FINAL EXAM: DATABASES ("BASES DE DATOS") – 3/6/09 – SCHEMA

Consider the following relational schema, which will be referred to as WORKING SCHEMA, which maintains information about several elections:

Politician(**id**: string(4), **natdoc**: integer,
name: string(35), **party**: string(6),
n_of_times: integer)
PK: {id}
NNV:{natdoc, name}
Uni: {natdoc}
FK: {party} → Party f(party)=acronym

Party(**acronym**: string(6), **name**: string(40),
web: string(80), **founded**: integer)
PK: {acronym}

Election_type(**code**: string(6), **description**:
string(30))
PK: {code}
NNV: {description}

Election(**type**: string(6), **year**: integer, **date**:
date, **invalid_votes**: integer,
blank_votes: integer, **total_votes**:
integer)
PK: {type, year}
NNV: {date}
FK: {type} → Election_type f(type)=code

Candidate_list(**num**: integer, **type**: string(6),
year: integer, **party**: string(6), **illegal**:
boolean, **votes**: integer)
PK: {num}
Uni: {type, year, party}
FK: {type, year} → Election
Complete referential integrity
FK: {party} → Party f(party) = acronym

Participates(**type**: string(6), **year**: integer,
party: string(6), **position**: integer,
candidate: string(4))
PK: {type, year, party, position}
FK: {type, year, party} → Candidate_list
f(type)=type, f(year)=year, f(party)=party
FK: {candidate} → Politician f(candidate)=id

where the attributes and tables have the following meaning:

Politician: for each politician we store a unique identifier (id), their national document number (natdoc), their name, the party to which he or she is enrolled and the number of times that he or she has been candidate.

Party: for each party we store its acronym, its name, the webpage address and the year in which it was founded.

Election_type: for each type of election we store its code and a short description.

Election: for each election we store the type of election, the year, the complete date and the number of total votes, null (invalid) votes and blank votes.

Candidate_list: for each candidate list which is presented, we store its party, the number of received votes and whether the candidate list has been illegalised or not.

Participates: for each member of a candidate list we store the politician who is the candidate, the candidate list in which he or she appears, and the position in the list.

And consider the following extension of the previous schema. We will refer to this extension as database (DB). The symbol '?' represents null values:

Politician

id	natdoc	name	party	n_of_times
P1	11111	Alberta Márquez	?	1
P2	22222	Antonio Roig	CAU	1
P3	333	Roberto Whiskers	CAU	1
P4	4444	Suzanne Belloch	?	0
P5	5555	María García	PEM	2

Election_type

code	description
EU	Elecciones Europeas
AUTO	Elecciones Autonómicas
LOC	Elecciones Locales

Party

acronym	name	web	founded
PFA	Party de los funcionarios autónomos	www.pfa.eu	1979
PEM	Party extremista moderado	?	1871
CAU	Caritas Unidos	www.com.cau	?
PUI	Party unionista independiente	?	1999

Election

type	year	date	invalid_votes	blank_votes	total_votes
EU	2009	07/06/2009	?	?	?
AUTO	2007	27/05/2007	10	25	250
LOC	2007	27/05/2007	11	32	270

Candidate_list

num	type	year	party	illegal	votes
1	EU	2009	PEM	No	?
2	EU	2009	PUI	Yes	?
3	AUTO	2007	PEM	No	25
4	AUTO	2007	CAU	No	225
5	AUTO	2007	PUI	Yes	?
6	LOC	2007	CAU	No	170
7	LOC	2007	PFA	No	100

Participates

type	year	party	position	candidate
AUTO	2007	PEM	1	P5
AUTO	2007	CAU	2	P2
AUTO	2007	CAU	1	P3
LOC	2007	PFA	1	P1
LOC	2007	CAU	2	P5

Final exam: "Databases" – 3/06/09 – QUESTIONNAIRE TYPE A

This questionnaire has 14 questions; for each one we propose four possible answers. Only one of them is correct. The answer must be included in the answer sheet that has been handed with the exam. The maximum mark for the questionnaire is 3.5 points. The result is obtained through the formula: (Right – Wrong/3) × 0.25.

1. Given the working schema, which of the following statements is FALSE?

a) A party can have two candidate lists in the same year and with the same type with two different numbers.
b) A party can have many candidates for the same candidate list.
c) A politician need not be enrolled in a party.
d) A politician can participate in a candidate list of a party which is different to the party where he or she is enrolled.

2. Considering the working schema, what expression in Relational Algebra corresponds to the query "Natdoc and name of the politicians who have not been the first ones in any candidate list"?

a) (POLITICIAN ⋈ (POLITICIAN[id] –
(PARTICIPATES((candidate, id)) WHERE position = 1) [id]))[natdoc, name]
b) (POLITICIAN ⋈
(PARTICIPATES((candidate, id)) WHERE position = 1) [id])[natdoc, name]
c) (POLITICIAN ⋈ (POLITICIAN[id] ∩
(PARTICIPATES WHERE position = 1) [id]))[natdoc, name]
d) (POLITICIAN ⋈ (POLITICIAN[id] –
(PARTICIPATES((candidate, id)) WHERE position ≠ 1) [id]))[natdoc, name]

3. Assuming that all the foreign keys are defined with "on delete cascade", what would happen if over the database DB we execute the instruction DELETE to delete all the tuples from the relation Election?

a) All the tuples of the relations Election and Election_type would be deleted. The rest of relations would not change.
b) All the tuples of the relations Election would be deleted. The rest of relations would not change.
c) All the tuples of the relations Election_type, Election, Candidate_list and Participates would be deleted. The rest of relations would not change.
d) All the tuples of the relations Election and Participates would be deleted and also all the tuples in Candidate_list such that the type and the year are not null. The rest of relations would not change.

4. What would be the consequence of executing the following SQL instruction over the working schema?

```
CREATE ASSERTION r1
CHECK (NOT EXISTS  (SELECT * FROM Participates C
              WHERE EXISTS (SELECT * FROM Politician P
                     WHERE P.party <> C.party AND P.id = C.candidate)));
```

a) We would add an integrity constraint to force all the politicians to have a party.
b) We would add an integrity constraint to avoid that a politician could participate in a candidate list of a party which is different to the one he or she is enrolled.
c) Nothing would happen because what is expressed by this constraint is already considered by the constraints of the schema.
d) We would add an integrity constraint that would forbid having a candidate in two candidate lists.

5. Assume that all the integrity constraints are defined as a DEFERRABLE INITIALLY IMMEDIATE and consider that the following transaction is executed over the database DB in the DBMS Oracle:

```
COMMIT;
SET CONSTRAINT ALL IMMEDIATE;
INSERT INTO Election VALUES (‘UNI’, 2009, 21/5/2009, 1, 7, 200);
INSERT INTO Election_type VALUES(‘UNI’, 'Elecciones Universitarias');
COMMIT;
```

Which of the followings statements is TRUE?

a) The transaction will finish and will only insert one election.
b) The transaction will fail, given that one of the instructions violates an integrity constraint, and nothing will be inserted.
c) The transaction will perform the two instructions and will finish correctly.
d) The transaction will finish and will only insert the election type.

6. Consider the following assertion over the working schema:
The foreign key {type, year, party} → Candidate_list f(type)=type, f(year)=year, f(party)=party of the relation Participates is not well defined because it does not make reference to the primary key and also because in Candidate_list these attributes can be null.
Is it TRUE?

a) No, the foreign key is not well defined, but only because of the attributes in candidate_list can~~not~~ be null.
b) Yes, the assertion is correct.
c) No, the foreign key is well defined.
d) No, the foreign key is not well defined, but only because it does not make reference to the primary key.

7. Consider the following assertion over the database:
‘In the relation Participates, the row (‘LOC’, 2007, ‘CAU’, 2, ‘P5’) is wrong because the candidate ‘P5’ is not in the party ‘CAU’ and, moreover, she also participates in another candidate_list the same year.’
Is it TRUE?

a) Yes, it is true.
b) No, it is not, because the row is wrong only because the candidate ‘P5’ is not in the party ‘CAU’
c) No, it is not, because the row is wrong only because the candidate ‘P5’ does participate in another candidate list the same year.
d) No, it is not, because the row is correct.

8. What does the following expression in relational algebra return when applied over the working schema?:
((Participates((candidate, id)) $\bowtie$ Politician) [party] $\bowtie$ Party ((acronym, party)))[name]

a) Name of all the politicians that have participated as candidates.
b) Name of the politicians that have participated as a candidate in candidate lists of their same party.
c) Name of the politicians that are in a party that presents some candidate list.
d) Name of the parties that have politicians in some candidate list of their party.

9. **Consider the following statement over the DBMS:**
'The updates which have been performed during a transaction can be stored into secondary memory even if the transaction has not finished yet. That means that they eventually should be undone if the transaction fails.'

a) It is always true.
b) It is not true; unconfirmed data are never stored.
c) It is true if the DBMS uses the strategy known as "immediate update".
d) It is true if the DBMS uses the strategy known as "deferred update".

10. **The internal schema …**

a) Is the description of the database in terms of their physical representation.
b) Is the partial view of the information that each group of user has.
c) Is the logical description of the information system which is based in the data independence given by the ANSI/SPARC architecture.
d) Is the description of the information system independently from the data model which is finally used.

11. **The constraint PK: {type, year} of the relation "Election"…**

a) Implies NNV: {type}; NNV: {year}; UNI: {type}; UNI:{year}.
b) Implies NNV:{type}; NNV: {year} ;UNI: {type, year}
c) Implies FK: {type, year} → Election
d) Implies NNV: {type, year}; UNI: {type}; UNI:{year}.

12. **Which of the followings statements is FALSE?**

a) The property of atomicity of a transaction means that either all the instructions in the transaction are performed or none of them.
b) The property of consistency of a transaction guarantees that their execution over a consistent state in the database will lead to another consistent state.
c) The property of isolation of a transaction guarantees that the execution of a transaction will not put the security of a database at risk.
d) The persistence of a transaction guarantees that when the transaction is confirmed, its changes are recorded into the database and cannot be lost due to failures of other transactions or due to a system failure.

13. **If we create the following view:**

```
CREATE VIEW Politicians_with_Party AS
SELECT * FROM Politician WHERE party IS NOT NULL
WITH CHECK OPTION;
```

What will happen when performing the following instruction?

```
INSERT INTO Politicians_with_Party (id, natdoc, name, party, n_of_times)
    VALUES ('P6', 6789, 'Pepe Perez', 'PEM', 0);
```

a) The tuple is inserted into the view, the table Politician does not change.
b) The tuple is inserted in the table Politician.
c) The insertion is not allowed, because the view definition is violated.
d) The instruction is invalid, because we cannot insert rows into a view which has been created with the clause "WITH CHECK OPTION".

14. **Indicate the FALSE answer:**

a) There can be parties for which we do not have any politicians.
b) There can be elections for which we do not know their type.
c) There can be several candidate lists in the same year for the same party.
d) There can be candidate lists for which we do not know their party.

FINAL EXAM: DATABASES – 3/6/09 – Problems

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

1. **Obtain the natdoc and the name of the politicians who are not enrolled in any party (0.25 points).**

2. **Create a general constraint to ensure that a politician cannot participate as a candidate in two candidate lists from different parties (0.5 points).**

3. **Obtain the id and the name of all the politicians who have been candidates during two consecutive years. (0.75 points).**

4. **Obtain the name of the candidates that have been candidates more than 5 times, but never at the 1st position in any candidate list (0.75 points).**

5. **Obtain the acronym, name and the number of politicians of the parties with the highest number of politicians (1 point).**

6. **Obtain the acronym and the name of the parties that have presented a candidate list in all the elections (1 point).**

7. **Obtain for all the parties their acronym, their name and the number of candidate lists that they have presented (1 point).**

8. **Considering that the value of the attribute *n_of_times* in the relation POLITICIAN is a derived attribute which corresponds to the number of times he or she has been candidate, please:**

    a) Indicate the events that could affect the value of this attribute and how this value should change (0.5 points).

    b) Implement the trigger which corresponds to the event "update the attribute *candidate* in the relation PARTICIPATES" (0.75 points).

SOLUTIONS TO THE QUESTIONNAIRE:

Question	Answer
1	A
2	A
3	D
4	B
5	D
6	C
7	D

Question	Answer
8	D
9	C
10	A
11	B
12	C
13	B
14	B

SOLUTIONS TO THE PROBLEMS:

1. **Obtain the natdoc and the name of the politicians who are not enrolled in any party (0.25 points).**

```
SELECT natdoc, name
FROM Politician
WHERE party IS NULL;
```

2. **Create a general constraint to ensure that a politician cannot participate as a candidate in two candidate lists from different parties (0.5 points).**

```
CREATE ASSERTION anti_turncoat
CHECK(NOT EXISTS (SELECT * FROM Participates P1, Participates P2
WHERE P1.candidate = P2.candidate AND P1.party <> P2.party));
```

3. **Obtain the id and the name of all the politicians who have been candidates during two consecutive years. (0.75 points).**

```
SELECT DISTINCT PO.id, PO.name
FROM Politician PO, Participates P1, Participates P2
WHERE PO.id=P1.candidate AND PO.id=P2.candidate AND P2.year = P1.year + 1
```

--- OR ---

```
SELECT id, name
FROM Politician
WHERE id IN        (SELECT P1.candidate
                   FROM Participates P1, Participates P2
                   WHERE P1.candidate = P2.candidate AND P2.year = P1.year + 1);
```

4. **Obtain the name of the candidates that have been candidates more than 5 times, but never at the 1st position in any candidate list (0.75 points).**

```
SELECT name
FROM Politician
WHERE n_of_times > 5 AND id NOT IN (SELECT candidate
                                    FROM Participates
                                    WHERE position = 1);
```

--- OR ---

```
SELECT P.name
FROM Politician P, Participates PAR
WHERE P.ID = PAR.candidate
GROUP BY P.name, P.id
HAVING COUNT(*)> 5 AND min(PAR.position) > 1;
```

5. **Obtain the acronym, name and the number of politicians of the parties with the highest number of politicians (1 point).**

```
SELECT PA.acronym, PA.name, COUNT(*)
FROM Party PA , Politician PO
WHERE PA.acronym =PO.party
GROUP BY PA.acronym, PA.name
HAVING COUNT(*) = (SELECT MAX(COUNT(*)
                   FROM politician
                    WHERE party IS NOT NULL
                   GROUP BY party)
```

6. **Obtain the acronym and the name of the parties that have presented a candidate list in all the elections (1 point).**

```
SELECT acronym, name
FROM Party P
WHERE NOT EXISTS (SELECT * FROM Election E
                  WHERE NOT EXISTS (SELECT * FROM Candidate_list C
                                    WHERE E.type = C.type AND E.year = C.year AND P.acronym = C.party))
       AND EXISTS (SELECT * FROM Election E);
```

--- OR ---

```
SELECT acronym, name
FROM Party P
WHERE   (SELECT COUNT(*)
           FROM Candidate_list C
           WHERE P.acronym = C.party) = (SELECT COUNT(*) FROM Election);
```

--- OR ---

```
SELECT P.acronym, P.name
FROM Party P, Candidate_list C
WHERE P.acronym = C.party
GROUP BY P.acronym, P.name
HAVING COUNT(*) = (SELECT COUNT(*) FROM Election)
```

7. **Obtain for all the parties their acronym, their name and the number of candidate lists that they have presented (1 point).**

```
SELECT P.acronym, P.name, COUNT(C.num)
FROM Party P LEFT JOIN Candidate_list C ON P.acronym = C.party
GROUP BY P.acronym, P.name;
```

8. **Considering that the value of the attribute *n_of_times* in the relation POLITICIAN is a derived attribute which corresponds to the number of times he or she has been candidate, please:**

   a) Indicate the events that could affect the value of this attribute and how this value should change (0.5 points).

   - Insert into PARTICIPATES, n_of_times is incremented by 1 for the candidate
   - Delete PARTICIPATES, n_of_times is decremented by 1 for the candidate
   - Update candidate in PARTICIPATES, n_of_times is decremented by 1 for the new candidate and n_of_times is decremented by 1 for the old candidate
   - Insert into POLITICIAN → n_of_times=0
   - Update n_of_times in POLITICIAN, forbidden

   b) Implement the trigger which corresponds to the event "update the attribute *candidate* in the relation PARTICIPATES" (0.75 points).

```
CREATE OR REPLACE TRIGGER modify_candidate
AFTER UPDATE OF candidate ON Participates
FOR EACH ROW
BEGIN
        UPDATE Politician SET n_of_times= n_of_times -1
                WHERE id=:OLD.candidate;
        UPDATE Politician SET n_of_times= n_of_times +1
                WHERE id=:NEW.candidate;
END;
```