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 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

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](https://www.tutorialspoint.com/sql/sql-not-null.htm) − Ensures that a column cannot have a NULL value.
* [DEFAULT Constraint](https://www.tutorialspoint.com/sql/sql-default.htm) − Provides a default value for a column when none is specified.
* [UNIQUE Constraint](https://www.tutorialspoint.com/sql/sql-unique.htm) − Ensures that all the values in a column are different.
* [PRIMARY Key](https://www.tutorialspoint.com/sql/sql-primary-key.htm) − Uniquely identifies each row/record in a database table.
* [FOREIGN Key](https://www.tutorialspoint.com/sql/sql-foreign-key.htm) − Uniquely identifies a row/record in any another database table.
* [CHECK Constraint](https://www.tutorialspoint.com/sql/sql-check.htm) − The CHECK constraint ensures that all values in a column satisfy certain conditions.
* [INDEX](https://www.tutorialspoint.com/sql/sql-index.htm) − 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.

Database Normalization

Database normalization is the process of efficiently organizing data in a database. There are two reasons of this normalization process −

* Eliminating redundant data, for example, storing the same data in more than one table.
* Ensuring data dependencies make sense.

Both these reasons are worthy goals as they reduce the amount of space a database consumes and ensures that data is logically stored. Normalization consists of a series of guidelines that help guide you in creating a good database structure.

Normalization guidelines are divided into normal forms; think of a form as the format or the way a database structure is laid out. The aim of normal forms is to organize the database structure, so that it complies with the rules of first normal form, then second normal form and finally the third normal form.

It is your choice to take it further and go to the fourth normal form, fifth normal form and so on, but in general, the third normal form is more than enough.

* [First Normal Form (1NF)](https://www.tutorialspoint.com/sql/first-normal-form.htm)
* [Second Normal Form (2NF)](https://www.tutorialspoint.com/sql/second-normal-form.htm)
* [Third Normal Form (3NF)](https://www.tutorialspoint.com/sql/third-normal-form.htm)

The First normal form (1NF) sets basic rules for an organized database −

* Define the data items required, because they become the columns in a table.
* Place the related data items in a table.
* Ensure that there are no repeating groups of data.
* Ensure that there is a primary key.

First Rule of 1NF

You must define the data items. This means looking at the data to be stored, organizing the data into columns, defining what type of data each column contains and then finally putting the related columns into their own table.

For example, you put all the columns relating to locations of meetings in the Location table, those relating to members in the MemberDetails table and so on.

Second Rule of 1NF

The next step is ensuring that there are no repeating groups of data. Consider we have the following table −

CREATE TABLE CUSTOMERS(

ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25),

ORDERS VARCHAR(155)

);

So, if we populate this table for a single customer having multiple orders, then it would be something as shown below −

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **AGE** | **ADDRESS** | **ORDERS** |
| 100 | Sachin | 36 | Lower West Side | Cannon XL-200 |
| 100 | Sachin | 36 | Lower West Side | Battery XL-200 |
| 100 | Sachin | 36 | Lower West Side | Tripod Large |

But as per the 1NF, we need to ensure that there are no repeating groups of data. So, let us break the above table into two parts and then join them using a key as shown in the following program −

CUSTOMERS table −

CREATE TABLE CUSTOMERS(

ID INT NOT NULL,

NAME VARCHAR (20) NOT NULL,

AGE INT NOT NULL,

ADDRESS CHAR (25),

PRIMARY KEY (ID)

);

This table would have the following record −

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NAME** | **AGE** | **ADDRESS** |
| 100 | Sachin | 36 | Lower West Side |

ORDERS table −

CREATE TABLE ORDERS(

ID INT NOT NULL,

CUSTOMER\_ID INT NOT NULL,

ORDERS VARCHAR(155),

PRIMARY KEY (ID)

);

This table would have the following records −

|  |  |  |
| --- | --- | --- |
| **ID** | **CUSTOMER\_ID** | **ORDERS** |
| 10 | 100 | Cannon XL-200 |
| 11 | 100 | Battery XL-200 |
| 12 | 100 | Tripod Large |

Third Rule of 1NF

The final rule of the first normal form, create a primary key for each table which we have already created.

The Second Normal Form states that it should meet all the rules for 1NF and there must be no partial dependences of any of the columns on the primary key −

Consider a customer-order relation and you want to store customer ID, customer name, order ID and order detail and the date of purchase −

CREATE TABLE CUSTOMERS(

CUST\_ID INT NOT NULL,

CUST\_NAME VARCHAR (20) NOT NULL,

ORDER\_ID INT NOT NULL,

ORDER\_DETAIL VARCHAR (20) NOT NULL,

SALE\_DATE DATETIME,

PRIMARY KEY (CUST\_ID, ORDER\_ID)

);

This table is in the first normal form; in that it obeys all the rules of the first normal form. In this table, the primary key consists of the CUST\_ID and the ORDER\_ID. Combined, they are unique assuming the same customer would hardly order the same thing.

However, the table is not in the second normal form because there are partial dependencies of primary keys and columns. CUST\_NAME is dependent on CUST\_ID and there's no real link between a customer's name and what he purchased. The order detail and purchase date are also dependent on the ORDER\_ID, but they are not dependent on the CUST\_ID, because there is no link between a CUST\_ID and an ORDER\_DETAIL or their SALE\_DATE.

To make this table comply with the second normal form, you need to separate the columns into three tables.

First, create a table to store the customer details as shown in the code block below −

CREATE TABLE CUSTOMERS(

CUST\_ID INT NOT NULL,

CUST\_NAME VARCHAR (20) NOT NULL,

PRIMARY KEY (CUST\_ID)

);

The next step is to create a table to store the details of each order −

CREATE TABLE ORDERS(

ORDER\_ID INT NOT NULL,

ORDER\_DETAIL VARCHAR (20) NOT NULL,

PRIMARY KEY (ORDER\_ID)

);

Finally, create a third table storing just the CUST\_ID and the ORDER\_ID to keep a track of all the orders for a customer −

CREATE TABLE CUSTMERORDERS(

CUST\_ID INT NOT NULL,

ORDER\_ID INT NOT NULL,

SALE\_DATE DATETIME,

PRIMARY KEY (CUST\_ID, ORDER\_ID)

);