

CITS3240 Databases Midterm – 2007

Question 1 (10 marks)

Draw a single ER diagram that captures the following information about the following operations of an airport, using natural keys where possible.

- Each airplane has a registration number and is a specific model of aircraft.
- Each aircraft model has a model number (e.g. A380), a manufacturer and a fully-laden weight.
- A number of engineers work at the airport. Each engineer has a name, address and phone number.
- Every engineer is licensed to service at least one model of aircraft; each such license has a separate expiry date before which the engineer must get it renewed.
- The airport manages a number of standard airworthiness tests that are periodically applied to the aircraft; each test has a Civil Aviation Safety Authority (CASA) code, a name and a maximum possible score.
- The CASA requires the airport to keep track of each time a given airplane is tested by a given engineer using a given test. For each such testing event, the date and score must be recorded.

Notes: Use the textbook style for your ER diagram (not your own favourite style) and draw it as one connected picture, not a collection of isolated pieces. Do not give a database schema instead of an ER diagram!

PUT YOUR ANSWER ON THE NEXT PAGE

ANSWER QUESTION ONE HERE

Entity sets should be **Airplanes**, **Models**, **Tests** and **Engineers**. Relationship sets should be a binary relationship between **Airplanes** and **Models** indicating which model each airplane is, a binary relationship between **Engineers** and **Models** indicating licensing and a *ternary* relationship (called, say, **TestEvent**) that relates an airplane, an engineer and a particular test. Each **Airplane** is *exactly one Model* and so there is total participation with a key constraint here. **Engineers** are licensed to service **Models** (not individual planes) and this relationship should have the expiry-date attribute. Every engineer is licensed for *some* model and so there is total participation here. The ternary relationship **TestEvent** can possibly (though awkwardly) be modelled as a binary relationship where one component is an aggregate, but it is easy to misplace the attributes in this case. More importantly there is no natural aggregation — the three components are related purely by the particular test event.

Question 2 (9 marks)

Consider the following DB schema recording employees working for departments for a certain percentage of their time (where the fields *eid* and *did* in the Works relation are foreign keys to the same-named fields in Employee and Department).

Employee (*eid* : integer, *name* : string , *salary* : real)

Works (*eid* : integer, *did* : integer, *percent* : real)

Department (*did* : integer, *name* : string, *budget* : real)

1. Write the MySQL statements necessary to create the table **Works** (you may assume that the other tables have been created) in such a way that referential integrity is assured and that the entry is removed if the employee leaves or the department is closed.

```
CREATE TABLE works (  
  eid INT,  
  did INT,  
  percent DOUBLE,  
  PRIMARY KEY (eid, did),  
  FOREIGN KEY (eid) REFERENCES employee (eid) ON DELETE CASCADE,  
  FOREIGN KEY (did) REFERENCES department (did) ON DELETE CASCADE  
 ) Engine = InnoDB;
```

Key points here are to declare the foreign keys, the behaviour on delete and to remember that MySQL needs the InnoDB engine to enforce referential integrity.

2. Write the MySQL statement needed to add a new employee Bill Gates with employee id 123 and salary \$100000.

```
INSERT INTO employee VALUES (123, 'Bill Gates', 100000)
```

Don't miss out the word VALUES.

3. Write a relational algebra expression that yields the names of all the employees that work for both the Toys and Hardware departments.

$$\pi_{\text{name}}(E \bowtie (\pi_{\text{eid}}(\sigma_{\text{name}='Toys'}D \bowtie W \bowtie E) \cap \pi_{\text{eid}}(\sigma_{\text{name}='Hardware'}D \bowtie W \bowtie E)))$$

Get the Employee ids for the Toys and Hardware department, form the intersection, then re-join this table with the Employee table for one final projection to extract the names. Just intersecting the names can result in “false positives” if two people have the same name. Don't do the intersection with all the fields in place or the result will be empty! Finally, working in *both* Toys and Hardware means that the people must be in the *intersection* not the *union* of the two lists of employees.

Question 3 (16 marks)

Consider the familiar suppliers/parts/catalogue schema (where *pid* and *sid* in Catalogue are foreign keys to the same fields in Parts and Suppliers).

Suppliers(*sid* : integer, *sname* : string)

Parts(*pid* : integer, *pname* : string, *colour* : string)

Catalogue(*sid* : integer, *pid* : integer, *price* : real)

Write single MySQL queries to do each of following things:

1. List the names and prices of all red parts, most expensive first.

```
SELECT P.pname, C.price
FROM parts P, catalogue C
WHERE P.pid = C.pid
AND P.pcolour = 'red'
ORDER BY C.price DESC;
```

2. For every supplier, list their name, the number of different colours of spanner they sell and the average price they charge for those spanners.

```
SELECT S.sname, COUNT(*), AVG(C.price)
FROM suppliers S, catalogue C, parts P
WHERE S.sid = C.sid
AND C.pid = P.pid
AND P.pname = 'spanner'
GROUP BY S.sname;
```

3. Find the names of all the parts that come in just one colour.

```
SELECT P.pname FROM parts P WHERE
NOT EXISTS (SELECT * FROM parts P2 WHERE
P.pid <> P2.pid AND P.pname = P2.pname);
or
SELECT P.pname, COUNT(*) FROM parts P
GROUP BY P.pname HAVING COUNT(*) = 1;
```

4. Make a list containing the name and colour of every individual part, along with the name of the cheapest supplier of that part and its price.

```
SELECT P.pname, P.pcolour, C.price, S.sname
FROM parts P, catalogue C, suppliers S
WHERE P.pid = C.pid and C.sid = S.sid
AND C.price = (SELECT MIN(C2.price) FROM catalogue C2 WHERE C2.pid
= P.pid)
```