CSE 444: Database Internals

Lecture 12

Query Optimization (part 3)

Announcements

- Lab 2 deadline EXTENDED by 24 hours
- Lab 2 quiz moved to Monday

Selinger Optimizer History

- 1960's: first database systems
  - Use tree and graph data models
- 1970: Ted Codd proposes relational model
  - E.F. Codd. A relational model of data for large shared data banks. Communications of the ACM, 1970
- 1974: System R from IBM Research
  - One of first systems to implement relational model
- 1979: Seminal query optimizer paper by P. Selinger et. al.
  - Invented cost-based query optimization
  - Dynamic programming algorithm for join order computation

References

- P. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price. Access Path Selection in a Relational Database Management System. Proceedings of ACM SIGMOD, **1979**. Pages 22-34.

Selinger Algorithm

Selinger enumeration algorithm considers

- Different logical and physical plans *at the same time*
- Cost of a plan is IO + CPU
- Concept of *interesting order* during plan enumeration
  - Same order as that requested by ORDER BY or GROUP GY
  - Or order on attributes that appear in equi-join predicates
    - Because they may enable cheaper sort-merge joins later

More about the Selinger Algorithm

- Step 1: Enumerate all access paths for a single relation
  - File scan or index scan
  - Keep the cheapest for each *interesting order*
- Step 2: Consider all ways to join two relations
  - Use result from step 1 as the outer relation
  - Consider every other possible relation as inner relation
  - Estimate cost when using sort-merge or nested-loop join
  - Keep the cheapest for each *interesting order*
- Steps 3 and later: Repeat for three relations, etc.

Example From Selinger Paper

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
8	SALES
12	MECHANIC

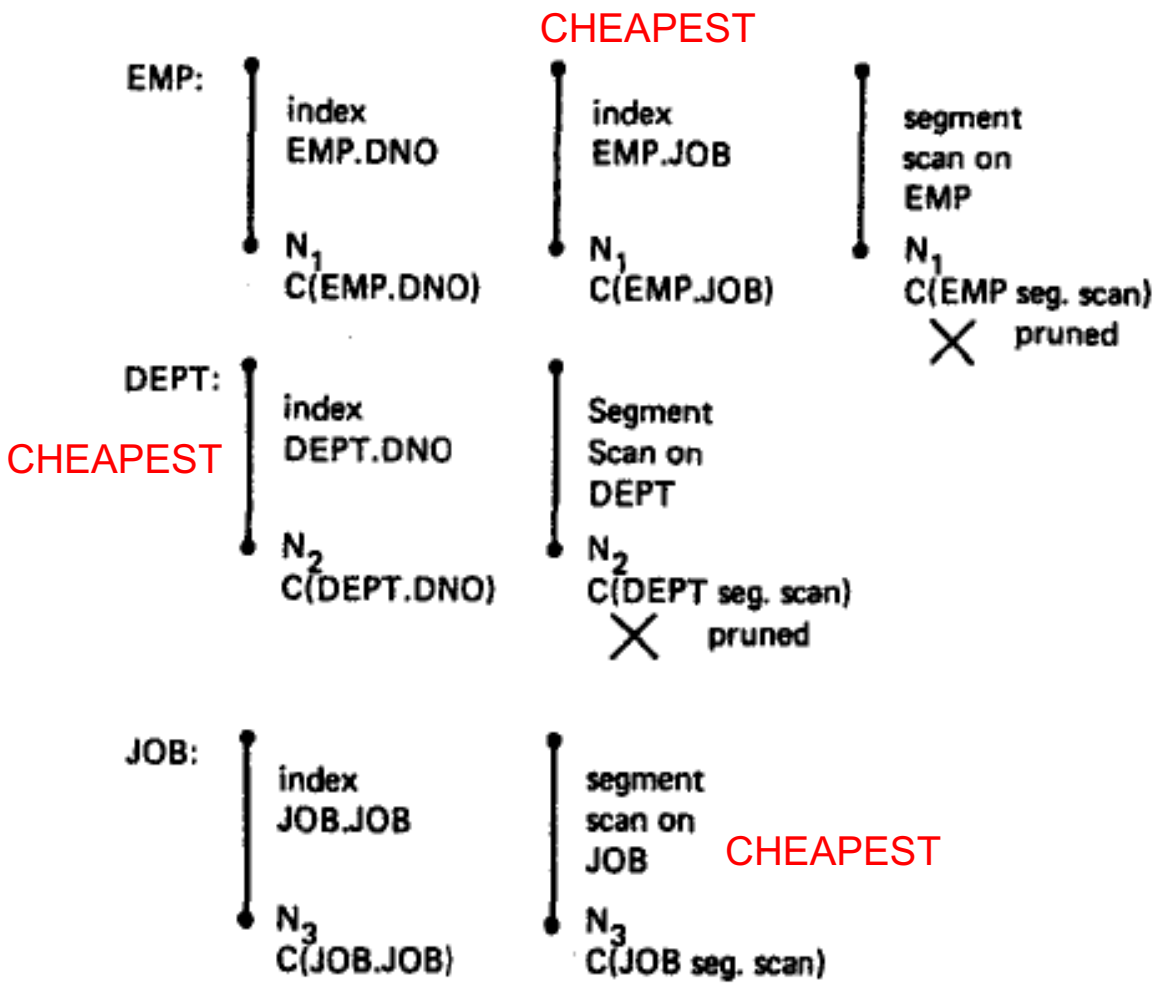
```
SELECT   NAME,TITLE,SAL,DNAME
FROM     EMP,DEPT,JOB
WHERE    TITLE='CLERK'
AND      LOC='DENVER'
AND      EMP.DNO=DEPT.DNO
AND      EMP.JOB=JOB.JOB
```

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

Figure 1. JOIN example

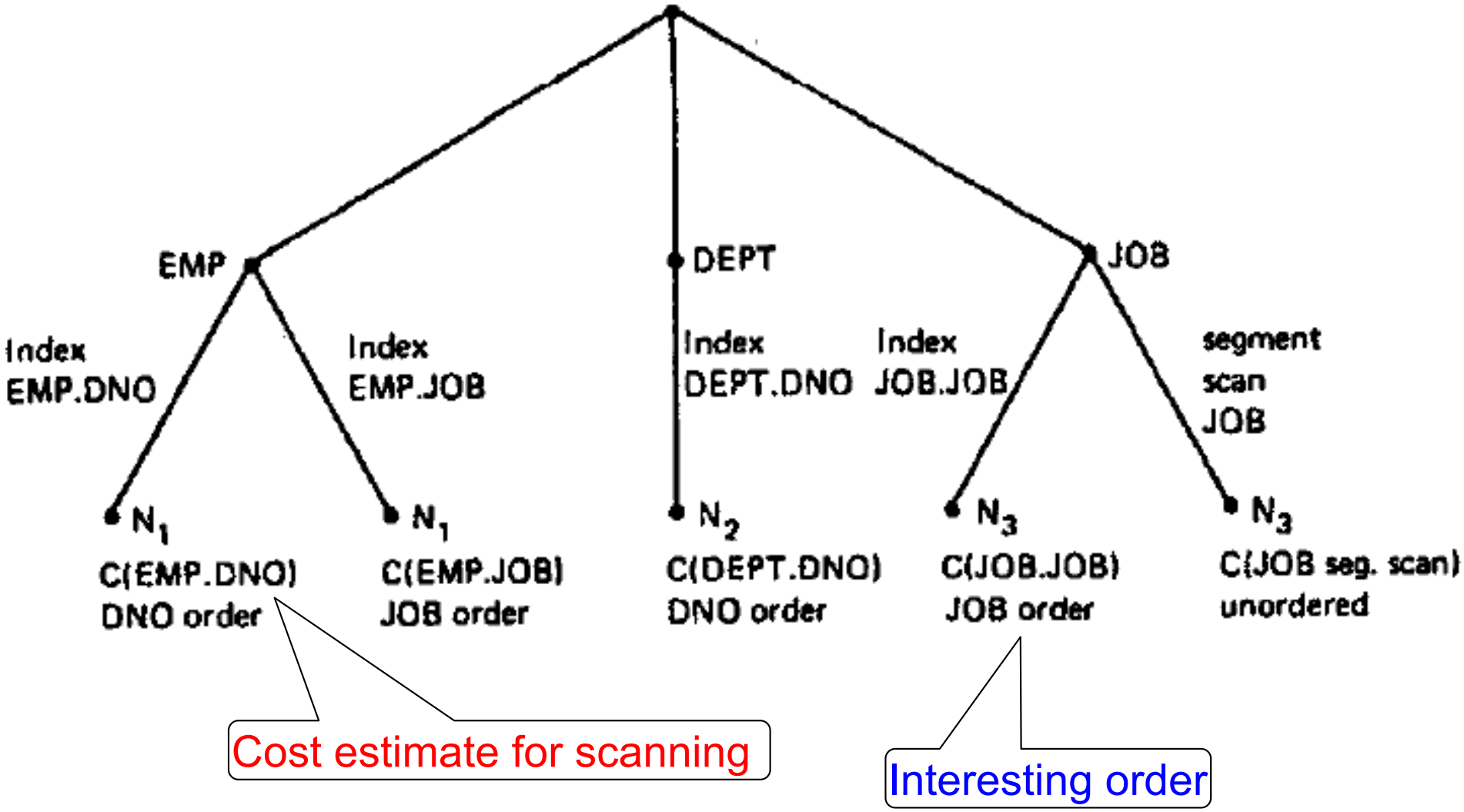
Step1: Access Path Selection for Single Relations

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



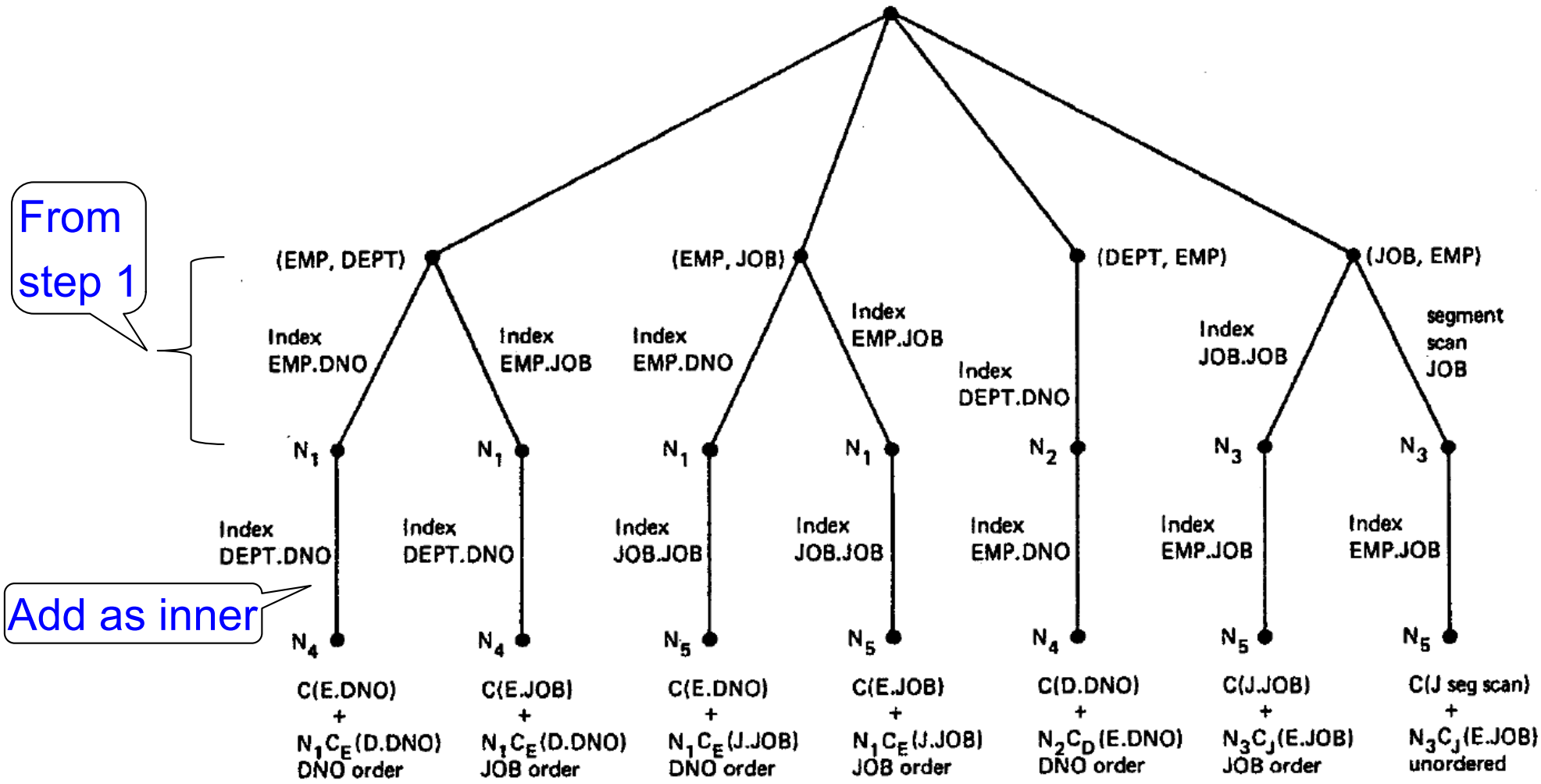
SELECT NAME, TITLE, SAL, DNAME
FROM EMP, DEPT, JOB
WHERE TITLE='CLERK' AND LOC='DENVER' AND EMP.DNO=DEPT.DNO AND EMP.JOB=JOB.JOB

Step1: Access Path Selection for Single Relations
Resulting Plan Search Tree for Single Relations



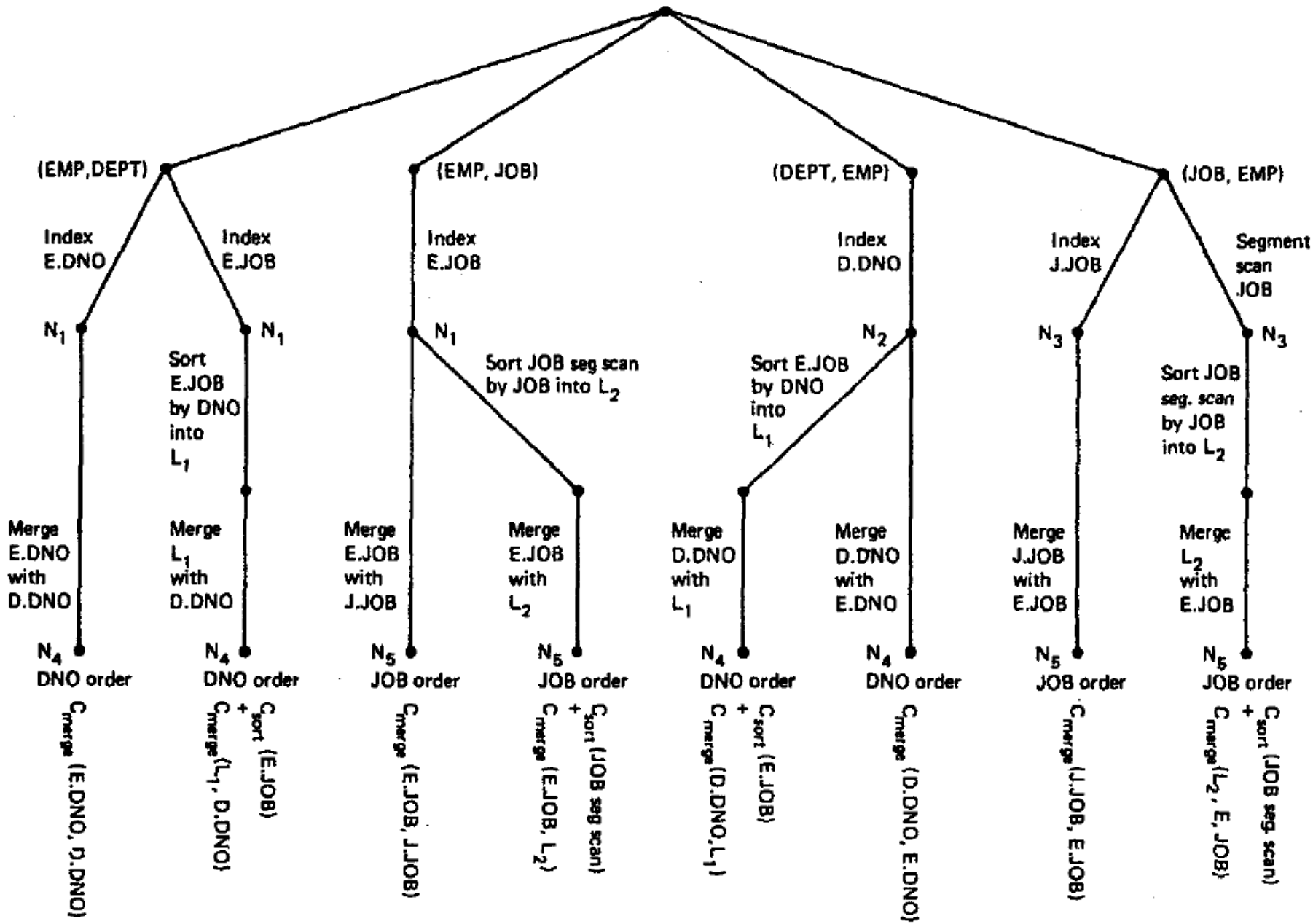
SELECT NAME, TITLE, SAL, DNAME
FROM EMP, DEPT, JOB
WHERE TITLE='CLERK' AND LOC='DENVER' AND EMP.DNO=DEPT.DNO AND EMP.JOB=JOB.JOB

Step2: Pairs of Relations (nested loop joins)

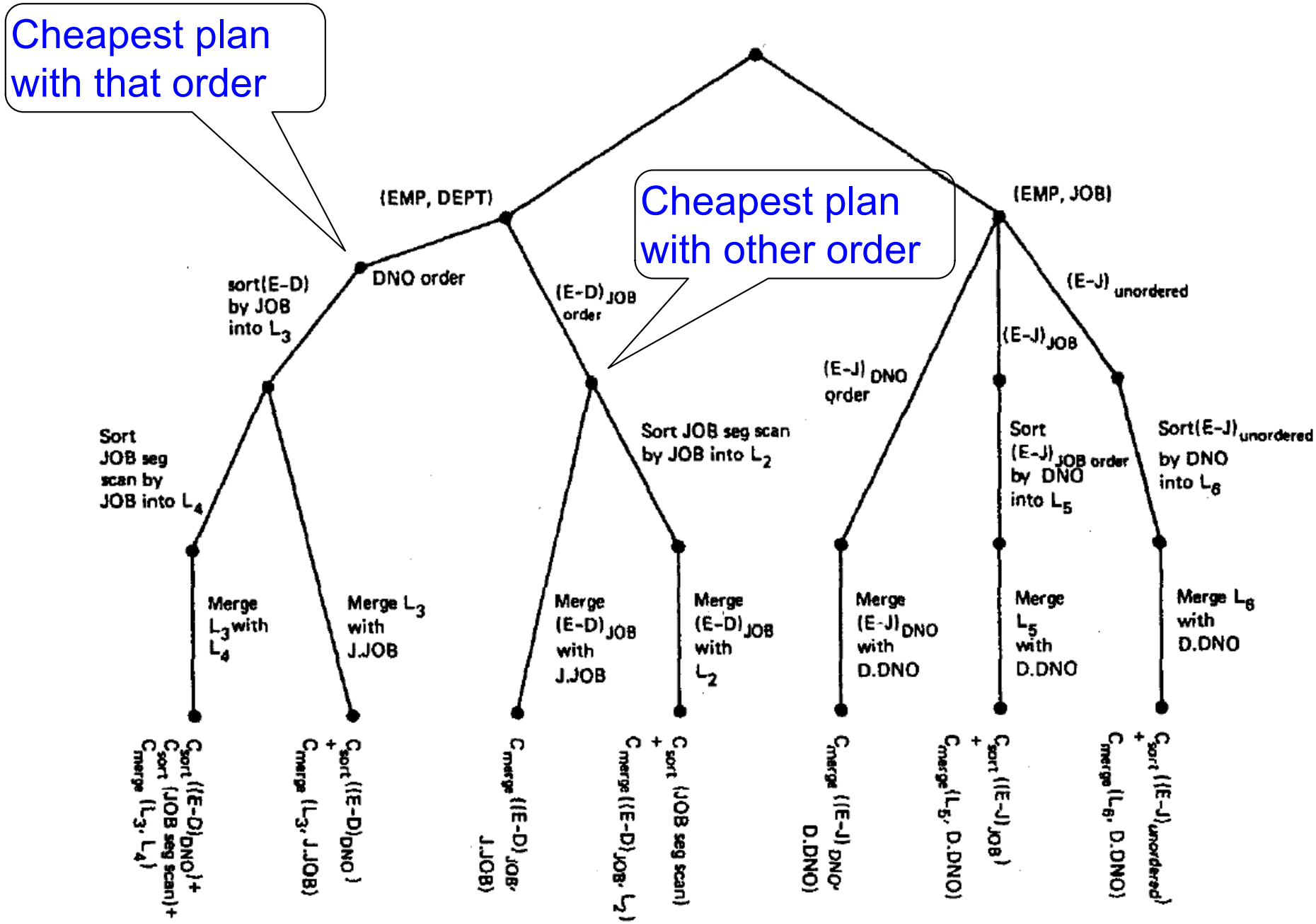


SELECT NAME, TITLE, SAL, DNAME
FROM EMP, DEPT, JOB
WHERE TITLE='CLERK' AND LOC='DENVER' AND EMP.DNO=DEPT.DNO AND EMP.JOB=JOB.JOB

Step2: Pairs of Relations (sort-merge joins)



Step3:Add Third Relation (sort-merge join)



Next Example Acks

Implement Selinger optimizer in SimpleDB

Designed to help you with Lab 5

Many following slides from Sam Madden at MIT

Dynamic Programming

OrderJoins:

SimpleDB Lab5:
you implement orderJoins

R = set of relations to join

For d = 1 to N: /* where N = |R| */

For S in {all size-d subsets of R}:

Use: enumerateSubsets

optjoin(S) = (S – a) join a,

where a is the single relation that minimizes:

cost(**optjoin**(S – a)) +

Use:
computerCostAndCardOfSubplan

min.cost to join (S – a) with a +

min.access cost for a

Note: **optjoin**(S-a) is cached from previous iterations

Example

Subplan S	optJoin(S)	Cost(OptJoin(S))
A		

- **orderJoins(A, B, C, D)**
- Assume all joins are NL

Example

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
C	Seq scan	120
D	B+tree scan	400

- **orderJoins(A, B, C, D)**
- Assume all joins are NL
- d = 1
  - A = best way to access A (sequential scan, predicate-pushdown on index, etc)
  - B = best way to access B
  - C = best way to access C
  - D = best way to access D
- Total number of steps: choose(N, 1)

Example

- **orderJoins(A, B, C, D)**
- d = 2
  - {A,B} = AB or BA
    use previously computed
    best way to access A and B

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
…		

Example

- **orderJoins(A, B, C, D)**
- d = 2
  - {A,B} = AB or BA
    use previously computed
    best way to access A and B

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
…		
{A, B}	BA	156

Example

- **orderJoins(A, B, C, D)**

- d = 2
  - {A,B} = AB or BA
    use previously computed
    best way to access A and B
  - {B,C} = BC or CB

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
…		
{A, B}	BA	156
{B, C}	BC	98

Example

- **orderJoins(A, B, C, D)**

- d = 2
  - {A,B} = AB or BA
    use previously computed
    best way to access A and B
  - {B,C} = BC or CB

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
…		
{A, B}	BA	156
{B, C}	BC	98

Example

- **orderJoins(A, B, C, D)**

- d = 2
  - {A,B} = AB or BA
    use previously computed
    best way to access A and B
  - {B,C} = BC or CB
  - {C,D} = CD or DC
  - {A,C} = AC or CA
  - {B,D} = BD or DB
  - {A,D} = AD or DA

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
…		
{A, B}	BA	156
{B, C}	BC	98
……..		

Example

- **orderJoins(A, B, C, D)**

- d = 2
  - {A,B} = AB or BA
    use previously computed
    best way to access A and B
  - {B,C} = BC or CB
  - {C,D} = CD or DC
  - {A,C} = AC or CA
  - {B,D} = BD or DB
  - {A,D} = AD or DA
- Total number of steps: choose(N, 2) × 2

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
…		
{A, B}	BA	156
{B, C}	BC	98
……..		

Example

- **orderJoins(A, B, C, D)**
- d = 3
  - {A,B,C} =
    Remove A: compare A({B,C}) to ({B,C})A

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
….		
{A, B}	BA	156
{B, C}	BC	98
….		

Example

- **orderJoins(A, B, C, D)**
- d = 3
  - {A,B,C} =
    Remove A: compare A({B,C}) to ({B,C})A

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
….		
{A, B}	BA	156
{B, C}	BC	98
….		

optJoin(B,C) and its cost are already cached in table

Example

- **orderJoins(A, B, C, D)**
- d = 3
  - {A,B,C} =
    Remove A: compare A({B,C}) to ({B,C})A
    Remove B: compare B({A,C}) to ({A,C})B
    Remove C: compare C({A,B}) to ({A,B})C

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
….		
{A, B}	BA	156
{B, C}	BC	98
….		
{A, B, C}	BAC	500
……..		

optJoin(B,C) and its cost are already cached in table

Example

- **orderJoins(A, B, C, D)**
- d = 3
  - {A,B,C} =
    Remove A: compare A({B,C}) to ({B,C})A
    Remove B: compare B({A,C}) to ({A,C})B
    Remove C: compare C({A,B}) to ({A,B})C

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
….		
{A, B}	BA	156
{B, C}	BC	98
…		
{A, B, C}	BAC	500
……..		

optJoin(B,C) and its cost are already cached in table

Example

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
….		
{A, B}	BA	156
{B, C}	BC	98
…		
{A, B, C}	BAC	500
……..		

- **orderJoins(A, B, C, D)**
- d = 3
  - {A,B,C} =
    Remove A: compare A({B,C}) to ({B,C})A
    Remove B: compare B({A,C}) to ({A,C})B
    Remove C: compare C({A,B}) to ({A,B})C
  - {A,B,D} =
    Remove A: compare A({B,D}) to ({B,D})A
    …
  - {A,C,D} =…
  - {B,C,D} =…
- Total number of steps: choose(N, 3) × 3 × 2

optJoin(B,C) and its cost are already cached in table

Example

- **orderJoins(A, B, C, D)**

Subplan S	optJoin(S)	Cost(OptJoin(S))
A	Index scan	100
B	Seq. scan	50
{A, B}	BA	156
{B, C}	BC	98
{A, B, C}	BAC	500
{B, C, D}	DBC	150
……..		

optJoin(B, C, D) and its cost are already cached in table

- d = 4
  - {A,B,C,D} =

Remove A: compare A({B,C,D}) to ({B,C,D})A
Remove B: compare B({A,C,D}) to ({A,C,D})B
Remove C: compare C({A,B,D}) to ({A,B,D})C
Remove D: compare D({A,B,C}) to ({A,B,C})D

- Total number of steps: choose(N, 4) × 4 × 2