

SQL: Introduction to Joins

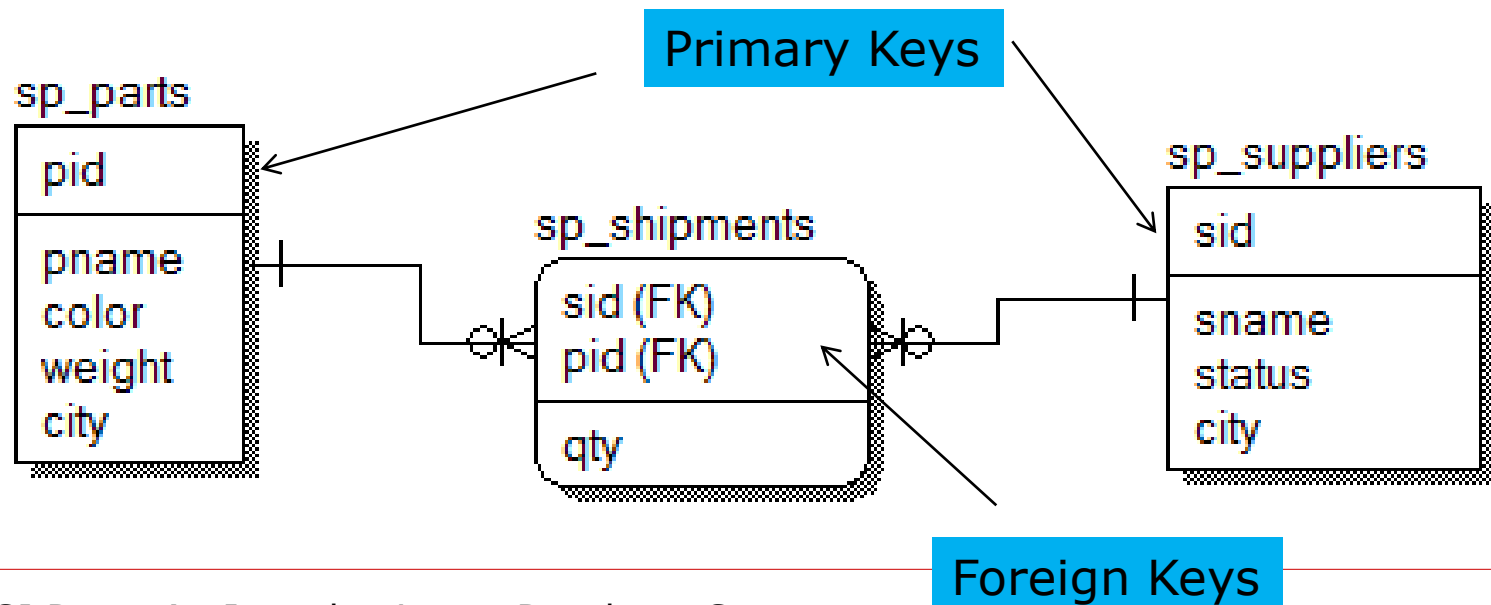


Basics of an Inner Join

- How do we display information from more than one table? i.e.
 - How do we show the department name for an employee?
 - How do we show the supplier name for a food?
 - With a subquery? With a join?
 - There are several ways to do a join – traditional syntax and newer ANSI syntax.
 - Both work and you should become comfortable with both.
-

Another Model: Suppliers, Parts, Shipments

- This is a VERY simple database!
- The primary and foreign keys are the columns we can use to JOIN tables.



Suppliers and Parts Data

Sp_suppliers

sid	sname	status	city
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Sp_shipments

sid	pid	qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S2	P1	300
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Sp_parts

pid	pname	color	weight	city
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris

Inner Join - Traditional Syntax

- Example (from suppliers and parts)
SELECT sh.sid, sh.pid, p.pname, p.weight, sh.qty
FROM sp_shipments sh, sp_parts p
WHERE sh.pid = p.pid;
 - Notice that the FROM clause now contains TWO tables.
 - **Join condition** is specified in the WHERE clause. Must list the matching columns from each table.
 - **'sh'** and **'p'** are alias names for shipment and parts tables. They simplify the rest of the SQL.
-

```
SELECT sh.sid, sh.pid, p.pname, p.weight, sh.qty
FROM sp_shipments sh, sp_parts p
WHERE sh.pid = p.pid;
```

Join Result

sh.sid	sh.pid	p.pname	p.weight	sh.qty
S1	P1	Nut	12	300
S1	P2	Bolt	17	200
S1	P3	Screw	17	400
S1	P4	Screw	14	200
S1	P5	Cam	12	100
S2	P1	Nut	12	300
S3	P2	Bolt	17	200
S4	P2	Bolt	17	200
S4	P4	Screw	14	300
S4	P5	Screw	12	400

Basically, this query shows all columns from the **shipments table** and then adds the details of part name and weight from the **parts table**.

Variations on Join Condition

- Single column from each table matches
 - WHERE a.col1 = b.col1
 - Two or more columns match
 - WHERE a.col1 = b.col1 AND a.col2 = b.col2
 - Row function applied to one of columns
 - WHERE a.col1 = substr(b.col1, 1,4)
 - This allows you to join data which may be similar, but formatted differently. The data should have the same logical meaning.
-

3 Ways to Write an Inner Join

- “Long hand” - using complete table-name.column-name
SELECT **shipments**.sid, **shipments**.pid,
 parts.pname, **parts**.weight, **shipments**.qty
FROM shipments, parts
WHERE **shipments**.pid = **parts**.pid;
- Using table alias names:
SELECT **sh**.sid, **sh**.pid, **p**.pname, **p**.weight, **sh**.qty
FROM shipments **sh**, parts **p**
WHERE **sh**.pid = **p**.pid;
- Using * to list all columns:
SELECT sh.*, p.*
FROM shipments sh, parts p
WHERE sh.pid = p.pid;

BUT.... These are ALL older, more traditional syntax

“Newer” ANSI Join Syntax

- New syntax:

```
SELECT sh.sid, p.pname, sh.pid, sh.qty
```

```
FROM sp_shipments sh INNER JOIN sp_parts p
```

```
ON sh.pid= p.pid
```

```
WHERE p.pname = 'NUT';
```

- Join condition listed in the FROM clause **after the keyword ON**
 - The join condition is no longer placed in the WHERE clause.
-

Old v. New Join Syntax

- Which is better?
 - Try both, you'll find one that you prefer.
 - Just remember that using JOIN with ON is the more modern way.
 - !!! Be familiar with both
 - Lots of advantages to new syntax:
 - Clarity, join is stated explicitly.
 - ANSI syntax does not mix join conditions with other WHERE conditions.
-

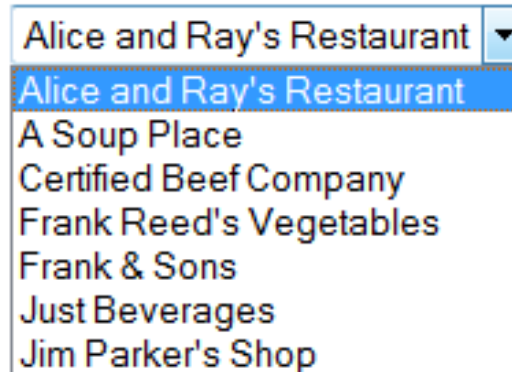
Applications for Joins

- A look-up table is one that only stores a code and perhaps a description or name that belongs to the code.
 - **Stop now** and look at the lunches database schema. Find the look up tables.
 - Some tables only store codes (keys). We then join with the correct "look up" table to get the descriptive information we need.
-

Look Up Tables

- The raw data is here. →
- We can run a query that will populate the drop down list box.
- In a front-end application, we often hide the codes and show just the descriptive information.
- The codes are always available to the application.

SUPPLIER_ID	SUPPLIER_NAME
ARR	ALICE & RAY'S RESTAURANT
ASP	A SOUP PLACE
CBC	CERTIFIED BEEF COMPANY
FRV	FRANK REED'S VEGETABLES
FSN	FRANK & SONS
JBR	JUST BEVERAGES
JPS	JIM PARKER'S SHOP



Alice and Ray's Restaurant ▾
Alice and Ray's Restaurant
A Soup Place
Certified Beef Company
Frank Reed's Vegetables
Frank & Sons
Just Beverages
Jim Parker's Shop

More Applications for Joins

- Put data back together when it has been stored in separate tables (normalized).
 - The basic rule for joins is:
 - Use a join when you need columns of data that exist in different tables.
 - A join is possible if there is a "path" between the tables.
 - If you look at the database diagram, there must be relationship lines connecting the tables.
-

More on Joins

- What does this query do?

SELECT

e.employee_id, e.first_name,

e.last_name, e.dept_code,

d.department_name

FROM l_employees e JOIN

l_departments d

ON e.dept_code = d.dept_code

WHERE e.employee_id < 206;

Discussion

- ON clause: sets up the join condition between 2 tables. Result set will only include data which has matching keys from both tables.
 - WHERE Clause: Restricts result set to certain employees
 - After you establish the join condition, you can add many more conditions in the WHERE clause.
-

Join Example

List all employees who are attending more than one lunch, but exclude employee 208.

```
SELECT e.employee_id,  
       e.first_name,  
       e.last_name,  
       COUNT(*) AS number_of_lunches  
FROM I_employees e JOIN I_lunches l  
ON e.employee_id = l.employee_id  
WHERE e.employee_id != 208  
GROUP BY e.employee_id, e.first_name,  
         e.last_name  
HAVING COUNT(*) > 1  
ORDER BY e.employee_id;
```

Joining > 2 Tables

```
SELECT    e.employee_id,  
          e.first_name, e.last_name,  
          l.lunch_date,  
          f.description, i.quantity  
FROM      l_employees e , l_lunches l,  
          l_lunch_items i, l_foods f  
WHERE     e.employee_id = l.employee_id  
          AND l.lunch_id = i.lunch_id  
          AND i.product_code = f.product_code  
          AND i.supplier_id = f.supplier_id  
          AND e.dept_code = 'SHP'  
ORDER BY  e.employee_id, l.lunch_date;
```

notice the older syntax here

These are the join conditions

-- This is just a filter

What's this query doing?

Answer

- Show all information about the lunches ordered by employees in the shipping department.
 - Show the employee ID, names of the employees, the lunch date, and the descriptions and quantities of the foods they will eat.
 - Sort the result by the `employee_id` and the `lunch_date`.
-

Query Results

Output returns 28 rows,
but is truncated to fit on
slide.

ID	FIRST_NAME	LAST_NAME	LUNCH_DATE	DESCRIPTION	QTY
203	MARTHA	WOODS	16-NOV-11	FRENCH FRIES	1
203	MARTHA	WOODS	16-NOV-11	DESSERT	1
203	MARTHA	WOODS	16-NOV-11	COFFEE	1
203	MARTHA	WOODS	16-NOV-11	SODA	1
203	MARTHA	WOODS	16-NOV-11	GRILLED STEAK	1
203	MARTHA	WOODS	16-NOV-11	FRESH SALAD	1
203	MARTHA	WOODS	05-DEC-11	SOUP OF THE DAY	1
203	MARTHA	WOODS	05-DEC-11	DESSERT	1
203	MARTHA	WOODS	05-DEC-11	SODA	1
203	MARTHA	WOODS	05-DEC-11	COFFEE	2
203	MARTHA	WOODS	05-DEC-11	GRILLED STEAK	1

To Do

- Now try writing the same query using ANSI join syntax.
-

Solution with ANSI Join

```
SELECT    e.employee_id, e.first_name, e.last_name,
          l.lunch_date, f.description, i.quantity
FROM I_employees e
      JOIN I_lunches l ON e.employee_id = l.employee_id
      JOIN I_lunch_items i ON l.lunch_id = i.lunch_id
      JOIN I_foods f ON (i.product_code = f.product_code
                        AND i.supplier_id = f.supplier_id)
WHERE e.dept_code = 'SHP'
ORDER BY e.employee_id, l.lunch_date;
```

Can you join a table with itself?

- YES!! This is a **SELF JOIN**. For example,

```
SELECT [the columns you want]
FROM l_employees e join l_employees m
ON e.manager_id = m.employee_id;
```

- Notice that the table must have two aliases.
 - Notice that columns in the join condition do not have the same name, but both columns contain employee ids.
-