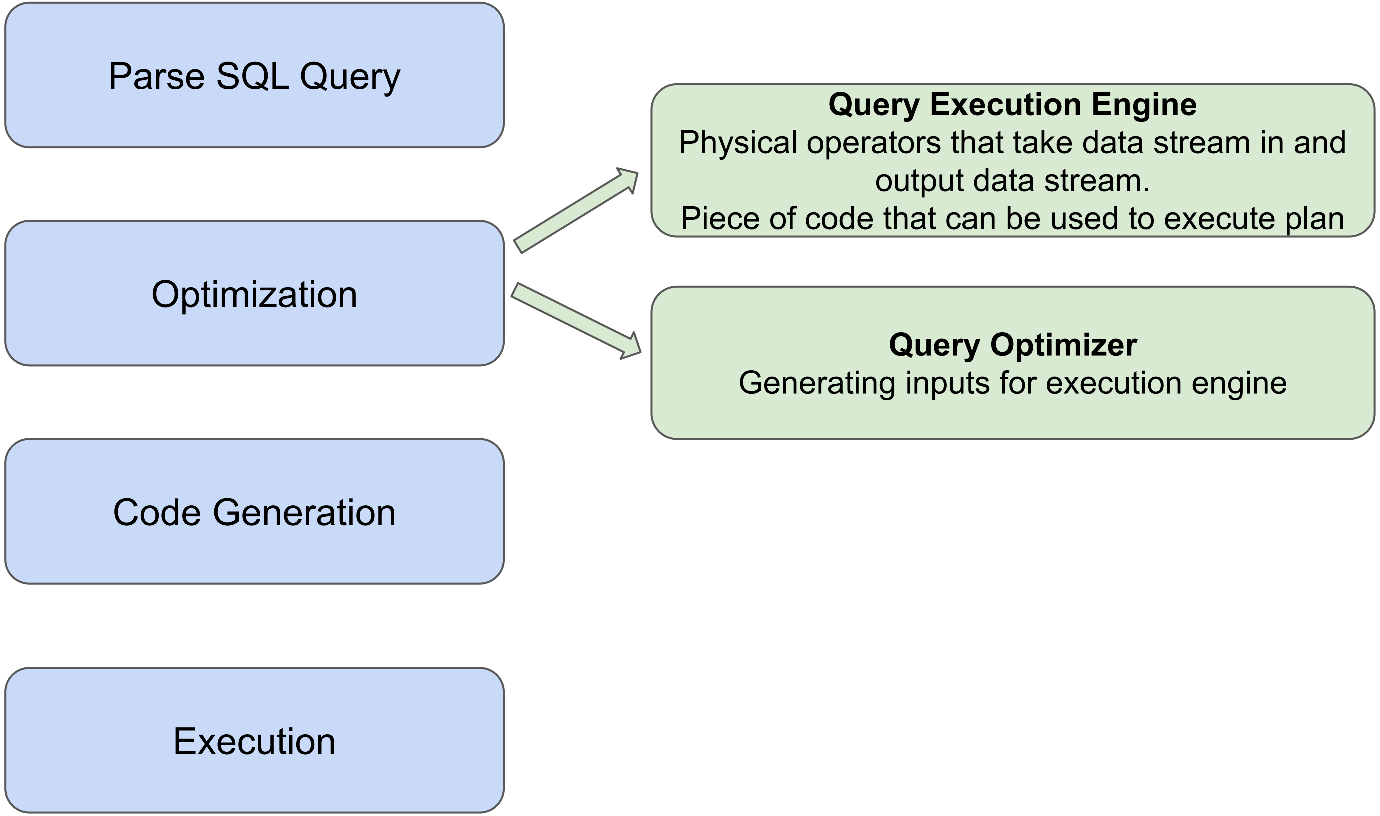
Query Optimization Overview

Presenter: Xuechun Cao
Discussion Lead: Rut Vora

*some slides/texts are borrowed from Rachel’s slides

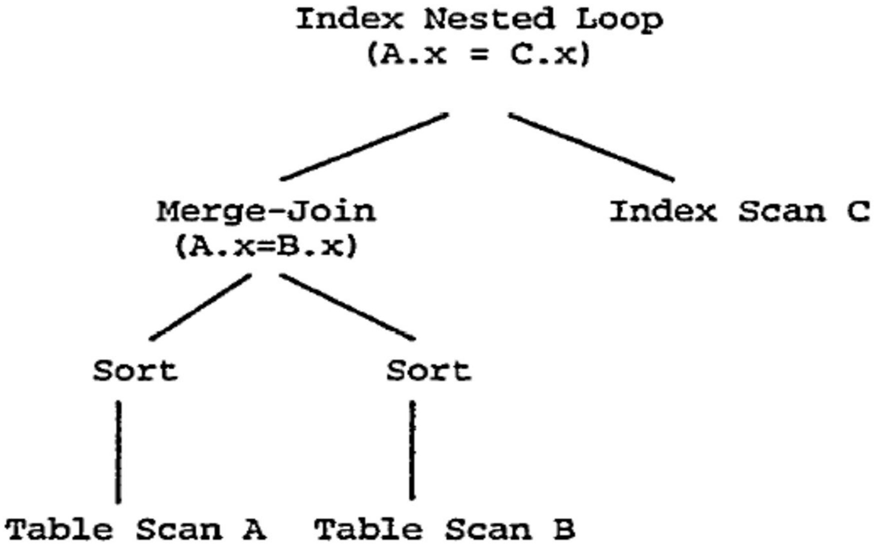
An Overview of Query Optimization in Relational Systems

Introduction



Introduction

Example Operator Tree



Discussion 1: Type of Paper and Target Audience

- Not a comprehensive SoK or Literature Review

Discussion in groups of ~4 people

Query Optimizer – key point: complex search problem

Input: parsed SQL representation
Output: an efficient execution plan

Search Space
algebraic transformations
and physical operators

Need to be **reduced** and
have the **lowest cost plan**

Cost Estimation
collect statistics to
estimate of resource
needed of candidate plans

Need to be **accurate**

Enumeration Algorithm
search in search space to
find lowest cost plan

Need to be **efficient**

Search Space

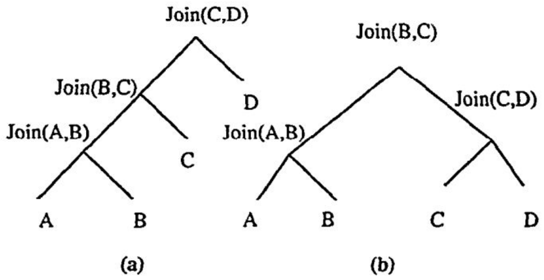
Depends on:
Equivalence performing algebraic transformations.
Physical operators supported in an optimizer

Transformations may not reduce cost and therefore must be applied in a cost-based manner to ensure a positive benefit

Search Space - Commutativity

Generalized Join Sequencing

- Most of time join relations follow commutativity rule so the order of joined relations can be arranged freely
- Most of system focuses on **linear** join not **bushy** join



- Some special case such as **outerjoin**–not freely commute

Join(R, S LOJ T) = Join (R,S) LOJ T

- Some special case such as **groupby**– can be pushed down the tree to provide more candidate plans

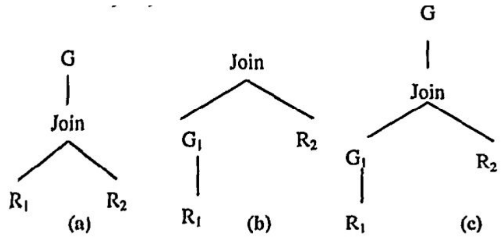


Figure 4. Group By and Join

Search Space - Merging Multi-block to single block

Merging Views

- extend search space by combine view relations into single block
- Q = join(R,V)
- view V = join(S, T)
- Q = join(R,join(S,T))

```
SELECT Emp.Name
FROM Emp
WHERE Emp.Dept#  IN
        SELECT Dept.Dept#  FROM Dept
        WHERE Dept.Loc='Denver'
        AND Emp.Emp# = Dept.Mgr
```

Merging Nest Block

- rewrite nested block

```
SELECT E.Name
FROM Emp E, Dept D
WHERE E.Dept# = D.Dept#
AND D.Loc = 'Denver' AND E.Emp# = D.Mgr
```

Discussion 2: Let the user optimise?

- Some languages allow user to optimise
- Useful for application-specific optimisations (e.g. applications that can tolerate less accuracy)

Discussion in groups of 2

Cost Estimation

Expect to be accurate and efficient

How it works

- Collect statistics summaries of data stored
- Give statistical summary of output data and estimate plan cost

Cost Estimation - Statistics

Example: Histogram

- get general data distribution information
- help estimate cardinality of predicates

Example: other statistics

- number of physical pages, order of stored indexes, distinct value of columns (more in second paper)

Open Research Questions

- difficult to estimate accurately based on base data statistics → can use sampling to estimate but for distinct values it can be error prone
- need to propagate statistics for different operators → assumptions made and inability of capturing correlation are important error source

Cost Estimation - Cost Computation

Computing Resource Consideration

- CPU, I/O → communication cost in parallel/distributed databases

other optimization interests

- data synchronization for distributed system and effective scheduling for parallel databases
- modeling buffer utilization

Discussion 3: Varying cost across decades

- More computing power
- Parallel computation
- Faster permanent storage (SSDs)
- Novel architectures like PIMs

Discussion in groups of 4
(2 systems + 2 non-systems)

Enumartion Algorithm

- Want enumerator to adapt to changes in search space
  - New transformations and physical operators
  - Changes in cost estimation techniques
- Solution
  - Use generalized cost functions and physical properties with operator nodes
  - Use rule engine that allows transformations to modify the query expression or the operator trees
  - Expose “knobs” to tune behavior of system

Other Optimizations

- Distributed and parallel systems
  - communication cost, data synchronization
- Materized views
  - cached relations
  - need to reformulate plan and determining effective sufficient conditions is nontrivial

Discussion 4: Can AI/ML be used for optimisation?

- Are heuristics sufficient?
- Large search space
  (covered in previous discussion)

Discussion in groups of 2/3
(1 AI/ML at least)

Summary

- Query optimization as a search problem whose solution requires:
  - a search space
  - cost estimation technique,
  - an enumeration algorithm
- Query optimization can be considered an art
- No one knows what the best execution plan for a given query is

Access Path Selection in a Relational database Management System

Introduction

Parse SQL Query

Optimization

Code Generation

Execution

Query Optimizer

- search space
- **cost estimation**
  - catalog lookup statistics
  - calculate estimcated cost for plans
- **enumeration algorithm**

Introduction - what is the problem?

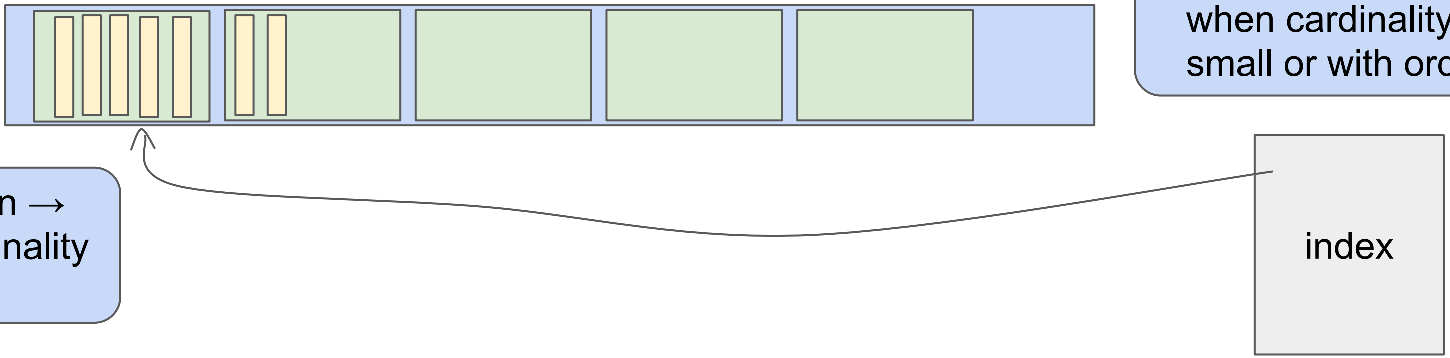
```
SELECT *
FROM A,B,C
WHERE A.n = B.n AND
B.m = C.m
```

- How should we execute this query?
  - must have a plan
  - but there are so many → need to trim search space
  - and need to be able to compare and check which one is the cheapest

Okay, for comparison, how do we estimate costs for plans?

Single Access Path - Catalog - How data stored and what we need to evaluate

Index scan → good when cardinality is small or with order



Sequential scan → good when cardinality is large

Statistics
- cardinality of tuples
- cardinality of pages
- etc

Physical Operator
- nested loop join
- merged join
- other sorting algorithms

Fetching Data
- sequential scan
- index scan
- buffer utilization

Single Access Path - Calculating Costs - How to get accurate cost

Predicates

- Helper reduce data size → evaluate as early as possible
- Use predicates with statistics can estimate more accurate selectivity factor (i.e. cardinality)

Interesting Order

- GROUP BY, ORDER BY may specify order of output data
- This helps determine what physical operator (e.g sort, sort-merge join)

Discussion 1: Limitations of Cost Calculation Methods

- Cost Formula too simple. Include other params?
- Cost calculation for distributed databases?

Discuss in groups of 4

Multiple Access Pass Selection - Bottom-up Dynamic Programming with memorization

Dynamic Programming

- Bottom-up → find local optimal in smaller subsets and use those value to build up larger sets

- N relations join is just as same as a sequence of 2-relation join
- Find the cheapest join of a subset of the N tables and store (memoization)
- Reduce complexity from n! to 2**n (number of subsets of n tables)

Bottom-up DP for Access Path Selection

Step

- Enumerate access path for single relation
  - sequential scan
  - stored index scan
  - consider interesting orders

- Enumerate access paths by joining an entra relation
  - nested loop (unordered)
  - merged join (interesting order)
  - prune by leaving cheapest for groups of equivalence solutions

Example

```
SELECT  NAME,TITLE,SAL,DNAME
FROM    EMP,DEPT,JOB
WHERE   TITLE='CLERK'
AND     LOC='DENVER'
AND     EMP.DNO=DEPT.DNO
AND     EMP.JOB=JOB.JOB
```

"Retrieve the name, salary, job title, and department name of employees who are clerks and work for departments in Denver."

Figure 1. JOIN example

Example

Access Paths for Single Relations

- Eligible Predicates: Local Predicates Only
- "Interesting" Orderings: DNO, JOB

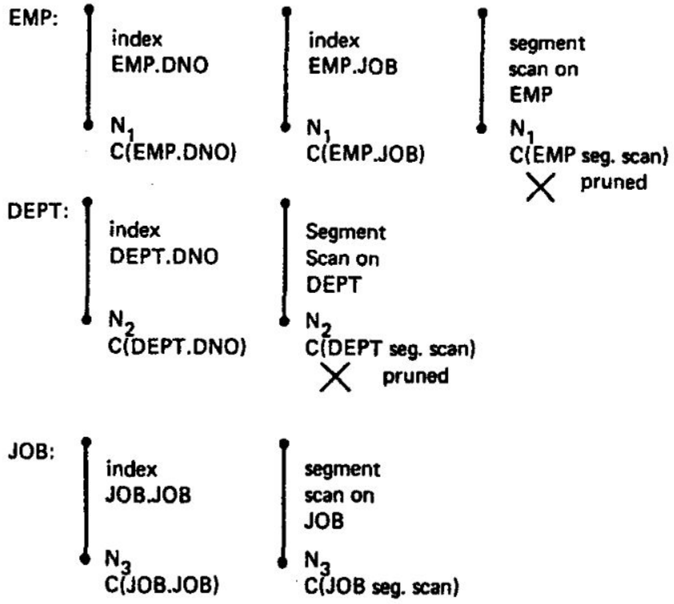


Figure 2.

Example

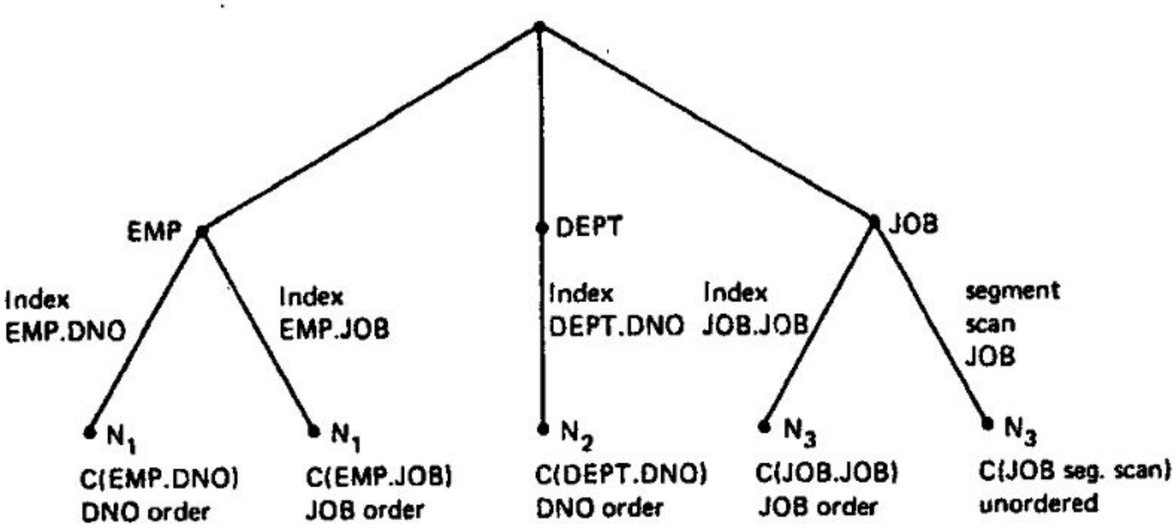


Figure 3. Search tree for single relations

Example

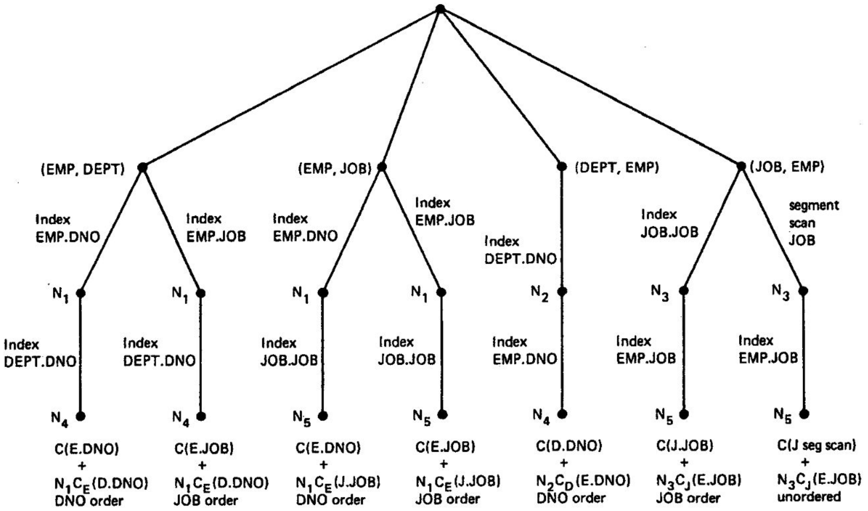


Figure 4. Extended search tree for second relation (nested loop join)

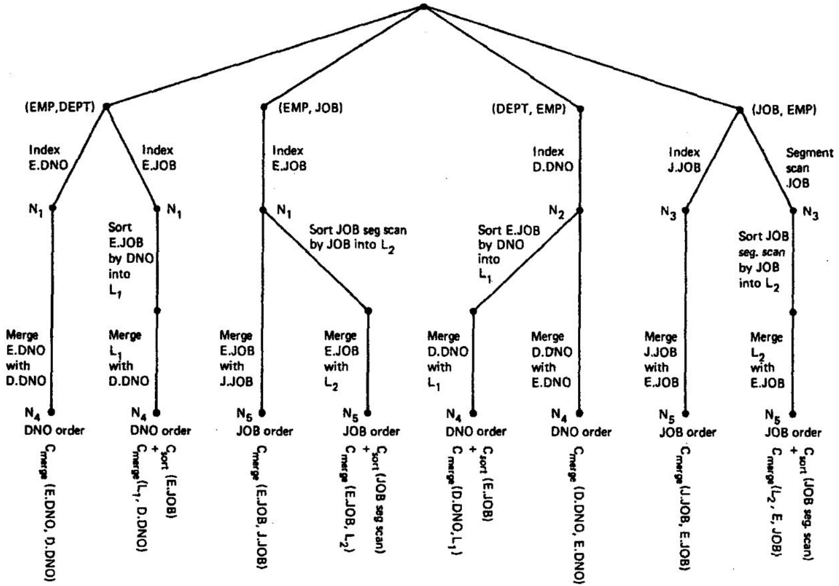


Figure 5. Extended search tree for second relation (merged join)

Example

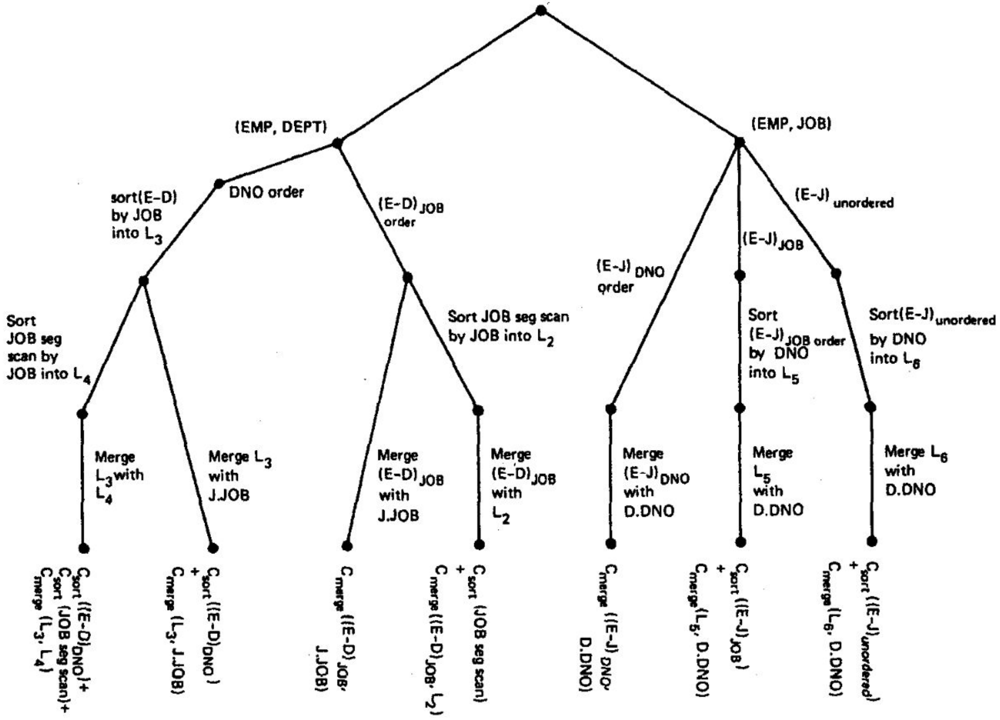


Figure 6. Extended search tree for third relation

Discussion 2: Feasibility of similar project in academia

- Large Dataset + Users
  - Might not be available in academia

Discuss in groups of 3

Discussion 3: Lack of full evaluation in System R papers

(Skip if out of time)

- No comparison with existing work at all
- Not waiting for a full evaluation. Why?

Class Discussion

Summary

- Introduce how cost is estimated in optimization
  - how statistics stored in system
  - how to use statistics to calculate cost
- Factors need to be considered
  - selectivity
  - interesting order
  - predicates
- Bottom-up dynamic programming for more than 2 joins
  - find suboptimal and build up solution for global optimal