# SQL Joins

Usually, single table queries are not sufficient to answer the complex questions desired from a database. You will recall that the relational database is driven by entities that are related in some maner. These relationships between the tables are what provide a great deal of descriptive capability, as well as data integrity.

This reading discusses table JOINs in SQL.

The JOIN is the declaration that two or more tables should have their rows merged together.

## Keys, Foreign and Primary/Candidate

Recall our definitions of **Keys** from the previous day.

### Key

A sub-tuple of one or more fields used to identify rows in the table. Note a sub-tuple is simply one or more columns from the table.

### Candidate Key

A key that uniquely identifies a row in the table.

### Primary Key (PK)

A candidate key that is chosen as the primary index for the table.

### Foreign Key (FK)

A sub-tuple in Table B that is a candidate key in Table A is a Foreign Key in Table B.

That is, it links Table B row(s) back to a single "parent" row of data in Table A.

**Example:** Orders belong to Customers by way of the Order's Foreign Key reference (customer\_id) to the Customer Primary Key (customer\_id).

### Keys in the Sample Database?

We can see in **Person** table that it has an id column. This is standard nomenclature for a PRIMARY KEY column.

**Person**: people who took readings.

| **id** | **personal** | **family** |
| --- | --- | --- |
| dyer | William | Dyer |
| pb | Frank | Pabodie |
| lake | Anderson | Lake |
| roe | Valentina | Roerich |
| danforth | Frank | Danforth |

The **Site** table has name, which looks to be an identifier.

**Site**: locations where readings were taken.

| **name** | **lat** | **long** |
| --- | --- | --- |
| DR-1 | -49.85 | -128.57 |
| DR-3 | -47.15 | -126.72 |
| MSK-4 | -48.87 | -123.4 |

We can see in **Visited** table that it has an id column. Again, we can expect this to be a PRIMARY KEY column.

Additionally, we can se that site appears to align the values in the Site table, name column (or Site.name)

Since Site.name is the primary key of the **Site** table, and Visited.site aligns to a Site.name; we say Visited.site is a **FOREIGN KEY**.

**Visited**: when readings were taken at specific sites.

| **id** | **site** | **dated** |
| --- | --- | --- |
| 619 | DR-1 | 1927-02-08 |
| 622 | DR-1 | 1927-02-10 |
| 734 | DR-3 | 1930-01-07 |
| 735 | DR-3 | 1930-01-12 |
| 751 | DR-3 | 1930-02-26 |
| 752 | DR-3 | -null- |
| 837 | MSK-4 | 1932-01-14 |
| 844 | DR-1 | 1932-03-22 |

The **Survey** table is a little different.  
Looking at the first two rows, we can see that taken is not unique. Furthermore, the combination of (taken,person) is not unique. However, it appears that the combination of (taken,person, quant) is a unique set of columns that will identify a single row. Therefore, we will use this sub-tuple as the **Primary Key**.

Additionally, we can

**Survey**: the actual readings.

| **taken** | **person** | **quant** | **reading** |
| --- | --- | --- | --- |
| 619 | dyer | rad | 9.82 |
| 619 | dyer | sal | 0.13 |
| 622 | dyer | rad | 7.8 |
| 622 | dyer | sal | 0.09 |
| 734 | pb | rad | 8.41 |
| 734 | lake | sal | 0.05 |
| 734 | pb | temp | -21.5 |
| 735 | pb | rad | 7.22 |
| 735 | -null- | sal | 0.06 |
| 735 | -null- | temp | -26.0 |
| 751 | pb | rad | 4.35 |
| 751 | pb | temp | -18.5 |
| 751 | lake | sal | 0.1 |
| 752 | lake | rad | 2.19 |
| 752 | lake | sal | 0.09 |
| 752 | lake | temp | -16.0 |
| 752 | roe | sal | 41.6 |
| 837 | lake | rad | 1.46 |
| 837 | lake | sal | 0.21 |
| 837 | roe | sal | 22.5 |
| 844 | roe | rad | 11.25 |

## Relationships

Recall some details about relationships between the entities in the relational model.

* Entities have some relationship to other entities in the system.
* Relationships illustrate an association between two entities.
* Cardinality Constraints:
  + Zero or More
  + One or More
  + One and only One
  + Zero or One

### Example:

We see that the relationship between Site and Visited has some cardinatilties.

* The FK from Visited to Site refers to **One and only One** Site.
* The relationship from Site to Visited shows that a Site is visited **One or More** times.
* Relationship is : **Visited.site = Site.name**

**Site**

| **name** | **lat** | **long** |
| --- | --- | --- |
| DR-1 | -49.85 | -128.57 |
| DR-3 | -47.15 | -126.72 |
| MSK-4 | -48.87 | -123.4 |

**Visited**

| **id** | **site** | **dated** |
| --- | --- | --- |
| 619 | DR-1 | 1927-02-08 |
| 622 | DR-1 | 1927-02-10 |
| 734 | DR-3 | 1930-01-07 |
| 735 | DR-3 | 1930-01-12 |
| 751 | DR-3 | 1930-02-26 |
| 752 | DR-3 | -null- |
| 837 | MSK-4 | 1932-01-14 |
| 844 | DR-1 | 1932-03-22 |

## Join Syntaxes

Various database and DBMS support extensions to the standard JOIN sytnax. Let's look at an example with the tables above, Site and Visited.

**SELECT** Visited.id, Site.name, Site.lat, Site.long, Visited.dated

**FROM**

Site **JOIN** Visited

**ON** Visited.site = Site.name

Note, the Syntax is

**FROM**

LeftTable **JOIN** RightTable

**ON** BooleanCondition

In [ ]:

**%**load\_ext sql

**%**sql sqlite:**///**..**/**..**/**..**/**datasets**/**survey.db

#### Example query and alternatives

In [ ]:

**%%**sql

SELECT Visited.id, Site.name, Site.lat, Site.long, Visited.dated

FROM

Site JOIN Visited

ON Visited.site = Site.name;

In [ ]:

**%%**sql

SELECT Visited.id, Site.name, Site.lat, Site.long, Visited.dated

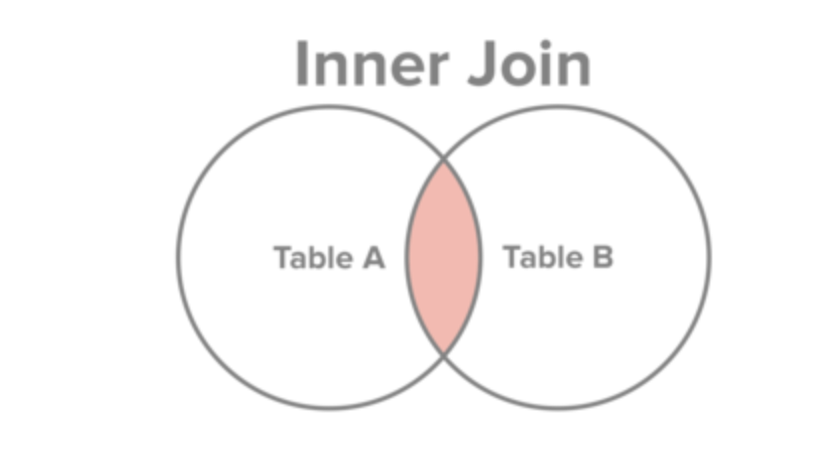
FROM Site, Visited

WHERE Visited.site = Site.name;

### INNER JOIN

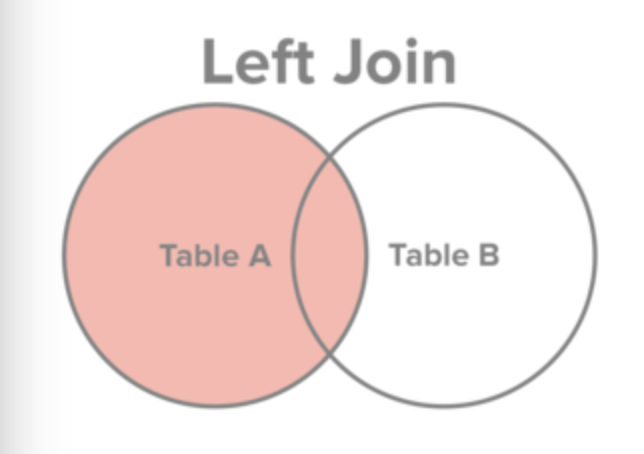
The above is an example of an **INNER JOIN**.

In an INNER JOIN, only the matched rows are shown. Data from unmatched rows in the left and right tables are excluded.



### LEFT JOIN

In an LEFT JOIN, all the rows from the left table are shown. If these rows are unmatched in the left table, then NULL values are returned for the right table's columns.



In [ ]:

**%%**sql

SELECT Person.id, Person.family, Survey.quant, Survey.reading

FROM

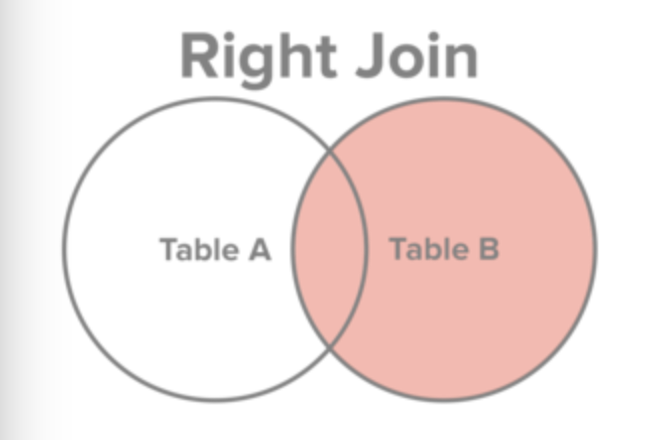
Person LEFT JOIN Survey

ON Person.id = Survey.person;

**Note:** the last row with danforth has 'None','None' as data from the right table because no rows matched.

### RIGHT JOIN

In an RIGHT JOIN, all the rows from the right table are shown. If these rows are unmatched in the right table, then NULL values are returned for the left table's columns.



**CAUTION:** SQLite does not current support Right or Full Outer Joins.

If you rewrote the previous SQL to be a RIGHT JOIN, e.g.,

**SELECT** Person.id, Person.family, Survey.quant, Survey.reading

**FROM**

Survey RIGHT **JOIN** Person

**ON** Survey.person = Person.id;

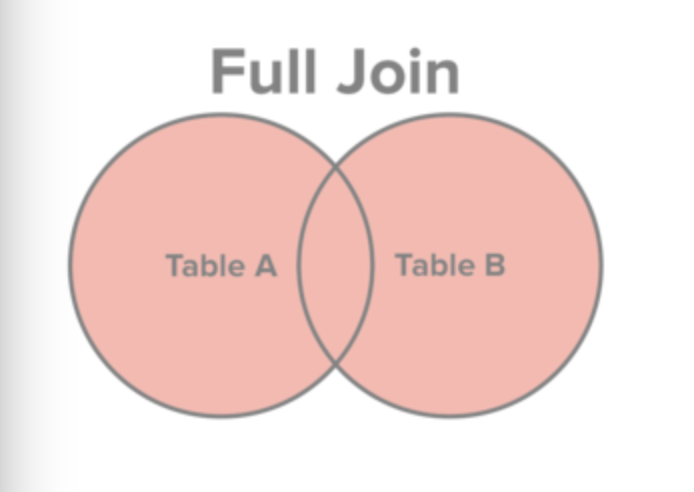
You would get an error similar to below:

(sqlite3.OperationalError) RIGHT and FULL OUTER JOINs are not currently supported

### OUTER JOIN

In an OUTER JOIN, or FULL OUTER JOIN, all the rows from the both tables are shown. Rows that are unmatched in the left table are shown with NULL values for the right table's columns. Rows that are unmatched in the right table are shown with NULL values for the left table's columns.

Essentially, this is the combination of the LEFT JOIN and the RIGHT JOIN.



## For additional reading

* please revisit the Join portion of [the DBMS review](https://indigo.sgn.missouri.edu/static/PDF/RDBMS_review_handout.pdf).
  + Foreign Keys, Sect. 3.3.4
  + Joins, Sect. 3.5.2