What is Database?

What is Database?

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

You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

Database handlers create a database in such a way that only one set of software program provides access of data to all the users.

The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data.

There are many **dynamic websites** on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.

There are many **databases available** like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

SQL or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

A cylindrical structure is used to display the image of a database.

Evolution of Databases

The database has completed more than 50 years of journey of its evolution from flat-file system to relational and objects relational systems. It has gone through several generations.

The Evolution

File-Based

1968 was the year when File-Based database were introduced. In file-based databases, data was maintained in a flat file. Though files have many advantages, there are several limitations.

One of the major advantages is that the file system has various access methods, e.g., sequential, indexed, and random.

It requires extensive programming in a third-generation language such as COBOL, BASIC.

Hierarchical Data Model

1968-1980 was the era of the Hierarchical Database. Prominent hierarchical database model was IBM's first DBMS. It was called IMS (Information Management System).

In this model, files are related in a parent/child manner.

Below diagram represents Hierarchical Data Model. Small circle represents objects.

What is Database Management System (DBMS)?

DBMS (Data Base Management System)

Database management System is software which is used to store and retrieve the database. For example, Oracle, MySQL, etc.; these are some popular DBMS tools.

- DBMS provides the interface to perform the various operations like creation, deletion, modification, etc.
- DBMS allows the user to create their databases as per their requirement.
- DBMS accepts the request from the application and provides specific data through the operating system.
- DBMS contains the group of programs which acts according to the user instruction.
- It provides security to the database.

Advantage of DBMS

Controls redundancy

It stores all the data in a single database file, so it can control data redundancy.

Data sharing

An authorized user can share the data among multiple users.

Backup

It providesBackup and recovery subsystem. This recovery system creates automatic data from system failure and restores data if required.

Multiple user interfaces

It provides a different type of user interfaces like GUI, application interfaces.

Disadvantage of DBMS

Size

It occupies large disk space and large memory to run efficiently.

Cost

DBMS requires a high-speed data processor and larger memory to run DBMS software, so it is costly.

Complexity

DBMS creates additional complexity and requirements.

File Based System Issues

File-based System

One way to keep information on a computer is to store it in permanent files. A company system has a number of application programs; each of them is designed to manipulate data files. These application programs have been written at the request of the users in the organization. New applications are added to the system as the need arises. The system just described is called the *file-based system*.

Consider a traditional banking system that uses the file-based system to manage the organization's data shown in Figure 1.1. As we can see, there are different departments in the bank. Each has its own applications that manage and manipulate different data files. For banking systems, the programs may be used to debit or credit an account, find the balance of an account, add a new mortgage loan and generate monthly statements.

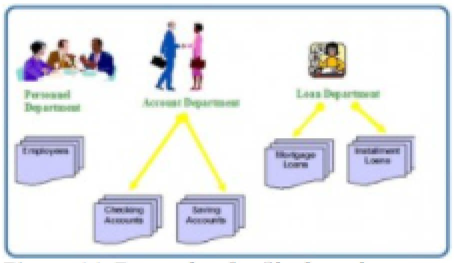


Figure 1.1. Example of a file-based system used by banks to manage data.

Disadvantages of the file-based approach

Using the file-based system to keep organizational information has a number of disadvantages. Listed below are five examples.

Data redundancy

Often, within an organization, files and applications are created by different programmers from various departments over long periods of time. This can lead to *data redundancy*, a situation that occurs in a database when a field needs to be updated in more than one table. This practice can lead to several problems such as:

- Inconsistency in data format
- The same information being kept in several different places (files)
- *Data inconsistency*, a situation where various copies of the same data are conflicting, wastes storage space and duplicates effort

Data isolation

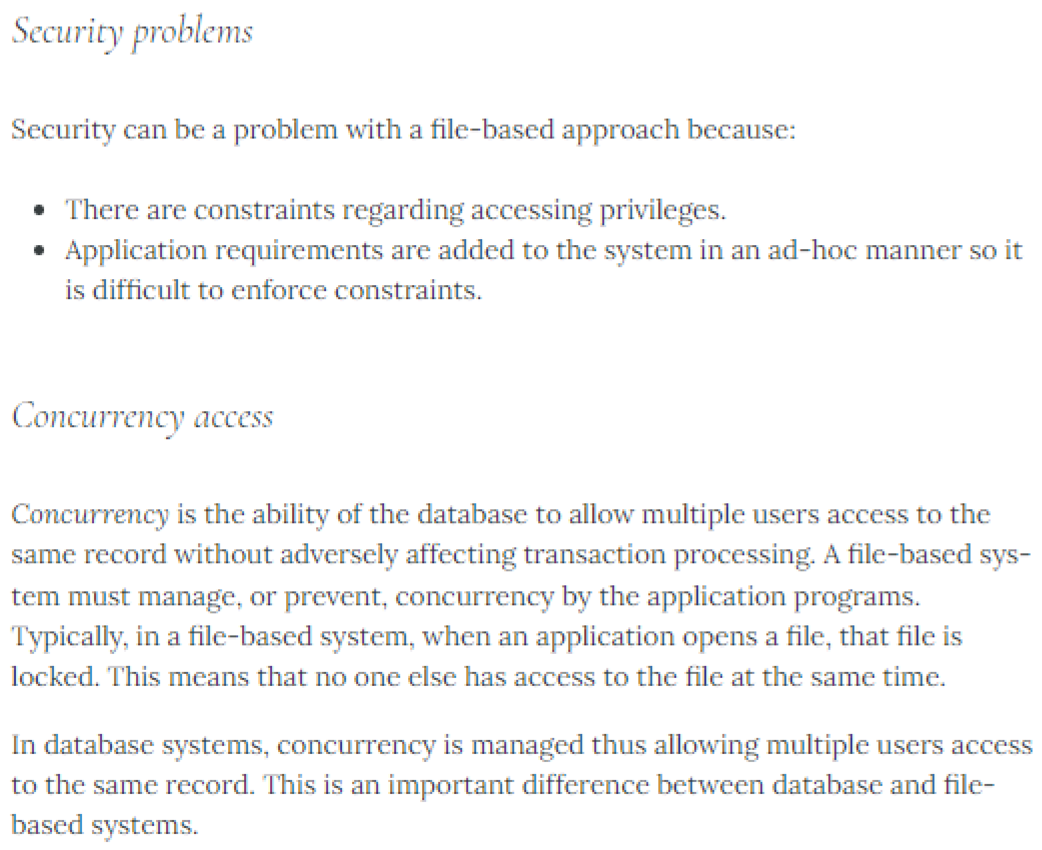
Data isolation is a property that determines when and how changes made by one operation become visible to other concurrent users and systems. This issue occurs in a concurrency situation. This is a problem because:

- It is difficult for new applications to retrieve the appropriate data, which might be stored in various files.

Integrity problems

Problems with *data integrity* is another disadvantage of using a file-based system. It refers to the maintenance and assurance that the data in a database are correct and consistent. Factors to consider when addressing this issue are:

- Data values must satisfy certain consistency constraints that are specified in the application programs.
- It is difficult to make changes to the application programs in order to enforce new constraints.



Security problems

Security can be a problem with a file-based approach because:

- There are constraints regarding accessing privileges.
- Application requirements are added to the system in an ad-hoc manner so it is difficult to enforce constraints.

Concurrency access

Concurrency is the ability of the database to allow multiple users access to the same record without adversely affecting transaction processing. A file-based system must manage, or prevent, concurrency by the application programs. Typically, in a file-based system, when an application opens a file, that file is locked. This means that no one else has access to the file at the same time.

In database systems, concurrency is managed thus allowing multiple users access to the same record. This is an important difference between database and file-based systems.

Adding and Dropping Constraints

Joins (equi, inner, outer, self, natural, using keyword)

Roles in DBMS

Privilege and Roles in DBMS

Last Updated : 21 Jan, 2021

Confidentiality, integrity, and availability are the stamps of database security. Authorization is the allowance to the user or process to access the set of objects. The type of access granted can be any like, read-only, read, and write. Privilege means different Data Manipulation Language(DML) operations which can be performed by the user on data like INSERT, UPDATE, SELECT and DELETE, etc.

There are two methods by which access control is performed is done by using the following.

1. Privileges
2. Roles

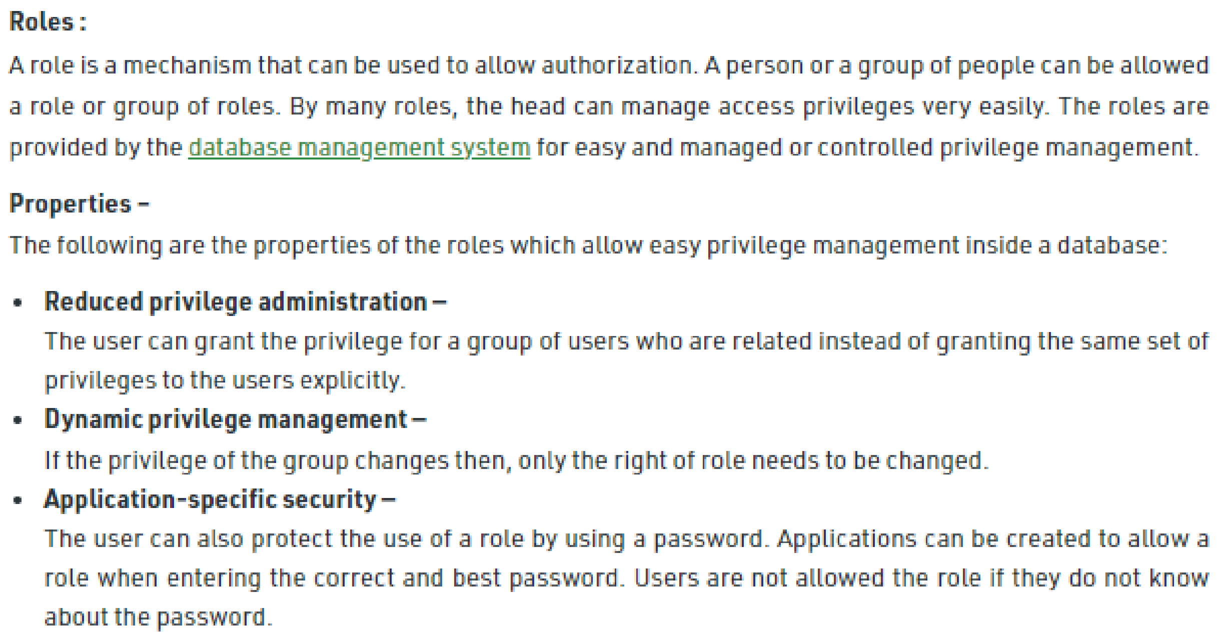
Let's discuss one by one.

Privileges :
The authority or permission to access a named object as advised manner, for example, permission to access a table. Privileges can allow permitting a particular user to connect to the database. In, other words privileges are the allowance to the database by the database object.

- **Database privileges –**
  A privilege is permission to execute one particular type of SQL statement or access a second persons' object. Database privilege controls the use of computing resources. Database privilege does not apply to the Database administrator of the database.

- **System privileges –**
  A system privilege is the right to perform an activity on a specific type of object. for example, the privilege to delete rows of any table in a database is system privilege. There are a total of 60 different system privileges. System privileges allow users to CREATE, ALTER, or DROP the database objects.

- **Object privilege –**
  An object privilege is a privilege to perform a specific action on a particular table, function, or package. For example, the right to delete rows from a table is an object privilege. For example, let us consider a row of table GEEKSFORGEEKS that contains the name of the employee who is no longer a part of the organization, then deleting that row is considered as an object privilege. Object privilege allows the user to INSERT, DELETE, UPDATE, or SELECT the data in the database object.



Roles :

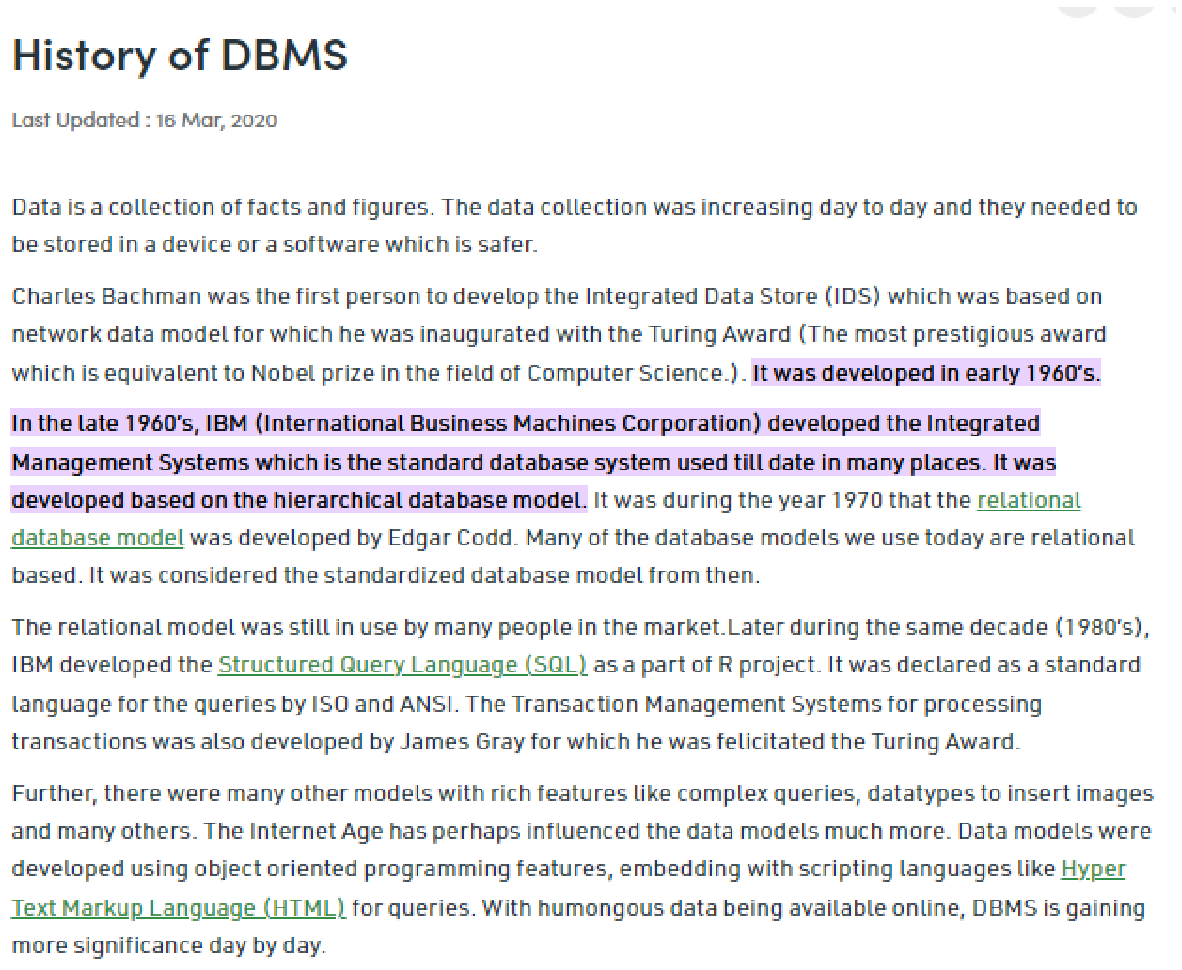
A role is a mechanism that can be used to allow authorization. A person or a group of people can be allowed a role or group of roles. By many roles, the head can manage access privileges very easily. The roles are provided by the database management system for easy and managed or controlled privilege management.

Properties –

The following are the properties of the roles which allow easy privilege management inside a database:

- **Reduced privilege administration –**
  The user can grant the privilege for a group of users who are related instead of granting the same set of privileges to the users explicitly.
- **Dynamic privilege management –**
  If the privilege of the group changes then, only the right of role needs to be changed.
- **Application-specific security –**
  The user can also protect the use of a role by using a password. Applications can be created to allow a role when entering the correct and best password. Users are not allowed the role if they do not know about the password.

History of DBMS



History of DBMS

Last Updated : 16 Mar, 2020

Data is a collection of facts and figures. The data collection was increasing day to day and they needed to be stored in a device or a software which is safer.

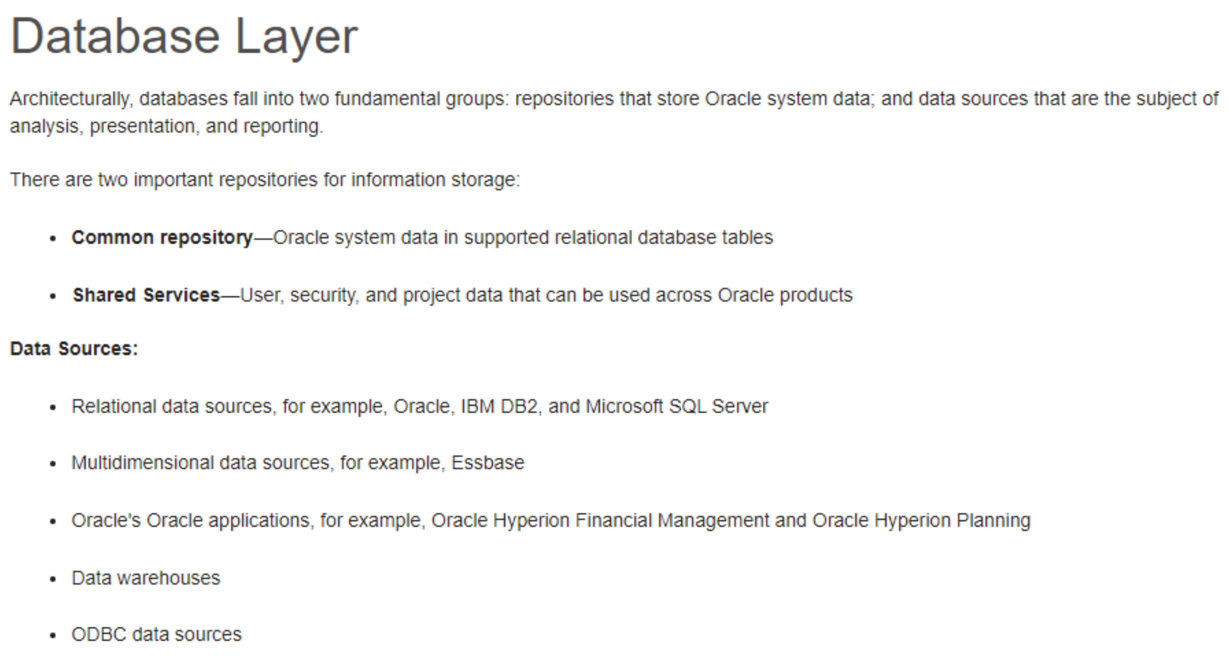
Charles Bachman was the first person to develop the Integrated Data Store (IDS) which was based on network data model for which he was inaugurated with the Turing Award (The most prestigious award which is equivalent to Nobel prize in the field of Computer Science.). **It was developed in early 1960's.**

In the late 1960's, IBM (International Business Machines Corporation) developed the Integrated Management Systems which is the standard database system used till date in many places. It was developed based on the hierarchical database model. It was during the year 1970 that the relational database model was developed by Edgar Codd. Many of the database models we use today are relational based. It was considered the standardized database model from then.

The relational model was still in use by many people in the market.Later during the same decade (1980's), IBM developed the Structured Query Language (SQL) as a part of R project. It was declared as a standard language for the queries by ISO and ANSI. The Transaction Management Systems for processing transactions was also developed by James Gray for which he was felicitated the Turing Award.

Further, there were many other models with rich features like complex queries, datatypes to insert images and many others. The Internet Age has perhaps influenced the data models much more. Data models were developed using object oriented programming features, embedding with scripting languages like Hyper Text Markup Language (HTML) for queries. With humongous data being available online, DBMS is gaining more significance day by day.

Database Layer



Database Layer

Architecturally, databases fall into two fundamental groups: repositories that store Oracle system data; and data sources that are the subject of analysis, presentation, and reporting.

There are two important repositories for information storage:

- **Common repository**—Oracle system data in supported relational database tables
- **Shared Services**—User, security, and project data that can be used across Oracle products

Data Sources:

- Relational data sources, for example, Oracle, IBM DB2, and Microsoft SQL Server
- Multidimensional data sources, for example, Essbase
- Oracle's Oracle applications, for example, Oracle Hyperion Financial Management and Oracle Hyperion Planning
- Data warehouses
- ODBC data sources

Basic Concepts of Relational DB

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.

Types of Keys in DB

Types of SQL Keys

We have the following types of keys in SQL which are used to fetch records from tables and to make relationships among tables or views.

01. Super Key

A super key is a set of one or more than one key that can be used to identify a record uniquely in a table. **Example:** Primary key, Unique key, Alternate key are a subset of Super Keys.

02. Candidate Key

A Candidate Key is a set of one or more fields/columns that can identify a record uniquely in a table. There can be multiple Candidate Keys in one table. Each Candidate Key can work as a Primary Key.

Example: In the below diagram ID, RollNo and EnrollNo are Candidate Keys since all these three fields can work as Primary Key.

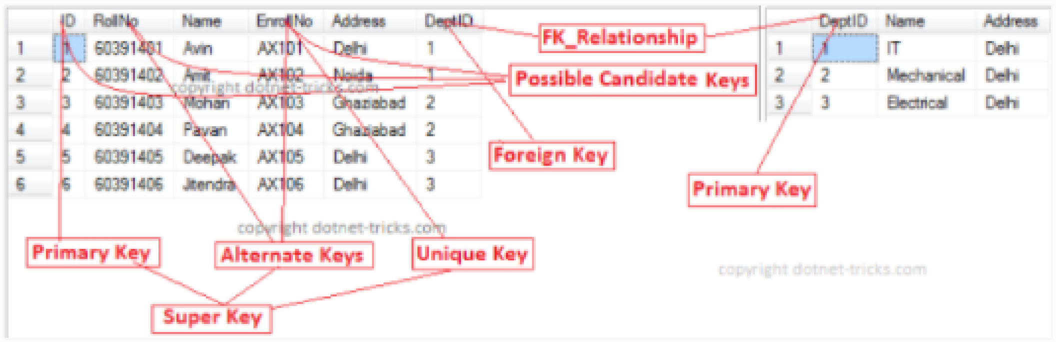
03. Primary Key

A primary key is a set of one or more fields/columns of a table that uniquely identify a record in a database table. It can not accept null, duplicate values. Only one Candidate Key can be Primary Key.

04. Alternate key

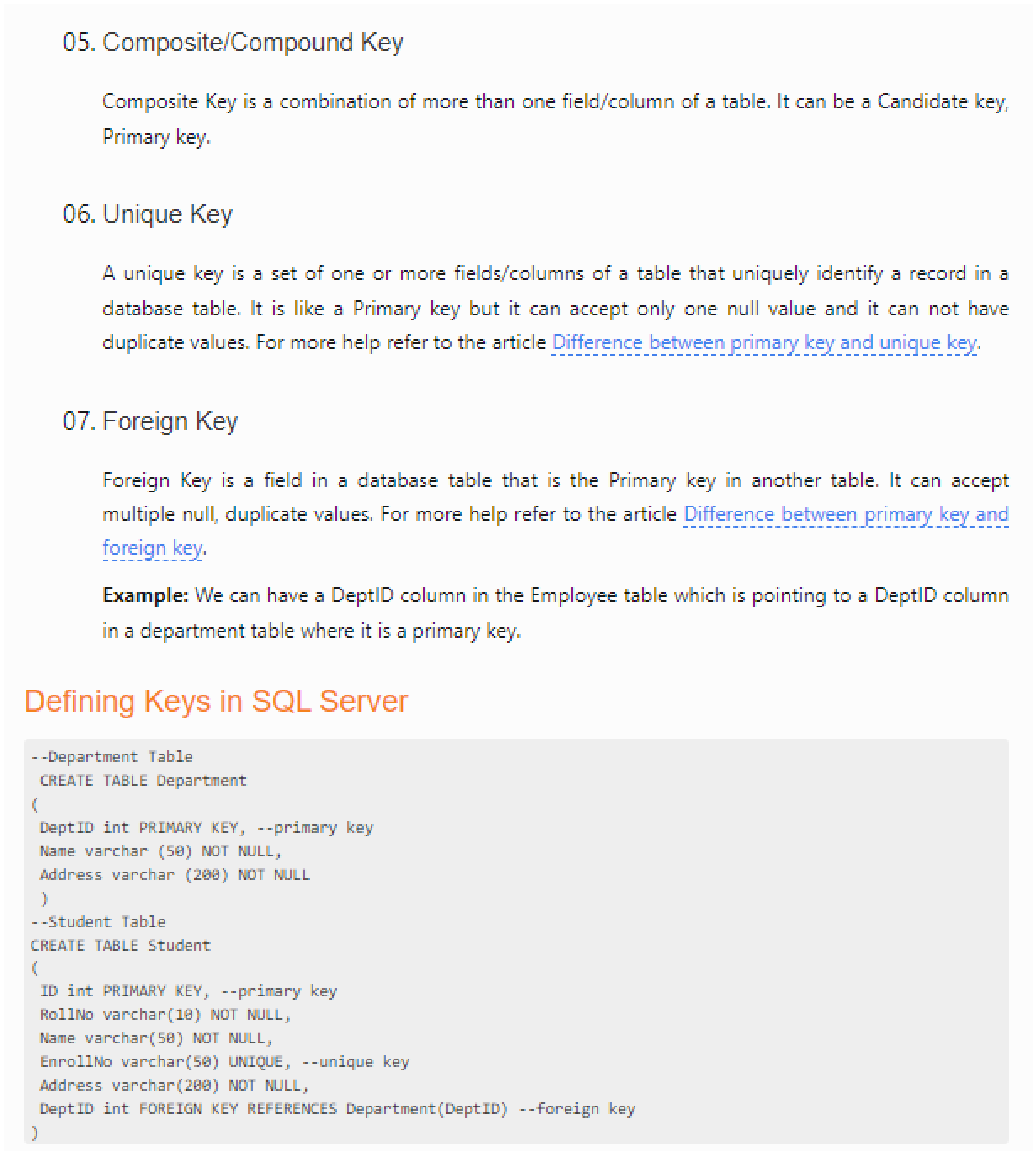
An Alternate key is a key that can work as a primary key. Basically, it is a candidate key that currently is not a primary key.

Example: In the below diagram RollNo and EnrollNo become Alternate Keys when we define ID as the Primary Key.



	ID	RollNo	Name	EnrollNo	Address	DeptID
1	1	60391401	Avin	AX101	Delhi	1
2	2	60391402	Amit	AX102	Noida	1
3	3	60391403	Mohan	AX103	Ghaziabad	2
4	4	60391404	Pavan	AX104	Ghaziabad	2
5	5	60391405	Deepak	AX105	Delhi	3
6	6	60391406	Jitendra	AX106	Delhi	3

	DeptID	Name	Address
1	1	IT	Delhi
2	2	Mechanical	Delhi
3	3	Electrical	Delhi



05. Composite/Compound Key

Composite Key is a combination of more than one field/column of a table. It can be a Candidate key, Primary key.

06. Unique Key

A unique key is a set of one or more fields/columns of a table that uniquely identify a record in a database table. It is like a Primary key but it can accept only one null value and it can not have duplicate values. For more help refer to the article Difference between primary key and unique key.

07. Foreign Key

Foreign Key is a field in a database table that is the Primary key in another table. It can accept multiple null, duplicate values. For more help refer to the article Difference between primary key and foreign key.

Example: We can have a DeptID column in the Employee table which is pointing to a DeptID column in a department table where it is a primary key.

Defining Keys in SQL Server

```
--Department Table
 CREATE TABLE Department
(
 DeptID int PRIMARY KEY, --primary key
 Name varchar (50) NOT NULL,
 Address varchar (200) NOT NULL
 )
--Student Table
CREATE TABLE Student
(
 ID int PRIMARY KEY, --primary key
 RollNo varchar(10) NOT NULL,
 Name varchar(50) NOT NULL,
 EnrollNo varchar(50) UNIQUE, --unique key
 Address varchar(200) NOT NULL,
 DeptID int FOREIGN KEY REFERENCES Department(DeptID) --foreign key
)
```

Attributes in DB

What Does Attribute Mean?

In general, an attribute is a characteristic. In a database management system (DBMS), an attribute refers to a database component, such as a table.

It also may refer to a database field. Attributes describe the instances in the column of a database.

Techopedia Explains Attribute

In relational databases, attributes are the describing characteristics or properties that define all items pertaining to a certain category applied to all cells of a column.

The rows, instead, are called tuples, and represent data sets applied to a single entity to uniquely identify each item. Attributes are, therefore, the characteristics of every individual tuple that help describe its unique properties.

Think of a table in a relational database as being analogous to an electronic spreadsheet. An attribute is simply one non-null cell in the spreadsheet, or the conjunction of a column and row.

It stores only one piece of data about the object represented by the table in which the attribute belongs. For example, the tuple can be an Invoice entity. The attributes of an invoice might be Price, Number, Date or Paid/unpaid.

Beyond the self-explanatory simple or single-valued attributes, there are several types of attributes available.

- **Composite attribute:** is an attribute composed of several other simple attributes. For example, the Address attribute of an Employee entity could consist of the Street, City, Postal code and Country attributes.

- **Multivalued attribute:** is an attribute where more than one description can be provided. For example, an Employee entity may have more than one Email ID attributes in the same cell.

- **Key attribute or primary attribute:** is an ID, key, letter or number that uniquely identifies that item. For example, it can be the number of a certain invoice (e.g. the individual ID of that invoice). A table that contains a single key attribute is considered a strong entity. However, a table might contain more than one key attribute if it's derived from other tables.

- **Derived attribute**: as the name implies, these are derived from other attributes, either directly or through specific formula results. For example, the Age attribute of an Employee could be derived from the Date of Birth attribute. In other instances, a formula might calculate the VAT of a certain payment, so that whenever the cell with the attribute Payment is filled, the cell with the derived attribute VAT automatically calculates its value.

Aliases

SQL | Aliases

Difficulty Level : **Medium** • Last Updated : 09 Jan, 2019

Aliases are the temporary names given to table or column for the purpose of a particular SQL query. It is used when name of column or table is used other than their original names, but the modified name is only temporary.

- Aliases are created to make table or column names more readable.
- The renaming is just a temporary change and table name does not change in the original database.
- Aliases are useful when table or column names are big or not very readable.
- These are preferred when there are more than one table involved in a query.

Basic Syntax:

- For column alias:

```
SELECT column as alias_name FROM table_name;
column: fields in the table
alias_name: temporary alias name to be used in replacement of original column name
table_name: name of table
```

- For table alias:

```
SELECT column FROM table_name as alias_name;
column: fields in the table
table_name: name of table
alias_name: temporary alias name to be used in replacement of original table name
```

Student_Details

ROLL_NO	Branch	Grade
1	Information Technology	O
2	Computer Science	E
3	Computer Science	O
4	Mechanical Engineering	A

Tuple

What is a tuple in the database?

A tuple is simply a row contained in a table in the tablespace. A table usually contains columns and rows in which rows stand for records while columns stand for attributes. A single row of a table that has a single record for such a relation is known as a tuple. A Tuple is, therefore, a single entry in a table; it is also called a row or record. They usually represent a set of related data, in Math it is simply an ordered list of elements.

What Are The Relational Model Concepts?

Some of the relational model concepts include the following:

- **Tables**: here, relations are usually saved in the format of a table in the relational model. It is usually saved together with its entities. The basic features of a table include columns and rows where the columns represent attributes while the rows stand for records.
- **Attribute:** each relation is usually defined by certain properties known as attributes such as Name.
- **Degree:** this is simply the total of attributes which is referred to as the degree of the relation.
- **Tuple:** This is simply a single table or row that contains one record.
- **Relation instance:** this is simply a finite number of tuples contained in the RDBMS system and they usually never have multiple tuples.
- **Cardinality:** This is the total of rows as indicated in the table.
- **Column:** this stands for a set of values which represent a specific attribute.
- **Attribute domain:** each attribute usually comes with some pre-defined scope and value referred to as the attribute domain.
- **Relation key:** each row usually has at least one attribute and they are called relation keys.

Cardinality & Relationship Types

Cardinality in DBMS

Difficulty Level : **Medium** • Last Updated : 10 Feb, 2022

In database management, cardinality plays an important role. Here cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship. Types of cardinality in between tables are:

- one-to-one
- one-to-many
- many-to-one
- many-to-many

Mapping Cardinalities

In a database, the mapping cardinality or cardinality ratio means to denote the number of entities to which another entity can be linked through a certain relation set. Mapping cardinality is most useful in describing binary relation sets, although they can contribute to the description of relation sets containing more than two entity sets. Here, we will focus only on binary relation sets means we will find the relation between entity sets A and B for the set R. So we can map any one of following the cardinality:

1. One-to-one: In this type of cardinality mapping, an entity in A is connected to at most one entity in B. Or we can say that a unit or item in B is connected to at most one unit or item in A.

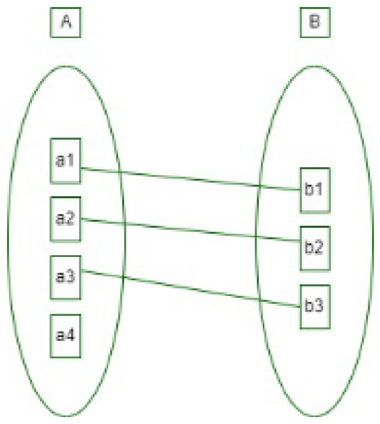
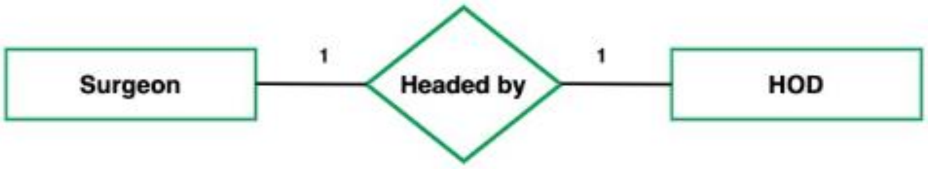


Figure 1

Example:

In a particular hospital, the surgeon department has one head of department. They both serve one-to-one relationships.



2. One-to-many: In this type of cardinality mapping, an entity in A is associated with any number of entities in B. Or we can say that one unit or item in B can be connected to at most one unit or item in A.

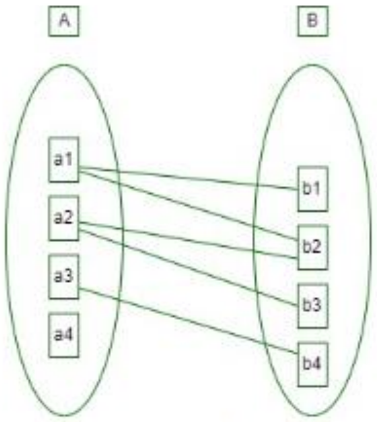
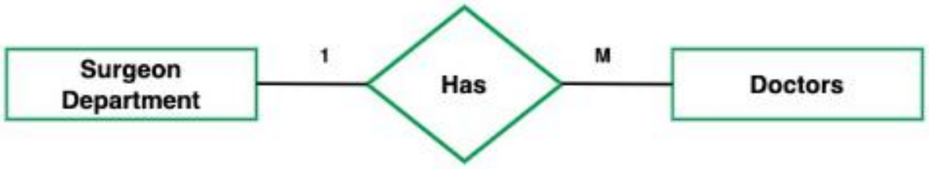


Figure 2

Example:

In a particular hospital, the surgeon department has multiple doctors. They serve one-to-many relationships.



3. Many-to-one: In this type of cardinality mapping, an entity in A is connected to at most one entity in B. Or we can say a unit or item in B can be associated with any number (zero or more) of entities or items in A.

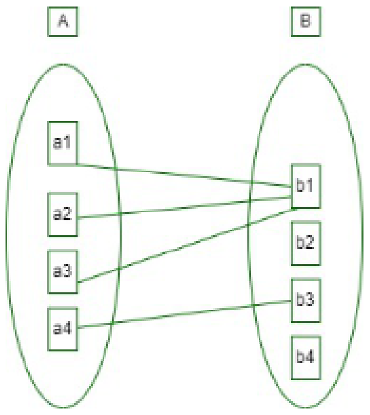
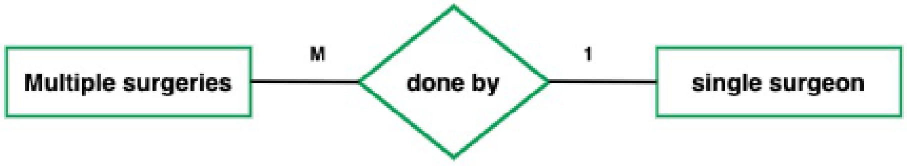


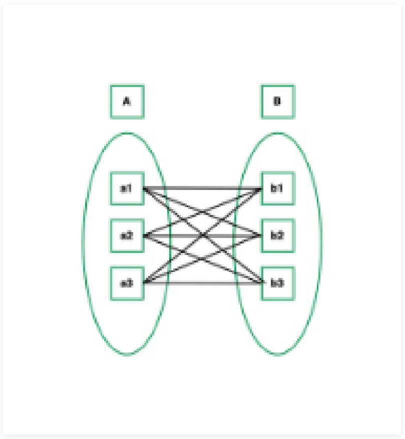
Figure 3

Example:

In a particular hospital, multiple surgeries are done by a single surgeon. Such a type of relationship is known as a many-to-many relationship.

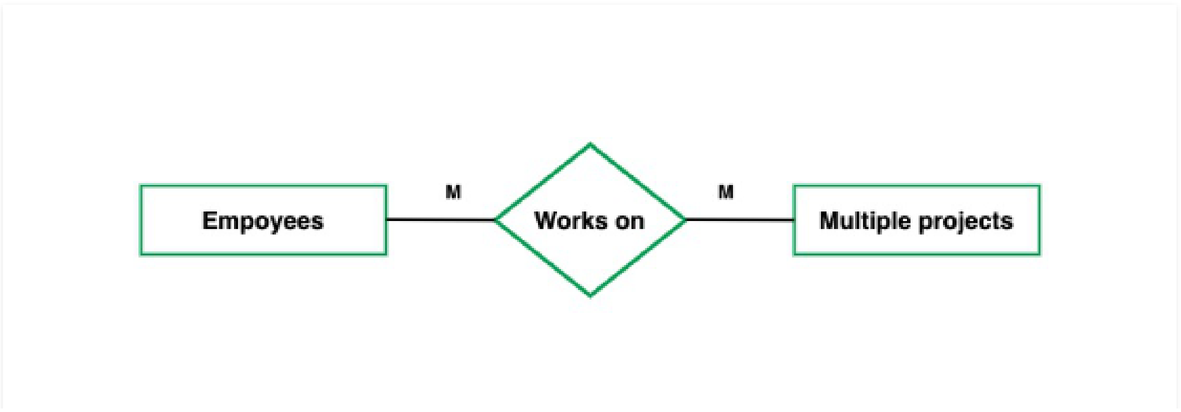


4. Many-to-many: In this type of cardinality mapping, an entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.



Example:

In a particular company, multiple people work on multiple projects. They serve many-to-many relationships.



Degree

Degree of Relations in DBMS

Last Updated : 03 Sep, 2021

We are living in a world where every entity has relationships with one other whether a living or non-living being. For example, you are a single entity but you share different relations with your family, friends. Even within a family-like, you are the son of your father at the same time you are also a sibling of your brother. Similarly the relationships exist in Database Management (DBMS). In this article, we are going to learn about what is the degree of relationships, and the types of relationships.

Degree of Relationship

In DBMS, a degree of relationship represents the number of entity types that associate in a relationship. For example, we have two entities, one is a student and the other is a bag and they are connected with the primary key and foreign key. So, here we can see that the degree of relationship is 2 as 2 entities are associating in a relationship.

Types of degree

Now, based on the number of linked entity types, we have 4 types of degrees of relationships.

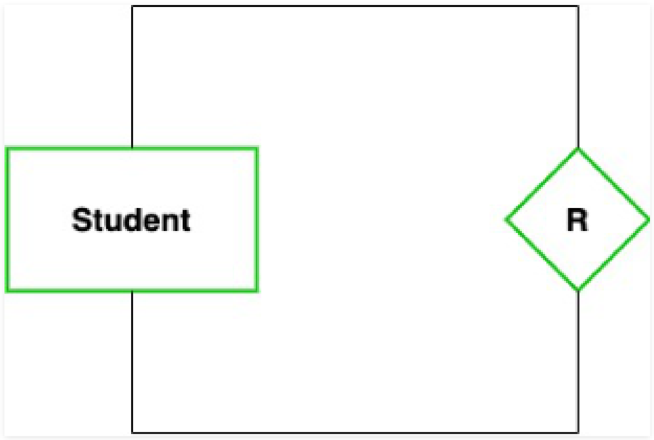
1. Unary
2. Binary
3. Ternary
4. N-ary

Let's discuss them one by one with the help of examples.

Unary

In this type of relationship, both the associating entity type are the same. So, we can say that unary relationships exist when both entity types are the same and we call them the degree of relationship is 1. Or in other words, in a relation only one entity set is participating then such type of relationship is known as a unary relationship.

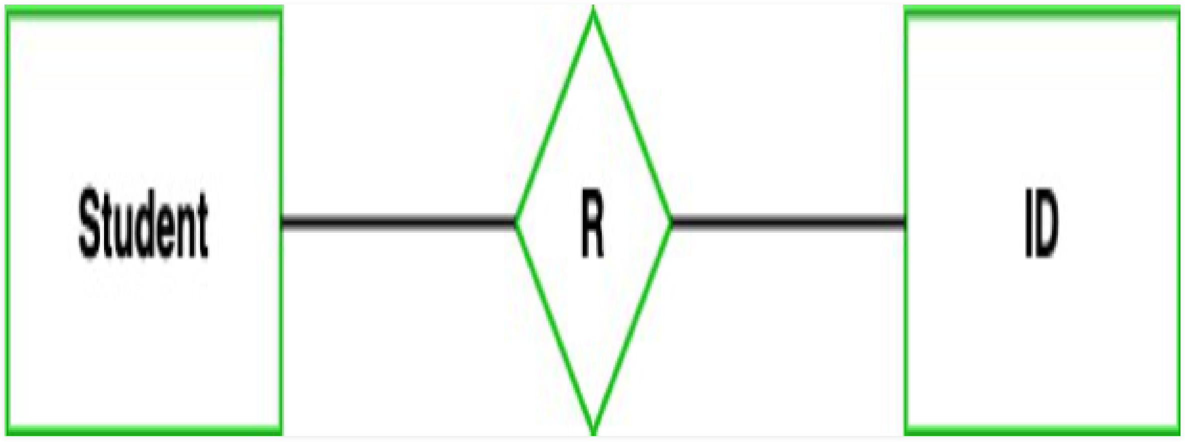
Example: In a particular class, we have many students, there are monitors too. So, here class monitors are also students. Thus, we can say that only students are participating here. So the degree of such type of relationship is 1.



Binary (degree 2)

In a Binary relationship, there are two types of entity associates. So, we can say that a Binary relationship exists when there are two types of entity and we call them a degree of relationship is 2. Or in other words, in a relation when two entity sets are participating then such type of relationship is known as a binary relationship. This is the most used relationship and one can easily be converted into a relational table.

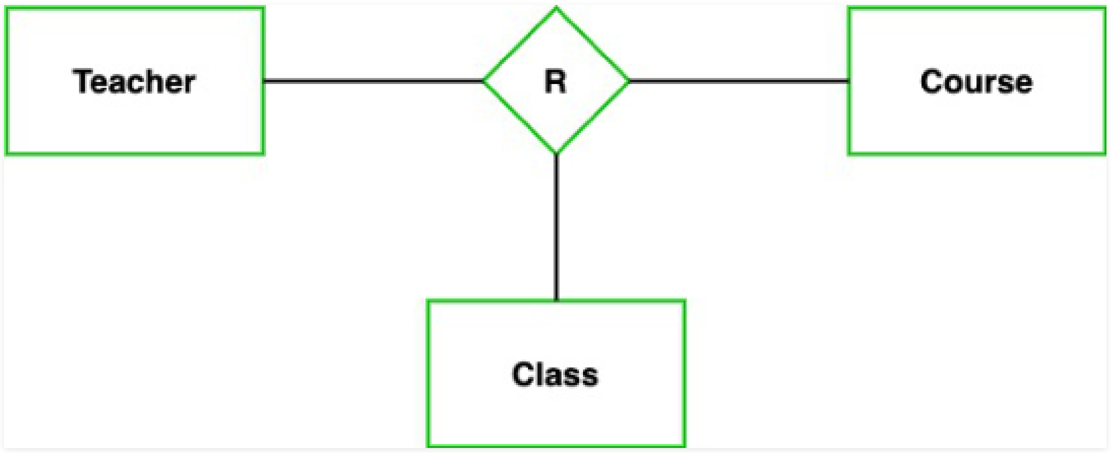
Example: We have two entity types 'Student' and 'ID' where each 'Student' has his 'ID'. So, here two entity types are associating we can say it is a binary relationship. Also, one 'Student' can have many 'daughters' but each 'daughter' should belong to only one 'father. We can say that it is a one-to-many binary relationship.



Ternary (degree 3)

In the Ternary relationship, there are three types of entity associates. So, we can say that a Ternary relationship exists when there are three types of entity and we call them a degree of relationship is 3. Since the number of entities increases due to this, it becomes very complex to turn E-R into a relational table. Now let's understand with the examples.

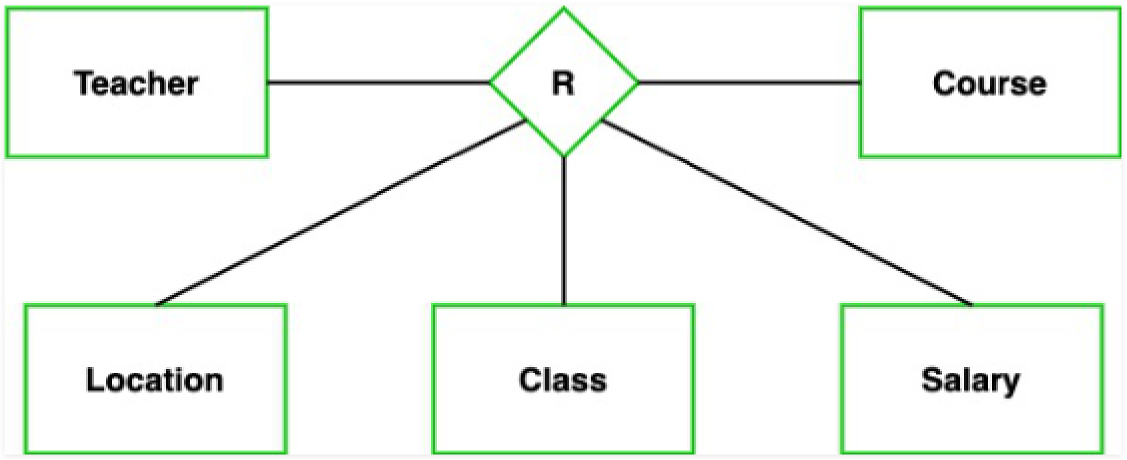
Example: We have three entity types 'Teacher', 'Course', and 'Class'. The relationship between these entities is defined as the teacher teaching a particular course, also the teacher teaches a particular class. So, here three entity types are associating we can say it is a ternary relationship.



N-ary (n degree)

In the N-ary relationship, there are n types of entity that associates. So, we can say that an N-ary relationship exists when there are n types of entities. There is one limitation of the N-ary relationship, as there are many entities so it is very hard to convert into an entity, rational table. So, this is very uncommon, unlike binary which is very much popular.

Example: We have 5 entities Teacher, Class, Location, Salary, Course. So, here five entity types are associating we can say an n-ary relationship is 5.



Domain

A table is DBMS is a set of rows and columns that contain data. Columns in table have a unique name, often referred as **attributes in DBMS**. A domain is a unique set of values permitted for an attribute in a table. For example, a domain of month-of-year can accept January, February....December as possible values, a domain of integers can accept whole numbers that are negative, positive and zero.

Definition: Domain constraints are **user defined data type** and we can define them like this: Domain Constraint = data type + Constraints (NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK / DEFAULT)

Example:
For example I want to create a table "student_info" with "stu_id" field having value greater than 100, I can create a domain and table like this:

```
create domain id_value int
constraint id_test
check(value > 100);

create table student_info (
stu_id id_value PRIMARY KEY,
stu_name varchar(30),
stu_age int
);
```

DDL, DML in DBMS

SQL | DDL, DQL, DML, DCL and TCL Commands

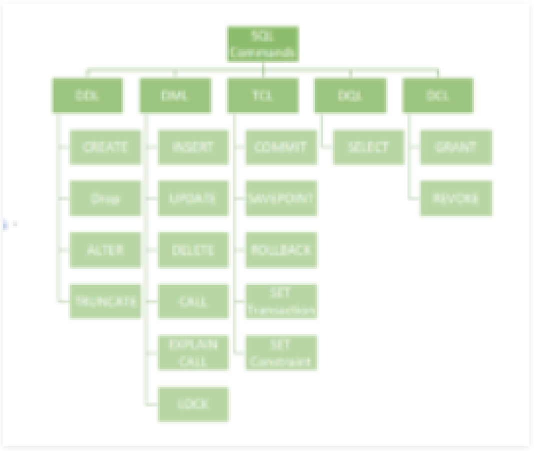
Difficulty Level : Easy • Last Updated : 30 Sep, 2021

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
2. DQl – Data Query Language
3. DML – Data Manipulation Language
4. DCL – Data Control Language

Though many resources claim there to be another category of SQL clauses **TCL – Transaction Control Language**. So we will see in detail about TCL as well.



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.DDL is a set of SQL commands used to create, modify, and delete database structures but not data. These commands are normally not used by a general user, who should be accessing the database via an application.

List of DDL commands:

- **CREATE**: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
- **DROP**: This command is used to delete objects from the database.
- **ALTER:** This is used to alter the structure of the database.
- **TRUNCATE:** This is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT**: This is used to add comments to the data dictionary.
- **RENAME:** This is used to rename an object existing in the database.

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. We can define DQL as follows it is a component of SQL statement that allows getting data from the database and imposing order upon it. It includes the SELECT statement. This command allows getting the data out of the database to perform operations with it. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e. a front-end.

List of DQL:

- **SELECT:** It is used to retrieve data from the database.

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. It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

List of DML commands:

- **INSERT** : It is used to insert data into a table.
- **UPDATE:** It is used to update existing data within a table.
- **DELETE** : It is used to delete records from a database table.
- **LOCK:** Table control concurrency.
- **CALL:** Call a PL/SQL or JAVA subprogram.
- **EXPLAIN PLAN:** It describes the access path to data.

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.

List of DCL commands:

- **GRANT:** This command gives users access privileges to the database.
- **REVOKE:** This command withdraws the user's access privileges given by using the GRANT command.

Though many resources claim there to be another category of SQL clauses TCL – Transaction Control Language. So we will see in detail about TCL as well. TCL commands deal with the transaction within the database.

List of TCL commands:

- **COMMIT:** Commits a Transaction.
- **ROLLBACK:** Rollbacks a transaction in case of any error occurs.
- **SAVEPOINT:** Sets a savepoint within a transaction.
- **SET TRANSACTION:** Specify characteristics for the transaction.

Database Basic Terminologies

Basic Database Terminology

Before we get into the interesting stuff, it might be useful to get acquainted with a few of the terms that you will encounter in your PostgreSQL life. PostgreSQL has a long history?you can trace its history back to 1977 and a program known as Ingres. A lot has changed in the relational database world since 1977. When you are breaking ground with a new product (as the Ingres developers were), you don't have the luxury of using standard, well-understood, and well-accepted terminology?you have to make it up as you go along. Many of the terms used by PostgreSQL have synonyms (or at least close analogies) in today's relational marketplace. In this section, I'll show you a few of the terms that you'll encounter in this book and try to explain how they relate to similar concepts in other database products.

- Database

  A database is a named collection of tables. (see table). A database can also contain views, indexes, sequences, data types, operators, and functions. Other relational database products use the term catalog.

- Command

  A command is a string that you send to the server in hopes of having the server do something useful. Some people use the word statement to mean command. The two words are very similar in meaning and, in practice, are interchangeable.

- Query

  A query is a type of command that retrieves data from the server.

- Table (relation, file, class)

  A table is a collection of rows. A table usually has a name, although some tables are temporary and exist only to carry out a command. All the rows in a table have the same shape (in other words, every row in a table contains the same set of columns). In other database systems, you may see the terms `relation`, `file`, or even `class`?these are all equivalent to a table.

- Column (field, attribute)

  A column is the smallest unit of storage in a relational database. A column represents one piece of information about an object. Every column has a name and a data type. Columns are grouped into rows, and rows are grouped into tables. In Figure 1.1, the shaded area depicts a single column.

- Row (record, tuple)

  A row is a collection of column values. Every row in a table has the same shape (in other words, every row is composed of the same set of columns). If you are trying to model a real-world application, a row represents a real-world object. For example, if you are running an auto dealership, you might have a `vehicles` table. Each row in the `vehicles` table represents a car (or truck, or motorcycle, and so on). The kinds of information that you store are the same for all `vehicles` (that is, every car has a color, a vehicle ID, an engine, and so on). In Figure 1.2, the shaded area depicts a row.

  - View

    A view is an alternative way to present a table (or tables). You might think of a view as a "virtual" table. A view is (usually) defined in terms of one or more tables. When you create a view, you are not storing more data, you are instead creating a different way of looking at existing data. A view is a useful way to give a name to a complex query that you may have to use repeatedly.

  - Client/server

    PostgreSQL is built around a client/server architecture. In a client/server product, there are at least two programs involved. One is a client and the other is a server. These programs may exist on the same host or on different hosts that are connected by some sort of network. The server offers a service; in the case of PostgreSQL, the server offers to store, retrieve, and change data. The client asks a server to perform work; a PostgreSQL client asks a PostgreSQL server to serve up relational data.

  - Client

    A client is an application that makes requests of the PostgreSQL server. Before a client application can talk to a server, it must connect to a postmaster (see `postmaster`) and establish its identity. Client applications provide a user interface and can be written in many languages. Chapters 8 through 17 will show you how to write a client application.

  - Server

    The PostgreSQL server is a program that services commands coming from client applications. The PostgreSQL server has no user interface?you can't talk to the server directly, you must use a client application.

  - Postmaster

    Because PostgreSQL is a client/server database, something has to listen for connection requests coming from a client application. That's what the `postmaster` does. When a connection request arrives, the `postmaster` creates a new server process in the host operating system.

  - Transaction

    A transaction is a collection of database operations that are treated as a unit. PostgreSQL guarantees that all the operations within a transaction complete or that none of them complete. This is an important property?it ensures that if something goes wrong in the middle of a transaction, changes made before the point of failure will not be reflected in the database. A transaction usually starts with a `BEGIN` command and ends with a `COMMIT` or `ROLLBACK` (see the next entries).

  - Commit

    A commit marks the successful end of a transaction. When you perform a commit, you are telling PostgreSQL that you have completed a unit of operation and that all the changes that you made to the database should become permanent.

  - Rollback

    A rollback marks the unsuccessful end of a transaction. When you roll back a transaction, you are telling PostgreSQL to discard any changes that you have made to the database (since the beginning of the transaction).

  - Index

    An index is a data structure that a database uses to reduce the amount of time it takes to perform certain operations. An index can also be used to ensure that duplicate values don't appear where they aren't wanted. I'll talk about indexes in Chapter 4, "Query Optimization."

  - Result set

    When you issue a query to a database, you get back a result set. The result set contains all the rows that satisfy your query. A result set may be empty.

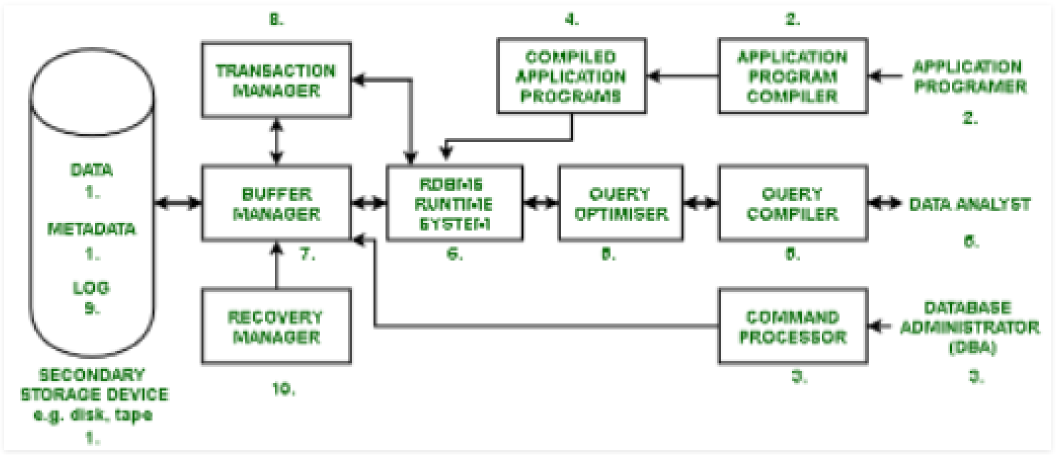
RDBMS Architecture

RDBMS Architecture

Difficulty Level : Expert • Last Updated : 09 Jun, 2020

RDBMS stands for Relational Database Management System and it implements SQL.
In the real-world scenario, people use the Relational Database Management System to collect information and process it, to provide service. E.g. In a ticket processing system, details about us (e.g. age, gender) and our journey (e.g. source, destination), are collected, and the ticket is provided to us.

RDBMS Architecture :



Note –
Each term in the diagram is explained below in the point number associated with the term.

1. All data, data about data (metadata) and logs are stored in the Secondary Storage devices (SSD), such as Disks and Tapes.The programs that are used to do the day-to-day tasks of an enterprise are called Application programs. These programs provide the functionality for the day-to-day operations of the enterprise. They are written in high level languages (HLL) like Java, C etc, which along with the SQL, are used to communicate with the databases.
2. RDBMS has a compiler that converts the SQL commands to lower level language, processes it and stores it into the secondary storage device.
3. It is the job of Database Administrator (DBA) to set up the structure of the database using command processor. The DDL stands for Data Definition Language and is used by the DBA to create or drop tables, add columns etc. The DBA also uses other commands which are used to set constraints and access controls.
4. Application Programmers compile the applications using a compiler and create executable files (compiled application programs) and then store the data on the secondary storage device.
5. Job of Data Analyst is to use the Query Compiler and Query Optimizer (uses relational properties for executing queries) to manipulate the data in the database.

6. RDBMS Run Time System executes the compiled queries and application programs and also interacts with the transaction manager and buffer manager.
7. Buffer Manager temporarily stores the data of the database in the main memory and uses paging algorithm so that operations can be performed faster and the disk space can be managed.
8. Transaction Manager deals with the principle of either completely doing a task or not doing it at all (Atomicity property). E.g. Suppose a person named Geeks, wants to send money to his sister. He sends the money and system crashes in between. In no case should it happen that he has sent money but his sister has not received it. This is handled by the transaction manager. The transaction manager would either refund the money to Geeks or transfer it to his sister.
9. Log is a system, which records the information about all the transactions, so that whenever a system failure (disk failure, system shut down due to no power etc.) arises, the partial transactions can be undone.
10. Recovery Manager takes control of the system so that it reaches a steady state after failure. The Recovery Manager takes into account the log files and undoes the partial transactions and reflects the complete transaction in the database.

Relational Schema

Relation Schema in DBMS

Difficulty Level : Basic • Last Updated : 06 Aug, 2021

Relation schema defines the design and structure of the relation like it consists of the relation name, set of attributes/field names/column names. every attribute would have an associated domain.

There is a student named Geeks, she is pursuing B.Tech, in the 4th year, and belongs to IT department (department no. 1) and has roll number 1601347 She is proctored by Mrs. S Mohanty. If we want to represent this using databases we would have to create a student table with name, sex, degree, year, department, department number, roll number and proctor (adviser) as the attributes.

```
student (rollNo, name, degree, year, sex, deptNo, advisor)
```

Note –
If we create a database, details of other students can also be recorded.

Similarly, we have the IT Department, with department Id 1, having Mrs. Sujata Chakravarty as the head of department. And we can call the department on the number 0657 228662 .

This and other departments can be represented by the department table, having department ID, name, hod and phone as attributes.

```
department (deptId, name, hod, phone)
```

The course that a student has selected has a courseid, course name, credit and department number.

```
course (coursId, ename, credits, deptNo)
```

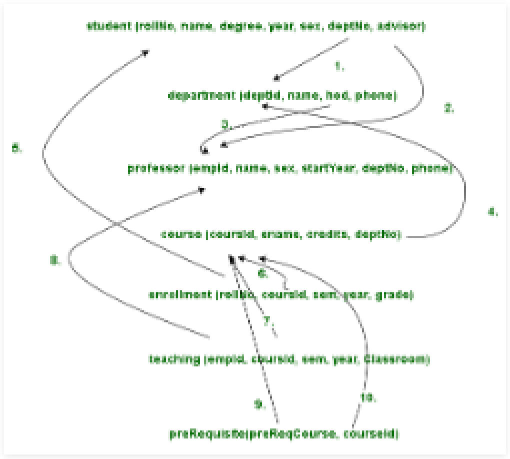
The professor would have an employee Id, name, sex, department no. and phone number.

```
professor (empId, name, sex, startYear, deptNo, phone)
```

We can have another table named enrollment, which has roll no, courseId, semester, year and grade as the attributes.

```
enrollment (rollNo, coursId, sem, year, grade)
```

The relations between them is represented through arrows in the following **Relation diagram,**


student (rollNo, name, degree, year, sex, deptNo, advisor)
1.
department (deptId, name, hod, phone)
2.
3.
5.
professor (empId, name, sex, startYear, deptNo, phone)
4.
course (courseId, cname, credits, deptNo)
8.
6.
enrollment (rollNo, courseId, sem, year, grade)
7.
teaching (empId, courseId, sem, year, Classroom)
9.
10.
preRequisite(preReqCourse, courseId)

1. This represents that the deptNo in student table table is same as deptId used in department table. deptNo in student table is a foreign key. It refers to deptId in department table.
2. This represents that the advisor in student table is a foreign key. It refers to empId in professor table.
3. This represents that the hod in department table is a foreign key. It refers to empId in professor table.
4. This represents that the deptNo in course table table is same as deptId used in department table. deptNo in student table is a foreign key. It refers to deptId in department table.
5. This represents that the rollNo in enrollment table is same as rollNo used in student table.
6. This represents that the courseId in enrollment table is same as courseId used in course table.
7. This represents that the courseId in teaching table is same as courseId used in course table.
8. This represents that the empId in teaching table is same as empId used in professor table.
9. This represents that preReqCourse in prerequisite table is a foreign key. It refers to courseId in course table.
10. This represents that the deptNo in student table is same as deptId used in department table.

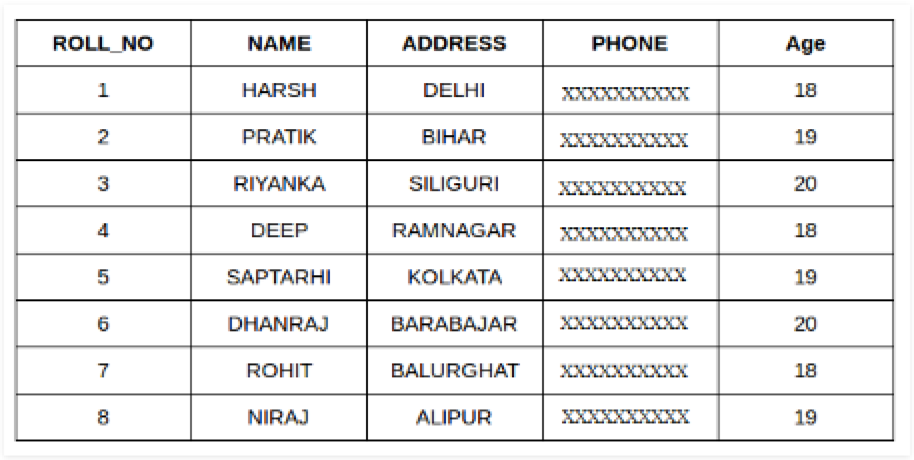
SQL joining

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

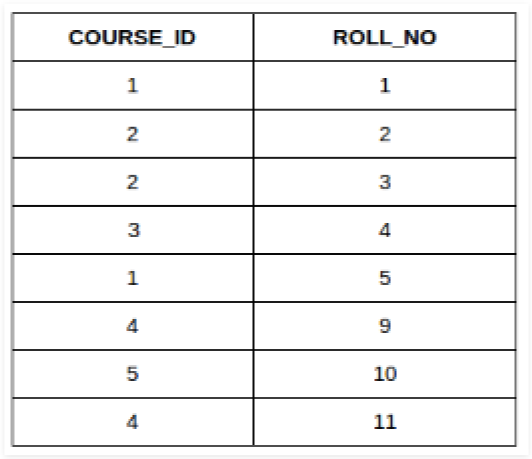
Consider the two tables below:

Student



ROLL_NO	NAME	ADDRESS	PHONE	Age
1	HARSH	DELHI	XXXXXXXXXX	18
2	PRATIK	BIHAR	XXXXXXXXXX	19
3	RIYANKA	SILIGURI	XXXXXXXXXX	20
4	DEEP	RAMNAGAR	XXXXXXXXXX	18
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19
6	DHANRAJ	BARABAJAR	XXXXXXXXXX	20
7	ROHIT	BALURGHAT	XXXXXXXXXX	18
8	NIRAJ	ALIPUR	XXXXXXXXXX	19

StudentCourse



COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

A. INNER JOIN

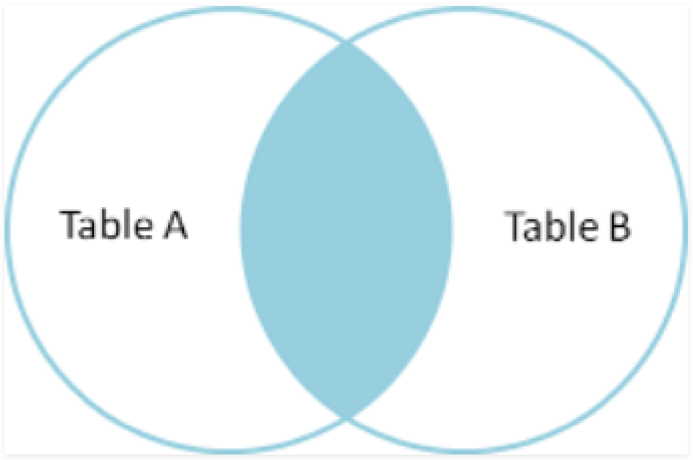
The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;


table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```

***Note**: We can also write JOIN instead of INNER JOIN. JOIN is same as INNER JOIN.*

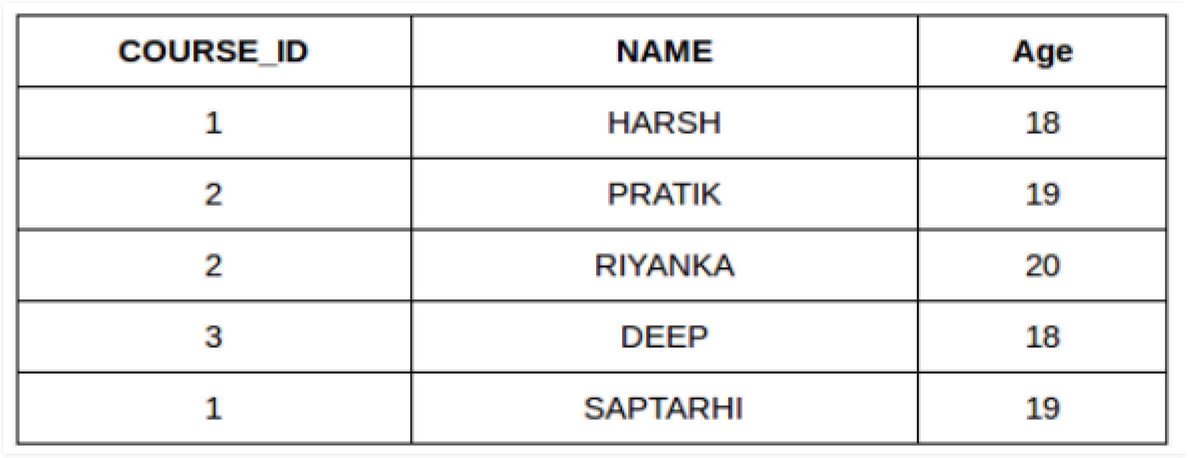


Example Queries(INNER JOIN)

This query will show the names and age of students enrolled in different courses.

```
SELECT StudentCourse.COURSE_ID, Student.NAME, Student.AGE FROM Student
INNER JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

Output:



COURSE_ID	NAME	Age
1	HARSH	18
2	PRATIK	19
2	RIYANKA	20
3	DEEP	18
1	SAPTARHI	19

B. LEFT JOIN

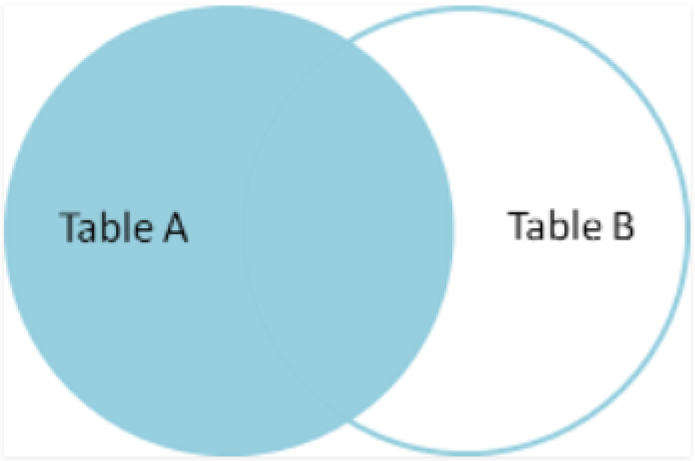
This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

```
table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```

Note*: We can also use LEFT OUTER JOIN instead of LEFT JOIN, both are the same.*



Example Queries(LEFT JOIN):

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output:

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1

C. RIGHT JOIN

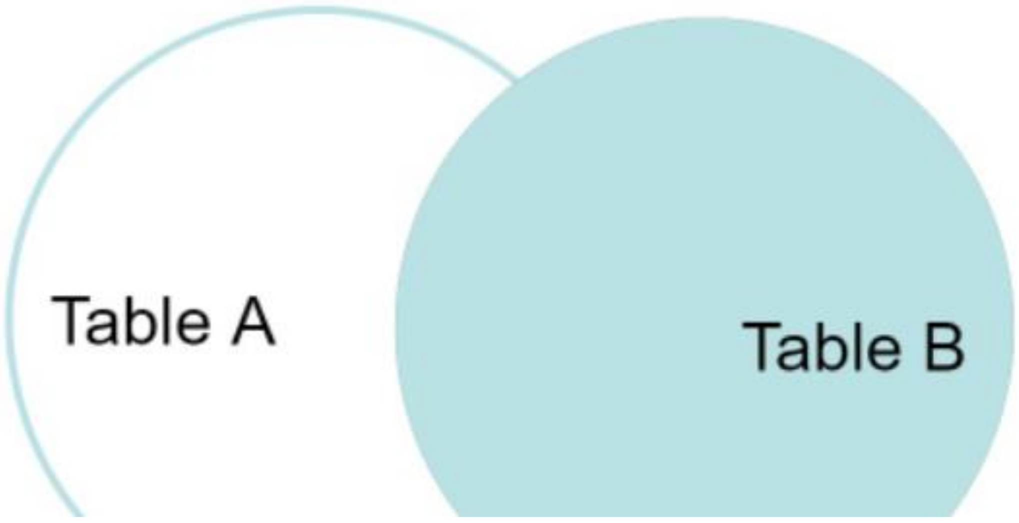
RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;


table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```

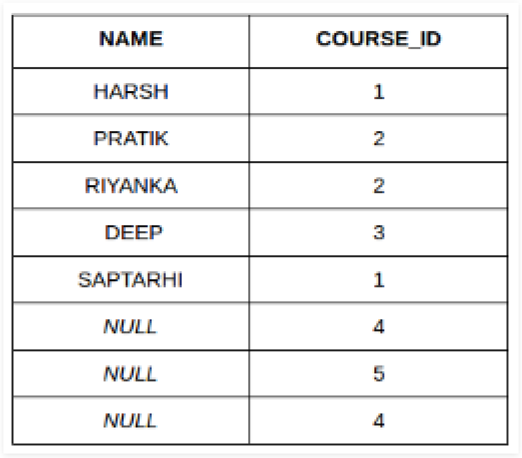
Note: *We can also use RIGHT OUTER JOIN instead of RIGHT JOIN, both are the same.*



Example Queries(RIGHT JOIN):

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

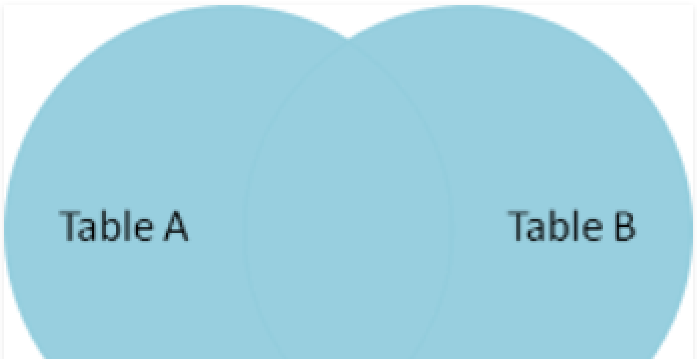
Output:



NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
NULL	4
NULL	5
NULL	4

D. FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.



Syntax:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;


table1: First table.
table2: Second table
matching_column: Column common to both the tables.
```

Example Queries(FULL JOIN):

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
FULL JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output:

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1

Creating and Altering tables

SQL - CREATE Table

Previous Page

Next Page

Creating a basic table involves naming the table and defining its columns and each column's data type.

The SQL **CREATE TABLE** statement is used to create a new table.

Syntax

The basic syntax of the CREATE TABLE statement is as follows −

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

CREATE TABLE is the keyword telling the database system what you want to do. In this case, you want to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement.

Then in brackets comes the list defining each column in the table and what sort of data type it is. The syntax becomes clearer with the following example.

A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement. You can check the complete details at Create Table Using another Table.

Example

The following code block is an example, which creates a CUSTOMERS table with an ID as a primary key and NOT NULL are the constraints showing that these fields cannot be NULL while creating records in this table −

```
SQL> CREATE TABLE CUSTOMERS(
   ID   INT              NOT NULL,
   NAME VARCHAR (20)     NOT NULL,
   AGE  INT              NOT NULL,
   ADDRESS  CHAR (25) ,
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

You can verify if your table has been created successfully by looking at the message displayed by the SQL server, otherwise you can use the **DESC** command as follows −

```
SQL> DESC CUSTOMERS;
+---------+---------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| ID | int(11) | NO | PRI | | |
| NAME | varchar(20) | NO | | | |
| AGE | int(11) | NO | | | |
| ADDRESS | char(25) | YES | | NULL | |
| SALARY | decimal(18,2) | YES | | NULL | |
+---------+---------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Now, you have CUSTOMERS table available in your database which you can use to store the required information related to customers.

SQL - ALTER TABLE Command

Previous Page Next Page

The SQL **ALTER TABLE** command is used to add, delete or modify columns in an existing table. You should use the ALTER TABLE command to add and drop various constraints on an existing table.

Syntax

The basic syntax of an ALTER TABLE command to add a **New Column** in an existing table is as follows.

```
ALTER TABLE table_name ADD column_name datatype;
```

The basic syntax of an ALTER TABLE command to **DROP COLUMN** in an existing table is as follows.

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

The basic syntax of an ALTER TABLE command to change the **DATA TYPE** of a column in a table is as follow

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

The basic syntax of an ALTER TABLE command to add a **NOT NULL** constraint to a column in a table i follows.

```
ALTER TABLE table_name MODIFY column_name datatype NOT NULL;
```

The basic syntax of ALTER TABLE to **ADD UNIQUE CONSTRAINT** to a table is as follows.

```
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);
```

The basic syntax of an ALTER TABLE command to **ADD CHECK CONSTRAINT** to a table is as follows.

```
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);
```

The basic syntax of an ALTER TABLE command to **ADD PRIMARY KEY** constraint to a table is as follows.

```
ALTER TABLE table_name
ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);
```

The basic syntax of an ALTER TABLE command to **DROP CONSTRAINT** from a table is as follows.

If you're using MySQL, the code is as follows –

```
ALTER TABLE table_name
DROP INDEX MyUniqueConstraint;
```

The basic syntax of an ALTER TABLE command to **DROP PRIMARY KEY** constraint from a table is as follows.

```
ALTER TABLE table_name
DROP CONSTRAINT MyPrimaryKey;
```

If you're using MySQL, the code is as follows –

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

Example

Consider the CUSTOMERS table having the following records –

```
+----+----------+-----+-----------+----------+
| 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 |
+----+----------+-----+-----------+----------+
```

Following is the example to ADD a **New Column** to an existing table –

```
ALTER TABLE CUSTOMERS ADD SEX char(1);
```

Now, the CUSTOMERS table is changed and following would be output from the SELECT statement.

```
+----+---------+-----+-----------+----------+------+
| ID | NAME | AGE | ADDRESS | SALARY | SEX |
+----+---------+-----+-----------+----------+------+
| 1 | Ramesh | 32 | Ahmedabad | 2000.00 | NULL |
| 2 | Ramesh | 25 | Delhi | 1500.00 | NULL |
| 3 | kaushik | 23 | Kota | 2000.00 | NULL |
| 4 | kaushik | 25 | Mumbai | 6500.00 | NULL |
| 5 | Hardik | 27 | Bhopal | 8500.00 | NULL |
| 6 | Komal | 22 | MP | 4500.00 | NULL |
| 7 | Muffy | 24 | Indore | 10000.00 | NULL |
+----+---------+-----+-----------+----------+------+
```

Following is the example to DROP sex column from the existing table.

```
ALTER TABLE CUSTOMERS DROP SEX;
```

Now, the CUSTOMERS table is changed and following would be the output from the SELECT statement.

```
+----+---------+-----+----------+---------+
| ID | NAME | AGE | ADDRESS | SALARY |
+----+---------+-----+----------+---------+
| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |
| 2 | Ramesh | 25 | Delhi | 1500.00 |
| 3 | kaushik | 23 | Kota | 2000.00 |
| 4 | kaushik | 25 | Mumbai | 6500.00 |
| 5 | Hardik | 27 | Bhopal | 8500.00 |
| 6 | Komal | 22 | MP | 4500.00 |
| 7 | Muffy | 24 | Indore | 10000.00 |
+----+---------+-----+----------+---------+
```

Selection and Projection

Difference between Selection and Projection in DBMS

Last Updated : 12 Jun, 2020

Prerequisite – Relational Algebra

1. Selection **:**

This operation chooses the subset of tuples from the relation that satisfies the given condition mentioned in the syntax of selection.

Notation –

```
σc (R)
```

Here, 'c' is selection condition and 'σ (sigma)' is used to denote **Select Operator**.

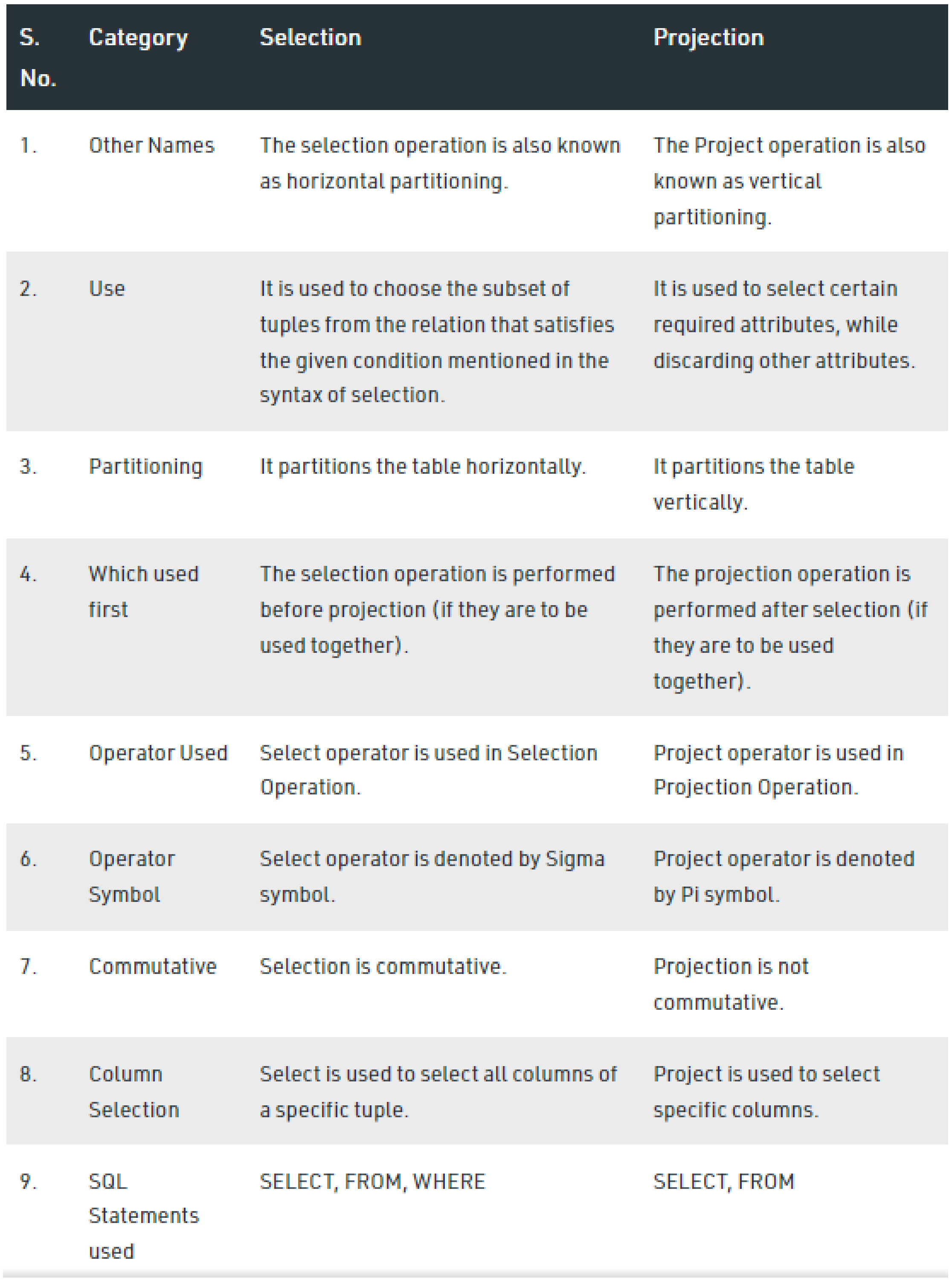
2. Projection **:**

This operation selects certain required attributes, while discarding other attributes.

Notation –

```
πA (R)
```

where 'A' is the attribute list, it is the desired set of attributes from the attributes of relation(R),
symbol 'π(pi)' is used to denote the Project operator,
R is generally a relational algebra expression, which results in a relation.



S. No.	Category	Selection	Projection
1.	Other Names	The selection operation is also known as horizontal partitioning.	The Project operation is also known as vertical partitioning.
2.	Use	It is used to choose the subset of tuples from the relation that satisfies the given condition mentioned in the syntax of selection.	It is used to select certain required attributes, while discarding other attributes.
3.	Partitioning	It partitions the table horizontally.	It partitions the table vertically.
4.	Which used first	The selection operation is performed before projection (if they are to be used together).	The projection operation is performed after selection (if they are to be used together).
5.	Operator Used	Select operator is used in Selection Operation.	Project operator is used in Projection Operation.
6.	Operator Symbol	Select operator is denoted by Sigma symbol.	Project operator is denoted by Pi symbol.
7.	Commutative	Selection is commutative.	Projection is not commutative.
8.	Column Selection	Select is used to select all columns of a specific tuple.	Project is used to select specific columns.
9.	SQL Statements used	SELECT, FROM, WHERE	SELECT, FROM

Single Row Functions

Single row functions

Single row functions can be character functions, numeric functions, date functions, and conversion functions. Note that these functions are used to manipulate data items. These functions require one or more input arguments and operate on each row, thereby returning one output value for each row. Argument can be a column, literal or an expression. Single row functions can be used in SELECT statement, WHERE and ORDER BY clause. Single row functions can be -

- **General functions** - Usually contains NULL handling functions. The functions under the category are NVL, NVL2, NULLIF, COALESCE, CASE, DECODE.
- **Case Conversion functions** - Accepts character input and returns a character value. Functions under the category are UPPER, LOWER and INITCAP.
    - UPPER function converts a string to upper case.
    - LOWER function converts a string to lower case.
    - INITCAP function converts only the initial alphabets of a string to upper case.
- **Character functions** - Accepts character input and returns number or character value. Functions under the category are CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE.
    - CONCAT function concatenates two string values.
    - LENGTH function returns the length of the input string.
    - SUBSTR function returns a portion of a string from a given start point to an end point.
    - INSTR function returns numeric position of a character or a string in a given string.
    - LPAD and RPAD functions pad the given string upto a specific length with a given character.
    - TRIM function trims the string input from the start or end.
    - REPLACE function replaces characters from the input string with a given character.
- **Date functions** - Date arithmetic operations return date or numeric values. Functions under the category are MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND and TRUNC.
    - MONTHS_BETWEEN function returns the count of months between the two dates.
    - ADD_MONTHS function add 'n' number of months to an input date.
    - NEXT_DAY function returns the next day of the date specified.
    - LAST_DAY function returns last day of the month of the input date.
    - ROUND and TRUNC functions are used to round and truncates the date value.
- **Number functions** - Accepts numeric input and returns numeric values. Functions under the category are ROUND, TRUNC, and MOD.
    - ROUND and TRUNC functions are used to round and truncate the number value.
    - MOD is used to return the remainder of the division operation between two numbers.

Creating users, Granting Permissions, Creating Roles in SQL

Creating a New User

Upon installation, MySQL creates a **root** user account which you can use to manage your database. This user has full privileges over the MySQL server, meaning it has complete control over every database, table, user, and so on. Because of this, it's best to avoid using this account outside of administrative functions. This step outlines how to use the **root** MySQL user to create a new user account and grant it privileges.

In Ubuntu systems running MySQL `5.7` (and later versions), the **root** MySQL user is set to authenticate using the `auth_socket` plugin by default rather than with a password. This plugin requires that the name of the operating system user that invokes the MySQL client matches the name of the MySQL user specified in the command. This means that you need to precede the `mysql` command with `sudo` to invoke it with the privileges of the **root** Ubuntu user in order to gain access to the **root** MySQL user:

```
$ sudo mysql
```

Note: If your **root** MySQL user is configured to authenticate with a password, you will need to use a different command to access the MySQL shell. The following will run your MySQL client with regular user privileges, and you will only gain administrator privileges within the database by authenticating with the correct password:

```
$ mysql -u root -p
```

Once you have access to the MySQL prompt, you can create a new user with a `CREATE USER` statement. These follow this general syntax:

```
mysql> CREATE USER 'username'@'host' IDENTIFIED WITH authentication_plugin BY 'passw
```

After `CREATE USER`, you specify a username. This is immediately followed by an `@` sign and then the hostname from which this user will connect. If you only plan to access this user locally from your Ubuntu server, you can specify `localhost`. Wrapping both the username and host in single quotes isn't always necessary, but doing so can help to prevent errors.

You have several options when it comes to choosing your user's authentication plugin. The `auth_socket` plugin mentioned previously can be convenient, as it provides strong security without requiring valid users to enter a password to access the database. But it also prevents remote connections, which can complicate things when external programs need to interact with MySQL.

As an alternative, you can leave out the `WITH authentication_plugin` portion of the syntax entirely to have the user authenticate with MySQL's default plugin, `caching_sha2_password`. The MySQL documentation recommends this plugin for users who want to log in with a password due to its strong security features.

Run the following command to create a user that authenticates with `caching_sha2_password`. Be sure to change `sammy` to your preferred username and `password` to a strong password of your choosing:

```
mysql> CREATE USER 'sammy'@'localhost' IDENTIFIED BY 'password';
```

Note: There is a known issue with some versions of PHP that causes problems with `caching_sha2_password`. If you plan to use this database with a PHP application — phpMyAdmin, for example — you may want to create a user that will authenticate with the older, though still secure, `mysql_native_password` plugin instead:

```
mysql> CREATE USER 'sammy'@'localhost' IDENTIFIED WITH mysql_native_password BY '
```

If you aren't sure, you can always create a user that authenticates with `caching_sha2_plugin` and then `ALTER` it later on with this command:

```
mysql> ALTER USER 'sammy'@'localhost' IDENTIFIED WITH mysql_native_password BY '
```

After creating your new user, you can grant them the appropriate privileges.

Granting a User Permissions

The general syntax for granting user privileges is as follows:

```
mysql> GRANT PRIVILEGE ON database.table TO 'username'@'host';
```

The `PRIVILEGE` value in this example syntax defines what actions the user is allowed to perform on the specified `database` and `table`. You can grant multiple privileges to the same user in one command by separating each with a comma. You can also grant a user privileges globally by entering asterisks (`*`) in place of the database and table names. In SQL, asterisks are special characters used to represent "all" databases or tables.

To illustrate, the following command grants a user global privileges to `CREATE`, `ALTER`, and `DROP` databases, tables, and users, as well as the power to `INSERT`, `UPDATE`, and `DELETE` data from any table on the server. It also grants the user the ability to query data with `SELECT`, create foreign keys with the `REFERENCES` keyword, and perform `FLUSH` operations with the `RELOAD` privilege. However, you should only grant users the permissions they need, so feel free to adjust your own user's privileges as necessary.

You can find the full list of available privileges in the official MySQL documentation.

Run this `GRANT` statement, replacing `sammy` with your own MySQL user's name, to grant these privileges to your user:

```
mysql> GRANT CREATE, ALTER, DROP, INSERT, UPDATE, DELETE, SELECT, REFERENCES, RELOAD
```

Note that this statement also includes `WITH GRANT OPTION`. This will allow your MySQL user to grant any permissions that it has to other users on the system.

Warning: Some users may want to grant their MySQL user the `ALL PRIVILEGES` privilege, which will provide them with broad superuser privileges akin to the **root** user's privileges, like so:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'sammy'@'localhost' WITH GRANT OPTION;
```

Such broad privileges **should not be granted lightly**, as anyone with access to this MySQL user will have complete control over every database on the server.

Many guides suggest running the `FLUSH PRIVILEGES` command immediately after a `CREATE USER` or `GRANT` statement in order to reload the grant tables to ensure that the new privileges are put into effect:

```
mysql> FLUSH PRIVILEGES;
```

However, according to the official MySQL documentation, when you modify the grant tables indirectly with an account management statement like `GRANT`, the database will reload the grant tables immediately into memory, meaning that the `FLUSH PRIVILEGES` command isn't necessary in our case. On the other hand, running it won't have any negative effect on the system.

If you need to revoke a permission, the structure is almost identical to granting it:

```
mysql> REVOKE type_of_permission ON database_name.table_name FROM 'username'@'host'
```

Note that when revoking permissions, the syntax requires that you use `FROM`, instead of `TO` which you used when granting the permissions.

You can review a user's current permissions by running the `SHOW GRANTS` command:

```
mysql> SHOW GRANTS FOR 'username'@'host';
```

Just as you can delete databases with `DROP`, you can use `DROP` to delete a user:

```
mysql> DROP USER 'username'@'localhost';
```

After creating your MySQL user and granting them privileges, you can exit the MySQL client:

```
mysql> exit
```

In the future, to log in as your new MySQL user, you'd use a command like the following:

```
$ mysql -u sammy -p
```

The `-p` flag will cause the MySQL client to prompt you for your MySQL user's password in order to authenticate.

SQL | Creating Roles

Difficulty Level : **Medium** • Last Updated : 27 Sep, 2018

A role is created to ease setup and maintenance of the security model. It is a named group of related privileges that can be granted to the user. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles:

- You can grant or revoke privileges to users, thereby automatically granting or revoking privileges.
- You can either create Roles or use the system roles pre-defined.

Some of the privileges granted to the system roles are as given below:

System Roles	Privileges granted to the Role
Connect	Create table, Create view, Create synonym, Create sequence, Create session etc.
Resource	Create Procedure, Create Sequence, Create Table, Create Trigger etc. The primary usage of the Resource role is to restrict access to database objects.
DBA	All system privileges

Creating and Assigning a Role –

First, the (Database Administrator)DBA must create the role. Then the DBA can assign privileges to the role and users to the role.

Syntax –

```
CREATE ROLE manager;
Role created.
```

In the syntax:
'manager' is the name of the role to be created.

- Now that the role is created, the DBA can use the GRANT statement to assign users to the role as well as assign privileges to the role.
- It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user.
- If a role is identified by a password, then GRANT or REVOKE privileges have to be identified by the password.

Grant privileges to a role –

```
GRANT create table, create view
TO manager;
Grant succeeded.
```

Grant a role to users

```
GRANT manager TO SAM, STARK;
Grant succeeded.
```

Revoke privilege from a Role :

```
REVOKE create table FROM manager;
```

Drop a Role :

```
DROP ROLE manager;
```

Explanation –
Firstly it creates a manager role and then allows managers to create tables and views. It then grants Sam and Stark the role of managers. Now Sam and Stark can create tables and views. If users have multiple roles granted to them, they receive all of the privileges associated with all of the roles. Then create table privilege is removed from role 'manager' using Revoke.The role is dropped from the database using drop.

Adding and Removing Constraints

Adding and removing constraints

Last Updated: 2021-04-14

Constraints can be added to a new table or to an existing table. To add a unique or primary key, a referential constraint, or a check constraint, use the CREATE TABLE or the ALTER TABLE statement. To remove a constraint, use the ALTER TABLE statement.

For example, add a primary key to an existing table using the ALTER TABLE statement:

```
ALTER TABLE CORPDATA.DEPARTMENT
  ADD PRIMARY KEY (DEPTNO)
```

To make this key a unique key, replace the keyword PRIMARY with UNIQUE.

You can remove a constraint using the same ALTER TABLE statement:

```
ALTER TABLE CORPDATA.DEPARTMENT
  DROP PRIMARY KEY (DEPTNO)
```