FINAL EXAM: DATABASES ("DATABASES") – 22/06/2010 – SCHEMA

Consider the following relational schema, which will be referred to as WORKING SCHEMA, which maintains information about an airport which operates with lowcost airlines with their own crew of pilots and flight attendants.

AIRLINE(**airline_code**:string(4), **name**:string(10), **telephone**:string(9))
PK:{airline_code}
NNV:{telephone}

AIRPLANE(**airplane_code**:string(8), **model**:string(15), **seats**:integer, **check_date**:date,
airline_code:string(4))
PK:{airplane_code}
NNV:{check_date, airline_code}
FK:{airline_code} -> AIRLINE
Restricted deletion
On update cascade

PILOT(**pilot_code**:string(6), **name**:string(20), **address**:string(40), **telephone**:string(9),
flight_hours:real)
PK:{pilot_code}
NNV:{name, telephone, flight_hours}

CABIN_CREW(**crew_id**:string(6), **name**:string(20), **address**:string(40), **telephone**:string(9))
PK:{crew_id}
NNV:{name, telephone}

DESTINATION(**airport**:string(20), **city**:string(10), **country**:string(10))
PK:{airport, city}
NNV:{country}

FLIGHT_ATTENDANCE(**crew_id**:string(6), **flight_code**:string(8), **duty**:string(15))
PK:{crew_id, flight_code}
FK:{crew_id}-> CABIN_CREW
On delete cascade
On update cascade
FK:{flight_code}->FLIGHT
On delete cascade
Restricted update

FLIGHT(**flight_code**:string(8), **airplane_code**:string(8), **pilot_code**:string(6),
departure_date:date, **duration**:real, **airport**:string(20), **city**:string(10))
PK:{flight_code}
NNV:{pilot_code}
FK:{airplane_code} -> AIRPLANE
On delete cascade
On update set to null
FK:{pilot_code} -> PILOT
Restricted deletion
Restricted update
FK:{airport, city}- > DESTINATION
On delete cascade
On update cascade
WEAK referential integrity.

where the attributes and tables have the following meaning:

AIRLINE: contains the information referring to the airlines which operate in the airport. The attributes are the code which identifies the airline (**airline_code**), the name of the airline (**name**) and its telephone (**telephone**).

AIRPLANE: contains the information about the airplanes which operate in the airport. The attributes are the code of the airplane (**airplane_code**), **model**, number of seats (**seats**), date of the last maintenance check (**check_date**) and the code of the airline which operates with this airplane (**airline_code**).

PILOT: contains the information of the pilots who flight from this airport. In particular, the relation stores the code of the pilot (**pilot_code**), his/her **name**, his/her **address**, his/her **telephone** and the total flight hours since he/she started to work in the airport (**flight_hours**).

CABIN_CREW: contains the information referring to the cabin crew who attend during flights. In particular, it includes the crew member identifier (**crew_id**), his/her **name**, his/her **address** and his/her **telephone**.

DESTINATION: contains the information of the possible destinations where this airport has connections with. In particular, we store the destination **airport**, the **city** and the **country** in which this destination airport is located.

FLIGHT_ATTENDANCE: contains the relation of cabin crew attending each of the flights. Each tuple of this relation associates a cabin crew member (**crew_id**) with a flight (**flight_code**) and indicates his/her **duty** during the flight.

FLIGHT: contains the information of the flights which operate in this airport. In particular, it contains the code of the flight (**flight_code**), the code of the airplane (**airplane_code**), the code which identifies the pilot (**pilot_code**), the date of departure (**departure_date**), the **duration** of the flight, and the **airport** and **city** of destination.

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

AIRLINE		
airline_code	**name**	**telephone**
IBR	SIBERIA	0345552050
ANS	AIR NOSTRE	0345552123
GKR	GREEK AIR	0345552345
BTW	BIT AIRWAY	0345559193

AIRPLANE				
airplane_code	**model**	**seats**	**check_date**	**airline_code**
IB535	Airbus320	200	11/4/2010	IBR
AN8889	Boeing737	242	12/5/2010	ANS
IB889	Boeing787	250	7/5/2010	IBR
GK9051	Airbus380	525	1/5/2010	GKR
GK1040	?	?	30/4/2010	GKR

DESTINATION		
airport	**city**	**country**
Manises	Valencia	Spain
John Wayne	Los Angeles	USA
John Wayne	Springfield	USA
Heathrow	London	UK
Gatwick	London	UK

FLIGHT_ATTENDANCE		
crew_id	**flight_code**	**duty**
A555444	V887	Supercargo
A123654	V887	Steward1
A889912	V856	Supercargo
A902399	V887	?
A555444	V856	Steward2
A902399	V888	?

PILOT				
pilot_code	**name**	**address**	**telephone**	**flight_hours**
P20809	Manolo Llamas	C/Perdidos 7	6458989	0
P21592	Pedro Peñas	?	6457878	15.5
P12345	Sonia Mares	Avda The Paz 58	6451234	10
P54321	María Ríos	?	6454321	0

CABIN_CREW			
crew_id	**name**	**address**	**telephone**
A555444	Felipe Descalzo	?	6334545
A123654	Antonia Martina	Avda Baleares 106	6445511
A889912	Sergio Domos	C/ Soldador 2	6666772
A902399	María Seguí	?	6677234

FLIGHT						
flight_code	**airplane_code**	**pilot_code**	**departure_date**	**duration**	**airport**	**city**
V886	IB535	P21592	1/6/2010	3.5	Manises	Valencia
V887	GK9051	P20809	?	?	?	?
V856	AN8889	P12345	4/6/2010	10	John Wayne	Springfield
V888	IB535	P21592	5/6/2010	12	John Wayne	Los Angeles

Deliberate blank page

FINAL EXAM: DATABASES – 22/06/2010 – Problems

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

1. Obtain the code and the model of the airplanes which have more than one assigned flight.
   (**0.50 points**)

2. Write the following constraint in standard SQL: “A pilot cannot be assigned to two flights with the same date of departure” (**0.5 points**)

3. Obtain the code and the name of the cabin crew member who has no assigned flight for tomorrow ('23/6/2010') and has no address. (**0.75 points**)

4. Obtain the code and the name of the cabin crew member who has never flown to the USA. (**0.75 points**)

5. Obtain the name of all the airlines, also showing the number of airplanes for each of them. (**0.75 points**)

6. Obtain the code and the model of the airplanes with more than 100 seats which also accumulate more than 10,000 flight hours, and also showing this total of flight hours. (**1 point**)

7. Obtain the names of the cabin crew members who have attended all the flights with destination to any of the airports in London. We assume there is at least a flight with destination to London. (**1 point**)

8. Having into account that the value of the attribute *flight_hours* in the relation PILOT is a derived attribute which indicates the number of hours a pilot has flown, please:
   a) Indicate the instructions which may affect the value of this attribute and the way in which they would affect. (**0.75 points**)
   b) Implement a trigger which corresponds to the event: “update the attribute *duration* in relation FLIGHT”. (**0.50 points**)

Final exam: "Databases" – 22/06/2010 – 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 by the formula: (Right – Wrong/3) × 0.25.

1. Given the working schema, which of the following statements is **TRUE**?
   a) Every flight must have an assigned airplane and pilot.
   b) A cabin crew member can only be assigned to one flight.
   c) A cabin crew member can be assigned to more than one flight with the same date of departure.
   d) A flight can have more than one destination, depending on the airplane code.

2. A database is shared between disks D1 and D2, the logfile is located in disk D2 and the backup copies of the database and logfile are in a tape C3. Assume that the copy of the logfile is more recent than the copy of the database and also assume that the DBMS works with deferred update. In front of a system failure with loss of main memory, how should the DBMS operate?
   a) All the *write* instructions found in confirmed transactions since the last checkpoint are remade in the same order in which they appear in the logfile.
   b) The backup copy of the logfile is recovered and the confirmed transactions in the logfile since the date of the backup copy of the database are *automatically* remade.
   c) The backup copies of the database and logfile are recovered, the cancelled transactions found in the logfile after the date of the database copy are *automatically* undone. And, finally, all the transactions which have been performed since the date of the database copy are *manually* remade.
   d) All the *write* instructions in unconfirmed transactions since the last checkpoint are undone in the reverse order to their sequence in the logfile. All the *write* instructions of the confirmed transactions since the last checkpoint are remade in the same order in which they appear in the logfile.

3. What would be the effect of changing the referential integrity of the foreign key {airport, city} in FLIGHT to a PARTIAL referential integrity, assuming the state which is shown in the database DB?
   a) The relation FLIGHT would not be correct, because the aforementioned foreign key would be violated.
   b) It would have no effect because in this case the two types of referential integrity are equivalent.
   c) We could not insert a flight with airport='John Wayne' and city=NULL.
   d) We could not insert a flight with airport=NULL and city='Madrid'.

4. Which of the following statements over the database schema is **FALSE**:
   a) A flight always has some flight attendants.
   b) A flight always has a pilot.
   c) A pilot could have not ever flown.
   d) A destination always has a country.

5. Considering the working schema, which expression in Relational Algebra corresponds to the query "Airline codes which have never performed a flight"?
   a) AIRLINE[airline_code] – ((AIRPLANE[airplane_code, airline_code] ⋈ FLIGHT[airplane_code]) [airline_code])
   b) AIRLINE[airline_code] – AIRPLANE[airline_code]
   c) ((AIRLINE[airline_code] ⋈ AIRPLANE[airplane_code, airline_code]) ⋈ FLIGHT[airplane_code]) [airline_code]
   d) ((AIRLINE[airline_code] ⋈ (FLIGHT((airplane_code, airline_code))) [airline_code]

6. Given the working schema, which of the following options is **TRUE**?
   a) We can have several flight attendants which perform the same duty in the same flight.
   b) Every flight must have an assigned airplane.
   c) A pilot cannot have two assigned flights with the same date of departure.
   d) The same person can have two different duties as a flight attendant in the same flight.

7. If we perform the following instruction over the database DB:

   UPDATE Airplane SET airplane_code='IB777' WHERE airplane_code='IB535'

   Which of the following options is **TRUE**?
   a) Nothing would be modified, because the foreign key of the relation FLIGHT would be violated.
   b) In the relation FLIGHT, the tuples with airplane_code=IB535 will become airplane_code=NULL.
   c) In the relation FLIGHT, the tuples with airplane_code=IB535 will become airplane_code=IB777.
   d) Nothing would be modified, because we cannot modify the primary key in a relation.

8. Which of the following instructions would raise an error if performed over the database DB?
   a) Assign 'Springfield' to the attribute city for the flight with code 'V888'.
   b) Add a new flight with city=null and airport='Barajas'.
   c) Assign 'London' to the attribute city for the flight with code 'V886'.
   d) Assign 'Heathrow' to the attribute airport for the flight of code 'V887'.

9. Which of the following statements referring to database implementation is **TRUE**?
   a) Indexes ease the insertion and deletion of records in a file.
   b) The hash file organization eases the retrieval of a list of records ordered by the hash fielld.
   c) A cluster consists in storing all the relations found in a database into a single file.
   d) The insertion of records into a file is more efficient if the file organisation is disordered than if it is ordered.

10. Assume that we have defined the following view over the working schema:

    ```
    CREATE VIEW Long_flight AS
      SELECT * FROM Flight WHERE duration>5;
    ```

    and we execute the following instructions, starting with the initial state shown in database DB:

    ```
    INSERT INTO Long_flight(flight_code, pilot_code, duration)
            VALUES ('V900', 'P20809', 3);
    SELECT COUNT(*) FROM Long_flight;
    ```

    Which of the following options is **TRUE**?
    a) The flight V900 is inserted into the relation FLIGHT. The result of the SELECT will be 3.
    b) The flight V900 is inserted into the relation FLIGHT. The result of the SELECT will be 2.
    c) The flight V900 is not inserted because the view does not have the "WITH CHECK OPTION" clause.
    d) The flight V900 is inserted into the view Long_flight. The relation FLIGHT is not affected.

11. If we execute the following instruction over the database extension DB:

    DELETE FROM FLIGHT WHERE flight_code = ‘V887’

    How would this affect the database DB?
    a) Only the corresponding tuple in FLIGHT would be deleted.
    b) The tuple in FLIGHT would be deleted and also the 3 tuples of FLIGHT_ATTENDANCE which refer to it.
    c) The tuple in FLIGHT would be deleted, the 3 tuples of FLIGHT_ATTENDANCE which refer to it and also the tuple in the relation PILOT with pilot_code = ‘P20809’.
    d) No tuple would be deleted because the foreign key which is defined over pilot_code in FLIGHT has restricted deletion.

12. What would the following relational algebra expression return?

    $$(\text{PILOT}[\text{pilot\_code}] \cap (\text{PILOT} \bowtie \text{FLIGHT})[\text{pilot\_code}]) \bowtie \text{PILOT}$$

    a) All the information of the pilots with assigned flights.
    b) All the information of the pilots without assigned flights.
    c) Only the code of the pilots without assigned flights.
    d) Only the code of the pilots with assigned flights.

13. Assuming a DBMS with deferred update, if we have a system failure affecting main and secondary memory, what actions should be undertaken?
    a) Only the transactions which are confirmed after the last checkpoint should be remade.
    b) Undo all the unconfirmed transactions and remake all the confirmed transactions since the last check point.
    c) Recover the last backup copy of the database and remake the transactions which have been confirmed after the last checkpoint.
    d) Recover the last backup copy of the database and remake the transactions which have been confirmed after the last backup copy of the database.

14. If all the integrity constraints which are defined in the database schema have been defined as DEFERRABLE INITIALLY DEFERRED, and starting at the initial state shown in the database DB, how many tuples will the following transaction insert? (Assume a standard SQL DBMS)

    ```
    COMMIT;
    INSERT INTO PILOT VALUES(‘P20809’, ‘Carlos Brown’, ‘Carrer Piletes’, ‘65656565’, 0);
    INSERT INTO CABIN_CREW VALUES(‘A44444’, ‘Sergi Zaft’, ‘Carrer Piletes’, ‘67586758’);
    INSERT INTO FLIGHT_ATTENDANCE VALUES(‘A44444’, ‘V856’, ‘Flight attendant1’);
    COMMIT;
    ```

    a) 0
    b) 1
    c) 2
    d) 3

SOLUTIONS

Problems

1. Obtain the code and the model of the airplanes which have more than one assigned flight. (0.50 points)

```
SELECT a.airplane_code, a.model FROM Airplane a
WHERE (SELECT COUNT(*) FROM Flight v
       WHERE v.airplane_code=a.airplane_code) > 1;
```

   or:

```
SELECT DISTINCT a.airplane_code, a.model
FROM Airplane a, Flight v1, Flight v2
WHERE v1.airplane_code=a.airplane_code AND
v2.airplane_code=a.airplane_code AND
      v1.flight_code<>v2.flight_code;
```

2. Write the following constraint in standard SQL: “A pilot cannot be assigned to two flights with the same date of departure” (0.5 points)

```
CREATE ASSERTION R1
CHECK (NOT EXISTS (SELECT * FROM FLIGHT V1, FLIGHT V2
                   WHERE V1.flight_code <> v2.flight_code
                         AND V1.departure_date = V2.departure_date
                         AND V1.pilot_code = V2.pilot_code));
```

3. Obtain the code and the name of the cabin crew member who has no assigned flight for tomorrow ('23/6/2010') and has no address. (0.75 points)

```
SELECT p.crew_id, p.name
FROM Cabin_crew p
WHERE p.crew_id NOT IN (SELECT a.crew_id
                        FROM Flight_attendance a, Flight v
                        WHERE a.flight_code=v.flight_code AND
                              v.departure_date='23/6/2010')
      AND p.address IS NULL;
```

4. Obtain the code and the name of the cabin crew member who has never flown to the USA. (0.75 points)

```
SELECT PC.crew_id, PC.name
FROM Cabin_crew PC
WHERE PC.crew_id NOT IN (SELECT AV.crew_id
                         FROM Flight_attendance AV, Flight V,
                              Destination D
                         WHERE AV.flight_code = V.flight_code
                               AND V.airport = D.airport
                               AND V.city = D.city
                               AND D.country = ‘USA’);
```

5. Obtain the name of all the airlines, also showing the number of airplanes for each of them. (0.75 points)

```
SELECT C.name, COUNT(A.airplane_code)
FROM airline C left join airplane A
     ON C.airline_code = A.airline_code
GROUP BY C.airline_code, C.name;
```

6. Obtain the code and the model of the airplanes with more than 100 seats which also accumulate more than 10,000 flight hours, and also showing this total of flight hours. (1 point)

```
SELECT a.airplane_code, a.model, SUM(duration)
FROM airplane a, flight v
WHERE a.seats > 100 AND a.airplane_code = v.airplane_code
GROUP BY a.airplane_code, a.model
HAVING SUM(duration) > 10000;
```

7. Obtain the names of the cabin crew members who have attended all the flights with destination to any of the airports in London. We assume there is at least a flight with destination to London. (1 point)

```
SELECT PC.name
FROM CABIN_CREW PC
WHERE NOT EXISTS(SELECT * FROM FLIGHT V
                 WHERE V.city = 'London' AND
                       NOT EXISTS(SELECT *
                                  FROM FLIGHT_ATTENDANCE AV
                                WHERE AV.flight_code = V.flight_code
                                    AV.crew_id=PC.crew_id))
```

Given the assumption, it's not necessary to include the add-on (if included it's also fine):

```
      AND EXISTS (SELECT * FROM FLIGHT V
                  WHERE V.city = 'London');
```

8. Having into account that the value of the attribute flight_hours in the relation PILOT is a derived attribute which indicates the number of hours a pilot has flown, please:

   REMARK: In this solution we haven't considered whether the hours which are assigned to a flight correspond to a past flight or a future (scheduled) flight. We have assumed that the derived attribute includes the information of the pilot's total flight hours according to a database state. If a student considers the dates in the conditions or inside the triggers, by checking that they are previous to the current date (and hence finished), the exercise would also be correct.

   a) Indicate the instructions which may affect the value of this attribute and the way in which they would affect. (0.75 points)

EVENT	ACTION
Insert into Pilot	***flight_hours* must be 0**
Update of *flight_hours* in Pilot	**Forbidden**
Update of *pilot_code* in Flight	**Subtract the hours to the *flight_hours* which correspond to the old pilot and add them to the *flight_hours* which correspond to the new.**
Update of *duration* in Flight	**Subtract the old duration to *flight_hours* and add the new duration for the pilot in that flight.**
Insert into Flight	**Add the duration (if it is not null) of the inserted tuple to the total *flight_hours* for the pilot in that flight.**
Delete from Flight	**Subtract the duration (if it is not null) of the deleted tuple to the total *flight_hours* for the pilot in that flight.**

   b) Implement a trigger which corresponds to the event: "update the attribute duration in relation FLIGHT". (0.50 points)

```
CREATE OR REPLACE TRIGGER Update_duration
AFTER UPDATE OF duration ON Flight
FOR EACH ROW
BEGIN
   UPDATE Pilot
   SET flight_hours = flight_hours - :old.duration + :new.duration
   WHERE pilot_code = :new.pilot_code;
END
```

Questionnaire

1	**C**
2	**A**
3	**D**
4	**A**
5	**A**
6	**A**
7	**B**
8	**C**
9	**D**
10	**B**
11	**B**
12	**A**
13	**D**
14	**A**