CompSci 516
Database Systems

Lecture 22
Query Optimization

Instructor: Sudeepa Roy
Where are we now?

Relational model, queries, db design
- Relational Model
- Normal Forms, FD
- Query in SQL / RA / RC
- Recursion

DBMS Internals and Query Processing
- Storage
- Index
- Join algo/Sorting
- Execution/Optimization

Beyond Relational Model
- XML
- NOSQL
  - JSON/MongoDB

Transactions
- Basics
- Concurrency Control
- Recovery

(Basic) Big Data Processing
- Map-Reduce/Spark
- Parallel DBMS
- Distributed DBMS

Other Topics
- Data Mining
- Data Cube

Covered
- To be covered

Duke CS, Spring 2022
CompSci 516: Database Systems
Announcements (Tues, 03/29)

• HW3 due 4/5 (Tues) noon
  – Let us know if you need someone to work with
• More frequent check in for all teams by mentors
• Project report deadline 04/13
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 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
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

\[ T(R) : \text{Number of tuples in } R \]
\[ B(R) : \text{Number of blocks in } R \]
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
\( T(R) \times f \) – if unclustered

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

Reduction factors

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

value selection
\[ f_2 = \frac{1}{V(R, B)} \]

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

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

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

Value selection

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

$R.A = S.B$

$T(R)$: 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 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 = \( T(R) \times T(S) \times f \)

\( R.A = S.B \)

\( R \)
\( S \)

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

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: \( 1/V(\ldots, \ldots) \)
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**: $C$  
  On disk initially

- $S(sid, name, age, addr)$
- $B(bid, title, author)$
- $C(sid, bid, date)$

- 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) \( \sigma_{12<age<20 \land author = 'Olden Fames'} \)
(Total-oriented -nested loop, S outer, C inner)

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

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

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

\[V(B, \text{author}) = 500\]

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

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

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

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

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

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

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

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

(On the fly) (b)

(Pe-page-oriented
-nested loop,
S outer, C inner)

Book B
(File scan)

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

\( \text{(Total)} \)

\( (\text{On the fly}) \quad (d) \quad \prod_{\text{name}} \)

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

\( \text{(Tuple-based nested loop} \quad \text{B inner)} \)

\( \text{(Page-oriented} \quad -\text{nested loop,} \quad \text{S outer, C inner)} \)

\( \text{Student S} \quad \text{(File scan)} \quad \text{Checkout C} \quad \text{(File scan)} \quad \text{Book B} \quad \text{(File scan)} \)

\( \text{Total cost} = 1,515,001,000 \)

\( \text{Final cardinality} = 234 \text{ (approx)} \)
Physical Query Plan – 2

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

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

**Student S**
- Index scan

**Checkout C**
- Index scan

**Book B**
- Index scan

**Dataset**
- S(sid, name, age, addr)
- B(bid, title, author)
- C(sid, bid, date)

**Table Statistics**
- \( T(S) = 10,000 \)
- \( T(B) = 50,000 \)
- \( T(C) = 300,000 \)

**Table Sizes**
- \( B(S) = 1,000 \)
- \( B(B) = 5,000 \)
- \( B(C) = 15,000 \)

**Variable Costs**
- \( V(S, \text{author}) = 500 \)
- \( 7 \leq \text{age} \leq 24 \)

**Queries**
(a) \( \sigma_{\text{author} = \text{"Olden Fames"}} \)
(b) \( \Pi_{\text{bid}} \)
(c) \( \Pi_{\text{sid}} \) (On the fly)
(d) \( \Pi_{\text{sid}} \) (On the fly)
(e) (Block nested loop, S inner)
(f) \( \sigma_{12<\text{age}<20} \)
(g) \( \Pi_{\text{name}} \) (On the fly)
Student S
Bid
Checkout C

<table>
<thead>
<tr>
<th>T(S)</th>
<th>B(S)</th>
<th>V(B, author)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>1,000</td>
<td>500</td>
</tr>
<tr>
<td>50,000</td>
<td>5,000</td>
<td></td>
</tr>
<tr>
<td>300,000</td>
<td>15,000</td>
<td></td>
</tr>
</tbody>
</table>

7 <= age <= 24

Cost =
T(B) \div 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

||| (a) σ author = 'Olden Fames'
||| (b) π bid
||| (c) Student S (File scan)
||| (d) π sid (On the fly)
||| (e) (Block nested loop S inner)
||| (f) σ 12<age<20
||| (g) π name

||| (b) π bid
||| (c) Checkout C (Index scan)
||| (d) π sid (On the fly)
||| (e) (Indexed-nested loop, B outer, C inner)

Cost = 0 (on the fly)
Cardinality = 100

Duke CS, Spring 2022
CompSci 516: Database Systems
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

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

Cost <=
100 * 2 = 200

Cardinality =
100 * 6 = 600
**Student S**

**Checkout C**

\( f \)  \[ 7 \leq \text{age} \leq 24 \]

\( \sigma_{\text{age} = \text{'Olden Fames'}} \)

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

\( \Pi_{\text{sid}} \)

\( \prod_{\text{name}} \)

\( \sigma_{12 < \text{age} < 20} \)

\( B(S) = 1,000 \)

\( B(B) = 5,000 \)

\( B(C) = 15,000 \)

\( T(S) = 10,000 \)

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

\( T(C) = 300,000 \)

\( V(B, \text{author}) = 500 \)

**Cost** = 0 (on the fly)

**Cardinality** = 600

Duke CS, Spring 2022

CompSci 516: Database Systems
Student S
Checkout C
Book B

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

\( T(S) = 10,000 \)
\( B(S) = 1,000 \)
\( V(B, \text{author}) = 500 \)
\( 7 \leq \text{age} \leq 24 \)

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

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

\( B(S) = 1,000 \)

\( B(S) = 1000 \)

Cardinality = 600
(one student per checkout)

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

Cost = 77

Student S
Checkout C
Book B

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

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

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

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

Block nested loop S inner

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

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

(On the fly)  (e)
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
T(B) = 50,000, B(B) = 5,000
T(C) = 300,000, B(C) = 15,000

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

Cost = 0 (on the fly)
Cardinality = 600 * 7/18 = 234 (approx)
\begin{itemize}
  \item S(sid, name, age, addr)
  \item B(bid, title, author): Un. B+ on author
  \item C(sid, bid, date): Cl. B+ on bid
\end{itemize}

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

\textbf{Cardinality = 234}

T(S) = 10,000 \quad B(S) = 1,000 \quad V(B, author) = 500

B(B) = 5,000

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

7 \leq \text{age} \leq 24

\begin{tikzpicture}
  \node (s) at (0,0) {Student S (File scan)};
  \node (b) [below of=s] {Book B (Index scan)};
  \node (c) [below of=b] {Checkout C (Index scan)};
  \node (p) [above of=s, yshift=-2cm] {Student S}
      child {node (a) {\(\sigma_{\text{author} = 'Olden Fames'}\)}
        child {node (b) {\(\Pi_{\text{bid}}\)}
          child {node (c) {\(\Pi_{\text{sid}}\)}
            child {node (d) {\(\Pi_{\text{sid}}\)}
              child {node (e) {\(\Pi_{\text{name}}\)}
                child {node (f) {\(\sigma_{12<\text{age}<20}\)}
                  child {node (g) {\(\prod_{\text{name}}\)}}}}}}}};
\end{tikzpicture}
\[ \begin{align*}
S(\text{sid}, \text{name}, \text{age}, \text{addr}) \\
B(\text{bid}, \text{title}, \text{author}) &: \text{Un. B+ on author} \\
C(\text{sid}, \text{bid}, \text{date}) &: \text{Cl. B+ on bid}
\end{align*} \]

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

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

\[ \begin{align*}
T(S) &= 10,000 \\
B(S) &= 1,000 \\
V(B, \text{author}) &= 500 \\
7 &\leq \text{age} &\leq 24
\end{align*} \]

\[ \begin{align*}
T(B) &= 50,000 \\
B(B) &= 5,000
\end{align*} \]

\[ \begin{align*}
T(C) &= 300,000 \\
B(C) &= 15,000
\end{align*} \]
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:  \( R1 \Join R2 \Join R3 \Join R4 \Join R5 \)

• Several possible structure 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

(left-deep join tree)

Why?

(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)
  – but 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 \( R1 \ldots R5 \):
Principle of Optimality

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

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

Query: \( R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5 \)

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 \(R_3, R_2, R_4\)
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 \( R1, R2, R3 \)

Suppose you chose the sub-optimal one
Leads to sub-Optimal for joining \( R1, \ldots, Rn \)
Notation

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

Cost of optimal plan to join $R1,R2,R3$

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

Number of tuples in $R1 \bowtie R2 \bowtie R3$
Simple Cost Model

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

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 }):

\[
\text{Min} \begin{cases} 
\text{OPT ( } \{ R1, R2 \} \text{ ) } + T( \{ R1, R2 \} ) + T( R3 ) \\
\text{OPT ( } \{ R2, R3 \} \text{ ) } + T( \{ R2, R3 \} ) + T( R1 ) \\
\text{OPT ( } \{ R1, R3 \} \text{ ) } + T( \{ R1, R3 \} ) + T( R2 )
\end{cases}
\]

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

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

Progress of algorithm
Selinger Algorithm:

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

Suppose this path is chosen by the algorithm. How to translate to a query plan?
Selinger Algorithm:

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

Q. How to optimally compute join of \( \{R_1, R_2, R_3, R_4\} \)?

Ans: First optimally join \( \{R_1, R_3, R_4\} \) then join with \( R_2 \) as inner.

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

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

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

Progress of algorithm

\[
\begin{align*}
\{R1\} & \quad \{R2\} & \quad \{R3\} & \quad \{R4\} \\
\{R1, R2\} & \quad \{R1, R3\} & \quad \{R1, R4\} & \quad \{R2, R3\} & \quad \{R2, R4\} & \quad \{R3, R4\} \\
\{R1, R2, R3\} & \quad \{R1, R2, R4\} & \quad \{R1, R3, R4\} & \quad \{R2, R3, R4\} \\
\{R1, R2, R3, R4\} &
\end{align*}
\]
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: $R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4$

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? 😊

Reduces $n!$ to $2^n$

Other optimizations employed too..
More on Query Optimizations

• See the survey:
  “An Overview of Query Optimization in Relational Systems” by Surajit Chaudhuri (link)
  – Pushing group by before joins
  – Merging views and nested queries
  – “Semi-join”-like techniques for multi-block queries
    • Recall joins in distributed databases
  – Statistics and optimizations
  – Starburst and Volcano/Cascade architecture, etc

• New research topics: Robust query optimization, “learned” query optimization, approximate selectivity estimation...