RDBMS: Relational Data Base Management System

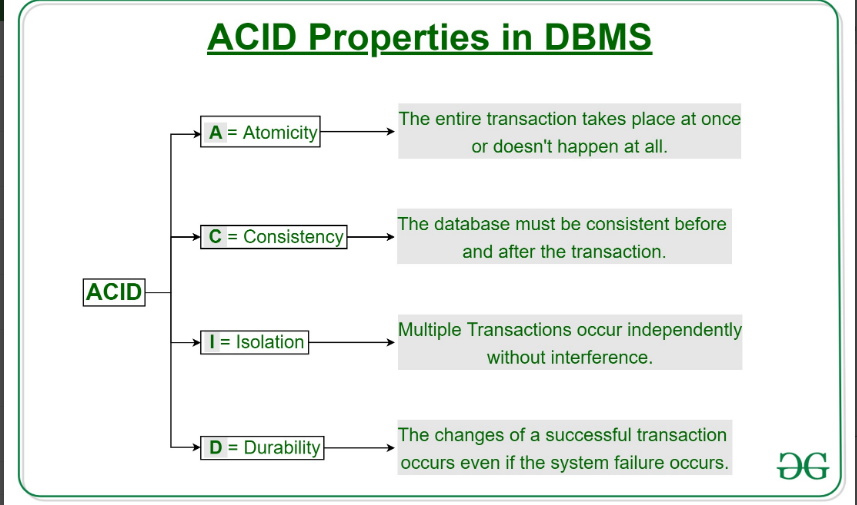
ACID Properties:

Automacity :

Consistency:

Isolation:

Durability:



How to see the database;

Show databases;

How to create data base;

Create database databasename;

If we want to save the data in RDBMS we have to create table;

MySQL uses many different data types broken into three categories −

* Numeric
* Date and Time
* String Types.

Let us now discuss them in detail.

Numeric Data Types

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you.

The following list shows the common numeric data types and their descriptions −

* **INT** − A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
* **TINYINT** − A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
* **SMALLINT** − A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.
* **MEDIUMINT** − A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
* **BIGINT** − A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
* **FLOAT(M,D)** − A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
* **DOUBLE(M,D)** − A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.
* **DECIMAL(M,D)** − An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types

The MySQL date and time datatypes are as follows −

* **DATE** − A date in YYYY-MM-DD format, between “1000-01-01” and 9999-12-31. For example, December 30th, 1973 would be stored as “1973-12-30”.
* **DATETIME** − A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
* **TIMESTAMP** − A timestamp between midnight, January 1st, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 ( YYYYMMDDHHMMSS ).
* **TIME** − Stores the time in a HH:MM:SS format.
* **YEAR(M)** − Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

String Types

Although the numeric and date types are fun, most data you'll store will be in a string format. This list describes the common string datatypes in MySQL.

* **CHAR(M)** − A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
* **VARCHAR(M)** − A variable-length string between 1 and 255 characters in length. For example, VARCHAR(25). You must define a length when creating a VARCHAR field.
* **BLOB or TEXT** − A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data. The difference between the two is that the sorts and comparisons on the stored data are **case sensitive** on BLOBs and are **not case sensitive** in TEXT fields. You do not specify a length with BLOB or TEXT.
* **TINYBLOB or TINYTEXT** − A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
* **MEDIUMBLOB or MEDIUMTEXT** − A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
* **LONGBLOB or LONGTEXT** − A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
* **ENUM** − An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

How to connect database;

* Use database\_name;

How to see the tables in database?

* Show tables;

Why we have to create a table?

To store the data in any RDBMS(MYSQL, Oracle, Postgresql etc) we will create a table.

Table contains the columns.

How to create a table?

Student

(s\_id,s\_name,s\_gender,s\_dob, s\_grade,s\_father\_name,s\_school,s\_joining\_date)

Create table student

(s\_id varchar(20) ,

s\_name varchar(20),

s\_gender char,

s\_dob date,

s\_grade varchar(10),

s\_father\_name varchar(20),

s\_school varchar(50)

,s\_joining\_date date);

How to see the columns from a particular table?

* Show columns from table\_name;

How to insert the values in to the table?

Insert into table\_name (column\_names) values (values);

Ex for Student table:

* Insert into student (s\_id, s\_name, s\_gender, s\_dob) values (‘1001’, ‘Naresh’, ‘M’, ‘1992-04-17’);
* Insert into student values (‘1001’, ‘Naresh’, ‘M’, ‘1992-04-17’);

How to see the table data?

* Select \* from student;

Employee ->

emp\_id, emp\_name,emp\_salary, emp\_designation, emp\_gender,emp\_doj

How to delete a record or row from a table?

Delete from table\_name where field=value;

Ex: employee table

Delete from rroot

employee where employee\_id = ‘103’;

Update Table?

Update table\_name set column\_name=value, column\_name=value where column\_name=value;

Update employee set empSalary=’15000’, designation=’SSE’ where empId=’101’;

How to delete table?

Drop table table\_name;

Drop table student;

How to add primary key while creating table?

Create table student

(s\_id varchar(20) primary key,

s\_name varchar(20) not null,

s\_gender char not null,

s\_dob date,

s\_grade varchar(10) not null,

s\_father\_name varchar(20),

s\_school varchar(50) not null,

s\_joining\_date date);

What is Primary Key?

It is unique value in table, it is used to identify the particular person or object.

18-12-2021

Not Null

Not null is key word it won’t allow the null values into the columns.

Unique

DDL vs DML

Alter Table

If want add extra columns in the table or delete the columns in table or rename columns in the table we can use alter key word.

If we want the rename the table name also we can use alter key word.

* + To add the columns

alter table student add column(s\_gender char not null, s\_grade varchar(20) not null, s\_school varchar(50) not null);

to rename the column

alter table student change column s\_school s\_school\_name varchar(20);

to rename the table name

alter table student rename student\_details;

to drop or delete columns in table

alter table student drop column(column names);

Distinct

Count

19-12-2021

Where clause

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Equal | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_equal_to) |
| > | Greater than | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_greater_than) |
| < | Less than | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_less_than) |
| >= | Greater than or equal | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_greater_than2) |
| <= | Less than or equal | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_less_than2) |
| <> | Not equal. **Note:** In some versions of SQL this operator may be written as != | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_not_equal_to) |
| BETWEEN | Between a certain range (SELECT \* FROM employee  WHERE empId BETWEEN 50 AND 60); | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_between) |
| LIKE | Search for a pattern (SELECT \* FROM employee  WHERE name LIKE 's%'); | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_like) |
| IN | To specify multiple possible values for a column (\*SELECT \* FROM Customers  WHERE City IN ('Paris','London')); |  |

## The SQL AND, OR and NOT Operators

### AND Syntax

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 AND condition2 AND condition3 ...;

SELECT \* FROM employee  
WHERE Country='Germany' AND City='Berlin';

### OR Syntax

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

SELECT \* FROM employee  
WHERE City='Berlin' OR City='München';

### NOT Syntax

SELECT column1, column2, ...  
FROM table\_name  
WHERE NOT condition;

SELECT \* FROM employee  
WHERE NOT Country='Germany';

## Combining AND, OR and NOT

You can also combine the AND, OR and NOT operators.

The following SQL statement selects all fields from "Customers" where country is "Germany" AND city must be "Berlin" OR "München" (use parenthesis to form complex expressions):

### Example

SELECT \* FROM employee  
WHERE Country='Germany' AND (City='Berlin' OR City='München');

select \* from employee where empLocation="Hyderabad" and (empAddress="Guntur" or empAdress="Karimnagar");

The following SQL statement selects all fields from "Customers" where country is NOT "Germany" and NOT "USA":

### Example

SELECT \* FROM employee  
WHERE NOT Country='Germany' AND NOT Country='USA';

True and (True or false)

# SQL ORDER BY Keyword

## The SQL ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

### ORDER BY Syntax

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

## ORDER BY Example

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" column:

### Example

SELECT \* FROM employee  
ORDER BY empName;

SELECT \* FROM Customers  
ORDER BY Country DESC;

## ORDER BY Several Columns Example

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column. This means that it orders by Country, but if some rows have the same Country, it orders them by CustomerName:

### Example

SELECT \* FROM Customers  
ORDER BY Country, CustomerName;

SELECT \* FROM Customers  
ORDER BY Country ASC, CustomerName DESC;

## The SQL GROUP BY Statement

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

### GROUP BY Syntax

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)ORDER BY column\_name(s);

SELECT \* FROM employee group by empName;

In keyword

if we want to update or delete or select multiple records at a time we can use the in keyword;

update employee set empSalary=122000 where empId in ('IBM01', 'IBM02', 'IBM03');

!= (not equal to)

If we want to update or delete or select other than mentioned value we can use != operator.

30-1-2022

========

Is null

If we want to add where condition where record is null then we can use is null keyword.

Is not null

If we want to add where condition where record is not null then we can use is not null keyword.

Unique(for column)(DDL)

If want allow unique values for a particular column in the table we can use unique keyword.

alter table javabatch3.employee add column systemId varchar(15) unique;

create table employee(empID varchar(10) primary key, empName varchar(25), doj date, empSalary integer, empGender char, isActive boolean, system\_id varchar(15) unique);

not null(for column)(DDL)

we can apply not null for any column.

If we apply not null for any column we should give values for particular column.

Ex: alter table javabatch3.employee add column city varchar(15) not null;

Ex: alter table javabatch3.employee modify column city varchar(20) not null;

Ex: create table employee(empID varchar(10) primary key, empName varchar(25), doj date, empSalary integer, empGender char, isActive boolean, system\_id varchar(15) unique, city varchar(10) not null);

Auto\_Increment

If we want increase the primary key value automatically then we can use auto\_increment.

create table student (std\_id integer primary key auto\_increment, std\_name varchar(20));

alter table student modify std\_id auto\_increment;

count() aggregate function

if we want do get any records count we can use count function.

select sum(empSalary) as SalarySum, city from javabatch3.employee

Sum() aggregate function

if we want do get any records sum we can use sum function.

select sum(empSalary) as SalarySum from javabatch3.employee

Avg() aggregate function

if we want do get any records average we can use avg function

select avg(empSalary) as averageSalary from javabatch3.employee

concat() aggregate function

if we want combine 2 or more records we can use concat function.

select concat(empName," ", empId) as fullname from javabatch3.employee

max() aggregate function

if we want get minimum value of particular fields we can use min function.

select max(empSalary) as salary, city from javabatch3.employee group by city;

min() aggregate function

if we want get minimum value of particular fields we can use min function.

select min(empSalary) as salary, city from javabatch3.employee group by city;

as keyword alias name

if we want use give alias names to aggregate function results we can use as keyword.

select max(empSalary) as highestSalary from javabatch3.employee group by city;

limit key word

if we want get particular count of records we can use limit keyword.

select \* from javabatch3.employee limit 4;

select \* from javabatch3.employee order by empName limit 4;