CompSci 516

Data Intensive Computing Systems

Lecture 10

Cost-based Query Optimization

Instructor: Sudeepa Roy
Announcements

• Solution of Homework-1 has been posted on sakai
  – Many equivalent solutions of the queries are possible

• Homework-2 has been posted
  – Due on February 29, Monday, 11:55 pm
  – Goal: review all key concepts covered so far, and practice for exams
  – Start early
  – Ask questions on piazza

• Xiaodan’s office hour canceled this week
  – Will be rescheduled

• Lecture Pdfs will be (mostly) posted right before the class
  – Don’t forget to see the updated version after the class
What will we learn?

• Last lecture:
  – Estimating cost of all operators and join algorithms

• Next:
  – Combine cost in a plan
  – Query Optimization
Reading Material

• [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
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
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

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

(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

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

Reduction factors

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

- Value selection
  
  \[ f_2 = \frac{T(R)}{V(R, B)} \]

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

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

Index Scan

R

Duke CS, Spring 2016

CompSci 516: Data Intensive Computing Systems

14
Cost of Index Scan with Selection (and multiple conditions)

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

What is $f$ if

- **Cost**: $B(R) \times f$ – if clustered
- **Size**: $T(R) \times f$ – if unclustered

Reduction factors

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

$$f = f_1 \times f_2 \text{ (assumes independence and uniform distribution)}$$

**Range 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 $R$

**Value selection**
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 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:
   • $V(R \bowtie S, A$ or $B) = V(R, A) = V(S, B)$
   • No value is lost in join

$R.A = S.B$
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
- \( B(R) \) : 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 \( 1/V(S.B) = \) reduction factor
- Under the two assumptions stated earlier + uniformity

\( T(R) \): 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
\( S(\text{sid}, \text{name}, \text{age}, \text{addr}) \quad T(S) = 10,000 \)

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

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

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

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

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

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

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

**Cost**

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

**Cardinality**

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

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

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

since \( V(S, \text{sid}) \geq V(C, \text{sid}) \) and

\( T(S) = V(S, \text{sid}) \)
$S(sid, name, age, addr)$  \hspace{1cm}  \text{T(S)}=10,000

$B(bid, title, author)$  \hspace{1cm}  \text{T(B)}=50,000

$C(sid, bid, date)$  \hspace{1cm}  \text{T(C)}=300,000

$v(B, author) = 500$

$7 <= age <= 24$

$B(S)=1,000$

$B(B)=5,000$

$B(C)=15,000$

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

\begin{equation}
\text{Cardinality} = \text{T(S} \bowtie \text{C}) = 300,000
\end{equation}

- foreign key join, don't need scanning for outer relation
S(sid,name,age,addr) \hspace{1cm} T(S)=10,000
B(bid,title,author) \hspace{1cm} T(B)=50,000
C(sid,bid,date) \hspace{1cm} 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)

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

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

(Tuple-based nested loop
B inner)

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

Cost = 0 (on the fly)

Cardinality = 300,000 * 1/500 * 7/18
= 234 (approx)
(assuming uniformity and independence)
\[
\begin{align*}
\text{S}(\text{sid}, \text{name}, \text{age}, \text{addr}) & \quad \text{T}(\text{S})=10,000 \\
\text{B}(\text{bid}, \text{title}, \text{author}) & \quad \text{T}(\text{B})=50,000 \\
\text{C}(\text{sid}, \text{bid}, \text{date}) & \quad \text{T}(\text{C})=300,000
\end{align*}
\]

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

\[
\text{Total cost} = 1,515,001,000
\]

\[
\text{Final cardinality} = 234 \text{ (approx)}
\]

(Tuple-based nested loop
B inner)

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

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

(Book B (File scan))

(On the fly) (a)

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

(B)
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  V(B,author) = 500

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  V(B,author) = 500

(a) σ_{author = 'Olden Fames'}

Checkpoint C
(Index scan)

Student S
(File scan)

Book B
(Index scan)

(b) \Pi_{bid}

(c) \Pi_{sid}

(d) \Pi_{bid}

(e) \Pi_{name}

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

(g) \Pi_{name}

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

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

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

(On the fly) (c) \Pi_{bid}

(Block nested loop S inner)

(Indexed-nested loop, B outer, C inner)
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_{name} |
| (d) Π_{sid} (On the fly) |
| (e) Π_{bid} |
| (f) σ_{12<age<20} |
| (g) Π_{name} |

(Book nested loop S inner)

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

(Student S (File scan))

(Checkout C (Index scan))

(Book B (Index scan))

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

Cardinality =
100
Student S
Checkout C
bid

T(S)=10,000  B(S)=1,000  V(B,author) = 500
B(B)=5,000  B(C)=15,000
T(C)=300,000

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

V
0 (on the fly)
Cardinality =
100
\( S(\text{sid}, \text{name}, \text{age}, \text{addr}) \)
\( B(\text{bid}, \text{title}, \text{author}): \text{Un. B+ on author} \)
\( C(\text{sid}, \text{bid}, \text{date}): \text{Cl. B+ on bid} \)

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

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

\( \text{S(id)} \) (File scan)
\( \text{B(bid)} \) (Index scan)
\( \text{C(sid,bid,date):} \) (Indexed-nested loop, B outer, C inner)

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

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

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

Cost \( \leq 100 \times 2 = 200 \)

Cardinality = \( 100 \times 6 = 600 \)

\( = 100 \times \frac{T(C)}{\text{MAX}(100, V(C, \text{bid}))} \)
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

<table>
<thead>
<tr>
<th>Table</th>
<th>rows</th>
<th>columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>10,000</td>
<td>4</td>
</tr>
<tr>
<td>T</td>
<td>50,000</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>300,000</td>
<td>3</td>
</tr>
</tbody>
</table>

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

Book B (Index scan)

Checkout C (File scan)

Student S (File scan)

Block nested loop S inner

Indexed-nested loop, B outer, C inner

 indexed

(On the fly) (a) σ author = 'Olden Fames'

(On the fly) (b) Π bid

(On the fly) (d) Π sid

(On the fly) (f) σ 12<age<20

(On the fly) (g) Π name

Cost = 0 (on the fly)
Cardinality = 600
\[ \text{Student } S \]
\[ \text{Checkout } C \]

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

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

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

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

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

\[ (\text{Index scan}) \quad (\text{e}) \quad \Pi \text{ bid} \]

\[ \sigma_{\text{author} = 'Olden Fames'} \]

\[ \text{Book } B \]

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

\[ \text{Outer relation is already in (unlimited) memory need to scan } S \text{ relation} \]

\[ \text{Cost} = \]

\[ B(S) = 1000 \]

\[ \text{Cardinality} = \]

\[ 600 \]

\[ \text{File scan} \]
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

7 \leq age \leq 24

B(B) = 5,000  
B(C) = 15,000  
T(C) = 300,000

Cost = 0 (on the fly)

Cardinality = 600 \times 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

Cost = 0 (on the fly)
Cardinality = 234
\( S(\text{sid, name, age, addr}) \)

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

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

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

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

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

\( \text{(total)} \)

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

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

(Block nested loop)

S inner

(On the fly) \( (e) \)

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

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

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

(Book B (Index scan))

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

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

(Student S (File scan))

(On the fly) \( (Total cost = 1300) \)

(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 \hspace{1cm} P2 \hspace{1cm} .... \hspace{1cm} Pn

C1 \hspace{1cm} C2 \hspace{1cm} .... \hspace{1cm} Cn

Pick minimum cost one

\{ \text{Physical plans} \}

\{ \text{Costs} \}
Join Trees

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

(left-deep join tree)

(bushy join tree)

(logical plan space)
- Several possible structures of the trees
- Each tree can have \( n! \) permutations of relations

(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 $R1 \ldots R5$: 
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

Suppose, this is an Optimal Plan for joining \( R1 \ldots R5 \):

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

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

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
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:

\[
\text{OPT} ( \{ R1, R2, R3 \} ) = \min\left\{ \begin{array}{l}
\text{OPT} ( \{ R1, R2 \} ) + T ( \{ R1, R2 \} ) + T(R3) \\
\text{OPT} ( \{ R2, R3 \} ) + T ( \{ R2, R3 \} ) + T(R1) \\
\text{OPT} ( \{ R1, R3 \} ) + T ( \{ R1, R3 \} ) + T(R2)
\end{array} \right\}
\]

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

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

Progress of algorithm

\[ \{ R1, R2, R3, R4 \} \]
\[ \{ R1, R2, R3 \} \quad \{ R1, R2, R4 \} \quad \{ R1, R3, R4 \} \quad \{ R2, R3, R4 \} \]
\[ \{ R1, R2 \} \quad \{ R1, R3 \} \quad \{ R1, R4 \} \quad \{ R2, R3 \} \quad \{ R2, R4 \} \quad \{ R3, R4 \} \]
\[ \{ R1 \} \quad \{ R2 \} \quad \{ R3 \} \quad \{ R4 \} \]
Selinger Algorithm:

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

Progress of algorithm
Selinger Algorithm:

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

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

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

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

\{ 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 \}
Full Example: Optimization with Selinger’s

Sailors (sid, sname, srating, age)
Boats(bid, bname, color)
Reserves(sid, bid, date, rname)

Query:
SELECT S.sid, R.rname
FROM Sailors S, Boats B, Reserves R
WHERE S.sid = R.sid
AND B.bid = R.bid
AND B.color = red

See yourself how to include actual operator algorithms and scanning methods while running Selinger’s

(Simple cost model is not useful in practice)
Available Indexes

- **Sailors**: \( S \) (sid, sname, srating, age)
- **Boats**: \( B \) (bid, bname, color)
- **Reserves**: \( R \) (sid, bid, date, rname)

  - Sid, bid foreign key in \( R \) referencing \( S \) and \( B \) resp.
  - **Sailors**
    - Unclustered B+ tree index on sid
    - Unclustered hash index on sid
  - **Boats**
    - Unclustered B+ tree index on color
    - Unclustered hash index on color
  - **Reserves**
    - Unclustered B+ tree on sid
    - Clustered B+ tree on bid
First Pass

• **Where to start?**
  – How to access each relation, assuming it would be the first relation being read
  – File scan is also available!

• **Sailors?**
  – No selection matching an index, use File Scan (no overhead)

• **Reserves?**
  – Same as Sailors

• **Boats?**
  – Hash index on color, matches $B\.color = \text{red}$
  – $B^+\text{ tree also matches the predicate, but hash index is cheaper}$
    • $B^+\text{ tree would be cheaper for range queries}$

```
S (sid, sname, srating, age):  B+tree - sid, hash index - sid
B (bid, bname, color) :      B+tree - color, hash index - color
R (sid, bid, date, rname) :  B+tree - sid, **Clustered** B+tree - bid
```

```
SELECT S.sid, R.rname
WHERE S.sid = R.sid
B.bid = R.bid, B.color = red
```
Second Pass

• What next?
  – For each of the plan in Pass 1 taken as outer, consider joining another relation as inner

• What are the combinations? How many new options?

<table>
<thead>
<tr>
<th>Outer</th>
<th>Inner</th>
<th>OPTION 1</th>
<th>OPTION 2</th>
<th>OPTION 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (file scan)</td>
<td>B</td>
<td>(B+-color)</td>
<td>(hash color)</td>
<td>(File scan)</td>
</tr>
<tr>
<td>R (file scan)</td>
<td>S</td>
<td>(B+-sid)</td>
<td>(hash sid)</td>
<td>„</td>
</tr>
<tr>
<td>S (file scan)</td>
<td>B</td>
<td>(B+-color)</td>
<td>(hash color)</td>
<td>„</td>
</tr>
<tr>
<td>S (file scan)</td>
<td>R</td>
<td>(B+-sid)</td>
<td>(Cl. B+ bid)</td>
<td>„</td>
</tr>
<tr>
<td>B (hash index)</td>
<td>R</td>
<td>(B+-sid)</td>
<td>(Cl. B+ bid)</td>
<td>„</td>
</tr>
<tr>
<td>B (hash index)</td>
<td>S</td>
<td>(B+-sid)</td>
<td>(hash sid)</td>
<td>„</td>
</tr>
</tbody>
</table>
S (sid, sname, srating, age): 1. B+tree - sid, 2. hash index - sid

SELECT S.sid, R.rname
WHERE S.sid = R.sid
B.bid = R.bid, B.color = red

Second Pass

• Which outer-inner combinations can be discarded?
  – B, S and S, B: Cartesian product!

<table>
<thead>
<tr>
<th>Outer</th>
<th>Inner</th>
<th>OPTION 1</th>
<th>OPTION 2</th>
<th>OPTION 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (file scan)</td>
<td>B</td>
<td>(B+-color)</td>
<td>(hash color)</td>
<td>(File scan)</td>
</tr>
<tr>
<td>R (file scan)</td>
<td>S</td>
<td>(B+-sid)</td>
<td>(hash sid)</td>
<td>”</td>
</tr>
<tr>
<td>S (file scan)</td>
<td>B</td>
<td>(B+-color)</td>
<td>(hash color)</td>
<td>”</td>
</tr>
<tr>
<td>S (file scan)</td>
<td>R</td>
<td>(B+-sid)</td>
<td>(Cl. B+ bid)</td>
<td>”</td>
</tr>
<tr>
<td>B (hash index)</td>
<td>S</td>
<td>(B+-bid)</td>
<td>(hash sid)</td>
<td>”</td>
</tr>
<tr>
<td>B (hash index)</td>
<td>R</td>
<td>(B+-sid)</td>
<td>(Cl. B+ bid):</td>
<td>”</td>
</tr>
</tbody>
</table>

OPTION 3 is not shown on next slide, expected to be more expensive
S (sid, sname, srating, age): 1. B+tree - sid, 2. hash index - sid  

```
SELECT S.sid, R.rname
WHERE S.sid = R.sid
B.bid = R.bid, B.color = red
```

<table>
<thead>
<tr>
<th>Outer</th>
<th>Inner</th>
<th>OPTION 1</th>
<th>OPTION 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R (file scan)</td>
<td>S</td>
<td>(B+-sid) Slower than hash-index</td>
<td>(hash sid): likely to be faster</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(need Sailor tuples matching S.sid = value, where value comes from an outer R tuple)</td>
<td>2A. <strong>Index nested loop join</strong></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>(B+-color) Not useful</td>
<td><strong>2B. Sort Merge based join:</strong> (no index is sorted on sid, need to sort, output sorted by sid, retained if cheaper)</td>
</tr>
<tr>
<td>S (file scan)</td>
<td>R</td>
<td>(B+-sid) Consider all methods</td>
<td><strong>(Cl. B+ bid): Not useful</strong></td>
</tr>
<tr>
<td>B (hash index)</td>
<td>R</td>
<td>(B+-sid) Not useful</td>
<td><strong>(Cl. B+ bid)</strong></td>
</tr>
</tbody>
</table>

**Keep the least cost plan between**
- (R, S) and (S, R)
- (R, B) and (B, R)

2A. **Index nested loop join**
(no H. I. on bid)

2B. **Sort-merge join**
(clustered, index sorted on bid, produces outputs in sorted order by bid, retained if cheaper)
Third Pass

- Join with the third relation
- For each option retained in Pass 2, join with the third relation
- E.g.
  - Boats (B+tree on color) – sort-merged-join – Reserves (B+tree on bid)
  - Join the result with Sailors (B+tree on sid) using sort-merge-join
    - Need to sort (B join R) by sid, was sorted on bid before
    - Outputs tuples sorted by sid
    - Not useful here, but will be useful if we had GROUP BY on sid
    - In general, a higher cost “interesting” plans may be retained (e.g. sort operator at root, grouping attribute in group by query later, join attribute in a later join)