INFS2200/7903 – Relational Database Systems

School of Information Technology and Electrical Engineering (ITEE), UQ

Tutorial 8

Question 1 Assume that relation R has attributes A, B, C, D, and relation S has attributes D, E, F. Which of the following are valid relational algebra transformations during query optimization and which are not?

Please explain the reason briefly when one is not a valid transformation; you do not need to give any explanation for valid transformations. Note that $\bowtie$ corresponds to the natural join operator on the common attribute (i.e., attribute D).

A. $\Pi_{A,B}(\Pi_{A,B,C,D}(R \bowtie S)) = \Pi_{A,B}(R \bowtie S)$

B. $\sigma_{A>15 \text{ or } B<10 \text{ or } C=20}(R) = \sigma_{A>15}(\sigma_{B<10}(\sigma_{C=20}(R)))$

C. $\Pi_{A,B}(\sigma_{C<12}(R)) = \sigma_{C<12}(\Pi_{A,B}(R))$

D. $\Pi_{A,F}(R \bowtie S) = (\Pi_{A}(R)) \bowtie (\Pi_{F}(S))$

Question 2 Consider the following relational schema and SQL query. The schema captures information about employees, departments, and company finances (organized on a per department basis).

```
Emp (eid:integer, did:integer, sal:integer, hobby:char(20))

Dept (did:integer, dname:char(20), floor:integer, phone:char(10))

Finance (did:integer, budget:real, sales:real, expenses:real)
```

Consider the following query:

```
SELECT D.dname, F.budget
FROM Emp E, Dept D, Finance F
WHERE E.did = D.did AND D.did = F.did
      AND D.floor = 1 AND E.hobby = 'camping';
```

A. List the join orders (i.e., orders in which pairs of relations can be joined to compute the query result) that a relational query optimizer will consider. (Assume that the optimizer follows the heuristic of never considering plans that require the computation of cross-products.)

B. For one of the join orders above, identify a relational algebra tree (or a relational algebra expression) that reflects the order of operations a query optimizer would choose.

C. Suppose that the following additional information is available: B+ tree indexes exist on Emp.did, Emp.sal, Dept.floor, Dept.did, and Finance.did. The system's statistics indicate that employees enjoy 200 different hobbies, and the company owns two floors in the building, with a uniform value distribution. There are a total of 50,000 employees and 5,000 departments (each with corresponding financial information) in the database. The DBMS used by the company has only one join method available: single nested loop join.

a. For each of the query's base relations (Emp, Dept, and Finance), estimate the number of tuples that would be initially selected from that relation if all of the non-join predicates on that relation were applied to it before any join processing begins.
b. Given your answer to the above question, which of the join orders considered by the query optimizer would be more efficient?

Question 3 Consider the following relational schemas and instances:

```
Student (SID, Name, Class, Major)

Student_Dir (SID, Address, Phone)
FK: (SID) → Student (SID)

Course (Course_No, Name, Level)

Course_Taken (Course_No, Term, SID, Grade)
FK: (Course_No) → Course (Course_No); (SID) → Student (SID)
```

Student

SID	Name	Class	Major
123	John	3	CS
124	Mary	3	CS
126	Sam	2	CS
129	Julie	2	Math

Student_Dir

SID	Address	Phone
123	333 Library St	555‑535‑5263
124	219 Library St	555‑963‑9635
129	555 Library St	555‑123‑4567

Course

Course_No	Name	Level
CS1520	Web Programming	UGrad
CS1555	Database Management Systems	UGrad
CS1550	Operating Systems	UGrad
CS1655	Secure Data Management and Web Applications	Ugrad
CS2550	Database Management Systems	Grad

Course_Taken

Course_No	Term	SID	Grade
CS1520	Fall 11	123	3.75
CS1520	Fall 11	124	4
CS1520	Fall 11	126	3
CS1555	Fall 11	123	4
CS1555	Fall 11	124	NULL
CS1550	Spring 12	123	NULL
CS1550	Spring 12	124	NULL
CS1550	Spring 12	126	NULL
CS1550	Spring 12	129	NULL
CS2550	Spring 12	124	NULL
CS1520	Spring 12	126	NULL

For each of the relational algebra expressions below, identify the expected arity (number of attributes), schema, and min/max cardinality (number of tuples) of the relation resulting from the query, without actually evaluating the query and based only on the schemas and cardinalities of the four given relations.

A. $\sigma_{\texttt{Term = 'Spring 12'}}(\texttt{Course\_Taken})$

B. `Course_Taken * Course`

Note that '' corresponds to the natural join operator on the common attribute (i.e., attribute Course_No).*

Answers for Question 1 are given below:

A. Valid.

B. Invalid. Because it is a disjunction query. For instance, records with A>15 or B<10 but C≠20 should appear in the query answer. However, under this transformation all those records will be eliminated when evaluating $\sigma_{C=20}(R)$.

C. Invalid. Because the projection does not contain the attributes in the selection condition (i.e., attribute C). Under this transformation, applying $\pi_{A,B}$ first will eliminate attribute C from the intermediate result and it will not be possible to apply the selection condition $\sigma_{C<12}$.

D. Invalid. Because the projection does not contain the attributes in the join condition (i.e., attribute D). The condition for a natural join is R.D=S.D, since D is the common attribute. However, under this transformation, applying π_A will eliminate attribute D from relation R. Similarly, applying π_F will also eliminate attribute D from relation S.

Answers for Question 2 are given below:

A. There are two join orders considered, assuming that the optimizer ignores cross-products:
- First possible join order: $((E \bowtie D) \bowtie F)$
- Second possible join order: $((D \bowtie F) \bowtie E)$

B. A query optimizer would typically push down the selection and projection as far down the tree as possible. For the join order $((E \bowtie D) \bowtie F)$, this would result in:

$$\Pi_{D.dname,F.budget}\Big(\big(\Pi_{E.did}(\sigma_{E.hobby='camping'}(E)) \bowtie \Pi_{D.did,D.dname}(\sigma_{D.floor=1}(D))\big) \bowtie \Pi_{F.budget,F.did}(F)\Big)$$

C. Given the additional information and statistics of the database:

a. Emp size = 50,000 records, E.hobby = ‘camping’
Resulting size = 50,000 * (1/200) = 250 records.

Dept size = 5,000 records, D.floor = 1
Resulting size = 5,000 * (1/2) = 2,500 records.

Finance size = 5,000 records, there are no non-join predicates
Resulting size = 5,000 records.

b. Plan $((E \bowtie D) \bowtie F)$ is more efficient because in that plan, the optimizer executes the most restrictive operations first.

   As opposed to plan $((D \bowtie F) \bowtie E)$, where there are no restrictions on the Finance relation.

Answers for Question 3 are given below:

A. Arity = Arity of Course_Taken = 4.

Schema = Schema of Course_Taken = (Course_No, Term, SID, Grade).
Cardinality = Cardinality of Course_Taken * Selectivity of $\sigma_{\texttt{Term = 'Spring 12'}}$

- Cardinality of Course_Taken = 11;
- Selectivity is in the range of 0 to 1;

Hence, Min Cardinality = 0 and Max Cardinality = 11.

B. Arity = Arity of Course_Taken + Arity of Course - number of common attributes = 4 + 3 - 1 = 6.

Schema = (Course_No, Term, SID, Grade, Name, Level).

Attribute Course_No is a foreign key of Course_Taken that refers to Course, which means that for every Course_Taken tuple there is exactly one matching Course tuple (because Course.Course_No is a primary key). Hence, Cardinality = Cardinality of Course_Taken = 11.