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.:
    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
\[
T(R) \times f \text{ – if unclustered}
\]

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: \( B(R) \times f - \text{if clustered} \)

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

Reduction factors

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

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

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

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

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

\[ B(R) : \text{Number of blocks in } R \]

\[ V(R, A) : \text{Number of distinct values of attribute } A \text{ in } R \]
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

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

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

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

**Range selection**

\( T(R) \) : 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**

Duke CS, Spring 2016

CompSci 516: Data Intensive Computing Systems
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) * T(S)

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

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:
   • $V(R \bowtie S, A \text{ or } B) = V(R, A) = V(S, B)$
   • No value is lost in join

$\bowtie$
Size of Join

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

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

\[ R.A = S.B \]

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 \( 1/V(S.B) = \text{reduction factor} \)
- 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

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

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

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

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

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

7 <= \text{age} <= 24
**S** (sid, name, age, addr)  
**B** (bid, title, author)  
**C** (sid, bid, date)

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

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

\( V(B, author) = 500 \)

\( 7 \leq age \leq 24 \)

### 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

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

since \( V(S, sid) \geq V(C, sid) \) and
\( T(S) = V(S, sid) \)
\[ \text{Student } S : \quad \text{(File scan)} \]
\[ \text{Checkout } C : \quad \text{(File scan)} \]

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

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

\[ \text{V(B, author)} = 500 \quad 7 \leq \text{age} \leq 24 \]

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

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

\[ \text{(Block-nested loop, S outer, C inner)} \quad \text{sid} \quad \text{(a)} \quad \text{Book B} \quad \text{(File scan)} \]

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

\[ \sigma \quad 12 \leq \text{age} \leq 20 \quad \land \quad \text{author} = \text{\textquotesingle} \text{Olden Fames\textquotesingle} \]

\[ \text{Cost} = \quad T(S \bowtie C) \times B(B) \]

\[ = 300,000 \times 5,000 \quad = 15 \times 10^8 \]

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

\[ \cdot \quad \text{foreign key join, don't need scanning for outer relation} \]
\[
(c, d)
\]

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

(Book-based nested loop, S outer, C inner)

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

\[
\begin{align*}
\text{Cost} & = 0 \text{ (on the fly)} \\
\text{Cardinality} & = 300,000 \times \frac{1}{500} \times \frac{7}{18} \\
& = 234 \text{ (approx)} \text{ (assuming uniformity and independence)}
\end{align*}
\]
\[\sigma_{12<\text{age}<20} \land \text{author} = \text{'Olden Fames'}\]

\[\Pi_{\text{name}}\]

Total cost = 1,515,001,000

Final cardinality = 234 (approx)
Physical Query Plan – 2

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

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

Book B
(Index scan)

Checkout C
(Index scan)

Student S
(File scan)

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

\begin{align*}
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
\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  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) Student S inner
(f) σ_{12 < age < 20}
(g) Π_{name}

(On the fly) (Indexed-nested loop, 
B outer, C inner)

(On the fly) (Block nested loop
S inner)

Cost =
T(B) / V(B, author)
= 50,000/500
= 100 (unclustered)

Cardinality =
100
\begin{align*}
S(sid, name, age, addr) & \quad T(S) = 10,000 \quad B(S) = 1,000 \quad V(B, author) = 500 \\
B(bid, title, author): \text{Un. B+ on author} & \quad T(B) = 50,000 \quad B(B) = 5,000 \\
C(sid, bid, date): \text{Cl. B+ on bid} & \quad T(C) = 300,000 \quad B(C) = 15,000
\end{align*}

\begin{align*}
\sigma_{12 \leq \text{age} \leq 20} \quad (f) & \\
\Pi_{\text{name}} \quad (g) &
\end{align*}

\begin{align*}
\text{(Block nested loop S inner)} & \\
\Pi_{\text{sid}} \quad (d) &
\end{align*}

\begin{align*}
\text{(Indexed-nested loop, B outer, C inner)} & \\
\Pi_{\text{bid}} \quad (b) &
\end{align*}

\begin{align*}
\sigma_{\text{author} = \text{‘Olden Fames’}} \quad (a) & \\
\text{Student S (File scan)} & \\
\text{Checkout C} & \\
\text{Book B (Index scan)} & \\
\text{Cost} = 0 \text{ (on the fly)} & \\
\text{Cardinality} = 100
\end{align*}
S(sid,name,age,addr)  
B(bid,title,author): Un. B+ on author  
C(sid,bid,date): Cl. B+ on bid  

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

- 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

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

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

\[ = 100 \times \frac{T(C)}{\text{MAX}(100, V(C, bid))} \]
assuming
\[ V(C, bid) = V(B, bid) = T(B) = 50,000 \]
S(sid, name, age, addr)
B(bid, title, author): Un. B+ on author
C(sid, bid, date): Cl. B+ on bid

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

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

Cost = 0 (on the fly)
Cardinality = 600
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

\[
\sigma_{12<\text{age}<20}
\]

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

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

Block nested loop
S inner)  \(\Pi_{\text{bid}}\)

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

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

(a) \(\sigma_{\text{author} = \text{"Olden Fames"}}\)
Book B
(Index scan)

Student S
(File scan)

Checkout C
(Index scan)

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

Cost =
B(S) = 1000

Cardinality =
600
\[ S(\text{sid}, \text{name}, \text{age}, \text{addr}) \]
\[ B(\text{bid}, \text{title}, \text{author}) : \text{Un. B+ on author} \]
\[ C(\text{sid}, \text{bid}, \text{date}) : \text{Cl. B+ on bid} \]

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

\[ \sigma_{\text{age} < 20} \]

\[ \Pi_{\text{name}} \]

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

\[ \Pi_{\text{bid}} \]

\[ \Pi_{\text{sid}} \]

\[ \Pi_{\text{bid}} \]

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

\[ B(\text{bid}, \text{title}, \text{author}) \]

\[ C(\text{sid}, \text{bid}, \text{date}) \]

\[ S(\text{sid}, \text{name}, \text{age}, \text{addr}) \]

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

T(S) = 10,000  B(S) = 1,000  V(B, author) = 500
B(B) = 5,000
T(C) = 300,000  B(C) = 15,000  7 <= age <= 24

Student S

Checkout C

Book B

Cost = 0 (on the fly)
Cardinality = 234
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  

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

\( \Pi_{\text{name}} \)

(On the fly)

(On the fly)

(Block nested loop, S inner)

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

(On the fly)

(On the fly)

(On the fly)

(On the fly)

Index scan

File scan

Block nested loop

Student S

Checkout C

Book B

Total cost = 1300

(compare with 1,515,001,000 for plan 1!)

Final cardinality = 234 (approx)

(same as plan 1!)

(Duke CS, Spring 2016)
Task 4:
Efficiently searching the plan space

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

• Predicates as early as possible
• Avoid plans with cross products
• Only left-deep join trees
Physical Plan Selection

Logical Query Plan

P1  P2  ....  Pn

C1  C2  ....  Cn

Pick minimum cost one
Join Trees

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

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

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

left-deep join tree

bushy join tree

Duke CS, Spring 2016

CompSci 516: Data Intensive Computing Systems
Selinger Algorithm

• **Dynamic Programming based**

• **Dynamic Programming:**
  – General algorithmic paradigm
  – Exploits “principle of optimality”
  – Useful reading:
    – Chapter 16, Introduction to Algorithms, Cormen, Leiserson, Rivest

• **Considers the search space of left-deep join trees**
  – reduces search space (only one structure), still $n!$ permutations
  – interacts well with join algos (esp. NLJ)
  – e.g. might not need to write tuples to disk if enough memory
Principle of Optimality

Optimal for “whole” made up from optimal for “parts”
Principle of Optimality

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

Suppose, this is an Optimal Plan for joining $R1 \ldots R5$: 
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:  \( R1 \Join R2 \Join R3 \Join R4 \Join R5 \)

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:  \( R1 \Join R2 \Join R3 \Join R4 \Join R5 \)

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

We are using the associativity and commutativity of joins

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

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

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

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

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

Suppose you chose the sub-optimal one

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

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

OPT ( { R1, R2, R3 } ): 
Cost of optimal plan to join R1,R2,R3

T ( { R1, R2, R3 } ): 
Number of tuples in R1 \Join R2 \Join R3
Selinger Algorithm:

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

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

Progress of algorithm
Simple Cost Model

Cost (R \bowtie S) = T(R) + T(S)

All other operators have 0 cost

Note: The simple cost model used for illustration only, it is not used in practice
Cost Model Example

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

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

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

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

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

Progress of algorithm
Selinger Algorithm:

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

Progress of algorithm
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: \( R_1 \Join R_2 \Join R_3 \Join R_4 \)

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

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

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: (*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 \}

Progress of algorithm
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**: $S$, **Boats**: $B$, **Reserves**: $R$

- 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 = red
  - B+ tree also matches the predicate, but hash index is cheaper
    - B+ tree would be cheaper for range queries
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>
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+-sid)</td>
<td>(hash sid)</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

### B (bid, bname, color):
- 1. B+tree - color
- 2. hash index - color

### R (sid, bid, date, rname):
- 1. B+tree - sid
- 2. **Clustered** B+tree - bid

---

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

---

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