Database management System (DBMS)

DBMS

Database

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently.
It is also used to organize the data in the form of a table, schema, views, and reports, etc.
Example: Employee data base organizes the data about Recruiters, Managers, Accountants etc.

Database Management System

•Database management system is a software which is used to manage the database. For example: MySQL, Oracle, etc are a very popular commercial database which is used in different applications.

•DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.

•It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

Types of Database

Relational DBMS:

- A relational database is one that stores data in tables.
- The relationship between each data point is clear and searching through those relationships is relatively easy.
- The relationship between tables and field types is called a **schema**.
- Relational databases are also called **SQL databases**.
- SQL stands for Structured Query Language and it’s the language relational databases are written in.
- SQL is used to execute queries, retrieve data, and edit data by updating, deleting, or creating new records.

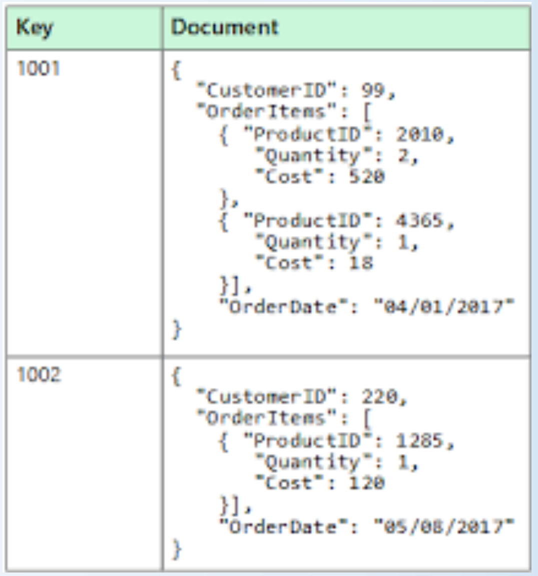
Non Relational DBMS:

- A non-relational database is any database that does not use the tabular schema of rows and columns like in relational databases.
- Rather, its storage model is optimized for the type of data it’s storing.
- Non-relational databases are also known as NoSQL databases which stands for “Not Only SQL.” Where relational databases only use SQL, non-relational databases can use other types of query language.

RDBMS vs Non RDBMS

Customer ID	OrderDate	ProductID	Quantity	Cost
99	04-01-2017	2010	2	520
99	04-01-2017	4365	1	18
220	05/082017	1285	1	120

Relational Database



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" }

Non Relational Database

RDBMS Objects

Tables:
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.

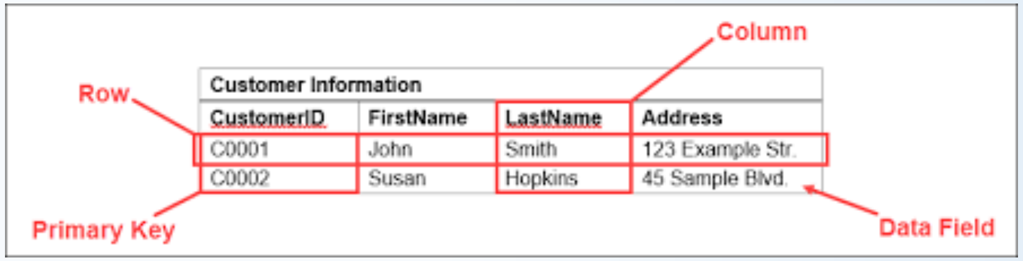
Field:
A field is a column in a table that is designed to maintain specific information about every record in the table.

Record or a Row:
A record is a horizontal entity in a table.

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

NULL value:
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.



Customer Information

CustomerID	FirstName	LastName	Address
C0001	John	Smith	123 Example Str.
C0002	Susan	Hopkins	45 Sample Blvd.

SQL

- SQL stands for Structured Query Language.
- This database language is mainly designed for maintaining the data in relational database management systems.
- It is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.
- **SQL is not a database system, but it is a query language.**

Why SQL?

•It also helps them to describe the structured data.

•It also helps in creating the view, stored procedure, and functions in the relational database.

•It allows you to define the data and modify that stored data in the relational database.

•It also allows SQL users to set the permissions or constraints on table columns, views, and stored procedures.

Data Types

Data Types

- Data types are used to represent the nature of the data that can be stored in the database table.
- For example, in a particular column of a table, if we want to store a string type of data then we will have to declare a string data type of this column.

Data types mainly classified into three categories for every database.

•String Data types
•Numeric Data types
•Date and time Data types

String Data Types

Data Type	Description
CHAR(Size)	Fixed length string that can contain numbers, letters, and special characters, Size can be 0 to 255 characters. Default is 1.
VARCHAR(Size)	Variable length string that can contain numbers, letters, and special characters, Size can be from 0 to 65535 characters.
BINARY(Size)	Same as CHAR() but stores binary byte strings. Its size parameter specifies the column length in the bytes. Default is 1.
VARBINARY(Size)	Same as VARCHAR() but stores binary byte strings. Its size parameter specifies the maximum column length in bytes.
TEXT(Size)	String that can contain a maximum length of 255 characters.
TINYTEXT	String with a maximum length of 255 characters.
MEDIUMTEXT	String with a maximum length of 16,777,215.
LONGTEXT	String with a maximum length of 4,294,967,295 characters.
ENUM(val1, val2, val3,...)	Used when a string object having only one value, chosen from a list of possible values. It contains 65535 values in an ENUM list. If you insert a value that is not in the list, a blank value will be inserted.
SET(val1,val2,val3,....)	Used to specify a string that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values at one time in a SET list.
BLOB(size)	Used for BLOBs (Binary Large Objects). It can hold up to 65,535 bytes.

Numeric Data Types

Data Type	Description
BIT(Size)	Used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. The default value is 1.
INT(size)	Used for the integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. The size parameter specifies the max display width that is 255.
INTEGER(size)	Same INT(size).
FLOAT(size, d)	Used to specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by **d** parameter.
FLOAT(p)	Used to specify a floating point number. MySQL used p parameter to determine whether to use FLOAT or DOUBLE. If p is between 0 to24, the data type becomes FLOAT (). If p is from 25 to 53, the data type becomes DOUBLE().
DOUBLE(size, d)	It is a normal size floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal is specified by d parameter.
DECIMAL(size, d)	It is used to specify a fixed point number. Its size parameter specifies the total number of digits. The number of digits after the decimal parameter is specified by **d** parameter. The maximum value for the size is 65, and the default value is 10. The maximum value for **d** is 30, and the default value is 0.
DEC(size, d)	Same as DECIMAL(size, d).
BOOL	Used to specify Boolean values true and false. Zero is considered as false, and nonzero values are considered as true.

Date and Time Data Types

Data Type	Description
DATE	It is used to specify date format YYYY-MM-DD. Its supported range is from '1000-01-01' to '9999-12-31'.
DATETIME(fsp)	It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to 9999-12-31 23:59:59'.
TIMESTAMP(fsp)	It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59'
YEAR	It is used to specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000.

Operators

Operators

- The SQL reserved words and characters are called operators, which are used with a WHERE clause in a SQL query.
- In SQL, an operator can either be a unary or binary operator.
- The unary operator uses only one operand for performing the unary operation, whereas the binary operator uses two operands for performing the binary operation.

Types of Operators:

1) Arithmetic Operators
2) Comparison Operators
3) Logical Operators
4) Set Operators

Arithmetic and Comparison Operators

Arithmetic Operators

Operator	Description
+ (Addition)	Adds values on either side of the operator.
- (Subtraction)	Subtracts right hand operand from left hand operand.
* (Multiplication)	Multiplies values on either side of the operator.
/ (Division)	Divides left hand operand by right hand operand.
% (Modulus)	Divides left hand operand by right hand operand and returns remainder.

Comparison Operators

Operator	Description
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
<>	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.
!<	Checks if the value of left operand is not less than the value of right operand, if yes then condition becomes true.
!>	Checks if the value of left operand is not greater than the value of right operand, if yes then condition becomes true.

Logical and Set Operators

Logical Operators

Operator	Description
ALL	The ALL operator is used to compare a value to all values in another value set.
AND	The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.
OR	The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.
BETWEEN	The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
IN	The IN operator is used to compare a value to a list of values that have been specified.
NOT	The NOT operator reverses the meaning of the logical operator with which it is used.
ANY	The ANY operator is used to compare a value to any applicable value in the list as per the condition.
LIKE	The LIKE operator is used to compare a value to similar values using wildcard(%) operators.

Set Operators

Operator	Description
Union	combines the result of two or more SELECT statements and provides the single output..
Union ALL	Union Operator is the same as the UNION operator, but the only difference is that it also shows duplicate record.
Intersect	shows the common record from two or more SELECT statements.
Minus	combines the result of two or more SELECT statements and shows only the results from the first data set.

SQL Commands

SQL Commands

- Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database.
- SQL uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks.

These SQL commands are mainly categorized into four categories as:

1) DDL – Data Definition Language

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database. (Example : CREATE,DROP,ALTER,TRUNCATE)

2) DQL – Data Query Language

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it. (**Example :** SELECT)

3) DML – Data Manipulation Language

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements. (**Example :** INSERT,UPDATE,DELETE)

4) DCL – Data Control Language

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system. (**Example :** GRANT,REVOKE)

SQL Keys

Key
a Key is defined as a column or a group of columns (attributes) leveraged to uniquely locate records in a table of a Relational Database.

1) Primary Key
Primary key is that column of table which does not have duplicate values, null values and its values can not be changed.
Each table should have only one primary key.

2) Unique Key
A group of one or more table fields/columns that uniquely identify a record in a database table is known as a unique key in MySQL Keys.
It's similar to a primary key in that it can only accept one null value and cannot have duplicate values.

3) Foreign Key
You can have more than one Foreign Key for a table where each Foreign Key references a Primary Key of various parent tables.

Database Commands

CREATE DATABASE

Used to create a new database

Syntax : CREATE DATABASE database_name

DROP DATABASE

Used to delete an existing database

Syntax : DROP DATABASE database_name

USE

Used to select a Database

Syntax : USE database_name

Table Commands

1) CREATE TABLE

Used to create a new table

Syntax :

CREATE TABLE tablename
(column1 data type **Key**,
column2 data type **Key**,
column3 data type,
...
columnN data type**);**

2) INSERT

Insert records in a table

Syntax:

INSERT INTO tablename (column1, column2,…)
VALUES (value1, value2,…);

3) DELETE

Delete specific rows from table

Syntax : **DELETE FROM** tablename **WHERE** condition

Note : If condition is not specified all the rows will be deleted

4) RENAME TABLE

Used to Rename an existing table

Syntax :

RENAME TABLE oldtablename **TO** newtablename

5) TRUNCATE TABLE

Used to delete all the rows from existing table

Syntax : **TRUNCATE TABLE** tablename

6) ALTER TABLE

add, modify, and delete columns of an existing table.

Syntax : **ALTER TABLE** tablename modification

7) DROP TABLE

Used to delete a table

Syntax : **DROP TABLE** tablename

SELECT Statements

SELECT

The SELECT statement is the most commonly used command in Structured Query Language.
It is used to access the records from one or more database tables.

Syntax:

SELECT Column_Name_1, Column_Name_2, ..., Column_Name_N **FROM** Table_Name;

OR

SELECT * **FROM** Table_Name; (To fetch all the records from the table)

WHERE

WHERE clause filters the records. It returns only those queries which fulfil the specific conditions.

Syntax :

SELECT * **FROM** table_name
WHERE conditions ;

ORDER BY

ORDER BY clause in SQL is used to sort the records based on the columns stored in the tables.

Syntax :

SELECT * **FROM** TableName **ORDER BY** ColumnName **ASC**; (For ascending order)

SELECT * **FROM** TableName **ORDER BY** ColumnName **DESC**; (For descending order)

SELECT Statements

GROUP BY

The **Group By** statement is used for organizing similar data into groups.
The data is further organized with the help of equivalent function.
It means, if different rows in a precise column have the same values, it will arrange those rows in a group.
Syntax :

SELECT column1, function_name(column2)
FROM tablename
WHERE condition
GROUP BY column1, column2
ORDER BY column1, column2;

Example:

SELECT DeptName, AVG(Salary) AS avg_sal
FROM DEPT
WHERE condition
GROUP BY DeptName
ORDER BY avg_sal ASC;

SELECT Statements

HAVING

The difference between the having and where clause in SQL is that the where clause cann*ot* be used with aggregates, but the having clause can.

The **where** clause works on row's data, not on aggregated data.

Syntax :

SELECT column1, function_name(column2)
FROM tablename
WHERE column1, column2
GROUP BY column1, column2
ORDER BY column1, column2
HAVING condition;

Example:

SELECT DeptName, AVG(Salary) AS avg_sal
FROM DEPT
GROUP BY DeptName
ORDER BY avg_sal ASC
HAVING avg_sal > 10000;

SQL Joins

Joins

Suppose we have one table we record demographic records of employees (Age, Gender, Postal Code…)
and one more table where we store their financial transactions records (Salary, Expenses …)
Each of these table have Employee ID in common.
Now if we want to combine the records of these two tables we can use join clause.

The SQL JOIN clause takes records from two or more tables in a database and combines it together.

SQL defines following types of JOIN :
1.inner join
2.left outer join
3.right outer join
4.full outer join

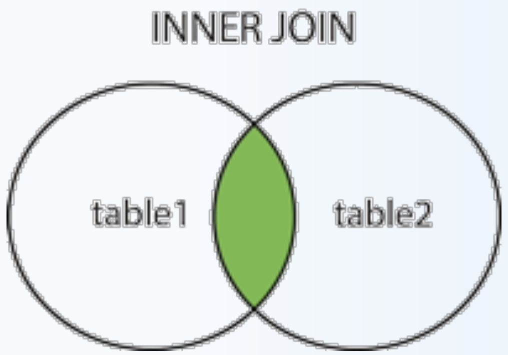
inner join

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

Syntax :

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

Note: column_name should be having some common values in both the tables

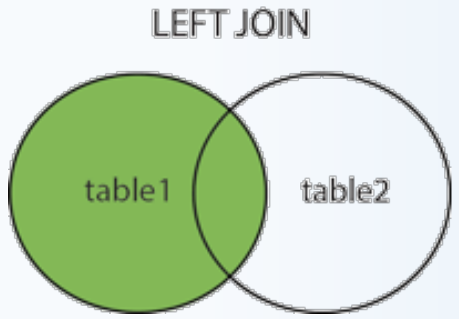


left join

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2).

Syntax :

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

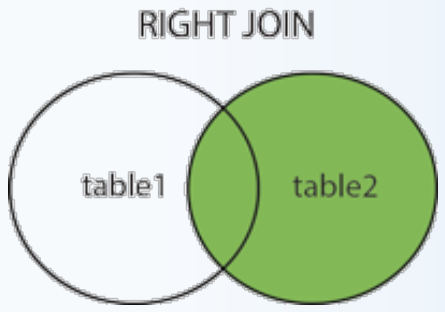


right join

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1).

Syntax :

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

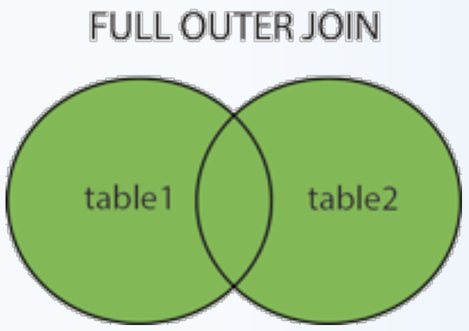


Full outer join

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

Syntax :

SELECT column_name(s)
FROM table1 **t1**
LEFT JOIN table2 **t2 ON t1.** column_name **= t2.** column_name
UNION ALL
SELECT column_name(s)
FROM table1 **t1**
RIGHT JOIN table2 **t2 ON t1.** column_name **= t2.** column_name



Keep Learning…………… Keep Coding…………… Keep going……………