Join Overview

- Data in multiple tables
- In relational DB, relationships are not stored
- How to relate multiple tables?
  - Connect records from one table with another
  - Specify a condition when two records should be joined
- Many ways to do this!

Abstract Example

Data in S and T are related by column Y →

Desired result →

Inner Join (WHERE Clause)

SELECT s.x, s.y, t.z
FROM s, t
WHERE s.y = t.y;

Cross Product

Without the join condition:

SELECT * FROM s, t;
WHERE Clause Join, Conceptually

- SELECTing from multiple tables gives cross product:
  - We don’t want irrelevant rows
  - Use WHERE clause to filter rows
- Also, typically use SELECT clause to remove duplicate columns

More Examples

Not all rows must match:

```
SELECT s.x, s.y, t.z
FROM s, t
WHERE s.y = t.y;
```

More Examples

Can match multiple times:

```
SELECT s.x, s.y, t.z
FROM s, t
WHERE s.y = t.y;
```

Less Abstract Example

```
SELECT course_id, section, title, instructor, office, email
FROM mines_courses,
mines_cs_faculty
WHERE instructor = name;
```

Less Abstract Example

```
mines_courses (sample data)
course_id | section | title                | instructor                      
----------|---------|----------------------|---------------------------------
CSCI262   | B       | DATA STRUCTURES     | Painter-Wakefield, Christopher 
CSCI403   | A       | DATABASE MANAGEMENT | Painter-Wakefield, Christopher 
CSCI406   | A       | ALGORITHMS          | Mehta, Dinesh                   
CSCI101   | B       | INTRO TO COMPUTER SCIENCE | Satish, Paul                  

mines_cs_faculty (sample data)
name                  office | email
Mehta, Dinesh         BB 280I | dmehta@mines.edu
Paone, Jeffrey        BB 280C | gpaone@mines.edu
Rader, Cynthia        BB 280I | crader@mines.edu
Painter-Wakefield, Christopher | BB 280I | crader@mines.edu
```

3 or More Tables

- You can join more than 2 tables:
  `SELECT columns
  FROM table1, table2, table3, ...
  WHERE join_condition1, join_condition2, ...
  ;`
- In general, need n – 1 join conditions for n tables (to avoid cross product)
  - Conceptually:
    - Join 2 tables using one join condition
    - Now join 3rd table to resulting rows from first join with another join condition
    - Repeat as necessary...
Normalization Teaser
Why bother?
Suppose all data in one big table:
  - What if you need to update office info for faculty?
  - What happens if a faculty member leaves?
  - What do you insert for new faculty?

Notes
- Join condition can be any Boolean (not just =)
- Don’t forget NULL cannot be (usefully) compared!
- Inner join vs outer join
  - Inner only keeps matching rows
  - Outer lets you keep all rows on either or both sides
  - We’ll see outer join later – different syntax!
- WHERE clause join vs JOIN clause
  - Different ways of achieving same thing
  - JOIN clause more verbose/unwieldy
  - JOIN clause needed to do outer join

Schemas
- We talked about this briefly before...
- Schemas act like namespaces to divide the database into segments
  - Makes things like security a lot easier
  - Helps organize different users/applications
- You each have your own schema; public is visible to all
- You can specify a particular schema using dot notation:
  SELECT * FROM schema.table;
  e.g.,
  SELECT * FROM public.mines_cs_faculty;

Multiple Tables
When working with multiple tables (e.g., joins, subqueries):
  - Column names might “collide”
  - Database can’t resolve ambiguity
  - Use dot notation, prefix with table (e.g., slide 5):
    SELECT s.x, s.y, t.z
    FROM s, t
    WHERE s.y = t.y;

Aliasing
Renaming things (tables & columns) can be useful.

The AS keyword accomplishes this:
  SELECT column_name AS new_name FROM...

  SELECT * FROM table AS new_name...
Aliasing Example 1

Rename tables to a shorter name:

```
SELECT mc.course_id, mcf.name, mcf.email
FROM mines_courses AS mc,
     mines_cs_faculty AS mcf
WHERE mc.instructor = mcf.name;
```

Aliasing Example 2a

Rename tables when joining a table to itself:

```
SELECT emp.name AS employee,
      sup.name AS supervisor
FROM person AS emp,
     person AS sup
WHERE emp.supervisor_id = sup.id;
```

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>supervisor_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1043</td>
<td>Carpenter, Wood</td>
<td>1222</td>
</tr>
<tr>
<td>1147</td>
<td>Miner, Opal</td>
<td>1222</td>
</tr>
<tr>
<td>1222</td>
<td>Person, Manager</td>
<td>0041</td>
</tr>
</tbody>
</table>

Aliasing Example 2b

How to think about this:

```
SELECT emp.name AS employee,
      sup.name AS supervisor
FROM person AS emp,
     person AS sup
WHERE emp.supervisor_id = sup.id;
```

Think 2 tables (just happen to be identical):

<table>
<thead>
<tr>
<th>emp</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1043</td>
<td>Carpenter, Wood</td>
</tr>
<tr>
<td>1147</td>
<td>Miner, Opal</td>
</tr>
<tr>
<td>1222</td>
<td>Person, Manager</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>supervisor_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1043</td>
<td>Carpenter, Wood</td>
<td>1222</td>
</tr>
<tr>
<td>1147</td>
<td>Miner, Opal</td>
<td>1222</td>
</tr>
<tr>
<td>1222</td>
<td>Person, Manager</td>
<td>0041</td>
</tr>
</tbody>
</table>

Aliasing Example 2c

Rename tables when joining a table to itself:

```
SELECT emp.name AS employee,
      sup.name AS supervisor
FROM person AS emp,
     person AS sup
WHERE emp.supervisor_id = sup.id;
```

Result:

```
employee      | supervisor
--------------|-----------
Carpenter, Wood | Person, Manager 
Person, Manager  | Person, Manager
```

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>supervisor_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1043</td>
<td>Carpenter, Wood</td>
<td>1222</td>
</tr>
<tr>
<td>1147</td>
<td>Miner, Opal</td>
<td>1222</td>
</tr>
<tr>
<td>1222</td>
<td>Person, Manager</td>
<td>0041</td>
</tr>
</tbody>
</table>

Aliasing Example 3

Rename columns for readability:

```
SELECT substring(office from 1 for 2)
FROM mines_cs_faculty;
```

```
substring
-----------
BB
BB
...
```

Using AS:

```
SELECT substring(office from 1 for 2) AS building
FROM mines_cs_faculty;
```

```
building
----------
BB
BB
...
```

AS is Optional

- Many keywords in SQL are optional
- Usually drop AS (at least in my experience):

```
SELECT mc.course_id, mcf.name, mcf.email
FROM mines_courses mc,
     mines_cs_faculty mcf
WHERE mc.instructor = mcf.name;
```
Double Quotes

Recall: string literals must use single quotes:

```sql
SELECT * FROM mines_courses WHERE course_id = 'CSCI403';
```

Double quotes can be used when naming objects (tables, columns, etc.):

```sql
SELECT "name" FROM "mines_cs_faculty";
```

Mostly useful when:
- Names contain spaces – `SELECT foo AS "My Stuff"`
- Names have mixed case (AVOID THIS!!!)

Up Next

- Next lecture:
  Intro to SQL: Types, table creation.
- Due on Friday, August 31: Project 1