Announcements

- WQ4 is due Friday 11pm
- HW3 is due next Tuesday 11pm
- Midterm is next Monday

Motivation

- To understand performance, need to understand a bit about how a DBMS works
  - my database application is too slow… why?
  - one of the queries is very slow… why?
- Under your direct control: index choice
  - understand how that affects query performance

Recap: Query Evaluation

SQL query

1. Parse & Check Query
2. Decide how best to answer query: query optimization
3. Query Execution
4. Return Results

Query Optimizer Overview

- Input: Parsed & checked SQL
- Output: A good physical query plan
- Basic query optimization algorithm:
  - Enumerate alternative plans (logical and physical)
  - Compute estimated cost of each plan
    - Compute number of I/Os
    - Optionally take into account other resources
  - Choose plan with lowest cost
  - This is called cost-based optimization

Query Optimizer Overview

- There are exponentially many query plans
  - exponential in the size of the query
  - simple SFW with 3 joins does not have too many
- Optimizer will consider many, many of them
- Worth substantial cost to avoid bad plans
Rest of Today

- Cost of reading from disk
- Cost of single RA operators
- Cost of query plans

Cost of Reading Data From Disk

Cost Parameters

- Cost = Disk I/O + CPU + Network I/O
  - We will focus on Disk I/O
- Parameters:
  - $B(R) =$ # of blocks (i.e., pages) for relation R
  - $T(R) =$ # of tuples in relation R
  - $V(R, A) =$ # of distinct values of attribute A
    - When A is a key, $V(R, A) = T(R)$
    - When A is not a key, $V(R, A)$ can be anything < $T(R)$
- Where do these values come from?
  - DBMS collects statistics about data on disk

Selectivity Factors for Conditions

- $A = c$ 
  - Selectivity = $1/V(R, A)$
- $A < c$ 
  - Selectivity = $(c - \text{Low}(R, A))/((\text{High}(R, A) - \text{Low}(R, A))$
- $c_1 < A < c_2$ 
  - Selectivity = $(c_2 - c_1)/((\text{High}(R, A) - \text{Low}(R, A))$

Example: Selectivity of $\sigma_{A=c}(R)$

- $T(R) = 100,000$
- $V(R, A) = 20$
- How many records are returned by $\sigma_{A=c}(R)$ = ?
- Answer: $X \times T(R)$, where $X =$ selectivity...
  - $X = 1/V(R, A) = 1/20$
- Number of records returned = $100,000/20 = 5,000$

Cost of Index-based Selection

- Sequential scan for relation R costs $B(R)$
- Index-based selection
  - Estimate selectivity factor $X$ (see previous slide)
  - Clustered index: $X \times B(R)$
  - Unclustered index $X \times T(R)$
- Note: we are ignoring I/O cost for index pages
Example: Cost of $\sigma_{A=c}(R)$

- Example: $B(R) = 2000$, $T(R) = 100,000$, $V(R, A) = 20$
- Table scan: $B(R) = 2,000$ I/Os
- Index based selection:
  - If index is clustered: $B(R)/V(R, A) = 100$ I/Os
  - If index is unclustered: $T(R)/V(R, A) = 5,000$ I/Os

Cost of $\sigma_{A=c}(R)$ is

Lesson: Don’t build unclustered indexes when $V(R, A)$ is small!

Cost of Executing Operators (Focus on Joins)

Outline

- Join operator algorithms
  - One-pass algorithms (Sec. 15.2 and 15.3)
  - Index-based algorithms (Sec 15.6)
- Note about readings:
  - In class, we discuss only algorithms for joins
  - Other operators are easier: read the book

Hash Join

- Hash join: $R \bowtie S$
  - Scan $R$, build buckets in main memory
  - Then scan $S$ and join
  - Cost: $B(R) + B(S)$
  - One-pass algorithm when $B(R) \leq M$ (memory size)
    - more disk access also when $B(R) > M$

Join Algorithms

- Hash join
- Nested loop join
- Sort-merge join

Hash Join Example

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid, name, address)</td>
<td>(pid, provider, policy_nb)</td>
</tr>
</tbody>
</table>

Patient

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 'Bob'</td>
<td>'Blue' 123</td>
</tr>
<tr>
<td>2 'Ela'</td>
<td>'Prem' 432</td>
</tr>
<tr>
<td>3 'Jill'</td>
<td>'Prep' 343</td>
</tr>
<tr>
<td>4 'Joe'</td>
<td>'GrpH' 554</td>
</tr>
</tbody>
</table>

Insurance

Two tuples per page
Hash Join Example

Step 1: Scan Patient and build hash table in memory

Disk
Patient | Insurance
---|---
12 | 2 4 6 6
34 | 4 3 1 3
96 | 2 8
85 | 8 9

Memory M = 21 pages
Hash h: pid % 5

Input buffer

Step 2: Scan Insurance and probe into hash table

Disk
Patient | Insurance
---|---
12 | 2 4 6 6
34 | 4 3 1 3
96 | 2 8
85 | 8 9

Memory M = 21 pages
Hash h: pid % 5

Input buffer

Write to disk

Step 2: Scan Insurance and probe into hash table

Disk
Patient | Insurance
---|---
12 | 2 4 6 6
34 | 4 3 1 3
96 | 2 8
85 | 8 9

Memory M = 21 pages
Hash h: pid % 5

Input buffer

Output buffer

Keep going until read all of Insurance

Cost: B(R) + T(R) B(S)

Nested Loop Joins

- Tuple-based nested loop R \bowtie S
- R is the outer relation, S is the inner relation

```plaintext
for each tuple t1 in R do
  for each tuple t2 in S do
    if t1 and t2 join then output (t1, t2)
```

- Cost: B(R) + T(R) B(S)
- Multiple-pass because S is read many times
Block-at-a-time Refinement

for each block of tuples r in R do
  for each block of tuples s in S do
    for all pairs of tuples t₁ in r, t₂ in s
      if t₁ and t₂ join then output (t₁, t₂)
  end for
end for

• Cost: \( B(R) + B(R)B(S) \)

Page-at-a-time Refinement

for each group of \( M-1 \) pages r in R do
  for each page of tuples s in S do
    for all pairs of tuples t₁ in r, t₂ in s
      if t₁ and t₂ join then output (t₁, t₂)
  end for
end for

• Cost: \( B(R) + \frac{B(R)B(S)}{M-1} \)

Block-Nested-Loop Refinement

for each group of \( M-1 \) pages r in R do
  for each page of tuples s in S do
    for all pairs of tuples t₁ in r, t₂ in s
      if t₁ and t₂ join then output (t₁, t₂)
    end for
  end for
end for

• Cost: \( B(R) + \frac{B(R)B(S)}{M-1} \)
Sort-Merge Join

- Scan R and sort in main memory
- Scan S and sort in main memory
- Merge R and S

Cost: $B(R) + B(S)$

Typically, this is NOT a one pass algorithm

Sort-Merge Join Example

Step 1: Scan Patient and sort in memory

Step 2: Scan Insurance and sort in memory

Step 3: Merge Patient and Insurance

Using PK, so only one can match
Sort-Merge Join Example

Step 3: Merge Patient and Insurance

Memory M = 21 pages

Disk

Patient

Insurance

Disk

Patient

Insurance

Output buffer

Patient

Insurance

Output buffer

Step 3: Merge Patient and Insurance

Memory M = 21 pages

Index Nested Loop Join

\( R \bowtie S \)

- Assume S has an index on the join attribute
- Iterate over R, for each tuple, fetch corresponding tuple(s) from S

- Cost:
  - If index on S is clustered: \( B(R) + (R)B(S) / V(S, A) \)
  - If index on S is unclustered: \( B(R) + (R)T(S) / V(S, A) \)

Physical Query Plan 1

(On the fly)

\( \pi \text{name} \)

Selection and project on-the-fly

-> No additional cost.

Physical Query Plan 2

(On the fly)

\( \sigma \text{scity='Seattle'} \text{ and } \text{sstate='WA'} \)

Scan

Write to \( T_2 \)

\( \bowtie \)

Scan

Write to \( T_1 \)

\( \sigma \text{pno=2} \)

\( \sigma \text{scity='Seattle'} \text{ and } \text{sstate='WA'} \)

\( \sigma \text{pno=2} \)

Cost of Query Plans
Physical Query Plan 3

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

(On the fly) (c) \( \sigma_{\text{scity='Seattle' \& sstate='WA'}} \)

(Use hash index) (b) \( \sigma_{\text{pno=2}} \)

Supply

Assume: clustered

Physical Query Plan 3

Total cost

= 1 (a) + 4 (b) + 0 (c) + 0 (d)

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

(On the fly)

(Use hash index)

4 tuples

(Use hash index)

3 tuples

(Use hash index)

4 tuples

(On the fly) (a) \( \sigma_{\text{pno=2}} \)

Supplier

(Index on sno)

Clustering does not matter

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

(Use hash index)

4 tuples

(On the fly) (b) \( \sigma_{\text{pno=2}} \)

(Use hash index)

3 tuples

(Use hash index)

4 tuples