

Database Technology

Exercise 9: Query Optimization

9.1 Size of Join

Consider the relations $r_1(A, B, C)$, $r_2(C, D, E)$, and $r_3(E, F)$. Assume that r_1 has 1000 tuples, r_2 has 1500 tuples, and r_3 has 750 tuples. Estimate the size of $r_1 \bowtie r_2 \bowtie r_3$ and give an efficient strategy for computing the join in following situations:

- Assume that A , C , and E are primary keys.
- Assume that there are no primary keys, except the entire schema.
Let $V(C, r_1)$ be 900, $V(C, r_2)$ be 1100, $V(E, r_2)$ be 50, and $V(E, r_3)$ be 100.

9.2. Query plan

Consider the following query:

```
SELECT *  
FROM r, s  
WHERE UPPER(r.A) = UPPER(s.A)
```

where “upper” is a function that returns its input argument with all lowercase letters replaced by the corresponding uppercase letters.

- Find out what plan is generated for this query on the database system you use.
- Some database systems would use a (block) nested-loop join for this query, which can be very inefficient. Briefly explain how hash-join or merge-join can be used for this query.

9.3. Interesting orders

Consider the issue of interesting orders in optimization. Suppose you are given a query that computes the natural join of a set of relations S . Given a subset S_1 of S , what are the interesting orders of S_1 ?

9.4. Nested subquery rewriting

Consider again the following bank database:

```
branch(branch_name, branch_city, assets)
customer(customer_name, customer_street, customer_city)
loan(loan_number, branch_name, amount)
borrower(customer_name, loan_number)
account(account_number, branch_name, balance)
depositor(customer_name, account_number)
```

Construct the following SQL queries for this relational database.

- Write a nested query on the relation account to find, for each branch with name starting with B, all accounts with the maximum balance at the branch.
- Rewrite the preceding query, without using a nested subquery; in other words, decorrelate the query.
- Give a procedure for decorrelating such queries.

9.5. Equivalence of Relational Algebra Expressions

Show how to derive the following equivalences by a sequence of transformations using the equivalence rules:

- $\sigma_{\theta_1 \wedge \theta_2 \wedge \theta_3}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(\sigma_{\theta_3}(E)))$
- $\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_{\theta_3} E_2) = \sigma_{\theta_1}(E_1 \bowtie_{\theta_3} (\sigma_{\theta_2}(E_2)))$, where θ_2 involves only attributes from E_2
- $\pi_{L_1}(\pi_{L_2}(\sigma_{\theta_1}(E_1 \times E_2)) \cup \sigma_{\theta_2 \wedge \theta_3}(E_3)) = \pi_{L_1}(\sigma_{\theta_3}(\sigma_{\theta_2}(E_3))) \cup \pi_{L_1}(E_2 \bowtie_{\theta_1} E_1)$