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

Midterm Exam with Solution

Time: 50 minutes

Total marks: 100

Problem 1 (Queries in relational algebra and SQL) (40 marks)

Consider the following schema of a *computer database*:

> Product(<u>model: string</u>, maker: string)
> PC(<u>model: string</u>, speed: float, ram: float, hd: int, price: float)
> Laptop(<u>model: string</u>, speed: float, ram: float, hd: int, screen: int, price: float)
> Printer(<u>model: string</u>, color: string, type: string, price: float)

A Product is either a PC, a Laptop or a Printer and must have a tuple in the corresponding table. There is a foreign key constraint on the model of PCs, Laptops and Printers referencing the primary key model of Product. You can assume that the maker attribute uniquely identifies the manufacturer of a Product. Assume that all of the non-key attributes allow NULL values.

Formulate each of the following queries in relational algebra (RA) and in SQL. Make sure that you do not return duplicate answers. If the query cannot be explained in a language, state this and explain why not.

a) Find the manufacturers (i.e. makers of Products) that make Laptops, but not Printers. (20 marks)

$$(p_{\text{maker}}(Product \infty\ Laptop)) - (p_{\text{maker}}(Product \infty\ Printer))$$

```
(SELECT (DISTINCT P.maker)
FROM Product P, Laptop L
WHERE P.model = L.model)
EXCEPT
(SELECT (DISTINCT P.maker)
FROM Product P, Printer Print
WHERE P.model = Print.model);
```

b) Find the manufacturers that make at least two different models of Laptops. (20 marks)

$$r_{R1}(p_{\text{model, maker}}(Laptop \infty Product))$$
$$r_{R2(1\rightarrow \text{model1}, 2\rightarrow \text{maker1}, 3\rightarrow \text{model2}, 4\rightarrow \text{maker2})}(R1 \times R1)$$
$$p_{\text{maker1}}(s_{\text{maker1}=\text{maker2 } AND \ \text{model1}\neq\text{model2}}\ R2))$$

```
SELECT (DISTINCT P1.maker)
FROM Product P1, Product P2, Laptop L1, Laptop L2
WHERE P1.model = L1.model AND P2.model = L2.model AND
          P1.maker = P2.maker AND P1.model <> P2.model;
```

Problem 2 (SQL assertions) (30 marks)

Consider again the following schema of a *computer database*:

> Product(<u>model: string</u>, maker: string)
> PC(model: string, speed: float, ram: float, hd: int, price: float)
> Laptop(<u>model: string</u>, speed: float, ram: float, hd: int, screen: int, price: float)
> Printer(<u>model: string</u>, color: string, type: string, price: float)

A Product is either a PC, a Laptop or a Printer and must have a tuple in the corresponding table. There is a foreign key constraint on the model of PCs, Laptops and Printers referencing the primary key model of Product. You can assume that the maker attribute uniquely identifies the manufacturer of a Product. Assume that all of the non-key attributes allow NULL values.

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

a) No manufacturer may make PCs and Laptops. (15 marks)

```
CREATE ASSERTION NoPCsAndLaptops CHECK
    (NOT EXISTS
             (SELECT P.maker
             FROM PC P, Product Prod
             WHERE P.model = Prod.model)
           INTERSECTS
             (SELECT P.maker
             FROM Laptop L, Product Prod
             WHERE L.model = Prod.model)
    );
```

b) A manufacturer of a PC must also make a Laptop with a speed at least as great as the PCs speed. (15 marks)

```
CREATE ASSERTION LaptopAtLeastAsFast CHECK
    (NOT EXISTS
            (SELECT *
            FROM PC P, Product Prod1
            WHERE P.model = Prod1.model AND NOT EXISTS
                    (SELECT *
                     FROM Laptop L, Product Prod2
                     WHERE L.model = Prod2.model AND Prod1.maker = Prod2.maker
                             AND L.speed >= P.speed))
    );
```

Problem 3 (SQL triggers) (30 marks)

Consider again the following schema of a *computer database*:

> Product(<u>model: string</u>, maker: string)
> PC(<u>model: string</u>, speed: float, ram: float, hd: int, price: float)
> Laptop(<u>model: string</u>, speed: float, ram: float, hd: int, screen: int, price: float)
> Printer(<u>model: string</u>, color: string, type: string, price: float)

A Product is either a PC, a Laptop or a Printer and must have a tuple in the corresponding table. There is a foreign key constraint on the model of PCs, Laptops and Printers referencing the primary key model of Product. You can assume that the maker attribute uniquely identifies the manufacturer of a Product. Assume that all of the non-key attributes allow NULL values.

Formulate the following integrity constraint as a set of SQL triggers: For every PC, there must be a Product with the same model.

Make sure that you formulate one trigger for each of the DB modifications that can potentially violate the integrity constraint. Your trigger(s) should not undo the database modification that violated the integrity constraint, but perform another modification of the DB that leads to a consistent DB state.

```
CREATE TRIGGER ProductForPC
   AFTER INSERT ON PC
   REFERENCING NEW ROW AS NewPC
   FOR EACH ROW
   WHEN
          (NOT EXISTS
                (SELECT *
                 FROM Products P
                 WHERE P.model = NewPC.model))
    INSERT INTO Product (model)
          SELECT model FROM NewPC;


CREATE TRIGGER NoPCWithoutProduct
   AFTER DELETE ON Product
   REFERENCING NEW ROW AS OldProduct
   FOR EACH ROW
   WHEN
                (SELECT *
                 FROM PC P
                 WHERE P.model = OldProduct.model)
    DELETE FROM PC WHERE model = OldProduct.model;
```