Database Systems
CSE 414

Lectures 14: Relational Algebra (part 2) and Query Evaluation (Ch. 5.2 & 16.3 (skim 16.3.2))

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**: $R \bowtie S = \sigma_\theta (R \times S)$
  - 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

More Examples
- **Name of supplier of parts with size greater than 10**
  $\pi_{sname} (\sigma_{psize > 10} (Part))$
- **Name of supplier of red parts or parts with size greater than 10**
  $\pi_{sname} (\sigma_{pcolor = 'red'} (Part) \cup \sigma_{psize > 10} (Part))$

### AnonPatient P
<table>
<thead>
<tr>
<th>age</th>
<th>zip</th>
<th>disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</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>

### AnonJob J


Query Evaluation Steps

- Parse & Check Query
  - Translate query string into internal representation
  - Check syntax, access control, table names, etc.

- Decide how best to answer query: query optimization

- Logical plan → physical plan

- Query Execution

- Return Results

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'
```

Extended RA: Operators on Bags

- Duplicate elimination \( \delta \)
- Grouping & aggregation \( \gamma \)
- Sorting \( \tau \)

Logical Query Plan

```
SELECT city, count(*)
FROM sales
GROUP BY city
HAVING sum(price) > 100
```

Product(pid, name, price)
Purchase(pid, cid, store)
Customer(cid, name, city)

T1(city, p, c)
T2(city, p, c)
T3(city, c)
T1, T2, T3 = temporary tables
Typical Plan for Block (1/2)

```
SELECT fields
FROM R, S, ...
WHERE condition
```

Typical Plan for Block (2/2)

```
SELECT fields
FROM R, S, ...
WHERE condition
GROUP BY fields
HAVING condition
```

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)
```

How about Subqueries?

```
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)
EXCEPT
(SELECT Q.sno
FROM Supplier Q
WHERE Q.sstate = 'WA')
```

Finally...

```
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)
```
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:
Product(pid, name, price) \(\bowtie\) pid=pid Purchase(pid, cid, 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 pid is a key)
**BRIEF Review of Hash Tables**

Separate chaining:

A (naïve) hash function:

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

Operations:

- \( f(103) = ?? \)
- \( i(488) = ?? \)

Duplicates OK, WHY ??

Value can be different tuple ids.

- \( f(k) = \text{returns the list of all values } v \text{ associated to the key } k \)

**Query Evaluation Steps Review**

SQL query → Parse & Rewrite Query → Select Logical Plan → Logical plan → Select Physical Plan → Physical plan → Query Execution → Disk

**Relational Algebra**

Give a relational algebra expression for this query:

\[
\pi_{\text{name}}(\sigma_{\text{scity} = \text{'Seattle'}} \land \text{state} = \text{'WA'} \land \text{pno} = 2 \ (\text{Supplier} \bowtie \text{Supply}))
\]

**Relational Algebra**

\[
\pi_{\text{name}}(\sigma_{\text{scity} = \text{'Seattle'}} \land \text{state} = \text{'WA'} \land \text{pno} = 2 \ (\text{Supplier} \bowtie \text{Supply}))
\]

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

\[
\sigma \text{ scity}=\text{'Seattle'} \land \text{sstate}=\text{'WA'} \land \text{pno}=2
\]

\[
\pi \text{ sname}
\]

(On the fly)

(On the fly)

(Nested loop)

Supplier (File scan)

Supply (File scan)

Physical Query Plan 2

\[
\sigma \text{ scity}=\text{'Seattle'} \land \text{sstate}=\text{'WA'} \land \text{pno}=2
\]

\[
\pi \text{ sname}
\]

(On the fly)

(Hash join)

Supplier (File scan)

Supply (File scan)

Physical Query Plan 3

\[
\sigma \text{ scity}=\text{'Seattle'} \land \text{sstate}=\text{'WA'}
\]

\[
\sigma \text{ pno}=2
\]

(Scan & write to T1)

(Scan & write to T2)

Supplier (File scan)

Supply (File scan)

Query Optimization Problem

- For each SQL query... many logical plans
- For each logical plan... many physical plans
- How to find a fast physical plan?
  - Will discuss in a few lectures