CompSci 516
Data Intensive Computing Systems

Lecture 15
Query Optimization

Instructor: Sudeepa Roy
Announcements (Thurs, 10/17)

• Midterm next week 10/24 (Thursday) in class!
  – Everything until and including 10/22 is included

• HW2
  – Part 1 due next Monday 10/21
  – Part-2 deadline extended to Thursday 10/31

• Midterm project report extended to Monday 11/4
  – Submit 1 report per group on Sakai + attach to your private group thread on Piazza
  – Work on your projects!

• If you have questions on grade, send an email to compsci516-staff@cs.duke.edu
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:
• The following slides have been created adapting the instructor material of the [RG] book provided by the authors Dr. Ramakrishnan and Dr. Gehrke.
• Some of the following slides have been created by adapting slides by Profs. Shivnath Babu and Magda Balazinska
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
  - intermediate results written to disk
Relational Algebra Equivalences

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

❖ **Selections:** \( \sigma_{c_1 \land \ldots \land c_n}(R) \equiv \sigma_{c_1}(\ldots \sigma_{c_n}(R)) \) (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 if interested.
Notation

• $T(R)$ : Number of tuples in $R$
• $B(R)$ : Number of blocks (pages) 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 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
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
  - 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:
1. size is independent of the implementation of the scan/index
2. Index scan is bad if unclustered
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)

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

Index Scan

\[ f1 = (\text{Max}(R.A) - 50) / (\text{Max}(R.A) - \text{Min}(R.A)) \]

\[ f2 = 1 / V(R, A) \]

\[ f = f1 \times f2 \] (assumes independence and uniform distribution)

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

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

What is \( f1 \) if the first condition is \( 100 > R.1 > 50 \)?

Reduction factors

\( B(R) : \) Number of blocks in \( R \)

\( T(R) : \) Number of tuples in \( R \)

\( V(R, A) : \) Number of distinct values of attribute \( A \) in \( R \)
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
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 \(T(R) \times T(S)\)

\[R \bowtie S\]

\(T(R)\) : 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 standard 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:
   • For all “non-joining” attributes, the set of distinct values is preserved in join
     • \( V(R \bowtie S, C) = V(R, C) \), where \( C \neq A \) is an attribute in \( R \)
     • \( V(R \bowtie S, D) = V(S, D) \), where \( D \neq B \) is an attribute in \( S \)
   • Helps estimate distinct set size in \( R \bowtie S \bowtie T \)
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
\[ \text{Size} = T(R) \times T(S) \times f \]

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 \( 1/10 \)
Announcements (Tues, 10/22)

• Sudeepa’s office hour today moved to tomorrow 12-1 pm and 4:30-5 pm in LSRC D325

• Midterm on Thursday 10/24 in class!
  – Everything up to today’s lecture is included

• HW2
  – Part-1 due tonight
  – Part-2 due next Thursday 10/31

• Midterm project report due Monday 11/4
Review: Cost-based Query Optimization

Pick the plan with least cost

Tasks:
1. Estimate the cost of individual operators  
   done
2. Estimate the size of output of individual operators  
   done
3. Combine costs of different operators in a plan  
   today
4. Efficiently search the space of plans  
   today
Review: Cost Estimation

• For any operator in the query plan, need to estimate both
  – Size = no. of output tuples
  – Cost = no. of pages I/O from disk

• We assume uniformity and independence

Q1. $\sigma_{R.A > 50 \text{ and } R.B = 25} R$

Suppose range of R.A is [10, 100], R.B has 50 distinct values, and R has 900 tuples. What is the size estimate of the output?

Q2. $R(A, B) \bowtie S(B, C)$. S has 100 distinct values of B and 500 tuples. What is the size estimate of the output?
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: $C$  
  On disk initially

- 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 (given):
• Data is not sorted on any attributes
• For (b), outer relation fit in memory

(Tuple-based nested loop
B inner)

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

(On the fly) (c) $\sigma_{7 \leq \text{age} \leq 24 \land \text{author} = \text{\textquotesingle}Olden Fames\text{\textquotesingle}}$

(Student S (File scan) Checkout C (File scan) Book B (File scan))

B(S)=1,000
B(B)=5,000
B(C)=15,000
V(B,author) = 500
7 <= age <= 24
\begin{align*}
S &= \{ \text{sid, name, age, addr} \} \\
B &= \{ \text{bid, title, author} \} \\
C &= \{ \text{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, \text{author}) &= 500 \\
7 \leq \text{age} \leq 24 \\
\text{Cost} &= B(S) + B(S) \times B(C) \\
&= 1000 + 1000 \times 15000 \\
&= 15,001,000 \\
\text{Cardinality} &= T(C) = 300,000 \\
&\text{foreign key join, output pipelined to next join} \\
&\text{Can apply the “formula” as well} \\
&\text{since } V(S, \text{sid}) \geq V(C, \text{sid}) \\
&\text{and } T(S) = V(S, \text{sid}) \\
\end{align*}
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

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

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

- foreign key join
- don’t need scanning for outer relation
  - outer relation fits in memory

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

(Tuple-based nested loop
B inner)

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

Student S
(File scan)

Checkout C
(File scan)
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

(c, d)

(On the fly) (d) Π name
(On the fly) (c) σ 12 < age < 20 ∧ author = 'Olden Fames'
(Tuple-based nested loop B inner)

(Book B (File scan))
(On the fly) (b)
(Page-oriented nested loop, S outer, C inner)

Student S (File scan)
Checkout C (File scan)

Cost = 0 (on the fly)
Cardinality = 300,000 * 1/500 * 7/18
= 234 (approx)
(assuming uniformity and independence)
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
C(sid, bid, date)      T(C) = 300,000  B(C) = 15,000

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 (given):
- Unclustered B+tree index on B.author
- Clustered B+tree index on C.bid
- All index pages are in memory
- Unlimited memory

Duke CS, Fall 2019

CompSci 516: Database Systems
S(sid, name, age, addr)  
Student S  
T(S) = 10,000  
B(S) = 1,000  
V(B, author) = 500 
7 <= age <= 24

B(bid, title, author): Un. B+ on author  
Book B  
T(B) = 50,000  
B(B) = 5,000

C(sid, bid, date): Cl. B+ on bid  
Checkout C  
T(C) = 300,000  
B(C) = 15,000

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

Block nested loop S inner
(On the fly)  (b) \Pi_{\text{bid}}
(On the fly)  (d) \Pi_{\text{sid}}
(Indexed-nested loop, B outer, C inner)
(On the fly)  (f) \sigma_{12<\text{age}<20}
(On the fly)  (g) \Pi_{\text{name}}

(Student S File scan)

Cost = 
T(B) / V(B, 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  7 <= age <= 24
T(C)=300,000  B(C)=15,000

(b) Cost = 0 (on the fly)
Cardinality = 100

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

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

(Block nested loop)  (e) S inner

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

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

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

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

Check out C
(Index scan)

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

Book B
(Index scan)
$S(\text{sid, name, age, addr})$  \hspace{1cm} $T(S)=10,000$  \hspace{1cm} $B(S)=1,000$ \hspace{1cm} $V(B, \text{author}) = 500$

$B(\text{bid, title, author})$: Un. B+ on author  \hspace{1cm} $T(B)=50,000$  \hspace{1cm} $B(B)=5,000$

$C(\text{sid, bid, date})$: Cl. B+ on bid  \hspace{1cm} $T(C)=300,000$  \hspace{1cm} $B(C)=15,000$

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

Cost $\leq$

$100 \times 2 = 200$

Cardinality $=$

$100 \times 6 = 600$
\begin{align*}
S(\text{sid}, \text{name}, \text{age}, \text{addr}) & \quad T(S) = 10,000 \\
B(\text{bid}, \text{title}, \text{author}): \text{Un. B+ on author} & \quad B(S) = 1,000 \\
C(\text{sid}, \text{bid}, \text{date}): \text{Cl. B+ on bid} & \quad T(B) = 50,000 \\
& \quad B(B) = 5,000 \\
& \quad T(C) = 300,000 \\
& \quad B(C) = 15,000 \\
\end{align*}

\begin{align*}
V(\text{B}, \text{author}) & = 500 \\
7 \leq \text{age} \leq 24
\end{align*}
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

Student S
(On the fly) (g) \( \Pi_{name} \)
(On the fly) (f) \( \sigma_{12<age<20} \)
(Block nested loop S inner)

Checkout C
(On the fly) (b) \( \Pi_{bid} \)

Book B
(On the fly) (a) \( \sigma_{author = 'Olden Fames'} \)
(Indexed-nested loop, B outer, C inner)

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

Cost = B(S) = 1000
Cardinality = 600
(one student per checkout)
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  

7 <= age <= 24

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

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

Block nested loop  
S inner

Indexed-nested loop,  
B outer, C inner

Student S  
(File scan)

Checkout C  
(Index scan)

Book B  
(Index scan)

Cost =  
0 (on the fly)

Cardinality =  
600 * 7/18 = 234 (approx)
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

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

Cost = 0 (on the fly)
Cardinality = 234

Duke CS, Fall 2019  CompSci 516: Database Systems  40
\begin{align*}
S(\text{sid},\text{name},\text{age},\text{addr}) & \quad T(S)=10,000 \\
B(\text{bid},\text{title},\text{author}) & \text{ Un. B+ on author } \\
C(\text{sid},\text{bid},\text{date}) & \text{ Cl. B+ on bid } \\
V(\text{B},\text{author}) & = 500 \\
7 \leq \text{age} \leq 24 & \\
\end{align*}

\textbf{(total)}

\begin{align*}
\text{(Block nested loop S inner)} \\
\text{(Indexed-nested loop, B outer, C inner)} \\
\text{(On the fly)(a) } \sigma_{\text{author} = \text{‘Olden Fames’}} \\
\text{(On the fly)(b) } \Pi_{\text{bid}} \\
\text{(On the fly)(On the fly)(c) Student S (File scan)} \\
\text{(On the fly)(g) } \Pi_{\text{name}} \\
\text{(On the fly)(f) } \sigma_{12 < \text{age} < 20} \\
\text{(On the fly)(e) } \Pi_{\text{sid}} \\
\text{Checkout C (Index scan)} \\
\text{Book B (Index scan)} \\
\end{align*}

\textbf{Total cost} = 1300 \\
\text{(compare with 1,515,001,000 for plan 1!)}

\textbf{Final cardinality} = 234 (approx) \\
\text{(same as plan 1!)}
Task 4: Efficiently searching the plan space

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

• Apply predicates as early as possible
• Avoid plans with cross products
• Consider only left-deep join trees
Join Trees

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

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

(physical plan space)
- Different implementation and scanning of intermediate operators for each logical plan

Why?

left-deep join tree

bushy join tree

(logical plan space)
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)
  – but still n! permutations
  – interacts well with join algs (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 \( R1 \ldots R5 \):
This has to be the optimal plan for joining $R3, R2, R4, R1$
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 \(R3, R2, R4\)
Exploiting Principle of Optimality

Query: $R1 \Join R2 \Join \ldots \Join Rn$

Both are giving the same result
$R2 \Join R3 \Join R1 = R3 \Join R1 \Join R2$

Optimal for joining $R1, R2, R3$

Sub-Optimal for joining $R1, R2, R3$

Suppose you chose the sub-optimal one

Leads to sub-Optimal for joining $R1, \ldots, Rn$
Notation

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

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

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

Number of tuples in \( R1 \Join R2 \Join R3 \)
Simple Cost Model

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

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:

OPT ( { R1, R2, R3 } ):

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

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

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

Progress of algorithm
Selinger Algorithm:

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

Progress of algorithm

Selinger Algorithm:

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

Progress of algorithm
Selinger Algorithm:

Query: \quad 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.
Selinger Algorithm:

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

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.
Selinger Algorithm:

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

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

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

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

e.g. All possible permutations of $R_1, R_3, R_4$ have been considered after $OPT\{R_1, R_3, R_4\}$ has been computed

Progress of algorithm
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:**
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).

Is it efficient? 😊
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
  – Starbust and Volcano/Cascade architecture, etc