Lectures 9: Relational Algebra (part 2) and Query Evaluation (Ch. 5.2 & 16.3 (skim 16.3.2))
Announcements

• Should have used SQL / Azure now
  – let us know if you had any setup problems

• WQ3 is due on Sunday

• HW3 is due one week from Tuesday

• HW1 grades *should* be posted tonight
Join Summary

• **Theta-join**: $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$
  – Join of R and S with a join condition $\theta$
  – Cross-product followed by selection $\theta$

• **Equijoin**: $R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$
  – Join condition $\theta$ consists only of equalities

• **Natural join**: $R \bowtie S = \pi_A (\sigma_{\theta}(R \times S))$
  – Equijoin
  – Equality on all fields with same name in R and in S
  – Projection $\pi_A$ drops all redundant attributes
So Which Join Is It?

When we write $R \bowtie S$ we usually mean an equijoin, but we often omit the equality predicate when it is clear from the context.
More Joins

• **Outer join**
  - Include tuples with no matches in the output
  - Use NULL values for missing attributes
  - Does not eliminate duplicate columns

• **Variants**
  - Left outer join
  - Right outer join
  - Full outer join
### Outer Join Example

#### AnonPatient P

<table>
<thead>
<tr>
<th>age</th>
<th>zip</th>
<th>disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>98125</td>
<td>heart</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
<td>flu</td>
</tr>
<tr>
<td>33</td>
<td>98120</td>
<td>lung</td>
</tr>
</tbody>
</table>

#### AnonJob J

<table>
<thead>
<tr>
<th>job</th>
<th>age</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>lawyer</td>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>cashier</td>
<td>20</td>
<td>98120</td>
</tr>
</tbody>
</table>

#### P ⨝ J

<table>
<thead>
<tr>
<th>P.age</th>
<th>P.zip</th>
<th>disease</th>
<th>job</th>
<th>J.age</th>
<th>J.zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>98125</td>
<td>heart</td>
<td>lawyer</td>
<td>54</td>
<td>98125</td>
</tr>
<tr>
<td>20</td>
<td>98120</td>
<td>flu</td>
<td>cashier</td>
<td>20</td>
<td>98120</td>
</tr>
<tr>
<td>33</td>
<td>98120</td>
<td>lung</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>
More Examples

Supplier(sno, sname, scity, sstate)
Part(pno, pname, psize, pcolor)
Supply(sno, pno, qty, price)

Name of supplier of parts with size greater than 10
\[ \pi_{\text{name}} (\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{\text{psize}>10} (\text{Part})) ) \]

Name of supplier of red parts or parts with size greater than 10
\[ \pi_{\text{name}} (\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{\text{psize}>10} (\text{Part}) \cup \sigma_{\text{pcolor}='\text{red}'} (\text{Part}) ) ) \]
Query Evaluation Steps

1. Parse & Check Query
   - Translate query string into internal representation
   - Check syntax, access control, table names, etc.
2. Decide how best to answer query: query optimization
3. Query Execution
4. Return Results
Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

From SQL to RA

SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid and
x.price > 100 and
z.city = 'Seattle'

\[ \delta \]

\[ \pi \]

\[ \sigma \]

\[ x \]

\[ x \]

\[ x \]

\[ x \]

Customer

Product

Purchase
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid and
  x.price > 100 and
  z.city = 'Seattle'

Can you think of another plan?
SELECT DISTINCT x.name, z.name
FROM Product x, Purchase y, Customer z
WHERE x.pid = y.pid and y.cid = z.cid and
  x.price > 100 and
  z.city = 'Seattle'

Push selections down the query plan!

Query optimization: find an equivalent optimal plan

Can you think of another plan?
Extended RA: Operators on Bags

- Duplicate elimination $\delta$
- Grouping & aggregation $\gamma$
- Sorting $\tau$
Logical Query Plan

\[
\begin{align*}
\text{SELECT} & \quad \text{city, count(*)} \\
\text{FROM} & \quad \text{sales} \\
\text{GROUP BY} & \quad \text{city} \\
\text{HAVING} & \quad \text{sum(price) > 100} \\
\end{align*}
\]

\[T1, T2, T3 \quad \text{= temporary tables} \quad \text{sales(product, city, price)}\]

\[\begin{align*}
T1(\text{city, p, c}) \\
\pi_{\text{city, c}} \\
\sigma_{\text{p > 100}} \\
\gamma_{\text{city, sum(price) \rightarrow p, count(*) \rightarrow c}} \\
\end{align*}\]
Typical Plan for Block (1/2)

\[
\text{SELECT fields} \\
\text{FROM } R, S, \ldots \\
\text{WHERE condition}
\]

\[
\text{SELECT-PROJECT-JOIN Query}
\]
Typical Plan for Block (2/2)

\[ \pi \text{ fields} \]
\[ \sigma \text{ having condition} \]
\[ \gamma \text{ fields, sum/count/min/max(fields)} \]
\[ \sigma \text{ where condition} \]
join condition

... ...

SELECT fields
FROM R, S, ...
WHERE condition
GROUP BY fields
HAVING condition
How about Subqueries?

```sql
SELECT Q.sno
FROM Supplier Q
WHERE Q.sstate = 'WA'
    and not exists
        (SELECT *
         FROM Supply P
         WHERE P.sno = Q.sno
            and P.price > 100)
```

Correlation!
How about Subqueries?

SELECT Q.sno
FROM Supplier Q
WHERE Q.sstate = 'WA'
   and not exists
      (SELECT *
       FROM Supply P
       WHERE P.sno = Q.sno
       and P.price > 100)

De-Correlation

SELECT Q.sno
FROM Supplier Q
WHERE Q.sstate = 'WA'
   and Q.sno not in
      (SELECT P.sno
       FROM Supply P
       WHERE P.price > 100)
How about Subqueries?

Un-nesting

```
(SELECT Q.sno
 FROM Supplier Q
 WHERE Q.sstate = 'WA')
 EXCEPT
(SELECT P.sno
 FROM Supply P
 WHERE P.price > 100)
 EXCEPT = set difference
```

```
SELECT Q.sno
 FROM Supplier Q
 WHERE Q.sstate = 'WA'
 AND Q.sno not in
(SELECT P.sno
 FROM Supply P
 WHERE P.price > 100)
```
How about Subqueries?

Finally...

\[
\begin{align*}
\text{(SELECT } & \text{ Q.sno} \\
\text{FROM } & \text{ Supplier Q} \\
\text{WHERE } & \text{ Q.sstate = ‘WA’) EXCEPT} \\
\text{(SELECT P.sno} \\
\text{FROM } & \text{ Supply P} \\
\text{WHERE } & \text{ P.price > 100)}
\end{align*}
\]
From Logical Plans to Physical Plans
Physical Operators

Each of the logical operators may have one or more implementations = physical operators

Will discuss several basic physical operators, with a focus on join
Main Memory Algorithms

Logical operator:
\[
\text{Product}(\text{pid}, \text{name}, \text{price}) \bowtie_{\text{pid} = \text{pid}} \text{Purchase}(\text{pid}, \text{cid}, \text{store})
\]

Propose three physical operators for the join, assuming the tables are in main memory:

1. Nested Loop Join \( O(??) \)
2. Merge join \( O(??) \)
3. Hash join \( O(??) \)

(note that \text{pid} is a key)
Main Memory Algorithms

Logical operator:

\[
\text{Product}(\text{pid}, \text{name}, \text{price}) \Join_{\text{pid} = \text{pid}} \text{Purchase}(\text{pid}, \text{cid}, \text{store})
\]

Propose three physical operators for the join, assuming the tables are in main memory:

1. Nested Loop Join \(O(n^2)\)
2. Merge join \(O(??)\)
3. Hash join \(O(??)\)
Main Memory Algorithms

Logical operator:

\[ \text{Product}(\text{pid, name, price}) \bowtie_{\text{pid}=\text{pid}} \text{Purchase}(\text{pid, cid, store}) \]

Propose three physical operators for the join, assuming the tables are in main memory:

1. Nested Loop Join \( O(n^2) \)
2. Merge join \( O(n \log n) \)
3. Hash join \( O(???) \)

sort both \( \rightarrow O(n \log n) \)
merge \( \rightarrow O(n) \)
Main Memory Algorithms

Logical operator:

\[ \text{Product}(\text{pid, name, price}) \Join_{\text{pid} = \text{pid}} \text{Purchase}(\text{pid, cid, store}) \]

Propose three physical operators for the join, assuming the tables are in main memory:

1. Nested Loop Join \(\mathcal{O}(n^2)\)
2. Merge join \(\mathcal{O}(n \log n)\)
3. Hash join \(\mathcal{O}(n) \ldots \mathcal{O}(n^2)\)

- add \(n\) to hash – \(\mathcal{O}(n)\)?
- lookup \(n\) in hash – \(\mathcal{O}(n)\)?
BRIEF Review of Hash Tables

Separate chaining:

A (naïve) hash function:

\[ h(x) = x \mod 10 \]

Operations:

- find(103) = ??
- insert(488) = ??
BRIEF Review of Hash Tables

• insert(k, v) = inserts a key k with value v

• Many values for one key
  – Hence, duplicate k’s are OK

• find(k) = returns the list of all values v associated to the key k
Query Evaluation Steps Review

- Parse & Rewrite Query
- Select Logical Plan
- Select Physical Plan
- Query Execution
- Disk

SQL query

Query optimization

Logical plan

Physical plan
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Relational Algebra

<table>
<thead>
<tr>
<th>SELECT  sname</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROM Supplier x, Supply y</td>
</tr>
<tr>
<td>WHERE x.sid = y.sid</td>
</tr>
<tr>
<td>and y.pno = 2</td>
</tr>
<tr>
<td>and x.scity = 'Seattle'</td>
</tr>
<tr>
<td>and x.sstate = 'WA'</td>
</tr>
</tbody>
</table>

Give a relational algebra expression for this query
Relational Algebra

\[ \pi_{\text{sname}}(\sigma_{\text{scity}= \text{Seattle} \land \text{sstate}= \text{WA} \land \text{pno}=2}(\text{Supplier} \bowtie_{\text{sid}=\text{sid}} \text{Supply})) \]
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Relational Algebra

SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'

Relational algebra expression is also called the “logical query plan”
Physical Query Plan 1

A physical query plan is a logical query plan annotated with physical implementation details.

$$\text{SELECT sname}$$
$$\text{FROM Supplier x, Supply y}$$
$$\text{WHERE x.sid = y.sid}$$
$$\text{and y.pno = 2}$$
$$\text{and x.scity = 'Seattle'}$$
$$\text{and x.sstate = 'WA'}$$

Supplier($$\text{sid, sname, scity, sstate}$$)
Supply($$\text{sid, pno, quantity}$$)
Physical Query Plan 2

Same logical query plan
Different physical plan

\[
\begin{align*}
\text{SELECT} & \quad \text{sname} \\
\text{FROM} & \quad \text{Supplier} \ x, \ \text{Supply} \ y \\
\text{WHERE} & \quad x.\text{sid} = y.\text{sid} \\
& \quad \text{and} \ y.\text{pno} = 2 \\
& \quad \text{and} \ x.\text{scity} = \text{‘Seattle’} \\
& \quad \text{and} \ x.\text{sstate} = \text{‘WA’}
\end{align*}
\]
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Physical Query Plan 3

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

(Sort-merge join) \( \sigma_{\text{sid} = \text{sid}} \)

(Scan & write to T1) \( \sigma_{\text{scity}=\text{Seattle} \land \text{sstate}=\text{WA}} \)

(Scan & write to T2) \( \sigma_{\text{pno}=2} \)

Different but equivalent logical query plan; different physical plan

SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
Query Optimization Problem

• For each SQL query… many logical plans

• For each logical plan… many physical plans

• How do find a fast physical plan?
  – Will discuss in a few lectures