CompSci 516
Data Intensive Computing Systems

Lecture 12
Query Optimization

Instructor: Sudeepa Roy
Announcements

• Reminder: HW2 due on Oct 31
  – if you have not started yet, now is the time!
  – guest lecture by Prajakta Kalmegh on Thursday – more on Spark and big data systems
• Work on your projects too
• Midterm viewing at the end of the class
  – Remember to give me the exam back (no exam, no grade)
  – Feel free to take photos
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
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:** \( \sigma_{c_1 \wedge ... \wedge c_n}(R) \equiv \sigma_{c_1}(\ldots \sigma_{c_n}(R)) \) (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

Next lecture: cost-based optimization and Selinger’s algorithm
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 in Lecture 9-11
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

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:
1. size is independent of the implementation of the scan/index
2. Index scan is bad if unclustered

\[ T(R) : \text{Number of tuples in } R \]
\[ B(R) : \text{Number of blocks in } R \]
Cost of Index Scan with Selection

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

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

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

Reduction factor
\[ f = \frac{(\text{Max}(R.A) - 50)}{(\text{Max}(R.A) - \text{Min}(R.A))} \]
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)

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

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

\[ T(R) \times f \] – if unclustered

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

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

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)

\[ R \]

Index Scan

\[ \text{X} \]

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

\[ \text{T}(R) \]: Number of tuples in \( R \)

\[ \text{B}(R) \]: 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 $T(R) \times T(S)$

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

Why max?
- Suppose \( V(R, A) \leq V(S, B) \)
- The probability of a \( A \)-value joining with a \( B \)-value is \( \frac{1}{V(S,B)} = \) 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

\( R.A = S.B \)

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

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

Physical Query Plan

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

### Tuple-based nested loop
- B inner

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

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

\( \pi_{\text{name}} \)

\( \sigma \text{ } 12 < \text{age} < 20 \land \text{author} = \text{‘Olden Fames’} \)

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

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

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

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

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

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

\( B(S) = 1,000 \)
\( B(B) = 5,000 \)
\( B(C) = 15,000 \)
\( V(B, \text{author}) = 500 \)

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

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

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
• outer relation fits in memory
\( S(\text{sid}, \text{name}, \text{age}, \text{addr}) \quad T(S) = 10,000 \quad B(S) = 1,000 \quad V(B, \text{author}) = 500 \)

\( B(\text{bid}, \text{title}, \text{author}) \quad T(B) = 50,000 \quad B(B) = 5,000 \)

\( C(\text{sid}, \text{bid}, \text{date}) \quad T(C) = 300,000 \quad B(C) = 15,000 \)

\( 7 \leq \text{age} \leq 24 \)

\( (c, d) \)

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

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

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

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

\( \text{Book B (File scan)} \)

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

\( \text{Checkout C (File scan)} \)

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

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

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

\[
\text{Student S} & \quad \text{Checkout C} \\
(\text{File scan}) & \quad (\text{File scan}) \\
\]

\(\text{(Total)}\)

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

\[
\text{Book B} \quad \text{(File scan)} \\
\]

\[
\text{Total cost} = 1,515,001,000 \\
\text{Final cardinality} = 234 \text{ (approx)}
\]
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

Student S
Checkout C
Book B

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

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

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
S(sid, name, age, addr)  T(S) = 10,000  V(B, author) = 500
B(bid, title, author): Un. B+ on author  T(B) = 50,000
C(sid, bid, date): Cl. B+ on bid  T(C) = 300,000

7 \leq \text{age} \leq 24

\begin{align*}
\text{Cost} &= \frac{T(B)}{V(B, \text{author})} \\
&= \frac{50,000}{500} \\
&= 100 \text{ (unclustered)}
\end{align*}

\text{Cardinality} = 100
$S(sid, \text{name, age, addr})$
$B(bid, \text{title, author})$: Un. B+ on author
$C(sid, bid, \text{date})$: 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 \leq \text{age} \leq 24

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

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

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

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

\(\text{Cardinality} = 100\)
\[ S(\text{sid}, \text{name}, \text{age}, \text{addr}) \quad \text{T}(S) = 10,000 \]
\[ B(\text{bid}, \text{title}, \text{author}): \text{Un. B+ on author} \quad \text{B}(S) = 1,000 \]
\[ C(\text{sid}, \text{bid}, \text{date}): \text{Cl. B+ on bid} \quad \text{T}(B) = 50,000 \]
\[ \text{B}(B) = 5,000 \quad \text{T}(C) = 300,000 \quad \text{B}(C) = 15,000 \]

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

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

\[ \text{V}(\text{C}, \text{bid}) = \frac{\text{T}(\text{C})}{\text{T}(\text{B})} = 6 \]

\[ \text{Cost} \leq 100 \times 2 = 200 \]

\[ \text{Cardinality} = 100 \times 6 = 600 \]

\[ \text{assuming} \]
\[ \text{V}(\text{C}, \text{bid}) = \text{V}(\text{B}, \text{bid}) = \text{T}(\text{B}) = 50,000 \]
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   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  

\( V(B, \text{author}) = 500 \)  
7 <= age <= 24  

\( S, B, C \)  

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

<table>
<thead>
<tr>
<th>Relation</th>
<th>Size</th>
<th>Block Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>T(S)</td>
<td>B(S)</td>
</tr>
<tr>
<td>B</td>
<td>T(B)</td>
<td>B(B)</td>
</tr>
<tr>
<td>C</td>
<td>T(C)</td>
<td>B(C)</td>
</tr>
</tbody>
</table>

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

\[ T(B) = 50,000 \quad B(B) = 5,000 \]
\[ T(C) = 300,000 \quad B(C) = 15,000 \]

\[ B(S) = 1000 \]

\[ \text{Cardinality} = 600 \]
\( \text{(one student per checkout)} \)
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
T(B) = 50,000   B(B) = 5,000   7 \leq age \leq 24
T(C) = 300,000  B(C) = 15,000

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

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

\text{Cardinality = 600 \times 7/18 = 234 (approx)
\( S(\text{sid}, \text{name}, \text{age}, \text{addr}) \)
\( B(\text{bid}, \text{title}, \text{author}) : \text{Un. B+ on author} \)
\( C(\text{sid}, \text{bid}, \text{date}) : \text{Cl. B+ on bid} \)

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

Block nested loop S inner

Indexed-nested loop, B outer, C inner

File scan

Student S

Checkout C

Book B

Cost = 0 (on the fly)
Cardinality = 234
\[ S(\text{sid}, \text{name}, \text{age}, \text{addr}) \]
\[ B(\text{bid}, \text{title}, \text{author}): \text{Un. B+ on author} \]
\[ C(\text{sid}, \text{bid}, \text{date}): \text{Cl. B+ on bid} \]

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

\[
\begin{align*}
S(\text{sid}, \text{name}, \text{age}, \text{addr}) &\rightarrow \Pi_{\text{name}}(S) \\
B(\text{bid}, \text{title}, \text{author}) &\rightarrow \sigma_{12<\text{age}<20}(B) \\
C(\text{sid}, \text{bid}, \text{date}) &\rightarrow \Pi_{\text{bid}}(C) \\
&\rightarrow B(\text{bid}) \\
&\rightarrow S(\text{sid})
\end{align*}
\]

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

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

End of Lecture 12
Task 4: Efficiently searching the plan space

Use dynamic-programming based
Selinger’s algorithm

To be covered in Lecture 14
Heuristics for pruning plan space

• Apply predicates as early as possible
• Avoid plans with cross products
• Only left-deep join trees
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 on leaves

(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)
  – 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…R5:
Principle of Optimality

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

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

This has to be the optimal plan for joining $R3, R2, R4, R1$

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

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

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

We are using the associativity and commutativity of joins:
\[
(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)
\]
\[
R \bowtie S = S \bowtie R
\]

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: $\text{R1} \bowtie \text{R2} \bowtie \ldots \bowtie \text{Rn}$

Both are giving the same result
$\text{R2} \bowtie \text{R3} \bowtie \text{R1} = \text{R3} \bowtie \text{R1} \bowtie \text{R2}$

Optimal for joining $\text{R1, R2, R3}$

Sub-Optimal for joining $\text{R1, R2, R3}$
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 $R1 \bowtie R2 \bowtie R3$
Simple Cost Model

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

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:

\[
\text{OPT ( \{ R1, R2, R3 \} ) : } \\
\min \left\{ \\
\text{OPT ( \{ R1, R2 \} ) } + T ( \{ R1, R2 \} ) + T(R3) \\
\text{OPT ( \{ R2, R3 \} ) } + T ( \{ R2, R3 \} ) + T(R1) \\
\text{OPT ( \{ R1, R3 \} ) } + T ( \{ R1, R3 \} ) + T(R2) \\
\right\}
\]

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

e.g. All possible permutations of R1, R3, R4 have been considered after OPT({R1, R3, R4}) has been computed.
Selinger Algorithm:

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

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

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

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

Progress of algorithm:

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

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

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

\( \{ R1 \} \)  \( \{ R2 \} \)  \( \{ R3 \} \)  \( \{ R4 \} \)
Selinger Algorithm:

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

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

**Progress of algorithm**

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).

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

\[ \{ R1, R2, R3 \} \quad \{ R1, R2, R4 \} \quad \{ R1, R3, R4 \} \quad \{ R2, R3, R4 \} \]

\[ \{ R1, R2 \} \quad \{ R1, R3 \} \quad \{ R1, R4 \} \quad \{ R2, R3 \} \quad \{ R2, R4 \} \quad \{ R3, R4 \} \]

\[ \{ R1 \} \quad \{ R2 \} \quad \{ R3 \} \quad \{ R4 \} \]
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
Where are we now?

We learnt
- Relational Model and Query Languages
  - SQL, RA, RC
  - Postgres (DBMS)
    - HW1
- Database Normalization
- DBMS Internals
  - Storage
  - Indexing
  - Query Evaluation
  - Operator Algorithms
  - External sort
  - Query Optimization
- Map-reduce and spark
  - HW2

Next
- Transactions
  - Basic concepts
  - Concurrency control
  - Recovery
  - (for the next 4-5 lectures)