CS44800 Assignment 1 Solutions

9/15/2021

Note: For some questions you'll need to see the assignment sheet to understand the question or the data the answer is based on. Also, in many cases (particularly the SQL queries and relational algebra), these represent one possible answer, but there may be many equally correct answers.

Question 1

1.1

Output:

Bruce Lee	Introduction to Relational Database Systems	9
Tony Stark	Algorithm Design, Analysis, and Implementation	8

English description: find each instructor with the name of the course they are teaching and the number of students enrolled in that course.

Relational Algebra:

$$\Pi_{P.Name,C.Name}(\rho_P(People) \bowtie_{C.InstructorID=P.ID} \rho_C(Courses) \bowtie_{C.CourseID=E.CourseID} \rho_{E(CourseID,Num)}(\gamma_{CourseID,count(*)}(Enrollment)))$$

1.2

Output:

Bruce Lee	7	USA
Tony Stark	14	USA

SQL Query:
SELECT P.Name, P.ID, P.Nationality from People as P join Courses as C on P.ID=C.InstructorID;

Relational Algebra:

$$\Pi_{P.Name,P.ID,P.Nationality}(\rho_P(People) \bowtie_{P.ID=C.InstructorID} \rho_C(Courses))$$

1.3

Output: We accept both

CS448	B

or

CS448	B
CS448	B

due to the inconsistency within the problem writeup.

English Description:
List student's course and grade who has ID=5.

SQL Query:
SELECT Courses.CourseID, Enrollment.Grade from Courses natural join Enrollment where Enrollment.Student_id=5;

1.4

Output:
We accept both it is an illegal query due to mismatch on columns of both sides of the natural join or we accept

CS448 B 7 Introduction to Relational Database Systems

List the course ID, grade, instructorID, and name of the course that the student with ID=5 is taking.

SQL Query:
SELECT * FROM (SELECT CourseID, Grade from Enrollment) natural join Courses;

1.5

Output:
We accept either
CS448 B
CS580 B

or

CS448 B
CS448 B

English Description:
Return the CourseID and Grade of an instance with student ID = 5 after a cross product.

SQL Query:
SELECT CourseID, Grade from Enrollment cross join Courses where Student_id=5;

1.6

Output:
USA 5
China 2
UK 4
Japan 1
France 1

A common mistake people made is they include the instructors in the final result.

SQL Query:
select Nationality, count(distinct Name) from Enrollment join People on Enrollment.Student_id=People.ID group by Nationality;

Relational Algebra:

$$\gamma_{Nationality,count(distinct\ StudenID)}(People \bowtie_{People.ID=Enrollment.StudenID} Enrollment)$$

1.7

Output:

1 3
3 3

English Description:
List the groups of CS448 with more than 2 students and the number of students in the group.

Relational Algebra:

$\sigma_{count(StudentID)>2}(\gamma_{GroupID,count(StudentID)}(\sigma_{CourseID="CS448"}(Enrollment)))$

1.8

Output:
We accept both answers of the students who are taking the same class with Richard White and students who are taking the same class in the same section with Richard White.

Jon Snow
James Bond
Winston Churchill
Luke Skywalker
Jackie Chan
Hugo Lafayette
Ben Kenobi
Harry Potter
Son Goku
Wonder Woman
Sun Tzu
Leia Organa

Or

Jame Bond
Winston Churchill
Luke Skywalker
Jackie Chan
Hogu Lafayette
Harry Potter
Sun Tzu

SQL Query:
SELECT distinct Name from (select * from People as P where P.ID in (select Student_id from Enrollment as E where E.Course_id in (SELECT Course_id from E where Student_id in (select P.ID where P.Name=”Richard White”)) and E.SectionNum in (select Section_num from E where E.StudentID in (SELECT distinct P.ID from P where P.Name!=”Richard White”)))) where Name!=”Richard White”;

Relational Algebra:

$$\rho_{A}(\sigma_{Name="Richard\ White"}(People))$$

$$\rho_{B}(A \bowtie_{A.ID=Enrollment.StudentID} \rho_{E} Enrollment)$$

$$\rho_{C}(E \bowtie_{E.CourseID=B.CourseID\ and\ E.SectionNum=B.SectionNum} B)$$

$$\Pi_{Name}(\sigma_{distinct\ Name \wedge\ Name!="Richard\ White"}(C))$$

For Questions 2 through 6, use the following relations (schema):

Student(sid: integer, sname: string)
Course(cid: string, iid: integer, cname: string)
Instructor(iid: integer, iname: string)
Grades(sid: integer, cid: string, grade: string)

Question 2: Relational Algebra

Write relational algebra for the following queries. If you need to make any particular assumptions, please list them.

1. Find the name of the students who have registered in the course with cid CS44800.
2. Find the ids of the courses taught by at least two different instructors.
3. Find the ids of the students who never received a grade F.
4. Give a list of all people (Instructors and Students), with their ID and name.

2.1) If you considered cid from Course table:

$\prod_{sname}(\prod_{sid}(\prod_{cid}(\sigma_{Course.cid='cs44800'}(Course))Grades)Student)$

If you considered cid from Grades table:

$\prod_{Student.sname}(\sigma_{Grades.cid = "CS44800"}(Grades \bowtie Student))$

2.2) $\prod_{Course.cid}(\sigma_{Course.count(iid)>1}(\gamma_{cid,count(iid)})(Course))$

2.3) $\prod_{Grades.sid}(Grades) - \prod_{Grades.sid}(\sigma_{Grades.grade="F"}(Grades))$

2.4) $\prod_{ID,Name}(\rho_{Student(ID,Name)}(Student) \cup \rho_{Instructor(ID,Name)}(Instructor))$

Question 3: Relational Model Attribute Domains

In the above schema, we list data types (domain) for each attribute. Do you need to know the domain to write the queries in the preceding question? Explain why or why not.

Yes- Since for 2.4 we need to make a list of people. We need to know the domain since if instructor id is different domain as compared to student id then we cannot union the two tables.

Also, it is possible for the cid of Course to be a int and cid of Grades to be string. Then we would not be able to join the two tables.

Question 4:

1. It is not legal relational algebra because although Course and Grade technically have the same domains for each of their attributes (int vs. int, and string vs. string), the actual domains are not compatible between them. They also do not have the same attributes. For example, the instructor ID may not be factually compatible with the student ID. Moreover, the grade and the course name, although both are strings, cannot be compatible. One is a letter grade and another one is a word or phrase.
2. Yes if you only project the cid attribute of the Grades and the Course relations you can do a union between the two projections. I don't think it will make sense since you are combining the course IDs of the courses offered and the course IDs of the course with at least one student grade. It will just return all the courses as I assume all courses which students have grades in should appear in the Course relation.

Q5:

1. **Student table another key:** sid, sname
   Course table another key: cid, iid, cname
2. **Instructor table:** The only other possible candidate key would be iname, and we know that names are not likely to be unique. Having both together (iid iname) as the key would not be a candidate key, as a subset (iid) is a key.
   Grades Table: If sid or cid were a key alone, then the two together would not be a candidate key. Likewise, if grade were a key, it would imply that every two students can every get the same grade. As to other possibilities:
   - Sid, grade: not a candidate key since a single student can have same grades for separate course

- cid, grade: not a candidate key since multiple students can have same grades for same course

3. If we didn't know the keys, it might be possible that iid alone was not a key for Instructor, in which case iid and iname together could be a candidate key. Likewise, if we didn't know that sid and cid formed a key, we could consider a case where a student could retake a course and get a new grade – and if we assume the old grade was kept as well, then all three would form a candidate key. Although this would mean that retaking a course and getting the same grade wasn't possible.

Q6:

The given query **Course** $\bowtie_{\text{Course.iid}\neq\text{Instructor.iid}}$ **Instructor** will not give us courses that do not have an instructor. In terms of SQL query this can be written as,

SELECT * FROM Course,Instructor Where Course.iid≠Instructor.iid

This basically joins each course in the Course table with the columns in the Instructor entries with an instructor other than the one teaching the course (which is basically nonsensical). So, if a course doesn't have any instructor (meaning NULL **Course.iid** field) then that course won't even be available in the joined table (since NULL value isn't true for any comparison). So the corresponding query will give us all the other courses except the ones we are looking for. The correct query should be:

SELECT cid FROM Course WHERE iid is NULL

Q7:

In relational database systems, a null represents missing or unknown information at the column level. A null is not the same as 0 (zero) or blank. Null means no entry has been made for the column and it implies that the value is either unknown or inapplicable.

With any relational DBMS that supports nulls you can use them to distinguish between a deliberate entry of 0 (for numerical columns) or a blank (for character columns) and an unknown or inapplicable entry (NULL for both numerical and character columns).

Example of appropriate null usage:

Name	Subj A	Subj B	Subj C
Tom	91	85	97
Nial	90	NULL	91
Louis	93	83	88

Here, in the above table we see some students along with the numbers they received in 3 different subjects in a semester. The NULL value indicates that the teacher of that particular subject has yet to update the mark of that particular student. It doesn't mean he/she received a 0.

Suppose, here we want to know the average mark in **Subj B.** In this case, we just use the aggregate function **AVG** and we'll have our intended average = (85+83)/2 = 84. But, here if we used 0 to indicate unknown value the **AVG** function would return wrong value (85+0+83)/3 = 56.

Example of null usage where different structure is better:

Name	Age	Dept_Name	Building
Timmy	24	CS	2
Harry	25	Stats	3
Susan	21	NULL	NULL

Here, we see a table containing students along with theirs age, department and the corresponding building. Now, for Susan the corresponding department and building is unknown. As a result there are 2 NULL entries here taking space in the database. But if we decompose the table into 2, like below, then we don't need this NULL value at all.

Name	Age
Timmy	24
Harry	25
Susan	21

Department	Building
CS	2
Stats	3

Now, we add a 3rd table containing relation between Student Name and Department.

Name	**Department**
Timmy	CS
Harry	Stats

We see that, we don't need a table with NULL value in the above case because the corresponding entry for Susan doesn't even appear in the 3rd table since it is unknown.

This is an interesting example to think about the semantics of queries involving null. If we wanted to list all of the students in CS, Susan would not be included (as expected.) But if we wanted a list of all students NOT in CS, Susan would still not be included. As an unknown, she might or might not be in CS. Breaking it into tables and joining makes it a little more obvious that such a query would only get students who were assigned to a department, and that we'd need something different to get those who weren't assigned to a department (think about how you might do this.)

Q8:

Intersection is not a primitive relational algebra operator, because it is can be written using only primitive operators. One way to do this is:

$$A \cap B = \big((A \cup B) - (A - B)\big) - (B - A)$$