CS 348 - Homework 2

Relational Algebra
(160 Points)

Fall 2020

Due on: 10/02/2020

This assignment is to be completed by individuals. You should only talk to the instructor, and the TA about this assignment. You may also post questions (and not answers) to Campuswire.

There will be a 10% penalty if the homework is submitted 24 hours after the due date, a 15% penalty if the homework is submitted 48 hours after the due date, or a 20% penalty if the homework is submitted 72 hours after the due date. The homework will not be accepted after 72 hours, as a solution will be posted by then.

For questions 1-3, write your answers on the hw2.tex template file (provided with this homework document) and generate a pdf file. Upload the pdf file to **Gradescope**. For questions 4-5, create a sql file named **q45.sql** and write the trigger and procedure there. Upload your **q45.sql** file to **Brightspace**.

1. (70 points) Given below is a relational schema about libraries. Write relational algebra queries for the following questions.

   Book (**BookId**, *Title*, *PublId*)
   Author (**AuthId**, *AuthName*)
   AuthorBook (**AuthId**, **BookId**)
   Publisher (**PublId**, *PublName*, *Address*, *Phone*)
   BookCopies (**BookId, BranId**, *Copies*)
   BookLoans (**BookId, BranId, MembId**, *IssueDate*, *DueDate*)
   Member (**MembId**, *MembName*, *Address*, *Phone*)
   LibraryBranch (**BranId**, *BranName*, *State*)

   A. (5 points) List the names of all branches in Indiana.
   Answer: $\pi_{BranName}(\sigma_{State='Indiana'}(LibraryBranch))$

   B. (10 points) List book titles for all the books in Indiana branches. Use **theta join** to write this query.

Answer:
$LibrBran \leftarrow \sigma_{State='Indiana'}(LibraryBranch)$
$LibrBookCopi \leftarrow LibrBran \bowtie_{LibrBran.BranId=BookCopies.BranId} BookCopies$
$LibrBook \leftarrow LibrBookCopi \bowtie_{LibrBookCopi.BookId=Book.BookId} Book$
$Result \leftarrow \pi_{Title}(LibrBook)$

C. (10 points) List the names of members who **have not** checked out any books.
Answer:
$WithLoans \leftarrow \pi_{MembId}(BookLoans)$
$AllMemb \leftarrow \pi_{MembId}(Member)$
$NoLoans \leftarrow AllMemb - WithLoans$
$Result \leftarrow \pi_{MembName}(NoLoans \ltimes Member)$

D. (15 points) For each author, list their name along with the book title, branch id and number of copies for all of their book copies. Include authors who do not have books.
Answer:
$BAB \leftarrow Book \rtimes_{Book.BookId=AuthorBook.BookId} AuthorBook$
$BookCop \leftarrow BAB \rtimes_{BAB.BookId=BookCopies.BookId} BookCopies$
$ABC \leftarrow Author \rtimes_{Author.AuthId=BookCop.AuthId} BookCop$
$Result \leftarrow \pi_{AuthName,Title,BranId,Copies}(ABC)$

E. (10 points) Retrieve the *bookid* of books that are borrowed at every branch.
Answer:

$$EveryBookWithBranch \leftarrow \pi_{BookId}(BookLoans) \times \pi_{BranId}(LibraryBranch)$$

$$BookNotEveryBranch \leftarrow \pi_{BookId}(EveryBookWithBranch - \pi_{BookId,BranId}(BookLoans))$$

$$Result \leftarrow \pi_{BookId}(BookLoans) - BookNotEveryBranch$$

F. (10 points) List the *bookid* of each book that has not been borrowed even though the book has at least one copy.
Answer:

$$NoCopies \leftarrow \sigma_{Copies=0}(BookCopies)$$

$$Result \leftarrow \pi_{BookId}(Book) - \pi_{BookId}(BookLoans) - \pi_{BookId}(NoCopies)$$

G. (10 points) List the *title* of books not written by Bob and borrowed in two branches with one of them being in Illinois and the other in Indiana.
Answer:

$$NotBobBooksLoaned \leftarrow \sigma_{AuthName\neq'Bob'}(Author) \bowtie AuthorBook \bowtie Book \bowtie BookLoans$$

$$IllinoisLoaned \leftarrow \sigma_{State='Illinois'}(LibraryBranch) \bowtie NotBobBooksLoaned$$

$$IndianaLoaned \leftarrow \sigma_{State='Indiana'}(LibraryBranch) \bowtie NotBobBooksLoaned$$

$$BothLoaned \leftarrow IllinoisLoaned \cap IndianaLoaned$$

$$Result \leftarrow \pi_{Title}(BothLoaned)$$

2. (10 points) Consider two simple relations $R_1(A, B)$ and $R_2(B, C)$. Which of the following relational algebra expressions is not equivalent to the other three? Provide a counterexample to show that your selected choice is not equivalent to one of the other three choices. The counterexample should include a data instance where the two compared relational algebra expressions return different results (to prove the inequivalence).

   A. $\pi_{A,B}(R_1 \bowtie R_2)$

   B. $R_1 \bowtie (\pi_B(R_2))$

   C. $R_1 \cap (\pi_A(R_1) \times \pi_B(R_2))$

   D. $\pi_{A,R_2.B}(R_1 \times R_2)$

   Answer: D

   D is not equivalent to A. Suppose we have following data instance for the two relations:

   R_1 :

A	B
a_1	b_1
a_2	b_2

   R_2 :

B	C
b_1	c_1
b_3	c_2

   In query A, a natural join on R_1 and R_2 will result in:

A	B	C
a_1	b_1	c_1

. Projecting on attributes A and B will generate the result of query A as:

A	B
a_1	b_1

.

   In query D, the Cartesian-product on R_1 and R_2 will result in:

A	$R_1.B$	$R_2.B$	C
a_1	b_1	b_1	c_1
a_1	b_1	b_3	c_2
a_2	b_2	b_1	c_1
a_2	b_2	b_3	c_2

.

   Projecting on attribute A and $R_2.B$ generates the result of query D as:

A	B
a_1	b_1
a_1	b_3
a_2	b_1
a_2	b_3

   So query A and query D are inequivalent.

3. (30 points) Consider the following schema of car dealerships. For each of the following queries, write one equivalent relational algebra expression.

   Dealers(**did: int**, *dname: string*, *dcity: string*)
   Cars(**cid: int**, *cname: string*, *cmake: string*, *ctype: string*)
   Selling(**did: int**, **cid: int**, *sprice: int*)

   A. (10 points) $\pi_{dname}(\sigma_{cmake='Nissan' \wedge sprice<=20000}(Dealers \bowtie Cars \bowtie Selling))$
   Answer:
   Varies. One possible mutation is:
   $\pi_{dname}(((\sigma_{cmake='Nessan'}(Cars)) \bowtie (\sigma_{sprice<=20000}(Selling))) \bowtie Dealers)$.

   B. (10 points)

```
SELECT dname FROM Dealers d, Cars c, Selling s
WHERE c.cmake = 'Jeep' AND s.sprice > 15000
AND d.did = s.did AND c.cid = s.cid;
```

Answer:

$\pi_{dname}((\sigma_{cmaker='Jeep'}(Cars)) \bowtie (\sigma_{sprice>15000}(Selling)) \bowtie Dealers)$

C. (10 points)

```
SELECT dname FROM Dealers d, Selling s1, Cars c
WHERE s1.cid = c.cid AND d.did = s1.did
AND c.cmake = 'Ford' AND NOT EXISTS
(SELECT did FROM Selling s2, Cars c2
WHERE s2.cid = c2.cid AND c2.cmake = 'Jeep'
AND s2.did = s1.did);
```

Answer:

$SellFord \leftarrow \pi_{did}((\sigma_{cmake='Ford'}(Cars)) \bowtie Selling)$
$SellJeep \leftarrow \pi_{did}((\sigma_{cmake='Jeep'}(Cars)) \bowtie Selling)$
$Result \leftarrow \pi_{dname}((sellFord - sellJeep) \bowtie Dealers)$

4. (20 points) Consider the following schema of a student scholarships database.

Students(**sid: int**, *sname: string*, *birthdate: datetime*)
Courses(**cid: int**, *cname: string*, *description: string*)
Grades(**sid: int**, **cid: int**, *grade: int*)
Scholarship(**sid: int**, *updateYear: int*)

Suppose students need to maintain an average grade of at least 90 (on a 0-100 scale) to be considered for a scholarship. Write a trigger that does the following:

1. Check the student's new avg grade once a new grading entry is inserted into the Grades table.
2. If the student becomes eligible, add the student into the Scholarship table with updateYear = 2020.
3. If the student lost eligibility, remove the student from the Scholarship table.

Answer:

```
delimiter $$
create trigger after_insert_grade after insert on Grades
for each row
begin
    if (select avg(grade) from Grades group by sid having sid =
       NEW.sid) < 90 AND exists(select * from Scholarship
      where sid = NEW.sid) then
```

```
        delete from Scholarship where sid = NEW.sid;
    end if;
    if (select avg(grade) from Grades group by sid having sid =
        NEW.sid) >= 90 AND not exists(select * from Scholarship
        where sid = NEW.sid) then
        insert into Scholarship values(NEW.sid, 2020);
    end if;
end;
$$
delimiter ;
```

5. (30 points) Consider the following schema:

   Organization (**OrgId**, *OrgName*, *HeadId*)
   SubOrg (**OrgId**, **SubOrgId)**
   Employee (**EmpId**, *Name*, *Job*, *Salary*, *OrgId*)

   Write a stored procedure to compute the total salaries for a specific college and its departments. Given a specific college (e.g., College of Science), the stored procedure should generate an XML report (see format and example below) that shows the hierarchical structure of the college, the head of the college and each department, the total salaries for each unit, and the names and salaries for all of the employees in each department. At the college level, the total salaries include the college head and all department employees. No recursion is required for this procedure. The report will always contain the college, department, and employee levels. The report will have the following XML format, which can be stored and returned in a text variable:

```
<college>
        <info>college_name, head_name, total_salaries</info>
        <Departments>
                <Department>
                        <info>dept_name, head_name, total_salaries</info>
                        <Employees>
                                <Employee>employee_name, salary</Employee>
                                ...
                        </Employees>
                </Department>
                ...
        </Departments>
</college>
```

   An example report for "College of Science" would look like the following (the example is based on the data included in **tables.sql** and **data.sql** provided with this homework):

```
<college>
   <info>College of Science, Smith, 850000</info>
```

```
    <Departments>
        <Department>
            <info>Dept of Computer Science, Fei, 330000</info>
            <Employees>
                <Employee>Fei, 130000</Employee>
                <Employee>Mary, 75000</Employee>
                <Employee>Michael, 125000</Employee>
            </Employees>
        </Department>
        <Department>
            <info>Dept of Statistics, Bob, 370000</info>
            <Employees>
                <Employee>Bob, 130000</Employee>
                <Employee>Suresh, 120000</Employee>
                <Employee>Ali, 120000</Employee>
            </Employees>
        </Department>
    </Departments>
</college>
```