Multiple-table queries

- Different ways to JOIN tables:
  - Cross join
  - Inner join
    - "Old-style" join
    - Equijoin
    - Theta join
  - JOIN ON
  - Outer join
    - Left join
    - Right join

Review

Mandatory:

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
```

Optional, but Must Appear in this Order:
Multiple Tables

**Ex:** Retrieve the last names of all employees who ate at "A Soup Place" last month

- **What do we do?**

![Employee, Supplier, Lunch tables]

Cross JOIN

```sql
SELECT column_list FROM table1, table2
```

- Provides the Cartesian product of two tables
  - Result is a single table of information:
    - # cols = # cols selected_{table1} + # cols selected_{table2}
    - # rows = # rows_{table1} * # rows_{table2}

**Ex:** SELECT * FROM Item, Supplier

Inner JOIN

```sql
SELECT column_list FROM table1, table2 WHERE table1.col1 = table2.col2
```

- This is an "old-style" inner join
- You can also have other formats - JOIN ON is common:

```sql
SELECT column_list FROM table1 JOIN table2 ON table1.col1 = table2.col2
```
Inner JOIN

- An **inner join** is a traditional join in which only rows that meet a given criteria are selected.
  - If the join criteria involves an equality, then it is called an **equijoin** or **natural join**
  - If the join criteria involves an inequality, then it is called a **theta join**
  - An inner join typically uses the PK - FK relationship to link tables together

**Ex:** List the description and price for all food items provided by suppliers whose name starts with 'A' through 'F'.

Specific types of Inner JOIN

- **Self-join**
  - Used to find information by comparing rows within the same table

```
SELECT column_list FROM table1 T1, table1 T2 WHERE T1.col1 = T2.col2
```

**Ex:** List the Emp_ID and the last name for all employees, along with the last name (alias *m_last*) and the Emp_ID (alias *m_id*) of their manager. Sort by employee ID in ascending order.

Inner JOINs with multiple tables

- **General approach:**
  - Specify attributes to be displayed (or referenced)
  - Include tables from which those attributes come
  - Include tables that lie between these tables
    - Every table must be linked directly to another to avoid cross joins
  - Include table linkages to tie all tables together:
    - WHERE (E.PK = F.FK) AND (F.PK = G.PK) AND...
  - Further restrict rows as necessary (WHERE X=Y)
Inner JOINs with multiple tables (cont.)

**Ex:** List every item description and the last name of each employee that bought one of those items, for all items supplied by "Just Beverages"

Multiple-table queries

- **Different ways to JOIN tables:**
  - Cross join
  - Inner join
    - "Old-style" join
      - Equijoin
      - Theta join
    - JOIN … ON or INNER JOIN … ON
  - Outer join
    - Left join
    - Right join

Outer JOIN

- Returns not only rows matching the join condition, but also the rows with unmatched values
  - Basically provides an Inner JOIN plus additional rows, with partial information, where the join condition wasn't true
  - These additional rows were those that were dropped when the original Cartesian product was turned into a proper inner JOIN

- Three types:
  - Full outer join (not supported in MS Access or MySQL)
  - Left outer join
  - Right outer join
Outer JOIN (cont.)

SELECT table1.attr1, table2.attr2, ... FROM table1
[LEFT | RIGHT] OUTER JOIN table2
ON table1.attr1 = table2.attr1

Ex: ...WHERE A.Employee_id = B.Emp_id;

<table>
<thead>
<tr>
<th>Employee_id</th>
<th>Fname</th>
<th>Lname</th>
<th>Emp_id</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Jim</td>
<td>Kern</td>
<td>202</td>
<td>40000</td>
</tr>
<tr>
<td>202</td>
<td>Jill</td>
<td>Smith</td>
<td>203</td>
<td>45000</td>
</tr>
<tr>
<td>203</td>
<td>John</td>
<td>Sanders</td>
<td>204</td>
<td>50000</td>
</tr>
</tbody>
</table>

Left outer

<table>
<thead>
<tr>
<th>Employee_id</th>
<th>Fname</th>
<th>Lname</th>
<th>Emp_id</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>Jill</td>
<td>Smith</td>
<td>203</td>
<td>45000</td>
</tr>
<tr>
<td>203</td>
<td>John</td>
<td>Sanders</td>
<td>204</td>
<td>50000</td>
</tr>
</tbody>
</table>

Right outer

<table>
<thead>
<tr>
<th>Employee_id</th>
<th>Fname</th>
<th>Lname</th>
<th>Emp_id</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>Jill</td>
<td>Smith</td>
<td>203</td>
<td>45000</td>
</tr>
<tr>
<td>203</td>
<td>John</td>
<td>Sanders</td>
<td>204</td>
<td>50000</td>
</tr>
</tbody>
</table>

Outer JOIN (Example)

• Ex: List the Emp_ID and the last name for all employees, along with the last name (alias m_last) and the Emp_ID (alias m_id) of their manager. Sort by employee ID in ascending order. Include the name and ID for employees that do not have a manager.

• Compare this result with that of the associated self-join
UNION of two SQL statements

- Combines the output of two SELECT statements
  - Supports combining databases (new customer info from another DB)
  - Supports representing multiple competing conditions simultaneously
- Requires the same number of attributes in each statement
- Requires corresponding data types to be compatible
- Removes duplicates

```
SELECT ... FROM ... WHERE ...
UNION
SELECT ... FROM ... WHERE ...
```