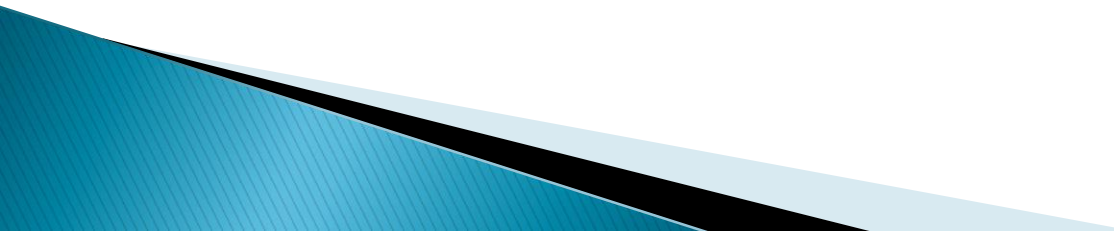


Query Processing, Chapter 23

Query Optimization



Query Optimization Example

Find all Managers who work at a London branch.

```
SELECT *  
FROM Staff, Branch  
WHERE Staff.branchNo=Branch.branchNo  
AND (Staff.position='Manager' AND  
      Branch.city='London');
```

Query Optimization Example

```
SELECT *  
FROM Staff, Branch  
WHERE Staff.branchNo=Branch.branchNo AND  
      (Staff.position='Manager' AND Branch.city='London');
```

Three equivalent relational algebra statements:


- 1.) $\sigma_{(\text{position}='Manager') \text{ AND } (\text{city}='London') \text{ AND } (\text{Staff.branchNo}=\text{Branch.branchNo})} (\text{Staff} \times \text{Branch})$
- 2.) $\sigma_{(\text{position}='Manager') \text{ AND } (\text{city}='London')} (\text{Staff} \bowtie_{\text{Staff.branchNo}=\text{Branch.branchNo}} \text{Branch})$
- 3.) $(\sigma_{\text{position}='Manager'} (\text{Staff})) \bowtie_{\text{Staff.branchNo}=\text{Branch.branchNo}} (\sigma_{\text{city}='London'} (\text{Branch}))$


Query Optimization Example

Say:

- ▶ Staff table has 1,000 tuples
- ▶ Branch has 50 tuples
- ▶ 50 of the staff people are managers (one for each branch)
- ▶ There are 5 branches in London

1.) $\sigma_{(\text{position}='Manager') \text{ AND } (\text{city}='London') \text{ AND } (\text{Staff.branchNo}=\text{Branch.branchNo})}$ (Staff x Branch)

2.) $\sigma_{(\text{position}='Manager') \text{ AND } (\text{city}='London')}$
(Staff  Staff.branchNo=Branch.branchNo Branch)

3.) $(\sigma_{\text{position}='Manager'}(\text{Staff}))$  Staff.branchNo=Branch.branchNo
 $(\sigma_{\text{city}='London'}(\text{Branch}))$

Query Optimization Example

Assume:

- ▶ No indexes
- ▶ Intermediate results are stored on disk

Compare these queries in terms of disk accesses.

Query Optimization Example

- ▶ Staff table has 1,000 tuples
- ▶ Branch has 50 tuples
- ▶ 50 of the staff people are managers (one for each branch)
- ▶ There are 5 branches in London

1.) $\sigma_{(\text{position}='Manager') \text{ AND } (\text{city}='London') \text{ AND } (\text{Staff.branchNo}=\text{Branch.branchNo})}$ (Staff x Branch)

(1,000+50) access to read the tuples

Create a relation with 50,000 tuples (which are unrealistically written back to the disk)

Read each of these to compare with the search criteria giving a total cost of:

$$(1,000+50) + 2*(50,000) = 101,050 \text{ disk accesses}$$

Query Optimization Example

- ▶ Staff table has 1,000 tuples
- ▶ Branch has 50 tuples
- ▶ 50 of the staff people are managers (one for each branch)
- ▶ There are 5 branches in London

2.) $\sigma_{(\text{position}='Manager') \text{ AND } (\text{city}='London')}$
(Staff \bowtie Staff.branchNo=Branch.branchNo Branch)

(1,000+50) access to read the tuples

Join makes 1,000 records, written to disk (a staff member can only work at one branch)

Must then check each against the selection criteria giving a total cost of:

$$(1,000+50) + 2*(1,000) = 3,050 \text{ disk accesses}$$

Query Optimization Example

- ▶ Staff table has 1,000 tuples
- ▶ Branch has 50 tuples
- ▶ 50 of the staff people are managers (one for each branch)
- ▶ There are 5 branches in London

3.) $(\sigma_{\text{position}='Manager'}(\text{Staff})) \bowtie_{\text{Staff.branchNo}=\text{Branch.branchNo}}$
 $(\sigma_{\text{city}='London'}(\text{Branch}))$

Read each Staff tuple to determine the managers – 1,000 reads and write back the result of 50

Second read each branch tuple and determine the London branches – 50 reads and write back the 5 results

Get the 50 + 5 results and join them giving a total cost of:
 $(1000 + 50) + (50 + 5) + (50 + 5) = 1,160$ disk accesses

Query Optimization

Create a tree of the query and use the following heuristic strategies:

- ▶ Perform selection operations as early as possible
 - ▶ Combine the Cartesian product with a subsequent Selection operation whose predicate represents a join condition into a Join operation.
 - ▶ When possible rearrange Selection operations so the most restrictive Selection operations are executed first.
 - ▶ Perform Projection operations as early as possible.
 - ▶ Compute common expressions once.
- 