Introduction to Database Systems
CSE 444

Lecture 20: Overview of
Query Optimization
Announcements

• Project 3 is due tonight
  – How is it going?

• HW3 is out and is due on next Wednesday
  – Rather short assignment
  – But start early in case you have questions

• Project 4 will be out shortly (last assignment)
  – Group assignment: 2 to 4 students (but 1 ok too)
Where We Are

• We are learning how a DBMS executes a query

• What we learned so far
  – How data is stored and indexed (lectures 15 and 16)
  – Logical query plans: relational algebra (lecture 17)
  – Steps involved in processing a query (lecture 18)
  – Operator algorithms (lecture 19)

• Today
  – How to select logical & physical query plans
  – Chapter 16 in the book (recommended, not required)
Query Optimization Goal

• For a query
  – There exists many logical and physical query plans
  – Query optimizer needs to pick a good one
Query Optimization Algorithm

• Enumerate alternative plans

• Compute estimated cost of each plan
  – Compute number of I/Os
  – Compute CPU cost

• Choose plan with lowest cost
  – This is called cost-based optimization
Outline

• Search space

• Algorithm for enumerating query plans

• Estimating the cost of a query plan
Relational Algebra Equivalences

• **Selections**
  – Commutative: \( \sigma_{c_1}(\sigma_{c_2}(R)) \) same as \( \sigma_{c_2}(\sigma_{c_1}(R)) \)
  – Cascading: \( \sigma_{c_1 \land c_2}(R) \) same as \( \sigma_{c_2}(\sigma_{c_1}(R)) \)

• **Projections**
  – Cascading

• **Joins**
  – Commutative: \( R \bowtie S \) same as \( S \bowtie R \)
  – Associative: \( R \bowtie (S \bowtie T) \) same as \( (R \bowtie S) \bowtie T \)
Left-Deep Plans and Bushy Plans

Left-deep plan

Bushy plan
Relational Algebra Equivalences

• Selects, projects, and joins
  – We can commute and combine all three types of operators
  – We just have to be careful that the fields we need are available when we apply the operator
  – Relatively straightforward. See book 16.2
Search Space Challenges

• Search space is huge!
  – Many possible equivalent trees
  – Many implementations for each operator
  – Many access paths for each relation
    • File scan or index + matching selection condition

• Cannot consider ALL plans
• Want search space that includes low-cost plans
Outline

• Search space

• Algorithms for enumerating query plans

• Estimating the cost of a query plan
Key Decisions

- When selecting a plan, some of the most important decisions include:
  - **Logical plan**
    - Can we push selections down?
    - Can we push projections or aggregations down?
    - What order to use for joins?
  - **Physical plan**
    - What join algorithms to use?
    - What access paths to use (file scan or index)?
Plan Enumeration Algorithms

• Rule-based vs cost-based algorithms

• Logical plans
  – Heuristic-based algorithms
  – Use size of intermediate results as cost measure

• Physical plans
  – Top-down algorithms or
  – Bottom-up: dynamic programming approaches
    • Also called “Selinger-style” optimizers
  – Use heuristics to limit search space
Outline

• Search space

• Algorithms for enumerating query plans

• Estimating the cost of a query plan
Computing the Cost of a Plan

• Collect statistical summaries of stored data

• Compute cost in a bottom-up fashion

• For each operator compute
  – Estimate cost of executing the operation
  – Estimate statistical summary of the output data
Statistics on Base Data

• Collected information for each relation
  – Number of tuples (cardinality)
  – Indexes, number of keys in the index
  – Number of physical pages, clustering info
  – Statistical information on attributes
    • Min value, max value, number distinct values
    • Histograms
  – Correlations between columns (hard)

• Collection approach: periodic, using sampling
Retrieving data from Storage

- **Access path**: a way to retrieve tuples from a table
  - A file scan
  - An index *plus* a matching selection condition

- Index matches selection condition if it can be used to retrieve just tuples that satisfy the condition
  - Example: `Supplier(sid, sname, scity, sstate)`
  - B+-tree index on `(scity, sstate)`
    - matches `scity='Seattle'`
    - does not match `sid=3`, does not match `sstate='WA'`
Access Path Selection

- Supplier(sid, sname, scity, sstate)

- Selection condition: sid > 300 ∧ scity='Seattle'

- Indexes: B+-tree on sid and B+-tree on scity

- Which access path should we use?

- We should pick the **most selective** access path
Access Path Selectivity

• Access path selectivity is the number of pages retrieved if we use this access path
  – Most selective retrieves fewest pages

• As we saw earlier, for equality predicates
  – Selection on equality: $\sigma_{a=v}(R)$
  – $V(R, a) = \# \text{ of distinct values of attribute } a$
  – $1/V(R,a)$ is thus the reduction factor
  – Clustered index on $a$: cost $B(R)/V(R,a)$
  – Unclustered index on $a$: cost $T(R)/V(R,a)$
  – (we are ignoring I/O cost of index pages for simplicity)
Selectivity for Range Predicates

- Selection on range: $\sigma_{a>v}(R)$

- How to compute the selectivity?
  - Assume values are uniformly distributed
  - Reduction factor $X$
  - $X = (\text{Max}(R,a) - v) / (\text{Max}(R,a) - \text{Min}(R,a))$

- Clustered index on $a$, cost is $B(R) \times X$
- Unclustered index on $a$, cost is $T(R) \times X$
Back to Our Example

• Selection condition: \( \text{sid} > 300 \land \text{scity} = \text{Seattle} \)
  – Index I1: B+-tree on sid clustered
  – Index I2: B+-tree on scity unclustered

• Let’s assume
  – \( V(\text{Supplier}, \text{scity}) = 20 \)
  – \( \text{Max}(\text{Supplier}, \text{sid}) = 1000, \text{Min}(\text{Supplier}, \text{sid}) = 1 \)
  – \( B(\text{Supplier}) = 100, T(\text{Supplier}) = 1000 \)

• Cost I1: \( B(R) \times \frac{\text{Max}-v}{\text{Max-Min}} = 100 \times \frac{700}{999} \approx 70 \)
• Cost I2: \( T(R) \times \frac{1}{V(\text{Supplier}, \text{scity})} = 1000 \div 20 = 50 \)
Selectivity with Multiple Conditions

What if we have an index on multiple attributes?
• Example selection $\sigma_{a=v_1 \land b= v_2}(R)$ and index on $<a,b>$

How to compute the selectivity?
• Assume attributes are independent
• $X = 1 / (V(R,a) \ast V(R,b))$

• Clustered index on $<a,b>$: cost $B(R)\ast X$
• Unclustered index on $<a,b>$: cost $T(R)\ast X$
Computing Cost of an Operator

• The cost of executing an operator depends
  – On the operator implementation
  – On the input data

• We learned how to compute this in the previous lecture, so we do not repeat it here
Statistics on the Output Data

• Most important piece of information
  – **Size of operator result**
  – i.e., the number of output tuples

• **Projection**: output size same as input size
• **Selection**: multiply input size by reduction factor
  – Similar to what we did for estimating access path selectivity
  – Assume independence between conditions in the predicate
  – (use product of the reduction factors for the terms)
Estimating Result Sizes

- For joins $R \bowtie S$
  - Take product of cardinalities of relations $R$ and $S$
  - Apply reduction factors for each term in join condition
  - Terms are of the form: $\text{column1} = \text{column2}$
  - Reduction: $\frac{1}{\text{MAX}(V(R,\text{column1}), V(S,\text{column2}))}$
  - Assumes each value in smaller set has a matching value in the larger set
Our Example

- Suppliers(sid, sname, scity, sstate)
- Supplies(pno, sid, quantity)
- Some statistics
  - $T(\text{Supplier}) = 1000$ records
  - $B(\text{Supplier}) = 100$ pages
  - $T(\text{Supplies}) = 10,000$ records
  - $B(\text{Supplies}) = 100$ pages
  - $V(\text{Supplier}, \text{scity}) = 20$, $V(\text{Supplier}, \text{state}) = 10$
  - $V(\text{Supplies}, \text{pno}) = 3,000$
  - Both relations are clustered
Physical Query Plan 1

(On the fly) \( \pi_{\text{sname}} \) Selection and project on-the-fly

(On the fly) \( \sigma_{\text{scity}='Seattle' \land \text{sstate}='WA' \land \text{pno}=2} \) -> No additional cost.

(Nested loop) \( \text{sno} = \text{sno} \)

Suppliers
(File scan)

Supplies
(File scan)

Total cost of plan is thus cost of join:

\[ \text{B(Supplier)} + \text{B(Supplier)} \times \text{B(Supplies)} \]

\[ = 100 + 100 \times 100 \]

\[ = 10,100 \text{ I/Os} \]
Physical Query Plan 2

(On the fly)

\[ \pi_{\text{sname}} \]

(4)

Total cost
\[ = 100 + 100 \times \frac{1}{20} \times \frac{1}{10} \]
\[ + 100 + 100 \times \frac{1}{3000} \]
\[ + 2 \]
\[ + 0 \]
\[ \approx 204 \text{ I/Os} \]

(Sort-merge join)

\[ \sigma_{\text{sno} = \text{sno}} \]

(3)

(Scan write to T1)

\[ \sigma_{\text{scity} = 'Seattle' \land \text{sstate} = 'WA'} \]

(1)

Suppliers (File scan)

(Scan write to T2)

\[ \sigma_{\text{pno} = 2} \]

(2)

Supplies (File scan)
Physical Query Plan 3

(On the fly) (4) \( \pi_{sname} \)

(On the fly)

(3) \( \sigma_{\text{scity}='Seattle'} \land \text{sstate}='WA' \)

(2) \( \text{sno} = \text{sno} \) (Index nested loop)

(Use hash index)

(1) \( \sigma_{pno=2} \)

Supplies

(Hash index on pno)

Assume: clustered

Suppliers

(Hash index on sno)

Clustering does not matter

Total cost

= 1 (1)
+ 4 (2)
+ 0 (3)
+ 0 (3)

Total cost \( \approx 5 \) I/Os
Simplifications

• In the previous examples, we assumed that all index pages were in memory
  – When this is not the case, we need to add the cost of fetching index pages from disk

• We also assumed that CPU time is irrelevant
  – Not the entire story in production systems