

Introduction to SQL

Joins

Prof. Dr. Jan Kirenz

HdM Stuttgart

JOIN example: Departments & Employees

CREATE the departments table:

```
CREATE TABLE departments (  
    dept_id bigserial,  
    dept varchar(100),  
    city varchar(100),  
    CONSTRAINT dept_key PRIMARY KEY (dept_id),  
    CONSTRAINT dept_city_unique UNIQUE (dept, city)  
);
```

- dept_id column is the table's primary key
- Table includes a **UNIQUE** constraint
 - Guarantees that values in a column are unique.
 - It requires that each row have a unique pair of values for *dept* and *city*
- We add these constraints to avoid duplicate data.

CREATE employees table

```
CREATE TABLE employees (  
  emp_id bigserial,  
  first_name varchar(100),  
  last_name varchar(100),  
  salary integer,  
  dept_id integer REFERENCES departments (dept_id),  
  CONSTRAINT emp_key PRIMARY KEY (emp_id),  
  CONSTRAINT emp_dept_unique UNIQUE (emp_id, dept_id)  
);
```

CREATE employees table

- emp_id is the table's **primary key**: it uniquely identifies each row in the employees table
- dept_id (we added it as a constraint when creating the table) is called a **foreign key**
 - It requires a value entered in a column to already exist in the primary key of the table it **references**.
 - Values in dept_id in the employees table must exist in dept_id in the departments table
- **UNIQUE**: each row must have a **unique** pair of emp_id and dept_id

INSERT values

```
INSERT INTO departments (dept, city)
```

```
VALUES
```

```
  ('Tax', 'Atlanta'),
```

```
  ('IT', 'Boston');
```

```
INSERT INTO employees (first_name, last_name, salary, dept_id)
```

```
VALUES
```

```
  ('Nancy', 'Jones', 62500, 1),
```

```
  ('Lee', 'Smith', 59300, 1),
```

```
  ('Soo', 'Nguyen', 83000, 2),
```

```
  ('Janet', 'King', 95000, 2);
```

SELECT values

```
SELECT *  
FROM departments;
```



SELECT values

```
SELECT *  
FROM employees;
```



JOIN the tables

```
SELECT *  
FROM employees JOIN departments  
ON employees.dept_id = departments.dept_id;
```



JOIN Types

- Creating two tables to explore JOIN types

```
CREATE TABLE schools_left (  
  id integer CONSTRAINT left_id_key PRIMARY KEY,  
  left_school varchar(30)  
);
```

```
CREATE TABLE schools_right (  
  id integer CONSTRAINT right_id_key PRIMARY KEY,  
  right_school varchar(30)  
);
```

Insert values

```
INSERT INTO schools_left (id, left_school) VALUES  
  (1, 'Oak Street School'),  
  (2, 'Roosevelt High School'),  
  (5, 'Washington Middle School'),  
  (6, 'Jefferson High School');
```

```
INSERT INTO schools_right (id, right_school) VALUES  
  (1, 'Oak Street School'),  
  (2, 'Roosevelt High School'),  
  (3, 'Morrison Elementary'),  
  (4, 'Chase Magnet Academy'),  
  (6, 'Jefferson High School');
```

JOIN

- we use JOIN or INNER JOIN, when we want to return rows that have a match in the columns we used for the join

```
SELECT *  
FROM schools_left JOIN schools_right  
ON schools_left.id = schools_right.id;
```

id	left_school	id..3	right_school
1	Oak Street School	1	Oak Street School
2	Roosevelt High School	2	Roosevelt High School
6	Jefferson High School	6	Jefferson High School

INNER JOIN

```
SELECT *  
FROM schools_left INNER JOIN schools_right  
ON schools_left.id = schools_right.id;
```

id	left_school	id..3	right_school
1	Oak Street School	1	Oak Street School
2	Roosevelt High School	2	Roosevelt High School
6	Jefferson High School	6	Jefferson High School

JOIN or INNER JOIN

- Here the join includes all columns in both tables (*).
- Then we specify the two tables to join around the JOIN keyword.
- At last we specify which columns we're joining on, here the id columns of both tables.
- Three school IDs match in both tables, JOIN or INNER JOIN returns only the three rows of those IDs that match.
- Values (schools) that exist only in one of the two tables don't appear in the result.
- Notice that the columns from the left table display on the left of the result table.
- Use JOIN or INNER JOIN when you're working with well-structured, well-maintained data sets and only need to find rows that exist in all the tables you're joining.

INNER JOIN with USING

```
SELECT *  
FROM schools_left INNER JOIN schools_right  
USING (id);
```

id	left_school	right_school
1	Oak Street School	Oak Street School
2	Roosevelt High School	Roosevelt High School
6	Jefferson High School	Jefferson High School

LEFT JOIN

- LEFT JOIN returns all rows from the left table and displays blank rows from the right table if no matching values are found in the joined columns.

```
SELECT *  
FROM schools_left LEFT JOIN schools_right  
ON schools_left.id = schools_right.id;
```

id	left_school	id..3	right_school
1	Oak Street School	1	Oak Street School
2	Roosevelt High School	2	Roosevelt High School
5	Washington Middle School	NA	NA
6	Jefferson High School	6	Jefferson High School

RIGHT JOIN

- RIGHT JOIN returns all rows from the right table and displays blank rows from the left table if no matching values are found in the joined columns

```
SELECT *  
FROM schools_left RIGHT JOIN schools_right  
ON schools_left.id = schools_right.id;
```

id	left_school	id..3	right_school
1	Oak Street School	1	Oak Street School
2	Roosevelt High School	2	Roosevelt High School
NA	NA	3	Morrison Elementary
NA	NA	4	Chase Magnet Academy
6	Jefferson High School	6	Jefferson High School

Aliasing (AS)

- Hint: Instead of writing the full table name, you can use table aliasing as a shortcut:

```
SELECT e.salary, d.city  
FROM employees AS e  
INNER JOIN departments AS d  
ON e.dept_id = d.dept_id;
```

salary	city
62500	Atlanta
59300	Atlanta
83000	Boston
95000	Boston

FULL OUTER JOIN

- Shows **all** rows from both tables in a join, regardless of whether any match:

```
SELECT *  
FROM schools_left FULL OUTER JOIN schools_right  
ON schools_left.id = schools_right.id;
```

id	left_school	id..3	right_school
1	Oak Street School	1	Oak Street School
2	Roosevelt High School	2	Roosevelt High School
5	Washington Middle School	NA	NA
6	Jefferson High School	6	Jefferson High School
NA	NA	4	Chase Magnet Academy
NA	NA	3	Morrison Elementary

FULL OUTER JOIN

```
SELECT *  
FROM schools_right FULL OUTER JOIN schools_left  
ON schools_right.id = schools_left.id;
```

id	right_school	id..3	left_school
1	Oak Street School	1	Oak Street School
2	Roosevelt High School	2	Roosevelt High School
3	Morrison Elementary	NA	NA
4	Chase Magnet Academy	NA	NA
6	Jefferson High School	6	Jefferson High School
NA	NA	5	Washington Middle School

FULL OUTER JOIN with USING

```
SELECT *  
FROM schools_left FULL OUTER JOIN schools_right  
USING (id);
```

id	left_school	right_school
1	Oak Street School	Oak Street School
2	Roosevelt High School	Roosevelt High School
5	Washington Middle School	NA
6	Jefferson High School	Jefferson High School
4	NA	Chase Magnet Academy
3	NA	Morrison Elementary

CROSS JOIN

```
SELECT *  
FROM schools_left CROSS JOIN schools_right;
```

- Lines up each row in the left table with each row in the right table to present all possible combinations of row.
- Because the join doesn't need to find matches between key fields, there is no need to provide the clause using ON or USING.

id	left_school	id..3	right_school
1	Oak Street School	1	Oak Street School
1	Oak Street School	2	Roosevelt High School
1	Oak Street School	3	Morrison Elementary
1	Oak Street School	4	Chase Magnet Academy
1	Oak Street School	6	Jefferson High School
2	Roosevelt High School	1	Oak Street School
2	Roosevelt High School	2	Roosevelt High School
2	Roosevelt High School	3	Morrison Elementary
2	Roosevelt High School	4	Chase Magnet Academy
2	Roosevelt High School	6	Jefferson High School
5	Washington Middle School	1	Oak Street School
5	Washington Middle School	2	Roosevelt High School
5	Washington Middle School	3	Morrison Elementary
5	Washington Middle School	4	Chase Magnet Academy
5	Washington Middle School	6	Jefferson High School
6	Jefferson High School	1	Oak Street School
6	Jefferson High School	2	Roosevelt High School

Using NULL to find missing values

```
SELECT *  
FROM schools_left LEFT JOIN schools_right  
ON schools_left.id = schools_right.id  
WHERE schools_right.id IS NULL;
```

id	left_school	id..3	right_school
5	Washington Middle School	NA	NA

- The result shows only the one row from the left table that didn't have a match on the right side.