CS122 Using Relational Databases and SQL Joins

Chengyu Sun
California State University, Los Angeles

Join
- When information is spread over multiple tables
  - Find the projects which John is working on

<table>
<thead>
<tr>
<th>Employees</th>
<th>Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>eid</td>
<td>ename</td>
</tr>
<tr>
<td>'01'</td>
<td>'John'</td>
</tr>
<tr>
<td>'02'</td>
<td>'Susan'</td>
</tr>
<tr>
<td>'02'</td>
<td>'Susan'</td>
</tr>
</tbody>
</table>

Combine Two Tables

<table>
<thead>
<tr>
<th>e.eid</th>
<th>ename</th>
<th>p.eid</th>
<th>pid</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Or what do we get when we do
  - `SELECT * FROM Employees, Projects;`

Cartesian Product

A = \{ a_1, a_2, a_3, a_4 \}
B = \{ b_1, b_2, b_3, b_4 \}
A x B = \{ \text{all possible pairs of a_i b_j} \}

For example:
A = \{ a_1, a_2, a_3 \}, B = \{ b_1, b_2 \}, C = \{ c_1, c_2 \}
A x B = ??
B x C = ??
A x B x C = ??

Combine Two Tables

<table>
<thead>
<tr>
<th>e.eid</th>
<th>ename</th>
<th>p.eid</th>
<th>pid</th>
</tr>
</thead>
<tbody>
<tr>
<td>'01'</td>
<td>'John'</td>
<td>'01'</td>
<td>'T4'</td>
</tr>
<tr>
<td>'01'</td>
<td>'John'</td>
<td>'01'</td>
<td>'X3'</td>
</tr>
<tr>
<td>'02'</td>
<td>'Susan'</td>
<td>'01'</td>
<td>'T4'</td>
</tr>
<tr>
<td>'02'</td>
<td>'Susan'</td>
<td>'02'</td>
<td>'S2'</td>
</tr>
</tbody>
</table>

- Or what do we get when we do
  - `SELECT * FROM Employees, Projects;`

All Kinds of Joins

- Theta join: =, >, >=, <, <=, <>
  - Equi-join: =
  - Nonequi-join: >, >=, <, <=, <>

- Self join

- Inner join
  - Natural join

- Outer join
  - Left join
  - Right join
  - Full outer join
Self Join

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>48</td>
</tr>
<tr>
<td>Susan</td>
<td>45</td>
</tr>
<tr>
<td>Amy</td>
<td>32</td>
</tr>
<tr>
<td>Bob</td>
<td>45</td>
</tr>
</tbody>
</table>

- Find the names of the people who are older than Susan
  - older than or of the same age

INNER Join

- Eliminate rows that do not make sense
- Avoid unintentional Cartesian products

```sql
SELECT * FROM Employees e INNER JOIN Projects p ON e.eid = p.eid;
```

```sql
SELECT * FROM Employees e, Projects p WHERE e.eid = p.eid;
```

Natural Join

- An inner join on all columns with matching names

```sql
SELECT * FROM Employees e INNER JOIN Projects p ON e.eid = p.eid;
```

```sql
SELECT * FROM Employees e NATURAL JOIN Projects p;
```

* Not supported in MS Access

Outer Join

- Left join (left outer join)
  - All rows of an inner join, and the remaining rows from the first table
  - E.g., find the employees who are not working on a project
- Right join (right outer join)
  - All rows of an inner join, and the remaining rows from the second table
- Full outer join

Join Multiple Tables

```sql
SELECT fields FROM Table1, Table2, Table3 WHERE Table1.field1 = Table2.field2 AND Table2.field2 = Table3.field3;
```

Or

```sql
SELECT fields FROM Table1 INNER JOIN ( Table2 INNER JOIN Table3 ON Table2.field2 = Table3.field3 ) ON Table1.field1 = Table2.field2;
```