Access Path Selection in a Relational Database Management System

CMPT 843

Pouya Fattahi

Outlines

- How System R processes an SQL query?

Outlines

- How System R processes an SQL query?
- How System R optimize an SQL query?

Four steps to process an SQL query

- Parsing
- Optimization
- Code generation
- Execution

Parsing

- Each statement consists of one or more <u>Query blocks</u>

```
SELECT e.ename, e.sal
  FROM (SELECT *
          FROM emp e
         WHERE e.sal > 300
           AND e.deptno IN (SELECT d.deptno
                              FROM dept d
                             WHERE d.dname IN ('SALES', 'ACCOUNTING'))) e
```

- Each statement has a predicate that may have one operand that is a query

```
SELECT e.ename, e.sal
  FROM (SELECT *
          FROM emp e
         WHERE e.sal > 300
           AND e.deptno IN (SELECT d.deptno
                              FROM dept d
                             WHERE d.dname IN ('SALES', 'ACCOUNTING'))) e
```

Optimizer

Query express **what** data to retrieve,

Not how to retrieve it

Optimizer

Query express **what** data to retrieve,

Not how to retrieve it

The Database must pick the **best** execution strategy through a process known as optimization.

Optimizer

- Example:

  SELECT name, title, sal

  FROM Emp, Job

  WHERE Emp.Job = Job.Job

  and Title = ‘CLERK’

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

JOB	TITLE
5	CLERK
6	TYPIST
9	SALES
12	MECHANIC

- Decide order to perform the different operators:
  - process “Title = ‘CLERK’” followed by the join
  - Process the join “Emp.Job = Job.Job” followed by “Title = ‘CLERK’”

Code Generator

- Code generator translates ASL(out put of optimizer) tree to executable machine code

ASL

1.Store in DB for later
2. execute

After executing

* Code calls upon the RSS(Research Storage System)
* Via RSI(Storage System Interface)

RSS(Research Storage System)

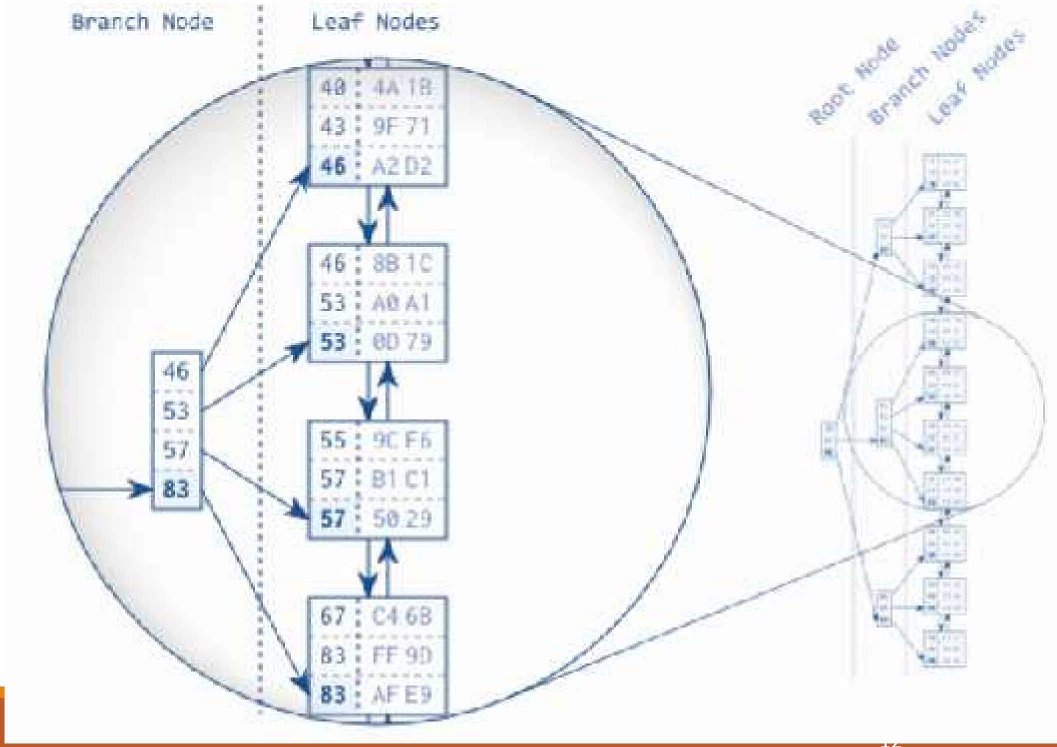
- Maintains physical storage of relations, access paths, locking, logging
- Relations are stored as a collection of tuples
- Tuples are stored on 4K pages; pages are organized into segments
- Pages are organized into logical units called segments.
- Segments may contain one or more relations
  - Each tuple is tagged with the identification of the relation to which it belongs
- At most one relation per segment.
- Tuples are accessed via a scan: segment scan or index scan

Segment scan

- A series of NEXTs on a segment scan examines all pages of the segment which contain tuple.
- Returns those tuples belonging to the given relations
- All the non-empty pages of a segment will be touched

Index scan

- These indexes are stored on separate pages from those containing the relation tuples.
- Indexes are implemented as B-tree



HOW ?

- Enumerating the different execution plans,
- Estimate the cost of performing each plan,
- Pick the cheapest plan.

- What is definition of cost?

HOW ?

- Enumerating the different execution plans,
- Estimate the cost of performing each plan,
- Pick the cheapest plan.

- What is definition of cost?
  - COST = Page fetches + W * (RSI Calls)

Cost computation

- Formulates a cost prediction for each access plan, using the following cost formula:

  - COST = Page fetches + W * (RSI Calls)
  - cost = IO costs + W * CPU costs

- W is an adjustable weighting factor between I/O and CPU.
- RSI calls is an approximation for CPU utilization.

Database Catalog

Statistics on the relations

- **For each relation T**
  - NCARD(T) , the cardinality of relation T
  - TCARD(T) , the number of pages in the Segment that hold tuples of relation T
  - P(T) , the fraction of data pages in the segment that hold tuples of relation T
    - P(T) = TCARD(T) / (# of non-empty pages in the segment)
- **For each index I on relation T**
  - ICARD(I) , number of distinct keys in index
  - NINDX(I) , the number of pages in index I

StudentID	Lastname	Firstname	Gender
101	Smith	John	M
102	Jones	James	M
103	Mayo	Ann	F
104	Jones	George	M
105	Smith	Suse	F

NCARD(StudentID) = 5
NCARD(Lastname) = 3

Next step in query Optimization

- Calculate a selectivity factor ‘F’ for each boolean factor in the predicate list
- For each relation, calculate the **cost** of scanning
- Find the efficient path for executing the query

Selectivity Factor

Columns	Selectivity Factor
attr = value	F = 1/ICARD(attr index) – if index exists F = 1/10 otherwise
attr1 = attr2	F = 1/max(ICARD(I1),ICARD(I2)) or F = 1/ICARD(Ii) – if only index i exists, or F = 1/10
val1 < attr < val2	F = (value2-value1)/(high key-low key) F = 1/4 otherwise
expr1 or expr2	F = F(expr1)+F(expr2)–F(expr1)*F(expr2)
expr1 and expr2 F	F = F(expr1) * F(expr2)
NOT expr	F = 1 – F(expr)

Relation Cost

- Based on different situation like having index ,clustered index, non-clustered index, group by or order by in our query block, the cost of relation will be calculated.

Cost Formulas

SITUATION	COST
Unique index matching an equal predicate	1 + 1 + W
Clustered index I matching one or more boolean factors	F(preds)* (NINDX(I)+TCARD) +W*RSICARD
Non clustered index I matching one or more boolean factors	F(preds)* (NINDX(I)+NCARD) + W*RSICARD F(preds)* (NINDX(I)+TCARD) + W*RSICARD
Clustered index I not matching any boolean factors	(NINDX(I) + TCARD) + W*RSICARD

Different number of relations

- Single relation
- 2-way join
- N-way join

Access path selection for joins

- **Outer relation :** Relation from which a tuple will be retrieved first
- **Inner relation :** Relation from which tuples will be retrieved, depending on the values obtained in the outer relation tuple.
- **Join predicate :** A predicate which relates columns of two tables to be joined.
- **Join column:** The column referenced in a join predicate

Join Method

1. **Nested Loops :** The scan on the outer relation is opened and the first tuple is retrieved. For each outer tuple obtained, a scan is opened on the inner relation to retrieve one at a time all tuples of inner relation that satisfy the join Predicate.

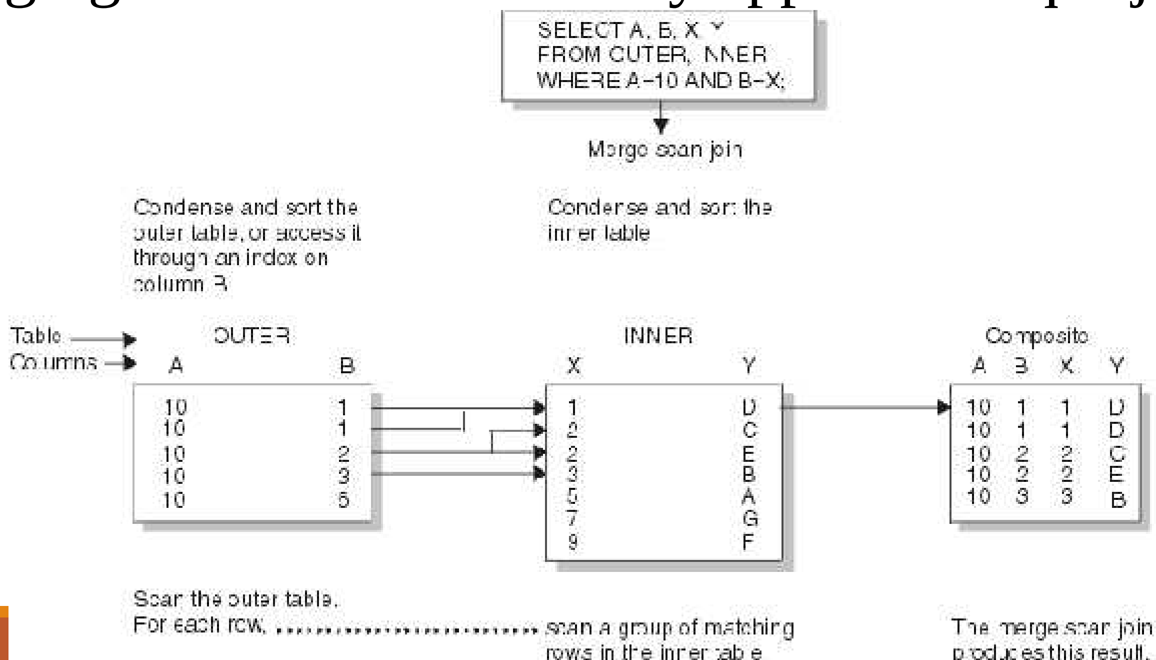
Table	JOIN TYPE
T1	range
T2	ref
T3	all

```
For each row in t1 matching range {
    For each row in t2 matching reference key {
        For each row in t3 {
          if row satisfies join conditions,
              do
        }}}
```

Join Method

2. Merging Scan : Relations are scanned in join column order.

Merging Scans method is only applied to equi-joins



Cost of different Join

- The cardinality of the join of n relations is the same
- The cost of joining in different orders can be different

Order of join

If there are n relations then there are n! ways of joining.

In joining relations t1, t2, t3, ..., tn only those orderings ti1, ti2, ti3, ..., tin are examined in which for all j (j=2, ..., n) either

(1) tij has at least one join predicate with some relation tik, where k<j or

(2) for all k > j, tik has no join predicate with ti1, ti2, ..., or ti(j-1)

Example: Let T1, T2, T3 be relations such that there are join predicates between T1 and T2 and between T2 and T3 on different columns than on T1-T2 join then

T1 T2 T3 T2 T1 T3 T3 T1 T2 (exclude)

T1 T3 T2 (exclude) T2 T3 T1 T3 T2 T1

Order of join

If there are n relations then there are n! ways of joining.

Order of join

If there are n relations then there are n! ways of joining.

A heuristic is used to reduce he join order permutations.

Computation of costs

- **C-outer(path1)** be the cost of scanning the outer relation via path 1, **N** be the cardinality of the outer relation tuples which satisfy the applicable predicates:

  N = (Product of cardinalities of all relations T of the join so far) *

  (Product of selectivity factors of all applicable predicates)

- **C-inner(path2)** be the cost of scanning the inner relation
- **C-nested loop join (path1,path2)** = C-outer(path1) + N * C-inner (path2)
- **C-merge (path1,path2)** = C-outer(path1) + N * C-inner (path2)

C-sort() includes the cost of retrieving the data, sorting the data, which may involve several passes and putting the results into a temporary list.

Example of Tree

EMP

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

DEPT

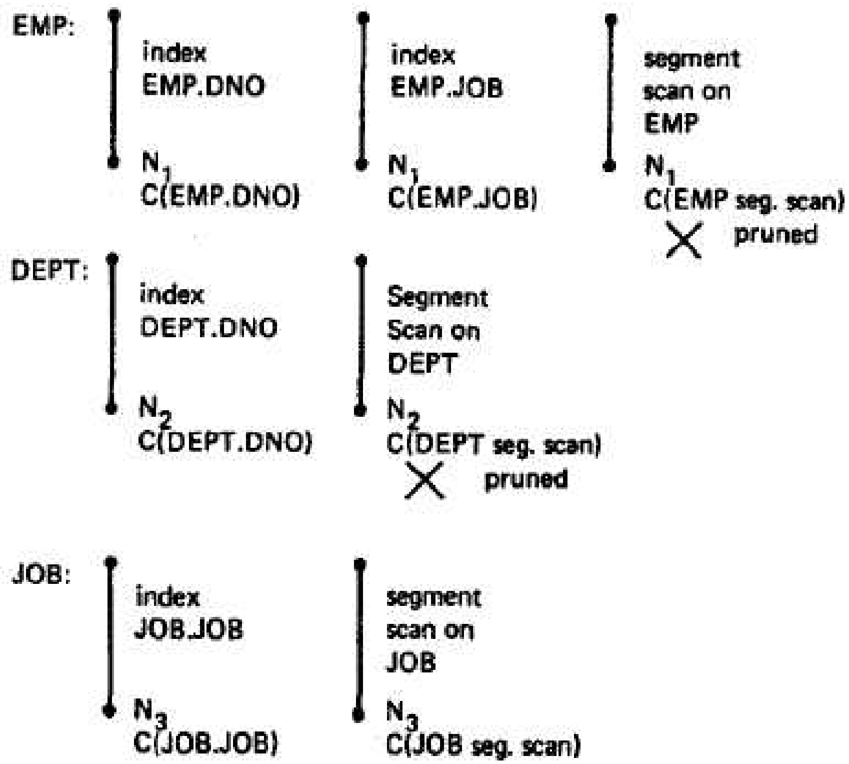
DNO	DNAME	LOC
50	MFG	DENVER
50	BILLING	BOULDER
51	SHIPPING	DENVER

JOB

JOB	TITLE
5	CLERK
6	TYPIST
8	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
```

Example of Tree



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

JOB	TITLE
5	CLERK
6	TYPIST
9	SALES
12	MECHANIC

Example of Tree

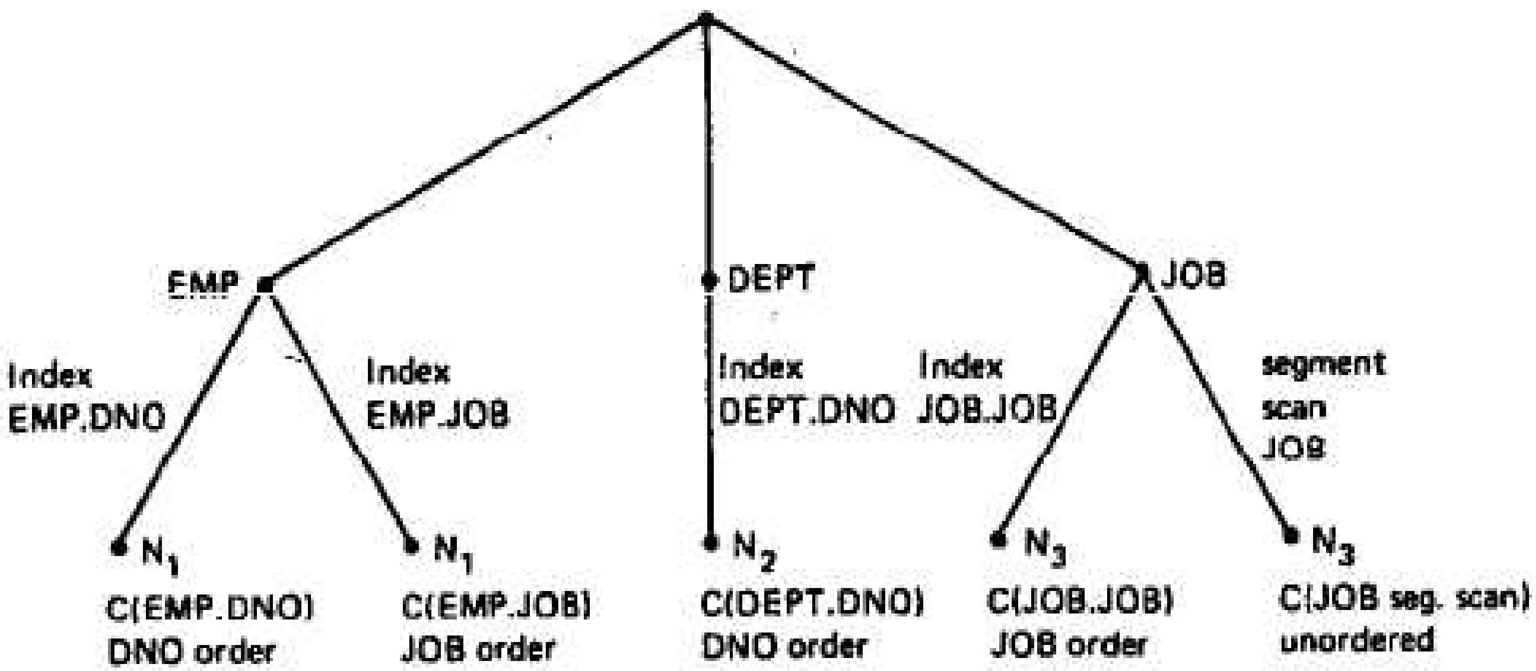


Figure 3. Search tree for single relations

Example of Tree

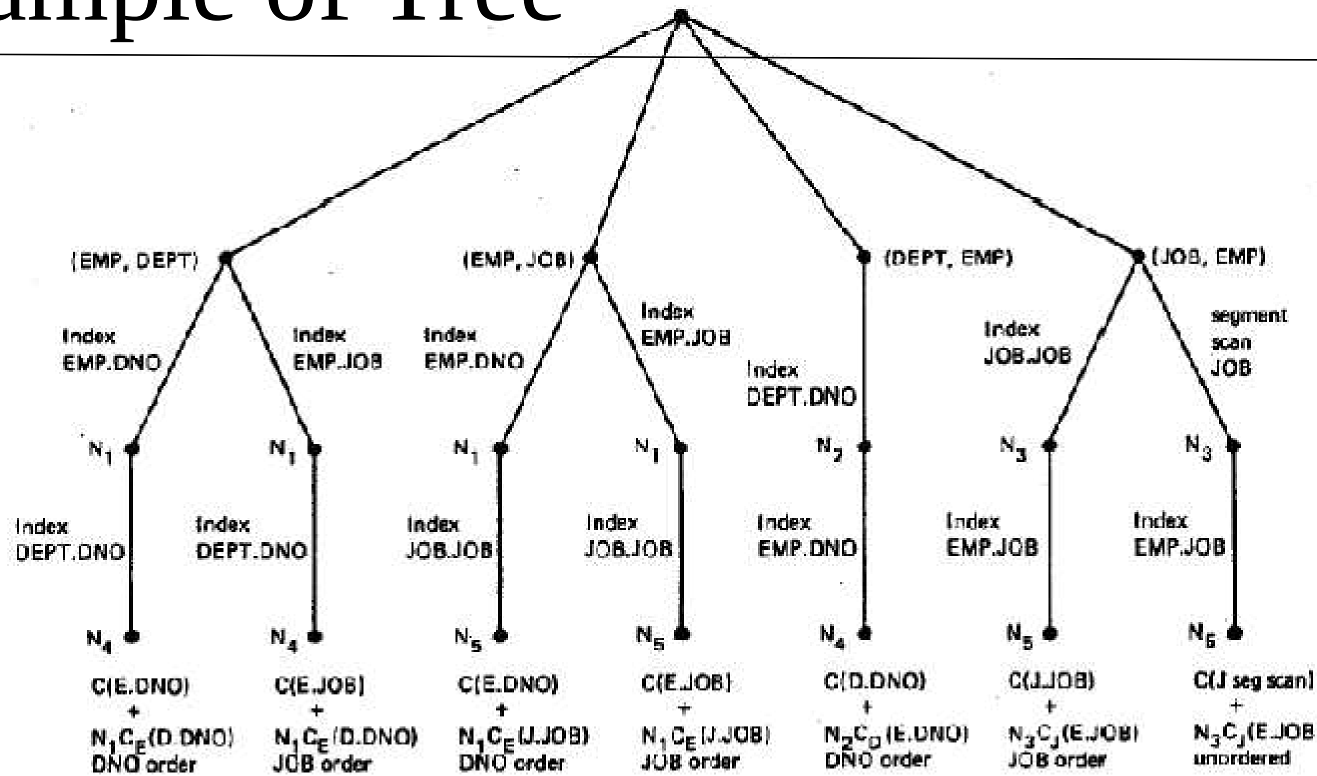


Figure 4. Extended search tree for second relation (nested loop join)

Example of Tree

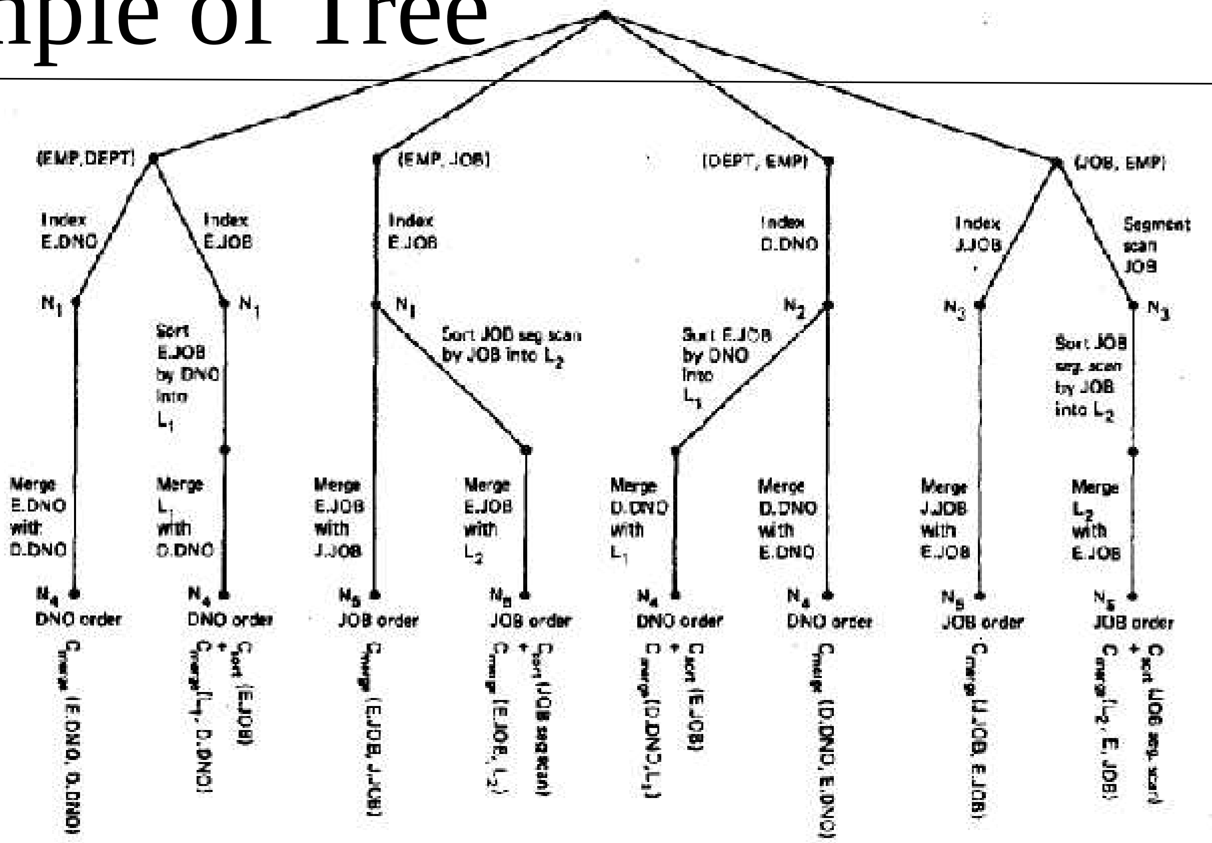


Figure 5. Extended search tree for second relation (merge join)

Nested Queries

- The most deeply nested Subquery is evaluated before the main query and is evaluated only once.

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

Return a single value

```
SELECT NAME
FROM EMPLOYEE
WHERE DEPNO IN
        ( SELECT DEPNO
          FROM DEPARTMENT
           WHERE LOCATION="DENVER"  )
```

Return a set of value

Correlation Subquery

- Subquery may contain a reference to a value obtained from a tuple from higher level.

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

Conclusion

- Database management systems can support non-procedural query languages with performance comparable to those supporting the current more procedural languages.

Q & A