Name

CWID

Homework Assignment 2

Due Date: October 17th, 2017

CS425 - Database Organization Results

Please leave this empty!

2.1 2.2

2.3 2.4 2.5 2.6 2.7 2.8 2.9 2.10

2.11 2.12

2.15 2.16 2.17 2.18 2.19 Sum

Instructions

- Try to answer all the questions using what you have learned in class
- **When writing a query, write the query in a way that it would work over all possible database instances and not just for the given example instance!**
- Some questions are marked as bonus. You do not have to answer these questions to get full points for the assignment. However, you can get bonus points for these questions!
- **Please submit the homework electronically using blackboard**

Consider the following library database schema and example instance storing:

book

bookid	title	price	total_copies
1	Introduction of Algorithms	84.66	4
2	Database System Concepts	74.99	5
3	Stochastic Calculus for Finance I	41.02	3
4	Stochastic Calculus for Finance II	55.22	3

course

courseid	title	instructorid	textbookid
1	Algorithms	1	1
2	DB Organization	2	2
3	Advanced DB Organization	3	2
4	Math Finance I	1	3
5	Math Finance II	4	4

student

studentid	name	gpa
1	Tom	3.3
2	John	3.8
3	Mary	3.0
4	Kris	3.6
5	Alex	3.5

faculty

facultyid	name	salary
1	James	70000
2	Sarah	60000
3	Jay	80000
4	Rachel	70000
5	Paul	85000

enroll

studentid	courseid
1	1
1	2
2	1
4	3
4	4
5	5

book_checkout

date	bookid	studentid
2017-08-29	1	1
2017-09-02	4	4
2017-09-07	1	4

Hints:

- All the attributes that have integer values are of type INT; numbers with decimal point are of type NUMERIC; the attribute *date* of *book_checkout* relation is of type DATE; others are of type VARCHAR
- Attributes with black background form the primary key of an relation
- The attribute *instructorid* of relation *course* is a foreign key to relation *faculty*, and *textbookid* is a foreign key to relation *book*.

- The attribute *studentid* of relation *enroll* is a foreign key to relation *student*, and *courseid* is a foreign key to relation *course*.
- The attribute *bookid* of relation *book_checkout* is a foreign key to relation *book*, and *studentid* is a foreign key to relation *student*.

Part 2.1 SQL DDL (Total: 14 Points)

Question 2.1.1 (7 Points)

Write an SQL statement that adds an *advisorid* attribute to relation *student*, and sets it to be a foreign key to *facultyid* in relation *faculty*. In case a faculty's id is changed, the change would be reflected on *advisorid* attribute; in case a student's advisor left the school, *advisorid* would be set to NULL.

Solution

```
ALTER TABLE student
  ADD [COLUMN] advisorid INT REFERENCES faculty(facultyid)
    ON UPDATE CASCADE
    ON DELETE SET NULL;
```

OR

```
ALTER TABLE student
  ADD [COLUMN] advisorid INT,
  ADD FOREIGN KEY (advisorid) REFERENCES faculty(facultyid)
    ON UPDATE CASCADE
    ON DELETE SET NULL;
```

Question 2.1.2 (7 Points)

Write an SQL statement that adds a constraint to the *student* relation to make sure that the *gpa* attribute cannot be `NULL`, and that the value of this attribute has to be between 0 and 4. Furthermore, the default value for this attribute should be 3.

Solution

```
ALTER TABLE student
  ALTER [COLUMN] gpa SET NOT NULL,
  ALTER [COLUMN] gpa SET DEFAULT 3,
  ADD CHECK(gpa >= 0 AND gpa <= 4);
```

Part 2.2 SQL Queries (Total: 56 + 10 BONUS Points)

Question 2.2.1 (5 Points)

Write an SQL query that returns the studentid and name of students who have overdue books (assume a book is due after 30 days), use construct `CURRENT_DATE` to access the current date in your query. Do not return any duplicate tuples.

Solution

```
SELECT DISTINCT studentid, name
FROM student NATURAL JOIN book_checkout
WHERE CURRENT_DATE > date + 30;
```

Natural join could be replaced by other correct forms, same below.

Question 2.2.2 (5 Points)

Write an SQL query that returns the studentid and name of the student(s) whose gpa is higher than the average gpa of all students.

Solution

```
SELECT studentid, name
FROM student
WHERE gpa > (SELECT AVG(gpa)
             FROM student);
```

Question 2.2.3 (7 Points)

Write an SQL query that returns the facultyid and name of any faculty that does not teach any course but has a salary that is higher than 80,000.

Solution

```
SELECT facultyid, name
FROM faculty
WHERE salary > 80000
      AND
      facultyid NOT IN (SELECT instructorid
                        FROM course);
```

Question 2.2.4 (7 Points)

Write an SQL query that returns the bookid and title of books that are used as textbooks for more than one course.

Solution

```
SELECT b.bookid, b.title
FROM book b
WHERE b.bookid IN (SELECT c.textbookid
                   FROM course c
                   GROUP BY c.textbookid
                   HAVING count(*) > 1);
```

Question 2.2.5 (7 Points)

Write an SQL query that returns the studentid and name of students who have checked out books that are worth more than $100 in total.

Solution

```
SELECT studentid, name
FROM student
     NATURAL JOIN book_checkout
     NATURAL JOIN book
GROUP BY studentid, name
HAVING sum(price) > 100
```

Question 2.2.6 (8 Points)

Write an SQL query that returns the studentid and name of students who checked out the textbook of a course that they did not enroll in.

Solution

```
WITH target_record AS(
  (SELECT b.*
   FROM book_checkout b,
       course c
   WHERE b.bookid = c.textbookid)
  EXCEPT
  (SELECT date, bookid, studentid
   FROM book_checkout b
       NATURAL JOIN enroll e
       NATURAL JOIN course c
   WHERE textbookid=bookid)
)
SELECT DISTINCT studentid, name
FROM student
     NATURAL JOIN target_record;
```

Alternatives include using `NOT IN`

Question 2.2.7 (9 Points)

Suppose a student can check out as many books as the number of courses he/she is enrolled in. Write an SQL query that returns studentid and name of students who can not check out any more books. Your answer should include students who did not enroll in any course.

Solution

```sql
WITH allowedBooks AS(
  SELECT studentid, count(*) AS allowed
  FROM enroll
  GROUP BY studentid
),
checkedBooks AS(
  SELECT studentid, count(*) AS checked
  FROM book_checkout
  GROUP BY studentid
)
SELECT studentid, name
FROM student
     NATURAL JOIN allowedBooks
     LEFT NATURAL OUTER JOIN checkedBooks
WHERE allowed - COALESCE(checked,0) = 0
UNION
SELECT studentid, name
FROM student
WHERE studentid NOT IN (SELECT studentid
                        FROM enroll);
```

Alternatively, using outer join:

```sql
WITH allowedBooks AS(
  SELECT studentid, count(*) AS allowed
  FROM enroll
  GROUP BY studentid
),
checkedBooks AS(
  SELECT studentid, count(*) AS checked
  FROM book_checkout
  GROUP BY studentid
)
SELECT studentid, name
FROM student s
     LEFT OUTER JOIN (allowedBooks a
     LEFT NATURAL JOIN checkedBooks c) ON (s.studentid = a.studentid)
WHERE allowed - COALESCE(checked,0) = 0 OR allowed IS NULL
```

Note that the `COALESCE` is not necessary here, but only added for clarity since any student that has not checked out any books and is registered for at least one couse should not be returned anyways.

Question 2.2.8 (8 Points)

Write an SQL query that returns the bookid and title of books that have 3 or more available copies.

Solution

```
SELECT bookid, title
FROM book b
     LEFT OUTER JOIN (SELECT bookid, count(*) AS num_checked
                      FROM book_checkout
                      GROUP BY bookid) c
     ON b.bookid=c.bookid
WHERE total_copies - num_checked >=3
      OR
      (total_copies>=3 AND num_checked IS NULL)
```

Alternatively, use COALESCE:

```
SELECT bookid, title
FROM book b
     LEFT OUTER JOIN (SELECT bookid, count(*) AS num_checked
                      FROM book_checkout
                      GROUP BY bookid) c
     ON b.bookid=c.bookid
WHERE total_copies - COALESCE(num_checked,0) >= 3
```

Question 2.2.9 BONUS (5 Points)

Write an SQL query which returns courseid and title of the course(s) that has/have the most expensive textbook.

Solution

```
SELECT c.courseid,c.title
FROM book b,course c
WHERE b.bookid = c.textbookid
      AND
      price = (SELECT max(price)
               FROM book);
```

Question 2.2.10 BONUS (5 Points)

Write an SQL query that returns the studentid and name of students that enrolled in all of the finance courses. A course is considered a finance course if the title of the course contains the string 'Finance'.

Solution

```
WITH finance AS(
  SELECT *
  FROM course
  WHERE title LIKE '%Finance%'
)
SELECT s.studentid, s.name
FROM student s
     NATURAL JOIN enroll e
     NATURAL JOIN finance f
GROUP BY s.studentid, s.name
HAVING count(*) = (SELECT count(*)
                   FROM finance);
```

The query returns no tuple in the sample instance. Alternatively, you can use a nested subquery here.

Part 2.3 SQL Updates (Total: 30 + 5 BONUS Points)

Question 2.3.1 (7 Points)

Delete all courses that no one is enrolled in.

Solution

```
DELETE FROM course
WHERE courseid NOT IN (SELECT courseid
                       FROM enroll);
```

Question 2.3.2 (8 Points)

4 copies of a new book *Distributed and Cloud Computing* has been added to the library. The price is $50.00 for each copy. Add the information to the *book* relation. Assume that *bookid* is automatically maintained by the system.

Solution

```
INSERT INTO book(title,price,total_copies)
VALUES ('Distributed␣and␣Cloud␣Computing',50.00,4);
```

Question 2.3.3 (6 Points)

One of the checkout records is incorrect. It turns out, the student with id 4 never checked out the book with id 1. He checked out the book with id 2. Update the information in *book_checkout* relation.

Solution

```
UPDATE book_checkout
SET bookid=2
WHERE bookid=1 AND studentid=4;
```

Question 2.3.4 (9 Points)

Update the *gpa* in the *student* relation according to this rule:

- if it is negative, set it to 0
- if it is larger than 4, then set it to 4
- if it is NULL, set it to 3
- if none of the above applies do not change the gpa

Note that we expect you to write a single statement that implements this.

Solution

```
UPDATE student
SET gpa=(CASE
           WHEN gpa<0 THEN 0
           WHEN gpa>4 THEN 4
           WHEN gpa IS NULL THEN 3
         END)
WHERE gpa<0 OR gpa>4 OR gpa IS NULL;
```

or

```
UPDATE student
SET gpa=(CASE
           WHEN gpa<0 THEN 0
           WHEN gpa>4 THEN 4
           WHEN gpa IS NULL THEN 3
           ELSE gpa
         END);
```

Question 2.3.5 BONUS (5 Points)

Update the salaries of faculty as their current salary + 10000 · (the number of courses they are teaching).

Solution

```
UPDATE faculty
SET salary = salary + 10000 * (SELECT count(*)
                               FROM course
                               WHERE instructorid=facultyid);
```