COMP207 Assignment 2 – Query Processing

Issue Date: Monday, 18 November 2019

Submission Deadline: Tuesday, 03 December 2019, 17:00

About This Assignment

This is the second of two assignments for COMP207. It is worth 10% of the total marks for this module. It consists of four questions, which you can find at the end of this document.

Submit your solutions to these questions in PDF format by the given submission deadline. Your solutions must be submitted on Vital (see the detailed submission instructions below).

Accuracy and relevance are more important in your answers, so don't write large volumes in your submission, but do ensure that what you write covers what is asked for and keeps to the problem statement.

Submission Details

Please submit **one PDF file with your solutions**. Name your file as follows:

<your student ID>-Assignment-2.pdf

If your student ID is 12345678, then your file should be named:

12345678-Assignment-2.pdf.

Please submit only this file (no archives).

To act as your 'signature' for the assignment, at the top of your PDF document put your Student ID number.

Your solutions must be submitted on Vital (see Vital for submission instructions).

The submission deadline for this assignment is **Tuesday, 03 December 2019, 17:00**. Earlier submission is possible, but any submission after the deadline attracts the standard lateness penalties. Plagiarism and collusion guidelines will apply throughout the assignment submission. For details on late submissions, how to claim extenuating circumstances, etc., please see the undergraduate student handbook, which can be found at http://intranet.csc.liv.ac.uk/student/ug-handbook.pdf , or in Section 6 of the Code of Practice on Assessment.[1]

Assessment information at a glance

Assignment Number	2 (of 2)
Weighting	10% of the final module mark
Assignment Circulated	Monday, 18 November 2019
Deadline	Tuesday, 03 December 2019, 17:00
Submission Mode	Electronically on Vital
Learning Outcome Assessed	LO2: Demonstrate an understanding of advanced SQL topics
Purpose of Assessment	Assessment of knowledge of SQL query processing
Marking Criteria	See description of this assignment
Submission necessary in order to satisfy module requirements?	N/A
Late Submission Penalty	Standard UoL Policy

[1] https://www.liverpool.ac.uk/media/livacuk/tqsd/code-of-practice-on-assessment/code_of_practice_on_assessment.pdf

Question 1 (10 marks)

The following tables form part of a hotel booking database held in a relational DBMS (primary keys are underlined):

Hotel (hotelNo, hotelName, city)

Room (roomNo, hotelNo, type, price)

Booking (hotelNo, guestNo, dateFrom, dateTo, roomNo)

Guest (guestNo, guestName, guestAddress)

- **Hotel** contains hotel details and hotelNo is the primary key.
- **Room** contains room details for each hotel and (roomNo, hotelNo) forms the primary key.
- **Booking** contains details of the bookings and (hotelNo, guestNo, dateFrom) forms the primary key.
- **Guest** contains guest details and guestNo is the primary key.

Give the relational algebra expressions to return the results for the following two queries:

(a) List the names and cities of those hotels who charge more than £85 for a room. **(5 marks)**

(b) List the names and addresses of guests who have made a booking to stay Christmas Day 2019. **(5 marks)**

Solutions

(a) $\pi_{\text{hotelName,city}}$ (Hotel $\bowtie$ ($\sigma_{\text{price} > 85}$ (Room)))

(b) π guestName, guestAddress (Guest $\bowtie$ ($\sigma_{(\text{dateFrom} \leq \text{‘25-Dec-2019’} \wedge \text{dateTo} \geq \text{‘25-Dec-2019’})}$ (Booking)))

Question 2 (10 marks)

Consider the following database schema and example instance for a property management system:

property

pId	price	owner	sqrFeet	location
1	100,000	Alice	560	Lake View
2	3,400,000	Bob	2,000	Hyde Park
3	1,200,000	Bob	1,200	Hyde Park
4	5,000,000	Martha	800	Evanston

repairs

rId	pId	company	date	type
1001	1	M.M. Plumbing Ltd.	2013-12-12	Bathroom
1002	2	M.M. Plumbing Ltd.	2013-12-13	Kitchen
1003	4	Rob's Double Glazing	2012-01-01	Windows

Hints:

- Attributes with a grey background form the primary key of a relation (e.g, *pId* for relation *property*).
- The attribute *pId* of relation *repairs* is a foreign key to relation *property*.

Give the relational algebra expressions to return the results for the following two queries:

(a) Get the pId, owner and location details of all properties that are larger than 900 square feet (sqrFeet). **(5 marks)**

(b) Get the names of repair companies (company) that did a repair on a property in Hyde Park. **(5 marks)**

Solutions

(a) $\pi_{\text{pId, owner, location}}(\sigma_{\text{sqrFeet} > 900}(\text{property}))$

(b) $\pi_{\text{company}}(\sigma_{\text{location} = \text{'Hyde Park'}}(\text{property}) \bowtie \text{repairs})$

Question 3 (20 marks)

(a) Consider the following relation:
studentCourses(StudentID, CourseNo, Quarter, Year, Units, Grade)

The relation contains the grades for the courses completed by students. Assume that in studentCourses there are 200,000 different students, each identified by their StudentID. On average, a student took 40 different courses.

If the file blocks hold 2000 bytes and each studentCourses tuple requires 50 bytes, how many blocks will then be needed to store the relation studentCourses? **(5 marks)**

(b) A database includes two relations Student (S) and Program (P).

S

Student_No	**F_Name**	**L_Name**	**Prog_Code**
04009991	Alicia	Smith	0001
04009992	Alan	Smith	0002
04009995	Alicia	Bush	0001
04009996	John	Smith	0001

P

Prog_Code	**P_Name**
0001	Computing
0002	Software Engineering

Give a relational expression that could possibly return the following result:

F_Name	L_Name	P_Name
Alicia	Smith	Computing
John	Smith	Computing

(5 marks)

(c) Translate the following relational algebra into SQL:

πstudId,lName(σ course='BSc'(STUDENT)) **(5 marks)**

(d) Given these relations, write the SQL statement that produced the equivalent queries below:

Course (<u>courseNo</u>, courseDept, courseLeader)

Student (<u>studNo</u>, name, type, tutorId, courseNo)

Two sample equivalent corresponding queries have been produced:

πstudno,name(σ(type='undergrad')^(courseDept='CompSci')(Student$\bowtie$s.courseNo=c.courseNo Course))
and
πstudno,name(σ type='undergrad'(Student))$\bowtie$s.courseNo=c.courseNo(σ courseDept='Comp Sci'(Course))

(5 marks)

Solutions

(a) (200000 * 40 * 50) / 2000 = 200000

(b) There are multiple correct answers to this question, one possible answer is:

πF_Name, L_Name, P_Name (σL_Name = 'Smith' AND P_Name = 'Computing' (S ⋈ P))

(c)
```
SELECT studId, lName
FROM Student
WHERE course = 'BSc';
```

(d)
```
SELECT     studNo, name
FROM       Student s, Course c
WHERE      s.courseNo=c.courseNo AND
(s.type='undergrad' AND c.courseDept='CompSci');
```

Question 4 (60 marks)

Consider a database with relations $R(A, B, C)$, $S(D, E)$, and $T(F, G)$.

(a) Give the initial query plan (constructed as in Lecture 13) for the SQL query

SELECT B, E, G FROM R, S, T
WHERE A = 10 AND C = D AND E = F AND A > G;

Then use the heuristics from Lecture 16 to transform the initial query plan into an optimised (logical) query plan. Perform the transformation step-wise, pushing a single operator over a single operator in each step, and indicate the heuristics you apply. **(20 marks)**

(b) Suppose that

- $|R| = 1000$, $|\pi_A(R)| = 1000$, $|\pi_B(R)| = 100$, $|\pi_C(R)| = 500$;
- $|S| = 5000$, $|\pi_D(S)| = 300$, $|\pi_E(S)| = 10$;
- $|T| = 4000$, $|\pi_F(T)| = 4000$, $|\pi_G(T)| = 1500$.

Estimate the number of tuples returned by the following queries. Explain your calculations.

i) $\sigma_{A=10}(R)$ **(6 marks)**

ii) $\sigma_{A=10 \text{ OR } B=b}(R)$ **(6 marks)**

iii) $R \bowtie_{C=D} S$ **(6 marks)**

(c) Suppose that in addition to the assumptions on R, S, and T from part (ii), we also have the following:

- Each disk block can hold up to 10 tuples.
- All relations are stored in consecutive blocks on disk.
- No indexes are available.

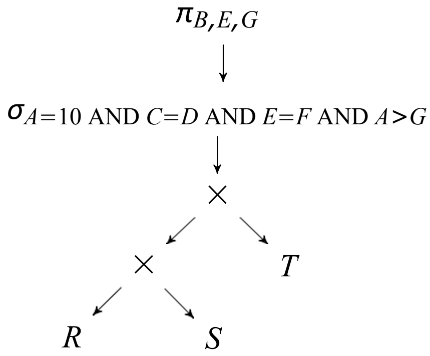
What is the best physical query plan (in terms of the number of disk access operations) you can find for $\sigma_{B=b \text{ AND } E=100}(R \bowtie_{C=D} S)$? Describe your plan and show the calculation of the number of disk access operations. **(22 marks)**

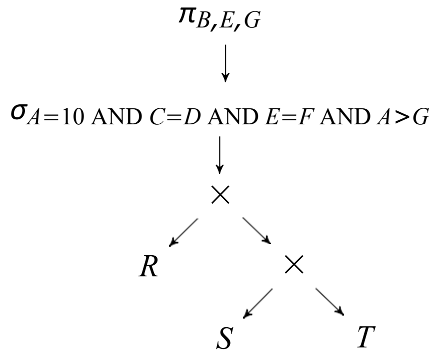
Solutions

(a) Initial query plan. The initial query plan is:

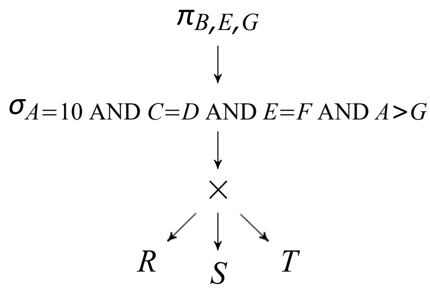
$$\pi_{B,E,G}(\sigma_{A=10 \text{ AND } C=D \text{ AND } E=F \text{ AND } A>G}(R \times S \times T))$$

Possible representations in tree form:



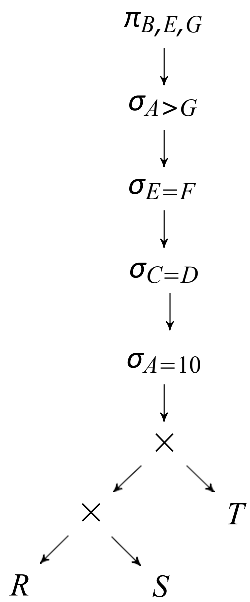


We have not introduced Cartesian products for three or more relations, so the following is *not* a correct initial query plan:

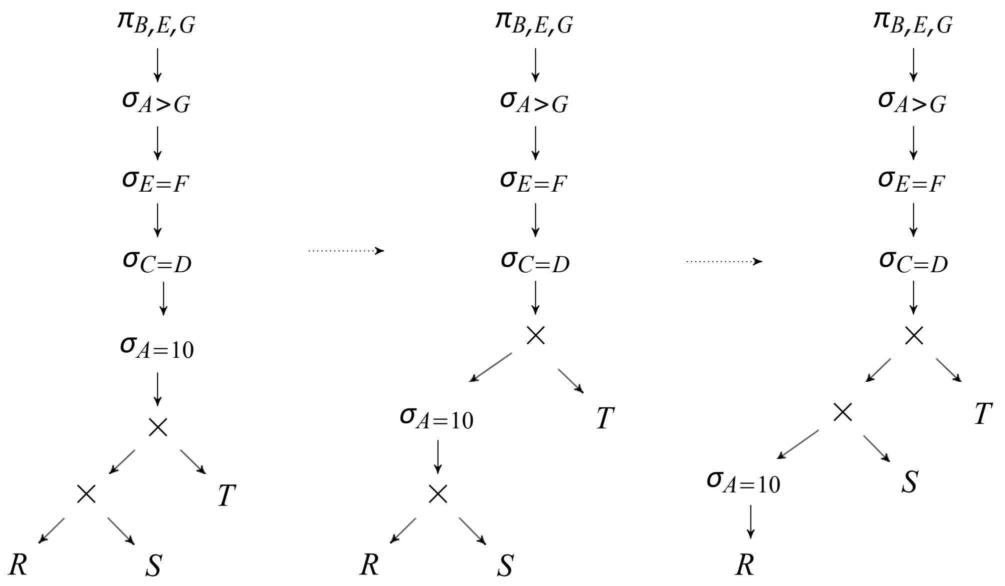


In what follows, we use the initial query plan on the left.

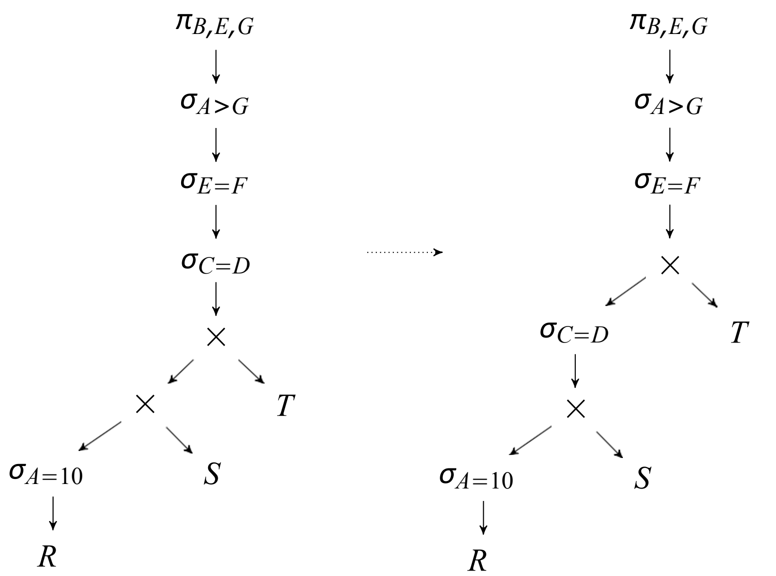
Applying heuristics. We first split the selection into selections with atomic selection conditions, which results in the following plan (the order of the selections is arbitrary):



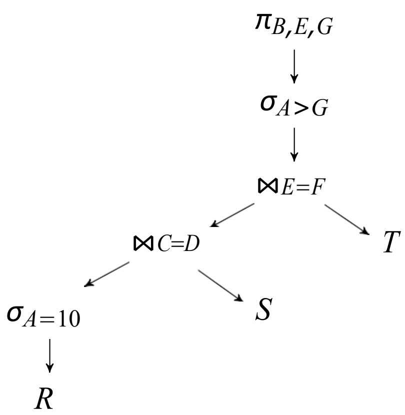
We now push $\sigma_{A=10}$ downwards to R:



We then push $\sigma_{C=D}$ downwards one node:



We now combine $\sigma_{C=D}$ and $\sigma_{E=F}$ with the Cartesian products below them into equijoins:



It remains to introduce projections wherever appropriate:

$$
\begin{tikzcd}
 & & \pi_{B,E,G} \arrow[d] & \\
 & & \sigma_{A>G} \arrow[d] & \\
 & & \pi_{A,B,E,G} \arrow[d] & \\
 & & \bowtie_{E=F} \arrow[dl] \arrow[dr] & \\
 & \pi_{A,B,E} \arrow[d] & & T \\
 & \bowtie_{C=D} \arrow[dl] \arrow[dr] & & \\
\sigma_{A=10} \arrow[d] & & S & \\
R & & &
\end{tikzcd}
$$

(b) The calculations for a) and c) use the rules for the estimation of intermediate result sizes from Lecture 16:

a) We estimate $|\sigma_{A=10}(R)|$ using the rule for the estimation of the size of selections from Lecture 16:

$$|\sigma_{A=10}(R)| = \frac{|R|}{|\pi_A(R)|} = \frac{1000}{1000} = 1$$

b) We have to generalise the method of estimation from lecture 16. The method is based on the assumption that in each column of a relation, all values occur equally often, so to estimate $\sigma_{A=10 \text{ OR } B=b}(R)$, we start by assuming:

- Value 10 occurs $|R|/|\pi_A(R)| = 1000/1000 = 1$ time in column A of R.
- Value b occurs $|R|/|\pi_B(R)| = 1000/100 = 10$ times in column B of R.

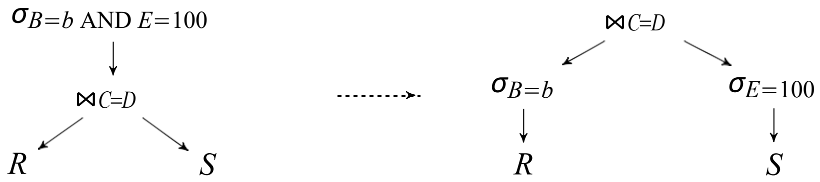
Thus, $\sigma_{A=10 \text{ OR } B=b}(R)$ is estimated to contain 10 or 11 tuples, depending on whether the tuple $t \in R$ with $t[A] = 10$ satisfies $t[B] = b$ or not.

c) We estimate $|R \bowtie_{C=D} S|$ using the rule for the estimation of the size of natural joins from Lecture 16, translated to equijoins in the obvious way:

$$|R \bowtie_{C=D} S| = \frac{|R| \cdot |S|}{\max\{|\pi_C(R)|, |\pi_D(S)|\}} = \frac{1000 \cdot 5000}{\max\{500, 300\}} = \frac{1000 \cdot 5000}{500} = 10000.$$

In general, any well justified estimate is fine. Here, well justified means that you either base your calculations (and show how you perform these calculations) on hints given during the lectures, on methods from the literature (with references, and an explanation of why those methods are suitable for this question), or on assumptions that you make yourself (this requires an explanation of your assumptions, including when are they good, when not, and why they might be reasonable).

(c) We may start with an optimised logical query plan:



Since no indexes are available, the best we can do in order to compute the two selections is to scan of the input relations (i.e., by reading the tuples of R and S one after the other). Since R and S are stored in consecutive blocks on disk and each disk block can hold 10 tuples, this requires $|R|/10 = 100$ read operations for $\sigma_{B=b}(R)$ and $|S|/10 = 500$ read operations for $\sigma_{E=100}(S)$.

Since we don't know if R is sorted on C or if S is sorted on D, the best we can do in order to pass the tuples of the selections to the join is to first write the result of the selections to disk and to pass the location of the files containing these tuples to the join algorithm. The estimated number of tuples in $\sigma_{B=b}(R)$ is $|R|/|\pi_B(R)| = 10$, so writing $\sigma_{B=b}(R)$ to disk requires $10/10 = 1$ write operation (in practice, one would not write out this block to disk, but keep it in memory, so if your query plan uses this optimisation, this is fine, but requires some justification). Similarly, the estimated number of tuples in $\sigma_{E=100}(S)$ is $|S|/|\pi_E(S)| = 500$, so writing $\sigma_{E=100}(S)$ to disk requires $500/10 = 50$ write operations.

We can now choose the Sort Join Algorithm to compute the join. Thus, we first sort the input relations. If we do this using External Memory Merge Sort, this requires about

$$O\left(\frac{|R|}{B}\log_M\frac{|R|}{B} + \frac{|S|}{B}\log_M\frac{|S|}{B}\right) = O\left(\frac{1000}{B}\log_M\frac{1000}{B} + \frac{5000}{B}\log_M\frac{5000}{B}\right)$$

disk access operations. Then, we use MergeJoin, which requires no more than

$$\frac{|\sigma_{B=b}(R)|}{10} + (|\sigma_{B=b}(R)| + 1)\cdot\frac{|\sigma_{E=100}(S)|}{10}$$

read and write operations. Note that the latter number of disk access operations is slightly worse than the number of disk access operations required by Nested Loop Join. We would only reach this number if there are very few values in column B of R that can be joined with most tuples in S. If the database system knows or assumes this, then it could select Nested Loop Join or a different join algorithm such as Nested Block Loop Join (not covered in the lectures) to compute the join and to avoid the overhead of sorting. In all other cases, the implementation using Sort Join yields a lower number of disk access operations.

Instead of using Sort Join, one could also introduce a sort step between the selections and the join, and then use MergeJoin. This yields the same number of disk access operations.

Note

The above is a very good solution, however, the answer could still be further optimized by sorting R first lexicographically on B,C and S first lexicographically on E,D. The sorting takes just as many operations, but allows you to do the selection on sorted data and still have the sorted data after the selection (because you take everything with 1 value for B and E respectively).