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

```
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:**
  \[ \sigma_{c_1 \land \ldots \land c_n}(R) \equiv \sigma_{c_1}(\ldots \sigma_{c_n}(R)) \quad \text{(Cascade)} \]
  \[ \sigma_{c_1}(\sigma_{c_2}(R)) \equiv \sigma_{c_2}(\sigma_{c_1}(R)) \quad \text{(Commute)} \]

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

• **Joins:**
  \[ R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T \quad \text{(Associative)} \]
  \[ (R \bowtie S) \equiv (S \bowtie R) \quad \text{(Commute)} \]

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

Pick the plan with least cost

Challenge:

- Do not want to execute more than one plans
- 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
today

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: $T(R)$

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
\[ f = \frac{\text{Max}(R.A) - 50}{\text{Max}(R.A) - \text{Min}(R.A)} \]
assumes uniform distribution
Cost of Index Scan with Selection (and multiple conditions)

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

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

Size: \( T(R) \times f \) – if unclustered

Reduction factors

- \( f_1 = \frac{\text{Max}(R.A) - 50}{\text{Max}(R.A) - \text{Min}(R.A)} \)
- \( f_2 = \frac{T(R)}{V(R, B)} \)

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

range selection

value selection

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 \( R \)

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

\( R \)

\( \pi_A \)
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 \( T(R) \times T(S) \)

\[ R.A = S.B \]

\( T(R) \) : 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

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

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

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

T (R) : Number of tuples in R
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
\[ \text{Size} = T(R) \times T(S) \times f \]

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: \( 1/V(\ldots, \ldots) \)
if no index: say \( 1/10 \)

\( T(R) \) : 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

\[
\begin{align*}
\sigma &\ 12<\text{age}<20 \ \wedge \ \text{author} = \text{Olden Fames}' \\
\Pi &\ name \\
\text{(On the fly) (d) } \\
\text{(On the fly) (c) } \\
\text{(Tuple-based nested loop) } \\
B &\ inner
\end{align*}
\]

\[
\begin{align*}
\text{(Page-oriented -nested loop, } \\
S &\ outer, C \ inner)
\end{align*}
\]

\[
\begin{align*}
\text{Student S} \\
\text{(File scan)} \\
\text{Checkout C} \\
\text{(File scan)} \\
\text{Book B} \\
\text{(File scan)}
\end{align*}
\]

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems
S(sid, name, age, addr)  
B(bid, title, author)  
C(sid, bid, date)  

T(S) = 10,000  
T(B) = 50,000  
T(C) = 300,000  

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

V(B, author) = 500  

7 <= age <= 24  

Cost =  
B(S) + B(S) * B(C)  
= 1000 + 1000 * 15000  
= 15,001,000  

Cardinality =  
T(C) = 300,000  

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

T(S) * T(C)/max (V(S, sid), V(C, sid))  
= T(S)  
since V(S, sid) >= V(C, sid) and  
T(S) = V(S, sid)
### Student S
- `S(sid, name, age, addr)`
- `T(S) = 10,000`

### Book B
- `B(bid, title, author)`
- `T(B) = 50,000`
- `B(S) = 1,000`
- `B(B) = 5,000`
- `B(C) = 15,000`
- `V(B, author) = 500`

### Checkout C
- `C(sid, bid, date)`
- `T(C) = 300,000`

### Tuple-based nested loop
- `B` inner

### Page-oriented nested loop
- `S` outer, `C` inner

### Cost
\[
\text{Cost} = T(S \bowtie C) \times B(B) \\
= 300,000 \times 5,000 \\
= 15 \times 10^8
\]

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

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

---

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems
\( S(\text{sid}, \text{name}, \text{age}, \text{addr}) \quad T(S) = 10,000 \)
\( B(\text{bid}, \text{title}, \text{author}) \quad T(B) = 50,000 \)
\( C(\text{sid}, \text{bid}, \text{date}) \quad T(C) = 300,000 \)
\(\)
\(\)
\( B(\text{sid}) = 1,000 \quad B(\text{bid}) = 5,000 \quad B(\text{C}) = 15,000 \quad \)
\( V(\text{B, author}) = 500 \quad 7 \leq \text{age} \leq 24 \)

\[
\begin{align*}
  \text{(c, d)} \\
  (\text{On the fly}) \quad (d) \quad \Pi_{\text{name}} \\
  (\text{On the fly}) \quad (c) \quad \sigma_{12 < \text{age} < 20 \land \text{author} = \text{‘Olden Fames’}} \\
  (\text{Tuple-based nested loop}) \quad \text{B inner} \\
  (\text{Page-oriented nested loop, S outer, C inner}) \\
  \text{Student S (File scan)} \\
  \text{Checkout C (File scan)} \\
  \text{Book B (File scan)} \\
\end{align*}
\]

\[
\begin{align*}
  \text{Cost} &= 0 \quad \text{(on the fly)} \\
  \text{Cardinality} &= 300,000 \times \frac{1}{500} \times \frac{7}{18} \\
  &= 234 \text{ (approx)} \quad \text{(assuming uniformity and independence)}
\end{align*}
\]
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

Student S
Checkout C
Book B

(a) \( \sigma_{\text{author}} = '\text{Olden Fames}' \)
(b) \( \Pi_{\text{bid}} \)
(c) \( \Pi_{\text{sid}} \)
(d) \( \Pi_{\text{sid}} \) (On the fly)
(e) \( \sigma_{12<\text{age}<20} \)
(f) \( \Pi_{\text{name}} \) (On the fly)
(g) \( \Pi_{\text{name}} \)

Book B (Index scan)
Checkout C (Index scan)
Student S (File scan)
$S(\text{sid}, \text{name}, \text{age}, \text{addr})$

$B(\text{bid}, \text{title}, \text{author})$: Un. B+ on author

$C(\text{sid}, \text{bid}, \text{date})$: Cl. B+ on bid

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

$T(B) = 50,000$  \quad $B(B) = 5,000$  \quad $7 \leq \text{age} \leq 24$

$T(C) = 300,000$  \quad $B(C) = 15,000$

**Cost =**

\[
\frac{T(B)}{V(B, \text{author})} = \frac{50,000}{500} = 100 \text{ (unclustered)}
\]

**Cardinality =**

100

---

**Student S**

File scan

---

**Checkout C**

Index scan

---

**Book B**

Index scan
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

Book B (Index scan)

Student S (File scan)

Checkout C

V(B,author) = 500

Cost = 0 (on the fly)

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

- one index lookup per outer B tuple
- 1 book has $T(C)/T(B) = 6$ checkouts (uniformity)
- # C tuples per page = $T(C)/B(C) = 20$
- 6 tuples fit in at most 2 consecutive pages (clustered) could assume 1 page as well

Cost <= 100 * 2 = 200

Cardinality = 100 * 6 = 600

= 100 * $T(C)/\text{MAX}(100, V(C, bid))$
assuming
$V(C, bid) = V(B, bid) = T(B) = 50,000$
\[ \text{S(sid, name, age, addr)} \quad T(S) = 10,000 \]
\[ \text{B(bid, title, author): Un. B+ on author} \quad T(B) = 50,000 \]
\[ \text{V(B, author) = 500} \]
\[ \text{C(sid, bid, date): Cl. B+ on bid} \quad T(C) = 300,000 \]
\[ 7 <= \text{age} <= 24 \]

\[ \begin{align*}
\text{B(S)} &= 1,000 \\
\text{B(B)} &= 5,000 \\
\text{B(C)} &= 15,000 \\
\end{align*} \]
Student S  
Checkout C  
Book B

(a) $\sigma_{\text{author} = 'Olden Fames'}$
(b) $\Pi_{\text{bid}}$
(c) $\Pi_{\text{sid}}$
(d) $\Pi_{\text{sid}}$
(On the fly)
(On the fly)
(On the fly)
(On the fly)

(Block nested loop  
S inner)
(Opened nested loop,  
B outer, C inner)

(e) $\sigma_{12 \leq \text{age} \leq 20}$
(g) $\Pi_{\text{name}}$

(On the fly)

(d) $\Pi_{\text{bid}}$

File scan

Student S  
Checkout C  
Book B

(On the fly)

V(B, author) = 500

7 $\leq$ age $\leq$ 24

T(S) = 10,000
B(S) = 1,000

T(B) = 50,000
B(B) = 5,000

T(C) = 300,000
B(C) = 15,000

Outer relation is already in (unlimited) memory need to scan S relation

Cost =
B(S) = 1000

Cardinality =
600
\begin{align*}
S(sid, name, age, addr) & \quad T(S) = 10,000 \\
B(bid, title, author): Un. B+ on author & \quad B(S) = 1,000 \\
C(sid, bid, date): Cl. B+ on bid & \quad V(B, author) = 500 \\
& \quad 7 \leq age \leq 24 \\
\end{align*}

\begin{align*}
B(bid, title, author): Un. B+ on author & \quad T(B) = 50,000 \\
B(bid, title, author): Un. B+ on author & \quad B(B) = 5,000 \\
C(sid, bid, date): Cl. B+ on bid & \quad T(C) = 300,000 \\
C(sid, bid, date): Cl. B+ on bid & \quad B(C) = 15,000 \\
\end{align*}

\begin{align*}
& \quad (f) \ \sigma_{12 < age < 20} \\
& \quad (e) \ \Pi_{name} \\
& \quad (d) \ \Pi_{sid} \\
& \quad (c) \ \text{Student S (File scan)} \\
& \quad (b) \ \Pi_{bid} \\
& \quad (a) \ \sigma_{author = 'Olden Fames'} \\
& \quad \text{Checkout C (Index scan)} \\
& \quad \text{Book B (Index scan)} \\
& \quad \text{Cost = 0 (on the fly)} \\
& \quad \text{Cardinality = 600 * 7/18 = 234 (approx)} \\
\end{align*}
\[ S(\text{sid}, \text{name}, \text{age}, \text{addr}) \]

\[ B(\text{bid}, \text{title}, \text{author}) : \text{Un. B+ on author} \]

\[ C(\text{sid}, \text{bid}, \text{date}) : \text{Cl. B+ on bid} \]

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

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

\[ T(B) = 50,000 \quad B(B) = 5,000 \]

\[ T(C) = 300,000 \quad B(C) = 15,000 \]

\( \text{Cost} = 0 \) (on the fly)

\( \text{Cardinality} = 234 \)
S(sid, name, age, addr)  T(S) = 10,000  B(S) = 1,000
B(bid, title, author): Un. B+ on author  T(B) = 50,000  B(B) = 5,000
C(sid, bid, date): Cl. B+ on bid  T(C) = 300,000  B(C) = 15,000

V(B, author) = 500
7 ≤ age ≤ 24

(Book nested loop, S inner)

(Student S (File scan))

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

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

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

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

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

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

(Book B (Index scan))

(On the fly) (c) Student S (File scan)

Total cost = 1300
(compare with 1,515,001,000 for plan 1!)

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

P1  P2  ....  Pn

C1  C2  ....  Cn

Pick minimum cost one
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)

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

(physical plan space)

left-deep join tree

bushy join tree
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: \[ R1 \Join R2 \Join R3 \Join R4 \Join R5 \]

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

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

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

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

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

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

This has to be the optimal plan for joining \( R3, R2, R4, R1 \)
Principle of Optimality

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

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

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

We are using the associativity and commutativity of joins:

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

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

Duke CS, Fall 2016

CompSci 516: Data Intensive Computing Systems
Exploiting Principle of Optimality

Query: \( R_1 \Join R_2 \Join \ldots \Join R_n \)

Both are giving the same result
\( R_2 \Join R_3 \Join R_1 = R_3 \Join R_1 \Join R_2 \)

Optimal for joining \( R_1, R_2, R_3 \)

Sub-Optimal for joining \( R_1, R_2, R_3 \)
Exploiting Principle of Optimality

Suppose you chose the sub-optimal one

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

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

\[
\text{OPT ( \{ R_1, R_2, R_3 \} ):}
\]

Cost of optimal plan to join \( R_1, R_2, R_3 \)

\[
\text{T ( \{ R_1, R_2, R_3 \} ):}
\]

Number of tuples in \( R_1 \bowtie R_2 \bowtie R_3 \)
Selinger Algorithm:

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

e.g. All possible permutations of \( R1, R2, R3 \) have been considered after \( \text{OPT}\{R1, R2, R3\} \) has been computed

Progress of algorithm
Simple Cost Model

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

Total Cost: $T(R) + T(S) + T(T) + T(X)$
Selinger Algorithm:

\[ \text{OPT ( \{ R1, R2, R3 \} )} : \]

\[ \min \]

\[ \text{OPT ( \{ R1, R2 \} ) + T ( \{ R1, R2 \} ) + T(R3) } \]

\[ \text{OPT ( \{ R2, R3 \} ) + T ( \{ R2, R3 \} ) + T(R1) } \]

\[ \text{OPT ( \{ R1, R3 \} ) + T ( \{ R1, R3 \} ) + T(R2) } \]

\textbf{Note: Valid only for the simple cost model}
Selinger Algorithm:

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

Progress of algorithm
Selinger Algorithm:

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

Progress of algorithm
Selinger Algorithm:

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

Q. How to optimally compute join of \{R1, R2, R3, R4\}?  
Ans: First optimally join \{R1, R3, R4\} then join with R2 as inner.

Progress of algorithm
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: \( R1 \bowtie R2 \bowtie R3 \bowtie R4 \)

Q. How to optimally compute join of \( \{R1, R3\}\)?

Ans: First optimally join \( \{R3\}\), then join with \( R1 \) as inner.

Progress of algorithm
Selinger Algorithm:

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

Q. How to optimally compute join of \{R3\}?

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

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

Final optimal plan:

NOTE: There is a one-one correspondence between the permutation (R3, R1, R4, R2) 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).

Progress of algorithm

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

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

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

\( \{ R1 \} \)  \( \{ R2 \} \)  \( \{ R3 \} \)  \( \{ R4 \} \)