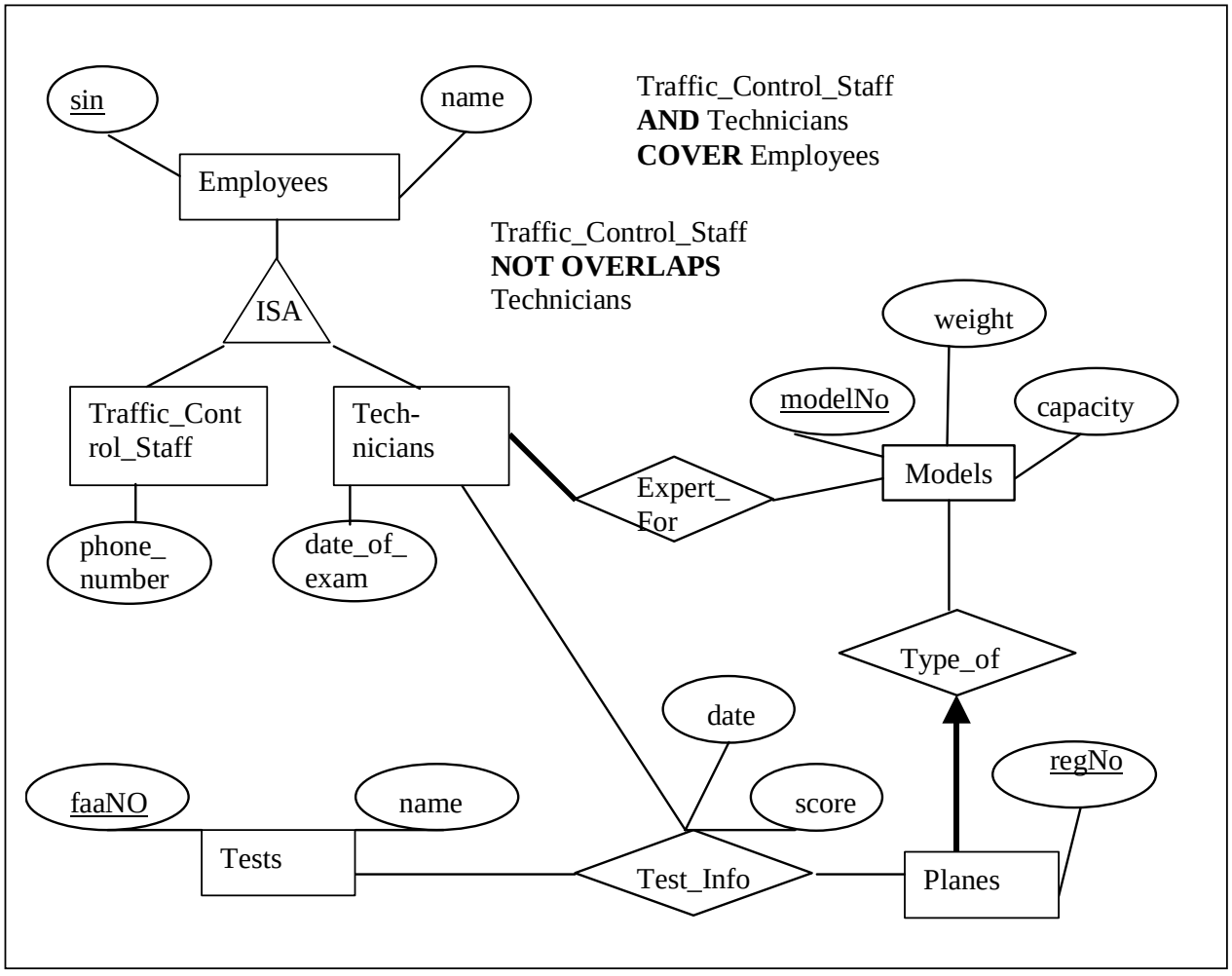
CMPT 354 3-05 Ester

Final Exam and Solution

Total marks: 620 (45 %)
Date: December 16, 2005
Time: 180 minutes

Problem 1 (Translation of ER Diagram into relational model) (120 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.

b) Explain your choice of the representation of the relationship sets including the ISA relationship. Furthermore, list the integrity constraints of the ER diagram that are not expressed in your relational schema.

1 a) 80 marks

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

CREATE  TABLE Technicians (          sin VARCHAR(10),
                                     name VARCHAR (20),
                                     date_of_exam dateTime),
                                     PRIMARY KEY (sin))


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

CREATE  TABLE Planes (               regNo VARCHAR(10),
                                     modelNo VARCHAR(10) NOT NULL,
                                     PRIMARY KEY (regNo),
                                     FOREIGN KEY (modelNo) REFERENCES Models)

CREATE  TABLE Expert_For (           sin VARCHAR(10),
                                     modelNo VARCHAR(10),
                                     PRIMARY KEY (sin, modelNo),
                                     FOREIGN KEY (sin) REFERENCES Technicians,
                                     FOREIGN KEY (modelNo) REFERENCES Models)

CREATE  TABLE Tests (                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 Tests,
                                     FOREIGN KEY (regNo) REFERENCES Planes,
                                     FOREIGN KEY (sin) REFERENCES Technicians)
```

1 b) 40 marks

The relationship sets are represented as follows:

- Type_of is a many-to-one relationship set which can be integrated with the entity set having a multiplicity of one (Planes).
- Expert_for and Test_Info are many-to-many relationship sets which require a separate table for their representation.

- Employees does not require a separate table because of the COVER constraint on Technicians and Traffic_Control_Staff.

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

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

Problem 2 (Queries in relational algebra and SQL) (120 marks)

Consider the following schema of a *company database*:

Employees(eid: integer, ename: string, address: string, supereid: integer)
Departments(did: integer, dname: string)
Projects(pid: integer, pname: string, did: integer)
Works_on(eid: integer, pid: integer, hours: integer)

Each Employee has a supervisor (another Employee) referenced by his/her supereid. Projects are uniquely assigned to a Department. The Works_on relation records which Employee works on which Project for how many hours a week.

Formulate each of the following queries in relational algebra (RA) and SQL (both, standard SQL and MS SQL notation are acceptable):

a) For each Employee, find his / her name and the name of his / her supervisor. (40 marks)

$\rho(Supervisors,\ (1 \rightarrow supeid,\ 2 \rightarrow supname),\ Employees)$
$\rho(EmpSup,\ (\sigma_{supereid=supeid}(Employees \times Supervisors)))$
$\pi_{ename,supname}\ EmpSup$

```
SELECT E.ename, S.ename
FROM Employees E, Employees S
WHERE E.supereid = S.eid
```

b) Find the eids of Employees who work on a project of every Department, i.e. find the eids of Employees who work for (a project of) every Department. (40 marks)

$(\pi_{eid,did}(Works\_on \infty Projects)) / (\pi_{did} Departments)$

```
(SELECT E.eid
FROM Employees E
WHERE NOT EXISTS
        (SELECT *
         FROM Departments D
         WHERE NOT EXISTS
                (SELECT
                 FROM Projects P, Works_on W
                 WHERE W.pid = P.pid AND P.did = D.did AND W.eid = E.eid)))
```

c) Find the pid of Projects of Department with dname = “Toys” for which at least two different Employees work. (40 marks)

$\rho(R1, \pi_{eid,\, pid}(Works\_on \infty \Pr ojects \infty (\sigma_{dname = "Toys"} Departments)))$

$\rho(R2, (1 \rightarrow eid1,\ 2 \rightarrow pid1,\ 3 \rightarrow eid2,\ 4 \rightarrow pid2), R1 \times R1)$

$\rho(R3, \sigma_{pid1 = pid2\ AND\ eid1 \neq eid2} R2)$

$\pi_{pid1} R3$

```
SELECT W1.pid
FROM Works_on W1, Works_on W2, Projects P, Department D
WHERE W1.pid = P.pid AND W2.pid = P.pid AND P.did = D.did AND
            D.dname = “Toys” AND W1.eid <> W2.eid
```

Problem 3 (Integrity constraints) (120 marks)

Consider again the company database of problem 2:

Employees(<u>eid: integer</u>, ename: string, address: string, supereid: integer)
Departments(<u>did: integer</u>, dname: string)
Projects(<u>pid: integer</u>, pname: string, did: integer)
Works_on(<u>eid: integer</u>, <u>pid: integer</u>, hours: integer)

a) An employee cannot work for a Project of Department with dname = “Toys” and for a Project of Department with dname = “Food”. (60 marks)

Formulate the above integrity constraint as SQL assertion.

```
CREATE ASSERTION CHECK
   (NOT EXISTS
         (SELECT *
          FROM Works_on W1, Works_on W2, Projects P1, Projects P2, Department D1,
                Department D2
          WHERE W1.eid = W2.eid AND W1.pid = P1.pid AND W2.pid = P2.pid AND
                P1.did = D1.did AND P2.did = D2.did AND D1.name = “Toys” AND
                D2.name = “Food”)
    )
```

Formulate an SQL trigger that ensures the above integrity constraint under updates of the Works_on table. Your trigger should explicitly undo the database modification that violated the integrity constraint, i.e. do not use the ROLLBACK command.

```
CREATE TRIGGER
ON Works_on
AFTER UPDATE
REFERENCING NEW TABLE AS NewTuple, OLD TABLE AS OldTuple
AS
BEGIN
   IF (SELECT *
          FROM Works_on W1, Works_on W2, Projects P1, Projects P2, Department D1,
                Department D2
          WHERE W1.eid = W2.eid AND W1.pid = P1.pid AND W2.pid = P2.pid AND
                P1.did = D1.did AND P2.did = D2.did AND D1.name = "Toys" AND
                D2.name = "Food")
    BEGIN
       DELETE FROM Works_on
        WHERE pid =  NewTuple.pid AND eid =  NewTuple.eid;
       INSERT INTO Works_on (SELECT * FROM OldTuple);
    END
 END
```

b) An Employee cannot work on more Projects than his / her supervisor. (60 marks)

Formulate the above integrity constraint as SQL assertion.

```
CREATE ASSERTION CHECK
   (NOT EXISTS
          (SELECT E.eid
           FROM Employees E, Employees S
           WHERE E.supereid = S.eid AND
                 ((SELECT COUNT (*) FROM Works_on WHERE eid = E.eid) >
                 (SELECT COUNT (*) FROM Works_on WHERE eid = S.eid))
    )
```

Formulate an SQL trigger that ensures the above integrity constraint under insertions of the Works_on table. Your trigger should explicitly undo the database modification that violated the integrity constraint, i.e. do not use the ROLLBACK command.

```
CREATE TRIGGER
   ON Works_on
   AFTER INSERT
   REFERENCING NEW TABLE AS NewTuple
   AS
   BEGIN
          IF (SELECT E.eid
              FROM Employees E, Employees S
              WHERE E.supereid = S.eid AND
                    ((SELECT COUNT (*) FROM Works_on WHERE eid = E.eid) >
                    (SELECT COUNT (*) FROM Works_on WHERE eid = S.eid))
```

```
        BEGIN
            DELETE FROM Works_on
              WHERE pid =  NewTuple.pid AND eid =  NewTuple.eid;
        END
    END
```

Problem 4 (Multiple Choice) (60 marks)

Mark the correct answers for the following questions. Note that multiple answers may be correct.

a) What are major advantages of database systems compared to traditional file systems?	**data independence**	direct OS support	**concurrency control**	better network support
b) Which of the following languages are relational query languages?		**SQL**	XQuery	**RA**
c) What advantages do stored procedures provide compared to pure SQL?	**Recursion**	**Parameterized queries**	Advanced database modifications	Better portability
d) What isolation level allows unrepeatable reads, but no dirty reads?	Serializable	Repeatable reads	**Read committed**	Read uncommitted
e) How can you avoid the phantom problem?	Logging	Isolation Level Serializable	**Predicate locking**	
f) What are the components of an SQL privilege?	**User**	**SQL operation**	Security level	**Database element**
g) What are main differences between XML DTDs and a relational database schema?	No references between different entities	**No primitive datatypes**	No single-valued attributes	**No integrity constraints**
h) What datacube operations allow an interactive change of the level of abstraction?	**Roll-up**	Slicing	Dicing	**Drill-down**

Problem 5 (XML) (100 marks)

Consider the following XML document type definition (DTD) for a product catalog:

```
<!DOCTYPE CATALOG [
<!ELEMENT CATALOG (TOOL | TOY)+>
<!ELEMENT TOOL (NAME,SPECIFICATIONS+,OPTIONS?)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT SPECIFICATIONS (#PCDATA)>
<!ELEMENT OPTIONS (#PCDATA)>
<!ELEMENT TOY (NAME,PRICE?)>
<!ELEMENT PRICE (#PCDATA)>]>
```

a) Design a relational database schema that captures the same information as the Catalog DTD. In particular, your relational schema must be expressive enough to be able to support the following queries b) and c). Use as few relational tables as possible. (40 marks)

You do not need to provide the corresponding SQL statements, but only an informal schema definition including the names of tables, names of attributes, the attribute domains and the specification of the primary keys, such as Departments(did: integer, dname: string).

Tools(toolid: integer, name: string, options: string)

Specifications(toolid: integer, specno: integer, spec: string)

Toys(toyid: integer, name: string, price: string)

b) Given your relational database schema, translate the following XML query into an equivalent SQL query: (30 marks)

```
FOR $T IN document("Catalog.XML")/Catalog/Tools
WHERE $T/Name = "Hammer"
RETURN <Result> $T/Specifications[1]  </Result>
```

```
SELECT S.spec
FROM Tools T, Specifications S
WHERE T.toolid = S.toolid AND S.specno = 1 AND T.name = "Hammer"
```

c) Given your relational database schema, translate the following XML query into an equivalent SQL query: (30 marks)

```
FOR $N IN document("Catalog.XML")/Catalog//Name
RETURN <Result> $N  </Result>
```

```
SELECT name FROM Tools
        UNION
SELECT name FROM Toys
```

Problem 6 (Data Warehousing) (100 marks)

Consider a data cube for the Vancouver Health Authority with the three dimensions Time, Disease and Age and the measure NumberOfPeople. The data cube records the number of people in Vancouver with a specified age (group) who have suffered from some disease at a given time (period).

a) Assume that there are only two relevant time values T1 and T2, three diseases D1, D2 and D3 and two different age values A1 and A2. Show all cells of this data cube. Introduce variables to denote the measures of the cells at the lowest level of abstraction and use these variables to express the measures of cells at higher levels of abstraction. (50 marks)

T1	D1	A1	a
T1	D1	A2	b
T1	D2	A1	c
T1	D2	A2	d
T1	D3	A1	e
T1	D3	A2	f
T2	D1	A1	g
T2	D1	A2	h
T2	D2	A1	i
T2	D2	A2	j
T2	D3	A1	k
T2	D3	A2	l
T1	D1	NULL	a + b
T1	D2	NULL	c + d
T1	D3	NULL	e + f
T2	D1	NULL	g + h
T2	D2	NULL	i + j
T2	D3	NULL	k + l
T1	NULL	A1	a + c + e
T1	NULL	A2	b + d + f
T2	NULL	A1	g + i + k
T2	NULL	A2	h + j + l
NULL	D1	A1	a + g
NULL	D1	A2	b + h
NULL	D2	A1	c + i
NULL	D2	A2	d + j
NULL	D3	A1	e + k
NULL	D3	A2	f + l
NULL	NULL	A1	a + c + e + g + i + k
NULL	NULL	A2	b + d + f + h + j + l
NULL	D1	NULL	a + b + g + h
NULL	D2	NULL	c + d + i + j
NULL	D3	NULL	e + f + k + l
T1	NULL	NULL	a + b + c + d + e + f
T2	NULL	NULL	g + h + g + h + k + l
NULL	NULL	NULL	a + . . . + l

b) Assume a fact table Patients(time, disease, age, numberOfPatients). List all SQL queries that, together, generate the measures for all cells of this data cube. (50 marks)

```
SELECT SUM(numberOfPatients)
FROM Patients
GROUP BY time, disease, age

SELECT SUM(numberOfPatients)
FROM Patients
GROUP BY time, disease

SELECT SUM(numberOfPatients)
FROM Patients
GROUP BY time, age

SELECT SUM(numberOfPatients)
FROM Patients
GROUP BY disease, age

SELECT SUM(numberOfPatients)
FROM Patients
GROUP BY time

SELECT SUM(numberOfPatients)
FROM Patients
GROUP BY disease

SELECT SUM(numberOfPatients)
FROM Patients
GROUP BY age

SELECT SUM(numberOfPatients)
FROM Patients
```