

**CMPT-354 D1 Fall 2008**  
**Instructor: Martin Ester**  
**TA: Gustavo Frigo**

**Sample Midterm Exam with Solution**

Time: 60 minutes

**Problem 1 (Queries in relational algebra and SQL)**

Consider the following schema of a *product database*:

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

The Catalog records that some Supplier sid supplies Part pid at a given price.

Formulate each of the following queries in relational algebra (RA) and in SQL. Name all attributes in the result appropriately. If the query cannot be explained in a language, state this and explain why not.

- a) For every green Part that is supplied by some Supplier, list the number of these Suppliers and the average price in descending order of that number.

The query cannot be formulated in RA, since RA has no group-by and aggregation operations.

```
SELECT COUNT (sid) AS numberSuppliers, AVG(price) AS avgPrice
FROM Catalog C, Parts P
WHERE C.pid = P.pid AND P.color = 'green'
GROUP BY C.pid
ORDER BY numberSuppliers DESC;
```

- b) Find the pids of Parts that are supplied by all Suppliers or are not supplied by any Supplier.

$$r_{R1}((\mathbf{p}_{pid, sid} Catalog) / (\mathbf{p}_{sid} Suppliers))$$
$$r_{R2}(\mathbf{p}_{pid} Parts - \mathbf{p}_{pid} (Parts \infty Catalog))$$
$$R1 \cup R2$$

```

(SELECT pid
FROM Parts P
WHERE NOT EXISTS
  (SELECT sid
   FROM Suppliers S
   WHERE NOT EXISTS
     (SELECT *
      FROM Catalog
      WHERE C.pid = P.pid AND C.sid = S.sid)))
UNION
(SELECT pid
FROM Parts P
WHERE NOT EXISTS
  (SELECT Catalog C
   WHERE C.pid = P.pid))

```

- c) Find the pids of Parts that are supplied by at least three Suppliers.

$R1(1 \rightarrow sid1, 2 \rightarrow pid1, 3 \rightarrow price1, 4 \rightarrow sid2, 5 \rightarrow pid2, 6 \rightarrow price2, 7 \rightarrow sid3, 8 \rightarrow pid3, 9 \rightarrow price3)$  (Catalog  $\times$  Catalog  $\times$  Catalog)  
 $P_{pid1}(S_{pid1=pid2 \text{ AND } pid1=pid3 \text{ AND } pid2=pid3 \text{ AND } sid1 \neq sid2 \text{ AND } sid1 \neq sid3 \text{ AND } sid2 \neq sid3} R1)$

```

SELECT P.pid
FROM Parts P
WHERE 2 <
  (SELECT COUNT (*)
   FROM Catalog C
   WHERE C.pid = P.pid);

```

## Problem 2 (SQL assertions)

Consider again the above schema of a product database.

Formulate each of the following integrity constraints as an SQL assertion:

- a) No Supplier may supply red and green Parts.

```

CREATE ASSERTION NoRedAndGreenParts CHECK
(NOT EXISTS
  (
    (SELECT C.sid
     FROM Catalog C, Parts P
     WHERE C.pid = P.pid AND P.color = 'red')
    INTERSECTS
    (SELECT C.sid
     FROM Catalog C, Parts P
     WHERE C.pid = P.pid AND P.color = 'green')
  )
);

```

b) For all Parts, no other Supplier has a lower price than the Supplier with sid = 1.

```
CREATE ASSERTION NoLowerPriceThanSid1 CHECK
  (NOT EXISTS
    (SELECT *
      FROM Catalog C1, C2
      WHERE C1.pid = C2.pid AND C2.sid = 1 AND C1.price < C2.price)
  );
```

### **Problem 3 (SQL triggers)**

Consider again the above schema of a product database.

Formulate the following integrity constraint as a set of SQL triggers: No Supplier may supply red and green Parts.

Make sure that you formulate one trigger for each of the DB modifications that can potentially violate the integrity constraint. Your trigger(s) should explicitly undo the database modification that violated the integrity constraint.

```
CREATE TRIGGER NoRedAndGreenParts1
  AFTER INSERT ON Catalog
  REFERENCING NEW ROW AS NewTuple
  FOR EACH ROW
  WHEN
    (SELECT C.sid
     FROM Catalog C, Parts P
     WHERE C.pid = P.pid AND P.color = "red")
    INTERSECTS
    (SELECT C.sid
     FROM Catalog C, Parts P
     WHERE C.pid = P.pid AND P.color = "green")
  DELETE FROM Catalog
  WHERE pid = NewTuple.pid AND sid = NewTuple.sid;
```

To improve the efficiency of this trigger, you could add a condition  $C.sid = NewTuple.sid$  to both SELECT statements.

```

CREATE TRIGGER NoRedAndGreenParts2
AFTER UPDATE ON Parts
REFERENCING NEW ROW AS NewTuple
                OLD ROW AS OldTuple
FOR EACH ROW
WHEN
    (
        (SELECT C.sid
         FROM Catalog C, Parts P
         WHERE C.pid = P.pid AND P.color = "red")
        INTERSECTS
        (SELECT C.sid
         FROM Catalog C, Parts P
         WHERE C.pid = P.pid AND P.color = "green")
    )
UPDATE Parts
    SET color = (SELECT color FROM OldTuple)
    WHERE pid = OldTuple.pid;

```

Alternatively, the action of trigger NoRedAndGreenParts2 could be formulated as follows:

```

BEGIN
    DELETE FROM Parts
    WHERE pid = OldTuple.pid;
    INSERT INTO Parts (SELECT * FROM OldTuple);
END

```