CS354: Database

Query Processing Introduction

- Some database operations are expensive
- Performance can be improved by being "smart"
  - Clever implementation techniques for operators
  - Exploiting "equivalences" of relational operators
  - Using statistics and cost models to choose better plans

Basic Steps in Query Processing

- Parse and translate: convert to RA query
- Optimize RA query based on the different possible plans
- Evaluate the execution plan to obtain the query results

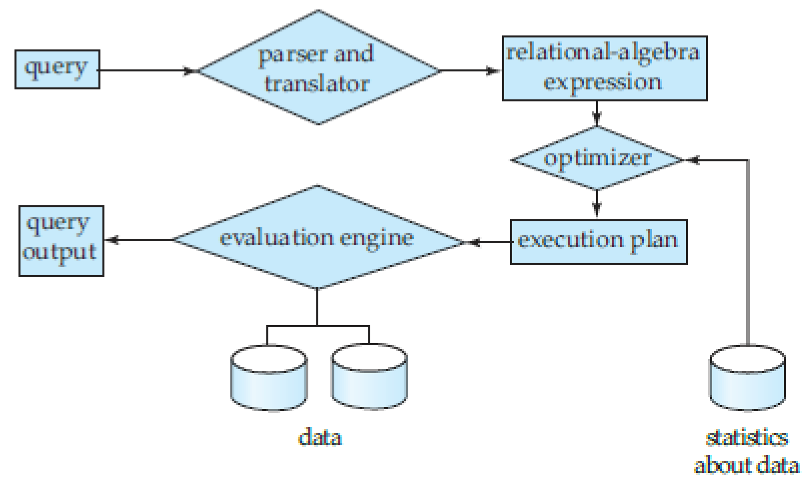
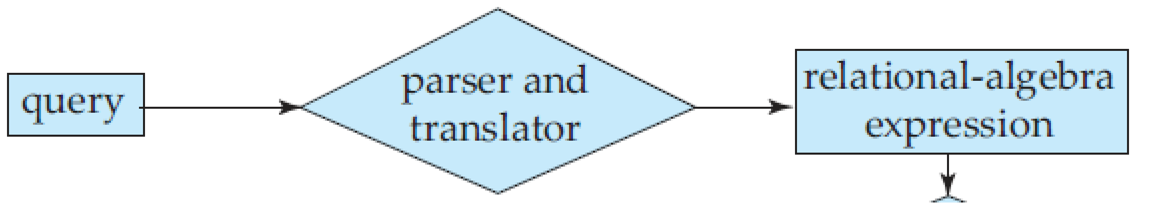


Figure 12.1 from Database System Concepts book

Query Processing Overview



WHY?

SQL

A **declarative** expression of the query result

Codd's theorem

RA

Operational description of a computation

Say what you want, not how to get it!

Systems optimize and execute RA query plan!

Example: SQL Query

Find movies with stars born in 1960

```
SELECT movieTitle
FROM StarsIn, MovieStar
WHERE starName = name
AND birthdate LIKE ‘%1960’;
```

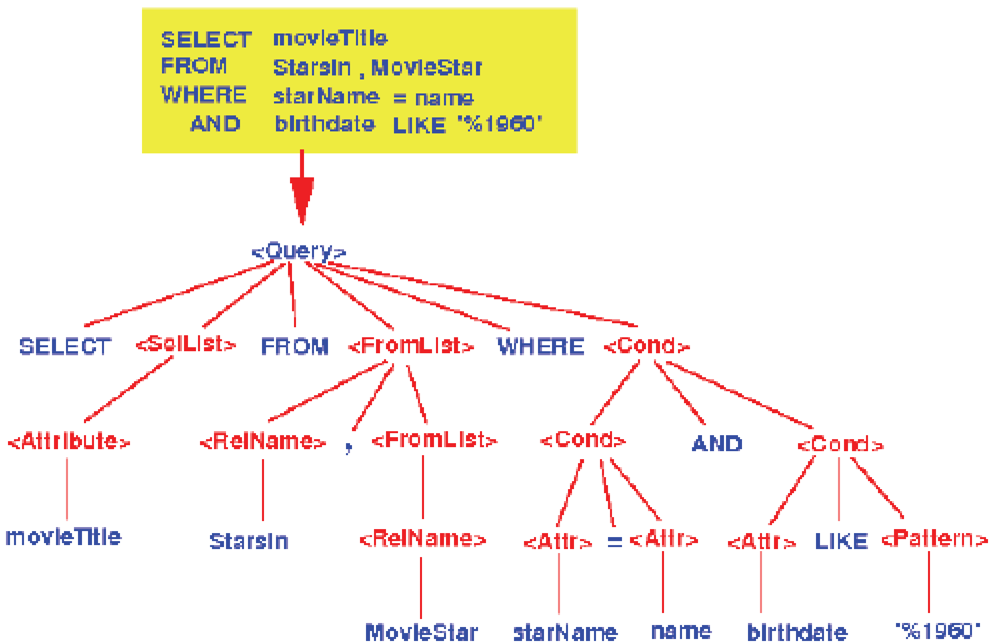
Example: Bad Query Optimization

- Cartesian product first:
  StarsIn x MovieStar
- Selection criteria next:
  starname = name AND birthdate LIKE ‘%1960’
- GROUP BY; HAVING (if available)
- Projections
  SELECT movietitle
- ORDER BY last

Incredibly inefficient with huge intermediate results!

Example: SQL Query Step 1

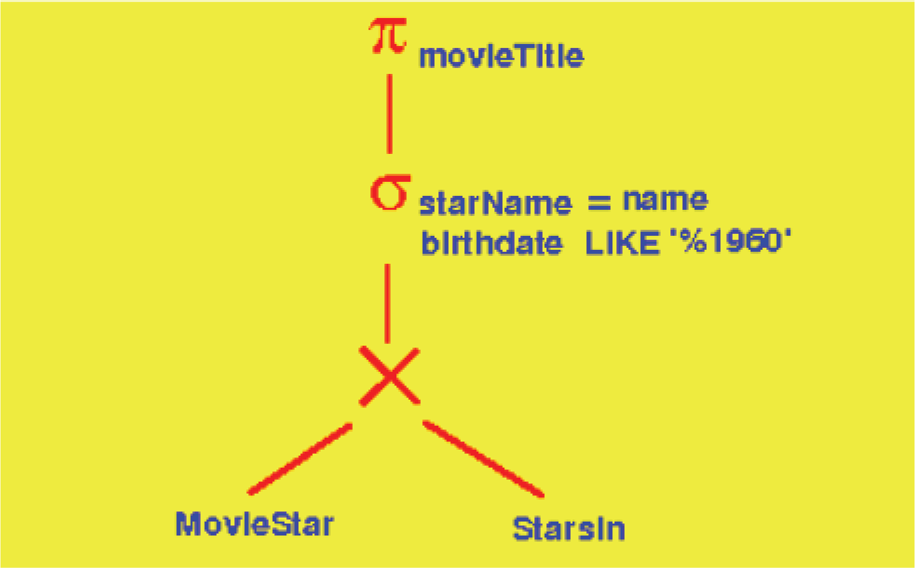
Step 1: Convert SQL query into a parse tree



http://www.mathcs.emory.edu/~cheung/Courses/554/Syllabus/5-query-opt/intro.html

Example: SQL Query Step 2

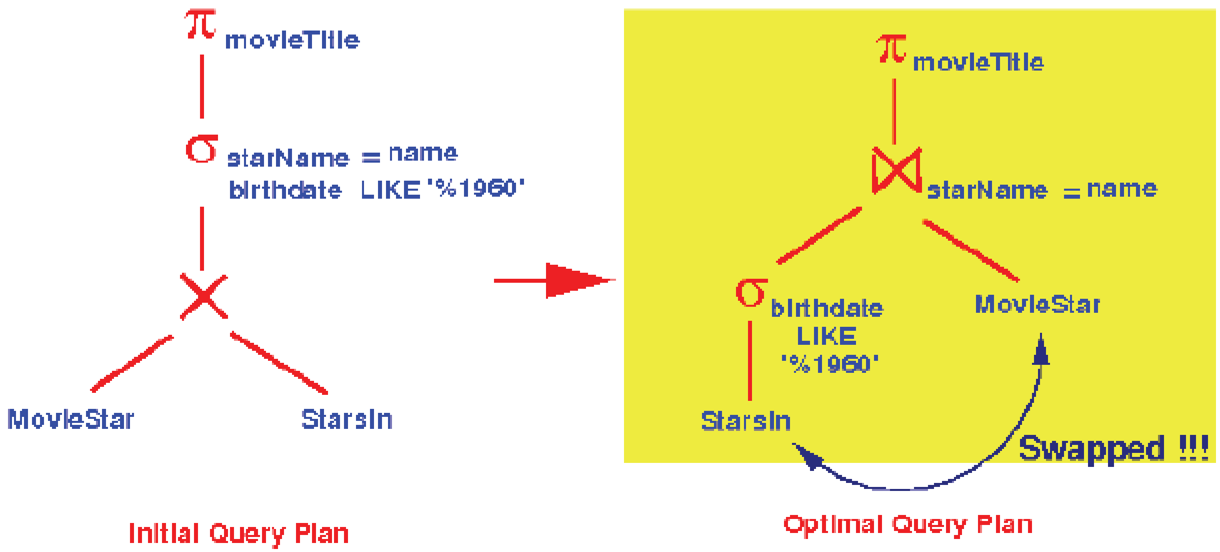
Step 2: Convert parse tree into initial logical query plan using RA expression



http://www.mathcs.emory.edu/~cheung/Courses/554/Syllabus/5-query-opt/intro.html

Example: SQL Query Step 3

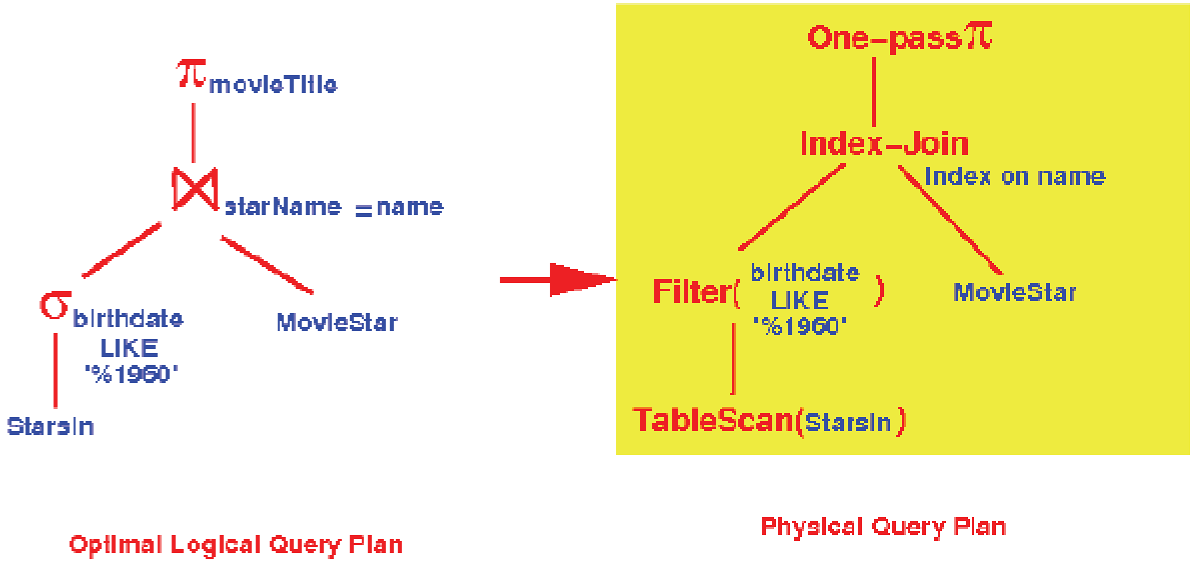
Step 3: Transform initial plan into optimal query plan using some measure of cost to determine which plan is better



http://www.mathcs.emory.edu/~cheung/Courses/554/Syllabus/5-query-opt/intro.html

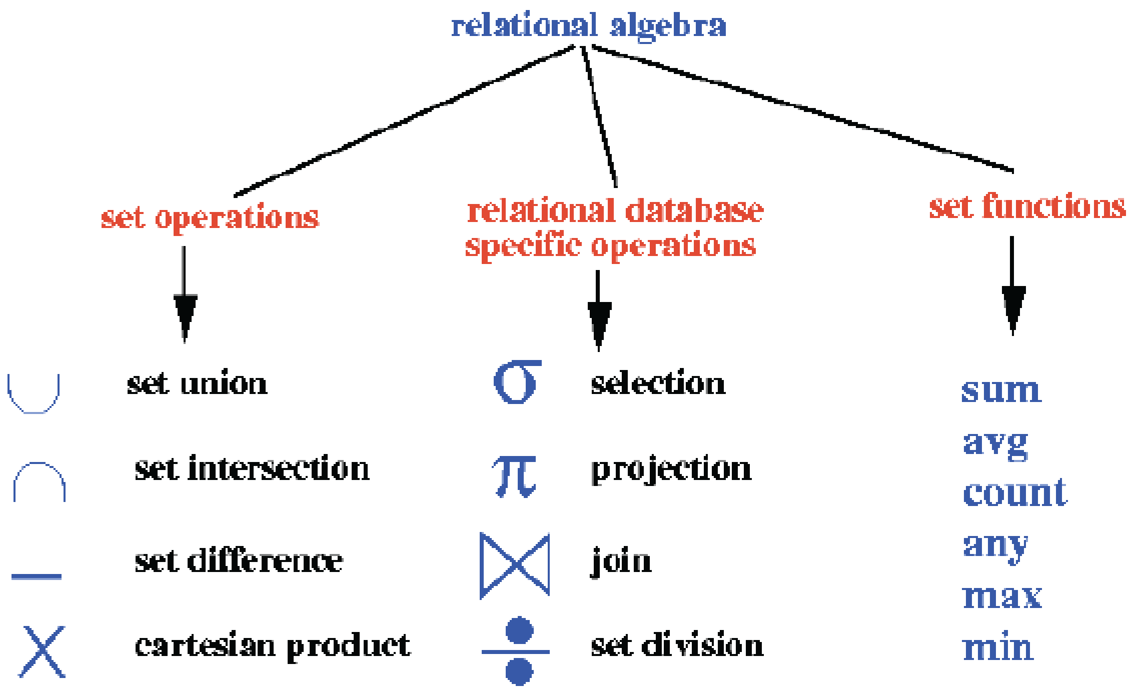
Example: SQL Query Step 4

Step 4: Select physical query operator for each relational algebra operator in the optimal query plan



http://www.mathcs.emory.edu/~cheung/Courses/554/Syllabus/5-query-opt/intro.html

Recap: Relational Algebra



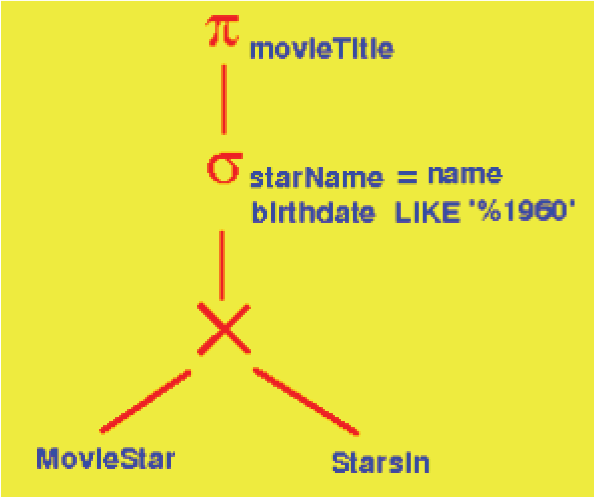
Recap: SQL Query to RA

- How do you represent queries in RA?
- Database: Students(sid, sname, gpa)
  People(ssn, pname, address)
- SQL query:
  SELECT DISTINCT gpa, address
  FROM Students, People
  WHERE gpa > 3.5 AND sname = pname;
- RA query:

$$\pi_{\text{gpa,address}}(\sigma_{\text{gpa}>3.5}(\text{Students} \bowtie_{\text{sname=name}} \text{People}))$$

Query Tree (Plan)

- A tree data structure that corresponds to a relational algebra expression
  - Leaf nodes = input relations
  - Internal nodes = RA operations
- Execution of query tree
  - Start at the leaf nodes
  - Execute internal node whenever its operands are available and replace node by result



Query Optimization Heuristics

- Apply heuristic rules on standard initial query tree to find optimized equivalent query tree
- Main heuristic: Favor operations that reduce the size of intermediate results first
    - Apply SELECT and PROJECT operations before join or other set operations
    - Apply more selective SELECT and join first
- General transformation rules for relational algebra operators

RA Transformation Rules

- Selection cascade: conjunctive selection condition can be broken into sequence of individual operations

$$\sigma_{\mathrm{c1\ AND\ c2\ AND\ \cdots AND\ cn}}(R) = \sigma_{\mathrm{c1}}(\sigma_{\mathrm{c2}}(\cdots(\sigma_{\mathrm{cn}}(R))\cdots))$$

- Commutativity of selection

$$\sigma_{\mathrm{c1}}(\sigma_{\mathrm{c2}}(R)) = \sigma_{\mathrm{c2}}(\sigma_{\mathrm{c1}}(R))$$

- Cascade of projection: ignore all but the last one

$$\pi_A(\pi_{A,B}(R)) = \pi_A(R)$$

- Commuting selection and projection: if the selection condition c involves only attributes in the projection list commute the two

$$\pi_{\mathrm{A,\ B}}(\sigma_{\mathrm{c}}(R)) = \sigma_{\mathrm{c}}(\pi_{\mathrm{A,\ B}}(R))$$

RA Transformation Rules (2)

- Commutativity of joins, cartesian product, union, intersection

$$R \; \theta \; S = S \; \theta \; R$$

- Associativity of join, cartesian product, union, intersection

$$(R \; \theta \; S) \; \theta \; T = R \; \theta \; (S \; \theta \; T)$$

- Selection and join: if attributes in the selection condition involves only attributes of one of the relations being joined
  - $\sigma_c(R \bowtie S) = \sigma_c(R) \bowtie S$

    $\sigma_c(R \bowtie S) = \sigma_{c1}(R) \bowtie \sigma_{c2}(S)$

RA Transformation Rules (3)

- Commuting projection with join: if join condition involves only attributes in the projection list, commute the operations

$$\pi_L(R \bowtie_c S) = (\pi_{L1}(R)) \bowtie_c (\pi_{L2}(S))$$

- Commuting selection with intersection, union, or difference

$$\sigma_c(R \; \theta \; S) = (\sigma_c(R)) \; \theta \; (\sigma_c(S))$$

- Several others in the book…

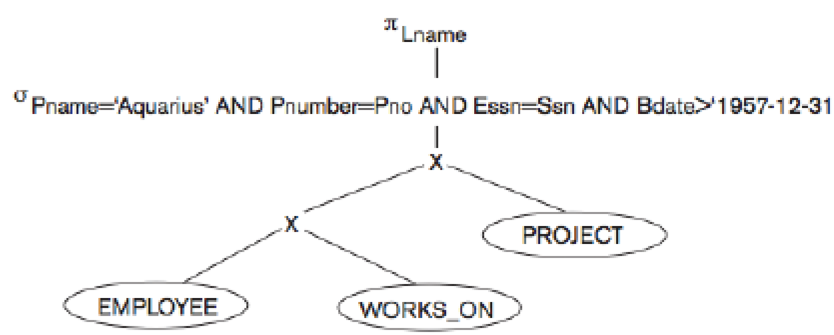
Query Optimization Heuristic Algorithm

- Break up any select operations with conjunctive conditions into cascade of select operations and move select operations as far down query tree as permitted
- Rearrange leaf nodes so leaf nodes with most restrictive select operations are executed first
- Combine cartesian product operation with a subsequent selection operation into join operation
- Break down and move lists of projection attributes down the tree as far as possible
- Identify subtrees that represent group of operations that can be executed as a single algorithm

Example: SQL Query Optimization

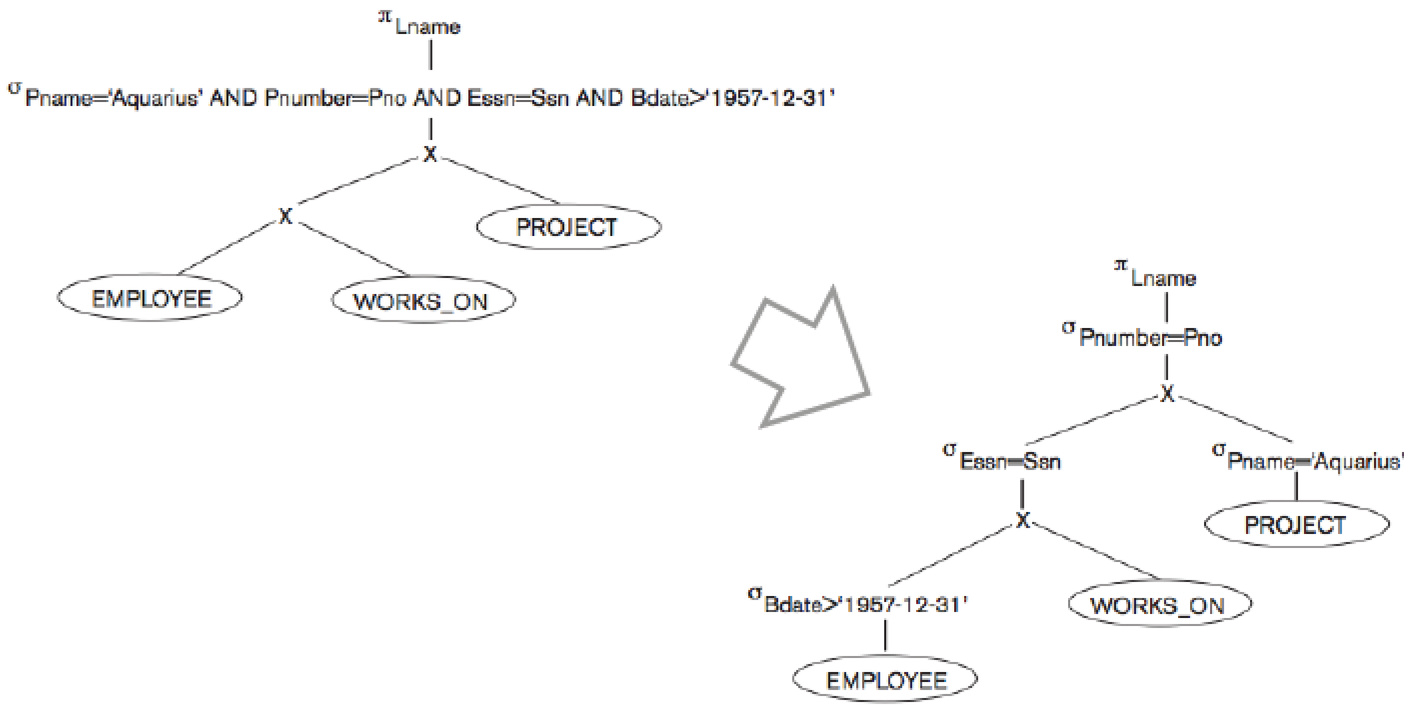
SELECT lname
FROM employee, works_on, project
WHERE pname = ‘Aquarius’ and pnumber = pno
AND bdate > ‘1957-12-31’;

Initial query tree



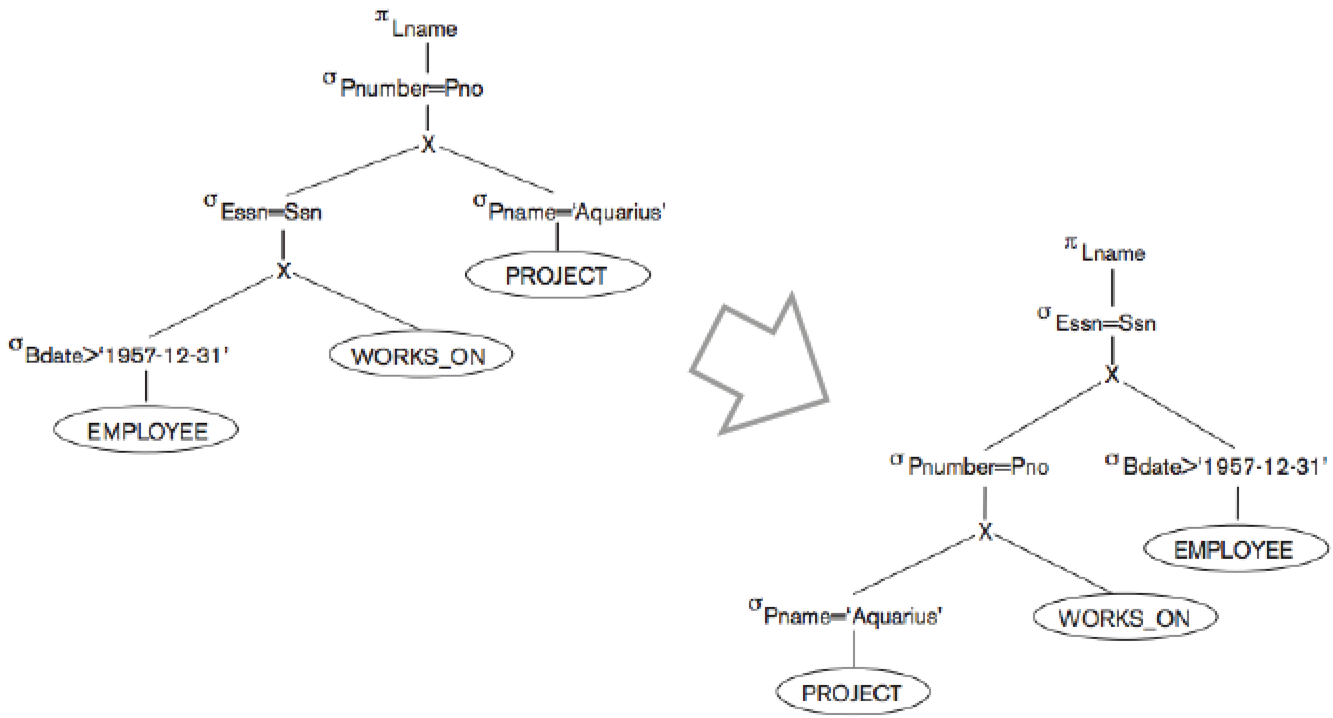
Example: SQL Query Optimization (2)

Move SELECT operations down the query tree



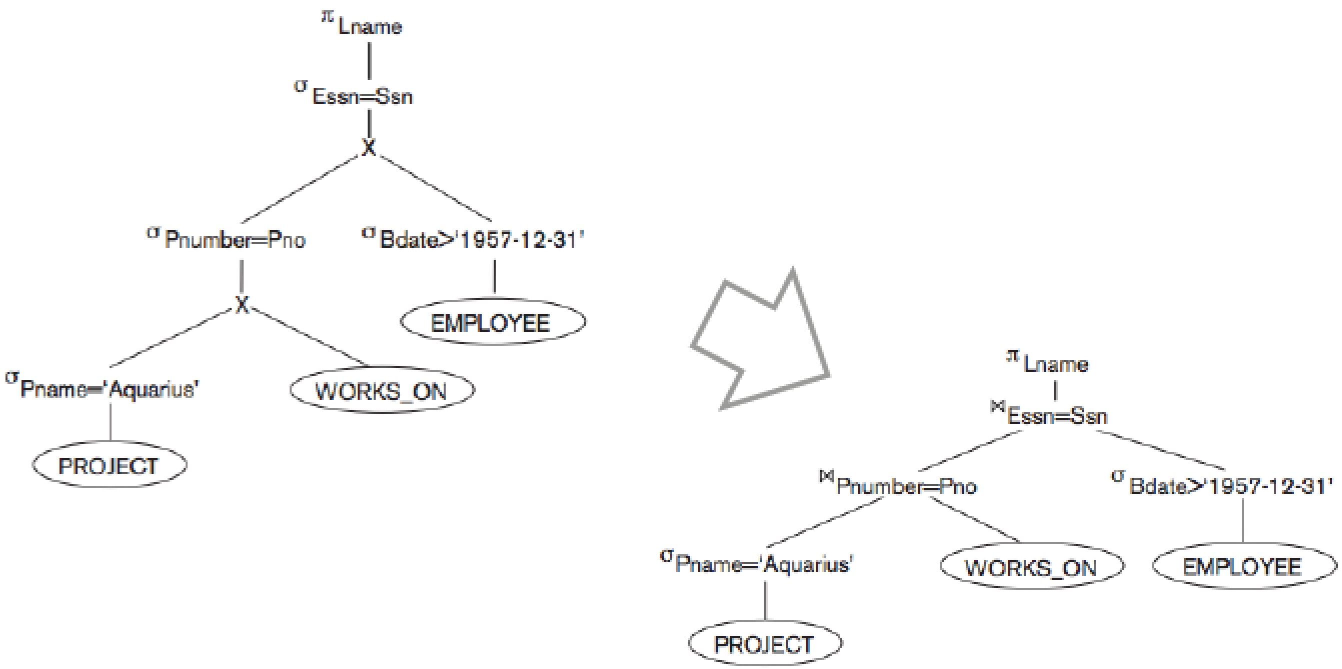
Example: SQL Query Optimization (3)

Apply more restrictive SELECT first (left most side of tree)



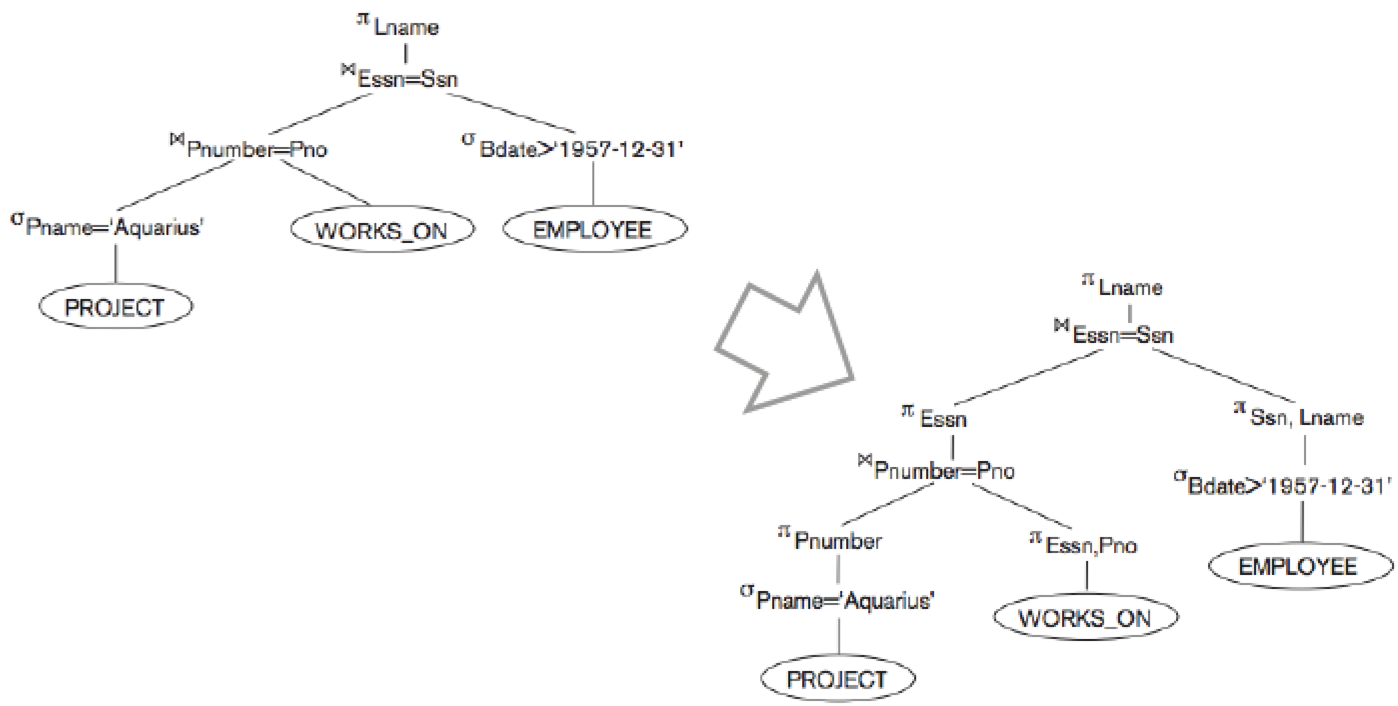
Example: SQL Query Optimization (4)

Replace cartesian product and select with join



Example: SQL Query Optimization (5)

Move projections down the tree



Query Optimization

- Logical level: heuristics based optimization to find a better RA query tree
    - SQL query —> initial logical query tree —> optimized query tree
- Physical level: cost-based optimization to determine “best” query plan
    - Optimized query tree —> query execution plans —> cost estimation —> “best” query plan

Cost-based Query Optimization

Estimate and compare the costs of executing a query using different execution strategies and choose the strategy with the lowest cost estimate

- Disk I/O cost
- Storage cost
- Computation cost
- Memory usage cost
- Communication cost (distributed databases)

Catalog Information

Database maintains statistics about each relation

- Size of file: number of tuples [n_r], number of blocks [b_r], tuple size [s_r], number of tuples or records per block [f_r], etc.
- Information about indexes and indexing attributes
  - Attribute values - number of distinct values [V(att, r)]
  - Selection cardinality - expected size of selection given value [SC(att, r)]
  - …

Catalog Information for Index

- Average fan-out of internal nodes of index i for tree-structured indices [f_i]

- Number of levels in index i (i.e., height of index i) [HT_i]

  - Balanced tree on attribute A of relation r: $\lceil \log_{f_i} V(A, r) \rceil$

  - Hash index: 1

- Number of lowest-level index blocks in i (i.e., number of blocks at the leaf level of the index) [LB_i]

Example: Bank Schema

Account relation

- $f_{account}$ = 20 (20 tuples per block)
- V(bname, account) = 50 (50 branches)
- V(balance, account) = 500 (500 different balance values)
- $n_{account}$ = 10000 (10,000 tuples in account)
- $b_{account}$ = 10000 / 20 = 500

SELECT Algorithms (Simple)

- Linear search (brute force): selection attribute is not ordered and no index on attribute
  - Cost: # blocks in relation = b_r
  - Reserves example: 500 I/Os
- Binary search: selection attribute is ordered and no index
  - Cost: $\underbrace{\lceil \log_2(b_r) \rceil}_{\text{locating first tuple}} + \underbrace{\lceil \mathrm{SC(att,\ r)}/f_r \rceil}_{\text{\# blocks with selection}} -1$

Example: Binary search

- How expensive is the following query if we assume Account is sorted by branch name?

$$\sigma_{\text{bname='Perryridge'}}(\text{Account})$$

- Ans:
    - # of tuples in the relation pertaining to Perryridge is total number of tuples divided by distinct values: 10000/50
    - Cost: $\lceil \log_2(500) \rceil + \lceil 200/20 \rceil - 1 = 18$

SELECT Algorithms (Simple w/ Index)

- Index search: cost depends on the number of qualifying tuples, cost of retrieving the tuples and the type of query
    - Primary index
        - Equality search on candidate key: $HT_i + 1$
        - Equality search on nonkey: $HT_i + \lceil \mathrm{SC(att,\ r)}/f_r \rceil$
        - Comparison search: $HT_i + \lceil c/f_r \rceil$

estimated number of tuples that satisfy condition

SELECT Algorithms (Simple w/ Index)

- Secondary index
  - Equality search on candidate key: $HT_i + 1$
  - Equality search on nonkey: $HT_i + SC(att, r)$
  - Comparison search: $HT_i + LB_i * c / n_r + c$

Note that linear file scan maybe cheaper if the number of tuples satisfying the condition is large!

Example: Index search

- How expensive is the following query if we assume primary index on branch name?

$$\sigma_{\text{bname='Perryridge'}}(\text{Account})$$

- Ans:
  - 200 tuples relating to Perryridge branch => clustered index
  - Assume B^+-tree index stores 20 pointers per node, then index must have between 3 and 5 leaf nodes with a depth of 2
  - Cost: $2 + \lceil 200/20 \rceil = 12$

SELECT Algorithms (Complex)

- Conjunctive selection (several conditions with AND)
  - Single index: retrieve records satisfying some attribute condition (with index) and check remaining conditions
  - Composite index
  - Intersection of multiple indexes
- Disjunctive selection (several conditions with OR)
  - Index/binary search if all conditions have access path and take union
  - Linear search otherwise

Example: Complex search

- How expensive if we want to find accounts where the branch name is Perryridge with a balance of 1200 if we assume there is a primary index on branch name and secondary on balance?

- Ans for using one index:

  - Cost for branch name: 12 block reads

  - Balance index is not clustered, so expected selection is 10,000 / 500 = 20 accounts

  - Cost for balance: 2 + 20 = 22 block reads

  - Thus use branch name index, even if it is less selective!

Example: Complex search (2)

- Ans for using intersection of two indexes:
    - Use index on balance to retrieve set of S1 pointers: 2 reads
    - Use index on branch name to retrieve set of S2 pointers: 2 reads
    - Take intersection of the two
    - Estimate 1 tuple in 50 * 500 meets both conditions, so we estimate the intersection of two has one pointer
    - Estimated cost: 5 block reads

Query Processing & Optimization: Recap

- Motivation for query optimization
- Query parse tree
- Query optimization heuristics
  - RA transformation rules
- Cost-based query optimization
  - SELECT