**Phase 2**

4 classes

1st class intro to database using oracle database

2nd class intro to database using mysql database

3rd class into to database mongo db database

4th class into to cloud using AWS.

Database

Program using any language like java, python, php, c or C++ etc

Input

Process

Output

System.out.println(“Welcome”);

If we want to store the data

1 file base system

Limitation of file base system

1. Data redundancy (Duplicate records).
2. Data inconsistency (format of the file).

Type of file

Format of data store in file

Id,name,salary

1,Ravi,12000

1/Ravi/1200

1 Ravi 12000

3. security ( mode of the file may be read or write mode or execute mode)

4. CRUD Operation more complex Create or insert new record, read or view, update and delete

2 database system

Data : raw fact.

Information :processed data or meaningful data etc

Database : storing the data in table format if database is RDBMS etc.

It is a software which help to store the data in table format using row and column

DBMS : Database Management System

StudentTrainerInformation

RDBMS : Relational Database Management System

Student

PK (Primary key): if column is pk that column doesn’t allow duplicate.

That column doesn’t allow null or empty value.

In single table we can make only one single column as pk.

We can combine two column to make primary key.

If we make two column as pk every individual column allow duplicate but combination must be unique.

Database Model

Hierarchical model OS

Network model

Relation model

Student

PK

SID(int/number) Sname(varchar(10)) age(int) FK

1 Reeta 21 100

2 Meeta 22 100

3 Veeta 23 102

4 Keeta 25 null

Trainer

PK

TId TName tech

100 Raj Java

101 Ravi Python

102 Ramesh Angular

FK : Foreign key : if column is FK always refer to PK of same table or different table. If column is FK that column allow only those values which present in PK column. FK can allow null value.

PK

CompanyID ProdK Price

A X 100

B Y 200

A Y 300

B X 400

A X 500 Error

RDBMS database

Database name Company name

Oracle Oracle

MySQL Sun micro system (Oracle )

SQL Server 2022 Micro soft

Db2 IBM

All above Database known as RDBMS databases.

Database internally use file base system to store the data.

SQL : Structured Query Language : English language.

This language provide set of command which help to interact with database using

Console base

Or

GUI base

Mainly divided into 5 sub language.

DRL or DQL (Data Retrieval language or Data Query language)

In DRL all command star with Select clause

Select is use to retrieve the records from table using different ways.

DDL Data Definition language

Structure of table

Create, Drop, alter (modify table structure ), truncate (delete the records from table)

DML Data Manipulation language

Data or information storing in table or update or delete

Insert, Delete and update

TCL Transactional control language

Transactional control language etc.

If all DML operation successfully then we need to do commit.

If any query get error we need to do rollback. As well as savepoint

DCL Data control language

Grant : giving the permission to do the task in database with particular table

Like insert, delete, update and retrieve

Revoke : take back the permission etc.

SQL

Oracle

MySQL

Db2

SQL Server

MySQL Vs Oracle

MySQL open source

Oracle not an open source

Username by default root

Password root

show databases; it will show all databases present in your account

use databasename; we can switch from one database to another database.

Create database databaseame; it is use to create new database.

show tables; it will display all table present in your database or existing database.

select \* from tab; error

In Oracle Database login like scott, or admin or oracle

Tiger or admin or welcome or oracle

Database login itself is database consider.

show databases; but in oracle login name itself is one database. In oracle error.

use databasename; error;

create database databaseame; error

show tables; error

select \* from tab; this query display all table present in your account.

select \* from tab;

create table employee(id int primary key, name varchar(10), salary float);

drop table employee;

desc employee;

No SQL : Mongo Db, HBase, Cassandra,

Day 2

select \* from tab;

DDL

create table employee(id int primary key,name varchar(10), salary float)

drop table employee;

DML

Insert query

insert into employee values(1,’Ravi’,12000);

update query

update employee set salary = 125000 all employee salary update

update employee set salary = 125000 where id=1;

update employee set salary = 130000 where name like ‘Ravi’;

update employee set name=’Ravi Kumar’ where id=1 and salary=230000;

update employee set name=’Ravi Kumar’ where id=1 or salary=230000;

DRL or DQL

select \* from employee we can view all records present in employee table with

all columns

select name, salary from employee; this query retrieve particular columns details..

order by to display records ascending or descending order.

select \* from employee order by salary desc;

select \* from employee order by name asc;

SQL provide where clause to apply filter for table.

1. Relational operator

>, >=, <, <=, = , !=

Select \* from employee where salary > 150000

Select \* from employee where salary >= 150000

Select \* from employee where salary < 120000

Select \* from employee where salary = 120000

1. Between operator : range

Select \* from employee where salary between 12000 and 18000

1. In operator : more than one values conditions.

Select \* from employee where id in(3,1,5);

1. Like operator

Select \* from employee where name =’Ramesh’;

Select \* from employee where name like ’Ramesh’;

Select \* from employee where name like ’R%’; start with R character

Select \* from employee where name like ’%n’; end with n character

Select \* from employee where name like ’%e%’; contains e character

**Logical operator**

and or operator

Select \* from employee where name =’Ramesh’ and salary > 150000;

Select \* from employee where name =’Ramesh’ or salary > 150000;

MySQL database : MySQL is an open source RDMBS database.

Relationship

ER – Diagram

1 to many

**Student**

**Trainer**

T

hass

Entity Relationship Diagram

Entity is like a Table

Trainer

Student

Trainer and Student one to many relationship

Product

Order

Same product we can place more than one time order.

Employee

Project

Day 3

create database testdb;

use testdb;

create table trainer(tid int primary key,tname varchar(30) , tech varchar(30));

desc trainer;

create table student(sid int primary key,sname varchar(30), age int, tid int,

foreign key(tid) references trainer(tid));

desc student;

insert into trainer values(1,'Ravi','Java');

insert into trainer values(2,'Ramesh','Python');

insert into trainer values(3,'Vijay','MySQL');

insert into student values(100,'Reeta',21,1);

insert into student values(101,'Veeta',22,1);

insert into student values(102,'Meeta',22,2);

insert into student values(103,'Teeta',22,null);

select \* from trainer;

select \* from student;

Join

Retrieve more than one column from more than one table with condition.(PK and FK).

Equi join

Where clause

select t.tname,t.tech,s.sname from trainer t, student s where t.tid=s.tid;

Inner Join

On clause

Common in both the tables

select t.tname,t.tech,s.sname from trainer t inner join student s on t.tid=s.tid;

common as well as first table remaining records

select t.tname,t.tech,s.sname from trainer t left outer join student s on t.tid=s.tid;

command as well as second table remaining records.

select t.tname,t.tech,s.sname from trainer t right outer join student s on t.tid=s.tid;

functions

function takes one or more than one parameter and it return the value.

1. Single row function : the function functionality apply for each records independently.
2. Multi row function : the function functionality apply for more than one records base hpon group.

Multi row function also known as aggregate function.

Sum()

Avg()

max()

min()

count()

Date format in MySQL default

Yyyy-mm-dd

Date format in oracle default

dd-mon-yy

<https://dev.mysql.com/doc/refman/8.0/en/built-in-function-reference.html>

create database testdb;

use testdb;

create table trainer(tid int primary key,tname varchar(30) , tech varchar(30));

desc trainer;

create table student(sid int primary key,sname varchar(30), age int, tid int,

foreign key(tid) references trainer(tid));

desc student;

insert into trainer values(1,'Ravi','Java');

insert into trainer values(2,'Ramesh','Python');

insert into trainer values(3,'Vijay','MySQL');

insert into student values(100,'Reeta',21,1);

insert into student values(101,'Veeta',22,1);

insert into student values(102,'Meeta',22,2);

insert into student values(103,'Teeta',22,null);

select \* from trainer;

select \* from student;

select \* from trainer;

select \* from student;

select tname,tech,tname from trainer, student where trainer.tid=student.tid;

select trainer.tname,trainer.tech,student.sname from trainer, student where trainer.tid=student.tid;

select t.tname,t.tech,s.sname from trainer t, student s where t.tid=s.tid;

select t.tname,t.tech,s.sname from trainer t inner join student s on t.tid=s.tid;

select t.tname,t.tech,s.sname from trainer t left outer join student s on t.tid=s.tid;

select t.tname,t.tech,s.sname from trainer t right outer join student s on t.tid=s.tid;

select \* from trainer;

select tid,tname from trainer;

select tid,upper(tname) from trainer;

select tid,lower(tname) from trainer;

select sysdate();

select curdate();

create table employee(id int primary key,name varchar(30) not null, dob date);

insert into employee values(1,'Ravi','2019-09-20');

insert into employee values(2,'Lokesh','2018-12-08');

select \* from employee;

select name,date\_format(dob,'%d-%m-%Y') from employee;

create table college(cid int primary key, name varchar(10),subject varchar(10),city varchar(10),numberOfStd int);

insert into college values(1,'Raj','Phy','Bangalore',30);

insert into college values(2,'Seeta','Che','Bangalore',40);

insert into college values(3,'Reeta','Bio','Bangalore',20);

insert into college values(4,'Veeta','Maths','Bangalore',50);

insert into college values(5,'Teeta','Phy','Mumbia',60);

insert into college values(6,'Leeta','Che','Mumbia',20);

insert into college values(7,'Meeta','Bio','Mumbia',10);

insert into college values(8,'Keeta','Maths','Mumbia',60);

insert into college values(9,'Raju','Phy','Delhi',30);

insert into college values(10,'Ramesh','Che','Delhi',60);

insert into college values(11,'Suresh','Bio','Delhi',70);

insert into college values(12,'Lokesh','Maths','Delhi',80);

select \* from college;

select sum(numberOfStd) from college;

select city,sum(numberOfStd) as total from college group by city;

select concat(tname,' teach ',tech) as trainer\_details from trainer;

select from employee e1, e2 where e1.emp\_id=e2.manager\_id;

select \* from trainer where tech='Java' union select \* from trainer where tech='Python';

create table developer(id int primary key, name varchar(10));

insert into developer values(100,'Ravi');

insert into developer values(101,'Raj');

create table tester(id int primary key,name varchar(10));

insert into tester values(100,'Ravi');

insert into tester values(102,'Lokesh');

select \* from developer union select \* from tester;

select \* from developer union all select \* from tester;

select max(numberOfStd) from college;

select min(numberOfStd) from college;

select avg(numberOfStd) from college;

select count(\*) from college;

Oracle

MySQL

Db2

SQL Server

RDBMS Database

Those database are schema base database.

Before storing records in database we need to create structure for table ie

Table Name 🡪 Number of columns the table contains with data types as well as keys like PK, FK, not null etc.

RDBMS always store the records in table format.

Data :

Always data mainly divided into 3 category.

1. Un structure
2. Semi structure : XML or JSON etc
3. Structure

No SQL Databases

Mongo DB Database

Mongo DB is an open source No SQL Database. Which help to store the using document concept in the form of JSON(JavaScript Object Notation).

JSON

Employee emp = new Employee();

emp.setId(100);

emp.setName(“Ravi”);

emp.setSalary(12000);

XML or JSON

<Employee>

<Id>100</Id>

<Name>Ravi</Name>

<Salary>12000</Salary>

</Employee>

JSON

{“id”:100,”name”:”Ravi”,”salary”:12000}

Limitation of RDBMS

Employee

PK not null

Id Name Salary Age City

1 Ravi 12000 null null

2 Ramesh 14000 null null

3 Ajay 16000 23 null

4 Reeta 18000 null Bangalore

Insert into Employee values(5,’Meeta’,34000,21); Error

Insert into Employee values(5,’Meeta’,34000,21,null); correct

Insert into Employee(id,name,salary,age) values(5,’Meeta’,34000,21); correct

Insert into Employee(id) values(6);

Create table employee(id int primary key,name varchar(10));

Id name

1 Ravi

2 null

alter table employee modify name varchar(10) not null;

In mongo DB we use collection concept. Collection is like a table.

Collection contains group of records or document. Each document can be same type or different types.

MySQL MongoDB

Database database

Table collection

Records table format document hold the data in json format.

Same type any types.

mongo to connect mongo db shell

to exit mongo terminal cntr + C

clear the screen cntr + L

show dbs;

or

show databases; this command display all databases present in mongo db account.

create database mydb; error in mongo db.

use mydb; if database not present it will create database and switch to that database.

If database present it directly switch that database.

Mongo db provided pre defined object or reference which contains lot of method to do some task in mongo db database

db

db.createCollection("Sample"); it is use to create the collection

show collections it is use to display all collection present in that current

or databases.

show tables

db.CollectionName.insert({key:value});

db.Sample.insert({name:"Ravi"}); document store which contains name property

> db.Sample.insert({name:"Ravi"});

WriteResult({ "nInserted" : 1 })

> db.Sample.insert({name:"Raj",age:21});

WriteResult({ "nInserted" : 1 })

> db.Sample.insert({name:"Steven",age:26,city:"Bangalore"});

WriteResult({ "nInserted" : 1 })

We want to view all document from collection like table

db.CollectionName.find();

db.Sample.find();

In mogo db \_id is like a PK. If we doesn’t pass \_id property value mongo db internally create \_id with random value. If you want to pass custom value we can pass but property name must be \_id with case sensitive.

db.Sample.insert({\_id:1,name:"John"});

first create the collection with 10 documents

db.Employee.insertMany([

{\_id:1,name:"Ravi",age:21,salary:56000,city:"Bangalore",dept:100},

{\_id:2,name:"Steven",age:25,salary:58000,city:"Delhi",dept:101},

{\_id:3,name:"Raju",age:28,salary:52000,city:"Mumbai",dept:102},

{\_id:4,name:"Lex",age:23,salary:56000,city:"Bangalore",dept:100},

{\_id:5,name:"John",age:28,salary:76000,city:"Delhi",dept:101},

{\_id:6,name:"Rooma",age:31,salary:86000,city:"Mumbai",dept:100},

{\_id:7,name:"Raju",age:29,salary:96000,city:"Mumbai",dept:102},

{\_id:8,name:"Ajay",age:24,salary:55000,city:"Delhi",dept:101},

{\_id:9,name:"Vijay",age:29,salary:44000,city:"Bangalore",dept:100},

{\_id:10,name:"Balaji",age:22,salary:32000,city:"Mumbai",dept:102}

])

View the document using index position

db.Employee.find()[0]; retrieve particular index position document

db.Employee.find()[2];

db.Employee.find()[2].name retrieve particular index position document field value.

db.CollectionName.find({condition},{field1:1,field2:1});

db.Employee.find({},{name:1,age:1}); it display name, age and \_id field values for all documents

db.Employee.find({},{name:1,age:1,\_id:0}); it display name and age field value for all documents.

db.Employee.find({},{name:1,city:1,\_id:0})[0]; retrieve name and city using index position

like where clause

db.collectionName.find({condition});

db.Employee.find({\_id:1});

db.Employee.find({city:"Bangalore"});

db.Employee.find({dept:100});

db.Employee.find({salary:{$gt:60000}});

db.Employee.find({salary:{$lt:60000}});

db.Employee.find({salary:{$gte:55000}});

db.Employee.find({salary:{$lte:55000}});

db.Employee.find({salary:{$ne:55000}});

db.Employee.find({salary:{$eq:55000}});

sort the document using any fields

db.Employee.find().sort({salary:1}); asc

db.Employee.find().sort({salary:-1}); desc

update document fields

db.collectionName.update({condition},{$set:{field:newValue}});

db.Employee.update({\_id:1},{$set:{salary:60000}});

db.Employee.update({dept:100},{$set:{city:"Banguluru"}});

db.Employee.updateMany({dept:100},{$set:{city:"Banguluru"}});

remove document

db.collectionName.remove({});

db.collectionName.remove({\_id:10});

db.collectionName.remove({city:”Bangalore”);

rename all documents fields

db.Employee.updateMany({},{$rename:{"dept":"deptId"}});

Student

Sid sname age marks

1 Raj 21 67,89,76,90

Students

PK

SID SName Age

1 Raj 21

Marks

PK FK

MID Sid Marks Subject

123 1 67 Phy

124 1 89 Maths

125 1 76 Bio

126 1 90 Che

In Mongo db we are storing information using json format

So value can be number, string, Boolean , array, object type as well as array object type.

{\_id:1,sname:”Ravi”,age:21,marks:[67,89,76,90]}

> db.StudentDetails.insert({\_id:1,sname:"Ravi",age:21,marks:[67,89,77,90]});

WriteResult({ "nInserted" : 1 })

> db.StudentDetails.insert({\_id:2,sname:"Steven",age:24,marks:[87,84,72,92]});

WriteResult({ "nInserted" : 1 })

> db.StudentDetails.find();

Mongo Db Relationship

One to One relationship Employee has Passport

One to Many relationship In one department or project many employees are working.

To make relationship in RDBMS we use Pk and FK.

Mongo db support two types of relationship

1. Embedded styling
2. Linking style

Mongo Db maintain the relationship using document not using collection.

But in RDBMS we maintain the relationship on table level.

Employees

1st Employee details \_id:1, name:”Ravi”,age:21,salary:12000

Address details city:”bangalore”,state:”Kar”

Project pid:100,typeofproject:”web”,tech:”java”

Pid:101,tyepofproject:”mobile app”,tech:”android”

Linking Style

Trainer1

PK

\_id TName tech

1 Ravi Java

2 Ramesh Python

db.Trainer1.find();

db.Trainer.insert({\_id:1,tname:"Ravi",tech:"Java"});

db.Trainer.insert({\_id:2,tname:"Ramesh",tech:"Python"});

Student1 we will store only trainer id

PK

\_id SName Age TID or TrainerDetails

100 Reeta 21 1 {\_id:1,tname:”Ravi”,”Java”}

101 Meeta 22 1

102 Veeta 23 2

103 Leeta 23 [1,2]

db.Student1.find();

db.Student1.insert({\_id:100,sname:"Reeta",age:21,tid:db.Trainer.find()[0].\_id});

db.Student1.insert({\_id:101,sname:"Meeta",age:22,tid:db.Trainer.find()[0].\_id});

db.Student1.insert({\_id:102,sname:"Veeta",age:23,tid:db.Trainer.find()[1].\_id})

db.Student1.insert({\_id:103,sname:"Leeta",age:23,tid:[db.Trainer.find()[0].\_id,db.Trainer.find()[1].\_id]});

Student2 we will store trainer details

db.Student2.insert({\_id:100,sname:"Reeta",age:21,tdetails:db.Trainer.find()[0]});

db.Student2.insert({\_id:101,sname:"Meeta",age:22, tdetails:db.Trainer.find()[0] });

db.Student2.insert({\_id:102,sname:"Veeta",age:23, tdetails:db.Trainer.find()[1] })

db.Student2.insert({\_id:103,sname:"Leeta",age:23, tdetails:[db.Trainer.find()[0],db.Trainer.find()[1]]});

Document present in Trainer, Student1, Student2 Collection

> db.Trainer.find();

{ "\_id" : 1, "tname" : "Ravi", "tech" : "Java" }

{ "\_id" : 2, "tname" : "Ramesh", "tech" : "Python" }

> db.Student1.find();

{ "\_id" : 100, "sname" : "Reeta", "age" : 21, "tid" : 1 }

{ "\_id" : 101, "sname" : "Meeta", "age" : 222, "tid" : 1 }

{ "\_id" : 102, "sname" : "Veeta", "age" : 222, "tid" : 2 }

{ "\_id" : 103, "sname" : "Leeta", "age" : 23, "tid" : [ 1, 2 ] }

> db.Student2.find();

{ "\_id" : 100, "sname" : "Reeta", "age" : 21, "tid" : { "\_id" : 1, "tname" : "Ravi", "tech" : "Java" } }

{ "\_id" : 101, "sname" : "Meeta", "age" : 22, "tid" : { "\_id" : 1, "tname" : "Ravi", "tech" : "Java" } }

{ "\_id" : 102, "sname" : "Veeta", "age" : 23, "tid" : { "\_id" : 2, "tname" : "Ramesh", "tech" : "Python" } }

{ "\_id" : 103, "sname" : "Leeta", "age" : 23, "tid" : [ { "\_id" : 1, "tname" : "Ravi", "tech" : "Java" }, { "\_id" : 2, "tname" : "Ramesh", "tech" : "Python" } ] }

>

Aggregate operator or function : it use to combine more than one collection and base upon operator

It will return one or more than one result.

db.Trainer.insert({\_id:1,tname:"Ravi",tech:"Java"});

db.Trainer.insert({\_id:2,tname:"Ramesh",tech:"Python"});

db.Student1.insert({\_id:100,sname:"Reeta",age:21,tid:db.Trainer.find()[0].\_id});

db.Student1.insert({\_id:101,sname:"Meeta",age:22,tid:db.Trainer.find()[0].\_id});

db.Student1.insert({\_id:102,sname:"Veeta",age:23,tid:db.Trainer.find()[1].\_id})

db.Student1.insert({\_id:103,sname:"Leeta",age:23,tid:[db.Trainer.find()[0].\_id,db.Trainer.find()[1].\_id]});

db.Student2.insert({\_id:100,sname:"Reeta",age:21,tdetails:db.Trainer.find()[0]});

db.Student2.insert({\_id:101,sname:"Meeta",age:22, tdetails:db.Trainer.find()[0] });

db.Student2.insert({\_id:102,sname:"Veeta",age:23, tdetails:db.Trainer.find()[1] })

db.Student2.insert({\_id:103,sname:"Leeta",age:23, tdetails:[db.Trainer.find()[0],db.Trainer.find()[1]]});

> db.Trainer.find();

{ "\_id" : 1, "tname" : "Ravi", "tech" : "Java" }

{ "\_id" : 2, "tname" : "Ramesh", "tech" : "Python" }

>db.Student1.find();

{ "\_id" : 100, "sname" : "Reeta", "age" : 21, "tid" : 1 }

{ "\_id" : 101, "sname" : "Meeta", "age" : 222, "tid" : 1 }

{ "\_id" : 102, "sname" : "Veeta", "age" : 222, "tid" : 2 }

{ "\_id" : 103, "sname" : "Leeta", "age" : 23, "tid" : [ 1, 2 ] }

db.Student1.aggregate([{

$lookup: {

from :"Trainer",

localField:"tid",

foreignField:"\_id",

as:"TrainerDetails"

}

}]).pretty();

db.Trainer.aggregate([{

$lookup:{

from :"Student1",

localField:"\_id",

foreignField:"tid",

as:"StudentDetails"

}

}]).pretty();

With aggregate function we will use $group operator

we make group city wise

db.Employee.aggregate([{

$group:{\_id:'$city'}

}]);

db.Employee.aggregate([{ $group:{\_id:'$city',totalSalary:{$sum:'$salary'}} }]);

db.Employee.aggregate([{ $group:{\_id:'$city',maxSalary:{$max:'$salary'}} }]);

db.Employee.aggregate([{ $group:{\_id:'$city',minSalary:{$min:'$salary'}} }]);

db.Employee.aggregate([{ $group:{\_id:'$city',avgSalary:{$avg:'$salary'}} }]);

we make group deptId

db.Employee.aggregate([{

$group:{\_id:'$deptId'}

}]);

db.Employee.aggregate([{ $group:{\_id:'$deptId',totalSalary:{$sum:'$salary'}} }]);