Lecture 24:
Query Execution

Monday, November 27, 2006
Outline

• Query optimization: algebraic laws 16.2
Example

`Product(pname, maker), Company(cname, city)`

```sql
Select Product.pname
From   Product, Company
Where  Product.maker=Company.cname
    and  Company.city = "Seattle"
```

• How do we execute this query?
Example

Product(pname, maker), Company(cname, city)

Assume:

Clustered index:  Product.pname, Company.cname
Unclustered index:  Product.maker, Company.city
Logical Plan:
Physical plan 1:

Index-based selection

\[ \sigma_{\text{city} = \text{"Seattle"}} \]

\[ \bowtie \text{cname} = \text{maker} \]

Company (cname, city)  Product (pname, maker)

Index-based join
Physical plans 2a and 2b:

Which one is better??

\[ \sigma_{\text{city} = \text{“Seattle”}} \]

Product (pname,maker) × Company (cname,city)

- Scan and sort (2a)
- Index scan (2b)

Merge-join

maker = cname
Physical plan 1:

\[ \sigma_{\text{city}=\text{"Seattle"}} \]

Company (cname,city) \hspace{2cm} Product (pname,maker)

\[ \times \frac{\text{T(Company)}}{\text{V(Company, city)}} \times \frac{\text{T(Product)}}{\text{V(Product, maker)}} \]

Total cost:

\[ \frac{\text{T(Company)}}{\text{V(Company, city)}} \times \frac{\text{T(Product)}}{\text{V(Product, maker)}} \]
Total cost:
(2a): \(3B(\text{Product}) + B(\text{Company})\)
(2b): \(T(\text{Product}) + B(\text{Company})\)

Physical plans 2a and 2b:

**Physical plans 2a:**
- **Table-scan**
- **Scan and sort (2a)**
- **index scan (2b)**

**Physical plans 2b:**
- **Merge-join**
- \(\sigma_{city=\text{“Seattle”}}\)
- **Product (pname,maker)(cname,city)**
- **B(Company)**

No extra cost (why?)
Plan 1: \( T(\text{Company})/V(\text{Company}, \text{city}) \times T(\text{Product})/V(\text{Product}, \text{maker}) \)

Plan 2a: \( B(\text{Company}) + 3B(\text{Product}) \)

Plan 2b: \( B(\text{Company}) + T(\text{Product}) \)

Which one is better ??

It depends on the data !!
Example

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

We may assume $V(\text{Product}, \text{maker}) \approx T(\text{Company})$ (why?)

- Case 1: $V(\text{Company, city}) \approx T(\text{Company})$
  
  $V(\text{Company, city}) = 2,000$

- Case 2: $V(\text{Company, city}) \ll T(\text{Company})$
  
  $V(\text{Company, city}) = 20$
Which Plan is Best?

| Plan 1: T(Company)/V(Company,city) × T(Product)/V(Product,maker) |
| Plan 2a: B(Company) + 3B(Product) |
| Plan 2b: B(Company) + T(Product) |

Case 1:

Case 2:
Lessons

- Need to consider several physical plan
  - even for one, simple logical plan
- No magic “best” plan: depends on the data
- In order to make the right choice
  - need to have statistics over the data
  - the B’s, the T’s, the V’s
Query Optimization

• Have a SQL query Q

• Create a plan P

• Find equivalent plans $P = P' = P'' = \ldots$

• Choose the “cheapest”.

HOW ??
Logical Query Plan

```
SELECT P.buyer
FROM Purchase P, Person Q
WHERE P.buyer=Q.name AND
  P.city='seattle' AND
  Q.phone > '5430000'
```

In class:
find a “better” plan $P'$

$P = \sigma_{\text{City='seattle'} \land \text{phone}>'5430000'}(\Pi_{\text{buyer}}(\text{Purchase}))$

$\text{Person(name, phone)}$

$\text{Purchase(buyer, city)}$
Logical Query Plan

Q =

```
SELECT city, sum(quantity)
FROM sales
GROUP BY city
HAVING sum(quantity) < 100
```

P =

```
T2(city,p)
σ p < 100
T1(city,p)
γ city, sum(quantity) → p
sales(product, city, quantity)
```

In class: find a “better” plan P’
The three components of an optimizer

We need three things in an optimizer:

• Algebraic laws
• An optimization algorithm
• A cost estimator
Algebraic Laws

• Commutative and Associative Laws
  \[ R \cup S = S \cup R, \quad R \cup (S \cup T) = (R \cup S) \cup T \]
  \[ R \times S = S \times R, \quad R \times (S \times T) = (R \times S) \times T \]
  \[ R \times S = S \times R, \quad R \times (S \times T) = (R \times S) \times T \]

• Distributive Laws
  \[ R \times (S \cup T) = (R \times S) \cup (R \times T) \]
Algebraic Laws

• Laws involving selection:
  \[ \sigma_{C \text{ AND } C'}(R) = \sigma_C(\sigma_{C'}(R)) = \sigma_C(R) \cap \sigma_{C'}(R) \]
  \[ \sigma_{C \text{ OR } C'}(R) = \sigma_C(R) \cup \sigma_{C'}(R) \]
  \[ \sigma_C(R \mid \times \mid S) = \sigma_C(R) \mid \times \mid S \]

• When \( C \) involves only attributes of \( R \)
  \[ \sigma_C(R - S) = \sigma_C(R) - S \]
  \[ \sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S) \]
  \[ \sigma_C(R \mid \times \mid S) = \sigma_C(R) \mid \times \mid S \]
Algebraic Laws

• Example: \( R(A, B, C, D), S(E, F, G) \)

\[
\sigma_{F=3} (R \mid \times \mid_{D=E} S) = \quad ?
\]

\[
\sigma_{A=5 \text{ AND } G=9} (R \mid \times \mid_{D=E} S) = \quad ?
\]
Algebraic Laws

• Laws involving projections

\[ \Pi_M(R \times S) = \Pi_M(\Pi_P(R) \times \Pi_Q(S)) \]
\[ \Pi_M(\Pi_N(R)) = \Pi_{M,N}(R) \]

• Example \( R(A,B,C,D), S(E, F, G) \)

\[ \Pi_{A,B,G}(R \times_{D=E} S) = \Pi (\Pi?(R) \times_{D=E} \Pi?(S)) \]
Algebraic Laws

• Laws involving grouping and aggregation:
  \[ \delta(\gamma_{A, \text{agg}(B)}(R)) = \gamma_{A, \text{agg}(B)}(R) \]
  \[ \gamma_{A, \text{agg}(B)}(\delta(R)) = \gamma_{A, \text{agg}(B)}(R) \] if agg is “duplicate insensitive”

• Which of the following are “duplicate insensitive”?
  sum, count, avg, min, max

\[ \gamma_{A, \text{agg}(D)}(R(A,B) \times_{B=C} S(C,D)) = \]
\[ \gamma_{A, \text{agg}(D)}(R(A,B) \times_{B=C} (\gamma_{C, \text{agg}(D)}S(C,D))) \]
Optimizations Based on Semijoins

THIS IS ADVANCED STUFF; NOT ON THE FINAL

• $\mathbf{R} \bowtie \mathbf{S} = \Pi_{A_1, \ldots, A_n} (\mathbf{R} \bowtie \mathbf{S})$

• Where the schemas are:
  – Input: $\mathbf{R}(A_1, \ldots, A_n)$, $\mathbf{S}(B_1, \ldots, B_m)$
  – Output: $\mathbf{T}(A_1, \ldots, A_n)$
Optimizations Based on Semijoins

Semijoins: a bit of theory (see [AHV])

• Given a query:

\[
Q ::= \Pi (\sigma (R_1 \mid x \mid R_2 \mid x \mid \ldots \mid x \mid R_n))
\]

• A full reducer for Q is a program:

\[
\begin{align*}
R_{i1} &:= R_{i1} \bowtie R_{j1} \\
R_{i2} &:= R_{i2} \bowtie R_{j2} \\
\ldots &\ldots \\
R_{ip} &:= R_{ip} \bowtie R_{jp}
\end{align*}
\]

• Such that no dangling tuples remain in any relation
Optimizations Based on Semijoins

• Example: \( Q(A,E) : - R1(A,B) \mid x \mid R2(B,C) \mid x \mid R3(C,D,E) \)

• A full reducer is:
  
  \[
  \begin{align*}
  R2(B,C) & := R2(B,C) \mid x \ R1(A,B) \\
  R3(C,D,E) & := R3(C,D,E) \mid x \ R2(B,C) \\
  R2(B,C) & := R2(B,C) \mid x \ R3(C,D,E) \\
  R1(A,B) & := R1(A,B) \mid x \ R2(B,C)
  \end{align*}
  \]

The new tables have only the tuples necessary to compute \( Q(E) \)
Optimizations Based on Semijoins

• Example:

\[
Q(E) :- R1(A, B) \ |x| \ R2(B, C) \ |x| \ R3(A, C, E)
\]

• Doesn’t have a full reducer (we can reduce forever)

**Theorem** a query has a full reducer iff it is “acyclic”
Optimizations Based on Semijoins

• Semijoins in [Chaudhuri’98]

CREATE VIEW DepAvgSal As (  
    SELECT E.did, Avg(E.Sal) AS avgsal  
    FROM Emp E  
    GROUP BY E.did)

SELECT E.eid, E.sal  
FROM Emp E, Dept D, DepAvgSal V  
WHERE E.did = D.did AND E.did = V.did  
   AND E.age < 30 AND D.budget > 100k  
   AND E.sal > V.avgsal
Optimizations Based on Semijoins

• First idea:

CREATE VIEW LimitedAvgSal As (  
SELECT E.did, Avg(E.Sal) AS avgsal  
FROM Emp E, Dept D  
WHERE E.did = D.did AND D.buget > 100k  
GROUP BY E.did)

SELECT E.eid, E.sal  
FROM Emp E, Dept D, LimitedAvgSal V  
WHERE E.did = D.did AND E.did = V.did  
   AND E.age < 30 AND D.budget > 100k  
   AND E.sal > V.avgsal
Optimizations Based on Semijoins

- Better: full reducer

```
CREATE VIEW PartialResult AS
 (SELECT E.id, E.sal, E.did
  FROM Emp E, Dept D
  WHERE E.did=D.did AND E.age < 30
  AND D.budget > 100k)

CREATE VIEW Filter AS
 (SELECT DISTINCT P.did FROM PartialResult P)

CREATE VIEW LimitedAvgSal AS
 (SELECT E.did, Avg(E.Sal) AS avgsal
  FROM Emp E, Filter F
  WHERE E.did = F.did GROUP BY E.did)
```
Optimizations Based on Semijoins

```
SELECT P.eid, P.sal
FROM PartialResult P, LimitedDepAvgSal V
WHERE P.did = V.did AND P.sal > V.avgsal
```