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**: $R \bowtie_\theta S = \sigma_\theta(R \times S)$
  – 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>

#### Join Operation

The outer join of Patient and Job tables on the common column `age` is denoted as `P ⋈ J`. The resulting table includes all records from both tables, with added null values for unmatched records:

<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
$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10} (\text{Part}))$

Name of supplier of red parts or parts with size greater than 10
$\pi_{sname}(\text{Supplier} \bowtie \text{Supply} \bowtie (\sigma_{psize>10} (\text{Part}) \cup \sigma_{pcolor='red'} (\text{Part})))$
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

CSE 414 - Fall 2017
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'

Customer

Product

Purchase

CSE 414 - Fall 2017
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'

Can you think of another plan?
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'

Can you think of another plan?

Push selections down the query plan!

Query optimization: find an equivalent optimal plan
Extended RA: Operators on Bags

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

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

\[
T_1(\text{city}, \text{p}, \text{c}) \\
\begin{align*}
\pi & \quad \text{city, c} \\
\sigma & \quad \text{p > 100} \\
\text{sales(product, city, price)} & \rightarrow \\
\gamma & \quad \text{city, sum(price)} \rightarrow \text{p}, \text{count(*)} \rightarrow \text{c}
\end{align*}
\]

T_1, T_2, T_3 = temporary tables

CSE 414 - Fall 2017
Typical Plan for Block (1/2)

\[
\begin{align*}
\text{SELECT fields} & \quad \text{FROM } R, S, \ldots \\
\text{WHERE condition} & \\
\{ & \text{SELECT-PROJECT-JOIN Query} \\
\end{align*}
\]

CSE 414 - Fall 2017
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

\[ \ldots \]

\[ \ldots \]

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)
```

Supplier(sno, sname, scity, sstate)
Part(pno, pname, psize, pcolor)
Supply(sno, pno, price)
How about Subqueries?

\[
\begin{align*}
(\text{SELECT} & \quad Q\.sno \\
\text{FROM} & \quad \text{Supplier} \ Q \\
\text{WHERE} & \quad Q\.sstate = \text{‘WA’}) \\
\text{EXCEPT} & \\
(\text{SELECT} & \quad P\.sno \\
\text{FROM} & \quad \text{Supply} \ P \\
\text{WHERE} & \quad P\.price > 100) \\
\text{EXCEPT} & = \text{set difference}
\end{align*}
\]

Un-nesting

\[
\begin{align*}
\text{SELECT} & \quad Q\.sno \\
\text{FROM} & \quad \text{Supplier} \ Q \\
\text{WHERE} & \quad Q\.sstate = \text{‘WA’} \\
\text{and} & \quad Q\.sno \not \in \\
(\text{SELECT} & \quad P\.sno \\
\text{FROM} & \quad \text{Supply} \ P \\
\text{WHERE} & \quad P\.price > 100)
\end{align*}
\]

Supplier(sno, sname, scity, sstate)
Part(pno, pname, psize, pcolor)
Supply(sno, pno, price)
How about Subqueries?

\[
\begin{align*}
\sigma_{\text{sstate}=\text{`WA'}} & \quad \text{Supplier(sno, sname, scity, sstate)} \\
\sigma_{\text{Price} > 100} & \quad \text{Part(pno, pname, psize, pcolor)} \\
\text{Supply(sno, pno, price)} & \quad \text{Supply(sno, pno, price)}
\end{align*}
\]

\[
(\text{SELECT } Q.\text{sno} \\
\text{FROM Supplier Q} \\
\text{WHERE } Q.\text{sstate} = \text{`WA'}}) \\
\text{EXCEPT} \\
(\text{SELECT P.sno} \\
\text{FROM Supply P} \\
\text{WHERE P.price} > 100)
\]

Finally…
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 pid is a key)
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 \( O(n^2) \)
2. Merge join \( O(??) \)
3. Hash join \( O(??) \)
Main Memory Algorithms

Logical operator:

Product\( (\text{pid}, \text{name}, \text{price}) \bowtie_{\text{pid} = \text{pid}} \) 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 \( O(n \log n) \)
3. Hash join \( O(??) \)

sort both \( O(n \log n) \)
merge \( O(n) \)
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 \( O(n) \ldots O(n^2) \)

- Add \( n \) to hash \( \sim O(n) \)?
- Lookup \( n \) in hash \( \sim O(n) \)?
BRIEF Review of Hash Tables

Separate chaining:

A (naïve) hash function:

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

Operations:

- find(103) = ??
- insert(488) = ??

Duplicates OK, WHY ??
Value can be different tuple ids.
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

1. Parse & Rewrite Query
2. Select Logical Plan
3. Select Physical Plan
4. Query Execution
SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    AND y.pno = 2
    AND x.scity = 'Seattle'
    AND x.sstate = 'WA'

Give a relational algebra expression for this query
Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)

Relational Algebra

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

Relational algebra expression is also called the “logical query 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'

Supplier(sid, sname, scity, sstate)
Supply(sid, pno, quantity)
A physical query plan is a logical query plan annotated with physical implementation details.

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

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

\[ \pi_{\text{sname}} \]

\( \text{SAME logical query plan} \)
\( \text{DIFFERENT physical plan} \)

\[ \text{SELECT sname} \] 
\[ \text{FROM} \ \text{Supplier} \ x, \ \text{Supply} \ y \] 
\[ \text{WHERE} \ x.\text{sid} = y.\text{sid} \] 
\[ \text{and} \ y.\text{pno} = 2 \] 
\[ \text{and} \ x.\text{scity} = \text{'Seattle'} \] 
\[ \text{and} \ x.\text{sstate} = \text{'WA'} \]
Physical Query Plan 3

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

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

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

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

Different but equivalent logical query plan; different physical plan

\[
\begin{align*}
\text{SELECT } & \text{name} \\
\text{FROM } & \text{Supplier } x, \text{Supply } y \\
\text{WHERE } & x.\text{sid} = y.\text{sid} \\
& \text{and } y.\text{pno} = 2 \\
& \text{and } x.\text{scity} = \text{\textquoteleft\textquoteleft Seattle\textquoteright\textquoteleft\textquoteleft} \\
& \text{and } x.\text{sstate} = \text{\textquoteleft\textquoteleft WA\textquoteright\textquoteleft\textquoteleft}
\end{align*}
\]
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