DBMS (Database Management System)

DBMS

↓

Database Management System

↓

Database + Management System

↓ Collection of data

↓ is a set of program through which we can insert or retrieve data

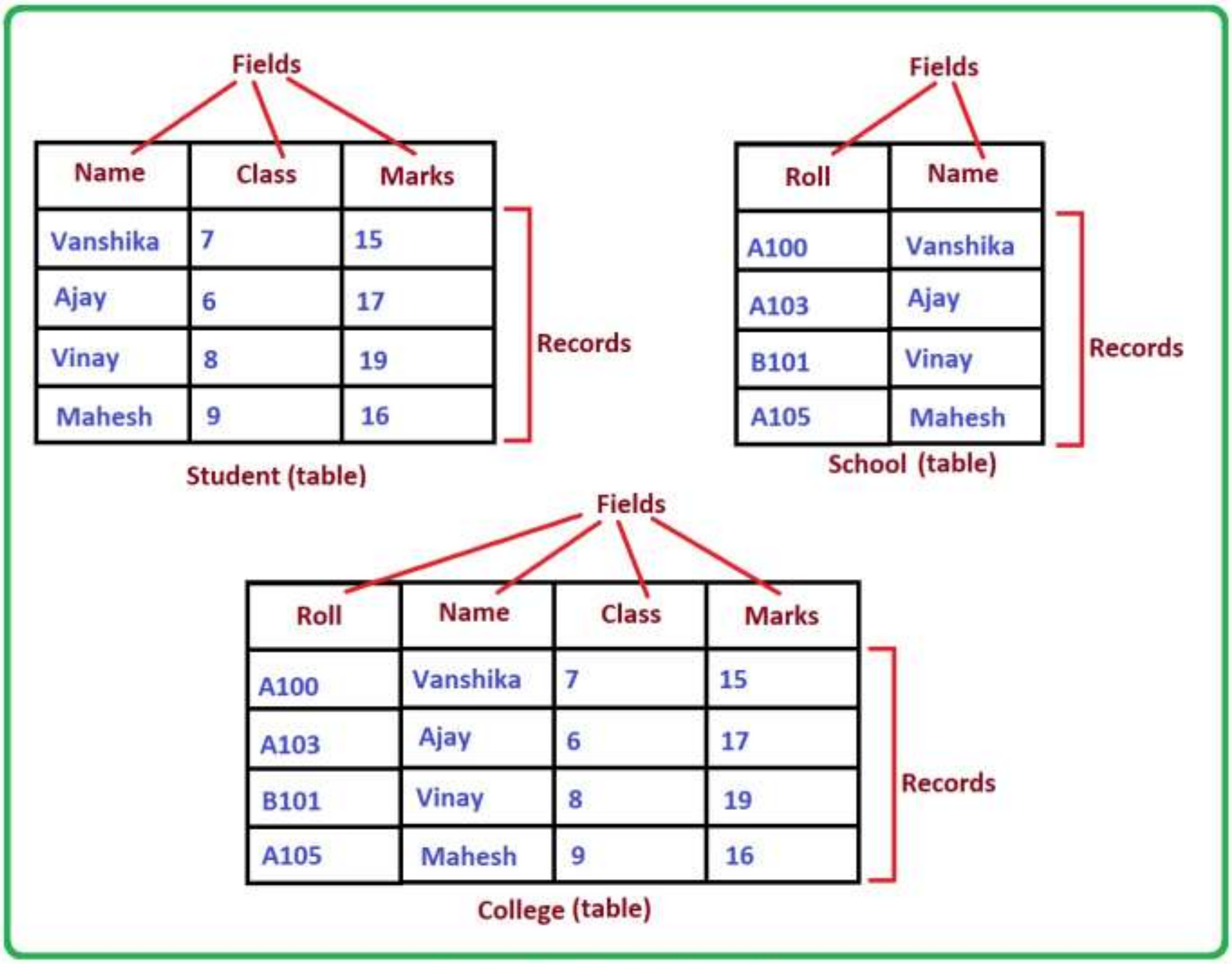
Definition : - DBMS is a collection of data and set of programs which is used to insert or obtain data in a easy manner. It is a software through which we can manage data.

Example : - MySQL, Oracle

****

RDBMS (Relational Database Management System)

The data in RDBMS stored in databases objects called tables. A table is a collection of related data entries and it consists of rows and columns.



Name	Class	Marks
Vanshika	7	15
Ajay	6	17
Vinay	8	19
Mahesh	9	16

Student (table)

Roll	Name
A100	Vanshika
A103	Ajay
B101	Vinay
A105	Mahesh

School (table)

Roll	Name	Class	Marks
A100	Vanshika	7	15
A103	Ajay	6	17
B101	Vinay	8	19
A105	Mahesh	9	16

College (table)

Testdb (Database)

SQL (Structured Query Language)

- ❖ SQL is a standard computer language for relational database management and data manipulation.
- ❖ SQL is used to query, insert, update and modify data.
- ❖ SQL is a domain specific language not general purpose language.

SQL first developed in the early 1970s at IBM by Raymond Boyce and Donald Chamberlin that time it was known as SEQUEL (Standard English Query Language). SQL was commercially released by Relational Software Inc. (now known as Oracle Corporation) in 1979.

SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

****

Sub-Divisions of SQL

1) **Data Definition Language (DDL):** – It is used for managing tables and index structures. Examples of DDL statements include Create, Alter, Truncate and Drop etc.
2) **Data Manipulation Language (DML):** – It is used to add, update or delete data. Examples of DML statements include Select, Insert, Delete, Update, Savepoint, Commit and Rollback etc.
3) **Data Control Language (DCL):** – It is used to assign and revoke database rights and permissions. Its main statements are Grant and Revoke.

****

SQL Data Types

A data type defines what kind of value a column can hold. Each column in a database table is required to have a name and a data type. An SQL developer must decide what type of data that will be stored inside each column when creating a table.

S.N.	Data Type	Description
1	CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
2	VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters.
3	TEXT	Holds a string with a maximum length of 65,535 characters
4	INT	It represents a number without decimal part.
5	FLOAT	It represents a floating point number.

****

SQL Statements (Based on MySQL)

1) **create database statement: –** The create database statement is used to create a new SQL database.
   Syntax: –
   ```
   create   database   database_name;
   ```
   Ex: –
   ```
   create   database   testdb;
   ```

2) **show database statement: –** The show database statement is used to display the list of SQL database.
   Ex: –
   ```
   show   databases;
   ```

3) **use database statement: –** The use database statement is used to apply an existing SQL database as current working database.

Syntax: –
```
use   database_name;
```
Ex: –
```
use   testDB;
```

4) **drop database statement: –** The drop database statement is used to drop an existing SQL database.

Syntax: –
```
drop database database_name;
```
Ex: –
```
drop database testDB;
```

5) **create table statement: –** The create table statement is used to create a new table in a database.

Syntax: –
```
create   table    table_name
(
field1  data_type,
field2  data_type
);
```
Ex: –
```
create    table    player
(
Name    varchar(20),
Marks    int(3)
);
```

6) **Display list of all tables: –**

(According MySQL)

Ex: –
```
show  tables;
```
(According Oracle)

Ex: –
```
Select * from tab;
```

7) **Inserting records into tables: –** The insert into statement is used to insert new records in a table.

a) **Syntax: –** insert   into   table_name   values(value1, value2, ……..);
   Ex: – insert   into   player   values('Rajesh',25);

b) **Syntax: –** insert   into   table_name(field_name) values(value);
   Ex: – insert   into   player(Name) values('Ramesh');

c) **Syntax: –** insert   into   table_name(field_name) values(value1), (value2),(value3), ………..;
   Ex: – insert   into   player(Name) values('Ajay'), ('Manoj'), ('Ravi');

d) **Syntax: –** insert   into   table_name   values(&column1,&Column2, ……..);
   Ex: – insert   into   player   values('&Name',&Marks);
   (**Note: – Above query no. d) is based on Oracle)**

8) **select statement: –** The select statement is used to select data from a database.

Syntax: –
```
select column1, column2, ……… from table_name;
```

Ex: –
a) select * from player;
b) select name from player;
c) select marks, name from player;

9) distinct statement: – The distinct keyword is used to eliminate duplicate records. It can be applicable with only one field.
Syntax: –
select distinct(field) .... from table_name;
Ex: –
select distinct(country) from customers;

10) all statement: – The all keyword displays all records as well as duplicate records.
Syntax: –
select all(field) .... from table_name;
Ex: –
select all(country) from customers;

11) where clause: – The where clause is used to filter records. The where clause is used to extract only those records that fulfill a specified condition.
Syntax: –
select field1, field2, ... from table_name where condition;
Ex: –
select * from player where marks>=20;
Note: – The where clause is not only used with select statement, it is also used with update, delete statement.

12) between operator: – The between operator selects values within a given range. The values can be numbers, text, or dates. The between operator is inclusive: begin and end values are included.
Syntax: –
select * from table_name where field_name between value1 AND value2;
Ex: –
select * from player where marks between 10 AND 20;

13) IN operator: – The IN operator allows us to specify multiple values in a where clause. The IN operator is a shorthand for multiple OR conditions.
Syntax: –
i) select * from table_name where field_name IN(value1, value2, ...);
ii) select * from table_name where field_name IN(select statement);
Ex: –
i) select * from player where marks IN(10,20,25);
ii) select * from player where marks NOT IN(10,20,25);
iii) select * from customers where country IN(select country from suppliers);

14) AND, OR, NOT operators: – The where clause can be combined with AND, OR, NOT operators. The AND, OR operators are used to filter records based on more than one condition.
- ❖ The AND operator displays a record if all the conditions separated by AND is TRUE.
- ❖ The OR operator displays a record if any of the conditions separated by OR is TRUE.
- ❖ The NOT operator displays a record if the condition(s) is NOT TRUE.

AND Syntax: –

select field1, field2, ……….. from table_name where condition1 AND condition2;

Ex: –

i) select * from player where marks=25 AND name='Rajesh';
ii) select * from player where marks=25 && name='Rajesh';

OR Syntax: –

select field1, field2, ……………. from table_name where condition1 OR condition2;

Ex: –

i) select * from player where name='Amit' OR name='Rajesh';
ii) select * from player where name='Amit' || name='Rajesh';

NOT Syntax: –

select field1, field2, ……….. from table_name where NOT condition;

Ex: –

select * from player where NOT name='Amit';

15) LIKE operator: – The LIKE operator is used in a where clause to search for a specified pattern in a column. There are two wildcards used in conjunction with the LIKE operator –

- ❖ **%** The percent sign represents zero, one, or multiple characters.
- ❖ _ The underscore represents a single character.

Syntax: –

select * from table_name where column_name like pattern;

Ex: –

select * from table_name where name like 'R%';

S.N.	LIKE Operator	Description
1	Where name like 'a%'	Finds any values that start with "a"
2	Where name like '%a'	Finds any values that end with "a"
3	Where name like '%p%'	Finds any values that have "p" in any position
4	Where name like '_r%'	Finds any values that have "r" in the second position
5	Where name like 'a%p'	Finds any values that start with "a" and ends with "p"
6	Where name like 'a_%_%'	Finds any values that start with "a" and are at least 3 characters in length

16) 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 but we can also use ASC keyword. To sort the records in descending order, use the DESC keyword.

Syntax: –

select * from table_name order by field_name ASC|DESC;

Ex: –

i) select * from player order by name ASC;
ii) select * from player order by marks DESC;

17) group by keyword: – 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.

Syntax: –

select function_name(field_name) from table_name group by field_name;

Ex: –

select COUNT(CustomerID) from customers group by country;

18) <u>**NULL values**</u>**: –** A field with a NULL value is a field with no value. If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

<u>**Syntax**</u>**: –**

select * from table_name where field_name IS NULL;

<u>**Ex**</u>**: –**

i) select * from player where name IS NULL;
ii) select * from player where name IS NOT NULL;

19) <u>**delete statement**</u>**: –** The <u>delete</u> statement is used to delete existing records in a table.

<u>**Syntax**</u>**: –**

delete from table_name where condition;

<u>**Ex**</u>**: –**

i) delete from player where name is null;
ii) delete from player where name='Amit';
iii) delete from player where marks<=15;

20) <u>**update statement**</u>**: –** The <u>update</u> statement is used to modify the existing records in a table.

<u>**Syntax**</u>**: –**

update table_name set field1=value1, field2=value2 where condition;

<u>**Ex**</u>**: –**

i) update player set name='Amit' where name='Rajesh';
ii) update player set name='Manish', marks=25 where name='Amit';

21) <u>**Aliases**</u>**:** – Aliases are used to give a table (or a column), a temporary name. Aliases are often used to make column names more readable. An alias only exists for the duration of the query.

<u>**Syntax**</u>**: –**

select column_name as alias_name from table_name;

<u>**Ex**</u>**: –**

select CustomerID as ID, CustomerName as Customer from Customers;

22) <u>**MIN() & MAX() functions**</u>**:** – The MIN() function returns the smallest value of the selected column. The MAX() function returns the largest value of the selected column.

<u>**MIN() Syntax**</u>**: –**

select min(field_name) from table_name;

<u>**Ex**</u>**: –**

select min(marks) from customers;

<u>**MAX() Syntax**</u>**: –**

select max(field_name) from table_name;

<u>**Ex**</u>**: –**

select max(marks) from customers;

23) <u>**COUNT(), AVG(), SUM() functions**</u>**:** – The COUNT() function returns the number of rows that matches a specified criteria. The AVG() function returns the average value of a numeric column. The SUM() function returns the total sum of a numeric column.

<u>**COUNT() Syntax**</u>**: –**

select count(field_name) from table_name where condition;

<u>**Ex**</u>**: –**

select count(marks) from player where marks<33;

<u>AVG() Syntax</u>: –

```
select avg(field_name) from table_name;
```

<u>Ex</u>: –

```
select avg(marks) from player;
```

<u>SUM() Syntax</u>: –

```
select sum(field_name) from table_name;
```

<u>Ex</u>: –

```
select sum(marks) from player;
```

24) <u>Creating a new table (duplicate table) from existing table</u>: – We can define a new table and put data into it without going through the usual data definition process. It can be done by using select statement with create table.

<u>Syntax</u>: –

```
create table new_table_name as
(
   select field1, field2, ………. from old_table_name where condition
);
```

<u>Ex</u>: –

```
create table  actor  as  select  *  from  player;
```

25) <u>drop table statement</u>: – The drop table statement is used to drop an existing SQL table.

<u>Syntax</u>: –

```
drop table table_name;
```

<u>Ex</u>: –

```
drop table player;
```

26) <u>rename table statement</u>: – The <u>rename table</u> statement is used to rename an existing SQL table.

(According MySQL)

<u>Syntax</u>: –

```
rename   table   old_table_name   to    new_table_name;
```

<u>Ex</u>: –

```
rename   table  player to actor;
```

(According Oracle)

<u>Syntax</u>: –

```
rename  old_table_name   to    new_table_name;
```

<u>Ex</u>: –

```
rename player to actor;
```

27) <u>alter table statement</u>: – The <u>alter table</u> statement is used to add, delete, or modify columns in an existing table. The <u>alter table</u> statement is also used to add and drop various constraints on an existing table.

i) <u>To add a column in a table</u>: –

<u>Syntax</u>: –

```
alter table table_name add column_name datatype;
```

<u>Ex</u>: –

```
alter table player add roll int(3);
```

ii) **To modify a column in a table: –**

Syntax: –

a) alter table table_name modify column field_name datatype;
   Ex: – alter table player modify column roll int(4);
   (Used with MY SQL, Prior version of Oracle 10G)

b) alter table table_name modify field_name datatype;
   Ex: – alter table player modify roll int(4);
   (Used with Oracle 10G & Later)

iii) **To drop a column in a table: –**

Syntax: –

alter table table_name drop column column_name;

Ex: –

alter table table_name drop column roll;

28) **Table Constraint: –** Constraints can be specified when the table is created with the create table statement or after the table is created with the alter table statement.

Syntax: –

```
create table table_name
(
field1 datatype constraint,
field2 datatype constraint,
field3 datatype constraint,
………………..
);
```

Ex: –

```
create table persons
(
ID int(3) NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255) NOT NULL,
Age int(2)
);
```

29) **Auto Increment Field: –** Auto increment allows a unique number to be generated automatically when a new record is inserted into a table. Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

In SQL we uses **serial** or **auto_increment** keyword to perform auto increment features. By default, the starting value for auto_increment is 1, and it will increment by 1 for each new record.

Ex: –

(Without using Constraint)

```
create table player(Student_ID  serial, Name char(15));
```

(Using Constraint)

```
create table player(Student_ID  int not null auto_increment, Name char(15));
```

To let the auto_increment sequence start with another value, use the following SQL statement –
alter table player auto_increment=100;

<u>Note</u>: – To insert a new record into the "player" table, we will not have to specify a value for the "Student_ID" column (a unique value will be added automatically).

<u>Ex</u>: –

i) insert into player(Name) values(‘Ramesh’);

ii) insert into player(Name) values(‘Ajay’), (‘Manoj’), (‘Ravi’);

****