



ACCESS PATH SELECTION IN A RELATIONAL DATABASE MANAGEMENT SYSTEM

Presenter: Jill Zhou



Overview

- Main topic: How system R chooses Access paths for queries?
 - *System R:*
 - Build by IBM
 - First implementation of SQL
 - *Queries:*
 - Simple (single relation)
 - Complex (like joins)
 - *Principle to choose Access paths: minimizes total cost*

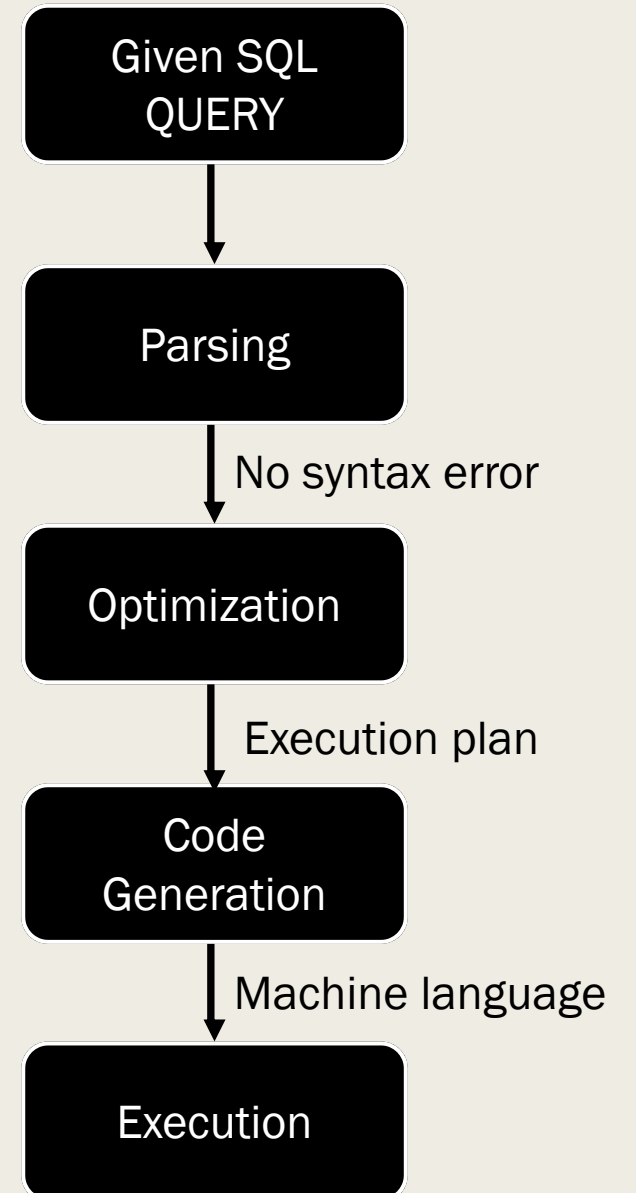
A thick black L-shaped frame surrounds the text. The top-left corner is a horizontal bar extending to the right, and the bottom-right corner is a vertical bar extending upwards. The text is centered within the open space of the frame.

PROCESSING OF AN SQL STATEMENT

Four Phases

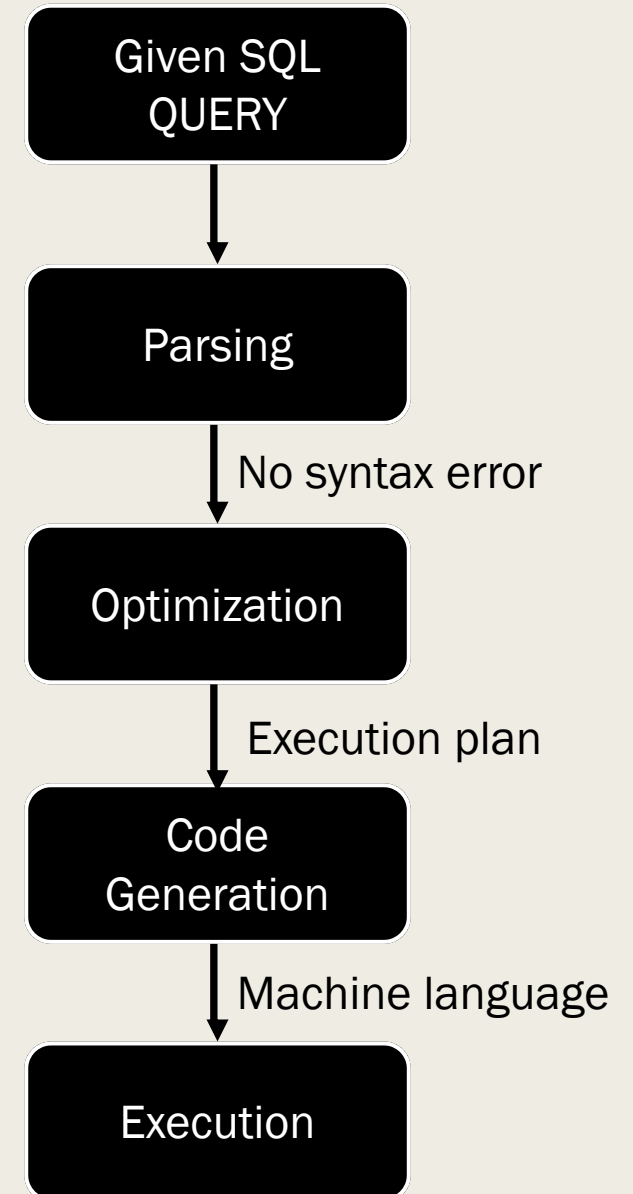
Processing of an SQL statement (four phases)

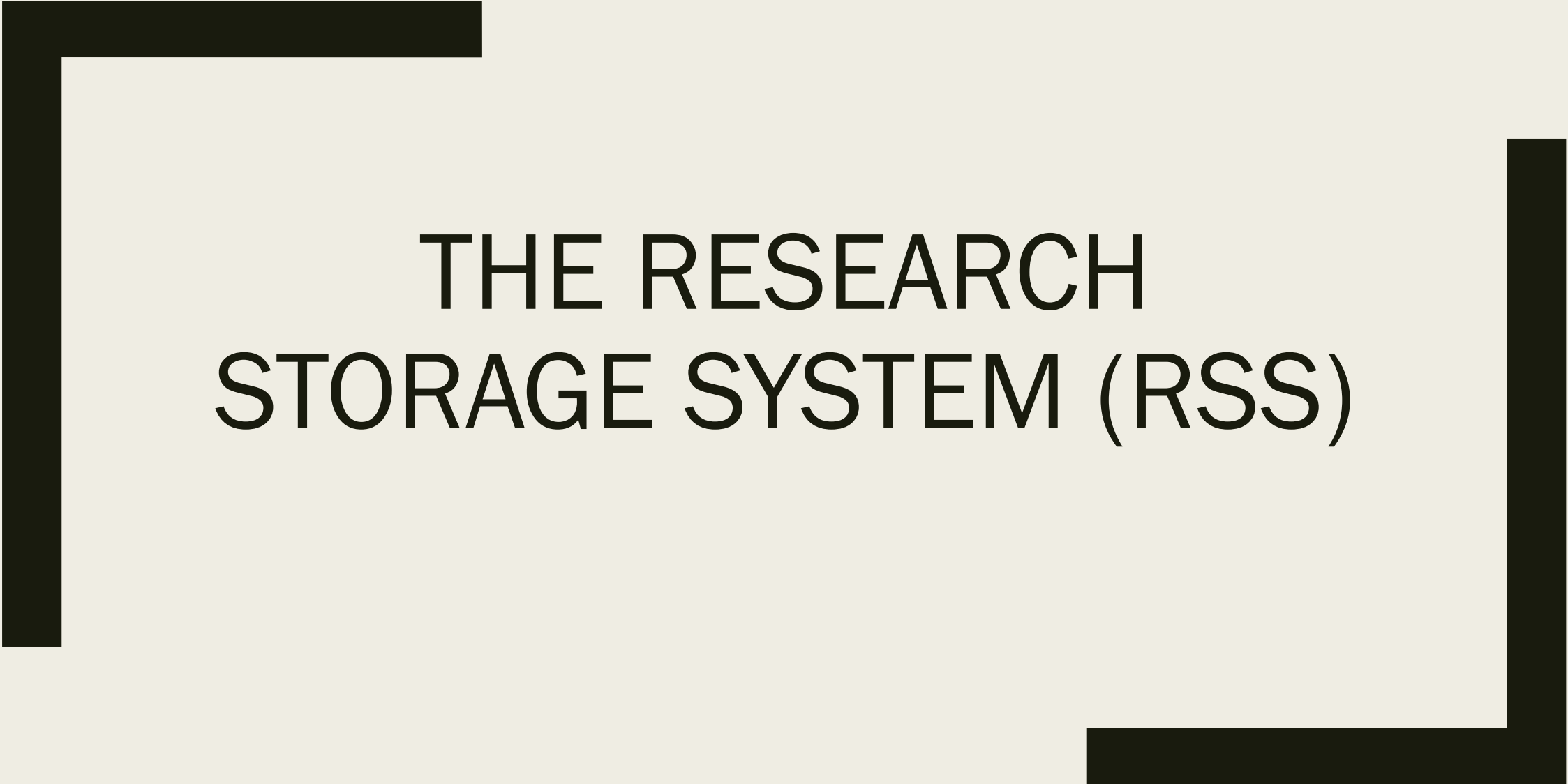
- Parsing:
 - *Accept a SQL query*
 - *check for correct syntax*
- Optimization:
 - ***Catalog lookup:***
 - Verify the existence of names of tables and columns in system R catalogs
 - retrieve information
 - Statistics of the referred relations
 - *E.g.: datatype and length of each column*
 - Check available access paths
 - ***Check for semantic error***
 - ***Access path selection (execution plan)***



Processing of an SQL statement (four phases)

- Code generation
 - *Translates execution plan into machine language*
 - *check for correct syntax*
- Execution
 - *Call the System R internal Storage System (RSS) to scan the storage location*



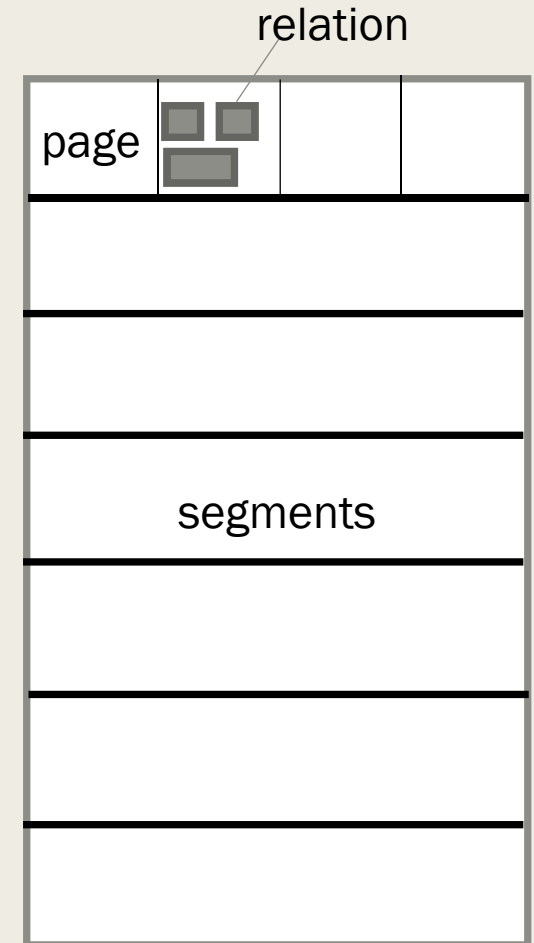


THE RESEARCH STORAGE SYSTEM (RSS)

The Research Storage System (RSS):

A brief introduction

- The RSS responsible for maintaining physical storage of:
 - *Relations*
 - *Access paths on these relations*
 - *Locking*
 - *Logging*
 - *Recovery facilities*
- RSS structure:
 - *Tuples composed relations*
 - *Relations stored in pages*
 - *Pages are organized into segments*



The Research Storage System (RSS): accessing tuples

- RSS scan: A scan returns a tuple at a time along a given access path.
- Two Types of scan:
 - *Segment Scan: find all tuples of a given relation*
 - All none empty pages in a segment will be touched for one time
 - *Index Scan:*
 - Index may be created by a system R on one or more columns relation
 - Stored in separate pages (not the pages containing the relations)
 - During the scan, Index page will be touched only once, but data pages may be touched more than once.

The image features two large, thick black L-shaped brackets. One is positioned in the top-left corner, and the other is in the bottom-right corner. They are oriented towards each other, framing the central text.

ACCESS PATHS

Cost for single relation access paths

- $COST = PAGE\ FETCHES + W * (RSI\ CALLS)$
 - *PAGE FETCHES: Weighted measure of I/O and CPU utilizations*
 - *W: an adjustable weighting factor between I/O and CPU*
 - *RSI CALLS: predicted number of tuple return from the RSS*
- Minimum cost path: a path need minimum resources (such as minimum number of tuples)

Access path selection for joins

- **Nested loops method**

- *E.g.: SQL: $r \text{ join } s$*

- r is called the outer relation

- s is called the inner relation

- For each tuple in the outer relation:

- *Scan for the tuple satisfied the join condition in the inner relation*

- **Merging scan method**

- *Require the outer and inner relations to be scanned in join column order*

- *Only applied to equi-joins*

- E.g. $\text{table1.column1} = \text{table2.column2}$

Access path selection for joins

- M-way joins:
 - *A sequence of 2-way joins*
 - Join the first two relations, and use the result to join the third, and so on.
 - *E.g. ((A JOIN B) JOIN C)*
 - *Nested loops joins and merge scan joins may be mixed*
 - *The cost of joining in different orders can be substantially different*
 - E.g. different orders may apply different join methods.
 - *A heuristic method to reduce the join order*
 - Only consider the join order which have join predicates relating the inner relation to the other relations already participating in the join
 - E.g. given $T1.col1 = T2.col2$, and $T2.col2 = T3.col3$
 - *There are no following permutations: T1-T3-T2 & T3-T1-T2*

Computation of costs for joins

- The costs for joins are computed from the costs of the scans on each of the relations and the cardinalities
- Nested loop join cost:
 - $C\text{-nested-loop-join}(\text{path1}, \text{path2}) = C\text{-outer}(\text{path1}) + N * C\text{-inner}(\text{path2})$
- Merge scan join cost:
 - $C\text{-merge}(\text{path1}, \text{path2}) = C\text{-outer}(\text{path1}) + N * C\text{-inner}(\text{path2})$
- Meaning of symbols:
 - $C\text{-outer}(\text{path1})$: cost of scanning the outer relation via path 1
 - $C\text{-inner}(\text{path2})$: cost of scanning the inner relations via path 2, applying all applicable predicates
 - N : be the cardinality of the outer relation tuples, which satisfy the applicable predicates.



A EXAMPLE

EMP	NAME	DNO	JOB	SAL
	SMITH	50	12	8500
	JONES	50	5	15000
	DOE	51	5	9500

DEPT	DNO	DNAME	LOC
	50	MFG	DENVER
	51	BILLING	BOULDER
	52	SHIPPING	DENVER

JOB	TITLE
5	CLERK
6	TYPIST
9	SALES
12	MECHANIC

```

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

A walk through of an example of searching cheapest path

- On EMP table: assume the index on job is the cheapest path.
 - *Segment scan is pruned*
- On DEPT table: We assume the index on DNO is cheaper
 - *Segment scan is pruned*
- On JOB table: Segment scan is cheaper
 - *Kept both*

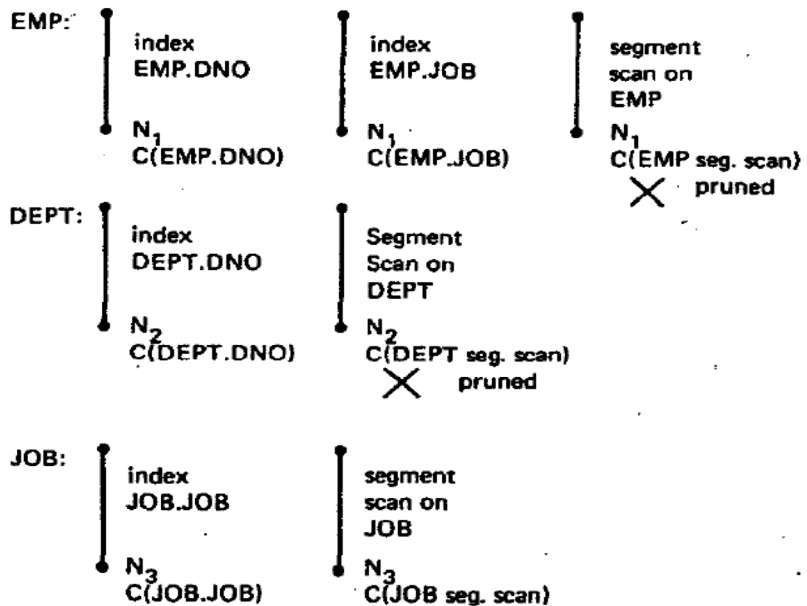


Figure 2.

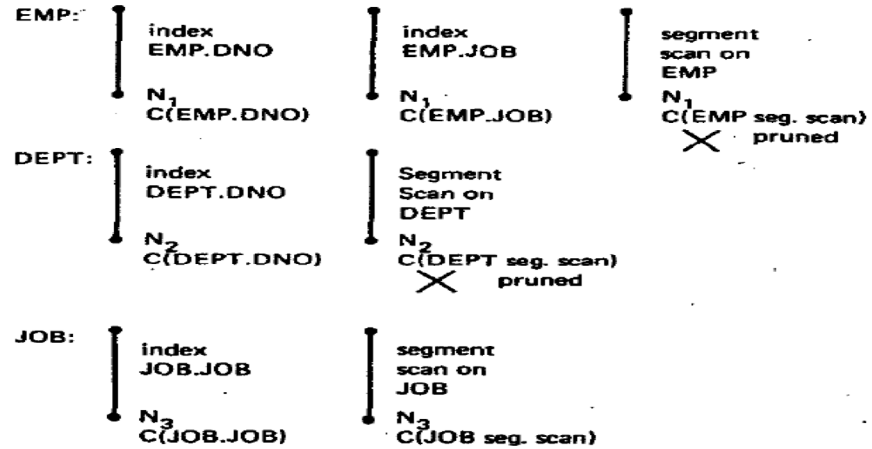


Figure 2.

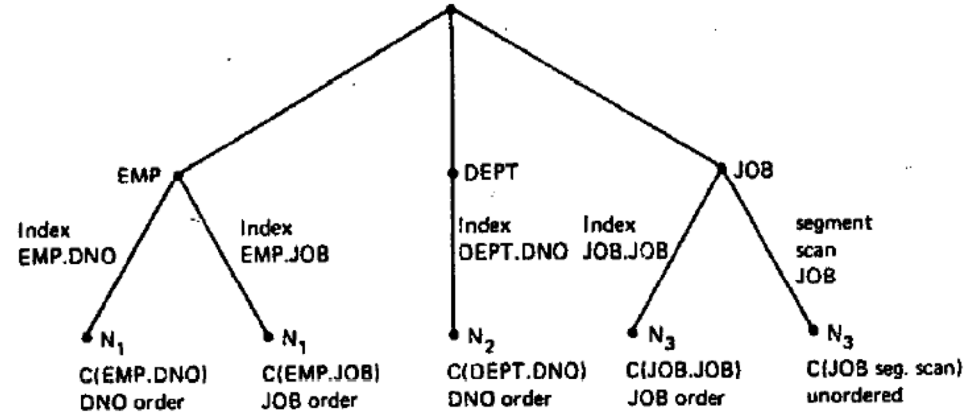


Figure 3. Search tree for single relations

CREATE SEARCH TREE FOR SINGLE RELATIONS

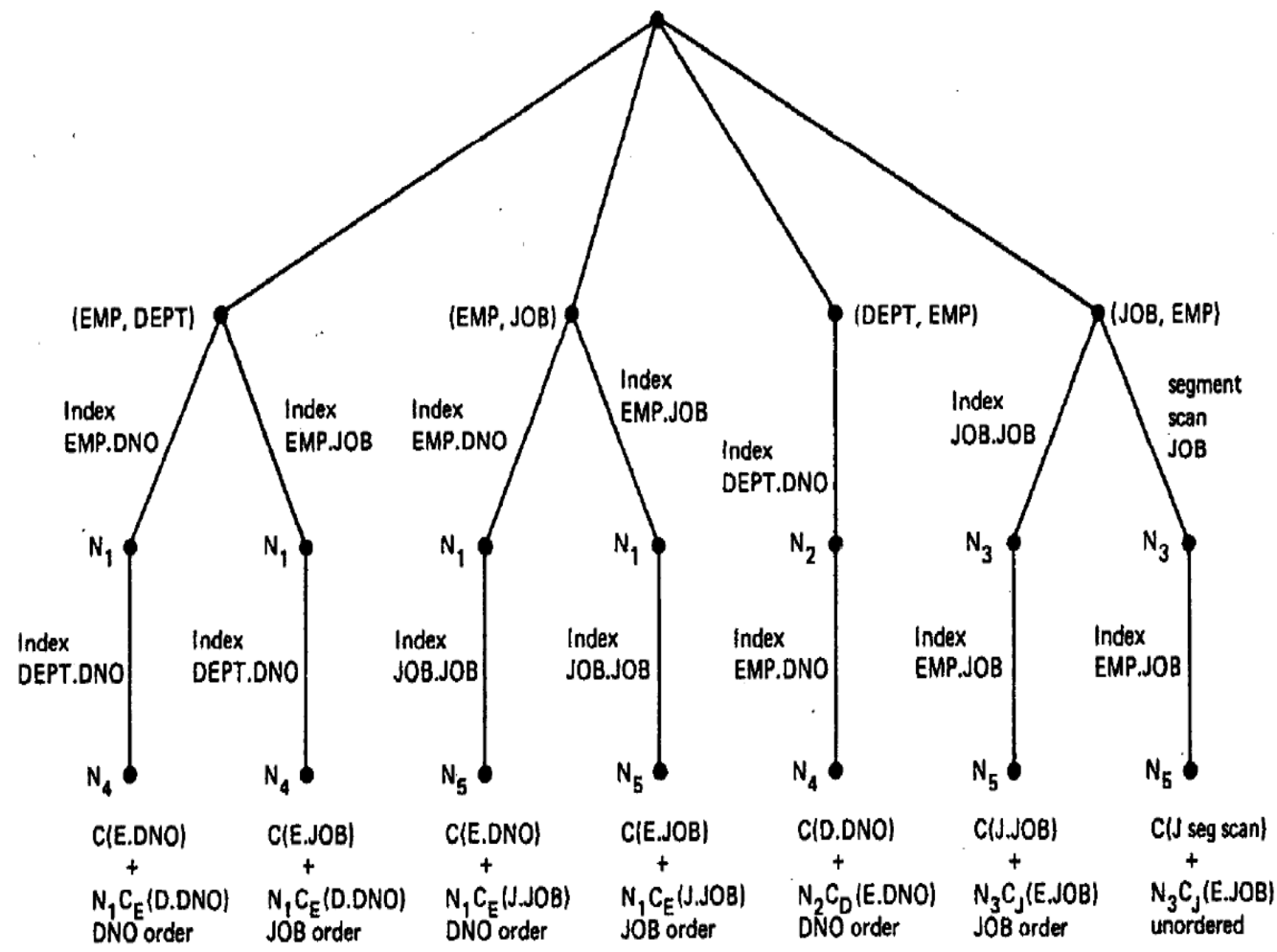


Figure 4. Extended search tree for second relation (nested loop join)

Joining second relation to the single relations

- For (EMP, JOB) join, we assume that is cheapest by accessing Job on the job index.
- For (EMP, DEPT) join, we assume that the DNO relation is cheaper

```

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
  
```

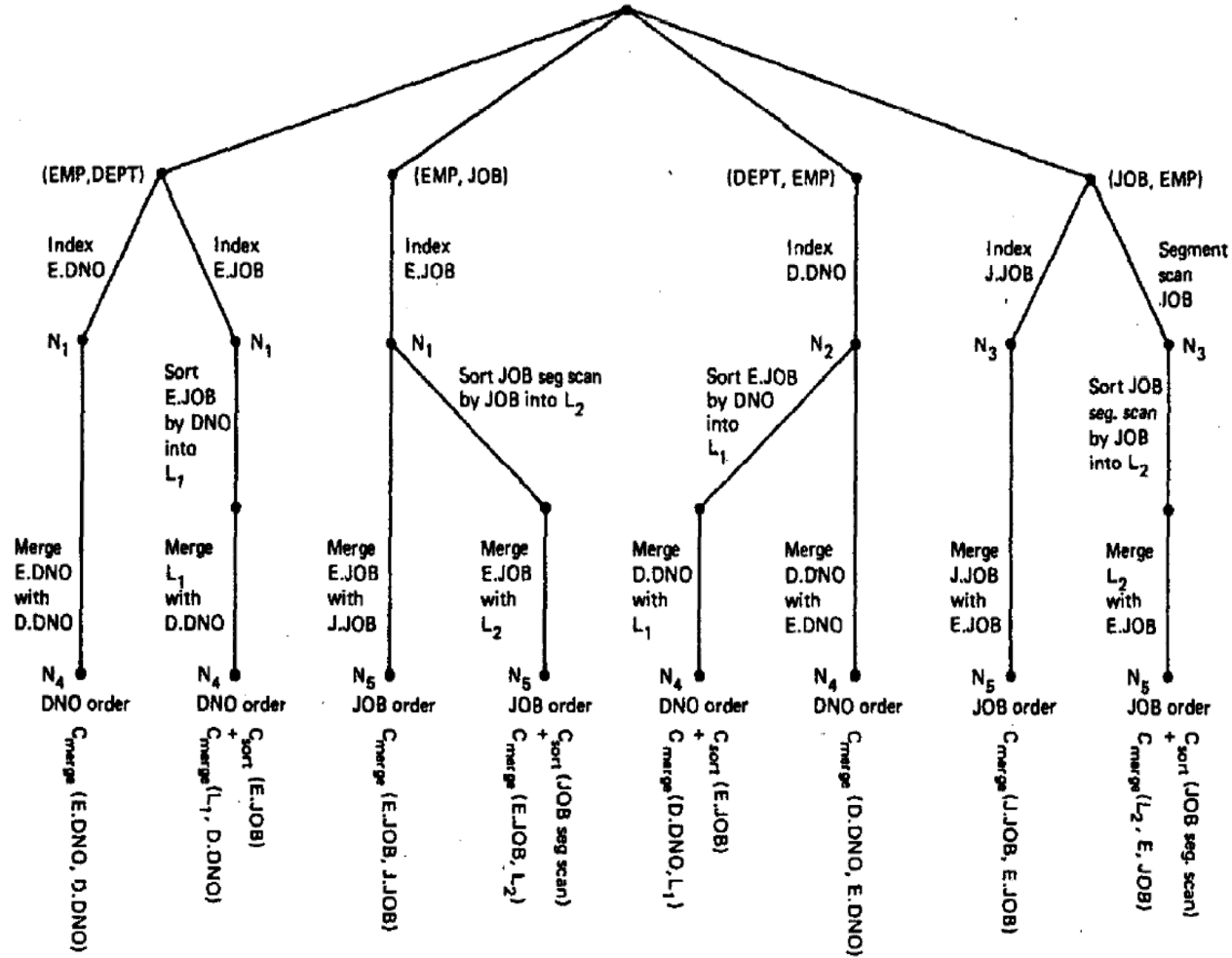



Figure 5. Extended search tree for second relation (merge join)

Joining second relation to the single relations

- In DEPT table, the cheapest scan is in DNO order.
- In EMP table, the job index is the cheapest access path.

The image features two large, thick black L-shaped brackets. One is positioned in the top-left corner, and the other is in the bottom-right corner. They are oriented towards each other, framing the central text.

NESTED QUERIES

```
SELECT NAME
FROM EMPLOYEE
WHERE SALARY =
      (SELECT AVG(SALARY)
       FROM EMPLOYEE)
```

If the operator is IN or NOT IN then the subquery may return a set of values. For example:

```
SELECT NAME
FROM EMPLOYEE
WHERE DEPARTMENT_NUMBER IN
      (SELECT DEPARTMENT_NUMBER
       FROM DEPARTMENT
       WHERE LOCATION='DENVER')
```

NESTED QUERY

- A query may appear as an operand of a predicate of the form “expression operator query”. Such a query is called a Nested Query or a Subquery.
- The optimizer will arrange the subquery to be evaluated at first.

Correlation Subquery

- Correlation Subquery: A subquery may contain a reference to a value obtained from a candidate tuple of a higher level query block
- A correlation subquery must in principle be re-evaluated for each candidate tuple from the referenced query block.

the query:

```
SELECT NAME  
FROM EMPLOYEE X  
WHERE SALARY > (SELECT SALARY  
                 FROM EMPLOYEE  
                 WHERE EMPLOYEE_NUMBER=  
                       X.MANAGER)
```

The image features two large, thick, black L-shaped brackets. One is positioned in the top-left corner, with its vertical bar extending downwards and its horizontal bar extending to the right. The other is in the bottom-right corner, with its horizontal bar extending to the left and its vertical bar extending upwards. These brackets frame the central text.

CONCLUSION

conclusion

- Describing the access path selection methods in system R
 - *Single paths*
 - *For joins*
 - Nested loop join method
 - Merging scan method
- How to calculate the cost of each type of paths?
 - $COST = PAGE\ FETCHES + W * (RSI\ CALLS)$
- Nested Queries
 - *Evaluate the lowest level to top level*



Q & A