Query Optimization

Introduction to Databases
CompSci 316 Spring 2022
Announcements (Tue Nov 8)

• MS-3 due Thursday 11/8
• HW6 to be released soon, two parts
  • Sort + join processing (due next Thursday 11/17)
  • JSON (Thursday + discussion this week) (may have a few extra days)
• Gradiance will be released today, due next Wednesday 11/16
• Sudeepa’s OH this week moved to Friday 10:30 (zoom + in person)
Query optimization

• One logical plan → “best” physical plan

• Questions
  • How to enumerate possible plans
  • How to estimate costs
  • How to pick the “best” one

• Often the goal is not getting the optimum plan, but instead avoiding the horrible ones

Any of these will do

1 second 1 minute 1 hour
Plan enumeration in relational algebra

- Apply relational algebra equivalences

Join reordering: \( \times \) and \( \bowtie \) are associative and commutative (except column ordering, but that is unimportant)

\[
\begin{align*}
R \bowtie S \bowtie T &= \quad = \quad = \quad = \\
S \bowtie R \bowtie T &= \\
S \bowtie T \bowtie R &= \\
R \bowtie T \bowtie S &= \ldots
\end{align*}
\]
More relational algebra equivalences

- Convert $\sigma_p \times$ to/from $\bowtie_p$: $\sigma_p (R \times S) = R \bowtie_p S$
- Merge/split $\sigma$’s: $\sigma_{p_1} (\sigma_{p_2} R) = \sigma_{p_1 \wedge p_2} R$
- Merge/split $\pi$’s: $\pi_{L_1} (\pi_{L_2} R) = \pi_{L_1} R$, where $L_1 \subseteq L_2$
- Push down/pull up $\sigma$:
  $\sigma_{p \wedge p_r \wedge p_s} (R \bowtie_{p'} S) = (\sigma_{p_r} R) \bowtie_{p \wedge p'} (\sigma_{p_s} S)$, where
  - $p_r$ is a predicate involving only $R$ columns
  - $p_s$ is a predicate involving only $S$ columns
  - $p$ and $p'$ are predicates involving both $R$ and $S$ columns
- Push down $\pi$:
  $\pi_L (\sigma_p R) = \pi_L \left( \sigma_p (\pi_{L'} R) \right)$, where
  - $L'$ is the set of columns referenced by $p$ that are not in $L$
- Many more (seemingly trivial) equivalences...
  - Can be systematically used to transform a plan to new ones
Relational query rewrite example

\[ \pi_{\text{Group.name}} \sigma_{\text{User.name} = \text{“Bart”} \land \text{User.uid = Member.uid} \land \text{Member.gid = Group.gid}} \]

Push down \( \sigma \)

Convert \( \sigma_p \times \) to \( \bowtie_p \)
Heuristics-based query optimization

• Start with a logical plan
• Push selections/projections down as much as possible
  • Why? Reduce the size of intermediate results
  • Why not? May be expensive; maybe joins filter better
• Join smaller relations first, and avoid cross product
  • Why? Reduce the size of intermediate results
  • Why not? Size depends on join selectivity too
• Convert the transformed logical plan to a physical plan (by choosing appropriate physical operators)
SQL query rewrite

• More complicated—subqueries and views divide a query into nested “blocks”
  • Processing each block separately forces particular join methods and join order
  • Even if the plan is optimal for each block, it may not be optimal for the entire query
• Unnest query: convert subqueries/views to joins
  ➥ We can just deal with select-project-join queries
    • Where the clean rules of relational algebra apply
SQL query rewrite example

• SELECT name
  FROM User
  WHERE uid = ANY (SELECT uid FROM Member);

• SELECT name
  FROM User, Member
  WHERE User.uid = Member.uid;
  
  • Wrong—consider two Bart’s, each joining two groups

• SELECT name
  FROM (SELECT DISTINCT User.uid, name
        FROM User, Member
        WHERE User.uid = Member.uid);
  
  • Right—assuming User.uid is a key
Dealing with correlated subqueries

• SELECT gid FROM Group
  WHERE name LIKE 'Springfield%' 
  AND min_size > (SELECT COUNT(*) FROM Member
    WHERE Member.gid = Group.gid);

• SELECT gid
  FROM Group, (SELECT gid, COUNT(*) AS cnt
    FROM Member GROUP BY gid) t
  WHERE t.gid = Group.gid AND min_size > t.cnt
  AND name LIKE 'Springfield%';
  • New subquery is inefficient (it computes the size for every group)
  • Suppose a group is empty?
Heuristics vs. cost-based optimization

- **Heuristics-based optimization**
  - Apply heuristics to rewrite plans into cheaper ones

- **Cost-based optimization**
  - **Rewrite** logical plan to combine “blocks” as much as possible
  - **Optimize** query block by block
    - Enumerate logical plans (already covered)
    - Estimate the cost of plans
    - Pick a plan with acceptable cost
  - **Focus**: select-project-join blocks
Cost estimation

Physical plan example:

• We have: cost estimation for each operator
  • Example: \( \text{SORT}(\text{gid}) \) takes \( O(B(\text{input}) \times \log_M B(\text{input})) \)
    • But what is \( B(\text{input}) \)?

• We need: size of intermediate results
Cardinality estimation
Selections with equality predicates

• $Q: \sigma_{A=\nu} R$

• Suppose the following information is available
  • Size of $R$: $|R|$
  • Number of distinct $A$ values in $R$: $|\pi_A R|$

• Assumptions
  • Values of $A$ are uniformly distributed in $R$
  • Values of $\nu$ in $Q$ are uniformly distributed over all $R.A$ values

• $|Q| \approx |R|/|\pi_A R|$
  • Selectivity factor of $(A = \nu)$ is $1/|\pi_A R|$
Conjunctive predicates

• $Q: \sigma_{A=u} \land B=v^R$

• Additional assumptions
  • $(A = u)$ and $(B = v)$ are independent
    • Counterexample: major and advisor
  • No “over”-selection
    • Counterexample: $A$ is the key

• $|Q| \approx \frac{|R|}{|\pi_A R| \cdot |\pi_B R|}$
  • Reduce total size by all selectivity factors
Negated and disjunctive predicates

- $Q: \sigma_{A \neq v} R$
  - $|Q| \approx |R| \cdot \left(1 - \frac{1}{|\pi_A R|}\right)$
    - Selectivity factor of $\neg p$ is $(1 - \text{selectivity factor of } p)$

- $Q: \sigma_{A=u \lor B=v} R$
  - $|Q| \approx |R| \cdot \left(\frac{1}{|\pi_A R|} + \frac{1}{|\pi_B R|}\right)$?
    - No! Tuples satisfying $(A = u)$ and $(B = v)$ are counted twice
  - $|Q| \approx |R| \cdot \left(\frac{1}{|\pi_A R|} + \frac{1}{|\pi_B R|} - \frac{1}{|\pi_A R||\pi_B R|}\right)$
    - Inclusion-exclusion principle
Range predicates

- $Q: \sigma_{A \geq v}R$

- Not enough information!
  - Just pick, say, $|Q| \approx |R| \cdot 1/3$

- With more information
  - Largest R.A value: $\text{high}(R.A)$
  - Smallest R.A value: $\text{low}(R.A)$
  - $|Q| \approx |R| \cdot \frac{\text{high}(R.A) - v}{\text{high}(R.A) - \text{low}(R.A)}$

- In practice: sometimes the second highest and lowest are used instead
  - The highest and the lowest are often used by inexperienced database designer to represent invalid values!
Two-way equi-join

- \( Q: R(A, B) \bowtie S(A, C) \)

- Assumption: containment of value sets
  - Every tuple in the “smaller” relation (one with fewer distinct values for the join attribute) joins with some tuple in the other relation
  - That is, if \( |\pi_A R| \leq |\pi_A S| \) then \( \pi_A R \subseteq \pi_A S \)
  - Certainly not true in general
  - But holds in the common case of foreign key joins

- \( |Q| \approx \frac{|R| \cdot |S|}{\max(|\pi_A R|, |\pi_A S|)} \)
  - Selectivity factor of \( R. A = S. A \) is \( \frac{1}{\max(|\pi_A R|, |\pi_A S|)} \)
Cost estimation: summary

• Using similar ideas, we can estimate the size of projection, duplicate elimination, union, difference, aggregation (with grouping)

• Lots of assumptions and very rough estimation
  • Accurate estimate is not needed
  • Maybe okay if we overestimate or underestimate consistently
  • May lead to very nasty optimizer “hints”

```
SELECT * FROM User WHERE pop > 0.9;
SELECT * FROM User WHERE pop > 0.9 AND pop > 0.9;
```

• Not covered: better estimation using histograms
Search strategy
Search space

• Huge!

• “Bushy” plan example:

> Just considering different join orders, there are \( \frac{(2n-2)!}{(n-1)!} \) bushy plans for \( R_1 \bowtie \cdots \bowtie R_n \)
  
  • 30240 for \( n = 6 \)

• And there are more if we consider:
  
  • Multiway joins
  
  • Different join methods
  
  • Placement of selection and projection operators
Left-deep plans

- Heuristic: consider only "left-deep" plans, in which only the left child can be a join
  - Tend to be better than plans of other shapes, because many join algorithms scan inner (right) relation multiple times—you will not want it to be a complex subtree

- How many left-deep plans are there for $R_1 \bowtie \cdots \bowtie R_n$?
  - Significantly fewer, but still lots—$n!$ (720 for $n = 6$)
Selinger’s algorithm: A dynamic programming approach

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

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 $R1...R5$: 
Suppose, this is an Optimal Plan for joining R1…R5:

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

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:

\[
(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)
\]

\[
R \bowtie S = S \bowtie R
\]
Selinger Algorithm:

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

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

Pat Selinger
IBM Fellow, ACM Fellow, AAAS Fellow
A.B. (1971), S.M. (1972), and Ph.D. (1975) degrees in applied mathematics from Harvard University
https://en.wikipedia.org/wiki/Patricia_Selinger
Selinger Algorithm:

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

Progress of algorithm
Selinger Algorithm:

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

- All possible permutations of \( R1, R3, R4 \) have been considered after \( \text{OPT}\{R1, R3, R4\} \) has been computed.

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.

\{ R_1, R_2, R_3, R_4 \}
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.

![Diagram showing the progress of the algorithm](image-url)
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: $R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4$

Q. How to optimally compute join of $\{R_3\}$?


Progress of algorithm
Selinger Algorithm: Output

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.
The need for “interesting order”

- Optimal plan may not have an optimal sub-plan in practice!
- Example: $R(A, B) \bowtie S(A, C) \bowtie T(A, D)$
- Best plan for $R \bowtie S$: hash join (beats sort-merge join)
- Best overall plan: sort-merge join $R$ and $S$, and then sort-merge join with $T$
  - Subplan of the optimal plan is not optimal!
- Why?
  - The result of the sort-merge join of $R$ and $S$ is sorted on $A$
  - This is an interesting order that can be exploited by later processing (e.g., join, dup elimination, GROUP BY, ORDER BY, etc.)!
Summary

• Relational algebra equivalence
• SQL rewrite tricks
• Heuristics-based optimization
• Cost-based optimization
  • Need statistics to estimate sizes of intermediate results
  • Greedy approach
  • Dynamic programming approach (Selinger Algorithm)
Practice problem:
Estimating the cost of the entire plan
Physical Query Plan

```plaintext
S(sid, name, age, addr)  no. of tuples  S
B(bid, title, author)  10,000  T(S)=10,000
C(sid, bid, date)  50,000  T(B)=50,000
no. of pages  300,000  T(C)=300,000

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

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

Block nested loop
S inner

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

(On the fly) (g) \( \Pi \) name

(On the fly) (f) \( \sigma \) 12<age<20

(On the fly) (d) \( \Pi \) sid

(On the fly) (c) bid

(On the fly) (b) \( \Pi \) bid

(On the fly) (a) \( \sigma \) author = 'Olden Fames'

Book B (Index scan)

Checkout C (Index scan)

Student S (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 <= age <= 24

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

Cardinality =
100
S(sid,name,age,addr) T(S)=10,000 B(S)=1,000 V(B,author) = 500
B(bid,title,author): Un. B+ on author T(B)=50,000 B(B)=5,000
7 <= age <= 24
C(sid,bid,date): Cl. B+ on bid T(C)=300,000 B(C)=15,000

\( B(S) = 1,000 \)
\( B(B) = 5,000 \)
\( B(C) = 15,000 \)
\( T(S) = 10,000 \)
\( T(B) = 50,000 \)
\( T(C) = 300,000 \)

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

\( \text{Cost} = 0 \text{ (on the fly)} \)
\( \text{Cardinality} = 100 \)
S(sid, name, age, addr) T(S) = 10,000
B(bid, title, author): Un. B+ on author B(S) = 1,000
C(sid, bid, date): Cl. B+ on bid B(B) = 5,000

V(B, author) = 500
T(B) = 50,000 7 <= age <= 24
T(C) = 300,000

B(C) = 15,000

- one index lookup per outer B tuple
- 1 book has T(C)/T(B) = 6 checkouts (uniformity)
- # C tuples per page = 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

= 100 * T(C)/ MAX(100, V(C, bid)) assuming
V(C, bid) = V(B, bid) = T(B) = 50,000
S(sid, name, age, addr)  T(S) = 10,000  V(B, author) = 500
B(bid, title, author): Un. B+ on author  B(S) = 1,000
C(sid, bid, date): Cl. B+ on bid  B(B) = 5,000
7 <= age <= 24
T(B) = 50,000
T(C) = 300,000
B(C) = 15,000

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

(On the fly) \( \sigma_{12 < \text{age} < 20} \)
(Block nested loop S inner)

(On the fly) \( \Pi_{\text{sid}} \)
(Indexed-nested loop, B outer, C inner)

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

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

(c) Student S
(File scan)

(b) Book B
(Index scan)

(d) Checkout C
(Index scan)

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

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

\[ T(B) = 50,000 \]
\[ B(B) = 5,000 \]
\[ 7 \leq age \leq 24 \]

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

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

\[ 7 \leq age \leq 24 \]

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

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

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

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

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

\( \prod_{\text{bid}} \)

\( \prod_{\text{sid}} \)

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

Total cost = 1300

Final cardinality = 234 (approx)