MySQL

What is a Database?

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

MySQL databases are relational

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

The MySQL Database Server is very fast, reliable, scalable, and easy to use

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

A **Relational DataBase Management System (RDBMS)** is a software that −

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

RDBMS Terminology

- **Database** − A database is a collection of tables, with related data.
- **Table** − A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- **Column** − One column (data element) contains data of one and the same kind, for example the column postcode.
- **Row** − A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
- **Redundancy** − Storing data twice, redundantly to make the system faster.
- **Primary Key** − A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row.
- **Foreign Key** − A foreign key is the linking pin between two tables.
- **Compound Key** − A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- **Index** − An index in a database resembles an index at the back of a book.
- **Referential Integrity** − Referential Integrity makes sure that a foreign key value always points to an existing row.

MySQL - Data Types

MySQL uses many different data types broken into three categories −

- Numeric
- Date and Time
- String Types.

The following list shows the common numeric data types and their descriptions −

- **INT** − A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
- **TINYINT** − A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
- **SMALLINT** − A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
- **MEDIUMINT** − A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
- **BIGINT** − A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
- **FLOAT(M,D)** − A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
- **DOUBLE(M,D)** − A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.

- **DECIMAL(M,D)** − An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types

The MySQL date and time datatypes are as follows −

- **DATE** − A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.
- **DATETIME** − A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
- **TIMESTAMP** − A timestamp between midnight, January 1st, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).
- **TIME** − Stores the time in a HH:MM:SS format.
- **YEAR(M)** − Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

String Types

Although the numeric and date types are fun, most data you'll store will be in a string format. This list describes the common string datatypes in MySQL.

- **CHAR(M)** − A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- **VARCHAR(M)** − A variable-length string between 1 and 255 characters in length. For example, VARCHAR(25). You must define a length when creating a VARCHAR field.

- **BLOB or TEXT** − A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data. The difference between the two is that the sorts and comparisons on the stored data are **case sensitive** on BLOBs and are **not case sensitive** in TEXT fields. You do not specify a length with BLOB or TEXT.
- **TINYBLOB or TINYTEXT** − A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
- **MEDIUMBLOB or MEDIUMTEXT** − A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- **LONGBLOB or LONGTEXT** − A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** − An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

Create MySQL Tables

To begin with, the table creation command requires the following details −

- Name of the table
- Name of the fields
- Definitions for each field

Syntax

Here is a generic SQL syntax to create a MySQL table −

```
CREATE TABLE table_name (column_name column_type);
```

```
create table tutorials_tbl( tutorial_id INT NOT NULL AUTO_INCREMENT, tuto
rial_title VARCHAR(100) NOT NULL, tutorial_author VARCHAR(40) NOT NULL, s
ubmission_date DATE, PRIMARY KEY ( tutorial_id ) );
```

- Field Attribute **NOT NULL** is being used because we do not want this field to be NULL. So, if a user will try to create a record with a NULL value, then MySQL will raise an error.
- Field Attribute **AUTO_INCREMENT** tells MySQL to go ahead and add the next available number to the id field.
- Keyword **PRIMARY KEY** is used to define a column as a primary key. You can use multiple columns separated by a comma to define a primary key.

Drop MySQL Tables

It is very easy to drop an existing MySQL table, but you need to be very careful while deleting any existing table because the data lost will not be recovered after deleting a table.

Syntax

Here is a generic SQL syntax to drop a MySQL table −

```
DROP TABLE table_name ;
```

MySQL - Insert Query

To insert data into a MySQL table, you would need to use the SQL **INSERT INTO** command. You can insert data into the MySQL table by using the mysql> prompt or by using any script like PHP.

Syntax

Here is a generic SQL syntax of INSERT INTO command to insert data into the MySQL table −

```
INSERT INTO table_name ( field1, field2,...fieldN ) VALUES ( value1, valu
e2,...valueN );
```

To insert string data types, it is required to keep all the values into double or single quotes. For example "**value**".

MySQL - Select Query

The SQL SELECT command is used to fetch data from the MySQL database.

Syntax

Here is generic SQL syntax of SELECT command to fetch data from the MySQL table –

```
SELECT field1, field2,...fieldN FROM table_name1, table_name2... [WHERE C
lause] [OFFSET M ][LIMIT N]
```

- You can use one or more tables separated by comma to include various conditions using a WHERE clause, but the WHERE clause is an optional part of the SELECT command.
- You can fetch one or more fields in a single SELECT command.
- You can specify star (*) in place of fields. In this case, SELECT will return all the fields.
- You can specify any condition using the WHERE clause.
- You can specify an offset using **OFFSET** from where SELECT will start returning records. By default, the offset starts at zero.
- You can limit the number of returns using the **LIMIT** attribute.

MySQL - WHERE Clause

We have seen the SQL **SELECT** command to fetch data from a MySQL table. We can use a conditional clause called the **WHERE Clause** to filter out the results. Using this WHERE clause, we can specify a selection criteria to select the required records from a table.

Syntax

The following code block has a generic SQL syntax of the SELECT command with the WHERE clause to fetch data from the MySQL table −

```
SELECT field1, field2,...fieldN table_name1, table_name2... [WHERE condit
ion1 [AND [OR]] condition2.....
```

- You can use one or more tables separated by a comma to include various conditions using a WHERE clause, but the WHERE clause is an optional part of the SELECT command.
- You can specify any condition using the WHERE clause.
- You can specify more than one condition using the **AND** or the **OR** operators.
- A WHERE clause can be used along with DELETE or UPDATE SQL command also to specify a condition.

The **WHERE** clause works like an **if condition** in any programming language. This clause is used to compare the given value with the field value available in a MySQL table. If the given value from outside is equal to the available field value in the MySQL table, then it returns that row.

MySQL - UPDATE Query

There may be a requirement where the existing data in a MySQL table needs to be modified. You can do so by using the SQL **UPDATE** command. This will modify any field value of any MySQL table.

Syntax

The following code block has a generic SQL syntax of the UPDATE command to modify the data in the MySQL table −

```
UPDATE table_name SET field1 = new-value1, field2 = new-value2 [WHERE Cla
use]
```

- You can update one or more field altogether.
- You can specify any condition using the WHERE clause.
- You can update the values in a single table at a time.

The WHERE clause is very useful when you want to update the selected rows in a table.

MySQL - DELETE Query

If you want to delete a record from any MySQL table, then you can use the SQL command **DELETE FROM**.

Syntax

The following code block has a generic SQL syntax of the DELETE command to delete data from a MySQL table.

```
DELETE FROM table_name [WHERE Clause]
```

- If the WHERE clause is not specified, then all the records will be deleted from the given MySQL table.
- You can specify any condition using the WHERE clause.
- You can delete records in a single table at a time.

The WHERE clause is very useful when you want to delete selected rows in a table.

MySQL - LIKE Clause

We have seen the SQL **SELECT** command to fetch data from the MySQL table. We can also use a conditional clause called as the **WHERE** clause to select the required records.

A WHERE clause with the 'equal to' sign (=) works fine where we want to do an exact match. Like if "tutorial_author = 'Sanjay'". But there may be a requirement where we want to filter out all the results where tutorial_author name should contain "jay". This can be handled using **SQL LIKE Clause** along with the WHERE clause.

If the SQL LIKE clause is used along with the % character, then it will work like a meta character (*) as in UNIX, while listing out all the files or directories at the command prompt. Without a % character, the LIKE clause is very same as the **equal to** sign along with the WHERE clause.

Syntax

The following code block has a generic SQL syntax of the SELECT command along with the LIKE clause to fetch data from a MySQL table.

```
SELECT field1, field2,...fieldN table_name1, table_name2... WHERE field1
LIKE condition1 [AND [OR]] filed2 = 'somevalue'
```

- You can specify any condition using the WHERE clause.
- You can use the LIKE clause along with the WHERE clause.
- You can use the LIKE clause in place of the **equals to** sign.
- When LIKE is used along with % sign then it will work like a meta character search.
- You can specify more than one condition using **AND** or **OR** operators.
- A WHERE...LIKE clause can be used along with DELETE or UPDATE SQL command also to specify a condition.

MySQL - Sorting Results

We have seen the SQL **SELECT** command to fetch data from a MySQL table. When you select rows, the MySQL server is free to return them in any order, unless you instruct it otherwise by saying how to sort the result. But, you sort a result set by adding an **ORDER BY** clause that names the column or columns which you want to sort.

Syntax

The following code block is a generic SQL syntax of the SELECT command along with the ORDER BY clause to sort the data from a MySQL table.

```
SELECT field1, field2,...fieldN table_name1, table_name2... ORDER BY fiel
d1, [field2...] [ASC [DESC]]
```

- You can sort the returned result on any field, if that field is being listed out.
- You can sort the result on more than one field.
- You can use the keyword ASC or DESC to get result in ascending or descending order. By default, it's the ascending order.
- You can use the WHERE...LIKE clause in the usual way to put a condition.

Using MySQl Joins

A `JOIN` clause is used to combine rows from two or more tables, based on a related column between them.

Let's look at a selection from the "Orders" table:

Untitled

Title	OrderID

Then, look at a selection from the "Customers" table:

Untitled

CustomerID	CustomerName
1	Alfreds Futterkiste
2	Ana Trujillo Emparedados y
3	Antonio Moreno Taquería

Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.

Then, we can create the following SQL statement (that contains an `INNER JOIN`), that selects records that have matching values in both tables:

Example

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDateFROM OrdersINNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

and it will produce something like this:

Untitled

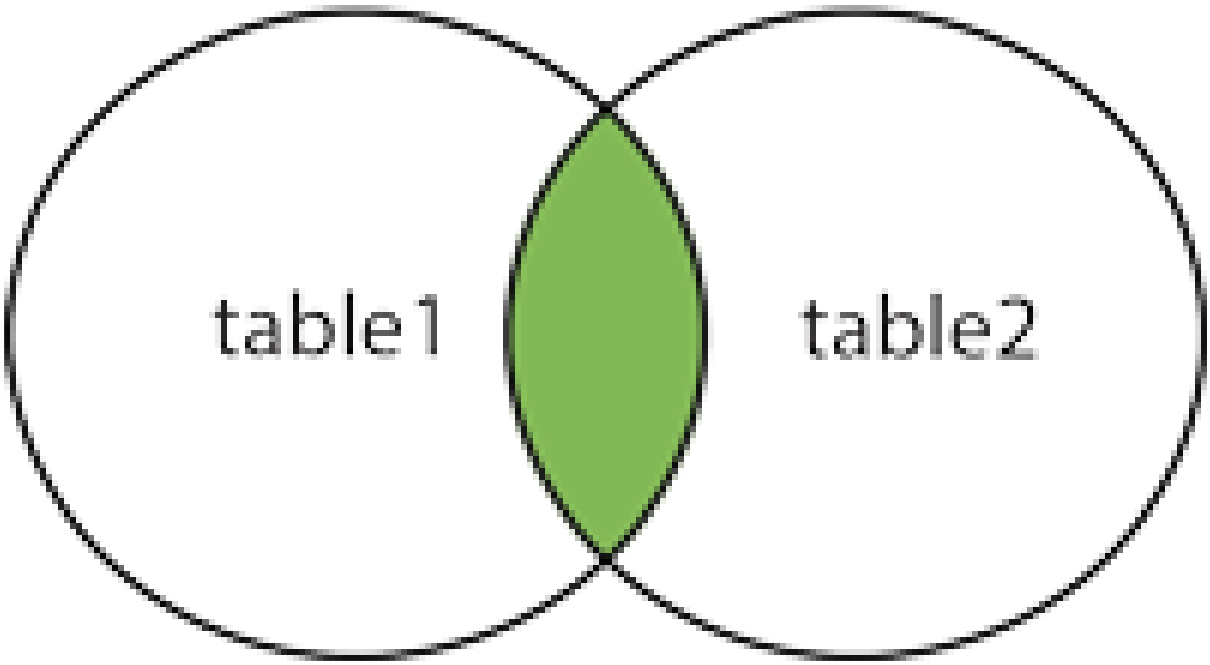
OrderID	CustomerName
10308	Ana Trujillo Emparedados y
10365	Antonio Moreno Taquería
10383	Around the Horn
10355	Around the Horn
10278	Berglunds snabbköp

Different Types of SQL JOINs

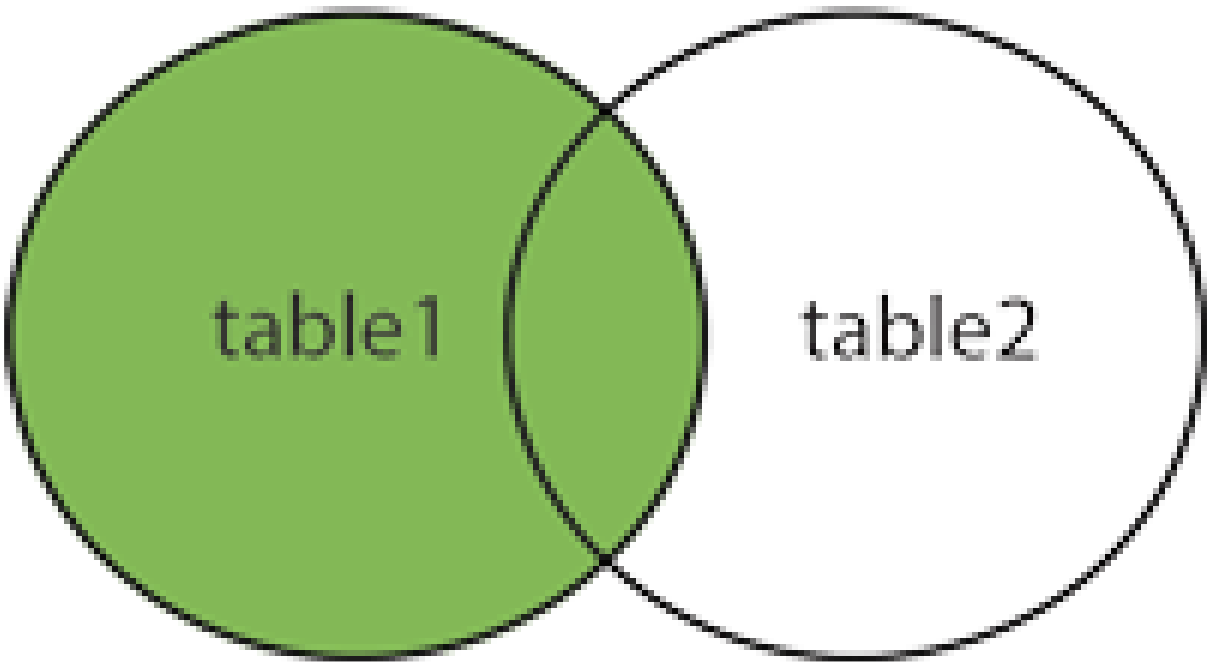
Here are the different types of the JOINs in SQL:

- `(INNER) JOIN`: Returns records that have matching values in both tables
- `LEFT (OUTER) JOIN`: Returns all records from the left table, and the matched records from the right table
- `RIGHT (OUTER) JOIN`: Returns all records from the right table, and the matched records from the left table
- `FULL (OUTER) JOIN`: Returns all records when there is a match in either left or right table

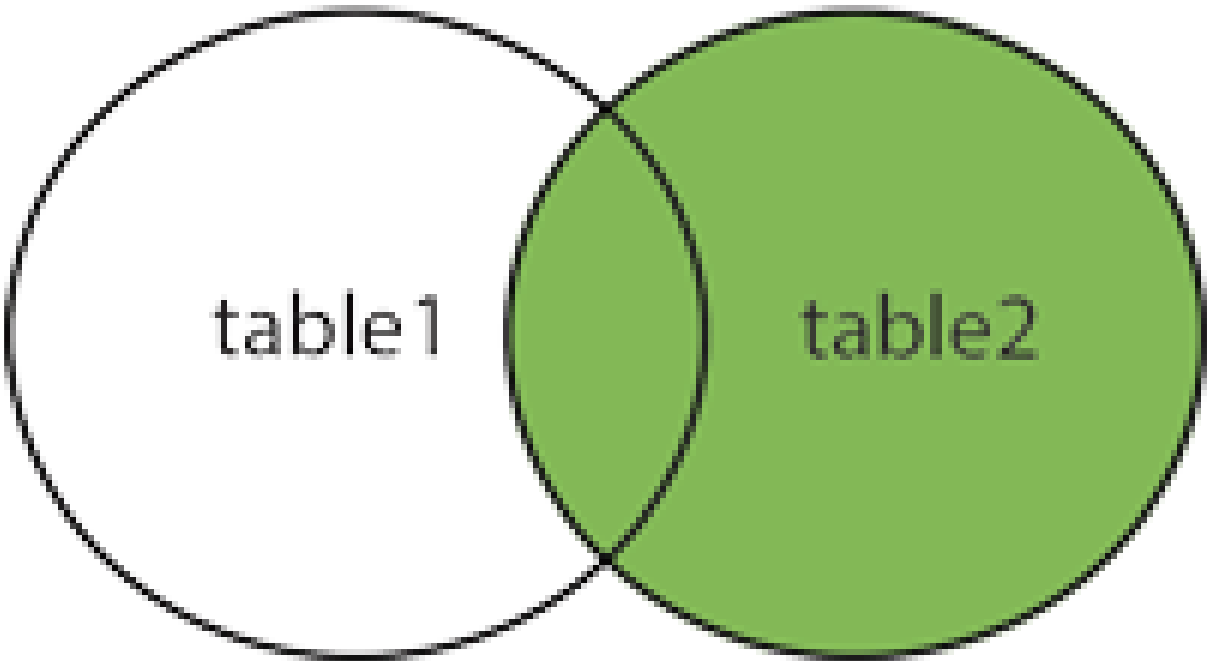
INNER JOIN



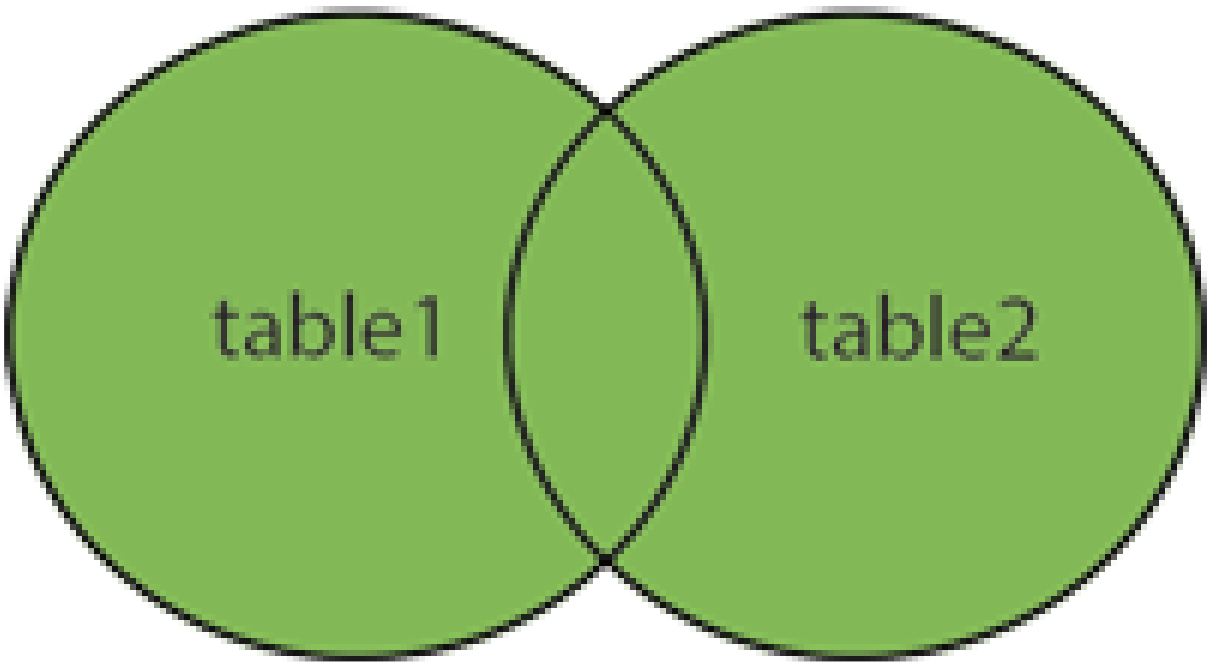
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



THANK YOU

FLOXUS EDUCATION,

Nurture the technical you