

SQL

Data and Information

- **What is Data?**

- Data is a collection of raw, unorganised facts and details like text, observations, figures, symbols and descriptions of things etc. In other words, data does not carry any specific purpose and has no significance by itself. Moreover, data is measured in terms of bits and bytes – which are basic units of information in the context of computer storage and processing.

- **What is Information?**

- Information is processed, organised and structured data. It provides context for data and enables decision making. For example, a single customer's sale at a restaurant is data – this becomes information when the business is able to identify the most popular or least popular dish.

What is database?

- A **database** is an organized collection of data, so that it can be easily accessed and managed.

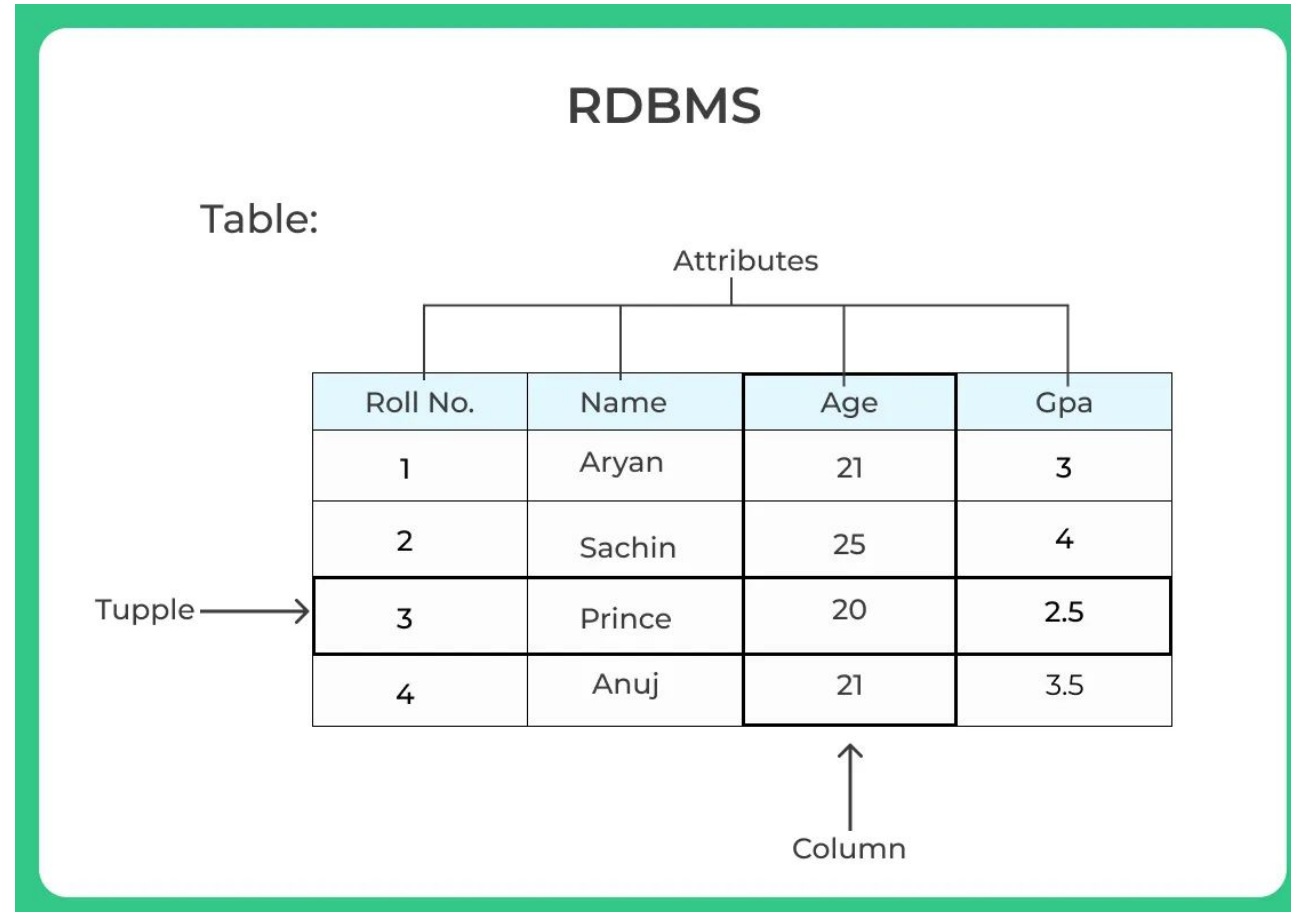
School



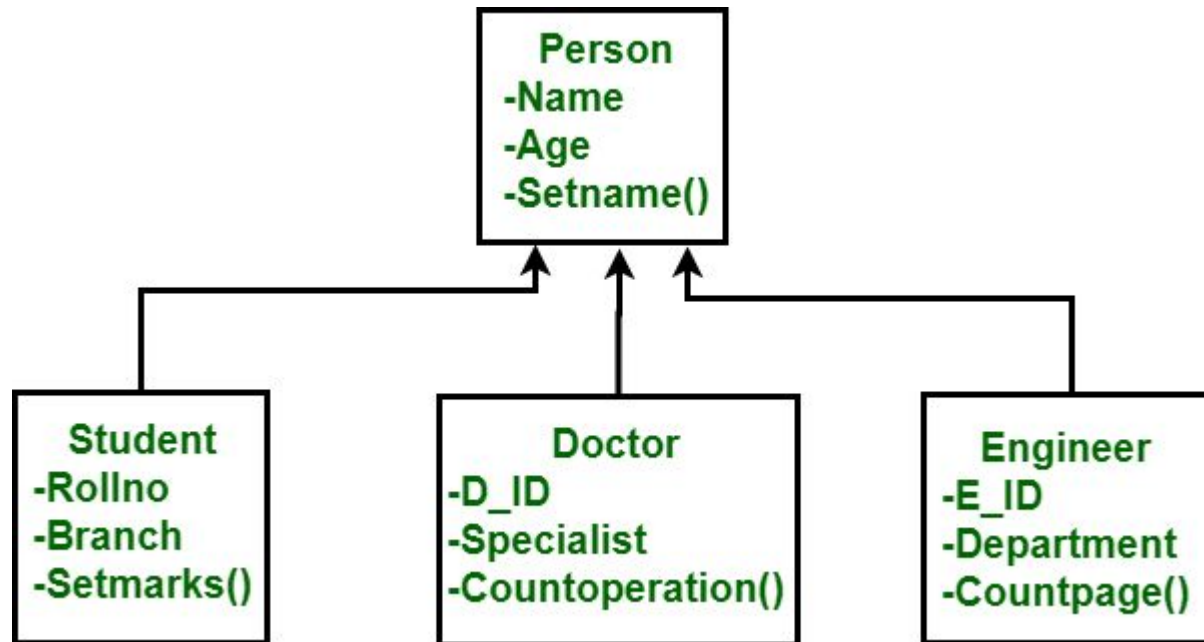
Types of database

- ● Relational Database Management System
- ● Object Oriented Database Management System
- ● Hierarchical Database Management System
- ● Network Database Management System.

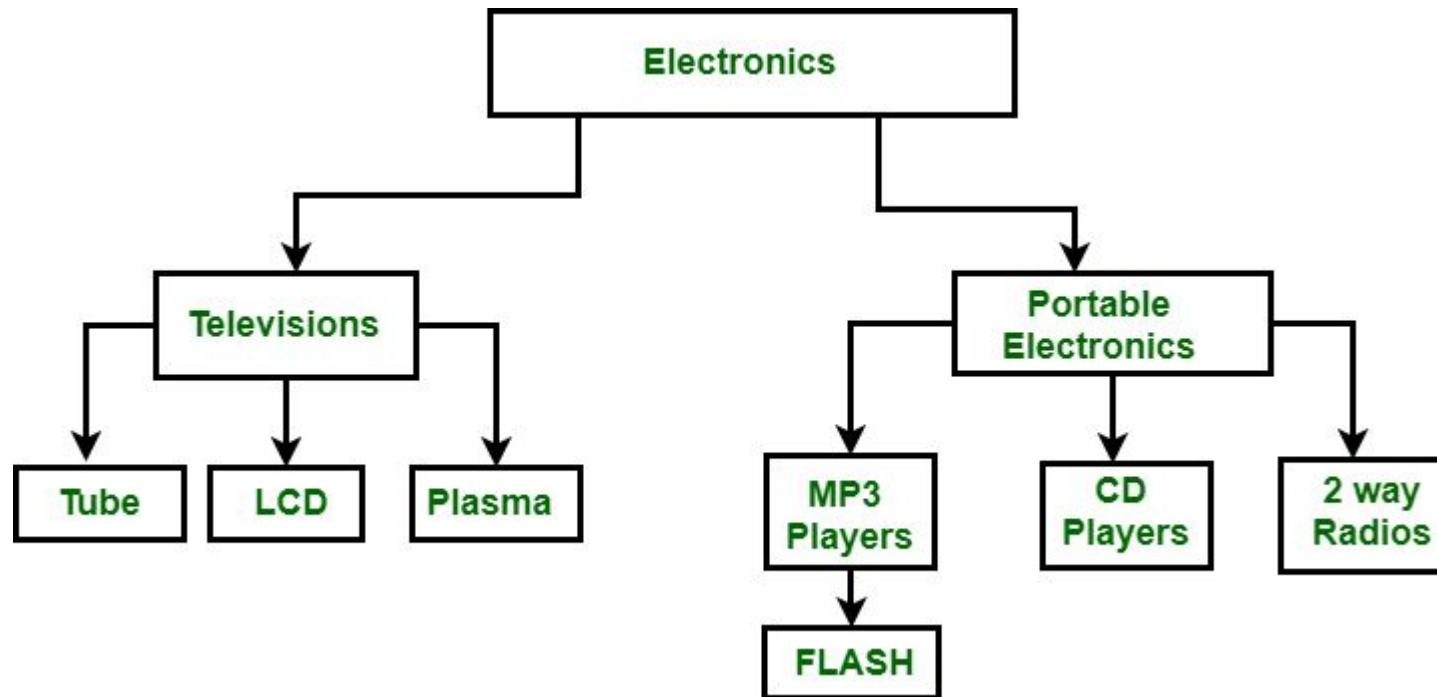
Relational database management system



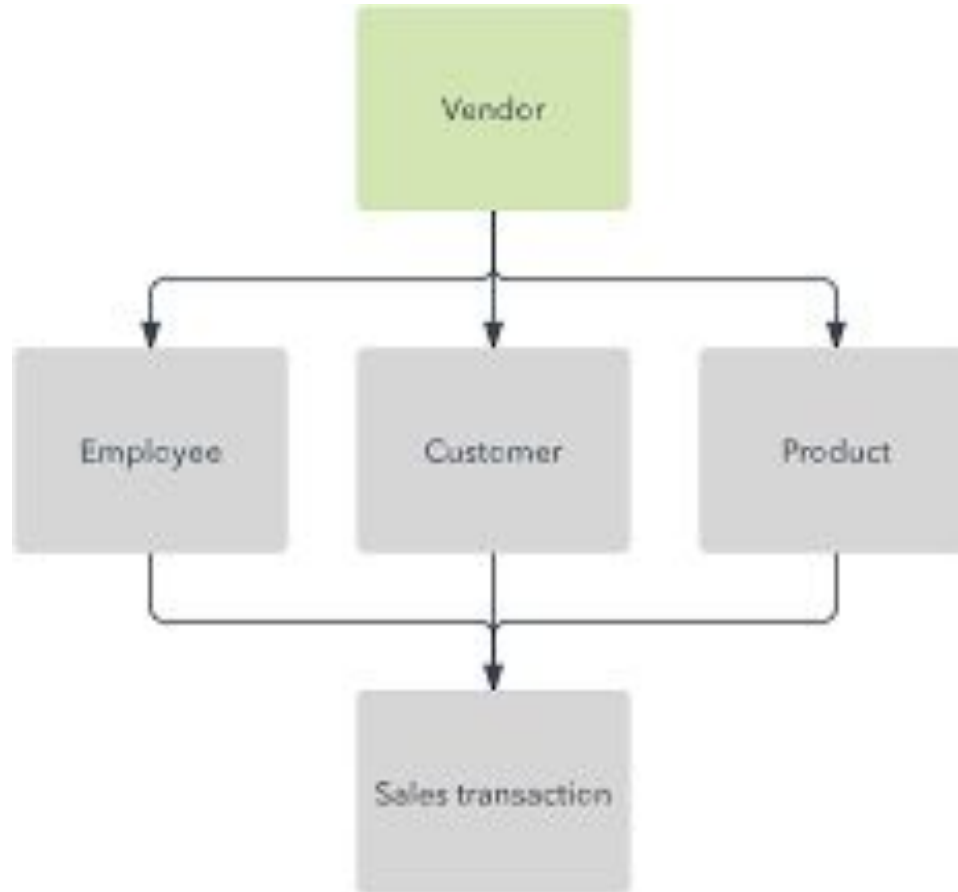
Object Oriented database management system



Hierarchical database management



Network based management system



RDBMS



ORACLE®

PostgreSQL



SYBASE®

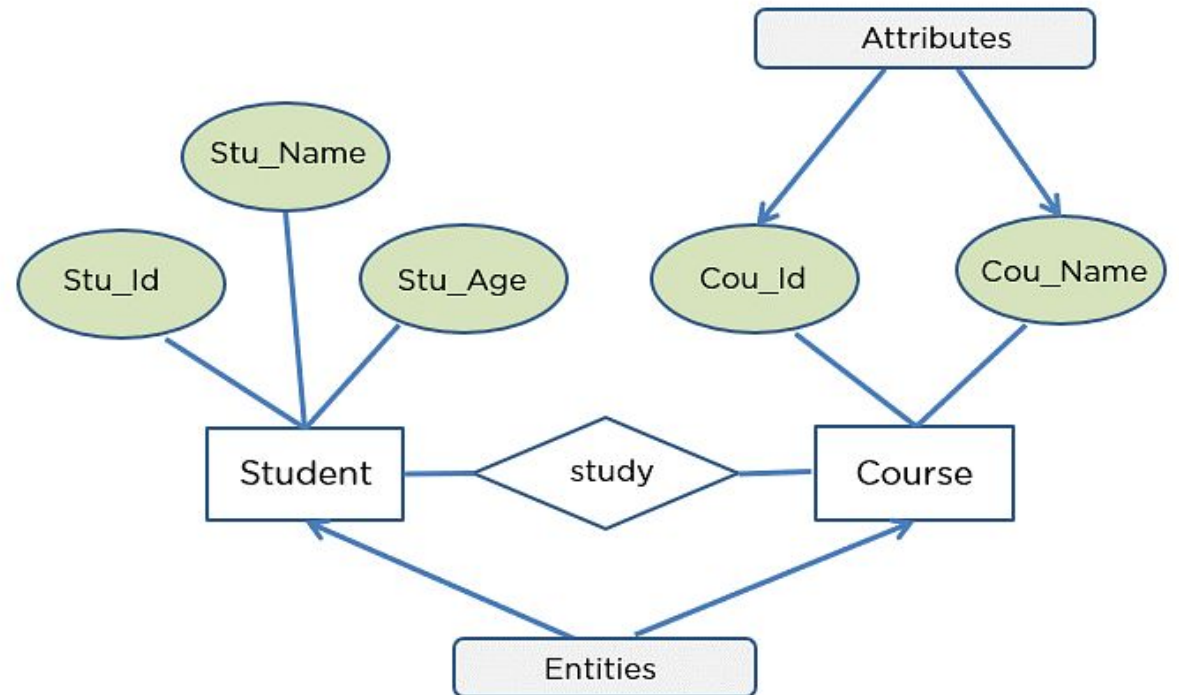


SQL?

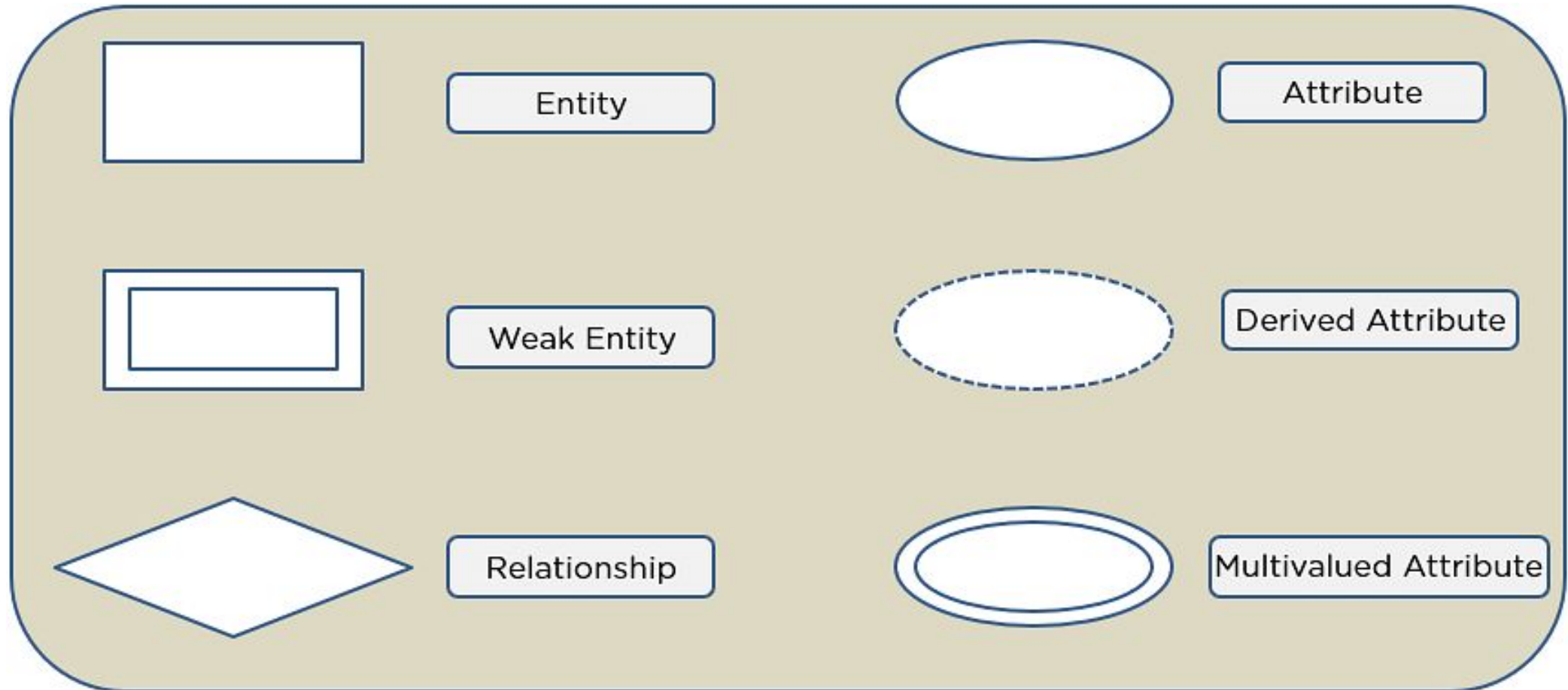
- SQL is a standard language for accessing and manipulating databases.
- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases

ER diagram

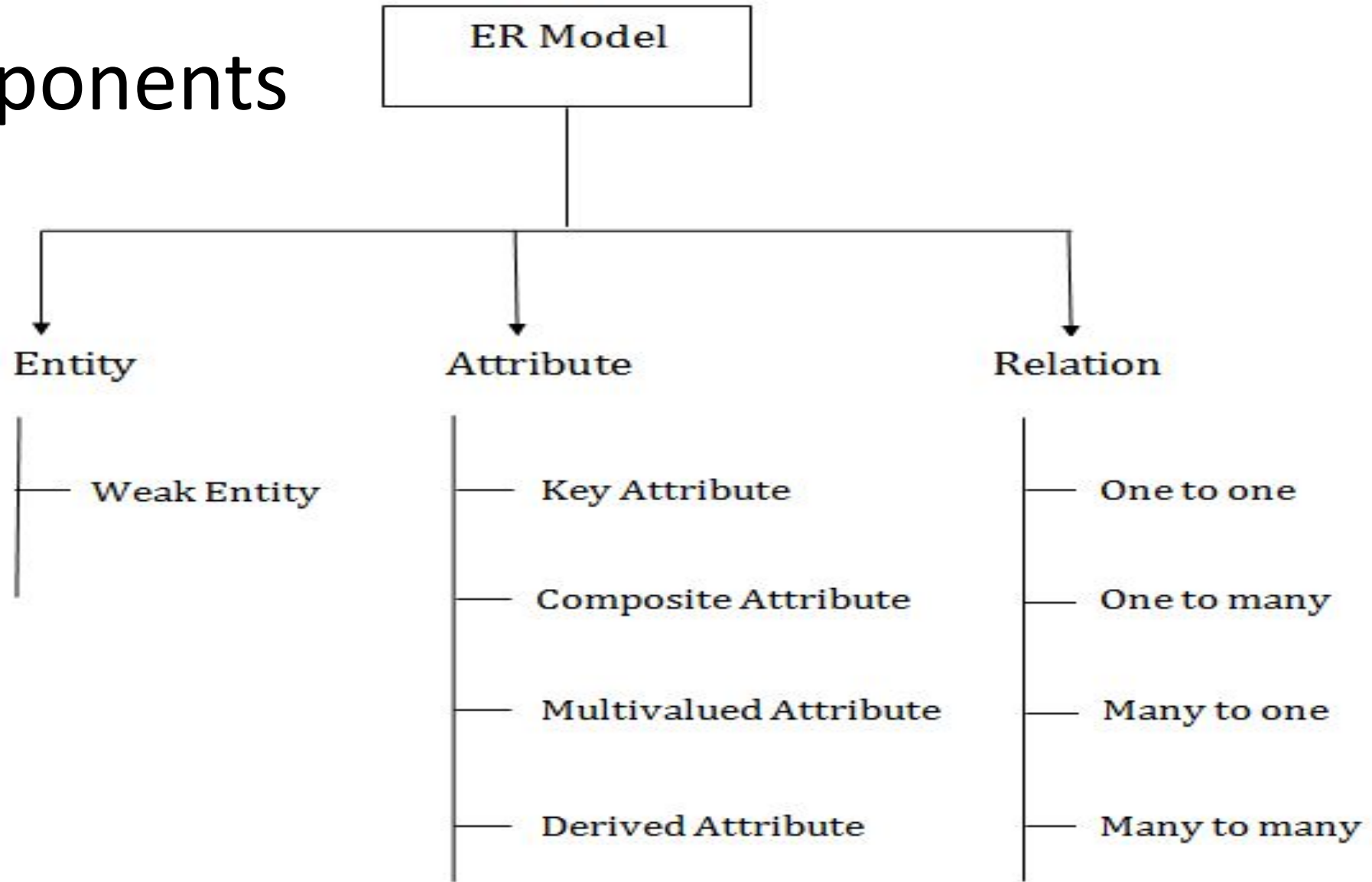
- An Entity Relationship Diagram is a diagram that represents relationships among entities in a database. It is commonly known as an ER Diagram. An ER Diagram in [DBMS](#) plays a crucial role in designing the database.



Shapes used in ER



Components



Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The <i>size</i> parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The <i>size</i> parameter specifies the maximum string length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The <i>size</i> parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The <i>size</i> parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters

TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LOBBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data

BIT(<i>size</i>)	A bit-value type. The number of bits per value is specified in <i>size</i> . The <i>size</i> parameter can hold a value from 1 to 64. The default value for <i>size</i> is 1.
TINYINT(<i>size</i>)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The <i>size</i> parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(<i>size</i>)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The <i>size</i> parameter specifies the maximum display width (which is 255)
MEDIUMINT(<i>size</i>)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The <i>size</i> parameter specifies the maximum display width (which is 255)
INT(<i>size</i>)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The <i>size</i> parameter specifies the maximum display width (which is 255)
INTEGER(<i>size</i>)	Equal to INT(<i>size</i>)
BIGINT(<i>size</i>)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The <i>size</i> parameter specifies the maximum display width (which is 255)

INTEGER(*size*) Equal to INT(*size*)

BIGINT(*size*) A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The *size* parameter specifies the maximum display width (which is 255)

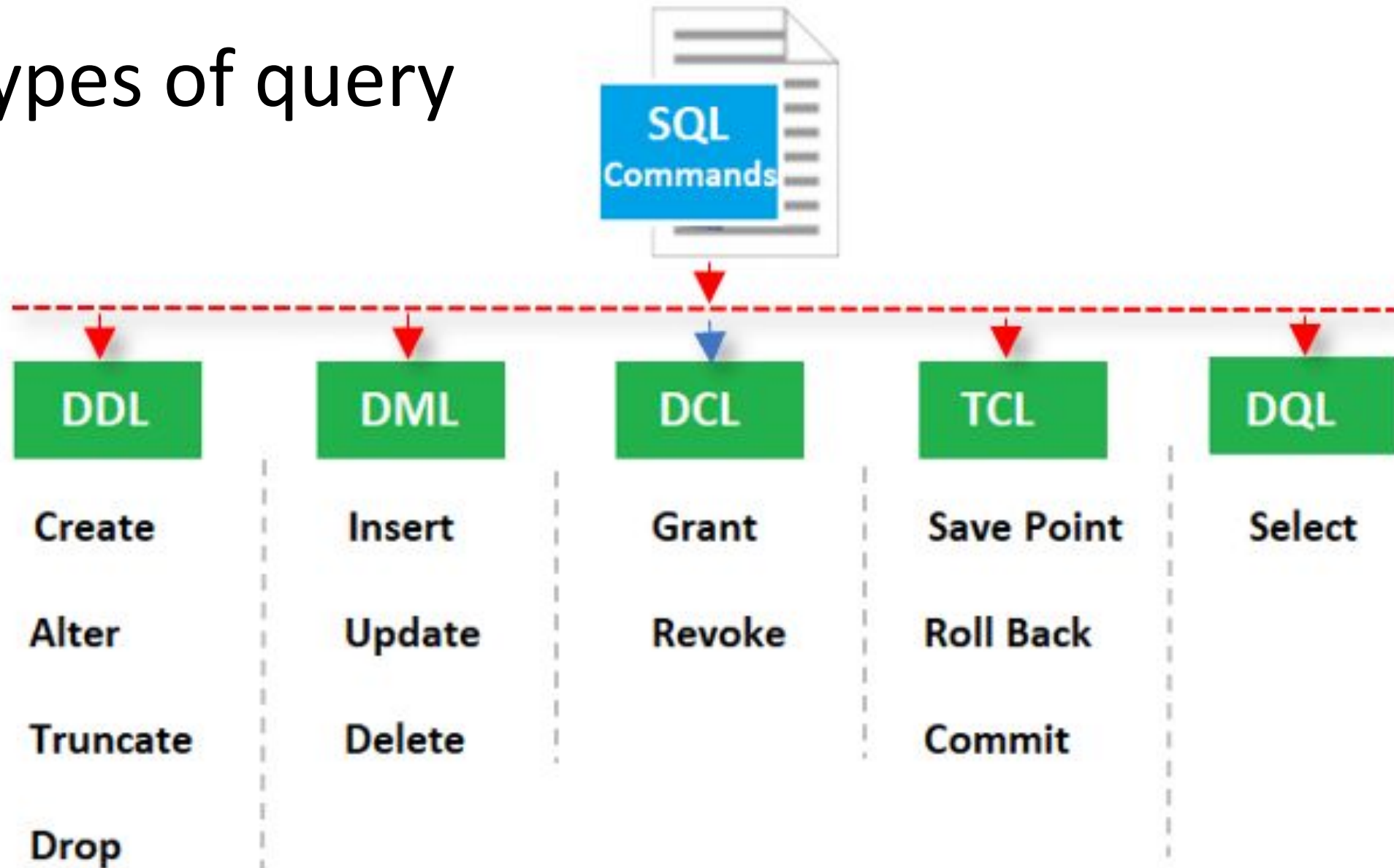
FLOAT(*size*, *d*) A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions

FLOAT(*p*) A floating point number. MySQL uses the *p* value to determine whether to use FLOAT or DOUBLE for the resulting data type. If *p* is from 0 to 24, the data type becomes FLOAT(). If *p* is from 25 to 53, the data type becomes DOUBLE()

DOUBLE(<i>size</i> , <i>d</i>)	A normal-size floating point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter
DECIMAL(<i>size</i> , <i>d</i>)	An exact fixed-point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter. The maximum number for <i>size</i> is 65. The maximum number for <i>d</i> is 30. The default value for <i>size</i> is 10. The default value for <i>d</i> is 0.
DEC(<i>size</i> , <i>d</i>)	Equal to DECIMAL(<i>size</i> , <i>d</i>)

DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(<i>fsp</i>)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(<i>fsp</i>)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(<i>fsp</i>)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

Types of query



Database related queries

To create database:	CREATE DATABASE db_name;
To view all databases:	SHOW DATABASES;
To use specific database:	USE db_name;
To delete database	DROP DATABASE database_name;

Creating table

```
CREATE TABLE <table_name> (  
  
    <column1_name> <data_type> <constraint_if_any>,  
    <column2_name> <data_type> <constraint_if_any>,  
    <column3_name> <data_type> <constraint_if_any>,  
    .  
    .  
);
```

Employee

Employee_Id	FirstName	LastName	City
101	Mahesh	Pandey	Mumbai
102	Nikita	Sharma	Pune
103	Manisha	More	Mumbai


```
CREATE TABLE employee (  
  
    employee_id int      ,  
    firstName varchar(20),  
    LastName  varchar(20),  
    City      varchar(20)  
);
```

To view structure of table: DESC <table_name>;

```
mysql> DESC employee;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| employee_id    | int           | YES  |     | NULL    |       |
| FirstName      | varchar(10)   | YES  |     | NULL    |       |
| LastName       | varchar(10)   | YES  |     | NULL    |       |
| city           | varchar(8)    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

To delete table: DROP TABLE <table_name>;

Create table Student


Student_id	Name	Course	dob
101	Mahesh	Full stack web developement	Mumbai
102	Nikita	Data Science	Pune
103	Manisha	Data Analytics	Mumbai

```
CREATE TABLE student (  
    student_id INT PRIMARY KEY,  
    first_name VARCHAR(255),  
    last_name VARCHAR(255),  
    dob DATE  
);
```

ALTER



Alter query is used to modify the table schema.



We can use an alter query to add a new column, drop the existing column, modifying or renaming the existing column.



Further it can also be used to add or remove the constraints on table fields

Adding new column to existing table

```
ALTER TABLE <table_name> ADD COLUMN  
<column_name> <data_type>;
```

- If you want to add a column after any particular column:

```
ALTER TABLE <table_name> ADD COLUMN  
<new_column_name> <data_type>  
AFTER <existing_column_name>;
```

- To add a column at first :

```
ALTER TABLE <table_name> ADD  
COLUMN <column_name> <data_type>  
FIRST;
```

- Dropping Existing Column

```
ALTER TABLE <table_name> DROP COLUMN <column_name>;
```

- Changing datatype of existing column

```
ALTER TABLE student MODIFY city CHAR(5);
```

- Changing column name along with the datatype//for xampp users

```
alter table <table-name> change <old-col-name> <new-col-name>  
datatype;
```

Renaming column

```
alter table <table-name> rename column <old_col_name> TO  
<new_col_name>;
```

Renaming table

```
ALTER TABLE <old_table_name> RENAME TO <new_table_name>;
```

OR

```
RENAME TABLE <old_table_name> TO <new_table_name>;
```

DML

Data Manipulation Language

DML



INSERT



UPDATE



DELETE



INSERT

DELETE

UPDATE

It is used to insert data into table

It is used to delete records from a database table.

It is used to update existing data within a table.

INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

Employee_Id	FirstName	LastName	City
101	Mahesh	Pandey	Mumbai
102	Nikita	Sharma	Pune
103	Manisha	More	Mumbai

- You can use insert into statement in two ways

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

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

Inserting into multiple rows

```
INSERT INTO table_name (c1,c2,c3)
VALUES
(v1,v2,v3),
(v1,v2,v3),
(v1,v2,v3),
(v1,v2,v3),
;
```

UPDATE statement

- Update statement is used to modify the existing records in a table

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

Warning

- Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

DELETE

```
DELETE FROM table_name WHERE condition;
```

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

If you forget to put where clause

```
DELETE FROM table_name;
```

It will delete all records



To delete the table completely, use the **DROP TABLE** statement:

```
DROP TABLE Customers;
```



TRUNCATE

- Truncate deletes all records from a table

```
TRUNCATE TABLE <table_name>;
```

Constraints

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

```
CREATE TABLE table_name (  
    column1 datatype constraint,  
    column2 datatype constraint,  
    column3 datatype constraint,  
    ....  
);
```

- The following constraints are commonly used in SQL:
- 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 satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified

NOT NULL

By default, a column can hold NULL values.

This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

```
ALTER TABLE <table_name> MODIFY <col_name> <datatype> NOT NULL;
```

In order to drop the not null constraint, set the column definition again using the modify

UNIQUE

```
CREATE TABLE your_table_name (  
    column1 datatype,  
    column2 datatype,  
  
    UNIQUE (column1)  
);
```

```
ALTER TABLE <table_name> ADD UNIQUE(col_name);  
ALTER TABLE <table_name> DROP INDEX(col_name);
```


Primary key

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.

```
ALTER TABLE <table-name> add primary key <col_name>
```

```
ALTER TABLE <table_name> DROP PRIMARY KEY;
```

DEFAULT

- SQL DEFAULT Constraint
- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.

```
ALTER TABLE <table_name>  
ALTER <col_name> SET DEFAULT (value);
```

- Drop a default constraint

```
ALTER TABLE <table_name>  
ALTER <col_name> DROP DEFAULT;
```

Check constraint

Create table demo

```
(  
    age int check(age>18)  
);
```

Or

```
Create table demo(  
Age int,  
Check(Age >18)  
)
```

Using alter query

- `ALTER TABLE <table-name> ADD CHECK (condition);`

```
ALTER TABLE <table-name> ADD CONSTRAINT  
<constraint_name>CHECK (condition);
```

- Drop a constraint

```
alter table <table_name> drop constraint <constraint_name>;
```

Foreign key

- CREATE TABLE Orders (
 OrderID int NOT NULL,
 OrderNumber int NOT NULL,
 PersonID int,
 PRIMARY KEY (OrderID),
 CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
 REFERENCES Persons(PersonID)
);

Alter table <table-name> add constraint <constraint_name> foreign key
(col_name) references
<other_table_name>(col_name_from that table)

```
ALTER TABLE <table_name>  
DROP FOREIGN KEY <name_of_constraint>;
```

E_id	Ename	City	salary	Department
1111	Nikita	Mumbai	67000	D1
1112	Prajakta	Pune	80000	D1
1113	Manisha	Bangalore	20000	D2
1114	Nilesh	Mumbai	35469	D3
1115	Monal	Pune	34452	D2

D_id	Department
D1	Marketing
D2	IT
D3	Human Resource

Apply foreign key on employee table (column:Department)

Studentdetails(Do no add Constraints)

s_id	S-name	Marks	presentDays	School_name
1111	Nikita	78	200	S1
1112	Prajakta	56	197	S2
1113	Manisha	90	160	S3
1114	Nilesh	100	140	S1
1115	Monal	20	200	s2

SELECT

- Select statement is used to fetch data from a database
- `SELECT * FROM <table-name>;`
- It will return all records

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

```
SELECT DISTINCT column1, column2, ...  
FROM table_name;
```

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

WHERE

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. Note: In some versions of SQL this operator may be written as !=

<>	Not equal. Note: In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

AND

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition1 AND condition2 AND condition3 ...;
```

OR

- SELECT *column1, column2, ...*
FROM *table_name*
WHERE *condition1 AND condition2 AND condition3 ...*;

LIKE

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

```
SELECT column1, column2, ...  
FROM table_name  
WHERE columnN LIKE pattern;
```


LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

ORDER BY

- ORDER by is used to sort records either in Ascending or descending
- ASC-Ascending
- DESC –for descending
- By default it sorts by ascending order

```
SELECT <col_name>  
FROM table_name  
ORDER BY <col_name> ASC|DESC;
```

IS NULL

```
SELECT column_names  
FROM table_name  
WHERE column_name IS NULL;
```

IS NOT NULL

```
SELECT column_names  
FROM table_name  
WHERE column_name IS NOT NULL;
```

Case

```
SELECT OrderID, Quantity,  
CASE  
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'  
    WHEN Quantity = 30 THEN 'The quantity is 30'  
    ELSE 'The quantity is under 30'  
END AS QuantityText  
FROM OrderDetails;
```

Functions In sql

String functions

Substr

Length

Cocat

Lower

upper

Numeric function

- `Mod(m,n)`-Returns remainder
- `Round(n)`-
- `Ceil(n)`
- `Floor(n)`
- `Pow(m,n)`
- `Sign(n)`- returns -1 for negative and 1 for positive
- `Sqrt(n)`
- `Truncate(m,n)`
- `Abs(n)`

Date Function

- Currdate()
- Curtime
- Now()
- Date("")
- Month("")
- Year("")
- Date_format(date, "format")
- Datediff(date1,date2)
- Dateadd(currdate,interval 10 days)

%a	Abbreviated weekday name (Sun to Sat)
%b	Abbreviated month name (Jan to Dec)
%c	Numeric month name (0 to 12)
%D	Day of the month as a numeric value, followed by suffix (1st, 2nd, 3rd, ...)
%d	Day of the month as a numeric value (01 to 31)
%e	Day of the month as a numeric value (0 to 31)
%f	Microseconds (000000 to 999999)
%H	Hour (00 to 23)
%h	Hour (00 to 12)
%I	Hour (00 to 12)
%i	Minutes (00 to 59)
%j	Day of the year (001 to 366)

Aggregate function

- Avg
 - Min
 - Max
 - Count
 - Sum
-
- `Select count(distinct(institute)) from itvedant`

- **COALESCE(value,...):** Returns the first non-NULL value in the list, or NULL if there are no non-NULL values.

```
MariaDB [myTestDB]> SELECT COALESCE(NULL,1);
+-----+
| COALESCE(NULL,1) |
+-----+
| 1 |
+-----+
1 row in set (0.05 sec)

MariaDB [myTestDB]> SELECT COALESCE(NULL,NULL,NULL);
+-----+
| COALESCE(NULL,NULL,NULL) |
+-----+
| NULL |
+-----+
1 row in set (0.00 sec)
```

- **ISNULL(expr):** If expr is NULL, ISNULL() returns 1, otherwise it returns 0.

```
MariaDB [myTestDB]> SELECT ISNULL(1+1);
+-----+
| ISNULL(1+1) |
+-----+
| 0 |
+-----+
1 row in set (0.09 sec)

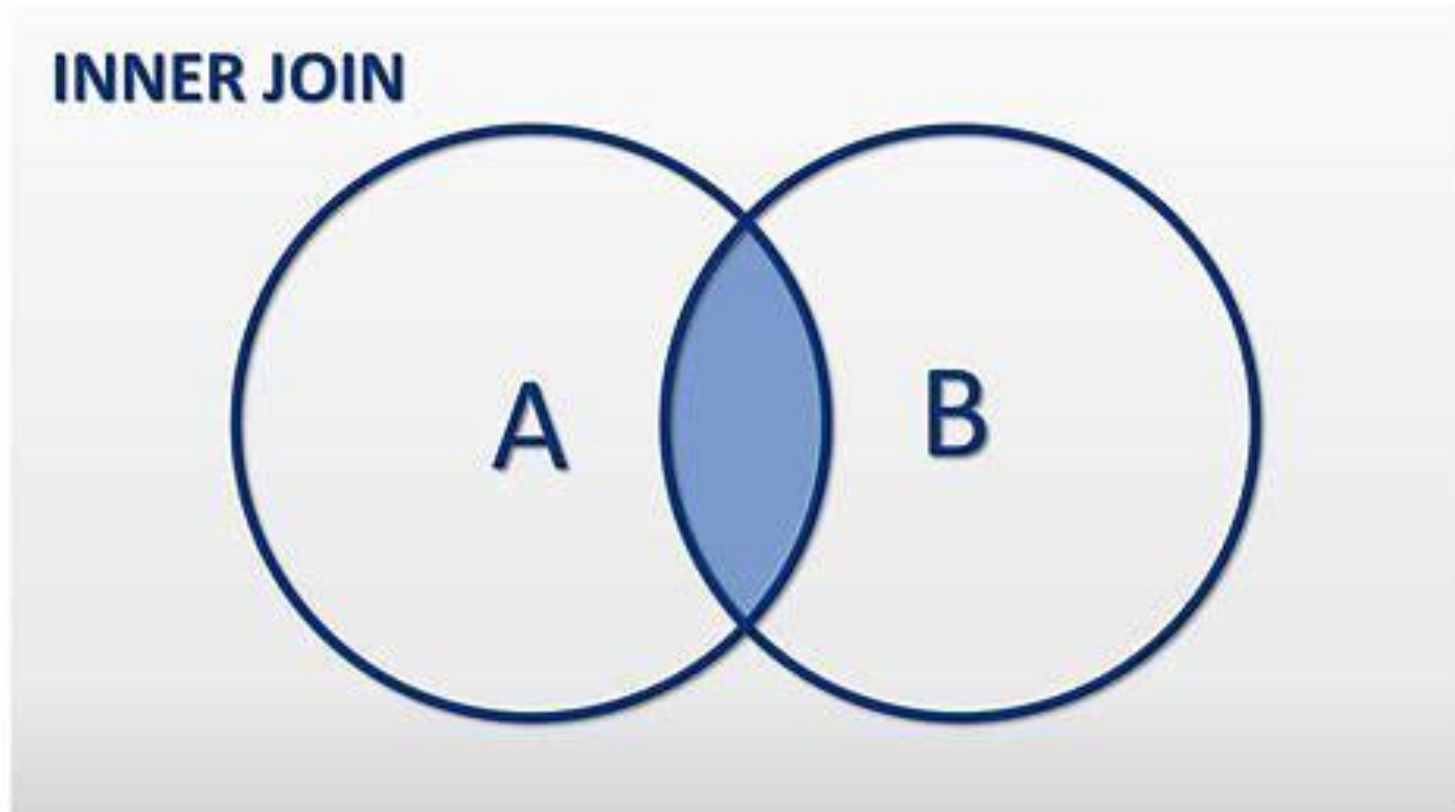
MariaDB [myTestDB]> SELECT ISNULL(1/0);
+-----+
| ISNULL(1/0) |
+-----+
| 1 |
+-----+
1 row in set (0.04 sec)
```

Join

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

Inner Join

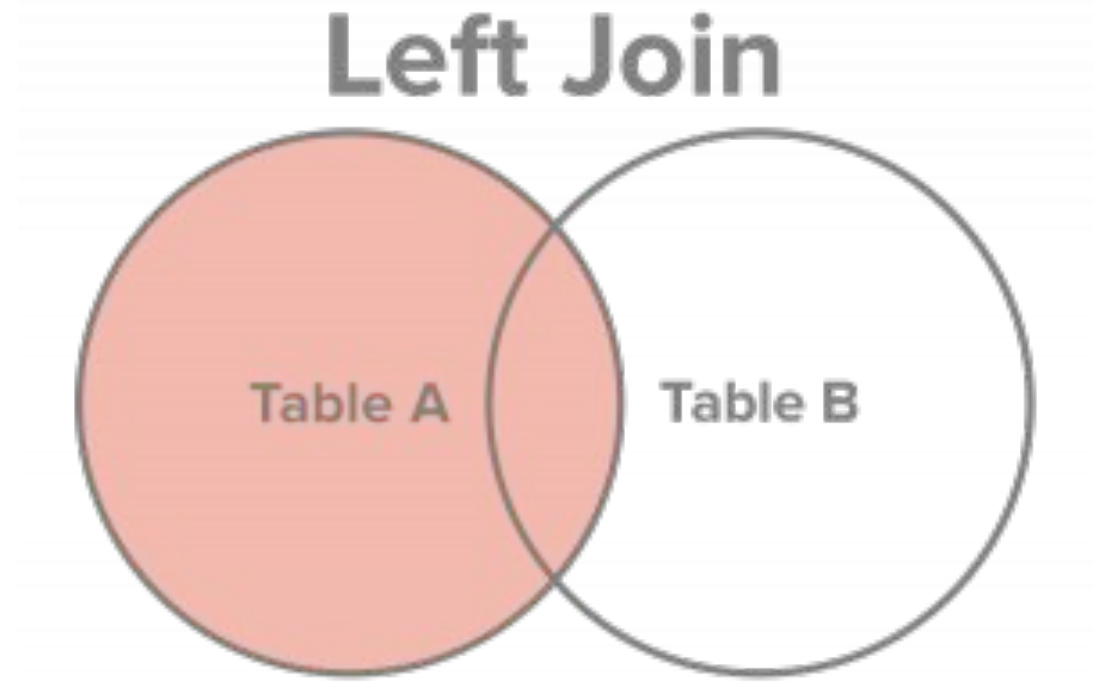
The INNER JOIN keyword selects records that have matching values in both tables.



```
SELECT column_name(s)  
FROM table1  
INNER JOIN table2  
ON table1.column_name = table2.column_name;
```

LEFT JOIN

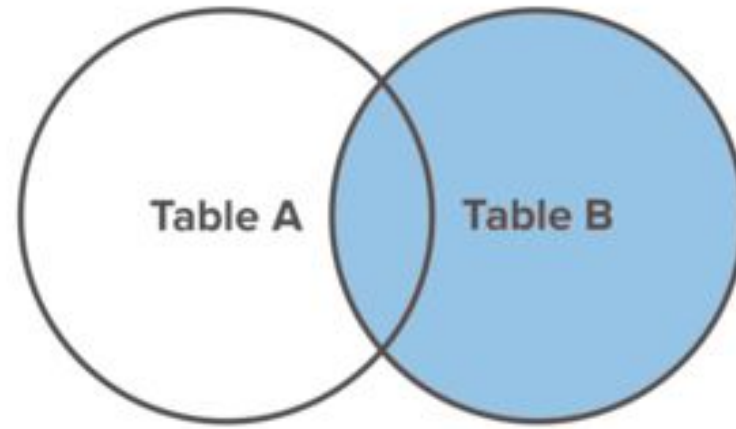
- The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.



```
SELECT column_name(s)  
FROM table1  
LEFT JOIN table2  
ON table1.column_name = table2.column_name;
```


Right Join

- The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.



SQL RIGHT JOIN

```
SELECT column_name(s)  
FROM table1  
RIGHT JOIN table2  
ON table1.column_name = table2.column_name;
```

- Se* from table1 cross join table 2