

SQL Notes - TheTestingAcademy (Pramod Sir)

What is the Database?

A collection of data that is organized and stored in a structured manner, allowing for efficient retrieval, manipulation, and analysis.

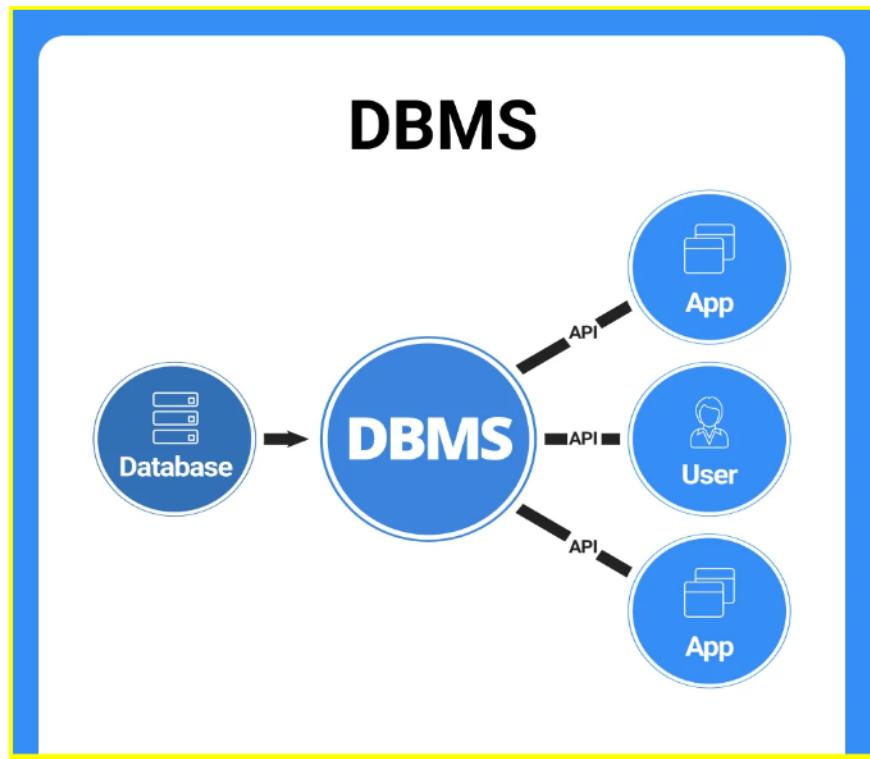
- Collection of Related Information
 - Phone Book
 - Shopping List
 - To-do list
 - Your Best Friends
- Databases can be stored in different ways
 - ON paper
 - In your mind
 - On A COMPUTER
 - This Google Document
 - Comment Sections
- Computers can store databases in a better way than anything else.
 - E.g. Old bank they kept data in paper What happened?
 - Amazon.com Products and Credits Cards, Users and Media, images and many more from the last 28 + years. (1994)
 - Security, Backup and Large amount of the database.

It typically consists of tables of information, with each table containing rows (records) and columns (fields) that represent different types of data.

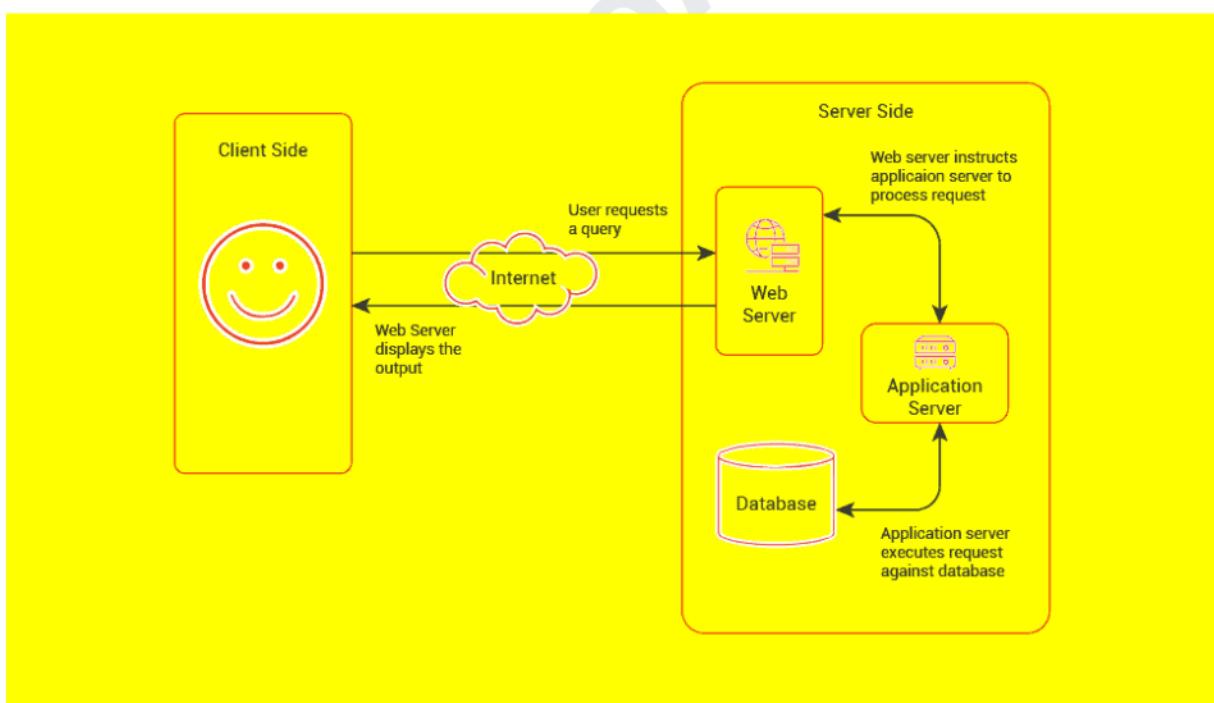
What is a Database Management System?

- A database management system (DBMS) is a software program that is used to create, manage, and manipulate databases.
- Handles Security, Backups, Import and Export Data
- Concurrency, Interact with Software Applications.
- Manipulate databases with Programming Languages.

Some examples of DBMSs include Oracle, MySQL, and Microsoft SQL Server.



Application Architecture



CRUD

Create, Read, Update and Delete



Types of Databases

- Relational Database
- Non-Relational (No SQL, Not just SQL)

Relational Database :

A relational database is a type of database that organizes data into **tables of rows and columns**, with each table containing data that is related to other tables through key fields.

This allows for **efficient querying and manipulation of the data**, and is the most common type of database in use today.

Non Relational Database :

A non-relational database, also known as a NoSQL database, is a type of database that is designed to handle large amounts of unstructured data and is not organized into tables of rows and columns.

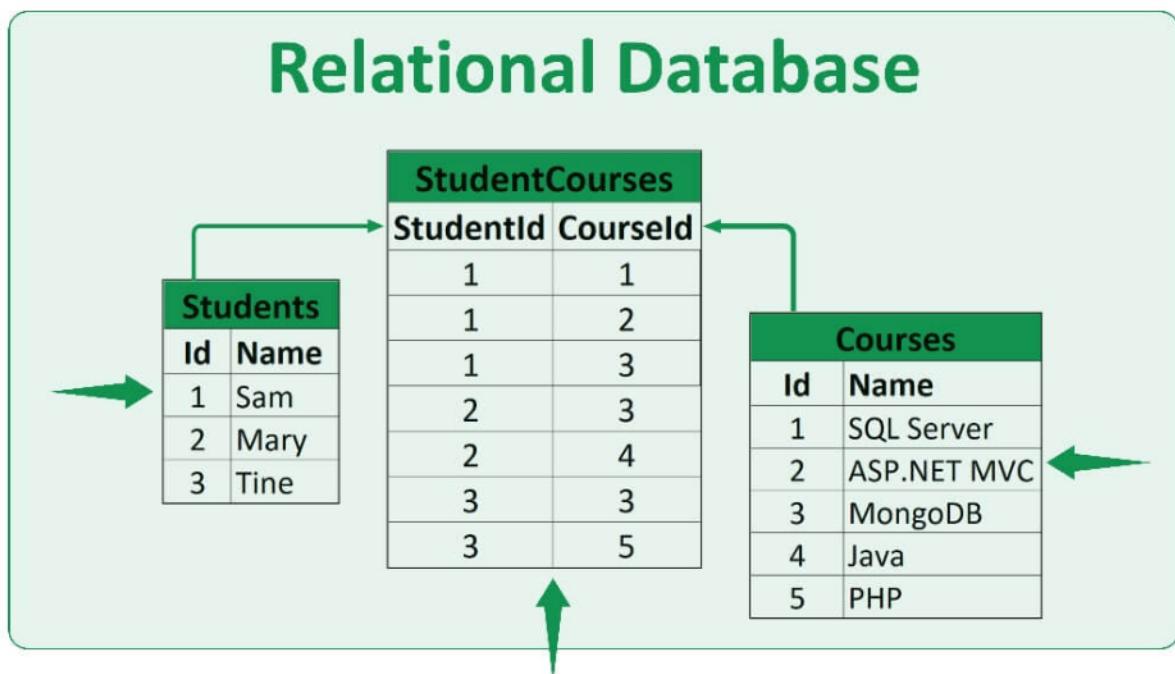
Instead, data is stored as **documents, key-value pairs, or other structures**, and is accessed through APIs or other interface methods. NoSQL databases are often used in applications that require high performance and scalability, and are well-suited for handling big data and real-time data processing. **JSON, XML, Graphs, Flexible Tables, Key-value**.

Installation of the Tools

- Install MySQL on the System
 - <https://dev.mysql.com/downloads/installer/>
 -
- Install DB Gate.
 - <https://dbgate.org>

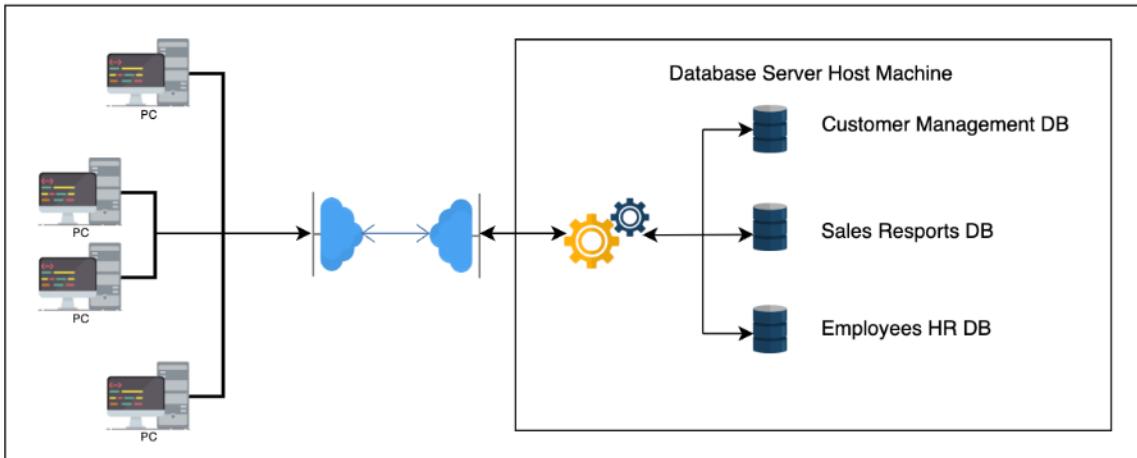
Relational Database

<https://drawsql.app/teams/k1-1/diagrams/student-and-course>



What is SQL?

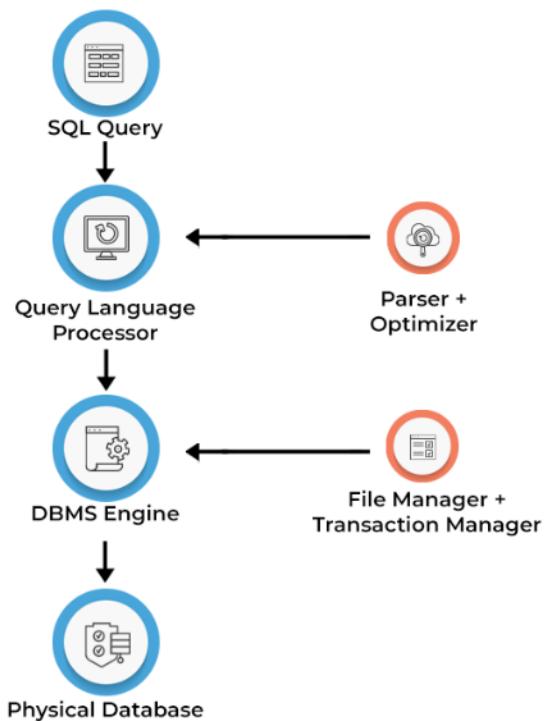
SQL is a standard programming language used to manage and change data in relational databases.



Client-server model

- Standardized language for interacting with RDBMS
- Used to perform CRUD
- Used to perform Admin type tasks like Backup, Recover etc
- Used to define tables and structures.

HOW DOES SQL WORK?



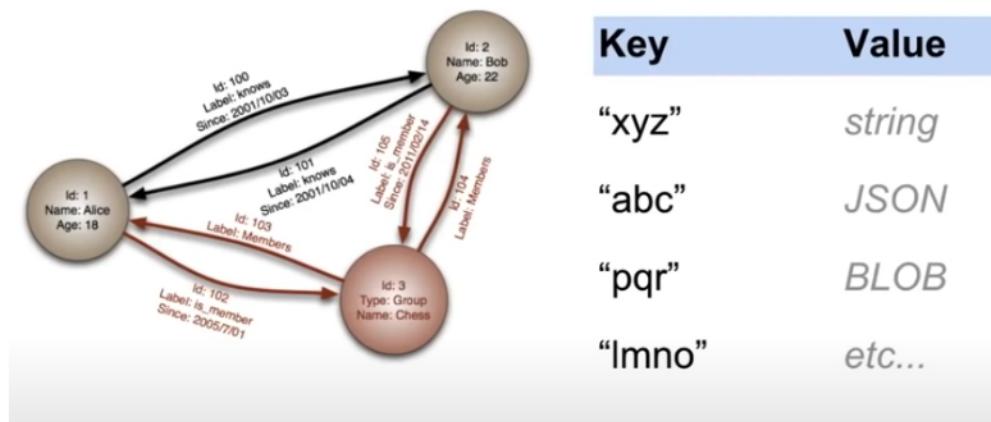
Non-Relational Databases (no SQL, Not just SQL)

A type of database that is designed to handle large amounts of unstructured data and is not organized into tables of rows and columns like a traditional relational database.

Instead, data is stored as documents, key-value pairs, or other structures, and is accessed through APIs or other interface methods.

- Most of the NRDMS will implement their own language for performing the CRUD.
- No standard language.
-

Examples of NoSQL databases include **MongoDB**, **Cassandra**, and **Redis**.



Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

Relational

Posts (id, Title)

1	Title		
Comments	01	1	Comment 1
	02	1	Comment 2

Jelvix

Non-relational

Posts (id, Title, Comments/Image)

1	Title	Comment 1
		Comment 2
		Comment 3
2	Title 2	Image

Source: Michal Bialecki Blog

jelvix.com

Database Queries

- Queries are requests made to the database management system for specific information.
- As we have more data, it is difficult to get information, so queries are used.
- e.g Google is a search query.
- Database queries are commands used to retrieve, manipulate, or modify data in a database.
- Queries are typically written in Structured Query Language (SQL) or a similar programming language, and are used to specify the criteria for selecting, updating, or deleting data in a database.

Here are some examples of database queries written in SQL:

- **Select all rows from a table:**
- `SELECT * FROM table_name;`
- **Select specific columns from a table:**
- `SELECT column_1, column_2 FROM table_name;`
- **Select rows based on a condition:**
- `SELECT * FROM table_name WHERE column_name = 'value';`
- **Select rows based on multiple conditions:**

- `SELECT * FROM table_name WHERE column_1 = 'value' AND column_2 = 'value';`
 - **Update data in a table:**
 - `UPDATE table_name SET column_name = 'new_value' WHERE column_name = 'old_value';`
 - **Delete rows from a table:**
 - `DELETE FROM table_name WHERE column_name = 'value';`
 - **Join two tables:**
 - `SELECT * FROM table_1 INNER JOIN table_2 ON table_1.column_name = table_2.column_name;`
-

Similar to Excel Sheet we have Databases

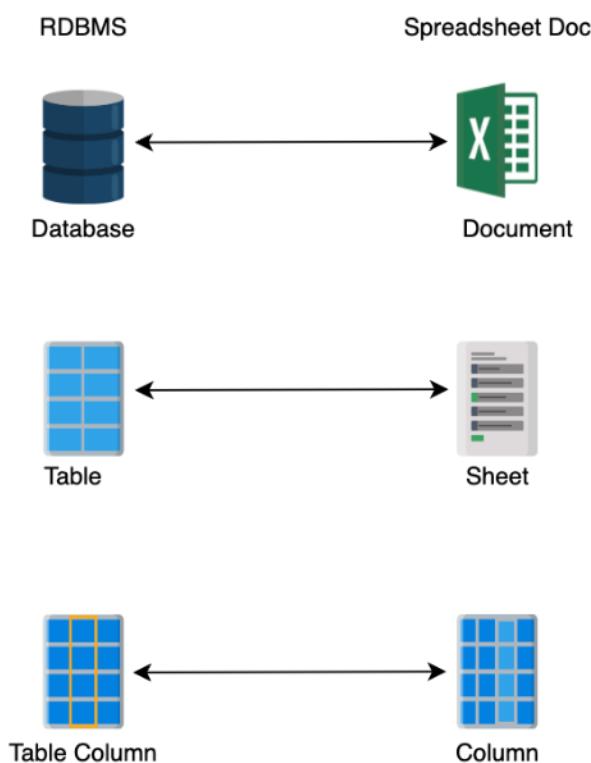


Table Structure

1. Row
2. Column
3. Attribute
4. Primary Key
5. Tuple / Record

Row: A row, also known as a record or tuple, is a single entry in a database table. It consists of a set of values that represent different attributes or fields of a single entity, such as a person or product.

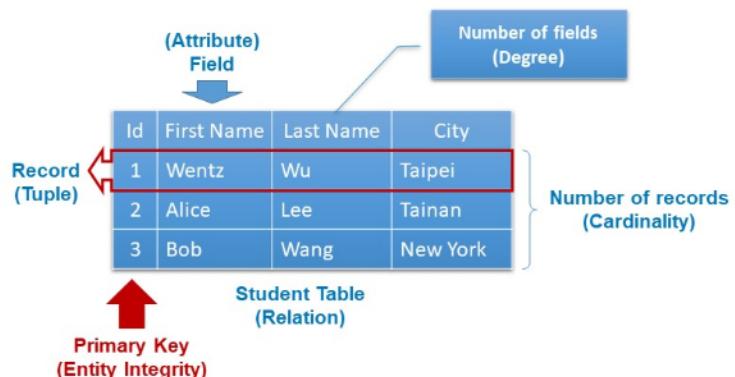
Column: A column, also known as a field or attribute, is a vertical set of values in a database table. Each column represents a different type of data, such as a person's name or a product's price.

Attribute: An attribute is a piece of data or a characteristic of an entity represented in a database. In a table, each column represents a different attribute, and each row represents a single instance of that attribute.

Primary key: A primary key is a field in a database table that uniquely identifies each row. It is typically used as the main reference field when linking tables together, and must be unique across all rows in the table.

Tuple / record: A tuple, also known as a record, is a row in a database table. It consists of a set of values that represent different attributes or fields of a single entity.

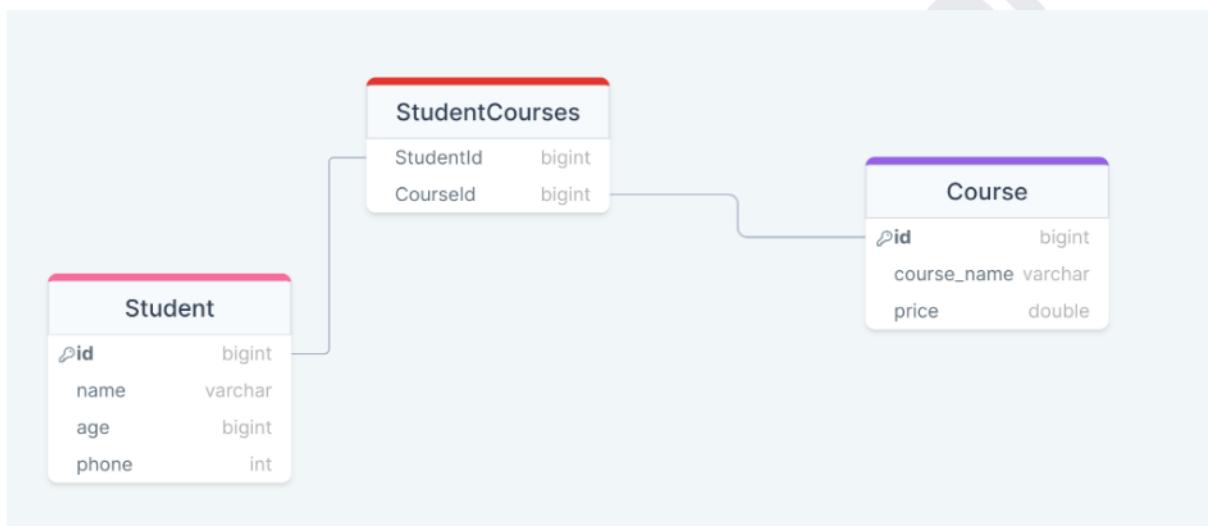
RDBMS - Table



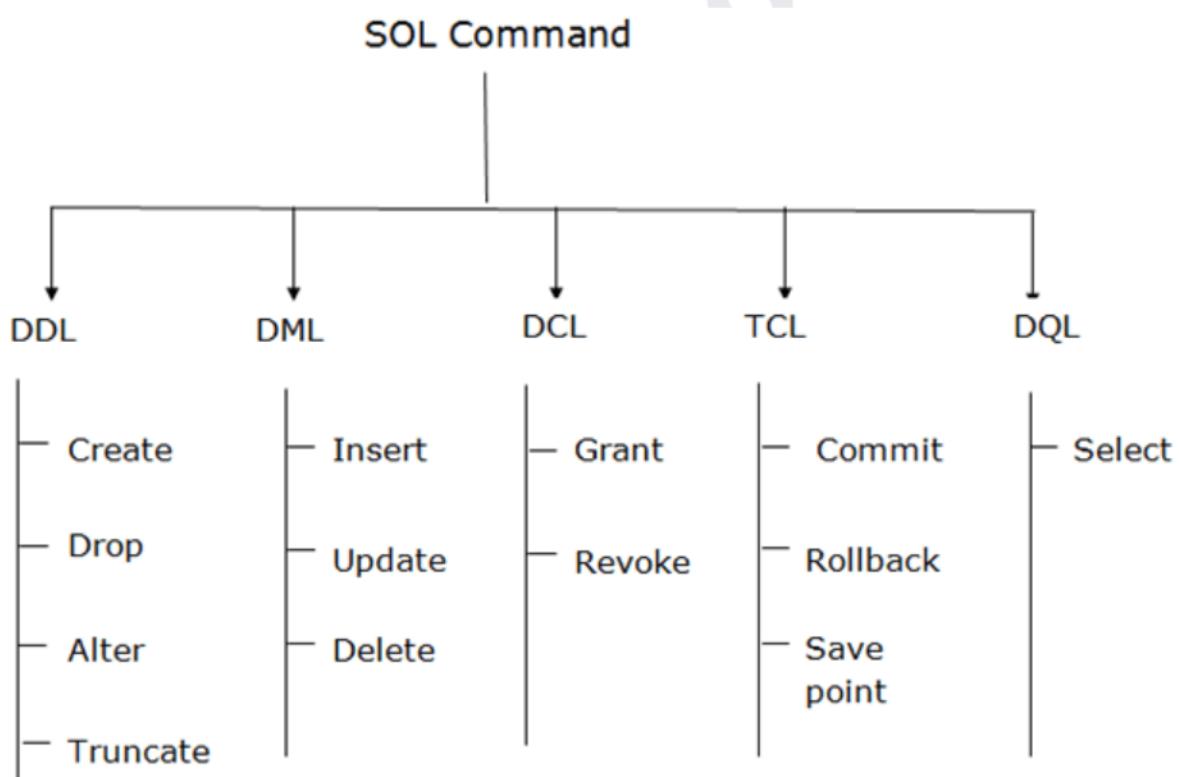
o

Relationships:

- One-to-one (1-1): One business entity is related to another single business entity.
 - E.g Order to 1 customer
- One-to-many (1-n): One business entity is related to many other business entities.
 - E.g Customer to Many Orders
- Many-to-many (m-n): Many business entities are related to many other business entities.
 - Many Student to Many Courses



SQL Commands



Data definition language (DDL)

- This is the set of SQL statements that describe the structure of the data to the server.
This is how the data has to be.

- It is used to create and modify the structure of database objects in the database.

Examples of DDL statements include CREATE, ALTER, and DROP.

Data manipulation language (DML):

This is the set of SQL statements that tell the server what we want to do with the data. Generally, this is limited to creating, retrieving, updating, and delete commands.

Keys

1. Primary Key
2. Natural Key
3. Foreign Key

Keys in Databases

What is the Primary Key?

A primary key is a field or set of fields that uniquely identifies each record in a table.

What is a Natural Key?

A natural key is a unique identifier that is inherent to the data, rather than being artificially assigned.

E.g - a social security number or passport number would be a natural key for identifying a person in a table.

What is a Foreign key?

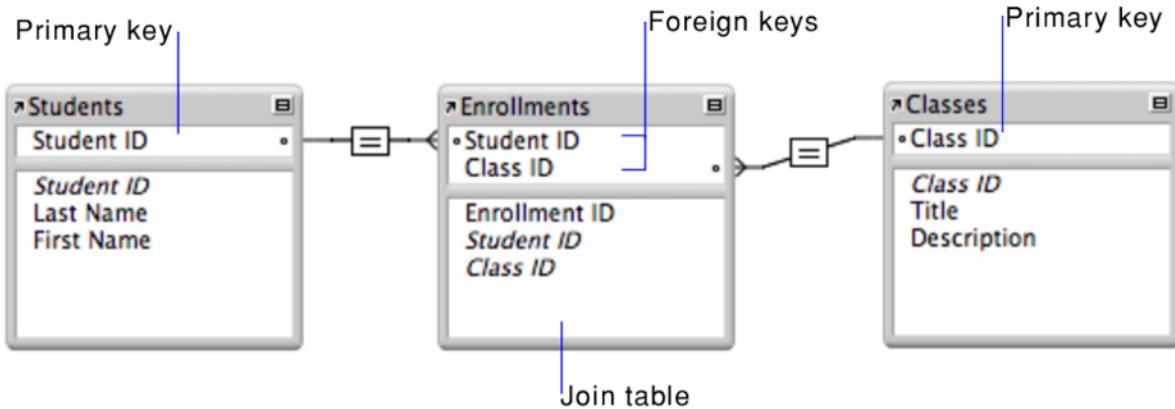
A foreign key is a field or set of fields in a table that references the primary key of another table.

It is used to establish a link between the data in two tables and enforce referential integrity

Referential integrity

Referential integrity is a set of rules that ensure the consistency and accuracy of data in a relational database by enforcing relationships between tables

	Primary Key	Foreign Key
1	Primary Key Can Not Accept Null Values.	Foreign Key Can Accept Multiple Null Values.
2	Only One Primary Key in a Table.	More than One Foreign Key in a Table.
3	Primary Key Uniquely Identify a Record in the Table	Foreign Key is a Field in the Table that is Primary Key in Another Table
4	Primary Key is Clustered Index. Ex: <pre>CREATE TABLE [country] ([id] INT IDENTITY (1, 1) NOT NULL, [name] VARCHAR (50) NOT NULL, UNIQUE NONCLUSTERED ([name]), CONSTRAINT [PK_country] PRIMARY KEY CLUSTERED ([id]));</pre>	Foreign Key is Non-Clustered Index. Ex: <pre>CREATE TABLE [reg] ([country] VARCHAR (50) NOT NULL, [name] VARCHAR (50) NOT NULL, CONSTRAINT [FK_reg_country] FOREIGN KEY ([name]) REFERENCES [dbo].[country] ([name]));</pre>

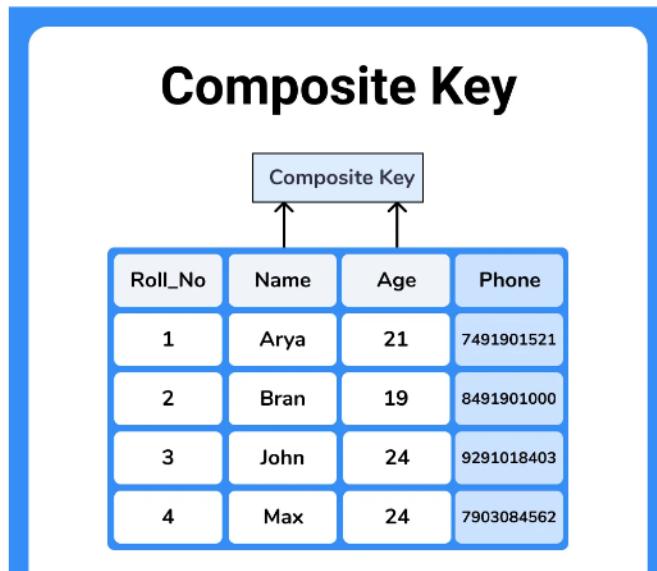


What is a composite key?

It is a type of primary key that is composed of multiple attributes to identify the unique records in a table.

```
CREATE TABLE Flights (
    FlightNumber VARCHAR(6) NOT NULL,
    DepartureDate DATE NOT NULL,
    DepartureAirport VARCHAR(3) NOT NULL,
    DepartureTime TIME NOT NULL,
```

```
        ArrivalTime TIME NOT NULL,  
        PRIMARY KEY (FlightNumber, DepartureDate, DepartureAirport)  
    );
```

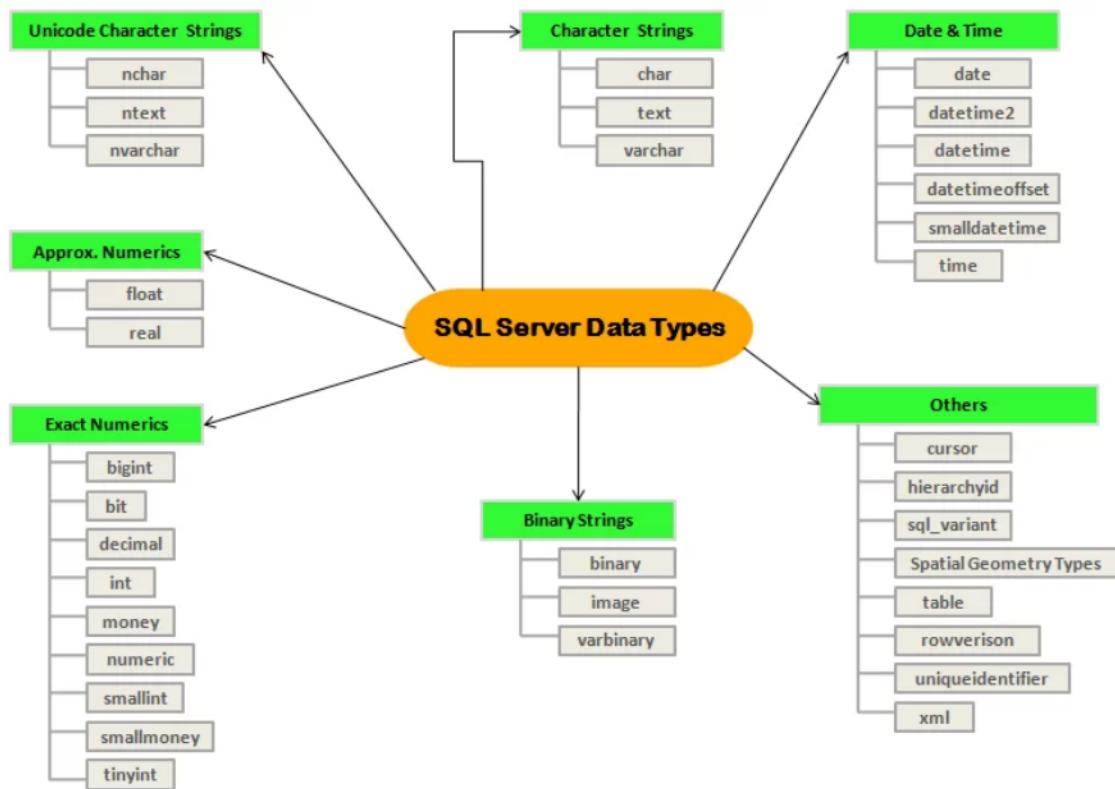


Data Types in SQL

Some of the most commonly used data types in MySQL are:

- **INT:** This data type is used to store integer values. It can be specified as INT or INTEGER, and can be defined with various widths, such as TINYINT, SMALLINT, MEDIUMINT, INT and BIGINT
- **DECIMAL:** This data type is used to store decimal (fixed-point) numbers. The precision and scale can be specified in the definition, for example DECIMAL(10,2)
- **FLOAT:** This data type is used to store floating-point numbers. It can be specified with optional precision, such as FLOAT(7,3)
- **DOUBLE:** This data type is used to store double precision floating-point numbers.
- **VARCHAR:** This data type is used to store variable-length strings of characters. The maximum length of the string can be specified in the definition, for example VARCHAR(255)
- **CHAR:** This data type is used to store fixed-length strings of characters. The maximum length of the string can be specified in the definition, for example CHAR(10)
- **DATE:** This data type is used to store date values in the format of 'YYYY-MM-DD'
- **DATETIME:** This data type is used to store date and time values in the format of 'YYYY-MM-DD HH:MM:SS'

- **TIMESTAMP:** This data type is used to store timestamp values that are based on the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC)
- **BLOB and TEXT:** These data types are used to store binary and text data, respectively. They can be used to store large amounts of data, such as images or text documents.



In addition to these basic data types, MySQL also provides other data types like ENUM, SET, YEAR, TIME, BIT etc.

These data types can be used to store different types of data such as enumerated values, sets of values, year values, time values, Bit values etc.

When creating a table, it's important to use the correct data type for each column to ensure the data is stored correctly and to optimize the performance of the database.

https://www.w3schools.com/sql/sql_datatypes.asp

SQL Operations

1. Select Command
2. Select Specific Column
3. Select DISTINCT Column
4. Select with WHERE
5. Select with WHERE with AND, OR and NOT Operators
6. ORDER BY - sort the result-set in ascending or descending order.
7. Is Not Null - **WHERE Address IS NULL;**
8. SELECT TOP - LIMIT number;
9. MAX, SELECT MIN(salary) FROM employee.
10. IN operator allows you to specify multiple values in a WHERE clause
11. BETWEEN – WHERE column_name BETWEEN value1 AND value2;
12. GROUP BY
13. SQL HAVING
14. SQL EXISTS Operator

Try SQL

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_join_left

Wildcard Characters

Wildcard Characters in SQL Server

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
-	Represents any single character within the specified range	c[a-b]t finds cat and cbt

Data insertion and CRUD

```
insert into customers (name) values ('Pramod Dutta');
```

Showing data and a specific

```
select * from customers;  
select name from customers;
```

Where clause with operators, Like,

```
select * from customers where name = 'Pramod';  
select * from customers where id = 2;  
select * from customers where id = 2 or name = 'Pramod';
```

Count the items

select count(*) from <table_name> where <where_clause>` statement.

```
select count(*) from customers;  
select count(*) from customers where name = 'Pramod';
```

Deleting Rows from Tables

```
delete from customers where name = 'Pramod';
```

Update Rows

```
update customers set name = 'oo' where id = 2;
```

Like Operator

```
select * from customers where name like '%oo';
```

Operator	Description	Example
=	Equal to	(x=y) is not true
!=	Equal or not	(x!=y) is true
< >	Not equal to	(x<>y) is true
>	Greater than	(x>y) is not true
<	Less than	(x<="" td="">
>=	Greater than or equal to	(x>=y) is not true
<=	Less than or equal to	(x<=y) is true
!<	Not less than	(x!<="" td="">
!>	Not greater than	(x!>y) is true

DROP DATABASE

```
DROP DATABASE databasename;  
DROP TABLE table_name;
```

ALTER TABLE

```
ALTER TABLE customers ADD Email varchar(255);
```

TRUNCATE TABLE

```
TRUNCATE TABLE table_name;
```

	DELETE	DROP	TRUNCATE
Usage	Removes rows from a table	Removes a table from the database / data dictionary	Removes all rows from a table
Type of command	DML	DDL	DDL
Rollback	Can be rolled back	Cannot be rolled back	Cannot be rolled back
Rows, indexes and privileges	Only table rows are deleted	Table rows, indexes and privileges are deleted	Table rows are deleted
DML trigger firing	Trigger is fired	No triggers are fired	No triggers are fired
Performance	Slower than TRUNCATE	Quick but could lead to complications	Faster than DELETE
Undo space	Uses "undo" space	Does not use "undo" space	Uses "undo" space, but not as much as DELETE
Permanent deletion	Does not remove the record permanently.	Removes all records, indexes, and privileges permanently.	Removes the record permanently.
Can you write conditions using a WHERE Clause?	Yes	No	No
Row deletion	Deletes all or some rows	Deletes all rows	Deletes all rows

Mini Project CRUD

```
CREATE TABLE `customers` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `name` varchar(255) DEFAULT NULL,
    PRIMARY KEY (`id`)
);

describe customers;

insert into customers(name) values ("Pramod");
insert into customers(name) values ("Dutta");
insert into customers(name) values ("PraD");
insert into customers(name) values ("Quick");

select * from customers;

select * from customers where id = 2;

select * from customers where name like 'pra%';
```

SQL Constraints

SQL constraints are used to specify rules for the data in a table.

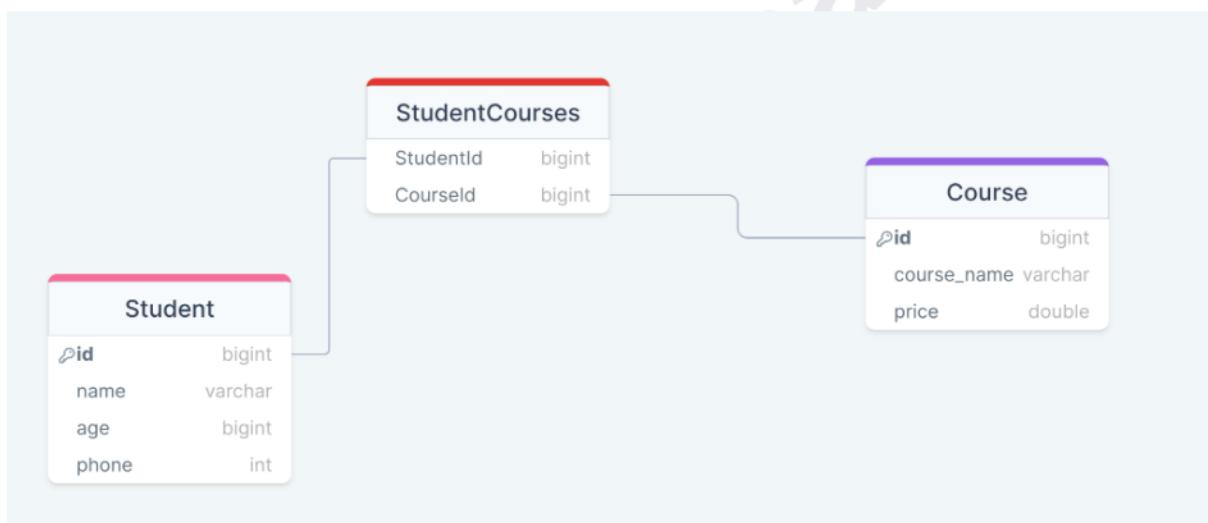
- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different

- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
 - **FOREIGN KEY** - Prevents actions that would destroy links between tables
 - **CHECK** - Ensures that the values in a column satisfy a specific condition
 - **DEFAULT** - Sets a default value for a column if no value is specified
 - **CREATE INDEX** - Used to create and retrieve data from the database very quickly
 - **AUTO_INCREMENT** - increment by own.
-

Project #1

Create this

1. Student with the Id, name and age and phone number , PK ID
2. Courses id, course_name, price PK ID
3. StudentCourses - sid, cid with PK.



Solution:

```

CREATE TABLE `Students` (
  `id` BIGINT AUTO_INCREMENT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `age` BIGINT NOT NULL,
  `phone` INT NOT NULL,
  CONSTRAINT `PRIMARY` PRIMARY KEY (`id`)
);
  
```

```

CREATE TABLE Courses (
  
```

```
    id bigint NOT NULL AUTO_INCREMENT,
    Name varchar(255) NOT NULL,
    Price int,
    PRIMARY KEY (id)
);

CREATE TABLE StudentCourses (
    StudentId BigInt NOT NULL,
    CourseId BigInt NOT Null
)

ALTER TABLE StudentCourses
ADD FOREIGN KEY (StudentId) REFERENCES Students(id);

ALTER TABLE StudentCourses ADD FOREIGN KEY (CourseId) REFERENCES
Courses(id)

INSERT INTO `Courses` (`Name`, `Price`) VALUES ('ATB','10000');

INSERT INTO `Students` (`name`, `age`, `phone`) VALUES
('Aju','32','989829323');
INSERT INTO `Students` (`name`, `age`, `phone`) VALUES
('Ajim','34','829329323');
INSERT INTO `Students` (`name`, `age`, `phone`) VALUES
('Pramod','32','239392323');

SELECT * from Students;
SELECT * from Courses;

SELECT * from StudentCourses;

INSERT INTO `StudentCourses` (`StudentId`, `CourseId`) VALUES ('1','1');
INSERT INTO `StudentCourses` (`StudentId`, `CourseId`) VALUES ('2','1');
INSERT INTO `StudentCourses` (`StudentId`, `CourseId`) VALUES ('3','2');

SELECT * from StudentCourses;
```

Project #2 Customer and Orders, with Products (CRM)

- Customer

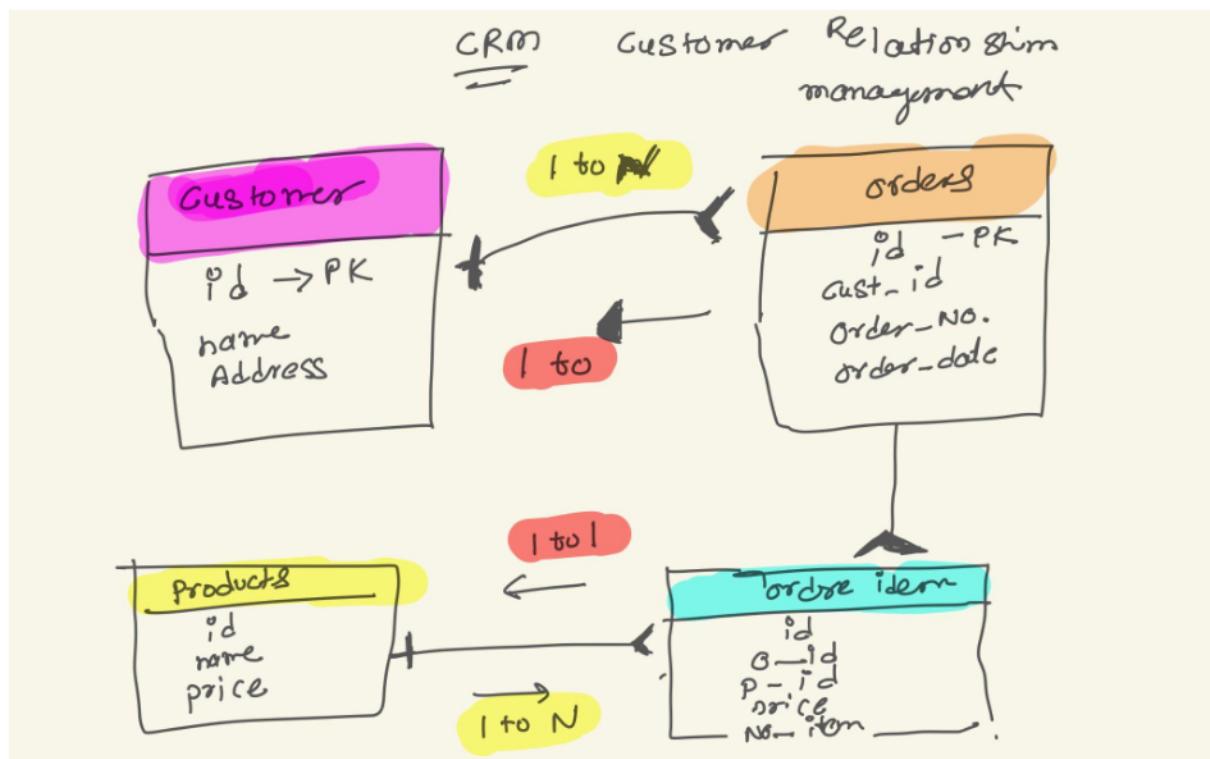
COLUMNS		<<	id	▼	name	▼	address	▼		
		Filter	:	▼	Filter	:	▼	Filter	:	▼
	<input checked="" type="checkbox"/> id	1			Pramod					
	<input checked="" type="checkbox"/> name	2			Pramod					
	<input checked="" type="checkbox"/> address	3			Dutta					
		4			PraD					
		5			Quick					
		6			Quick		ND			

- Product

COLUMNS		<<	id	▼	name	▼	price	▼		
		Filter	:	▼	Filter	:	▼	Filter	:	▼
	<input checked="" type="checkbox"/> id	1			ATB		139.00			
	<input checked="" type="checkbox"/> name	2	3		MTB		99.00			
	<input checked="" type="checkbox"/> price	3	4		APITB		99.00			

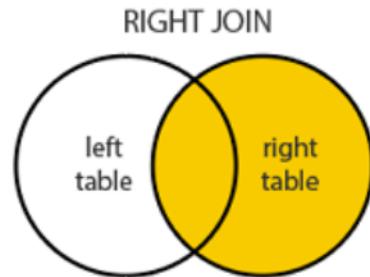
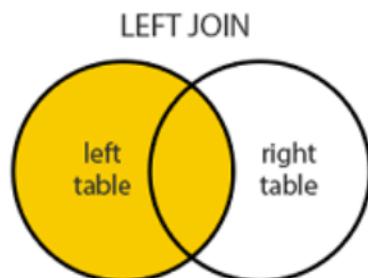
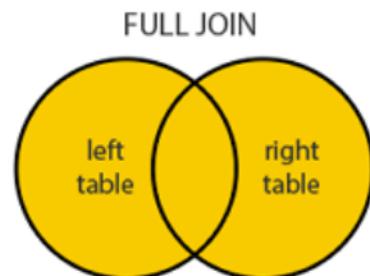
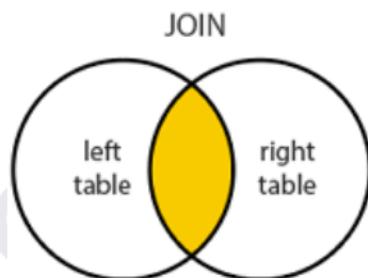
- Orders

COLUMNS		<<	id	▼	customers_id	▼	order_number	▼	ordered_at	▼
		Filter	:	▼	Filter	:	▼	Filter	:	▼
	<input checked="" type="checkbox"/> id	1	3	1			TTA01		2023-01-16 15:45:40	
	<input checked="" type="checkbox"/> customers_id	2	7	4			TTA02		2023-01-16 15:49:04	
	<input checked="" type="checkbox"/> order_number	3	9	3			TTA04		2023-01-16 16:18:43	



SQL Joins

- Join can combine information that exists in more than one table.



SQL Joins

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

- **(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

#How many customers do we have with orders?

```
select count(*) from customers join orders on orders.customers_id =  
customers.id;  
select count(distinct customers.id) from customers join orders on  
orders.customers_id = customers.id;
```

#How many customers do we have that haven't placed an order?

```
select * from customers left join orders on customers.id =  
orders.customers_id;
```

#Null also we don't want

```
select * from customers left join orders on customers.id =  
orders.customers_id where orders.id is NOT NULL;
```

#Who doesn't placed anything

```
select * from customers left join orders on customers.id =  
orders.customers_id where orders.id is NULL;
```

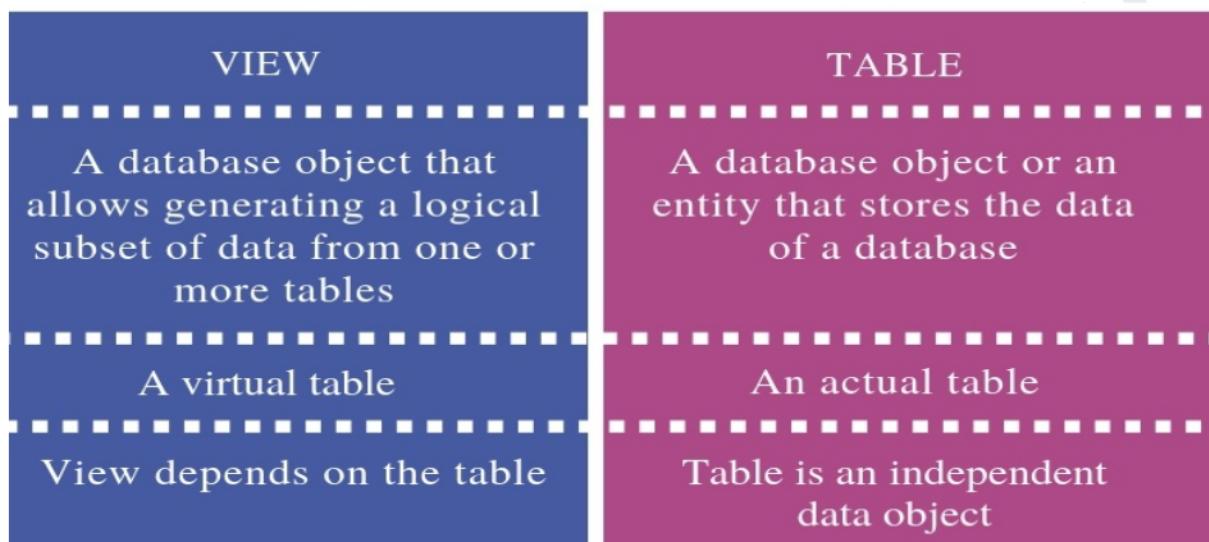
Self Join

```
SELECT A.name AS CustomerName1, B.name AS CustomerName2, A.id  
FROM customers A,customers B  
WHERE A.id <> B.id  
AND A.name = B.name;
```

Union

```
#SELECT column_name(s) FROM table1  
#UNION ALL  
#SELECT column_name(s) FROM table2;  
#Same columns  
  
select * from customers UNION ALL select * from customers;
```

Views



#VIEW

```
CREATE VIEW idg5 AS  
SELECT *  
FROM customers  
WHERE id < 5;  
  
select * from idg5;
```

- You can also **CREATE OR REPLACE VIEW**.
- Dropping a View - **DROP VIEW view_name;**

Null Functions:

The MySQL IFNULL() function lets you return an alternative value if an expression is NULL.

```
SELECT ProductName, UnitPrice * (UnitsInStock + IFNULL(UnitsOnOrder, 0))
```

```
FROM Products;
```

```
SELECT ProductName, UnitPrice * (UnitsInStock + COALESCE(UnitsOnOrder,  
0))  
FROM Products;
```

PROCEDURE

a collection of MySQL statements grouped together in a function that can be called on-demand with specific input parameters.

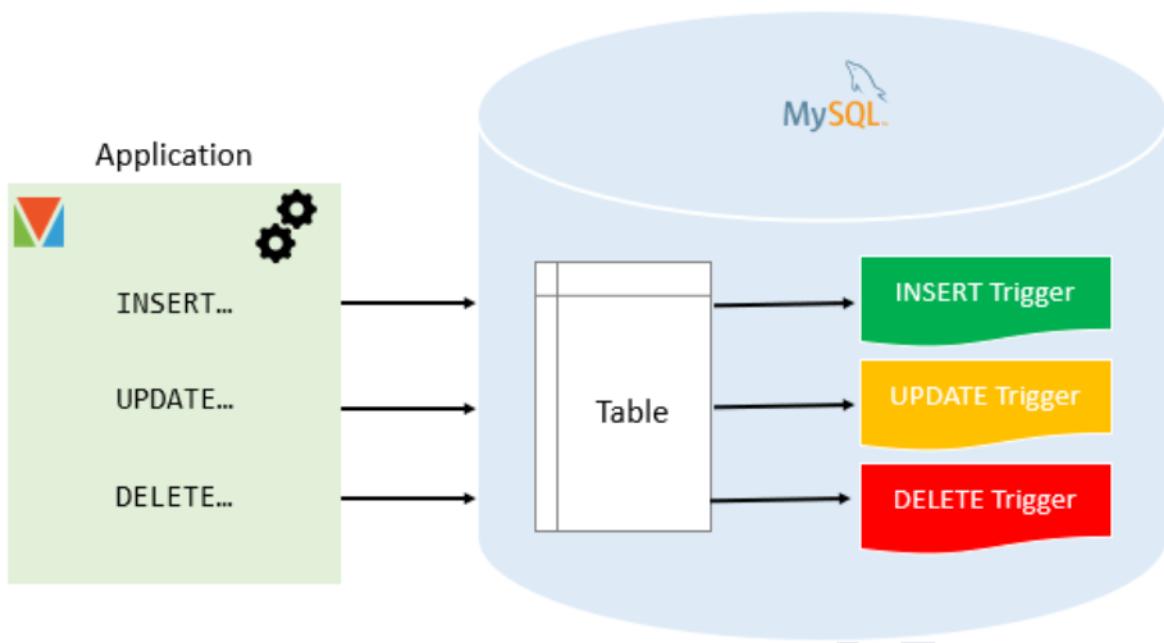
```
DELIMITER //  
  
CREATE PROCEDURE qq (IN arg1 INT)  
BEGIN  
    SELECT * from order_items where quantity > arg1;  
END  
  
DELIMITER ;
```

Call them as

```
#Set Value  
SET @q_value = 1;  
call qq(@q_value)
```

TRIGGERS

A MySQL trigger is a stored program (with queries) which is executed automatically to respond to a specific event such as insertion, updation or deletion occurring in a table.



```
CREATE TRIGGER NotNullAddress
{BEFORE | AFTER} {INSERT | UPDATE| DELETE }
ON table_name FOR EACH ROW
trigger_body;
```

```
SELECT * from customers;

insert into customers(name,address) values ("Test","NA");

#Trigger
delimiter //
CREATE TRIGGER before_address_update_null BEFORE INSERT ON customers
FOR EACH ROW
BEGIN
    IF (NEW.address = "NA") THEN
        SET NEW.address = "INDIA";
    END IF;
END
delimiter ;
```

Delete All

- delete from order_items;
- delete from orders;
- delete from customers;

Interview Interview Questions SQL

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

1. Write a query to identify a duplicate record.
2. Explain Group By and Order by
3. Union Intersect, Minus.
4. Difference between Joins.
5. Difference between subquery and co-related subquery.
6. Write a query to find out the 20th max salary.
7. Difference between NVL and coalesce. (Oracle)
8. Difference between Minus and Intersect clauses.
9. Difference between having and where clause
10. [Case statement usage](#)

Write a query to identify a duplicate record.

```
Select * from Students where count(name) > 1

SELECT COL1, COL2, ..., COLn, COUNT(*) FROM TABLE1 GROUP BY COL1,COL2, .., COLn
HAVING COUNT(*) > 1
```

Explain Group By and Order by

```
SELECT column1, column 2...
FROM table_name
WHERE [condition]
GROUP BY column1, column2
ORDER BY column1, column2;

SELECT * FROM Customers Group by City Order by City asc;
```

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_join_left

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

https://www.w3schools.com/sql/sql_isnull.asp

Practice SQL

here - <https://www.programiz.com/sql/online-compiler/>