Lecture 23:
Monday, November 25, 2002

Outline
• Query execution: 15.1 – 15.5
• Query optimization: algebraic laws 16.2

Indexed Based Algorithms
• Recall that in a clustered index all tuples with the same value of the key are clustered on as few blocks as possible

• Note: book uses another term: “clustering index”. Difference is minor…

Index Based Selection
• Selection on equality: \( \sigma_{=v}(R) \)
• Clustered index on \( a \): cost \( B(R)/V(R,a) \)
• Unclustered index on \( a \): cost \( T(R)/V(R,a) \)

Index Based Selection
• Example:

<table>
<thead>
<tr>
<th>B(R)</th>
<th>T(R) = 100,000</th>
<th>V(R, a) = 20</th>
</tr>
</thead>
</table>

| cost of \( \sigma_{=a}(R) \) = ? |

• Table scan:
  - If \( R \) is clustered: \( B(R) = 2,000 \) I/Os
  - If \( R \) is unclustered: \( T(R) = 100,000 \) I/Os

• Index based selection:
  - If index is clustered: \( B(R)/V(R,a) = 100 \)
  - If index is unclustered: \( T(R)/V(R,a) = 5,000 \)

• Notice: when \( V(R,a) \) is small, then unclustered index is useless

Index Based Join
• \( R \bowtie S \)
• Assume \( S \) has an index on the join attribute
• Iterate over \( R \), for each tuple fetch corresponding tuple(s) from \( S \)
• Assume \( R \) is clustered. Cost:
  - If index is clustered: \( B(R) + T(R)B(S)/V(S,a) \)
  - If index is unclustered: \( B(R) + T(R)T(S)/V(S,a) \)
Index Based Join

- Assume both R and S have a sorted index (B+ tree) on the join attribute
- Then perform a merge join
  - called zig-zag join
- Cost: B(R) + B(S)

Example

\[ \text{Product}(\text{pname}, \text{maker}), \text{Company}(\text{cname}, \text{city}) \]

Clustered index: \( \text{Product}. \text{pname}, \text{Company}. \text{cname} \)
Unclustered index: \( \text{Product}. \text{maker}, \text{Company}. \text{city} \)

Select \( \text{Product}. \text{pname} \)  
From \( \text{Product}, \text{Company} \)  
Where \( \text{Product}. \text{maker} = \text{Company}. \text{cname} \)  
and \( \text{Company}. \text{city} = \text{“Seattle”} \)

Logical Plan:

\[
\begin{align*}
\sigma_{\text{city} = \text{“Seattle”}} & \quad \text{Product} \\
\sigma_{\text{maker} = \text{pname}} & \quad \text{Company}
\end{align*}
\]

Plan 1:
- Index-based selection: \( T(\text{Company}) \div V(\text{Company}, \text{city}) \)
- Index-based join: \( \times T(\text{Product}) \div V(\text{Product}, \text{maker}) \)

Plan 2:
- Table scan and selection on Company: \( B(\text{Company}) \)
- Plan 2a: scan and sort: \( 3B(\text{Product}) \)
- Plan 2b: index-scan: \( T(\text{Product}) \)
- Merge-join: their sum

Example

<table>
<thead>
<tr>
<th>( T(\text{Company}) )</th>
<th>( 5,000 )</th>
<th>( B(\text{Company}) )</th>
<th>( 500 )</th>
<th>( M )</th>
<th>( 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T(\text{Product}) )</td>
<td>( 100,000 )</td>
<td>( B(\text{Product}) )</td>
<td>( 1,000 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Case 1: \( V(\text{Company}, \text{city}) = T(\text{Company}) \)  
  \( V(\text{Product}, \text{maker}) = T(\text{Product}) \)
- Case 2: \( V(\text{Company}, \text{city}) < T(\text{Company}) \)  
  \( V(\text{Product}, \text{maker}) = T(\text{Product}) \)
- Case 3: \( V(\text{Company}, \text{city}) < T(\text{Company}) \)  
  \( V(\text{Product}, \text{maker}) < T(\text{Product}) \)
Which Plan is Best?

Plan 1: $T(\text{Company}) \times V(\text{Company,city}) \times T(\text{Product}) \times V(\text{Product,make})$

Plan 2a: $T(\text{Company}) \times T(\text{Product})$

Plan 2b: $T(\text{Company}) \times T(\text{Product})$

Case 1:

Case 2:

Case 3:

Optimization

- Chapter 16
- At the heart of the database engine
- Step 1: convert the SQL query to some logical plan
- Step 2: find a better logical plan, find an associated physical plan

Converting from SQL to Logical Plans

Select $a_1, \ldots, a_n$
From $R_1, \ldots, R_k$
Where $C$

$\Pi_{a_1, a_2, \ldots, a_n}(T_1 \bowtie T_2 \bowtie \cdots \bowtie T_k)$

Select $a_1, \ldots, a_n$
From $R_1, \ldots, R_k$
Where $C$
Group by $b_1, \ldots, b_l$

Converting Nested Queries

Select distinct product.name
From product
Where product.make in (Select company.name
From company
where company.city=“Seattle”)

Select distinct product.name
From product, company
Where product.make = company.name AND company.city=“Seattle”

Converting Nested Queries

Select distinct x.name, x.make
From product x
Where x.color= “blue”
AND x.price >= ALL (Select y.price
From product y
Where x.make = y.make
AND y.color=“blue”)

How do we convert this one to logical plan?
Converting Nested Queries

This one becomes a SFW query:

```sql
SELECT distinct x.name, x.maker
FROM product x, product y
WHERE x.color = "blue" AND x.maker = y.maker
AND y.color = "blue" AND x.price < y.price
```

This returns exactly the products we DON’T want, so…

Optimization: the Logical Query Plan

- Now we have one logical plan
- Algebraic laws:
  - foundation for every optimization
- Two approaches to optimizations:
  - Heuristics: apply laws that seem to result in cheaper plans
  - Cost based: estimate size and cost of intermediate results, search systematically for best plan
- All modern database optimizers use a cost-based optimizer
  - Why?

The three components of an optimizer

- Algebraic laws
- An optimization algorithm
- A cost estimator

Algebraic Laws

- Commutative and Associative Laws
  - $R \cup S = S \cup R$, $R \cup (S \cup T) = (R \cup S) \cup T$
  - $R \cap S = S \cap R$, $R \cap (S \cap T) = (R \cap S) \cap T$
  - $R \bowtie S = R \bowtie R$, $R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$
- Distributive Laws
  - $R \bowtie (S \cup T) = (R \bowtie S) \cup (R \bowtie T)$

Algebraic Laws

- Laws involving selection:
  - $\sigma_{C \land R}(R) = \sigma_{R}(\sigma_{C}(R)) = \sigma_{C}(R) \cap \sigma_{C}(R)$
  - $\sigma_{C \lor R}(R) = \sigma_{R}(\sigma_{C}(R)) = \sigma_{C}(R) \cup \sigma_{C}(R)$
  - $\sigma_{C}(R \bowtie S) = \sigma_{C}(R) \bowtie S$
    - When $C$ involves only attributes of $R$
  - $\sigma_{C}(R - S) = \sigma_{C}(R) - S$
  - $\sigma_{C}(R \cup S) = \sigma_{C}(R) \cup \sigma_{C}(S)$
  - $\sigma_{C}(R \cap S) = \sigma_{C}(R) \cap S$
Algebraic Laws

- Example: $R(A, B, C, D), S(E, F, G)$
  - $\sigma_{p \land q} (R \bowtie_{D=E} S) =$
  - $\sigma_{A=5 \land G=3} (R \bowtie_{D=E} S) =$

Algebraic Laws

- Laws involving projections
  - $\Pi_{M} (R \bowtie S) = \Pi_{M} (\Pi_{P} (R) \bowtie \Pi_{Q} (S))$
  - Where $N, P, Q$ are appropriate subsets of attributes of $R$
  - $\Pi_{M} (\Pi_{q} (R)) = \Pi_{M \setminus S} (R)$
- Example $R(A,B,C,D), S(E, F, G)$
  - $\Pi_{A,B,C} (R \bowtie S) = \Pi_{A} (\Pi_{B} (R) \bowtie \Pi_{C} (S))$

Algebraic Laws

Laws involving grouping and aggregation:
- $\delta_{\text{agg}} (\delta_{\text{agg}} (R)) = \gamma_{\text{agg}} (\gamma_{\text{agg}} (R))$
- $\gamma_{\text{agg}} (\delta_{\text{agg}} (R)) = \gamma_{\text{agg}} (\delta_{\text{agg}} (R))$ if agg is “duplicate insensitive”
  - Which of the following are “duplicate insensitive”?
    - sum, count, avg, min, max
- $\gamma_{\text{agg}} (\gamma_{\text{agg}} (R(A,B) \bowtie_{B=C} S(C,D)))$
  - Why is this true?
  - Why would we do it?