FINAL EXAM: DATABASES ("BASES DE DATOS") – 11/06/04 - SCHEMA

When you have finished your questionnaire, you can copy your answers to the following table. In this way, once finished your exam, you will be able to calculate the result you obtained (Right – Wrong/3) × 0,25.

1	2	3	4	5	6	7	8	9	10	11	12	13	14

Consider the following relational schema, which will be referred to as WORKING SCHEMA, which maintains information on country houses:

User(id: d1, name: d2, age: d3, vil_code: d4)
PK: {id} NNV: {name, vil_code}
Village(vil_code: d4, name: d2, inhabitants: d8)
PK: {vil_code} NNV: {name}
Country_house(hou_code: d5, rooms: d3, price: d6, vil_code:d4, average_score: d7)
PK: {hou_code} NNV: {rooms, price, vil_code}
FK: {vil_code} → Village
RESTRICTIVE deletion and On update CASCADE
Has_visited(hou_code: d5, id: d1, times: d10)
PK: {hou_code, id} NNV: {times}
FK: {hou_code} → Country_house
RESTRICTIVE deletion and On update CASCADE
FK: {id} → User
On deletion CASCADE and On update CASCADE
Opinion(num: d9, score: d7, hou_code: d5, id: d1)
PK:{num} NNV: {hou_code, score}
FK: {hou_code, id} → Has_visited
PARTIAL Referential Integrity
On deletion CASCADE and On update CASCADE

where the attributes and tables have the following meaning

User:

id: User's identification
name: User's name
age: User's age
vil_code: code for the village where the user lives

Village:

vil_code: village's code
name: village's name
inhabitants: number of inhabitants in the village

Country_house:

hou_code: code for identifying the house
rooms: number of rooms the house has
price: euros per day and person
vil_code: the code of the village where the country house is located
average_score: average score obtained for this house through users' opinions

Has_visited:

User with id *id* has visited the country house with code *hou_code* in *times* occasions

Opinion:

There is an Opinion with identifier *num* made by the user with id *id* over the country house with code *hou_code* which values the house with a *score.*

And consider the following extension of the previous schema. We will refer to this extension as database (DB):

User			
Id	Name	Age	Vil_code
1	Luisa	48	44
2	María	21	45
3	Juan	32	45

Village		
Vil_code	Name	Inhabitants
44	Teruel	32.000
45	Toledo	68.000
16	Cuenca	46.000
10	Cáceres	82.000

Country_house				
Hou_code	Rooms	Price	Vil_code	Average_score
c1	5	180	16	6
c2	4	100	16	10
c3	2	60	45	8
c4	8	250	10	

Has_visited		
Hou_code	Id	Times
c1	1	1
c1	2	1
c2	1	4
c2	2	4
c3	1	2
c3	3	1
c4	1	1

Opinion			
Num	Score	Hou_code	Id
1	6	c1	1
2	7	c1	
3	5	c1	
4	10	c2	1
5	10	c2	2
6	8	c3	

And now solve the questions which follow.

FINAL EXAM: "DATABASES" – 11/06/04 – Questionnaire Type A

1. If we add to the working schema the following definition in standard SQL:
CREATE VIEW Good_Customers AS
SELECT id, name FROM User U WHERE id IN (SELECT id FROM Opinion)
AND NOT EXISTS (SELECT * FROM Opinion O WHERE O.id = U.id AND O.score<5),
Which of the following statement is TRUE?

a) The definition is correct but it cannot be added to the schema because the schema is already defined.

b) It is not correct since we cannot define views with subqueries.

c) The view which is created can be queried like any other table in the schema.

d) It can only be queried if we added the clause WITH CHECK OPTION.

2. According to the view *Good_customers* defined in the previous example, the query
SELECT id, name FROM Good_customers B WHERE (SELECT COUNT(hou_code) FROM Has_visited H WHERE H.id=B.id) >10,
transformed into the logical schema is:

a) SELECT id, name FROM User U WHERE (SELECT COUNT(hou_code) FROM Has_visited H WHERE H.id=U.id) >10

b) SELECT id, name FROM Good_customers B WHERE id IN (SELECT id FROM Opinion) AND NOT EXISTS (SELECT * FROM Opinion O WHERE O.id = U.id AND O.score<5) AND (SELECT COUNT(hou_code) FROM Has_visited H WHERE H.id=B.id) >10

c) SELECT id, name FROM Good_customers B WHERE (SELECT COUNT(hou_code) FROM Has_visited H WHERE H.id=B.id) >10

d) SELECT id, name FROM User U WHERE id IN (SELECT id FROM Opinion) AND NOT EXISTS (SELECT * FROM Opinion O WHERE O.id = U.id AND O.score<5) AND (SELECT COUNT(hou_code) FROM Has_visited H WHERE H.id=U.id) >10

3. The standard SQL instruction to get that the tuple in the table *Opinion* with *num*=5 in the database DB changes its *id* to null is:

a) DELETE id FROM Opinion WHERE num=5.

b) UPDATE Opinion SET id=NULL WHERE num=5.

c) UPDATE Opinion SET id=NULL, num=5.

d) UPDATE Opinion SET id=NULL WHERE id=5.

4. On the working schema, which of the following statements is FALSE?:

a) There cannot be opinions of a country house which hasn't been visited by anybody.

b) For each country house the database must contain the village where it is.

c) There can be country houses which haven't been visited by any user.

d) Opinions over a country house are always performed by some of the users which have been in the house.

5. According to the working schema, which of the following expressions in Relational Algebra answer the question: which villages have more than one house?

a) (Country_house[hou_code, vil_code](hou_code, CC1) ⋈
Country_house[hou_code, vil_code](hou_code, CC2))
WHERE CC1<>CC2 [vil_code]

b) (Country_house[hou_code, vil_code] ⋈
Country_house[hou_code, vil_code]) [vil_code]

c) (Country_house[hou_code, vil_code](hou_code, CC1) ⋈
Country_house[hou_code, vil_code](hou_code, CC2)) WHERE CC1=CC2 [vil_code]

d) (Country_house[hou_code, vil_code]–
Country_house[hou_code, vil_code]) [vil_code]

6. According to the working schema, which information returns the following expression in Relational Algebra?:

Country_house[hou_code] – (Has_visited⋈ User)[hou_code]

a) Country houses which have been visited by exactly one user.

b) Country houses which have been visited by no users.

c) Country houses which have been visited by more than one user.

d) Country houses which have been visited by exactly two users.

7. Which would be the consequence of executing the following SQL instruction over the working schema?:

```
CREATE ASSERTION R1
CHECK (NOT EXISTS (SELECT *
                   FROM User U
                   WHERE EXISTS (SELECT *
                                 FROM Country_house C, Has_visited H
                                 WHERE C.hou_code = H.hou_code
                                       AND C.vil_code = U.vil_code
                                       AND U.id = H.id
                                       AND H.times > 2)));
```

a) It would add a constraint to avoid that a user could be more than once in a country house.

b) It would create a derived relation with all the information of the users which have been more than twice in a country house.

c) It would add a constraint to force that all the users have been at least twice in a country house in their village.

d) It would add a constraint to avoid that a user would be three or more times in a country house in the user's village.

8. Considering the working schema, which of the following statements is TRUE?

a) When we delete the country house we also delete all the country houses in the same village.

b) We cannot delete a country house if there is another country house in the same village.

c) We can only delete a village if it does not have country houses.

d) All the villages have at least one country house.

9. Which of the following statements on the Foreign Key on *Opinion* is FALSE?:

a) If we define a not null constraint (NNV) over the attribute *id* it would not be necessary to express the type of the referential integrity.

b) Allows having opinions over country houses without knowing the user which gives the scores.

c) It would be equivalent to two foreign keys of the form:

FK:{hou_code} → Country_house

On deletion CASCADE and On update CASCADE

FK:{id} → User

On deletion CASCADE and On update CASCADE

d) A modification of the attribute *id* in *User*, of a User which has given an opinion on some country house, would be allowed if there is no other user with the new *id*.

10. Which would be the final state of tables *Has_visited* and *Opinion* after executing the instruction DELETE FROM Has_visited WHERE id <> 2 over the database DB?

a)

Has_visited		
Hou_code	id	Times
c1	2	1
c2	2	4

Opinion			
Num	Score	Hou_code	Id
2	7	c1	
3	5	c1	
5	10	c2	2

The rest of relations would remain the same.

b)

Has_visited		
Hou_code	id	Times
c1	1	1
c2	1	4
c3	1	2
c3	3	1
c4	1	1

Opinion			
Num	Score	Hou_code	Id
1	6	c1	1
2	7	c1	
3	5	c1	
4	10	c2	1
6	8	c3	

The rest of relations would remain the same.

c)

Has_visited		
Hou_code	id	Times
c1	2	1
c2	2	4

Opinion			
Num	Score	Hou_code	Id
5	10	c2	2

The rest of relations would remain the same.

d)

Has_visited		
Hou_code	id	Times
c1	1	1
c1	2	1
c2	1	4
c2	2	4
c3	1	2
c3	3	1
c4	1	1

Opinion			
Num	Score	Hou_code	Id
1	6	c1	1
2	7	c1	
3	5	c1	
4	10	c2	1
5	10	c2	2
6	8	c3	

The rest of relations would remain the same.

11. If we modify the definition of relation *Opinion* by adding the new attribute "*validity*": which of the following statements is FALSE?

a) If the binding is performed at compilation time, it would be necessary to recompile all the applications which use table *Opinion*.

b) It would not be necessary to modify the views which use table *Opinion*.

c) If the binding is performed at execution time, it won't be needed to recompile all the applications which use table *Opinion*.

d) Physical independence will guarantee that the physical schema won't change.

12. Consider the following transaction T1 which is executed over the database DB:

```
BEGIN T1
        INSERT INTO Opinion VALUES (10,8,'c3',2);
        INSERT INTO Has_visited VALUES ('c3',2,2);
        COMMIT WORK;
COMMIT
```

which of the following statements is TRUE?

a) T1 will not fail because transaction processing complies with the atomicity property.

b) T1 will not fail if the referential integrity in *Opinion* is defined as INITIALLY IMMEDIATE.

c) T1 will not fail if the referential integrity in *Opinion* has been defined as INITIALLY DEFERRED.

d) T1 will not fail if we include the clause WITH NO CHECK OPTION in the definition of a transaction.

13. Considering transaction T1 which has been defined previously and considering that U1, U2 and U3 are user identifiers in the DBMS. If we execute the SQL statements:

GRANT INSERT ON Opinion TO U1
GRANT INSERT ON Has_visited TO U2 WITH GRANT OPTION

which of the following statements is TRUE?

a) U1 will be able execute transaction T1 in any case.
b) U2 will be able to execute T1 if s/he receives permission from U1 to insert into *Opinion*.
c) U1 will be able to execute T1 if s/he receives permission from U2 to Insert into *Has_visited*.
d) U3 will be able to execute T1 if s/he receives permission from U1 and U2 to insert into *Opinion* and *Has_visited* respectively.

14. If there is a loss of secondary storage during the execution of T1 and this loss does not affect the log file, which of the following statements is TRUE?

a) It will be necessary to recover the most recent backup and repeat all the transactions since that date.
b) It will be necessary to recover the most recent backup and repeat all the confirmed transactions since that date.
c) We'll undo transaction T1 and all which depend from it.
d) We'll undo T1 and the rest of unconfirmed transactions.

FINAL EXAM: DATABASES – 11/06/04 – Problems

Given the working schema presented before, solve the following exercices in standard SQL:

1. Obtain the code and the name of the villages which have no country house with average score which is lower than 5. (0.5 points)
2. Obtain the id and the name of the users who have given opinions over some country house with more than three rooms with a score which is greater than 5. (0.5 points)
3. Obtain the id and the name of the users who have been in a country house more times than any other users. (0.5 points)
4. Obtain, for each and every village in the database, the village's code and name, and how many country houses it has. (0.75 points)
5. Assuming that we have opinions on all the country houses, obtain the code and the number of rooms of the country houses which have only opinions wiht a score greater or equal than 5. (0.75 points)
6. Obtain the id and the name of the users which have been in all the country houses in some village. (1 points)
7. Obtain the code and the name of all the villages which have more country houses. (1 points)
8. The attribute *average_score* in the relation *Country_house* is a derived attribute which is obtained by calculating the average value, for each country house, of the scores which have been given in several opinions, i.e.:

$$Average\_score\_house_i = \frac{\sum scores\_givent\_to\_house}{number\_opinions\_on\_the\_house}$$

   a. Apart from the insertion in relation *Opinion*, enumerate the operations over the database which affect the value of the derived attribute. (0.75 points)
   b. Design a standard SQL trigger to control the insertion event in *Opinion*. (0.75 points)

SOLUTIONS TO THE QUESTIONNAIRE:

	Answers Type A/E	Answer Type B/F	Answer Type C/G	Answer Type D/H
1	C	D	A	B
2	D	A	B	C
3	B	C	D	A
4	D	A	B	C
5	A	B	C	D
6	B	C	D	A
7	D	A	B	C
8	C	D	A	B
9	C	D	A	B
10	A	B	C	D
11	D	A	B	C
12	C	D	A	B
13	C	D	A	B
14	B	C	D	A

SOLUTIONS TO THE PROBLEMS:

1. SELECT vil_code, name FROM Village P
   WHERE vil_code NOT IN
      (SELECT vil_code FROM Country_house C WHERE average_score <5);

2. SELECT id, name FROM User U
   WHERE id IN (SELECT O.id FROM Opinion O, Country_house C
                  WHERE O.hou_code=C.hou_code AND C.rooms>3 AND O.score>5);

3. SELECT id, name FROM User U
   WHERE id IN (SELECT id FROM Has_visited
                  WHERE times = (SELECT MAX(times) FROM Has_visited));

4. SELECT P.vil_code, P.name, COUNT(hou_code)
   FROM Village P LEFT JOIN Country_house C ON P.vil_code= C.vil_code
   GROUP BY P.vil_code, P.name;

5. SELECT hou_code, rooms FROM Country_house C
   WHERE NOT EXISTS (SELECT * FROM Opinion O
                        WHERE C.hou_code=O.hou_code AND score<5));

6. SELECT id, name FROM User U
   WHERE EXISTS (SELECT * FROM Village P
                  WHERE NOT EXISTS (SELECT * FROM Country_house C
                                       WHERE C.vil_code=P.vil_code AND
                                    NOT EXISTS (SELECT * FROM Has_visited H
                                                   WHERE H.id=U.id AND
                                                      H.hou_code=C.hou_code)))
         AND EXISTS (SELECT * FROM Country_house C1
                        WHERE P.vil_code = C1.vil_code);

7. SELECT vil_code, name FROM Village P
   WHERE vil_code IN (SELECT vil_code FROM Country_house C
                        GROUP BY vil_code
                        HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                                             FROM Country_house
                                             GROUP BY vil_code));
      o bien
   SELECT vil_code, name FROM Village P
   WHERE vil_code IN (SELECT vil_code FROM Country_house C
                        GROUP BY vil_code
                        HAVING COUNT(*) >= ALL (SELECT COUNT(*)
                                                   FROM Country_house
                                                   GROUP BY vil_code));

8. a) The instructions we have to control are:
   – Insert into *Opinion* ➔ recalculate *average_score*
   – Delete from *Opinion* ➔ recalculate *average_score*
   – Update score in *Opinion* ➔ recalculate *average_score*
   – Update *hou_code* in *Opinion* ➔ recalculate *average_score*
   – Insert into *Country_house* ➔ the value in *average_score* must be null
   – Update *average_score* in *Country_house* ➔ forbidden

   b) In standard SQL:

```
CREATE TRIGGER Opinion_Insertion
AFTER INSERT ON Opinion
REFERENCING NEW ROW AS mynew
FOR EACH ROW
BEGIN ATOMIC
        DECLARE mymean AS INTEGER;
        SELECT AVG(score) INTO mymean
                FROM Opinion
                WHERE hou_code=mynew.hou_code;
        UPDATE Country_house SET average_score= mymean
                WHERE hou_code=mynew.hou_code;
END
```