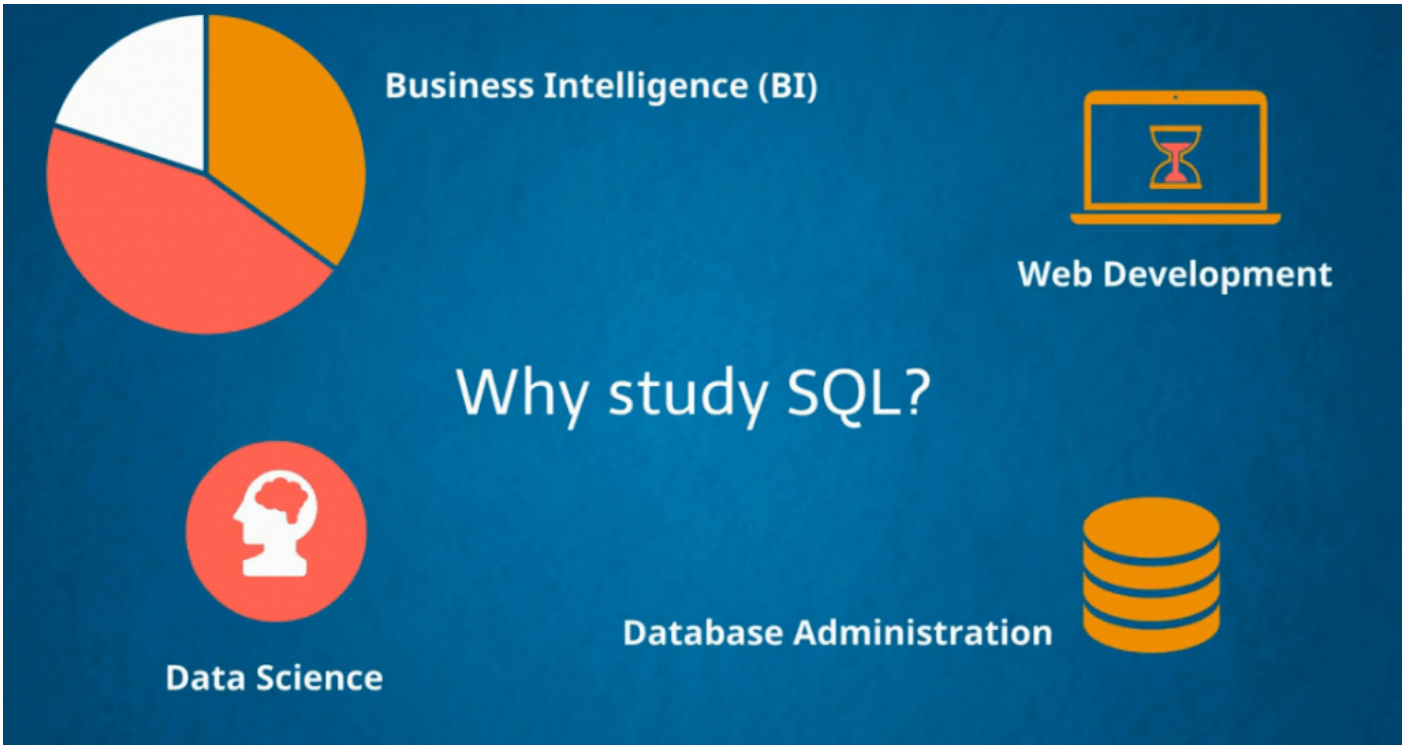
Road to Data Science in 50 Days - Day 7

Databases for Data Science



On Day 6 we were introduced to SQL and its operations, keep following the tutorialspoint guide i shared on day 6 to learn more about SQL.

On Day 7, we will learn about adding a database to our local server and running some queries on it.

Before moving forward, i would suggest you to download MySQL community server, refer the installation guide: https://mid.as/kb/00145/install-configure-mysql-on-windows (https://mid.as/kb/00145/install-configure-mysql-on-windows)

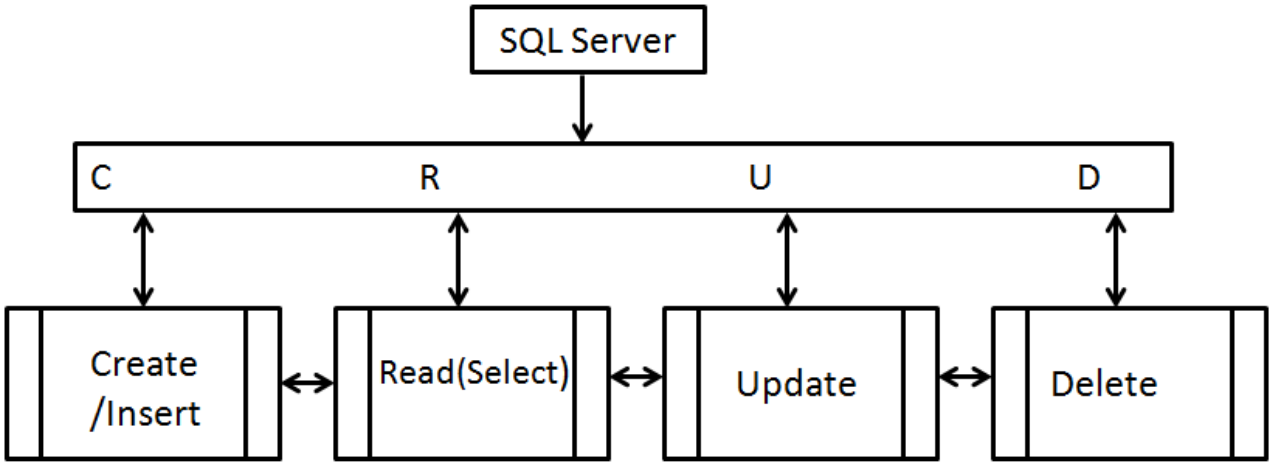
And also download a text editor for a good visual of your database and queries, the video i am going to suggest later has used POPSQL, which i found very useful, download it from the mentioned link: https://popsql.com/ (https://popsql.com/)

While connecting the database, we will enter the hostname as localhost (which means the database is hosted on our local computer)

Following this amazing tutorial, for a tutorial on SQL with live examples: https://www.youtube.com/watch?v=HXV3zeQKqGY (https://www.youtube.com/watch?v=HXV3zeQKqGY)

Let's understand some of the basic SQL operations.

SQL Operations



Create Database

1. CREATE DATABASE

The SQL **CREATE DATABASE** statement is used to create a new SQL database.

Syntax

```
CREATE DATABASE DatabaseName;
```

Example: Let's create a new database named Student

```
CREATE DATABASE Student;
```

This would create a new database in our RDBMS, Note that we should always create database with unique names in our RDBMS.

To view all the database in the system, use query:

```
SHOW DATABASES;
```

2. USE DATABASE

When you have multiple databases in your SQL Schema, then before starting your operation, you would need to select a database where all the operations would be performed.

The SQL **USE** statement is used to select any existing database in the SQL schema.

Syntax

```
USE DatabaseName;
```

Example: We can see all the database present in the system and use our database

```
SHOW DATABASES;
USE Student;
```

3. DROP DATABASE

The SQL **DROP DATABASE** statement is used to drop an existing database in SQL schema.

Syntax

```
USE DATABASE DatabaseName;
```

Create Table in Database

Once we create the database, we need to table to store the data in a tabular form.

1, CREATE TABLE

The SQL **CREATE TABLE** statement is used to create a new table.

Syntax

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
   PRIMARY KEY( one or more columns )
);
```

CREATE TABLE is the keyword telling the database system what you want to do. In this case, you want to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement.

Example The following code block is an example, which creates a STUDENTS table with an ID as a primary key and NOT NULL are the constraints showing that these fields cannot be NULL while creating records in this table

```
CREATE TABLE STUDENTS(
   ROLLNO   INT          NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL,
   PRIMARY KEY (ROLLNO)
);
```

2. DROP TABLE

The SQL **DROP TABLE** statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.

Syntax

```
DROP TABLE table_name;
```

3. INSERT INTO

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);
```

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

The SQL INSERT INTO syntax will be as follows −

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

Example: To enter the info of Students into our table

You can do it in either ways:

```
INSERT INTO STUDENT (ROLLNO,NAME,AGE)
VALUES (1, 'Saif', 24);
or
INSERT INTO STUDENT
VALUES (2, 'Zaid', 22);
```

4. SELECT QUERY

The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

Syntax

```
SELECT column1, column2, columnN FROM table_name;
```

Example: To select all the columns and rows from our students table, we can either write all the columns names or just asterik (*).

```
SELECT * FROM STUDENTS;
or
SELECT ROLLNO, NAME, AGE
FROM STUDENTS;
```

5. WHERE Clause

The SQL **WHERE** clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table. You should use the WHERE clause to filter the records and fetching only the necessary records.

The WHERE clause is not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc., which we would examine in the subsequent chapters.

Syntax

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition]
```

Example: Fetching the data of students whose age is greater than 20

```
SELECT * FROM STUDENTS
WHERE AGE > 20;
```

5. AND and OR Conjunctive Operators

The SQL **AND & OR** operators are used to combine multiple conditions to narrow data in an SQL statement. These two operators are called as the conjunctive operators.

These operators provide a means to make multiple comparisons with different operators in the same SQL statement.

Syntax

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition1] AND [condition2]...AND [conditionN];
```

Example: Suppose we also had the city from which the student belongs to, and we want to fetch details of all students who are more than 20 years old and live in Mumbai City

```
SELECT * FROM STUDENTS
WHERE AGE > 20 AND CITY = "MUMBAI";
```

Just like the operations we learnt, there are various other basic operations such as:

- UPDATE Query
- DELETE Query
- LIKE Clause
- Top Clause
- ORDER By
- GROUP By
- DISTINCT Keywords
- Sorting Results

I would suggest to go trough the tutorialspoint's website to learn and all try all the basic operations before moving to the advanced topics: https://www.tutorialspoint.com/sql/index.htm (https://www.tutorialspoint.com/sql/index.htm)

Simulatanueously, if you are comfortable with video tutorials, then follow this Full Database Course: https://www.youtube.com/watch?v=HXV3zeQKqGYhttps://www.youtube.com/watch?v=HXV3zeQKqGY (https://www.youtube.com/watch?v=HXV3zeQKqGYhttps://www.youtube.com/watch?v=HXV3zeQKqGY) by

freeCodeCamp.org

Let's understand some of the advanced topics of SQL, before moving to other aspects of Databases and Data Mining/Warehousing aspects.

Constraints

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table. We have already discussed these in our SQL Basics.

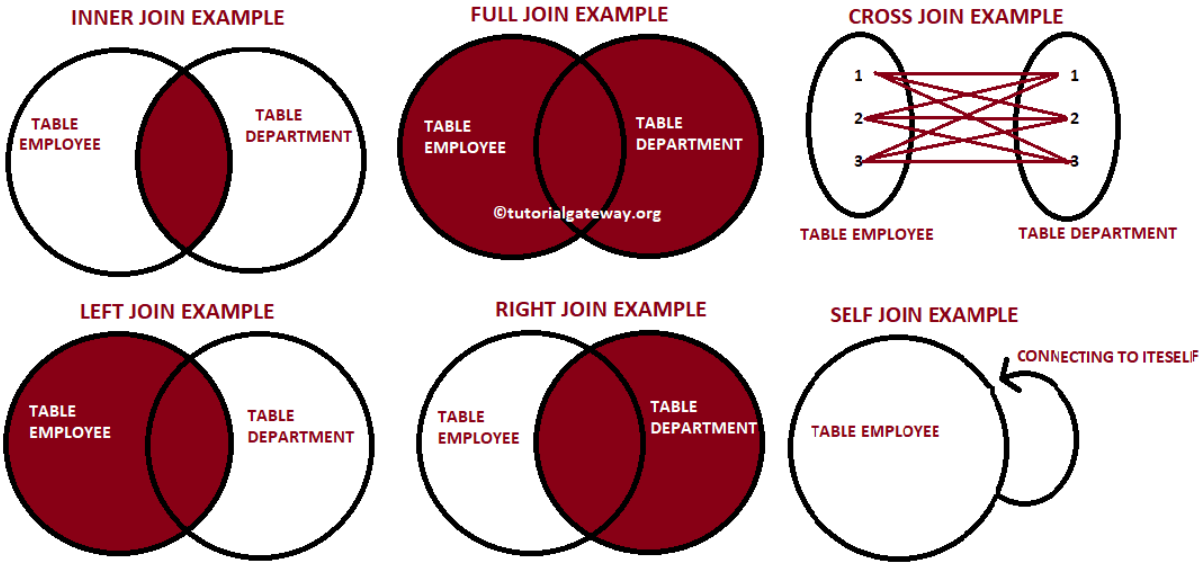
Following are some of the most commonly used constraints available in SQL.

- NOT NULL Constraint − Ensures that a column cannot have NULL value.
- DEFAULT Constraint − Provides a default value for a column when none is specified.
- UNIQUE Constraint − Ensures that all values in a column are different.
- PRIMARY Key − Uniquely identifies each row/record in a database table.
- FOREIGN Key − Uniquely identifies a row/record in any of the given database table.
- CHECK Constraint − The CHECK constraint ensures that all the values in a column satisfies certain conditions.
- INDEX − Used to create and retrieve data from the database very quickly.

Constraints can be specified when a table is created with the CREATE TABLE statement or you can use the ALTER TABLE statement to create constraints even after the table is created.

Using Joins

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.



There are different types of joins available in SQL −

INNER JOIN − returns rows when there is a match in both tables.

LEFT JOIN − returns all rows from the left table, even if there are no matches in the right table.

RIGHT JOIN − returns all rows from the right table, even if there are no matches in the left table.

FULL JOIN − returns rows when there is a match in one of the tables.

SELF JOIN − is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.

CARTESIAN JOIN − returns the Cartesian product of the sets of records from the two or more joined tables.

To learn more about joins in detail: https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/ (https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/)

UNIONS CLAUSE

The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.

To use this UNION clause, each SELECT statement must have

- The same number of columns selected
- The same number of column expressions
- The same data type and
- Have them in the same order

But they need not have to be in the same length.

Syntax

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

UNION

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

NULL Values

The SQL NULL is the term used to represent a missing value. A NULL value in a table is a value in a field that appears to be blank.

A field with a NULL value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces.

Alias Syntax

You can rename a table or a column temporarily by giving another name known as Alias. The use of table aliases is to rename a table in a specific SQL statement. The renaming is a temporary change and the actual table name does not change in the database. The column aliases are used to rename a table's columns for the purpose of a particular SQL query.

Syntax

For Table:

```
SELECT column1, column2....
FROM table_name AS alias_name
WHERE [condition];
```

For Column:

```
SELECT column_name AS alias_name
FROM table_name
WHERE [condition];
```

Example: To select ROLLNO as STUDENT_ROLLNO and NAME as STUDENT_NAME from our Students table

```
SELECT ROLLNO AS STUDENT_ROLLNO, NAME AS STUDENT_NAME
FROM STUDENTS
WHERE AGE > 20;
```

To learn more such advanced topics in SQL at your own pace, i would suggest to refer the tutorials point's

Different Database Systems

Some of the most popular database systems along with MySQL are:

1. Oracle 12c

2. Microsoft SQL Server

3. PostgreSQL

4. MongoDB

5. MariaDB

6. DB2

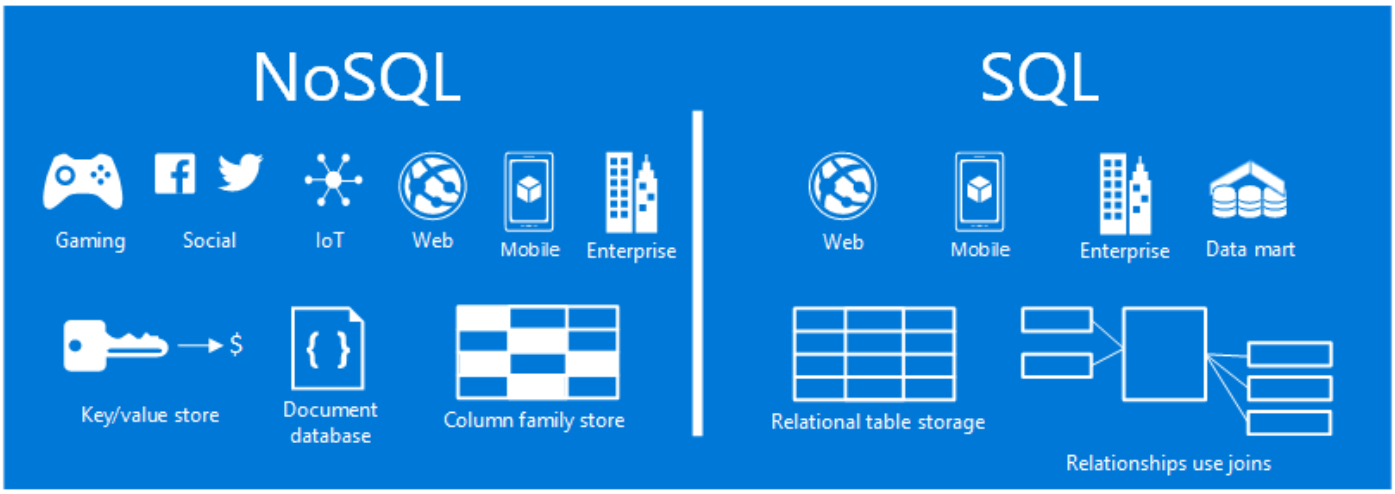
7. SAP HANA

SQL vs NoSQL

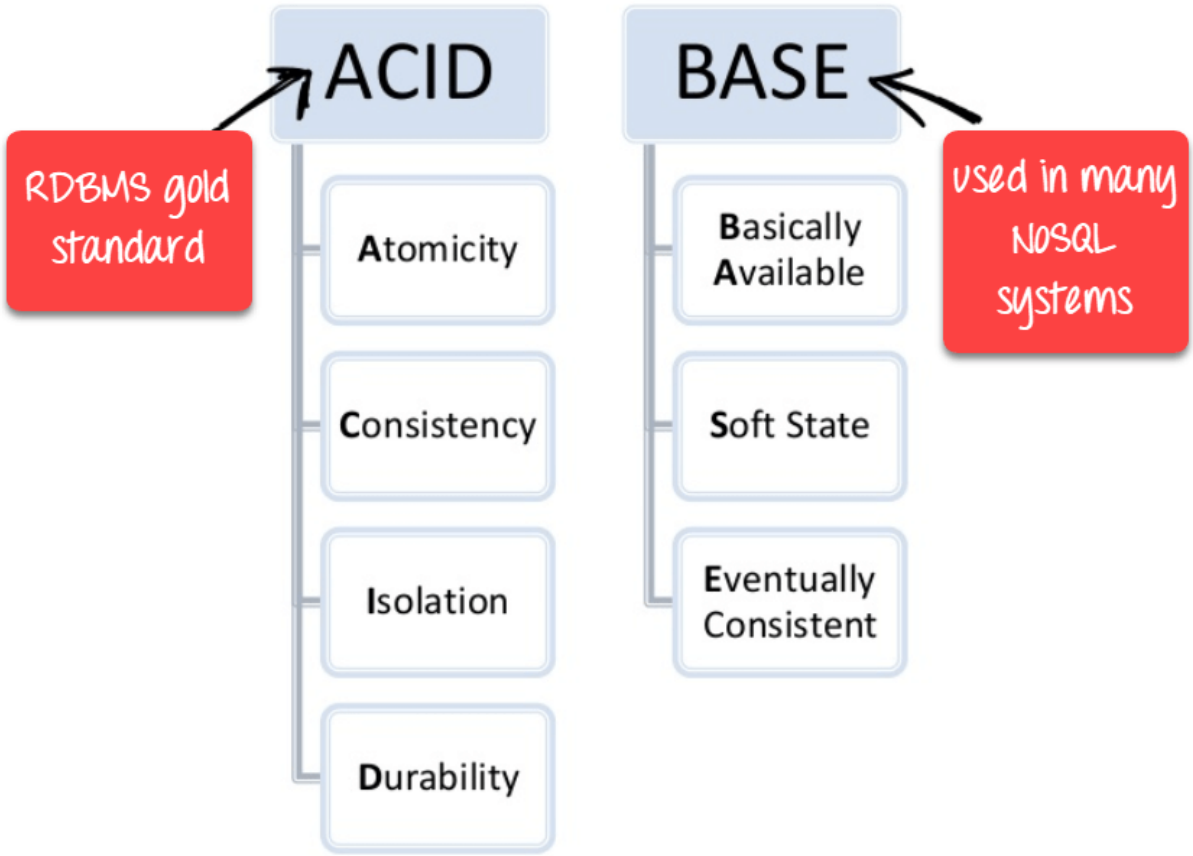
The five critical differences of SQL vs NoSQL:

1. SQL databases are relational, NoSQL are non-relational.
2. SQL databases use structured query language and have a predefined schema. NoSQL databases have dynamic schemas for unstructured data.
3. SQL databases are vertically scalable, NoSQL databases are horizontally scalable.
4. SQL databases are table based, while NoSQL databases are document, key-value, graph or wide-column stores.

5. SQL databases are better for multi-row transactions, NoSQL are better for unstructured data like documents or JSON.



When it comes to choosing a database, one of the biggest decisions is picking a relational (SQL) or non-relational (NoSQL) data structure. While both are viable options, there are key differences between the two that users must keep in mind when making a decision.



When use SQL?

- SQL is the easiest language used to communicate with the RDBMS
- Analyzing behavioral related and customized sessions
- Building custom dashboards
- It allows you to store and gets data from the database quickly
- Preferred when you want to use joins and execute complex queries

When use NoSQL?

- When ACID support is not needed
- When Traditional RDBMS model is not enough
- Data which need a flexible schema
- Constraints and validations logic not required to be implemented in database
- Logging data from distributed sources
- It should be used to store temporary data like shopping carts, wish list and session data

To learn more about the difference between SQL and NoSQL in brief, refer the following blog:
https://www.xplenty.com/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%20are%20relational%2C%20NoSQL,dynamic%20schemas%20for%20uns
(https://www.xplenty.com/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%20are%20relational%2C%20NoSQL,dynamic%20schemas%20for%20uns

That's all for the day, go through the referred blog / video tutorials, learn and play with the topics on your own database before catching up with the forward topics being covered in the series.

On Day 7, We will learn about Data Warehousing and Data Minining and it's importance in Data Science.