Exam: Databases and Information Systems 30/11/2014

ASSESSMENT P1: UNIT 2

Consider the following relational schema, which will be referred to as WORKING SCHEMA. This relational schema deals with an SQL competition held for the students of the course "Databases and Information Systems" in order to measure their skills in SQL query resolution:

QUERY(qcode: integer, wording: char(60), difficulty: char(5))
PK: {qcode}
NNV: {wording, difficulty}
TABLE(tname: char(15), degree: integer, cardinality: integer)
PK: {tname}
NNV: {degree, cardinality}
STUDENT(stid: integer, name: char(40), age: integer, mark: real)
PK: {stid}
NNV: {name, age}
NEEDS(qcode: integer, tname: char(15))
PK: {qcode, tname}
FK: {qcode}→QUERY
FK: {tname} →TABLE
SOLVES(qcode: integer, stid: integer, time: real, correct: char(2))
PK: {qcode, stid}
FK: {qcode} →QUERY
FK: {stid} →STUDENT
NNV: {time, correct}

Where the relations have the following meaning:

- **Query**:
  - *qcode*: code of the query
  - *wording*: text of the query
  - *difficulty*: ['Low', 'Average', 'High']
- **Table**:
  - *tname*: name of the table
  - *degree*: how many attributes it has
  - *cardinality*: how many tuples it has
- **Student**:
  - *stid*: identifier of the student
  - *name*: name of the student
  - *age*: her/his age
  - *mark*: the grade obtained in the most recent exam
- **Needs**: the resolution of the query of code *qcode* needs the table of name *tname*.
- **Solves**: the student with identifier *stid* has solved the query with code *qcode* in *time* seconds. The given solution is ok if *correct* take the value 'Yes', and is wrong if it takes the value 'No'.

Solve the following exercises:

1) Using the DDL of SQL we could define the logical schema of the previous database. For each of the following integrity constraints, please indicate in which cases the constraint could not be expressed inside the instruction "CREATE TABLE" and in which cases it could. In the latter, indicate how the constraint would be expressed in SQL inside the "CREATE TABLE" instruction. (0.4 points)
   Answer each case separately.

   a) Every query needs at least one table.

   b) The value of the attribute *correct* in *Solves* can only be "Yes" or "No".

   c) The value of the attribute *cardinality* in the relation *Table* cannot be negative.

   d) The constraint FK: $\{\text{qcode}\} \rightarrow \text{QUERY}$ in the relation *Needs.*

2) Write the SQL instructions that solve the following queries:

   a) Obtain the code and the wording of the queries that are solved by at least one student and that need more than two tables. (0.6 points)

   b) Obtain the name of the table that is needed in most queries (there might be more than one). (0.6 points)

   c) For all and every student in the database for which the mark is not known, please obtain the stid, the name, the number of queries that she/he has solved and the total time (in seconds) that she/he has used for their resolution. (0.8 points)

   d) Obtain the stid and the name of the students that have correctly solved all the queries of difficulty 'High', if there are any. (0.8 points)

   e) Obtain the code and the wording of the queries that have been solved (correctly or incorrectly) by some student whose age is lower than 20, indicating for each query how many tables are needed. We are only interested in those queries that need more than 2 tables. (0.8 points)

SOLUTIONS

1) Using the DDL of SQL we could define the logical schema of the previous database. For each of the following integrity constraints, please indicate in which cases the constraint could not be expressed inside the instruction "CREATE TABLE" and in which cases it could. In the latter, indicate how the constraint would be expressed in SQL inside the "CREATE TABLE" instruction. (0.4 points)
   Answer each case separately.

a) Every query needs at least one table.
   This cannot be represented with the DDL of ORACLE but it could be defined in standard SQL with an ASSERTION.

b) The value of the attribute *correct* in *Solves* can only be "Yes" or "No".
   In the table creation instruction for table SOLVES we should add the clause:
   `CHECK(correct IN ('Yes', 'No'))`

c) The value of the attribute *cardinality* in the relation *Table* cannot be negative.
   In the table creation instruction for table TABLE we should add the clause:
   `CHECK(cardinality >=0)`

d) The constraint FK: {qcode}→QUERY in the relation *Needs.*
   Just add in NEEDS the following: `REFERENCES Query(qcode).`

IMPORTANT: The queries can have alternative solutions that are not included in this document.

1) Write the SQL instructions that solve the following queries:

a) Obtain the code and the wording of the queries that are solved by at least one student and that need more than two tables. (0.6 points)

```
SELECT C.qcode, C.wording
FROM Query C
WHERE C.qcode IN (SELECT qcode FROM Solves) AND
      2<(SELECT COUNT(*)
         FROM Needs N WHERE C.qcode=N.qcode);
```

b) Obtain the name of the table that is needed in most queries (there might be more than one). (0.6 points)

```
SELECT N.nomtable
FROM Needs N
GROUP BY N.tname
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                   FROM Needs N1
                   GROUP BY N1.tname)
```

c) For all and every student in the database for which the mark is not known, please obtain the stid, the name, the number of queries that she/he has solved and the total time (in seconds) that she/he has used for their resolution. (0.8 points)

```
SELECT A.stid, A.name, COUNT(R.qcode),SUM(R.time)
FROM Student A LEFT JOIN Solves R ON R.stid=A.stid
WHERE A.mark IS NULL
GROUP BY A.stid, A.name;
```

d) Obtain the stid and the name of the students that have correctly solved all the queries of difficulty 'High', if there are any. (0.8 points)

Solution 1

```
SELECT A.stid, A.name
FROM Student A
WHERE NOT EXISTS
         (SELECT *
          FROM Query C
          WHERE C.difficulty ='High' AND
                NOT EXISTS (SELECT *
                            FROM Solves R
                            WHERE  R.stid=A.stid AND
                                   R.qcode=C.qcode AND
                                   R.correct ='Yes')) AND
      EXISTS (SELECT *
              FROM Query C1
              WHERE C1.difficulty ='High');
```

Solution 2

```
SELECT A.stid, A.name
FROM Student A
WHERE (SELECT COUNT(*)
       FROM Query C WHERE C.difficulty ='High')
     =
     (SELECT COUNT(*)
      FROM Solves R, Query C2
      WHERE   R.stid=A.stid AND R.qcode=C2.qcode AND
              C2.difficulty='High' AND R.correct ='Yes')) AND
 EXISTS (SELECT *
              FROM Query C1
              WHERE C1.difficulty ='Alta');
```

e) Obtain the code and the wording of the queries that have been solved (correctly or incorrectly) by some student whose age is lower than 20, indicating for each query how many tables are needed. We are only interested in those queries that need more than 2 tables. (0.8 points)

```
SELECT C.qcode, C.wording, COUNT(N.tname)
FROM Query C, Needs N
WHERE C.qcode=N.qcode
GROUP BY C.qcode, C.wording
HAVING C.qcode IN (SELECT R.qcode
                   FROM Solves R
                   WHERE R.stid IN (SELECT stid
                                    FROM Student A
                                    WHERE A.age<20))
       AND COUNT(N.tname)>2;
```