

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

Final Exam with Solution

Time: 180 minutes
Total marks: 300

Problem 1 (Queries in RA and SQL) (100 marks)

Consider the following schema of a *product database*:

Part(pid: integer, pname: string, color: string)
Supplier(sid: integer, sname: string, city: string, address: string)
Catalog(sid: integer, pid: integer, price: real)

The *Catalog* records that some *Supplier* sid supplies *Part* pid at a given price. Primary keys are underlined. There are foreign key constraints on Catalog.sid referencing Supplier and Catalog.pid referencing Part.

Formulate each of the following queries in relational algebra and in SQL. If a query cannot be formulated in one of the languages, state that and explain the reason. If necessary, eliminate duplicate results.

- a) For every green Part, find the pid, the cheapest price and the number of its Suppliers.

This query cannot be formulated in relational algebra, since it has no group-by and no aggregation operators.

```
SELECT P.pid, MIN(C.price), COUNT(C.sid)
FROM Part P, Catalog C
WHERE P.color = green AND P.pid = C.pid
GROUP BY P.pid;
```

- b) Find the pids of Parts that are supplied by every (!) Supplier in Vancouver and are supplied by some (!) Supplier in Burnaby.

$$r_{R1}(\mathbf{p}_{pid,sid}(Catalog) / (\mathbf{p}_{sid}(\mathbf{s}_{city=Vancouver}(Supplier))))$$
$$r_{R2}(\mathbf{p}_{pid}(Catalog \bowtie (\mathbf{s}_{city=Burnaby}(Supplier))))$$
$$R1 \cap R2$$

```
SELECT DISTINCT P.pid
FROM Part P, Catalog C, Supplier S
WHERE P.pid = C.pid AND C.sid = S.sid AND S.city = Burnaby
AND NOT EXISTS
    (SELECT S2.sid
     FROM Supplier S2)
```

```

WHERE S2.city = Vancouver AND
      (NOT EXISTS
        (SELECT *
         FROM Catalog C2
         WHERE C2.sid = S2.sid AND C2.pid = P.pid));

```

- c) Find the sid of Suppliers that supply exactly (!) one red Part.

```

rR1(Scolor=redPart)
rR2(1→sid,2→n,3→c,4→a,5→s,6→pid,7→p,8→pi,9→na,10→co)(Supplier×Catalog×R1)
rR3(psid1,pid1(R1))
rR4(1→sid,2→pid,3→sid2,4→pid2)(R2×R2)
rR5(Ssid1=sid2∧pid1≠pid2R2)
psid1R3 - psid1R5

SELECT S.sid
FROM Supplier S
WHERE 1 =
      (SELECT COUNT(P.pid)
       FROM Catalog C, Part P
       WHERE C.sid = S.sid AND C.pid = P.pid AND P.color = "red");

```

Problem 2 (SQL assertions and triggers) (50 marks)

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

```

Part(pid: integer, pname: string, color: string)
Supplier(sid: integer, sname: string, city: string, address: string)
Catalog(sid: integer, pid: integer, price: real)

```

The *Catalog* records that some *Supplier* sid supplies *Part* pid at a given price. Primary keys are underlined. There are foreign key constraints on *Catalog*.sid referencing *Supplier* and *Catalog*.pid referencing *Part*.

- a) Formulate the following integrity constraint as an SQL assertion: Suppliers in the same city cannot charge a different price for the same Part.

```

CREATE ASSERTION NoDifferentPricesInSameCity CHECK
(NOT EXISTS
  (SELECT *
   FROM Catalog C1, Catalog C2, Supplier S1, Supplier S2
   WHERE C1.sid = S1.sid AND C2.sid = S2.sid
        AND C1.pid = C2.pid AND S1.city = S2.city AND C1.price <> C2.price)
);

```

- b) In order to formulate the integrity constraint from b) with triggers, you have to create several triggers. Show the corresponding trigger for insertions on the *Catalog* table. Your trigger should not use the ROLLBACK statement to undo a database modification that violated the integrity constraint, but perform another modification of the DB that restores a consistent DB state.

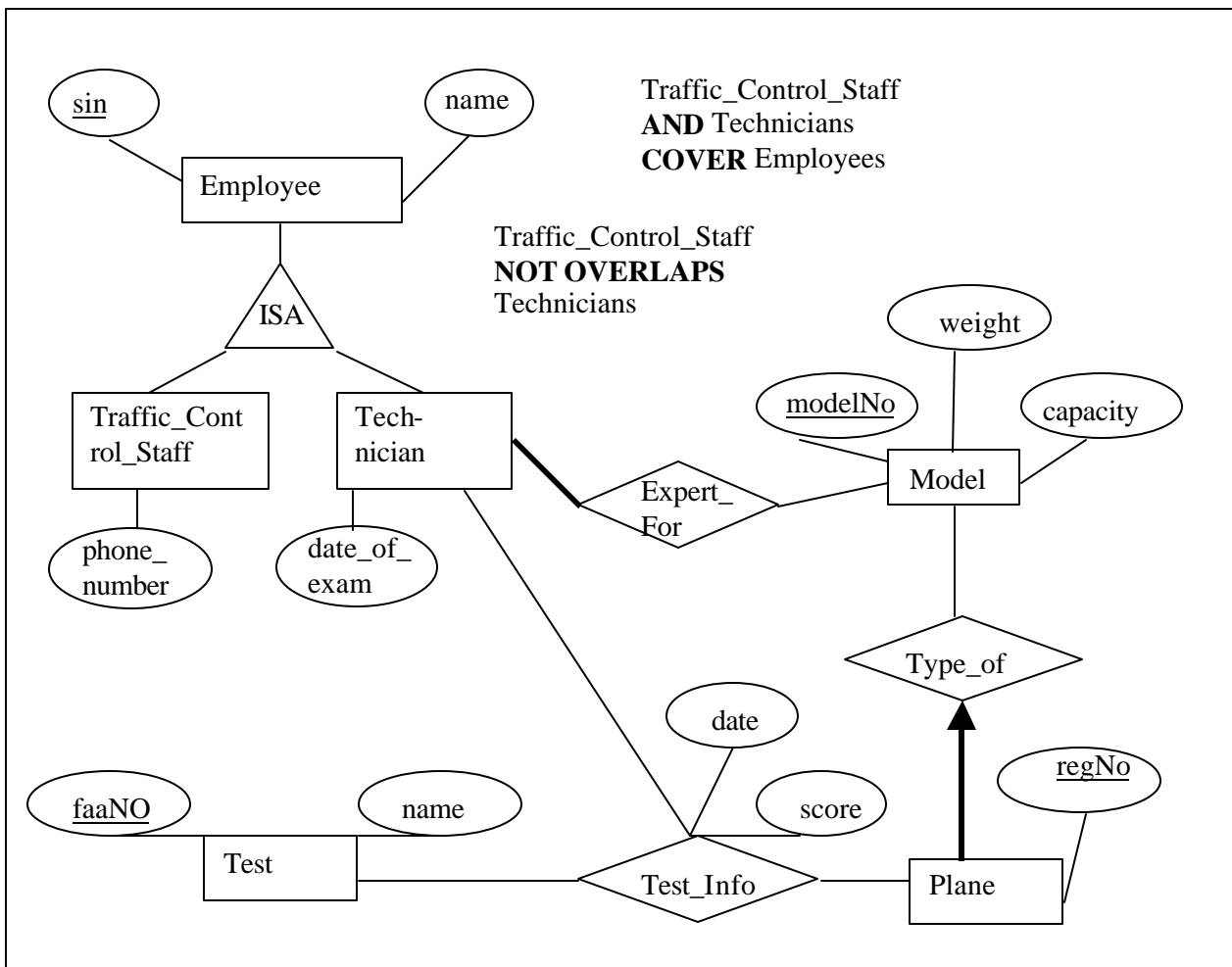
```

CREATE TRIGGER NoDifferentPricesInSameCity
AFTER INSERT ON Catalog
REFERENCING NEW ROW AS N
FOR EACH ROW
WHEN
  (EXISTS
    (SELECT *
     FROM Catalog C, N, Supplier S1, Supplier S2
     WHERE N.sid = S1.sid AND C.sid = S2.sid AND S1.sid <> S2.sid
           AND N.pid = C.pid AND S1.city = S2.city AND N.price <> C.price)
  )
DELETE FROM Catalog WHERE pid = N.pid AND sid = N.sid;

```

Problem 3 (Translation of ER diagram) (50 marks)

Translate the following *ER schema* of an *airport database* into an equivalent relational schema.



a) Write down the complete SQL statements to create the relational schema, including PRIMARY KEY, FOREIGN KEY and NOT NULL constraints. For the FOREIGN KEY constraints, you do not need to specify the reactions to violating updates.

Your relational schema should satisfy the following two design criteria:

- The number of tables should be minimal.
- As many integrity constraints from the ER diagram as possible should be expressed.

```
CREATE TABLE Traffic_Control_Staff (sin VARCHAR(10),
                                     name VARCHAR (20),
                                     phone_number VARCHAR (20),
                                     PRIMARY KEY (sin));
```

```
CREATE TABLE Technician ( sin VARCHAR(10),
                             name VARCHAR (20),
                             date_of_exam DATETIME),
                             PRIMARY KEY (sin));
```

```
CREATE TABLE Model ( modelNo VARCHAR(10),
                       weight INTEGER,
                       capacity INTEGER,
                       PRIMARY KEY (modelNo))
```

```
CREATE TABLE Plane ( regNo VARCHAR(10),
                       modelNo VARCHAR(10) NOT NULL,
                       PRIMARY KEY (regNo),
                       FOREIGN KEY (modelNo) REFERENCES Model)
```

```
CREATE TABLE Expert_For ( sin VARCHAR(10),
                             modelNo VARCHAR(10),
                             PRIMARY KEY (sin, modelNo),
                             FOREIGN KEY (sin) REFERENCES Technician,
                             FOREIGN KEY (modelNo) REFERENCES Model)
```

```
CREATE TABLE Test ( faaNo VARCHAR(10),
                      name VARCHAR(20),
                      PRIMARY KEY (faaNo))
```

```
CREATE TABLE Test_Info ( faaNo VARCHAR(10),
                           regNo VARCHAR(10),
                           sin VARCHAR(10),
                           date DATETIME,
                           score INTEGER,
                           PRIMARY KEY (faaNo, regNo, sin),
                           FOREIGN KEY (faaNo) REFERENCES Test,
                           FOREIGN KEY (regNo) REFERENCES Plane,
                           FOREIGN KEY (sin) REFERENCES Technician)
```

b) List the integrity constraints of the ER diagram that are not expressed in your relational schema.

The following integrity constraints are not expressed in the above relational schema:

- Participation constraint on Technician in relationship set Expert_For.
- NOT OVERLAPS constraint on Technician and Traffic_Control_Staff.

Problem 4 (Normalization) (50 marks)

Consider a table R with the following set of attributes and set of functional dependencies:

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

$AB \rightarrow C,$
 $C \rightarrow D,$
 $D \rightarrow B,$ and
 $D \rightarrow E.$

Design a sequence of (binary) decompositions with a minimal number of such decompositions that transforms the table R into a set of tables that are all in BCNF.

For every decomposition,

- state the input table to be decomposed,
- list the primary key of the input table and explain why it is a key,
- list the FD $\{A_1, \dots, A_n\} \rightarrow \{B_1, \dots, B_m\}$ that violates BCNF,
- and list the attribute sets of the two tables resulting from the decomposition.

Finally, list the set of tables that result from your decomposition (and are all in BCNF).

Decomposition 1

Input is table R. The key of R is $\{A,B\}$. It is a superkey, since all attributes are in the closure $\{A,B\}^+$, and it is a minimal attribute set with this property, because no subset functionally determines C.

Violating FD: $C \rightarrow D$

Decompose R into $R_1(A,B,C)$ and $R_2(C,D,E)$.

Decomposition 2

Input is table R_2 . The key of R_2 is $\{C\}$, since all attributes are (indirectly) functionally determined by $\{C\}$, and no subset has the same property.

Violating FD: $D \rightarrow E$

Decompose R_2 into $R_3(C,D)$ and $R_4(D,E)$.

The final result of the decomposition of R is

- $R_1(A,B,C),$
- $R_3(C,D),$
- $R_4(D,E).$

Note that the FD $D \rightarrow B$, although it violates BCNF in the initial table R, does not require a separate decomposition, since decomposition 1 distributes D and B to two different tables.

Problem 5 (XML and XQuery) (50 marks)

Consider the following XML schema, stored in file “bibliography.xsd”.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="articletype" />
  <xs:complexType>
    <xs:sequence>
      <xs:element name="author" type="xs:string"
        maxOccurs="unbounded"/>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="year" type="xs:string"/>
      <xs:element name="url" type="xs:string" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="key" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>

  <xs:element name="booktype" />
  <xs:complexType>
    <xs:sequence>
      <xs:element name="author" type="xs:string"
        maxOccurs="unbounded"/>
      <xs:element name="title" type="xs:string" />
      <xs:element name="publisher" type="xs:string" />
      <xs:element name="year" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="key" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>

  <xs:element name="bibliography">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="article" type="articletype"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="book" type="booktype"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

- a) Create a valid XML document (according to the above XML schema) with one book and one article element. Assume that your XML document is stored in the same directory as the XML schema “bibliography.xsd”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bibliography
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="bibliography.xsd">

  <article key="100" >
    <author>Author1 </author>
    <author>Author2 </author>
```

```

    <title>Title1 </title>
    <year>Year1 </year>
</article>

<book key="200" >
    <author>Author3 </author>
    <title>Title2 </title>
    <publisher>Publisher1 </publisher>
    <year>Year2 </year>
</book>

</bibliography>

```

b) Consider an XML document “bibliography.xml” that conforms to the above XML schema. Formulate the following query on this XML document in XQuery.

For all books and articles published after 2000, return a result element containing the key, the title and the authors.

```

for $x in doc("bibliography.xml")/bibliography/*
where $x/year > 2000
return <result>
    {
        {$x/@key}
        {$x/title}
        {$x/author}
    }
</result>

```

c) Consider again an XML document “bibliography.xml” that conforms to the above XML schema. Formulate the following query on this XML document in XQuery.

For the oldest (those published in the earliest year) articles that have “data mining” in their title, return the title and the authors.

```

let $as := doc("bibliography.xml")/bibliography/article
    [contains(title,'data mining')]
    $m := min($a/year)
for $a in $as
where $a/year = $m
return
    $a/title
    $a/author

```