

CS525: Advanced Database Organization

Notes 6: Query Processing Part I : Overview

Gerald Balekaki

Department of Computer Science

Illinois Institute of Technology

gbalekaki@iit.edu

July 17th 2023

Slides: adapted from courses taught by [Hector Garcia-Molina, Stanford](#), & [Shun Yan Cheung, Emory University](#)

Where we are

- How a DBMS processes queries and the methods it uses to optimize their performance.
- Goals: *Understand the basic concepts underlying the steps in query processing and optimization and estimating query processing cost; apply query optimization techniques*

Query Processing

- So far, we have looked
 - hardware features such as disks and memory
 - data structures allowing fast lookup
- Remember that SQL is a declarative language
 - User tells the DBMS what answer they want, not how to get the answer.
 - DBMS needs to translate a SQL statement into an executable query plan
 - This allows for optimization decisions, and for many queries there is a wide range of possible execution strategies, which can differ greatly in their resulting performance
 - i.e., there can be a big difference in performance based on plan is used
- A **query processor** must find a plan how to execute the query
 - query compilation
 - query execution

Query Processing

- There might be several ways to implement a query - the query compiler should find an appropriate plan
 - **parsing**: translating the query into a parsing tree
 - **query rewrite**: the parse tree is transformed into an expression tree of relational algebra (**logical query plan**¹)
 - **physical plan generation**: translate the **logical plan** into a **physical plan**²
 - select algorithms to implement each operator
 - choose order of operations
- *Logical Query Plan* the optimal sequence of relational algebra operations to perform the query.
- *Physical Query Plan* the optimal sequence of relational algebra algorithms to perform the query.

¹ The logical plan is roughly equivalent to the relational algebra expressions in the query.

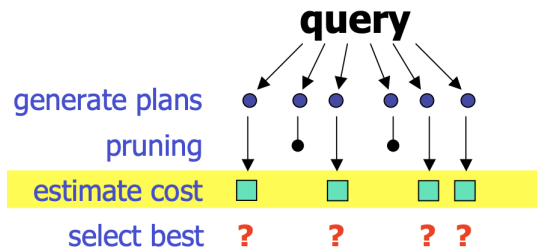
² Physical operators define a specific execution strategy using an access path for the different operators in the query plan.

Query Plan

- The DBMS converts a SQL statement into a query plan.
- Operators in the query plan are arranged in a tree.
- Data flows from the leaves of this tree towards the root.
- The output of the root node in the tree is the result of the query.
- Typically operators are binary (1-2 children).
- The same query plan can be executed in multiple ways.
- Most DBMSs will want to use an index scan as much as possible

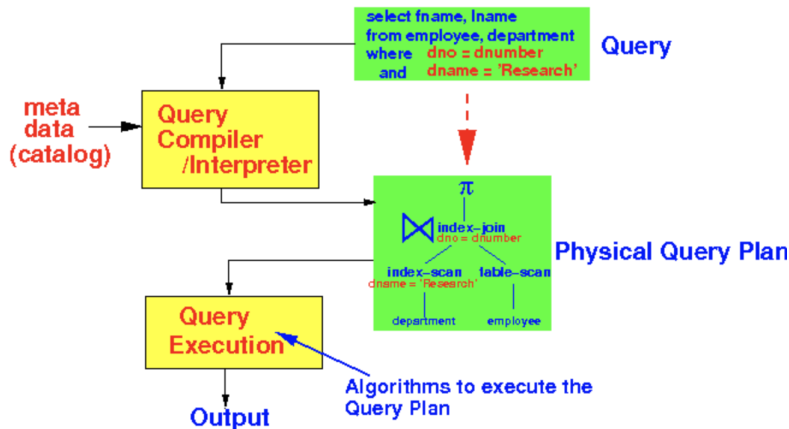
Query Processing

- Making logical and physical query plans are often called **query optimizing**



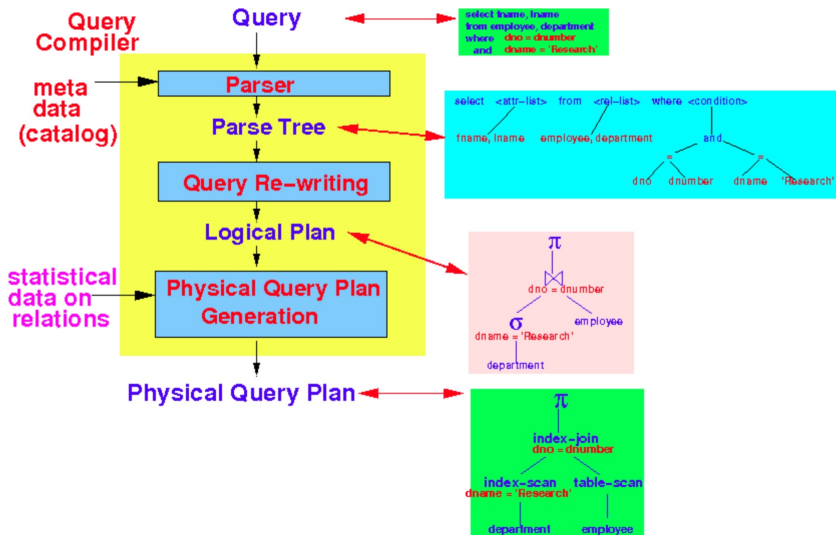
Query Processing: Steps to Process SQL Query

- Steps needed to process a query (SQL command)



Query Processing

- The **Query Compiler** consists of 3 major steps



Query Processing: Example

```
SELECT B,D  
FROM R,S  
WHERE R.A='c' and S.E=2 and R.c=S.c
```

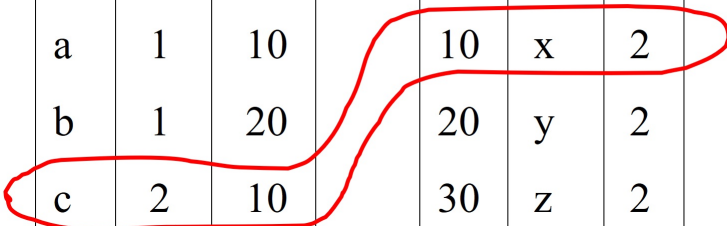
Query Processing: Example

| R | A | B | C |
|---|---|---|----|
| | a | 1 | 10 |
| | b | 1 | 20 |
| | c | 2 | 10 |
| | d | 2 | 35 |
| | e | 3 | 45 |

| S | C | D | E |
|---|----|---|---|
| | 10 | x | 2 |
| | 20 | y | 2 |
| | 30 | z | 2 |
| | 40 | x | 1 |
| | 50 | y | 3 |

Query Processing: Example

| R | A | B | C | S | C | D | E |
|---|---|---|----|---|----|---|---|
| | a | 1 | 10 | | 10 | x | 2 |
| | b | 1 | 20 | | 20 | y | 2 |
| | c | 2 | 10 | | 30 | z | 2 |
| | d | 2 | 35 | | 40 | x | 1 |
| | e | 3 | 45 | | 50 | y | 3 |



| | | |
|--------|---|---|
| Answer | B | D |
| | 2 | x |

How do we execute query?

- One idea
 - Scan relations
 - Do Cartesian product
 - Select tuples
 - Do projection

Query Processing: Example

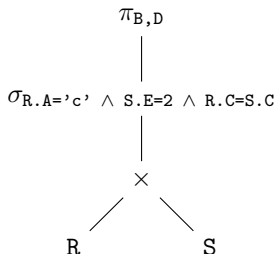
| $R \times S$ | R.A | R.B | R.C | S.C | S.D | S.E |
|--------------|-----|-----|-----|-----|-----|-----|
| a | 1 | 10 | 10 | x | 2 | |
| a | 1 | 10 | 20 | y | 2 | |
| . | | | | | | |
| . | | | | | | |
| C | 2 | 10 | 10 | x | 2 | |
| . | | | | | | |
| . | | | | | | |

Query Processing: Example

| RXS | R.A | R.B | R.C | S.C | S.D | S.E |
|------------------------|-----|-----|-----|-----|-----|-----|
| | a | 1 | 10 | 10 | x | 2 |
| | a | 1 | 10 | 20 | y | 2 |
| | . | | | | | |
| | . | | | | | |
| Bingo! → Got one... | C | 2 | 10 | 10 | x | 2 |
| | . | | | | | |
| | . | | | | | |

Relational Algebra

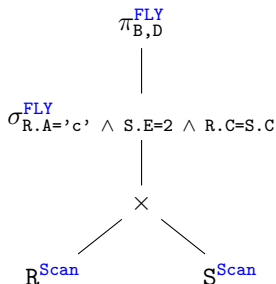
- Can be used to describe plans
- Example: Plan I: Initial query plan constructed directly from the query.



- **FROM** expressed by a product, **WHERE** by a selection above it, **Select** by a projection

Relational Algebra

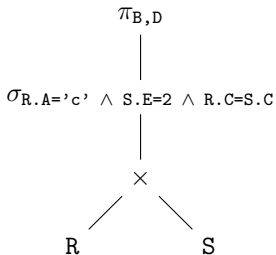
- Can be used to describe plans
- Example: Plan I: Initial query plan constructed directly from the query.



1. Scan R
 2. For each tuple \mathbf{r} of R scan S
 3. For each tuple \mathbf{r}, \mathbf{s} , where \mathbf{s} in S, select and project on the fly
- OR: $\pi_{B,D}^{\text{FLY}}[\sigma_{R.A='c' \wedge S.E=2 \wedge R.C=S.C}^{\text{FLY}}(R^{\text{Scan}} \times S^{\text{Scan}})]$

“FLY” and “SCAN” are the defaults

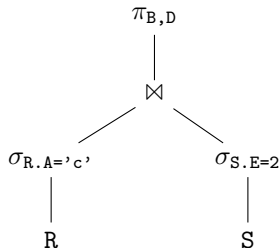
- Example: Plan I: Initial query plan constructed directly from the query.



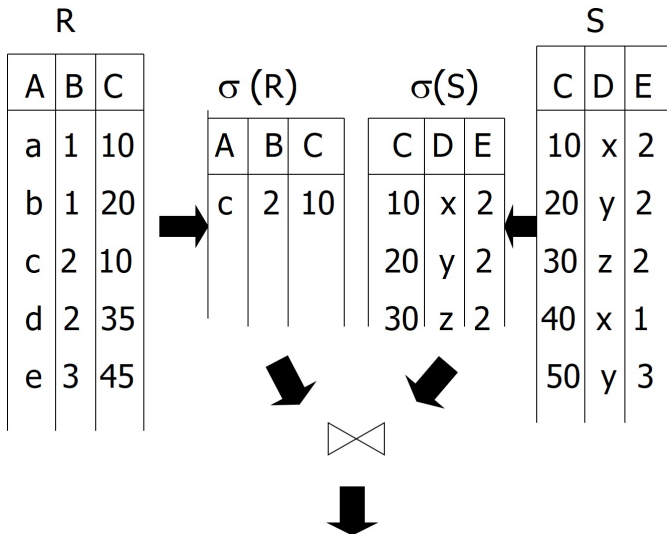
- OR: $\pi_{B,D}[\sigma_{R.A='c' \wedge S.E=2 \wedge R.C=S.C}(R \times S)]$

Another idea

- Example: Plan II: Scan **R** and **S**, perform on the fly selections, do natural join, project



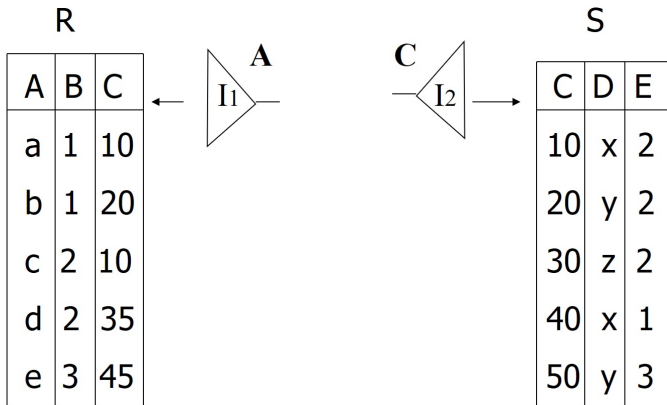
Another idea



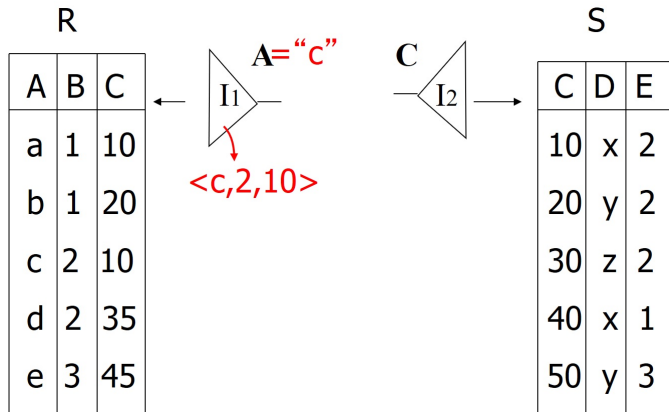
Plan III: Use R.A and S.C Indexes

1. Use R.A index to select R tuples with R.A = 'c'
2. For each R.C value found, use S.C index to find matching tuples
3. Eliminate S tuples S.E \neq 2
4. Join matching R,S tuples, project B,D attributes and place in result

Plan III: Use R.A and S.C Indexes

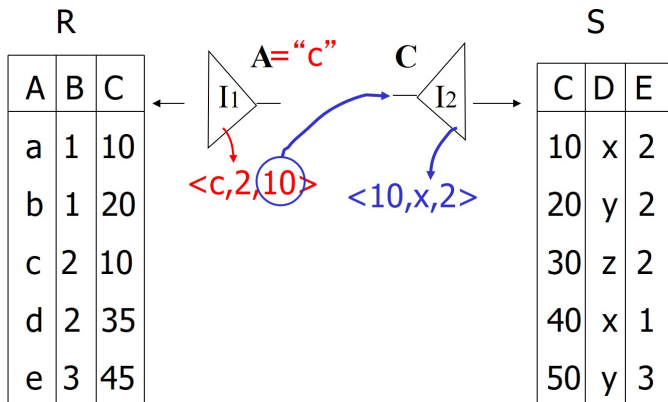


Plan III: Use R.A and S.C Indexes



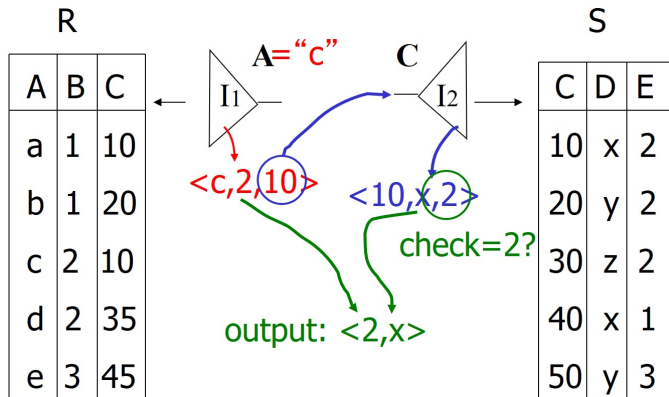
1. Use R.A index to select R tuples with R.A = 'c'

Plan III: Use R.A and S.C Indexes



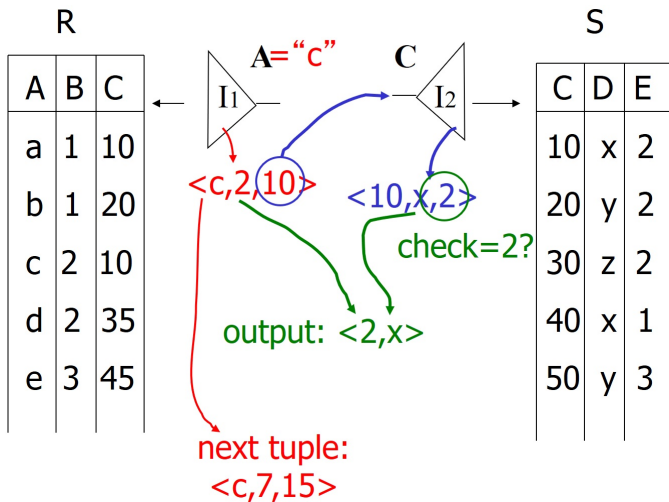
2. For each R.C value found, use S.C index to find matching tuples

Plan III: Use R.A and S.C Indexes

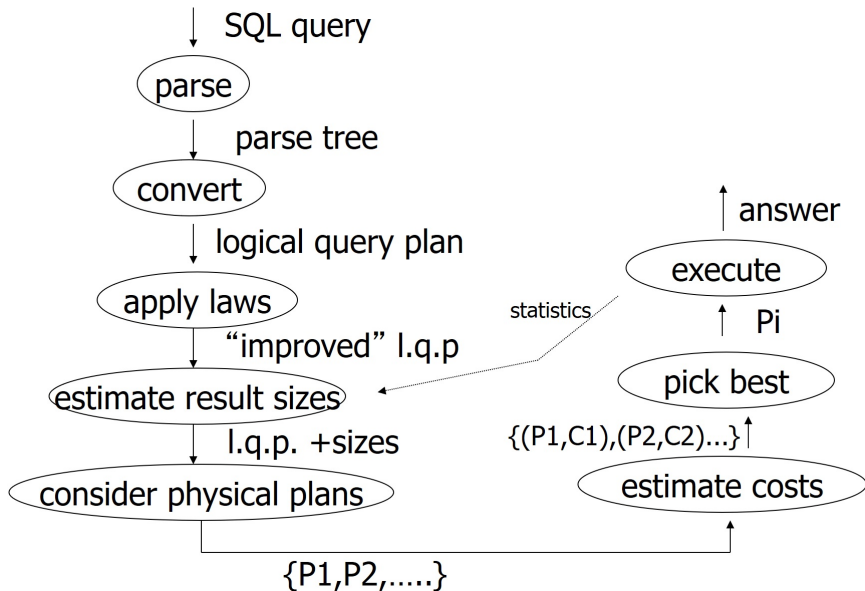


3. Eliminate S tuples S.E \neq 2
4. Join matching R,S tuples, project B,D attributes and place in result

Plan III: Use R.A and S.C Indexes



Overview of Query Optimization

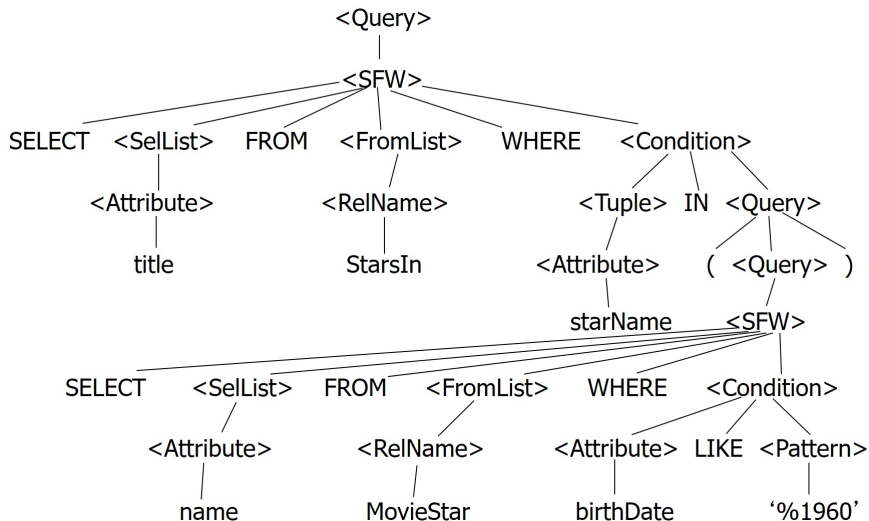


Example: SQL query

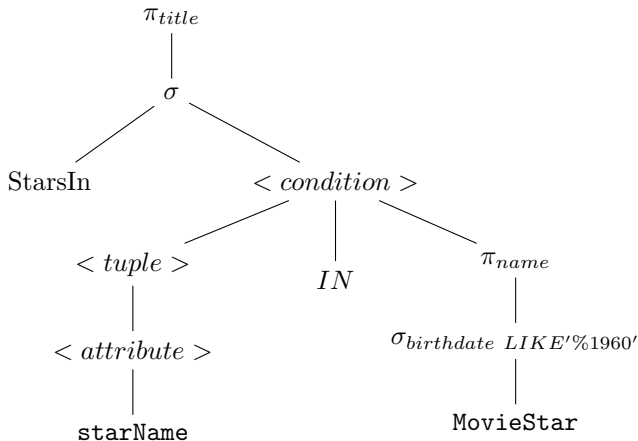
```
SELECT title
FROM StarsIn
WHERE starName IN (SELECT name
                    FROM MovieStar
                    WHERE birthdate LIKE '%1960');
```

- Find the movies with stars born in 1960

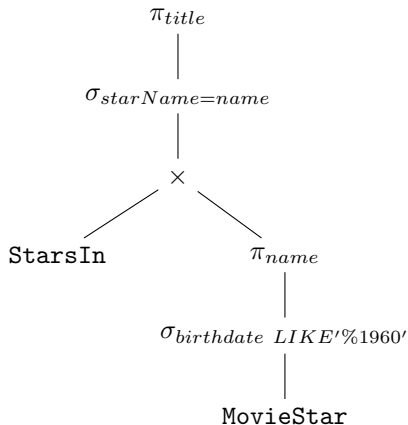
Example: Parse Tree



Example: Generating Relational Algebra

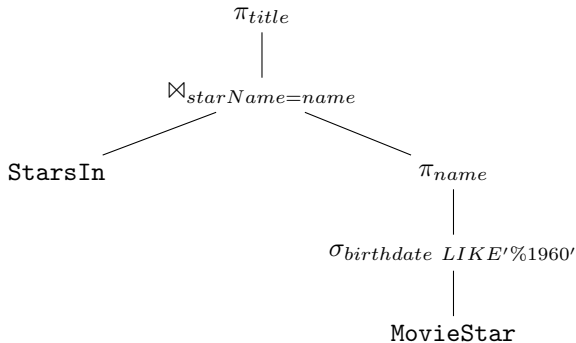


Example: Logical Query Plan



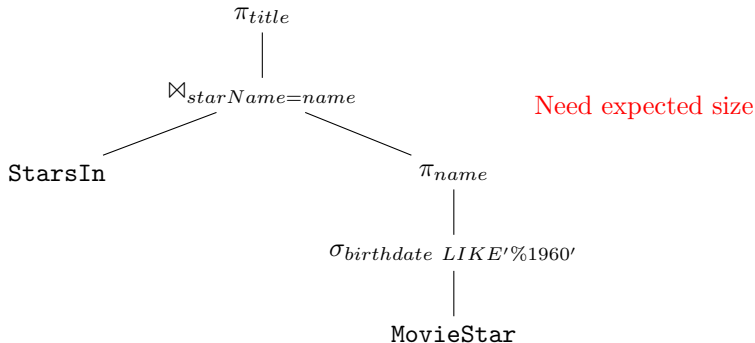
- May consider “IN” elimination as a rewriting in the logical plan generator or may consider it a task of the converter

Example: Improved Logical Query Plan

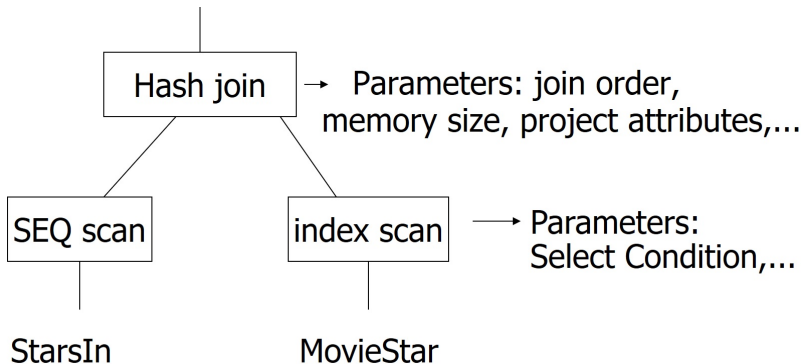


Result sizes are important for selecting physical plans

Estimate Result Sizes



Example: One Physical Plan



Example: Estimate costs

