RDBMS introduced by **E.F. Codd.**

## **NULL Values**

The NULL value of the table specifies that the field has been left blank during record creation. It is totally different from the value filled with zero or a field that contains space.

## **Data Integrity**

There are the following categories of data integrity exist with each RDBMS:

**Entity integrity**: It specifies that there should be no duplicate rows in a table.

**Domain integrity**: It enforces valid entries for a given column by restricting the type, the format, or the range of values.

**Referential integrity**: It specifies that rows cannot be deleted, which are used by other records.

**User-defined integrity**: It enforces some specific business rules that are defined by users. These rules are different from entity, domain or referential integrity.

|  |  |  |
| --- | --- | --- |
| **No.** | **DBMS** | **RDBMS** |
| 1) | DBMS applications store **data as file**. | RDBMS applications store **data in a tabular form**. |
| 2) | In DBMS, data is generally stored in either a hierarchical form or a navigational form. | In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables. |
| 3) | **Normalization is not** present in DBMS. | **Normalization is** present in RDBMS. |
| 4) | DBMS does **not apply any security** with regards to data manipulation. | RDBMS **defines the integrity constraint** for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property. |
| 5) | DBMS uses file system to store data, so there will be **no relation between the tables**. | in RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well. |
| 6) | DBMS has to provide some uniform methods to access the stored information. | RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information. |
| 7) | DBMS **does not support distributed database**. | RDBMS **supports distributed database**. |
| 8) | DBMS is meant to be for small organization and **deal with small data**. it supports **single user**. | RDBMS is designed to **handle large amount of data**. it supports **multiple users**. |
| 9) | Examples of DBMS are file systems, **xml**etc. | Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc. |

* **SQL** is not case sensitive. Generally SQL keywords are written in uppercase.
* SQL depends on relational algebra and tuple relational calculus.

|  |  |  |
| --- | --- | --- |
| **Data-type** | **Syntax** | **Explanation** |
| Integer | INTEGER | The integer data type is used to specify an integer value. |
| Smallint | SMALLINT | The smallint data type is used to specify small integer value. |
| Numeric | NUMERIC(P,S) | It specifies a numeric value. Here 'p' is precision value and 's' is scale value. |
| Real | REAL | The real integer is used to specify a single precision floating point number. |
| Decimal | DECIMAL(P,S) | It specifies a decimal value. Here 'p' is precision value and 's' is scale value. |
| Double precision | DOUBLE PRECISION | It specifies double precision floating point number. |
| Float | FLOAT(P) | It specifies floating-point value e.g. 12.3, 4.5 etc. Here, 'p' is precision value. |
| Character | CHAR(X) | Here, 'x' is the character's number to store. |
| Character varying | VARCHAR2(X) | Here, 'x' is the character's number to store |
| Bit | BIT(X) | Here, 'x' is the number of bits to store |
| Bit varying | BIT VARYING(X) | Here, 'x' is the number of bits to store (length can vary up to x). |
| Date | DATE | It stores year, month and days values. |
| Time | TIME | It stores hour, minute and second values |
| Timestamp | TIMESTAMP | The timestamp data type is used to store year, month, day, hour, minute and second values. |
| Time with time zone | TIME WITH TIME ZONE | It is exactly same as time but also store an offset from UTC of the time specified. |
| Timestamp with time zone | TIMESTAMP with TIME ZONE | It is same as timestamp but also stores an offset from UTC of the time specified. |

#### **We should always remember that database name should be unique in the RDBMS.**

In MySQL database, you need to select a database first before executing any query on table, view etc. To do so, we use following query:

1. USE DATABASE database\_name;

#### **Note: A table has a specified number of columns, but can have any number of rows.**

## **SQL CREATE TABLE Example in MySQL**

Let's see the command to create a table in MySQL database.

1. CREATE TABLE Employee
2. (
3. EmployeeID int,
4. FirstName varchar(255),
5. LastName varchar(255),
6. Email varchar(255),
7. AddressLine varchar(255),
8. City varchar(255)
9. );

## **SQL CREATE TABLE Example in Oracle**

Let's see the command to create a table in Oracle database.

1. CREATE TABLE Employee
2. (
3. EmployeeID number(10),
4. FirstName varchar2(255),
5. LastName varchar2(255),
6. Email varchar2(255),
7. AddressLine varchar2(255),
8. City varchar2(255)
9. );

## **SQL CREATE TABLE Example in Microsoft SQLServer**

Let's see the command to create a table in SQLServer database. It is same as MySQL and Oracle.

1. CREATE TABLE Employee
2. (
3. EmployeeID int,
4. FirstName varchar(255),
5. LastName varchar(255),
6. Email varchar(255),
7. AddressLine varchar(255),
8. City varchar(255)
9. );

|  |
| --- |
| [next>>](https://www.javatpoint.com/sql-copy-table)[<<prev](https://www.javatpoint.com/sql-rename-table) **SQL TRUNCATE TABLE** A truncate SQL statement is used to remove all rows (complete data) from a table. It is similar to the DELETE statement with no WHERE clause. **TRUNCATE TABLE Vs DELETE TABLE** Truncate table is faster and uses lesser resources than DELETE TABLE command. **TRUNCATE TABLE Vs DROP TABLE** Drop table command can also be used to delete complete table but it deletes table structure too. TRUNCATE TABLE doesn't delete the structure of the table. |

he concept of temporary table is introduced by SQL server. It helps developers in many ways:

**Temporary tables** can be created at run-time and can do all kinds of operations that a normal table can do. These temporary tables are created inside tempdb database.

There are two types of temp tables based on the behavior and scope.

1. Local Temp Variable
2. Global Temp Variable

## **Local Temp Variable**

Local temp tables are only available at current connection time. It is automatically deleted when user disconnects from instances. It is started with hash (#) sign.

1. CREATE TABLE #local temp table (
2. User id int,
3. Username varchar (50),
4. User address varchar (150)
5. )

## **Global Temp Variable**

Global temp tables name starts with double hash (##). Once this table is created, it is like a permanent table. It is always ready for all users and not deleted until the total connection is withdrawn.

1. CREATE TABLE ##new global temp table (
2. User id int,
3. User name varchar (50),
4. User address varchar (150)
5. )

### **SQL ALTER TABLE DROP Column**

The syntax of alter table drop column is given below:

1. ALTER TABLE table\_name DROP COLUMN column\_name;

### **SQL ALTER TABLE RENAME Column**

The syntax of alter table rename column is given below:

1. ALTER TABLE table\_name
2. RENAME COLUMN old\_name to new\_name;

a point is notable that only SELECT and FROM statements are necessary in SQL SELECT statements. Other clauses like WHERE, GROUP BY, ORDER BY, HAVING may be optional.

In simple words, we can say that SELECT UNIQUE statement is used to retrieve a unique or distinct element from the table.

Let's see the syntax of select unique statement.

1. SELECT UNIQUE column\_name
2. FROM table\_name;

#### **Note: SQL SELECT UNIQUE and SQL SELECT DISTINCT statements are same.**