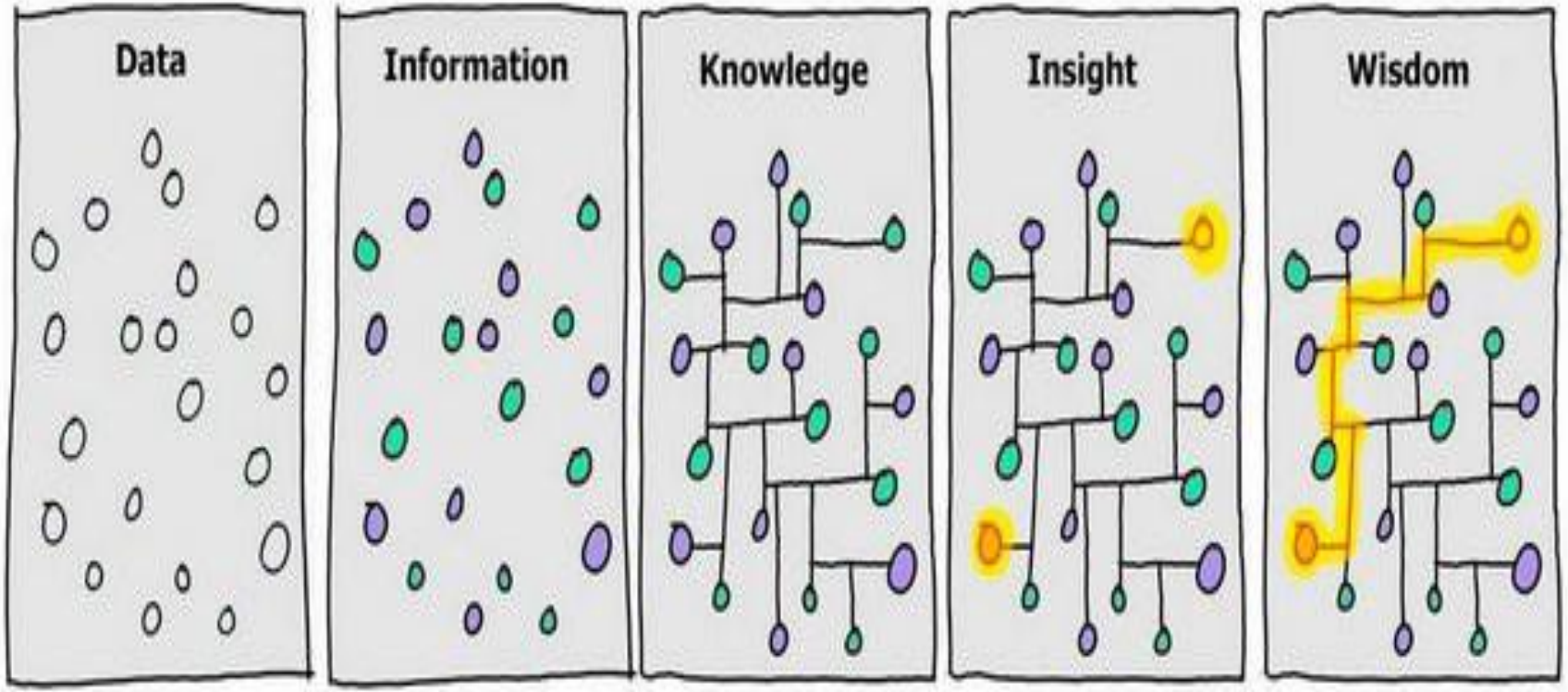
The DIKW Model - Unleashed

DIKW Model



Structure of Data, Collection

Data vs Information

Computer Data

- Data is a collection of values. Those values can be characters, numbers, or any other data type.
- Computer data is a bunch of 1's and 0's, known as binary data.
- Computer data is processed by the computer's CPU and is stored digitally in files and folders on the computer's hard disk.

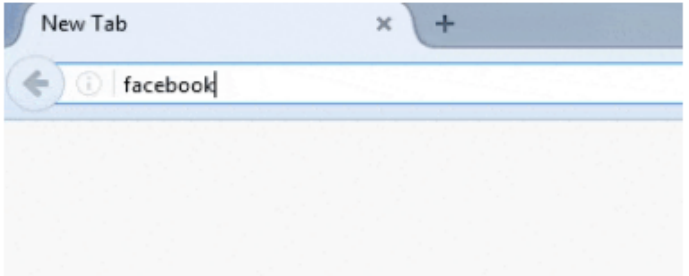
Information

- When data are processed, interpreted, organized, structured or presented in a given context so as to make them meaningful or useful, they are called information.
- Data is raw material (i.e. bits of information) and Information is the product.

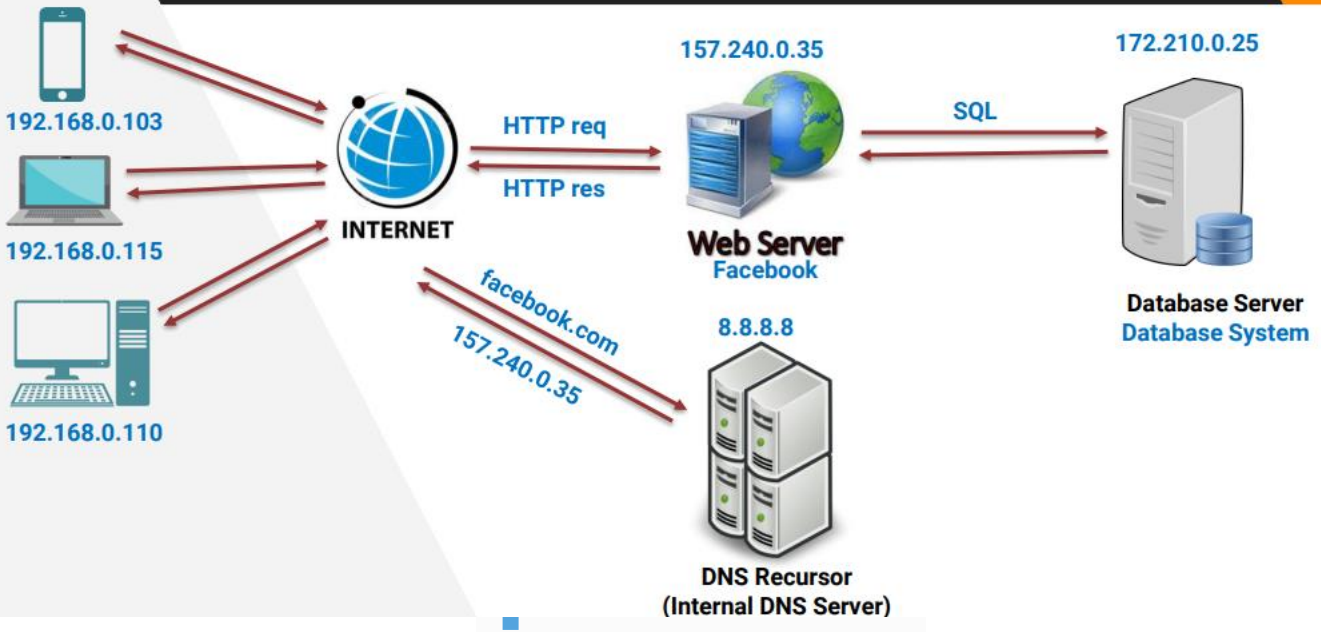
Web-based Information System

Web-based Information System

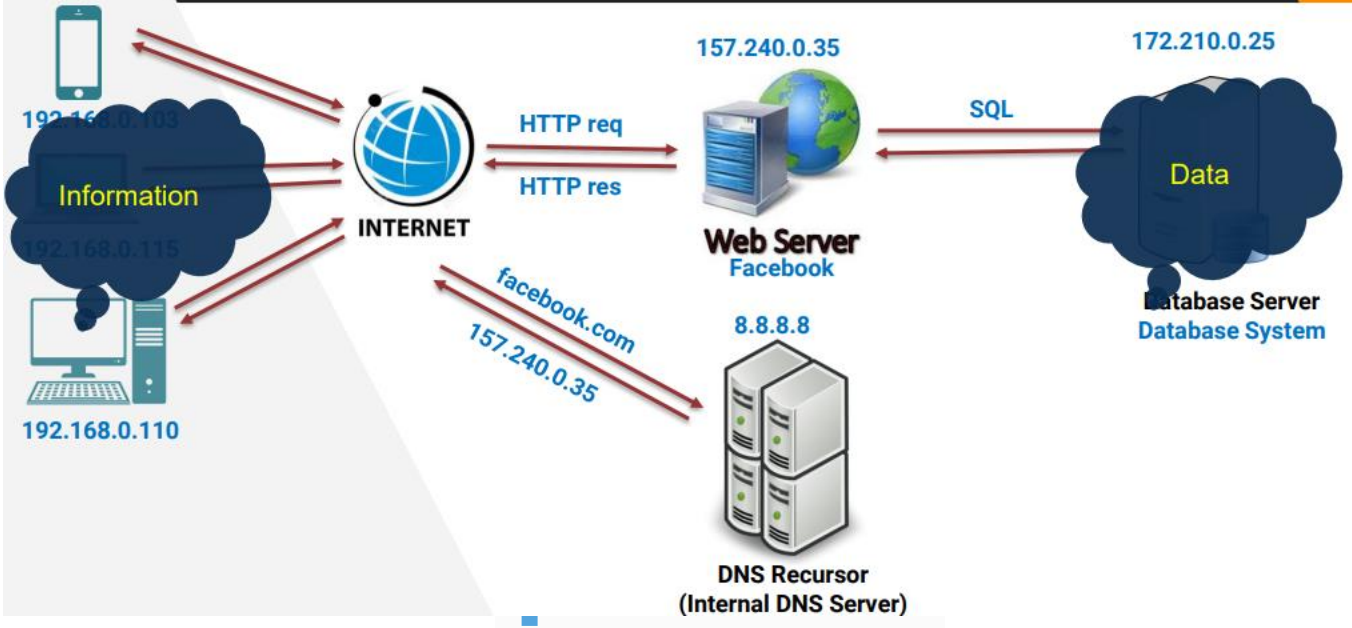
An information system that uses *Internet web technologies* to deliver information and services, to users or other information systems/applications.



Web-based System >> Background Process



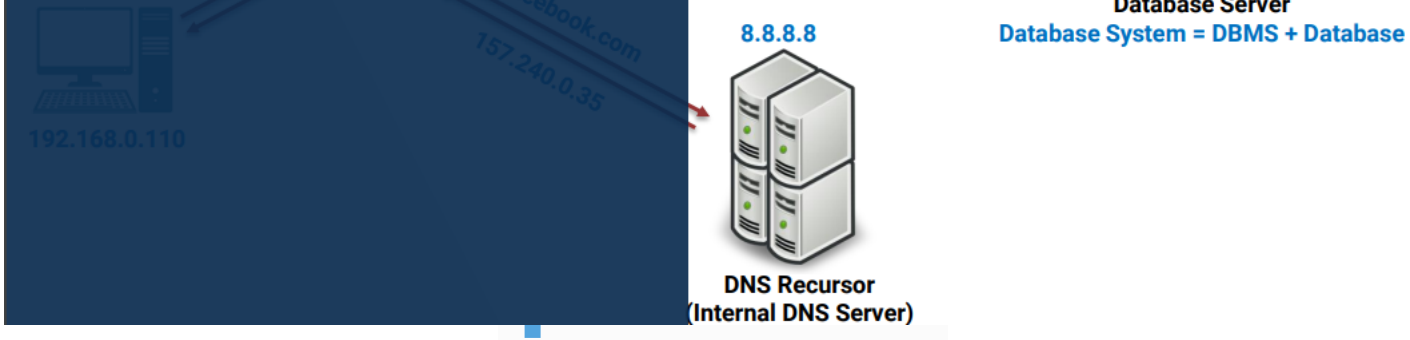
Web-based System >> Background Process



Database vs Database Management System

Database
A database is an organized collection of interrelated data, generally stored and accessed electronically from a computer system.

Database Management System
A DBMS is software that interacts with end users, applications, and the database itself to control the storage, organization, and retrieval of data.



Collection of Data

- The data which are originally collected for the first time for the purpose of the survey are called primary data. For example facts or data collected regarding the habit of taking tea or coffee in a village by an investigator.

- Method of Collecting Primary Data
- There are several methods for collecting primary data. Some of them are:
- □ Direct personal investigation
- □ Indirect investigations
- □ Through correspondent
- □ By mailed questionnaire
- □ Through schedules

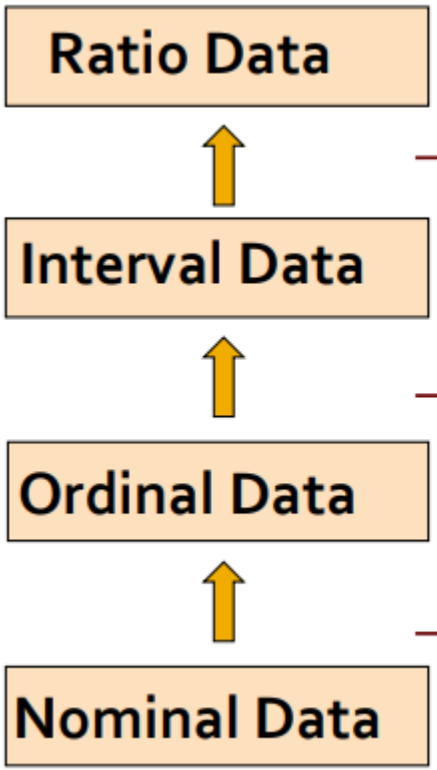
Collection of Data (Contd..)

- Secondary Data
- When we use the data, which have already been collected by others, the data are called secondary data. This data is said to be primary for the agency which collects it first, and it becomes secondary for all the other users.
- Method of Collecting Secondary Data
- Published reports of newspapers, RBI and periodicals.
- Publication from trade associations
- Financial data reported in annual reports
- Information from official publications
- Publication of international bodies such as UNO, World Bank etc.
- Internal reports of the government departments
- Records maintained by the institutions
- Research reports prepared by students in the universities

Types of Data

- Categorical Data
- Categorical data is the statistical data type consisting of categorical variables or of data that has been converted into that form, for example as grouped data. For example- Marital Status, Political Party, Eye Color, etc.
- Numerical Data
- Numerical values or observations can be measured. And these numbers can be placed in ascending or descending order. Numerical data can be divided into two groups:
- Discrete(Counted Items such as- number of children, defects per hour etc.)
- Continuous(Measured Characteristics such as- weight, voltage etc.)

Types of Data Level of Measurement



		EXAMPLES:
Ratio Data	Differences between measurements, true zero exists	Height, Age, Weekly Food Spending
Interval Data	Differences between measurements but no true zero	Temperature in Fahrenheit, Standardized exam score
Ordinal Data	Ordered Categories (rankings, order, or scaling)	Service quality rating, Standard & Poor's bond rating, Student letter grades
Nominal Data	Categories (no ordering or direction)	Marital status, Type of car owned

SQL

SQL is the standard language for Relational Database System.

All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

SQL is a database computer language designed for the retrieval and management of data in a relational database. SQL stands for Structured Query Language.

Name Place

SQL Process

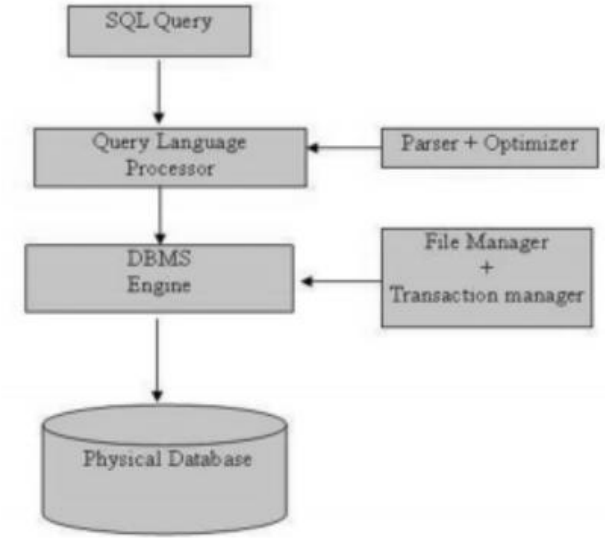
When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task. There are various components included in this process.

These components are –

- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.

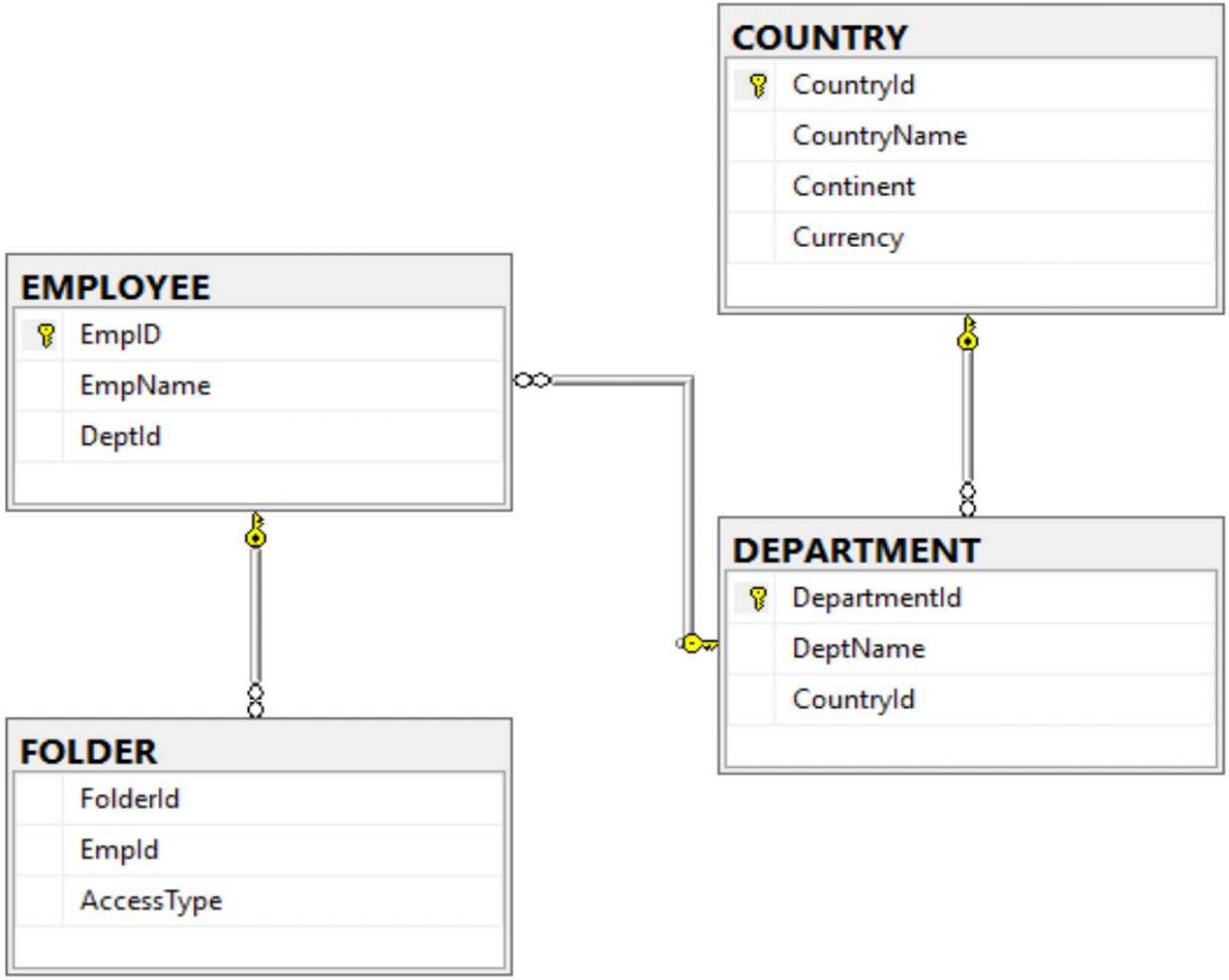
A classic query engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.

Following is a simple diagram showing the SQL Architecture:



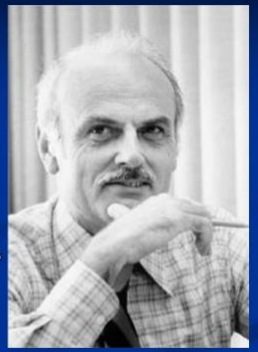
Why SQL? SQL is widely popular because

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.


COUNTRY
CountryId
CountryName
Continent
Currency
EMPLOYEE
EmpID
EmpName
DeptId
DEPARTMENT
DepartmentId
DeptName
CountryId
FOLDER
FolderId
EmpId
AccessType

Dr. Edgar F. Codd (1923-2003)

- Codd completed his PhD at the University of Michigan in 1963, and presented a thesis on the topic of a self-reproducing computer consisting of a large number of simple identical cells, each of which interacts in a uniform manner with its four immediate neighbors. Codd reported this work in a book entitled *Cellular Automata* published by Academic Press in 1968.



E.F. Codd Rules

1	Foundation of the Relational Model
2	Guaranteed Access to Data
3	Systematic Treatment of Null Values
4	Active Data Domain
5	Relational Completeness
6	View Updating
7	Physical Data Independence
8	Logical Data Independence
9	Integrity Independence
10	Distribution Independence
11	Non-Subversion of Relational Operators
12	Data Dictionary

Relational Databases

Tables

Tables are the building blocks of relational databases. They organize data into rows and columns.

Rows

Rows represent individual or entities in a table. Each row contains information about a specific instance.

Columns

Columns define the attributes or or characteristics of each record in record in a table. They represent represent the different data points points for each entity.

Types of SQL Commands

1 Data Definition Language (DDL)

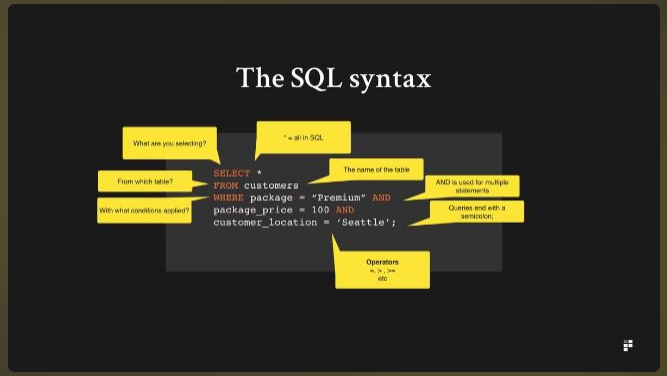
DDL commands are used to define and manage the structure of the including creating, altering, and dropping tables.

2 Data Manipulation Language (DML)

DML commands are used for manipulating data within the database, including including inserting, updating, and deleting records.

3 Data Control Language (DCL)

DCL commands are used for managing database access and including granting and revoking user privileges.

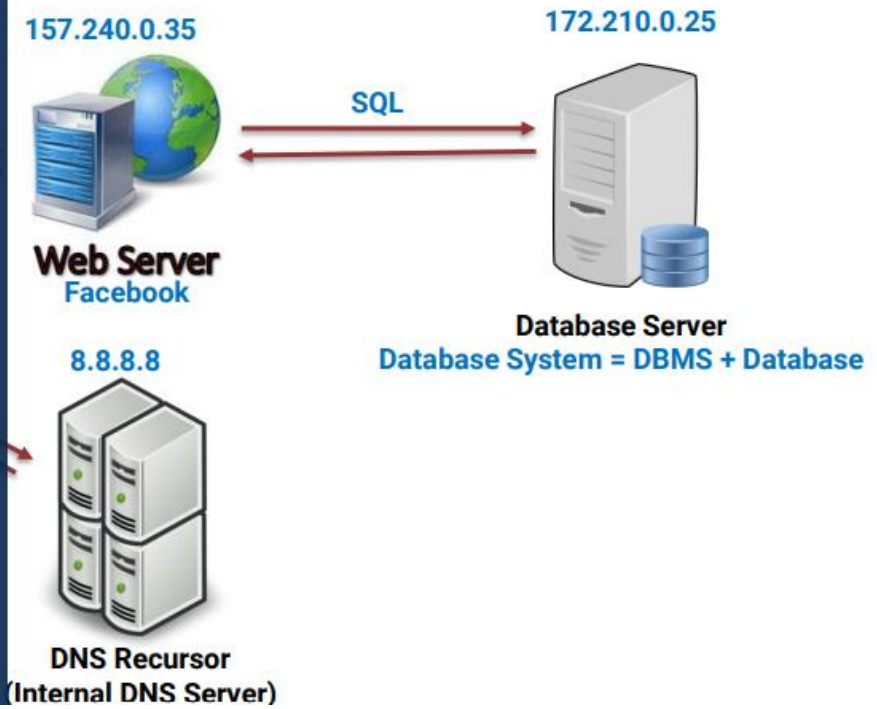


SQL – Structured Query Language

Structured Query Language (SQL)

- A special kind of programming language that is used for communicating with the database.
- It is designed for managing data held in a relational database management system (RDBMS).

- Types of SQL commands:
  - DDL – Data Definition Language
    CREATE, DROP, ALTER, TRUNCATE
  - DML – Data Manipulation Language
    INSERT, UPDATE, DELETE
  - DQL – Data Query Language
    SELECT
  - DCL – Data Control Language
    GRANT, REVOKE
  - TCL – Transaction Control Language
    COMMIT, ROLLBACK, SAVEPOINT



SQL Commands

The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into the following groups based on their nature:

DDL - Data Definition Language

Command	Description
CREATE	Creates a new table, a view of a table, or other object in the database.
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other objects in the database.

DML - Data Manipulation Language

Command	Description
SELECT	Retrieves certain records from one or more tables.
INSERT	Creates a record.
UPDATE	Modifies records.
DELETE	Deletes records.

DCL - Data Control Language

Command	Description
GRANT	Gives a privilege to user.
REVOKE	Takes back privileges granted from user.

Exact Numeric Data Types

DATA TYPE	FROM	TO
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	-10^38 +1	10^38 -1
numeric	-10^38 +1	10^38 -1
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

Date and Time Data Types

DATA TYPE	FROM	TO
datetime	Jan 1, 1753	Dec 31, 9999
smalldatetime	Jan 1, 1900	Jun 6, 2079
date	Stores a date like June 30, 1991	
time	Stores a time of day like 12:30 P.M.	

Character Strings Data Types

DATA TYPE	Description
char	Maximum length of 8,000 characters.(Fixed length non-Unicode characters)
varchar	Maximum of 8,000 characters.(Variable-length non-Unicode data).
varchar(max)	Maximum length of 231characters, Variable-length non-Unicode data (SQL Server 2005 only).
text	Variable-length non-Unicode data with a maximum length of 2,147,483,647 characters.

What is RDBMS?

RDBMS stands for **R**elational **D**atabase **M**anagement **S**ystem. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.

What is a table?

The data in an RDBMS is stored in database objects which are called as **tables**. This table is basically a collection of related data entries and it consists of numerous columns and rows.

Remember, a table is the most common and simplest form of data storage in a relational database. The following program is an example of a CUSTOMERS table:

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
```

What is a field? Every table is broken up into smaller entities called fields. The fields in the CUSTOMERS table consist of ID, NAME, AGE, ADDRESS and SALARY.
A field is a column in a table that is designed to maintain specific information about every record in the table.

What is a Record or a Row? A record is also called as a row of data is each individual entry that exists in a table. For example, there are 7 records in the above CUSTOMERS table. Following is a single row of data or record in the CUSTOMERS table:

```
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
+----+----------+-----+-----------+----------+
```

A record is a horizontal entity in a table.

What is a column?

A column is a vertical entity in a table that contains all information associated with a specific field in a table.

For example, a column in the CUSTOMERS table is ADDRESS, which represents location description and would be as shown below:

```
+-----------+
| ADDRESS   |
+-----------+
| Ahmedabad |
| Delhi     |
| Kota      |
| Mumbai    |
| Bhopal    |
| MP        |
| Indore    |
+----+------+
```

What is a NULL value?

A NULL value in a table is a value in a field that appears to be blank, which means 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. A field with a NULL value is the one that has been left blank during a record creation.

SQL Constraints

Constraints are the rules enforced on data columns on 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 can either be column level or table level. Column level constraints are applied only to one column whereas, table level constraints are applied to the entire table.

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

- NOT NULL Constraint: Ensures that a column cannot have a NULL value.
- DEFAULT Constraint: Provides a default value for a column when none is specified.
- UNIQUE Constraint: Ensures that all the 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 another database table.
- CHECK Constraint: The CHECK constraint ensures that all values in a column satisfy certain conditions.
- INDEX: Used to create and retrieve data from the database very quickly.

Data Integrity

The following categories of data integrity exist with each RDBMS:

- **Entity Integrity:** There are no duplicate rows in a table.
- **Domain Integrity:** Enforces valid entries for a given column by restricting the type, the format, or the range of values.
- **Referential integrity:** Rows cannot be deleted, which are used by other records.
- **User-Defined Integrity:** Enforces some specific business rules that do not fall into entity, domain or referential integrity.

```
CREATE TABLE CUSTOMERS(
       CUST_ID       INT              NOT NULL,
       CUST_NAME     VARCHAR (20)      NOT NULL,
       DOB           DATE,
       STREET        VARCHAR(200),
       CITY          VARCHAR(100),
       STATE         VARCHAR(100),
       ZIP           VARCHAR(12),
       EMAIL_ID      VARCHAR(256),
       PRIMARY KEY (CUST_ID)
```

Various Syntax in SQL

All the examples given in this tutorial have been tested with a MySQL server.

SQL SELECT Statement

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

SQL DISTINCT Clause

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

SQL WHERE Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION;
```

SQL AND/OR Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION-1 {AND|OR} CONDITION-2;
```

SQL IN Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name IN (val-1, val-2,...val-N);
```

SQL BETWEEN Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name BETWEEN val-1 AND val-2;
```

SQL LIKE Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name LIKE { PATTERN };
```

SQL ORDER BY Clause

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION
ORDER BY column_name {ASC|DESC};
```

SQL GROUP BY Clause

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name;
```

SQL COUNT Clause

```
SELECT COUNT(column_name)
FROM   table_name
WHERE  CONDITION;
```

SQL HAVING Clause

```
SELECT SUM(column_name)
FROM   table_name
WHERE  CONDITION
GROUP BY column_name
HAVING (arithematic function condition);
```

SQL CREATE TABLE Statement

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

SQL DROP TABLE Statement

```
DROP TABLE table_name;
```

SQL CREATE INDEX Statement

```
CREATE UNIQUE INDEX index_name
ON table_name ( column1, column2,...columnN);
```

SQL DROP INDEX Statement

```
ALTER TABLE table_name
DROP INDEX index_name;
```

SQL DESC Statement

```
DESC table_name;
```

SQL TRUNCATE TABLE Statement

```
TRUNCATE TABLE table_name;
```

SQL ALTER TABLE Statement

```
ALTER TABLE table_name {ADD|DROP|MODIFY} column_name {data_ype};
```

SQL ALTER TABLE Statement (Rename)

```
ALTER TABLE table_name RENAME TO new_table_name;
```

SQL INSERT INTO Statement

```
INSERT INTO table_name( column1, column2....columnN)
VALUES ( value1, value2....valueN);
```

SQL UPDATE Statement

```
UPDATE table_name
SET column1 = value1, column2 = value2....columnN=valueN
[ WHERE  CONDITION ];
```

SQL DELETE Statement

```
DELETE FROM table_name
WHERE  {CONDITION};
```

SQL CREATE DATABASE Statement

```
CREATE DATABASE database_name;
```

SQL DROP DATABASE Statement

```
DROP DATABASE database_name;
```

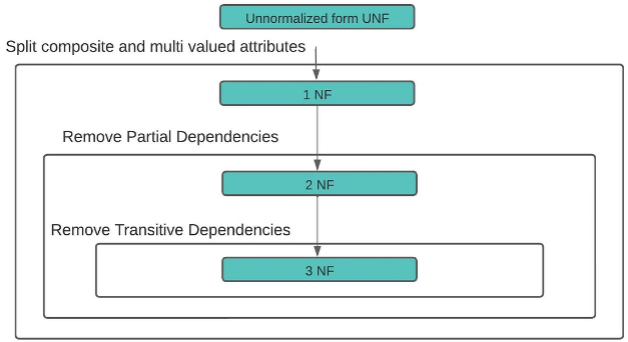
SQL USE Statement

```
USE database_name;
```

SQL COMMIT Statement

```
COMMIT;
```

SQL ROLLBACK Statement



Normalization

1

1NF

Eliminate repeating groups of data by separating them into distinct tables.

2

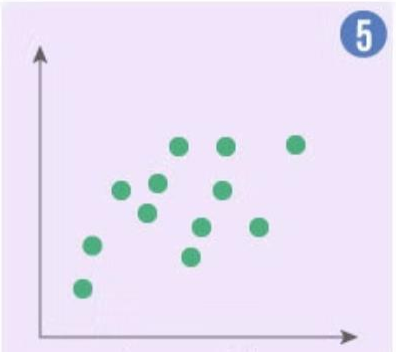
2NF

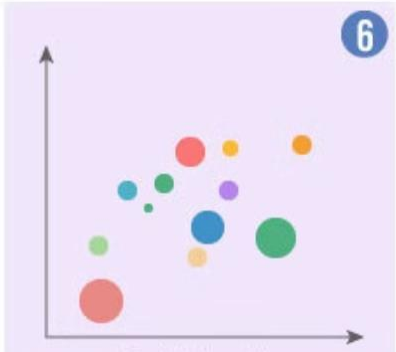
Remove redundancy by ensuring that all non-key attributes depend on primary key.

3

3NF

Ensure that non-key attributes depend only on the primary key and not on other on other non-key attributes.





Aggregate Functions (continued)

SUM

Calculates the total values in a column. for finding total sales, expenses, or inventory.

COUNT

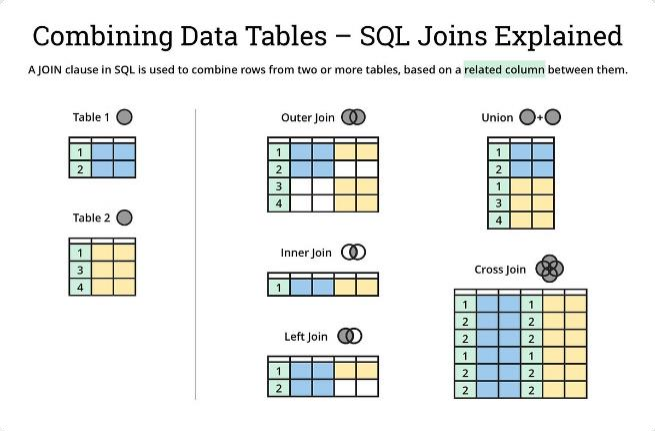
Determines the number of of rows that meet a specific condition. Useful Useful for counting customers, orders, or products.

AVG

Calculates the average value of a column. for finding the average price, age, or salary.

MIN/MAX

Finds the smallest or largest value in a column. column. Useful for identifying the lowest price, oldest customer, or or highest salary.



Joining Tables

1

INNER JOIN

Returns rows that have matching values in both tables.

2

LEFT JOIN

Returns all rows from the left table and matching rows from the right table. right table.

3

RIGHT JOIN

Returns all rows from the right table and matching rows from the left table. left table.

4

FULL JOIN

Returns all rows from both tables, regardless of whether they have matching matching values.

JOINING TABLES

JOIN combines data from two tables.

TOY

toy_id	toy_name	cat_id
1	ball	3
2	spring	NULL
3	mouse	1
4	mouse	4
5	ball	1

CAT

cat_id	cat_name
1	Kitty
2	Hugo
3	Sam
4	Misty

JOIN typically combines rows with equal values for the specified columns. **Usually**, one table contains a **primary key**, which is a column or columns that uniquely identify rows in the table (the `cat_id` column in the `cat` table).
The other table has a column or columns that **refer to the primary key columns** in the first table (the `cat_id` column in the `toy` table). Such columns are **foreign keys**. The JOIN condition is the equality between the primary key columns in one table and columns referring to them in the other table.

JOIN

JOIN returns all rows that match the ON condition. JOIN is also called INNER JOIN.

```
SELECT *
FROM toy
JOIN cat
  ON toy.cat_id = cat.cat_id;
```

toy_id	toy_name	cat_id	cat_id	cat_name
5	ball	1	1	Kitty
3	mouse	1	1	Kitty
1	ball	3	3	Sam
4	mouse	4	4	Misty

There is also another, older syntax, but it **isn't recommended**.
List joined tables in the FROM clause, and place the conditions in the WHERE clause.

```
SELECT *
FROM toy, cat
WHERE toy.cat_id = cat.cat_id;
```

LEFT JOIN

`LEFT JOIN` returns all rows from the **left table** with matching rows from the right table. Rows without a match are filled with `NULL`s. `LEFT JOIN` is also called `LEFT OUTER JOIN`.

```
SELECT *
FROM toy
LEFT JOIN cat
  ON toy.cat_id = cat.cat_id;
```

toy_id	toy_name	cat_id	cat_id	cat_name
5	ball	1	1	Kitty
3	mouse	1	1	Kitty
1	ball	3	3	Sam
4	mouse	4	4	Misty
2	spring	NULL	NULL	NULL

whole left table

RIGHT JOIN

`RIGHT JOIN` returns all rows from the **right table** with matching rows from the left table. Rows without a match are filled with `NULL`s. `RIGHT JOIN` is also called `RIGHT OUTER JOIN`.

```
SELECT *
FROM toy
RIGHT JOIN cat
  ON toy.cat_id = cat.cat_id;
```

toy_id	toy_name	cat_id	cat_id	cat_name
5	ball	1	1	Kitty
3	mouse	1	1	Kitty
NULL	NULL	NULL	2	Hugo
1	ball	3	3	Sam
4	mouse	4	4	Misty

whole right table

MULTIPLE JOINS

You can join more than two tables together. First, two tables are joined, then the third table is joined to the result of the previous joining.

TOY AS t

toy_id	toy_name	cat_id
1	ball	3
2	spring	NULL
3	mouse	1
4	mouse	4
5	ball	1

CAT AS c

cat_id	cat_name	mom_id	owner_id
1	Kitty	5	1
2	Hugo	1	2
3	Sam	2	2
4	Misty	1	NULL

OWNER AS o

id	name
1	John Smith
2	Danielle Davis

JOIN & JOIN

```
SELECT
  t.toy_name,
  c.cat_name,
  o.name AS owner_name
FROM toy t
JOIN cat c
  ON t.cat_id = c.cat_id
JOIN owner o
  ON c.owner_id = o.id;
```

toy_name	cat_name	owner_name
ball	Kitty	John Smith
mouse	Kitty	John Smith
ball	Sam	Danielle Davis

JOIN & LEFT JOIN

```
SELECT
  t.toy_name,
  c.cat_name,
  o.name AS owner_name
FROM toy t
JOIN cat c
  ON t.cat_id = c.cat_id
LEFT JOIN owner o
  ON c.owner_id = o.id;
```

toy_name	cat_name	owner_name
ball	Kitty	John Smith
mouse	Kitty	John Smith
ball	Sam	Danielle Davis
mouse	Misty	NULL

LEFT JOIN & LEFT JOIN

```
SELECT
  t.toy_name,
  c.cat_name,
  o.name AS owner_name
FROM toy t
LEFT JOIN cat c
  ON t.cat_id = c.cat_id
LEFT JOIN owner o
  ON c.owner_id = o.id;
```

toy_name	cat_name	owner_name
ball	Kitty	John Smith
mouse	Kitty	John Smith
ball	Sam	Danielle Davis
mouse	Misty	NULL
spring	NULL	NULL

JOIN WITH MULTIPLE CONDITIONS

You can use multiple JOIN conditions using the **ON** keyword once and the **AND** keywords as many times as you need.

CAT AS c

cat_id	cat_name	mom_id	owner_id	age
1	Kitty	5	1	17
2	Hugo	1	2	10
3	Sam	2	2	5
4	Misty	1	NULL	11

OWNER AS o

id	name	age
1	John Smith	18
2	Danielle Davis	10

```
SELECT
  cat_name,
  o.name AS owner_name,
  c.age  AS cat_age,
  o.age  AS owner_age
FROM cat c
JOIN owner o
  ON c.owner_id = o.id
 AND c.age < o.age;
```

cat_name	owner_name	age	age
Kitty	John Smith	17	18
Sam	Danielle Davis	5	10