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

Parse & Check Query

Translate query string into internal representation

Check syntax, access control, table names, etc.

Logical plan → physical plan

Decide how best to answer query: query optimization

Query Execution

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 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
  /* σ_{c}(R) */
  – Selectivity = 1/V(R, A)

• A < c
  /* σ_{<c}(R) */
  – Selectivity = (c - Low(R, A))/(High(R, A) - Low(R, A))

• c1 < A < c2
  /* σ_{c1<A<c2}(R) */
  – Selectivity = (c2 – c1)/(High(R, A) - Low(R, A))

Example: Selectivity of σ_{A=c}(R)

T(R) = 100,000
V(R, A) = 20

How many records are returned by σ_{A=c}(R) = ?

Answer: X * 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*B(R)
  – Unclustered index X*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 $\sigma_{A=c}(R) = ?$

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

Join Algorithms

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

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 $B(R) > M$

Hash Join Example

<table>
<thead>
<tr>
<th>Patient(pid, name, address)</th>
<th>Insurance(pid, provider, policy_nb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 'Bob' 'Seattle' 123</td>
<td>2 'Blue' 343</td>
</tr>
<tr>
<td>2 'Ela' 'Everett' 432</td>
<td>4 'Prem' 554</td>
</tr>
<tr>
<td>3 'Jill' 'Kent'</td>
<td></td>
</tr>
<tr>
<td>4 'Joe' 'Seattle' 554</td>
<td></td>
</tr>
</tbody>
</table>

Two tuples per page
Hash Join Example

Step 1: Scan Patient and build hash table in memory

Hash h: pid % 5

Input buffer

Disk

Memory M = 21 pages

Step 2: Scan Insurance and probe into hash table

Input buffer

Output buffer

Write to disk

Disk

Output buffer

Patient Insurance

1 2 2 4 6 6
3 4 4 3 1 3
9 6 2 8 1 3
8 5 8 9

Hash Join Example

Step 2: Scan Insurance and probe into hash table

Disk

Memory M = 21 pages

Hash h: pid % 5

Step 2: Scan Insurance and probe into hash table

Input buffer

Output buffer

Write to disk

Disk

Output buffer

Patient Insurance

1 2 2 4 6 6
3 4 4 3 1 3
9 6 2 8 1 3
8 5 8 9

Hash Join Example

Step 2: Scan Insurance and probe into hash table

Disk

Memory M = 21 pages

Hash h: pid % 5

Step 2: Scan Insurance and probe into hash table

Input buffer

Output buffer

Write to disk

Disk

Output buffer

Patient Insurance

1 2 2 4 6 6
3 4 4 3 1 3
9 6 2 8 1 3
8 5 8 9

Hash Join Example

Step 2: Scan Insurance and probe into hash table

Disk

Memory M = 21 pages

Hash h: pid % 5

Step 2: Scan Insurance and probe into hash table

Input buffer

Output buffer

Write to disk

Disk

Output buffer

Patient Insurance

1 2 2 4 6 6
3 4 4 3 1 3
9 6 2 8 1 3
8 5 8 9

Nested Loop Joins

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

for each tuple $t_1$ in R do
for each tuple $t_2$ in S do
if $t_1$ and $t_2$ join then output $(t_1, t_2)$

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

- Cost: \( B(R) + B(R)B(S) \)

```plaintext
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)
```

Page-at-a-time Refinement

- Cost: \( B(R) + B(R)B(S)/(M-1) \)

```plaintext
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)
```

Block-Nested-Loop Refinement

- Cost: \( B(R) + B(R)B(S)/(M-1) \)

```plaintext
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)
```

What is the Cost?
Sort-Merge Join

- Scan R and sort in main memory
- Scan S and sort in main memory
- Merge R and S
- Cost: \( B(R) + B(S) \)
- One pass algorithm when \( B(S) + B(R) \leq M \)
- 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
Sort-Merge Join Example

Step 3: Merge Patient and Insurance

Memory M = 21 pages

Disk

Patient

Insurance

1 2 3 4 5 6 8 9
1 2 2 3 4 4 4 6
6 8 8 9
3 3
3
Output buffer

Step 3: Merge Patient and Insurance

Memory M = 21 pages

Disk

Patient

Insurance

1 2 3 4 5 6 8 9
1 2 2 3 4 4 4 6
6 8 8 9
3 3
Output buffer

Keep going until end of first relation

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) + T(R)B(S)/V(S, A)
– If index on S is unclustered: B(R) + T(R)T(S)/V(S, A)

Physical Query Plan 1

(On the fly) π sname
(On the fly) σ scity='Seattle' ∧ sstate='WA' ∧ pno=2
(Nested loop) π sname
Supplier (File scan)
Supply (File scan)

Total cost of plan is thus cost of join:
= B(Supplier) + B(Supplier) * B(Supply)
= 100 * 100
= 10,100 I/Os

Physical Query Plan 2

(On the fly) π sname
(Sort-merge join) σ scity='Seattle' ∧ sstate='WA' (a) → 3
(Scan write to T1) (b) → 2
(Scan write to T2) (c) → 1

Total cost = 204 I/Os

Cost of Query Plans
Physical Query Plan 3

(On the fly) (d) \( \pi_{\text{name}} \) (Use hash index) (c) \( \sigma_{\text{scity}=\text{Seattle} \land \text{sstate}=\text{WA}} \) (On the fly) (b) \( [\text{sn} = \text{sno}] \) (Index nested loop) (a) \( \sigma_{\text{pno}=2} \) (Index on pno) Assume: clustered

Total cost:
- 1 (a)
- 4 (b)
- 0 (c)
- 0 (d)

Total cost = 5 I/Os

4 tuples

Supplier

Clustering does not matter

Physical Query Plan 3

(On the fly) (d) \( \pi_{\text{name}} \) (Use hash index) (c) \( \sigma_{\text{scity}=\text{Seattle} \land \text{sstate}=\text{WA}} \) (On the fly) (b) \( [\text{sn} = \text{sno}] \) (Index nested loop) (a) \( \sigma_{\text{pno}=2} \) (Index on pno) Assume: clustered

Total cost:
- 1 (a)
- 4 (b)
- 0 (c)
- 0 (d)

Total cost = 5 I/Os

4 tuples

Supplier

Clustering does not matter