Spring 2021 - **Finals week**

Spring 2021 - COM SCI143-1 - ROSARIO

Started on	Wednesday, 28 April 2021, 4:00 PM PDT
State	Finished
Completed on	Wednesday, 28 April 2021, 5:50 PM PDT
Time taken	1 hour 49 mins
Grade	**49.1** out of 55.0 (**89**%)

Question **1**
Complete
3.0 points out of 3.0

An airplane makes many flights per day, and thus visits many cities each day. Each flight on the airplane goes from an origin *A* to a destination *B*. An airline wants to make it look like it has more non-stop routes than it actually has, so it markets flights from *A* to *C* as a single flight as long as the same airplane makes the trip. (Unbeknownst to the passenger, he has to sit on the plane in between flights). That is, if a plane goes from *A* to *B* and then continues on to *C,* the airline will market it as one flight.

Suppose we have a table `flight` that contains flight data for **one day**: the airplane ID, the origin and the destination and assume an airplane only visits a city once a day.

Explain how you would **structure** a SQL query (do not write the query) to extract each of these new flights -- that is, for each existing flight from *A* to *B*, find all *C*s that can be reached from *A using the same airplane.* ***Also,*** how can you generalize this to identify all *Ds, E*s, etc.? **Hint:** what would you do with the table? Be specific, but the expected answer is pretty short. Thank you.

We do a self equijoin on the flight table where the destination equals the origin.

We could generalize this to identify Ds and Es by redoing another self equijoin on the newly created table once for Ds and twice more for Es.

Nearly identical to FOAF example from lecture.

(1) Use a self join. Join the flight table to itself on flight_left.destination = flight_right.origin. This gets us all routes A->C.

(2) To get A->D, E etc., just do another self join and another etc Recursive SQL is also fine.

Feedback:

Question **2**
Complete
2.0 points out of 2.0

Suppose we have a relation *R(A, B, C)* and suppose we compute the output of $\Pi_{A,B}\sigma_{A>6}\,(R)$. Suppose we also execute the following query.

```
SELECT A, B FROM R WHERE A > 6;
```

Note that in the relational algebra expression, *R* is a set of tuples, whereas in the SQL query, it's a table.

Give a case (briefly) where both expression will return different results. Why is that?

Answer: the SQL query will return duplicate rows of the table if there are any while the relat

When there are duplicates. Need to mention that projection removes duplicates in relational algebra, SQL does not (unless use DISTINCT)

The correct answer is: duplicate distinct

Feedback:

Question **3**

Complete

6.0 points out of 8.0

DIRECTIONS: For items (a) through (c), consider a relation *R(A, B)* that contains *r* tuples and a relation *S(B, C)* that contains *s* tuples. Assume that *r* > 0, *s* > 0. Make no assumptions about keys. For each of the following expressions, state in terms of *r* and *s* the minimum and maximum number of tuples that could be in the result of the expression. **Note:** The correct response may require a function (min, max, etc.) to be applied to *r* or *s*.

Due to CCLE limitations, all parts of this problem are listed below. **Please double check that you have written a response for all parts, before submitting.**

(a) $\Pi_{A,\ C} R \bowtie S$

(b) *R* LEFT OUTER JOIN *S*

(c) $\Pi_B\,(R) - (\Pi_B(R) - \Pi_B(S))$

(d) $S \bowtie (S \bowtie S)$

(a)

Minimum: 0

Maximum: r

(b)

Minimum: r

Maximum: r

(c)

Minimum: 0

Maximum: min(r,s)

(d)

Minimum: s

Maximum: s

(a) Min: **0** Max: **rs**

(b) Min: **r**, Max: r**s**

(c) Min: **0** Max: **min(r, s)**

(d) Min: **s** Max: **s**

Feedback:
a. -1 max

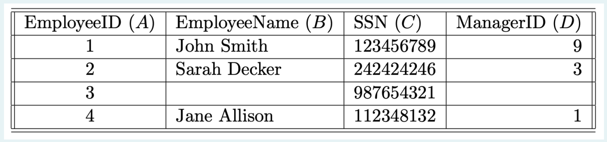
b. -1 max

Question **4**

Complete

3.0 points out of 3.0

Take a look at this table from Lecture 3. Fill in the blanks such that {*B*, *D*} is a **superkey (composite), but such that *B* and *D* are not *candidate keys* <u>for this table</u>.**



EmployeeID (A)	EmployeeName (B)	SSN (C)	ManagerID (D)
1	John Smith	123456789	9
2	Sarah Decker	242424246	3
3		987654321	
4	Jane Allison	112348132	1

Value for *B* in the incomplete row: Sarah Decker

Value for *D* in the incomplete row: 1

Several answers accepted here. Here are two examples:

B=John Smith
D=1 or 3

B=Jane Allison
D=3 or 9

B=Sarah Decker
D=1 or 9

Feedback:

Information

Suppose we want to store information about airline flights into a database... again. Each **flight** contains a set of fixed attributes (i.e. airplane number, airline code, flight number, takeoff & landing time, departure airport, arrival airport). Each **airplane** also has a set of fixed attributes (i.e. airplane number, owner airline, aircraft type). There is a lot of other data we can store about flights and the airline industry, but whatever data we store, it is based on a fixed set of attributes encoded in a schema.

Question **5**

Complete

2.0 points out of 2.0

Which of the following database models would be the ***LEAST*** effective for this use case?

- a. relational
- [✓ b. document]
- c. graph

Your answer is correct.

The correct answer is:
document

Question **6**
Complete
3.0 points out of 3.0

Explain your choice.

A relational database would allow us to relate all the data together and have information that can be easily parsed. A graph would allow us to relate flights together which would give us useful information for a dashboard or other type of information graphic.

If we use documents then we cannot relate the data together and it is essentially almost a data lake which is not useful in this application where the data has structure.

The problem clearly states that whatever data we work with in this use case, it has a fixed number of attributes. Therefore, a relational model would work fine. This is given away in the previous problem. We could easily convert a relational schema to a graph schema (airplanes belong to airlines which can be encoded using edges), and the relations are highly, well, related... airplanes fly flights and belong to airlines. Additionally, consider that each flight has origin/destination which itself induces a graph.

A document model does not have a fixed schema. We can create any attributes we want and not every object of the same type needs to have the same attributes. We really do not need a schema-less representation (document), we lose performance and the use case doesn't really make up for it.

Feedback:
Good reference to data lake

Information

Suppose we have a schema with a series of relations related to high school scheduling. In this section, you will work with various different types of logical constraints.

Time Saving Hint: to save time, read each question and scenario/logic constraint first, and *then* reference the schema to complete each task, as you don't need most of this schema.

```
CREATE TABLE student_schedule (
    student_id char(6) REFERENCES student(student_id),
    section_id smallint REFERENCES section(section_id),  -- MODIFY THIS LINE?
    PRIMARY KEY(student_id, section_id)
);
CREATE TABLE student (
    student_id char(6) PRIMARY KEY,
    last_name varchar(30),
    first_name varchar(30),
    middle_name varchar(30),
    grade smallint
);
CREATE TABLE section (
    section_id smallint PRIMARY KEY,  -- MODIFY THIS LINE? section_id represents group of students in a
classroom
    course_id char(6) REFERENCES course(course_id),  -- represents a course (i.e. ALGEBRA 2H)
    period smallint,  -- a number representing the time of day a class meets
    room_number char(6),  -- i.e. G-10
    teacher_id smallint REFERENCES teacher(teacher_id),
    enrollment_cap smallint DEFAULT 38
);
CREATE TYPE department AS ENUM ('Math', 'English', 'Science', 'History', 'Art', 'PE', 'Elective');
CREATE TABLE course (
    course_id char(6) PRIMARY KEY,
    course_title char(25),  -- the title ALGEBRA 2H
    dept department NOT NULL
);
```

Question **7**

Complete

3.1 points out of 4.0

MATCHING: Read each logical constraint. Match it with the mechanism you would use to enforce the logical constraint **based on the schema presented above.**

No two courses can share the same title.	PRIMARY KEY
We cannot enroll a student in a section that does not exist.	Foreign Key
A student cannot be enrolled in the same section twice.	UNIQUE
When we create a new section, the class period must be between 0 and 7.	CHECK CONSTRAINT
A student cannot be enrolled in a section if it will cause the enrollment in the section to exceed the maximum enrollment cap.	TRIGGER
When a student enrolls, the first class added must be an English class.	TRIGGER
Each course must be associated with one of a small set of departments.	Data Type Domain
We cannot associate a teacher with a section if it creates time conflict with any other section they teach.	TRIGGER
Each student must have a grade from 9-12.	CHECK CONSTRAINT

Your answer is partially correct.

You have correctly selected 7.
The correct answer is:
No two courses can share the same title. → UNIQUE,

We cannot enroll a student in a section that does not exist. → Foreign Key,

A student cannot be enrolled in the same section twice. → PRIMARY KEY,

When we create a new section, the class period must be between 0 and 7. → CHECK CONSTRAINT,

A student cannot be enrolled in a section if it will cause the enrollment in the section to exceed the maximum enrollment cap. → TRIGGER,

When a student enrolls, the first class added must be an English class. → TRIGGER,

Each course must be associated with one of a small set of departments. → Data Type Domain,

We cannot associate a teacher with a section if it creates time conflict with any other section they teach. → TRIGGER,

Each student must have a grade from 9-12. → CHECK CONSTRAINT

Question **8**

Complete

2.0 points out of 3.0

Take a look at the schema (`CREATE TABLE`) for the relations `student_schedule` and `section`, and notice the comments "`MODIFY THIS LINE?`". Copy and paste the correct line of the correct table into the textbox and modify it to complete the following task: **How would you change the selected line so that when a section is deleted from the master schedule (`section` table), it is removed from the student's schedule as well?**

Answer: section_id smallint PRIMARY KEY on delete cascade, -- MODIFY THIS LINE?

The correct answer is: section_id smallint REFERENCES section(section_id) ON DELETE CASCADE

Feedback:
-1 wrong table

Question **9**

Complete

4.0 points out of 5.0

A counselor in this high school needs to be able to see the full details of a student's schedule, so she creates a standard **`VIEW`** that joins together all of these tables to produce a screen on an application that lists the student's name, grade, and then each course the student is enrolled in (the period, section number, course ID, course title, room number, and teacher).

The counselor wants to make this an *updatable* view so she can modify student schedules easily (add and drop classes). **Explain** why the counselor cannot do this using an updatable view.

Hint: there are two main issues here: a technical one, and an issue with the use case. You need to identify both.

The technical issue is that if a view includes multiple tables then it cannot be updated.

The use case issue is if the counselor tries to add or drop a class from a student these changes might not be cascaded to the other tables because they might not have access to those tables.

(1) In the SQL standard, updatable views cannot contain joins and all columns not in the VIEW must be nullable.

(2) While this counselor is clever, this is not how a view works. Yes, she may be able to view the full details of a students schedule, but what she thinks is "adding a class" is simply "adding a row to the underlying data table". So, when she updates into the view there is no data integrity. She can make up whatever section IDs, periods, course titles and teachers as she wants, but it is non-sense. The use case is wrong. What she really wants the ability to do is add/drop section IDs, and have the proper information displayed on the screen. This needs to be done via an app instead.

Very few picked up on this, so some other answers were accepted as well. Some received credit if plausible, otherwise partial or no credit. Authorization was one that was accepted in some circumstances. Mentioning referential integrity, is another. **Most responses for the use case just repeated technical limitations.**

Heavy point deduction for not understanding concept of updatable view.

Feedback:
-1 nullable

Authorization is a fair point, but in order to have updatable view, must have SELECT privilege on VIEW and INSERT/DELETE privilege on table.

Question **10**

Complete

5.0 points out of 5.0

Each text below describes a particular data operation. Select the descriptions that is associated with a core operation in OLAP.

- ☑ a. Empower the user to see aggregated data for a particular level of a hierarchy and also aggregates for higher levels of the hierarchy.
- ☑ b. Empower the user to see aggregated data for a particular level of a hierarchy and also aggregates for lower levels of the hierarchy.
- ☐ c. Loading data into the data warehouse.
- ☐ d. Pulling data from the database for insertion into a data warehouse or OLAP system.
- ☑ e. Pick multiple dimensions of the data, of differing importance, to analyze.
- ☑ f. Pick the most predominant dimension of the data to analyze.
- ☐ g. General modification of the structure of the data, or individual rows, for efficiency, aesthetics or usability.
- ☑ h. Empowers analysis of data in two different important formats.

Your answer is correct.

The correct answers are: Pick the most predominant dimension of the data to analyze.,
Empower the user to see aggregated data for a particular level of a hierarchy and also aggregates for higher levels of the hierarchy.,

Empower the user to see aggregated data for a particular level of a hierarchy and also aggregates for lower levels of the hierarchy.,

Empowers analysis of data in two different important formats.,

Pick multiple dimensions of the data, of differing importance, to analyze.

Information

Suppose you are a troublemaker and you want to break into **`sam`**'s bank account. Here's how you do it:

- On the bank's login page, there is a link to a "Forgot Password?" site. You click on it.
- A regular, innocent user like Sam would enter the userid into the box, hit the "Recover My Account" button, and the system sends an email to **`sam`**'s **`email`** address to reset the **`password`**.
- **But you're not an innocent user. You want to issue a SQL injection using this textbox to change the `email` address associated with userid `sam`, so you can change the password and login as him. [How? Once the injection is complete, you refresh the page, enter `sam` into the box and hit "Recover my Account". You now receive the password reset link instead of Sam.]**

The table containing login information is called **`login`**, and contains columns **`userid, password`** and **`email`**.

The query code snippet that is executed when "Recover my Account" button is pressed is:

```
SELECT email FROM login WHERE userid='_______________________';
```

Question **11**
Complete
4.0 points out of 5.0

Fill in the blank to complete the attack: **update** **`sam`**'s **`email`** address to badguy@hackers.com.

Answer: 123' UPDATE login SET email='badguy@hackers.com' WHERE userid='sam'; --

The correct answer is: '; UPDATE login SET email = 'badguys@hackers.com' WHERE userid = 'sam'; --

Feedback:
-1 ; missing after the 123'

Question **12**
Complete
2.0 points out of 2.0

BRIEFLY explain how would you change the query, or method of querying, to prevent such an attack.

Answer: To prevent such attacks I would escape my input when creating queries from input

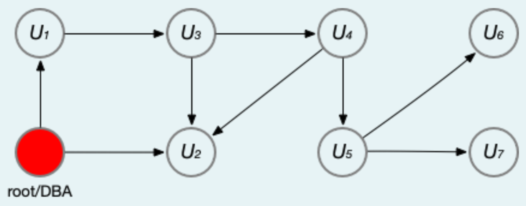
Prepared statement, or escape the inputs. Also accepted authorization if it was explained correctly.

The correct answer is: prepare

Feedback:
Prepared statements and escaping are two separate concepts

Information

DIRECTIONS: For this set of items, refer to the authorization graph displayed below. Read each **`REVOKE`** query, or set of **`REVOKE`** queries, and mark which nodes **LOSE** the privilege *p* as a result. The text on the left simulates a command prompt for a particular user. For example, **`U3=>`** means that U_3 is executing the **`REVOKE`**. Use the SQL92 standard method discussed in class.



Question 13
Complete
2.0 points out of 2.0

```
root=> REVOKE p ON *.* FROM U1;
```

- [ ] a. root/DBA
- [ ] b. U1
- [ ] c. U2
- [ ] d. U3
- [ ] e. U4
- [ ] f. U5
- [ ] g. U6
- [ ] h. U7
- [✓] i. None, or nothing happens, or an error occurs

Your answer is correct.

The correct answer is:
None, or nothing happens, or an error occurs

Question 14
Complete
2.0 points out of 2.0

```
U3> REVOKE p ON *.* FROM U2, U4 CASCADE;
```

- [ ] a. root/DBA
- [ ] b. U1
- [ ] c. U2
- [ ] d. U3
- [✓] e. U4
- [✓] f. U5
- [✓] g. U6
- [✓] h. U7
- [ ] i. None, or nothing happens, or an error occurs

Your answer is correct.

The correct answers are:
U4,

U5,

U6,

U7

Question 15
Complete
2.0 points out of 2.0

```
root=> REVOKE p ON *.* FROM U5 CASCADE; REVOKE p ON *.* FROM U6 CASCADE;
```

- [ ] a. root/DBA
- [ ] b. U1
- [ ] c. U2
- [ ] d. U3
- [ ] e. U4
- [ ] f. U5
- [ ] g. U6
- [ ] h. U7
- [✓] i. None, nothing happens or an error

Your answer is correct.

The correct answer is:

None, nothing happens or an error

Question **16**
Complete
2.0 points out of 2.0

```
U4=> REVOKE p ON *.* FROM U5 CASCADE;
```

- ☐ a. root/DBA
- ☐ b. U1
- ☐ c. U2
- ☐ d. U3
- ☐ e. U4
- ✓ f. U5
- ✓ g. U6
- ✓ h. U7
- ☐ i. None, nothing happens or an error

Your answer is correct.

The correct answers are:
U5,

U6,

U7

Question **17**
Complete
2.0 points out of 2.0

```
root=> REVOKE GRANT OPTION FOR p ON *.* FROM U1;
```

- ☐ a. root/DBA
- ☐ b. U1
- ☐ c. U2
- ☐ d. U3
- ☐ e. U4
- ☐ f. U5
- ☐ g. U6
- ☐ h. U7
- ✓ i. None, nothing happens or error.

Your answer is correct.

The correct answer is:
None, nothing happens or error.

◄ Sample Midterm Solutions

Jump to...

CS143 Spring 2021 Midterm, ... ►