CSCI 403
Database Management

5 – Joins; Naming

CS@Mines

JOINS

CS@Mines 2

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!

CS@Mines 3

Abstract Example

Data in S and T are related by column Y →

Table **s**

x	y
one	1
two	2
three	3

Table **t**

y	z
1	3.1415
2	6.6262
3	2.7183

Desired result →

x	y	z
one	1	3.1415
two	2	6.6262
three	3	2.7183

CS@Mines 4

Inner Join (WHERE Clause)

Table **s**

x	y
one	1
two	2
three	3

Table **t**

y	z
1	3.1415
2	6.6262
3	2.7183

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

Join condition

x	y	z
one	1	3.1415
two	2	6.6262
three	3	2.7183

CS@Mines 5

Cross Product

Without the join condition:

```
SELECT * FROM s, t;
```

Table **s**

x	y
one	1
two	2
three	3

Table **t**

y	z
1	3.1415
2	6.6262
3	2.7183

x	s.y	t.y	z
one	1	1	3.1415
one	1	2	6.6262
one	1	3	2.7183
two	2	1	3.1415
two	2	2	6.6262
two	2	3	2.7183
three	3	1	3.1415
…	…	…	…

CS@Mines 6

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:

Table **s**

x	y
one	1
two	2
four	**4**

Table **t**

y	z
1	3.1415
2	6.6262
3	2.7183

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

x	y	z
one	1	3.1415
two	2	6.6262

More Examples

Can match multiple times:

Table **s**

x	y
one	1
two	2
three	3

Table **t**

y	z
1	3.1415
2	**6.6262**
3	2.7183
2	**0.3028**

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

x	y	z
one	1	3.1415
two	**2**	**6.6262**
two	**2**	**0.3028**
three	3	2.7183

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	Sattizahn, Paul

mines_cs_faculty (sample data)

name	office	email
Mehta, Dinesh	BB 280J	dmehta@mines.edu
Paone, Jeffrey	BB 280C	jpaone@mines.edu
Rader, Cynthia		crader@mines.edu
Painter-Wakefield, Christopher	BB 280I	crader@mines.edu

Less Abstract Example

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


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

Naming things

SCHEMAS, NAME RESOLUTION, ALIASING

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;
```

person

id	name	supervisor_id
1043	Carpenter, Wood	1222
1147	Miner, Opal	1222
1222	Person, Manager	0041

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):

emp

id	name	supervisor_id
1043	Carpenter, Wood	1222
1147	Miner, Opal	1222
1222	Person, Manager	0041

sup

id	name	supervisor_id
1043	Carpenter, Wood	1222
1147	Miner, Opal	1222
1222	Person, Manager	0041

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;
```

person

id	name	supervisor_id
1043	Carpenter, Wood	1222
1147	Miner, Opal	1222
1222	Person, Manager	0041
0041	O, C.E.	

Result:

```
    employee     |   supervisor
-----------------+-----------------
 Person, Manager | O, C.E.
 Carpenter, Wood | Person, Manager
 Miner, Opal     | Person, Manager
(3 rows)
```


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:

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

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

```
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