CompSci 516
Data Intensive Computing Systems

Lecture 9
Query Optimization

Instructor: Sudeepa Roy
Announcements

• Project proposal due today
  – use template on sakai
  – submit to sakai

• Practice problem set#1 posted on sakai
  – try yourself before looking at the solutions!

• Tomorrow make up lecture “only” for students going to CS grad retreat on Friday
  – Thursday, LSRC D309, 4:40 pm, room accommodates ~10 people
  – Regular class on Friday
Reading Material

• [RG]
  – Query optimization: Chapter 15 (overview only)

• [GUW]
  – Chapter 16.2-16.7

• Original paper by Selinger et al. :
  – 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
  – No need to understand the whole paper, but take a look at the example (link on the course webpage)

Acknowledgement:
Some of the following slides have been created by adapting slides by Profs. Shivnath Babu and Magda Balazinska
Query Optimization
Query Blocks: Units of Optimization

- **Query Block**
  - No nesting
  - One SELECT., one FROM
  - At most one WHERE, GROUP BY, HAVING

- **SQL query**

- => parsed into a collection of query blocks

- => the blocks are optimized one block at a time

- Express single-block it as a relational algebra (RA) expression

```sql
SELECT S.sname
FROM Sailors S
WHERE S.age IN
  (SELECT MAX (S2.age)
   FROM Sailors S2
   GROUP BY S2.rating)
```
Cost Estimation

• For each plan considered, must estimate cost:

• Must estimate cost of each operation in plan tree.
  – Depends on input cardinalities
  – We’ve discussed how to estimate the cost of operations (sequential scan, index scan, joins, etc.)

• Must also estimate size of result for each operation in tree
  – gives input cardinality of next operators

• Also consider whether the output is sorted
Relational Algebra Equivalences

• Allow us to choose different join orders and to `push’ selections and projections ahead of joins.

• **Selections:**
  \[ \sigma_{c_1 \wedge \ldots \wedge c_n}(R) \equiv \sigma_{c_1}(\ldots \sigma_{c_n}(R)) \]  
  (Cascade)

  \[ \sigma_{c_1}(\sigma_{c_2}(R)) \equiv \sigma_{c_2}(\sigma_{c_1}(R)) \]  
  (Commute)

• **Projections:**
  \[ \pi_{a_1}(R) \equiv \pi_{a_1}(\ldots(\pi_{a_n}(R))) \]  
  (Cascade)

• **Joins:**
  \[ R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T \]  
  (Associative)

  \[ (R \bowtie S) \equiv (S \bowtie R) \]  
  (Commute)

There are many more intuitive equivalences, see 15.3.4 for details
Notation

• $T(R)$ : Number of tuples in $R$
• $B(R)$ : Number of blocks in $R$
• $V(R, A)$ : Number of distinct values of attribute $A$ in $R$
Query Optimization Problem

Pick the best plan from the space of physical plans
Cost-based Query Optimization

Challenge:

• Do not want to execute more than one plan

• Need to estimate the cost without executing the plan

“heuristic-based” optimizer (e.g. push selections down) have limited power and not used much
Cost-based Query Optimization

Pick the plan with least cost

Tasks:

1. Estimate the cost of individual operators
   done in Lecture 9

2. Estimate the size of output of individual operators
   today

3. Combine costs of different operators in a plan
   today

4. Efficiently search the space of plans today
Task 1 and 2
Estimating cost and size of different operators

- Size = #tuples, NOT #pages
- Cost = #page I/O
  - but, need to consider whether the intermediate relation fits in memory, is written back to/read from disk (or on-the-fly goes to the next operator), etc.
Desired Properties of Estimating Sizes of Intermediate Relations

Ideally,

• should give accurate estimates (as much as possible)
• should be easy to compute
• should be logically consistent
  – size estimate should be independent of how the relation is computed
  – e.g. which join algorithm/join order is used

• But, no “universally agreed upon” ways to meet these goals
Cost of Table Scan

Cost: $B(R)$
Size: $T(R)$

$T(R)$: Number of tuples in $R$
$B(R)$: Number of blocks in $R$
Cost of Index Scan

Cost: $B(R)$ – if clustered
     $T(R)$ – if unclustered

Size: $T(R)$

$T(R)$: Number of tuples in $R$
$B(R)$: Number of blocks in $R$

Note: size is independent of the implementation of the scan/index
Cost of Index Scan with Selection

\[ X = \sigma_{R.A > 50} R \]

Cost: \( B(R) \times f \) – if clustered
\[ T(R) \times f \] – if unclustered

Size: \( T(R) \times f \)

Reduction factor
\[ f = (\text{Max}(R.A) - 50) / (\text{Max}(R.A) - \text{Min}(R.A)) \]
assumes uniform distribution

\( T(R) \) : Number of tuples in \( R \)
\( B(R) \) : Number of blocks in \( R \)
Cost of Index Scan with Selection (and multiple conditions)

\[ X = \sigma_{R.A > 50 \text{ and } R.B = C} \]

Assume index on \((A, B)\)

Cost: \(B(R) \times f\) – if clustered
\(T(R) \times f\) – if unclustered

Size: \(T(R) \times f\)

Reduction factors

\(f_1 = \frac{\text{Max}(R.A) - 50}{\text{Max}(R.A) - \text{Min}(R.A)}\)

\(f_2 = \frac{1}{V(R, B)}\)

\(f = f_1 \times f_2\) (assumes independence and uniform distribution)

What is \(f_1\) if the first condition is \(100 > R.1 > 50\)?
Cost of Projection

\[ X = \pi_A R \]

Cost: depends on the method of scanning \( R \)

- \( B(R) \) for table scan or clustered index scan

Size: \( T(R) \)

- But tuples are smaller
- If you have more information on the size of the smaller tuples, can estimate \#I/O better

“....” Scan
Size of Join

Quite tricky
- If disjoint A and B values
  - then 0
- If A is key of R and B is foreign key of S
  - then T(S)
- If all tuples have the same value of R.A = S.B = x
  - then T(R) * T(S)

\[
R.A = S.B
\]

T(R) : Number of tuples in R
B(R) : Number of blocks in R
V(R, A) : Number of distinct values of attribute A in R
Size of Join

Two assumptions

1. Containment of value sets:
   • if $V(R, A) \leq V(S, B)$, then all A-values of R are included in B-values of S
   • e.g. satisfied when A is foreign key, B is key

2. Preservation of value sets:
   • $V(R \bowtie S, A) = V(R, A)$
   • $V(R \bowtie S, B) = V(S, B)$
   • No value is lost in join

$T(R)$ : Number of tuples in R
$B(R)$ : Number of blocks in R
$V(R, A)$ : Number of distinct values of attribute A in R
Size of Join

Reduction factor
\[ f = \frac{1}{\max(V(R, A), V(S, B))} \]

Size
\[ \text{Size} = T(R) \times T(S) \times f \]

- \( T(R) \): Number of tuples in \( R \)
- \( B(R) \): Number of blocks in \( R \)
- \( V(R, A) \): Number of distinct values of attribute \( A \) in \( R \)
Size of Join

Reduction factor
\[ f = \frac{1}{\max(V(R, A), V(S, B))} \]

Size = \( T(R) \times T(S) \times f \)

Why max?
- Suppose \( V(R, A) \leq V(S, B) \)
- The probability of a \( A \)-value joining with a \( B \)-value is \( \frac{1}{V(S.B)} = \text{reduction factor} \)
- Under the two assumptions stated earlier + uniformity

Assumes index on both \( A \) and \( B \)
if one index: \( \frac{1}{V(\ldots, \ldots)} \)
if no index: say \( \frac{1}{10} \)

\( T(R) \): Number of tuples in \( R \)
\( B(R) \): Number of blocks in \( R \)
\( V(R, A) \): Number of distinct values of attribute \( A \) in \( R \)
Task 3: Combine cost of different operators in a plan

With Examples
“Given” the physical plan

- Size = #tuples, NOT #pages
- Cost = #page I/O
  - but, need to consider whether the intermediate relation fits in memory, is written back to disk (or on-the-fly goes to the next operator) etc.
Example Query

Student (sid, name, age, address)
Book(bid, title, author)
Checkout(sid, bid, date)

Query:

```
SELECT S.name
FROM Student S, Book B, Checkout C
WHERE S.sid = C.sid
AND B.bid = C.bid
AND B.author = 'Olden Fames'
AND S.age > 12
AND S.age < 20
```
Assumptions

- **Student**: $S$, **Book**: $B$, **Checkout**: $C$

  - Sid, bid foreign key in $C$ referencing $S$ and $B$ resp.
  - There are 10,000 Student records stored on 1,000 pages.
  - There are 50,000 Book records stored on 5,000 pages.
  - There are 300,000 Checkout records stored on 15,000 pages.
  - There are 500 different authors.
  - Student ages range from 7 to 24.

Warning: a few dense slides next 😊
Physical Query Plan – 1

Q. Compute
1. the cost and cardinality in steps (a) to (d)
2. the total cost

Assumptions:
- Data is not sorted on any attributes
- For both in (a) and (b), outer relations fit in memory

(Tuple-based nested loop
  B inner)

(On the fly) (d) \( \Pi_{name} \)

(On the fly) (c) \( \sigma_{12<age<20 \land author = 'Olden Fames'} \)

(Tuple-based nested loop
  B inner)

(On the fly) (b)

(Tuple-based nested loop
  B inner)

(On the fly) (a)

S(sid,name,age,addr) T(S)=10,000
B(bid,title,author) T(B)=50,000
C(sid,bid,date) T(C)=300,000

B(S)=1,000
B(B)=5,000
B(C)=15,000

V(B,author) = 500
7 <= age <= 24

7 <= age <= 24

Student S (File scan)
Checkout C (File scan)
Book B (File scan)
\[ \text{Student S} (\text{sid, name, age, addr}) \]
\[ \text{B(bid, title, author)} \]
\[ \text{C(sid, bid, date)} \]

\[ \text{T(S)} = 10,000 \quad \text{B(S)} = 1,000 \quad \text{V(B, author)} = 500 \]
\[ \text{T(B)} = 50,000 \quad \text{B(B)} = 5,000 \]
\[ \text{T(C)} = 300,000 \quad \text{B(C)} = 15,000 \]

\[ 7 \leq \text{age} \leq 24 \]

\[ \text{Cost} = \]
\[ \text{B(S)} + \text{B(S)} \times \text{B(C)} \]
\[ = 1000 + 1000 \times 15000 \]
\[ = 15,001,000 \]

\[ \text{Cardinality} = \]
\[ \text{T(C)} = 300,000 \]

- foreign key join, output pipelined to next join
- Can apply the formula as well

\[ \text{T(S)} \times \text{T(C)} / \max (\text{V(S, sid)}, \text{V(C, sid)}) \]
\[ = \text{T(C)} \]

since \( \text{V(S, sid)} \geq \text{V(C, sid)} \) and
\[ \text{T(S)} = \text{V(S, sid)} \]
\[
\begin{align*}
S(\text{sid}, \text{name}, \text{age}, \text{addr}) & \quad T(S)=10,000 \\
B(\text{bid}, \text{title}, \text{author}) & \quad T(B)=50,000 \\
C(\text{sid}, \text{bid}, \text{date}) & \quad T(C)=300,000 \\
\end{align*}
\]

\[
\begin{align*}
B(S) & = 1,000 \\
B(B) & = 5,000 \\
B(C) & = 15,000 \\
V(\text{author}) & = 500 \\
7 \leq \text{age} \leq 24
\end{align*}
\]

Cost =
\[
T(S \bowtie C) * B(B) = 300,000 * 5,000 = 15 * 10^8
\]

Cardinality =
\[
T(S \bowtie C) = 300,000
\]

- foreign key join, don’t need scanning for outer relation

(On the fly) (d) \( \Pi_{\text{name}} \)

(On the fly) (c) \( \sigma_{12<\text{age}<20} \land \text{author} = \text{‘Olden Fames’} \)

(Tuple-based nested loop
B inner)

(Book B
(File scan)

(Student S
(File scan)

(Checkout C
(File scan)

\[(c, d)\]

\[(\text{On the fly}) (d) \Pi_{\text{name}} \]

\[(\text{On the fly}) (c) \sigma_{\text{12}<\text{age}<\text{20}} \land \text{author} = '\text{Olden Fames}' \]

(Tuple-based nested loop, B inner)

(On the fly) bid

(Page-oriented -nested loop, S outer, C inner)

Student S (File scan)

Checkout C (File scan)

Book B (File scan)

Cost = 0 (on the fly)

Cardinality = \[300,000 \times \frac{1}{500} \times \frac{7}{18} = 234 \text{ (approx)}\]

(assuming uniformity and independence)
\begin{align*}
S(&\text{sid, name, age, addr}) & \quad T(S) = 10,000 \\
B(&\text{bid, title, author}) & \quad T(B) = 50,000 \\
C(&\text{sid, bid, date}) & \quad T(C) = 300,000 \\
B(S) & = 1,000 \\
B(B) & = 5,000 \\
B(C) & = 15,000 \\
V(B, \text{author}) & = 500 \\
7 & \leq \text{age} \leq 24
\end{align*}

\begin{align*}
\text{(Total)} \\
(\text{On the fly}) & \quad (d) \quad \Pi_{\text{name}} \\
(\text{On the fly}) & \quad (c) \quad \sigma_{12<\text{age}<24 \land \text{author} = \text{\textquote{Olden Fames}}} \\
(\text{Tuple-based nested loop}) & \quad \text{B inner} \\
(\text{Page-oriented} & \quad \text{-nested loop,} \\
& \quad \text{S outer, C inner})
\end{align*}

\begin{align*}
\text{Book B} \\
(\text{File scan})
\end{align*}

\begin{align*}
\text{Student S} & \quad (\text{File scan}) \\
\text{Checkout C} & \quad (\text{File scan})
\end{align*}

Total cost = 1,515,001,000

Final cardinality = 234 (approx)
Physical Query Plan – 2

Q. Compute
1. the cost and cardinality in steps (a) to (g)
2. the total cost

Assumptions:
• Unclustered B+tree index on B.author
• Clustered B+tree index on C.bid
• All index pages are in memory
• Unlimited memory

S(sid,name,age,addr)  T(S)=10,000  B(S)=1,000  V(B,author) = 500
B(bid,title,author)  T(B)=50,000  B(B)=5,000  7 <= age <= 24
C(sid,bid,date)  T(C)=300,000  B(C)=15,000

(a) σ_{author = ‘Olden Fames’}

Book B
(Index scan)

Bid

Block nested loop S inner

(Book nested loop

On the fly)

(f) σ_{12<age<20}

(On the fly)

(g) \Pi_{name}

Assuming:
• Unclustered B+tree index on B.author
• Clustered B+tree index on C.bid
• All index pages are in memory
• Unlimited memory

Student S
(File scan)

Checkout C
(Index scan)
S(sid, name, age, addr)  T(S) = 10,000  B(S) = 1,000  V(B, author) = 500
B(bid, title, author): Un. B+ on author  T(B) = 50,000  B(B) = 5,000  7 <= age <= 24
C(sid, bid, date): Cl. B+ on bid  T(C) = 300,000  B(C) = 15,000

\( B(S) = 1,000 \)
\( B(B) = 5,000 \)
\( B(C) = 15,000 \)

Cost = \( T(B) / V(B, \text{author}) = 50,000/500 \)
\( = 100 \) (unclustered)

Cardinality = 100
S(sid, name, age, addr)
B(bid, title, author): Un. B+ on author
C(sid, bid, date): Cl. B+ on bid

T(S) = 10,000  B(S) = 1,000  V(B, author) = 500
T(B) = 50,000  B(B) = 5,000
T(C) = 300,000  B(C) = 15,000
7 <= age <= 24

(a) σ_{author = 'Olden Fames'}
(b) Π_{bid}
(c) Π_{sid} (On the fly)
(d) Π_{sid} (On the fly)
(e) (Block nested loop, S inner)
(f) σ_{12<age<20}
(g) Π_{name} (On the fly)

Cost = 0 (on the fly)
Cardinality = 100

Student S
(File scan)

Checkout C
(Index scan)

Book B
(Index scan)

Duke CS, Fall 2016
CompSci 516: Data Intensive Computing Systems
\[ S(\text{sid}, \text{name}, \text{age}, \text{addr}) \]
\[ B(\text{bid}, \text{title}, \text{author}) : \text{Un. B+ on author} \]
\[ C(\text{sid}, \text{bid}, \text{date}) : \text{Cl. B+ on bid} \]

\[ \begin{align*}
T(S) &= 10,000 & B(S) &= 1,000 \\
T(B) &= 50,000 & B(B) &= 5,000 \\
T(C) &= 300,000 & B(C) &= 15,000 \\
\end{align*} \]

\[ \text{V(B, author)} = 500 \]
\[ 7 \leq \text{age} \leq 24 \]

\[ \begin{align*}
\text{Cost} \leq & 100 \times 2 = 200 \\
\text{Cardinality} = & 100 \times 6 = 600 \\
& = 100 \times \frac{T(C)}{\text{MAX}(100, V(C, \text{bid}))} \\
& \text{assuming} \\
& V(C, \text{bid}) = V(B, \text{bid}) = T(B) = 50,000 \]
S(sid, name, age, addr)
B(bid, title, author): Un. B+ on author
C(sid, bid, date): Cl. B+ on bid

\[ T(S) = 10,000 \quad B(S) = 1,000 \quad V(B, author) = 500 \]
\[ 7 \leq age \leq 24 \]
\[ T(B) = 50,000 \quad B(B) = 5,000 \]
\[ T(C) = 300,000 \quad B(C) = 15,000 \]

Cost = 0 (on the fly)
Cardinality = 600
S(sid,name,age,addr)
B(bid,title,author): Un. B+ on author
C(sid,bid,date): Cl. B+ on bid

T(S)=10,000  B(S)=1,000  V(B,author) = 500
T(B)=50,000  B(B)=5,000  7 <= age <= 24
T(C)=300,000  B(C)=15,000

V(B,author) = 500
7 <= age <= 24

Outer relation is already in (unlimited) memory need to scan S relation
Cost = B(S) = 1000
Cardinality = 600
S(sid, name, age, addr)  
B(bid, title, author): Un. B+ on author  
C(sid, bid, date): Cl. B+ on bid  

\[ (\text{Block nested loop} \ S \ \text{inner}) \]

\[ (\text{Indexed-nested loop,} \ B \ \text{outer,} \ C \ \text{inner}) \]

\[ (\text{On the fly}) \ (g) \ \Pi_{\text{name}} \]

\[ (f) \ \sigma_{12<\text{age}<20} \]

\[ (\text{On the fly}) \ (e) \]

\[ (\text{On the fly}) \ (d) \ \Pi_{\text{sid}} \]

\[ (\text{On the fly}) \ (b) \ \Pi_{\text{bid}} \]

\[ (\text{On the fly}) \ (a) \ \sigma_{\text{author} = \text{Olden Fames}} \]

\[ \sigma_{12<\text{age}<20} \]

\[ \Pi_{\text{name}} \]

\[ \text{Student} \ S \ \text{(File scan)} \]

\[ \text{Checkout} \ C \ \text{(Index scan)} \]

\[ \text{Book} \ B \ \text{(Index scan)} \]

\[ T(S) = 10,000 \quad B(S) = 1,000 \quad V(B, \text{author}) = 500 \]

\[ B(B) = 5,000 \quad B(C) = 15,000 \quad 7 <= \text{age} <= 24 \]

\[ 7/18 = 234 \ (\approx) \]
\( S(\text{sid, name, age, addr}) \)
\( B(\text{bid, title, author}): \text{Un. B+ on author} \)
\( C(\text{sid, bid, date}): \text{Cl. B+ on bid} \)

\[
\begin{align*}
S(\text{sid, name, age, addr}) & \quad T(S) = 10,000 & B(S) = 1,000 & V(B, \text{author}) = 500 \\
B(\text{bid, title, author}) & \quad T(B) = 50,000 & B(B) = 5,000 & 7 \leq \text{age} \leq 24 \\
C(\text{sid, bid, date}) & \quad T(C) = 300,000 & B(C) = 15,000
\end{align*}
\]

\[ (\text{Block nested loop, S inner}) \]
\[ (\text{Indexed-nested loop, B outer, C inner}) \]
\[ (\text{File scan}) \]
\[ (\text{Index scan}) \]
(On the fly) \( (g) \Pi_{\text{name}} \)

(On the fly) \( (f) \sigma_{12<\text{age}<20} \)

(Block nested loop S inner)

(On the fly) \( (d) \Pi_{\text{sid}} \)

(Indexed-nested loop, B outer, C inner)

(b) \( \Pi_{\text{bid}} \)

(a) \( \sigma_{\text{author} = 'Olden Fames'} \)

(Book B (Index scan))

(On the fly) \( \text{Checkout C (File scan)} \)

(Student S (File scan))

(On the fly) \( \text{Total cost } = 1300 \)

(compare with 1,515,001,000 for plan 1!)

(Final cardinality = 234 (approx) (same as plan 1!))

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems 39
Task 4: Efficiently searching the plan space

Use dynamic-programming based Selinger’s algorithm
Heuristics for pruning plan space

- Predicates as early as possible
- Avoid plans with cross products
- Only left-deep join trees
Physical Plan Selection

Logical Query Plan

\[ P_1 \quad P_2 \quad \ldots \quad P_n \]

\[ C_1 \quad C_2 \quad \ldots \quad C_n \]

Pick minimum cost one

Physical plans

Costs
Join Trees

Query:  \( R1 \bowtie R2 \bowtie R3 \bowtie R4 \bowtie R5 \)

- Several possible structure of the trees
- Each tree can have \( n! \) permutations of relations

(logical plan space)

(physical plan space)

- Different implementation and scanning of intermediate operators for each logical plan
Selinger Algorithm

- Dynamic Programming based
- Dynamic Programming:
  - General algorithmic paradigm
  - Exploits “principle of optimality”
    - Useful reading: Chapter 16, Introduction to Algorithms, Cormen, Leiserson, Rivest
- Considers the search space of left-deep join trees
  - reduces search space (only one structure), still $n!$ permutations
  - interacts well with join algos (esp. NLJ)
  - e.g. might not need to write tuples to disk if enough memory
Principle of Optimality

Optimal for “whole” made up from optimal for “parts”
Principle of Optimality

Query: \( R1 \bowtie R2 \bowtie R3 \bowtie R4 \bowtie R5 \)

Suppose, this is an Optimal Plan for joining R1…R5:
Principle of Optimality

Query: $R1 \bowtie R2 \bowtie R3 \bowtie R4 \bowtie R5$

Suppose, this is an Optimal Plan for joining $R1...R5$: 
Principle of Optimality

Query: $R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5$

Then, what can you say about this sub-plan?

This has to be the optimal plan for joining $R_3, R_2, R_4, R_1$

Suppose, this is an Optimal Plan for joining $R_1...R_5$: 
 Principle of Optimality

Query: \( R1 \Join R2 \Join R3 \Join R4 \Join R5 \)

Then, what can you say about this sub-plan?

We are using the associativity and commutativity of joins:

\[
(R \Join S) \Join T = R \Join (S \Join T) \\
R \Join S = S \Join R
\]

Suppose, this is an Optimal Plan for joining R1…R5:

This has to be the optimal plan for joining \( R3, R2, R4 \)
Exploiting Principle of Optimality

Query: \( R1 \bowtie R2 \bowtie \ldots \bowtie Rn \)

Both are giving the same result
\( R2 \bowtie R3 \bowtie R1 = R3 \bowtie R1 \bowtie R2 \)

Optimal for joining \( R1, R2, R3 \)

Sub-Optimal for joining \( R1, R2, R3 \)
Exploiting Principle of Optimality

Suppose you chose the sub-optimal one

Leads to sub-Optimal for joining R1,…,Rn

A sub-optimal sub-plan cannot lead to an optimal plan
Notation

OPT ( \{ R1, R2, R3 \} ):

Cost of optimal plan to join \( R1, R2, R3 \)

T ( \{ R1, R2, R3 \} ):

Number of tuples in \( R1 \bowtie R2 \bowtie R3 \)
Selinger Algorithm:

Query: \( R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \)

e.g. All possible permutations of \( R_1, R_2, R_3 \)
have been considered
after \( \text{OPT} \{ R_1, R_2, R_3 \} \) has been computed

Progress of algorithm
Simple Cost Model

\[ \text{Cost} (R \bowtie S) = T(R) + T(S) \]

All other operators have 0 cost

Note: The simple cost model used for illustration only, it is not used in practice
Cost Model Example

\[
\text{Total Cost: } T(R) + T(S) + T(T) + T(X)
\]
Selinger Algorithm:

OPT ( { R1, R2, R3 } ):

Min

\[
\begin{align*}
\text{OPT ( } \{ R1, R2 \} ) & \quad + \quad T ( \{ R1, R2 \} ) + T(R3) \\
\text{OPT ( } \{ R2, R3 \} ) & \quad + \quad T ( \{ R2, R3 \} ) + T(R1) \\
\text{OPT ( } \{ R1, R3 \} ) & \quad + \quad T ( \{ R1, R3 \} ) + T(R2)
\end{align*}
\]

Note: Valid only for the simple cost model
Selinger Algorithm:

Query:  \( R1 \bowtie R2 \bowtie R3 \bowtie R4 \)

Progress of algorithm
Selinger Algorithm:

Query:  $R1 \Join R2 \Join R3 \Join R4$

Progress of algorithm

- $\{ R1, R2, R3, R4 \}$
- $\{ R1, R2, R3 \}$, $\{ R1, R2, R4 \}$, $\{ R1, R3, R4 \}$, $\{ R2, R3, R4 \}$
- $\{ R1, R2 \}$, $\{ R1, R3 \}$, $\{ R1, R4 \}$, $\{ R2, R3 \}$, $\{ R2, R4 \}$, $\{ R3, R4 \}$
- $\{ R1 \}$, $\{ R2 \}$, $\{ R3 \}$, $\{ R4 \}$
Q. How to optimally compute join of \{R1, R2, R3, R4\}?

Ans: First optimally join \{R1, R3, R4\} then join with R2 as inner.
Selinger Algorithm:

Query: \( R1 \bowtie R2 \bowtie R3 \bowtie R4 \)

Q. How to optimally compute join of \{R1, R3, R4\}?  
Ans: First optimally join \{R1, R3\}, then join with \( R4 \) as inner.
Selinger Algorithm:

Query:  $R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4$

Q. How to optimally compute join of $\{R_1, R_3\}$?

Ans: First optimally join $\{R_3\}$, then join with $R_1$ as inner.
Q. How to optimally compute join of \{R3\}?

Ans: Single relation – so optimally scan R3.
Selinger Algorithm:

Query: \( R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \)

Final optimal plan:

NOTE: There is a one-one correspondence between the permutation \((R_3, R_1, R_4, R_2)\) and the above left deep plan.
Selinger Algorithm:

Query: \( R1 \bowtie R2 \bowtie R3 \bowtie R4 \)

NOTE: (*VERY IMPORTANT*)
- This is *NOT* done by top-down recursive calls.
- This is done BOTTOM-UP computing the optimal cost of *all* nodes in this lattice only once (dynamic programming).

\{ \{ R1, R2, R3, R4 \} \\
\{ R1, R2, R3 \} \{ R1, R2, R4 \} \{ R1, R3, R4 \} \{ R2, R3, R4 \} \\
\{ R1, R2 \} \{ R1, R3 \} \{ R1, R4 \} \{ R2, R3 \} \{ R2, R4 \} \{ R3, R4 \} \\
\{ R1 \} \{ R2 \} \{ R3 \} \{ R4 \} \}

Progress of algorithm
More on Query Optimizations

• See the survey (on course website): “An Overview of Query Optimization in Relational Systems” by Surajit Chaudhuri

• Covers other aspects like
  – Pushing group by before joins
  – Merging views and nested queries
  – “Semi-join”-like techniques for multi-block queries
    • covered later in distributed databases
  – Statistics and optimizations
  – Starburst and Volcano/Cascade architecture, etc

• We may revisit Query Optimization again in our “Advanced Topics” later in the course.