.eid AND E.eid = W2.eid
    AND W.pid = W2.pid
    AND W.year < '2016';
```

```sql
SELECT DISTINCT E.ename
FROM Emp as E, Works W
WHERE E.eid = W.eid
    and W.year < '2016';
```

Indicate whether the above two SQL queries are equivalent. Assume that the database <u>does not contain any NULL values.</u>

TRUE

Assertion and Triggers

Consider the following relational schema. An employee can work in more than one department; the "*pct_time*" field of the Works relation shows the percentage time that a given employee works in a given department.

Emp (***eid***: integer, ***ename***: string, ***age***: integer, ***salary***: real)
Works (***eid***: integer, ***did***: integer, ***pct time***: integer)
Dept (***did***: integer, ***budget***: real, ***managerid***: integer)

Primary keys are underlined.
"eid" and "did" in "Works" are foreign keys referencing "eid" and "did" in "Emp" and "Dept", respectively.
"managerid" in "Dept" is a foreign key referencing "eid" in Emp.

Answer the following.

1. **(4pts)** Define an attribute-based CHECK constraint on Emp which will ensure that every employee makes at least $10,000. *(Add the constraint to the Emp table below.)*

```
CREATE TABLE Emp (
     eid INTEGER PRIMARY KEY,
     ename CHAR(20),
     age INTEGER,
     salary REAL CHECK(salary > 10000)
);
```

2. **(7pts)** Assume you removed the foreign key constraint for *"managerid" in "Dept"*. Enforce the following constraint using an assertion. *"Every manager must also be an employee (i.e., every managerid should be a valid eid.).*" (Part of the assertion is provided below – just complete the assertion condition.)

```
CREATE ASSERTION MgrIsEmp CHECK
( NOT EXISTS
     (SELECT *
      FROM Dept D
      WHERE D.managerid NOT IN
          (SELECT eid FROM Emp))
 )
```

3. **(8pts)** Consider the following PostgreSQL trigger. Explain the constraint that this trigger implements. Make sure to specify the event, condition, and action. (Please don't explain the details of the SQL code, just state the constraint and how the trigger handle the violation of that constraint.)

```
CREATE OR REPLACE FUNCTION myAction() RETURNS trigger AS '
BEGIN
  UPDATE Emp
  SET    salary = NEW.salary
  WHERE  salary < NEW.salary AND
         eid IN (SELECT Dept.managerid
                 FROM Works, Dept
                 WHERE Works.eid = NEW.eid AND Works.did = Dept.did);
  RETURN NEW;
END
' LANGUAGE plpgsql;

CREATE TRIGGER myTrigger
AFTER UPDATE ON Emp
FOR EACH ROW
WHEN (OLD.salary < NEW.salary)
EXECUTE PROCEDURE myAction();
```

Answer:

"Whenever an employee is given a raise, the manager's salary must be increased to be at least as much."

Event: When the Employee table is updated
Condition: If the employee is given a raise (salary is increased)
Action: The salary of that employee's manager must be increased to be at least as much (if the manager's salary is less than the employee's raised salary).

SQL

(30pts) Consider the following schema:

Suppliers(<u>**sid**: integer</u>, **sname**: string, **address**: string)
Parts(<u>**pid**: integer</u>, **pname**: string, **color**: string)
Catalog(<u>**sid**: integer, **pid**: integer</u>, **cost**: real)

The *Catalog* relation lists the prices charged for *Parts* by *Suppliers*.

The primary keys are underlined. In Catalog, "*sid*" is a foreign key referencing "*sid*" in Suppliers, and "*pid*" is a foreign key referencing "*pid*" in "Parts".

Based on the schema above, write the following queries in SQL:

(a) **(10pts)** Find the suppliers that supply at least 3 green parts each of which costs less than $50. Return *sid* and *sname* of the supplier.

Answer:

```
SELECT sid,sname
FROM Supplier as S, Catalog as C, Parts as P
WHERE S.sid=C.sid AND C.pid=P.pid AND
      P.color='Green' AND C.cost<50
GROUP BY sid,sname
HAVING count(C.pid)>=3;
```

(b) **(10pts)** Find the distinct "*sids*" of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

Answer:

```
SELECT DISTINCT C.sid
FROM Catalog C
WHERE C.cost > ( SELECT AVG (C1.cost)
                 FROM Catalog C1
                 WHERE C1.pid = C.pid )
```

Suppliers(**sid**: integer, **sname**: string, **address**: string)
Parts(**pid**: integer, **pname**: string, **color**: string)
Catalog(**sid**: integer, **pid**: integer, **cost**: real)

(c) **(10pts)** Find the distinct "*sids*" of suppliers who supply only red parts.

Answer:
```
SELECT DISTINCT S.sid
FROM Supplier S
WHERE NOT EXISTS
     (SELECT *
      FROM Catalog C, Parts P
      WHERE C.pid = P.pid AND
            P2.color <> 'Red' AND
            S.sid = C.sid )
```

BCNF

(22pts)

Consider a relation R(A,B,C,D,E), with FDs
C → ABD,
D → E

(8pts) (a) List all the minimal keys of R. Do not list superkeys which are not (minimal) keys.

Solution: Key is C

Since C doesn't appear on the right-hand-side of any FD, it can't be derived from other attributes, and therefor C needs to be part of every minimal key. So we should only check the closure of the subsets that include C.

{C}+ = {A,B,C,D,E} ---- key

Since C is a key and since all minimal keys need to include C, there are no other minimal keys in R. All other keys will be super-keys.

(14pts) (b) Is this relation in BCNF? If you answer is yes, explain why it is. If you answer is no, decompose the relation into BCNF, showing your decomposition steps.

No. The second FD (D → E) violates BCNF. We decompose R into R1(DE) and R2(ABCD).

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

apply D->E

R1(D,E)
FDs: D -> E
Key : D
In BCNF

R2(A,B,C,D)
FDs: C -> ABD
Key : C
In BCNF

FDs for R1:
E doesn't appear on the left-hand-side of any FD, so the subsets that include E won't drive new attributes. And therefore, we will only check the closures of the remaining subsets:
{D}+ = {D , E } so D -> E

FDs for R2:
A and B don't appear on the left-hand-side of any FD, so the subsets that include AB won't drive new attributes. And therefore, we will only check the closures of the remaining subsets:
{C}+ = {A , B , ~~C~~, D , ~~E~~} so C -> ABD
{D}+ = {D , ~~E~~}
{CD}+ = {A , B , ~~C~~, D , ~~E~~} so CD -> ABD (redundant)

R1 is in BCNF because the only relevant FD, D → E, does not violate BCNF (and because its a two-attribute relation).
R2 has the relevant FDs C → ABD which doesn't violate BCNF (C is a key for R2). Therefore we are done.
We decompose R into R1(DE) with FD "D->E" and
R2(ABCD) with FD "C->ABD".