Exam

SD202 – Databases

October 28, 2022

This is the final exam for the SD202 class, which will determine 100% of your grade for this class. The exam consists of two independent parts. **You must write your answer to each part on a *separate sheet* of paper.** You can choose to answer the questions in English or in French, as you prefer.

Write your name clearly on every sheet used for your exam answers, and number every page.

You are allowed **three A4 sheets** (i.e., six pages, one on each side) with personal notes of your choice. You may not use any other written material.

The exam is **strictly personal**: any communication or influence between students, or use of outside help, is prohibited. No electronic devices such as calculators, computers, or mobile phones, are permitted. Any violation of the rules may result in a grade of 0 and/or disciplinary action.

Part 1

Exercise 1: SQL for a messaging application

We consider the database of a messaging application, which consists of the following tables:

- Users, containing information about users: their userid, their name, and their phone_number. We assume that each user has a different phone number.
- Groups, containing information about chat groups: their groupid, their name, and their creation_date.
- Members, with attributes userid and groupid, describing which user is a member of which chat group.
- Messages, with information about messages sent by users to a group: a messageid (messages are numbered in the order in which they were stored), the sender which is the identifier of the user who sent the message, the destination which is the identifier of the chat group to which the user sent the message, and the content of the message.

An example database for the schema is below. (Of course, your queries should work on arbitrary databases for this schema, not just this specific database.)

Users

userid	name	phone_number
1	Jane	0123456789
2	Joe	0142424242

Groups

groupid	name	creation_date
1	Diary	2022-01-01 00:00
2	The Crew	2022-10-01 13:37

Members

userid	groupid
1	1
1	2
2	2

Messages

messageid	sender	destination	content
1	1	1	“Dear diary, today was a wonderful day”
2	1	2	“Hi Joe!”
3	2	2	“Hi Jane!”

Question 1. Write an SQL query to retrieve the name of all users.

Answer.

```
SELECT name FROM Users;
```

Question 2. Write an SQL query to retrieve the name of all groups of which the user with phone number 01 23 45 67 89 is a member.

Answer.

```
SELECT Groups.name FROM Users
    JOIN Members ON Members.userid = Users.userid
    JOIN Groups ON Members.groupid = Groups.groupid
    WHERE phone\_number = '0123456789';
```

Question 3. Write an SQL query to retrieve the message ID of all messages sent to a group of which the user with ID 42 is a member.

Answer.

```
SELECT messageid FROM Messages
    JOIN Members ON Members.groupid = Messages.destination
    WHERE userid = 42;
```

Question 4. Write an SQL query to retrieve all pairs of users that belong together to some common group. More precisely, the query result should have two attributes `u1` and `u2` corresponding to user IDs, and for each pair $\{i, j\}$ of different user IDs that are members of some common group, there should be one tuple $(\min(i, j), \max(i, j))$ in the result. Note that the value of `u1` is strictly less than that of `u2`, and that there are never any duplicates in the result.

For instance, in the example, the result of the query is:

u1	u2
1	2

Answer.

```
SELECT DISTINCT Members1.userid AS u1, Members2.userid AS u2
    FROM Members AS Members1, Members AS Members2
    WHERE Members1.groupid = Members2.groupid AND Members1.userid < Members2.userid
```

Question 5. We now extend the schema to add to the table Messages an attribute replyid so that, on the row describing the message with ID p, the column replyid contains NULL if message p is not a reply to another message, and contains q if p is a reply to the message with message ID q.

We say that a user i was *directly influenced* by another user j if i replied to a message written by j, both messages were sent to some group g of which i and j are both members, and g is the *only* group of which i and j are both members.

Write a query to compute the ID of all users that were directly influenced by the user with ID 42 (without duplicates).

Answer.

```
SELECT DISTINCT Mi.sender FROM
    Messages AS Mi, Messages AS Mj, Members AS Mei, Members AS Mej
    WHERE Mi.replyid = Mj.messageid
    AND Mi.sender = Mei.userid AND Mj.sender = Mej.userid
    AND Mei.groupid = Mi.destination AND Mej.groupid = Mj.destination
    AND Mi.destination = Mj.destination
    AND Mj.sender = 42
    AND Mi.sender <> Mj.sender
    AND NOT EXISTS (
      SELECT * FROM Members as Memi, Members as Memj
      WHERE Memi.userid = Mi.sender AND Memj.userid = Mj.sender
      AND Memi.groupid = Memj.groupid AND Memi.groupid <> Mj.destination);
```

Exercise 2: Functional dependencies

Question 1. Consider the following relations on attributes a, b, c.

R

a	b	c
1	2	3
4	5	6
7	8	9

S

a	b	c
1	2	3
4	2	6
4	2	9

T

a	b	c
1	5	6
4	5	6
4	7	7

For each of the following FDs, indicate by which relations it is satisfied:

1. $a \to b$
2. $b \to ac$
3. $ac \to b$

Answer.

1. R and S but not T
2. R but not S and T
3. R and S and T

Question 2. Given a combination of FDs and negations of FDs, we say that the combination is *satisfiable* if there can be a table over attributes a, b, c that satisfies all indicated FDs but does not satisfy any of the FDs given as negations.

For each of the following combinations, either prove that it is satisfiable by giving an example of a table that satisfies it, or explain why it is not satisfiable if it is not.

1. Satisfy the FDs $a \to b$ and $b \to c$ and $c \to a$

2. Satisfy the FDs $\mathsf{ab} \to \mathsf{c}$ but not $\mathsf{a} \to \mathsf{c}$ and not $\mathsf{b} \to \mathsf{c}$.

3. Satisfy the FDs $\mathsf{a} \to \mathsf{b}$ and $\mathsf{b} \to \mathsf{c}$ but not $\mathsf{a} \to \mathsf{c}$.

Answer. For point 1, the empty table, or the table with just one row, trivially satisfies all FDs.

For point 2, consider:

T		
a	b	c
1	1	3
1	2	4
2	1	5

For point 3, the combination is not satisfiable. Indeed, consider a table satisfying the two first FDs. For any two rows with the same value of a, by the first FD they have the same value of b, and by the second FD they have the same value of c, so these two tuples cannot be a violation of the third FD, so the third FD is also satisfied.

Reminder: Please write your answers to part 2 on a separate sheet of paper!

Part 2

Exercise 3: Schema design

We wish to design a database system for a public library. Here is a specification of the needs:

The library stores data about their books and also about registered users (i.e., the people borrowing books):

- *Each book has a title, a publication date, an author, and a unique identifier. The library might have one or several copies of each book (all copies share the same title, author, identifier, etc.).*
- *Each user has a name, an email address, and a date of expiration for their account. Two different users cannot share the same email address.*
- *Each copy of a book can either be in the library or borrowed by a user. When a user has borrowed a copy of a book, we know when the book has to be returned. By default this date is 4 weeks after the date when the book was borrowed, but the user can ask for an extension which puts the deadline to return the book to 8 weeks after the date it was borrowed. Users cannot ask for a second extension.*

Question 1. Write the SQL instructions to create a database schema satisfying the business need. Make sure to define primary keys and foreign keys.

Answer.

Question 2. Write an SQL query to retrieve all the books that are overdue, i.e., books that are not currently returned but whose scheduled return date is in the past. In SQL, you can use the function `now()` returning the current date and use comparison between dates ($d_1 < d_2$ when d_1 is before d_2).

Answer.

Question 3. Write an SQL query determining the number of books that a user has currently borrowed.

Answer.

Question 4. Explain with words and SQL queries how a librarian can determine whether a borrowed book already received an extension and, if not, how to give an extension to 8 weeks (to compute the date 13 weeks after the date `mydate`, use the syntax `mydate + INTERVAL '13 WEEKS'`).

Answer.

Question 5. Write an SQL query that returns all books whose title is "Alice in Wonderland" and, for each such book, returns the identifier and the earliest date when one copy of that book can be borrowed.

For instance if we have two copies of "Alice in Wonderland" published in 1865 identified by "id5", one of which is currently borrowed and due to be returned on the "15/11/2022" and the other one is not currently borrowed; and one copy of another edition published in 1869 identified by "id9" which is currently borrowed and due to be returned on the "27/11/2022"; then the query should return (here `now()` represents the current date):

identifier	available
id5	`now()`
id9	27/11/2022

Answer.

Exercise 4: Query evaluation

Consider a database where the schema was produced by the following SQL code:

```
CREATE TABLE person (
    id INTEGER PRIMARY KEY,
    firstname TEXT,
    lastname TEXT,
    address TEXT,
    phone TEXT);
CREATE TABLE course (
    id INTEGER PRIMARY KEY,
    teacher INT REFERENCES person,
    name TEXT);
```

```
CREATE TABLE grade (
    course INT REFERENCES course
    student INT REFERENCES person,
    score FLOAT,
    PRIMARY KEY (course, student));
```

Question 1. Consider the following query:

```
SELECT * FROM person p, course c, grade g WHERE p.id=g.student AND c.id=g.course ;
```

and let us suppose that the query plan chosen by PostgreSQL is the following

```
                                  QUERY PLAN
------------------------------------------------------------------------------
 Hash Join  (cost=58.48..94.44 rows=1700 width=192)
   Hash Cond: (g.course = c.id)
   ->  Hash Join  (cost=21.48..52.97 rows=1700 width=152)
         Hash Cond: (g.student = p.id)
         ->  Seq Scan on grade g  (cost=0.00..27.00 rows=1700 width=20)
         ->  Hash  (cost=15.10..15.10 rows=510 width=132)
               ->  Seq Scan on person p  (cost=0.00..15.10 rows=510 width=132)
   ->  Hash  (cost=22.00..22.00 rows=1200 width=40)
         ->  Seq Scan on course c  (cost=0.00..22.00 rows=1200 width=40)
```

Explain in English (or French) what this query computes and explain how PostgreSQL considers running this query.

Question 2. Consider the following query:

```
SELECT score FROM course c, grade g WHERE g.student=42 AND c.id=g.course ;
```

and let us suppose that the query plan chosen by PostgreSQL is the following

```
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Nested Loop  (cost=0.29..367.99 rows=100 width=8)
   ->  Seq Scan on course c  (cost=0.00..1.46 rows=46 width=4)
   ->  Index Scan using grade_pkey on grade g  (cost=0.29..7.96 rows=1 width=12)
         Index Cond: ((course = c.id) AND (student = 42))
```

Explain in English (or French) what this query computes and explain how PostgreSQL considers running this query.

Question 3. For this question, the schema has been changed by issuing the following command:

```
CREATE INDEX ON grade(student,course) ;
```

We consider the same query as for Q2 :

```
SELECT score FROM course c, grade g WHERE g.student=42 AND c.id=g.course ;
```

Let us suppose that, now, the query plan chosen by PostgreSQL is the following

```
                                          QUERY PLAN
-----------------------------------------------------------------------------------------------
 Merge Join  (cost=3.02..175.73 rows=100 width=8)
   Merge Cond: (g.course = c.id)
   ->  Index Scan using grade_student_course_idx on grade g  (cost=0.29..373.99 rows=100 width=12)
         Index Cond: (student = 42)
   ->  Sort  (cost=2.73..2.85 rows=46 width=4)
         Sort Key: c.id
         ->  Seq Scan on course c  (cost=0.00..1.46 rows=46 width=4)
```

Explain in English (or French) how PostgreSQL considers running this query and compare this query plan to the query plan of Q2.