**Java Full Stack Training Notes**

**Day 1**

**11/01/2020**

**Database :**

Program :

Input : Data

Process : Business rules

Output :Data

1 File based system

2 Database System

Limitation of File base system

1. Data Redundancy (Storing same records or duplicate records)
2. Data inconsistency (format of the file)
3. Data Security (read or write mode)

.txt

Employee.txt

Id,name,Salary

1,Ravi,12000

2,Ramesh,1400

Employee.txt

Id name Salary

1 Ravi 12000

2 Ramesh 1400

.doc

.pdf

**Database :**

Data : row fact

Information : Processed data or meaningful data.

Database : Storing data and information in proper format like table.

DBMS :Database Management System. It is a software which help to store the data in a table format.

Excel sheet is a small DBMS.

Employee – Table

Columns

Id Name Salary Row

1 Ravi 12000

2 Ramesh 14000

Database Model

Hierarchical Model

Network Model

RDBMS

Relational model

TrainerStudentRecords

TId TName Tech Sid SName Age

1 Raj Java 100 Seeta 21

1 Raj Java 101 Meeta 22

1 Raj Java 102 Leeta 23

2 Ravi Python 103 Keeta 24

2 Ravi Python 104 Veeta 25

Trainer

PK

Tid TName Tech

1 Raj Java

2 Ravi Python

Student

PK FK

Sid SName Age TSId

100 Seeta 21 1

101 Meeta 22 1

102 Veeta 23 2

103 Leeta 24 null

104 Teeta 25 3 Error

RDBMS : Table is known as Relation

Column is known as Attribute

Row is known as Tuple or records

1970

Dr EF Codd’s Rules

12 Rules start from 0 to 11

RDBMS

RDBMS Company

Oracle Oracle

MySQL Sun Microsystem (Oracle)

SQL Server

2020/19 Microsoft

Db2 IBM

SQL : Structure Query Language

5 types

1. DRL or DQL (Data Query Language or Data Retrieval Language)
   1. Select Clause
2. DDL : Data Definition Language
   1. Create, Drop, Alter , Truncate , rename
3. DML : Data Manipulation Language
   1. Insert , Delete and Update
4. TCL : Transaction Control Language
   1. Commit, rollback and savepoint
5. DCL : Data Control Language
   1. Grant and revoke

Oracle :

MySQL :

90% SQL Query

**Oracle :**

In Oracle Database Login itself is database.

**scott : username**

**By default oracle database username is scott/admin/oracle etc**

**Scott username**

**Tiger password**

**select \* from tab;** tab is pre-defined table provided by oracle database to check all pre-defined as well as user-defined tables.

MySQL/db2/sql server

**create database databaseName;**

**show databases;**

**use databaseName;**

**show tables;**

**DRL or DQL**

**use test**

**show tables**

**select \* from tableName;**

select \* from employees;

select \* from departments;

select \* from locations;

to check the table structure

**desc tableName;**

desc employees;

display specific columns

select columnname,columname from tableName;

**columnName alias**

select employee\_id as emp\_id, first\_name as f\_name, last\_name as l\_name from employees;

select first\_name,last\_name,concat(first\_name,last\_name) as Full\_name ,salary,salary+1000 as GrossSalary from employees;

Employee\_Id, FullName, Basic\_Salary, HRA, DA, PF, GrossSalary

**1st Assignment**

Full\_name --- First\_name and Last\_name

Salary --- Basic\_Salary

HRA –10%

DA – 5%

PF – 7%

GrossSalary – Salary +HRA + DA – PF

**Filter the Data**

Where clause

Select \* from tableName;

Select columnName,columName from tableName 🡪 filter the columns

1. Relational Operators

Select \* from tableName where columnName RO Value

>

>=

<

<=

= (is a equal Operator in SQL)

!= or <>

select \* from employees where salary > 10000;

select \* from employees where salary <> 17000;

1. Between operator (range of records)

Select \* from tableName where columnName between min and max

Select first\_name from employees where salary between 5000 and 10000

MySQL default date format is yyyy-mm-dd

Oracle default date format is dd-mon-yy

Display all employee first\_name and salary where hire date between 1990 to 1995.

1. In operator (specific more than one values)

Select \* from tableName where columName in (v1,v2,v3);

select first\_name,salary,hire\_date from employees where job\_id in('IT\_PROG','ST\_MAN');

select first\_name,salary,hire\_date from employees where employee\_id in(100,120,10000);

1. Like operator

Select \* from employees where first\_name like ‘Steven’;

Select \* from employees where first\_name = ‘Steven’;

Start with **S** Character ‘

Select \* from employees where first\_name like ‘S%’;

End with **a** character

Select \* from employees where first\_name like ‘%a’;

Contains **e** character

Select \* from employees where first\_name like ‘%e%’;

% zero or infinity

\_ any single character

Start with **L** as first character, second may be anything and third character must be **u**

select first\_name from employees where first\_name like 'L\_u%';

1. is null

select first\_name,salary,commission\_pct from employees where commission\_pct is null;

select first\_name,salary,commission\_pct from employees where commission\_pct is not null;

**Logical operators**

1. and
2. or
3. not

**Not**

select \* from employee where salary between 10000 and 20000

select \* from employee where salary not between 10000 and 20000

select \* from employee where job\_id in(‘IT\_PROG’,’ST\_MAN’)

select \* from employee where job\_id not in(‘IT\_PROG’,’ST\_MAN’)

select \* from employee where first\_name like ‘Steven’;

select \* from employee where first\_name not like ‘Steven’;

**and**

select first\_name from employees where salary > 12000 and department\_id=90;

**or**

select first\_name from employees where salary > 12000 or department\_id=90;

**Order by clause**

This clause mainly use to sort the records ascending or descending order.

Select \* from tableName order by columnName asc/desc

By default asc consider

select first\_name,salary from employees order by salary;

select first\_name,salary from employees order by salary asc;

select first\_name,salary from employees order by salary desc;

select first\_name,salary from employees where employee\_id between 110 and 150 order by salary desc;

Multi sort

select first\_name,department\_id,salary from employees order by department\_id desc,salary asc;

**Joins :** Joining is use to retrieve more than one columns from more than one table with or without conditions.

Create table table1(srno int, name varchar(10), salary float);

Insert into table1 values(1,’Ravi’,12000);

Insert into table1 values(2,’Ram’,14000);

Create table table2(accno int, name varchar(10), amount float);

Insert into table2 values(100,’Ravi’,500);

Insert into table2 values(101,’Ramesh’,1000);

1. **Cartesian product :**

Select columnname1,columnname2…… from table1, table2…. Tablen

M\*N

select srno,salary,accno,amount from table1,table2;

**Table Name with common columns**

select table1.srno,table1.salary,table1.name,table2.accno,table2.amount from table1,table2;

**Table alias**

select t1.srno,t1.salary,t1.name,t2.accno,t2.amount from table1 t1,table2 t2;

**Equi Join using where clause**

select t1.srno,t1.salary,t1.name,t2.accno,t2.amount from table1 t1,table2 t2 where t1.name = t2.name;

Common records

Left Outer Join : Left table remaining records + common records

select t1.srno,t1.salary,t1.name,t2.accno,t2.amount from table1 t1,table2 t2 where t1.name = (+)t2.name; **In oracle database**

Right Outer Join : right Table remaining records + common records

select t1.srno,t1.salary,t1.name,t2.accno,t2.amount from table1 t1,table2 t2 where t1.name(+) = t2.name; **In oracle database**

**Inner Join using on clause**

select t1.srno,t1.salary,t1.name,t2.accno,t2.amount from table1 t1 inner join table2 t2 on t1.name = t2.name;

**Left Outer join**

select t1.srno,t1.salary,t1.name,t2.accno,t2.amount from table1 t1 left outer join table2 t2 on t1.name = t2.name;

**Right Outer join**

select t1.srno,t1.salary,t1.name,t2.accno,t2.amount from table1 t1 right outer join table2 t2 on t1.name = t2.name;

**FK = PK**

**De-Normalization**

Employee

Id Name

10

Project 5 employee – 5 project

Pid ProjectName

10

**Normalization**

First\_name,Salary,Department\_name

select emp.employee\_id,emp.first\_name,emp.salary,dept.department\_name from employees emp, departments dept where emp.department\_id = dept.department\_id;

First\_name, Department\_name, City

Where clause

On clause

**Self Join** : Joining the same table itself is known as Self join

select emp1.employee\_id,emp1.first\_name,emp1.job\_id,emp2.first\_name,emp2.job\_id from employees emp1, employees emp2 where emp1.manager\_id = emp2.employee\_id;

**MySQL functions**

Function : It contains set of instruction to perform a specific task.

Database Function : Function take one or more parameter and return output or results.

2 types of

1. Pre-defined function
2. User-defined function (PL SQL).

Pre-defined functions

1. Single row function : The output or result apply for every records independently.
2. **String functions**

select **upper**(first\_name),**lower**(first\_name),**concat**(first\_name,' ',last\_name),**length**(first\_name) from employees;

nested function

select **length(concat**(first\_name,' ',last\_name)) from employees;

select **substring**(first\_name,2,2) from employees;

1. **Date functions**

select upper('raj deep'),curdate(),sysdate();

select datediff('2021-01-11','2021-01-05');

select datediff(curdate(),'2021-01-05');

select date\_format(curdate(),'%d-%m-%y');

1. **Maths functions**

select ceil(10.1),ceil(10.9),floor(10.1),floor(10.9);

select round(2356.4578,2);

select truncate(2356.4578,2)

1. Multi row function : The output or result apply for all records depending upon the group.

**Aggregate Functions**

sum(), count(), max(), min(), avg() etc

select sum(salary) as TotalSalary from employees;

select avg(salary) as AvgSalary,max(salary) as MaxSalary,min(salary) as MinSalary from employees;

select count(commission\_pct),count(manager\_id),count(employee\_id) from employees;

select count(\*), count(employee\_id) from employees;

**group by clause**

select aggregatefunction from tableName group by columnName(FK)

select sum(salary) from employees group by department\_id;

select department\_id,sum(salary) from employees group by department\_id;

**Where clause**

It is before group by

Condition apply for every individual records

Where clause we can use without group by

**Having clause**

After group by clause

Conditions apply for group by records.

But having clause we have to use with group by clause

select department\_id,sum(salary) from employees where department\_id is not null group by department\_id having sum(salary) > 50000 order by department\_id desc;

**Git : Sub Version Control System**

**Centralized repository**

Github :

Code commit : AWS

Azure

Google cloud

Oracle Cloud

DevOps tools

**Day 2**

**12/01/2020**

**DDL :**

**Create table**

**Syntax**

**Create table tableName(columnName dataType1,**

**columnName dataType2,**

**columnName datatTypen)**

**Emp**

**Id Name Salary**

**int varchar(10) float**

**Oracle : number(10), number(10,2)**

**create table emp(id int,name varchar(10), salary float);**

**DML :**

Insert into tableName(columnName1,columnName2,columnName3) values(v1,v2,v3);

insert into emp(id,name,salary) values(1,'Ravi',12000);

insert into emp values(2,'Ramesh',14000);

insert into emp(id,name) values(3,'Raju');

insert into emp values(4,'Ajay',null);

insert into emp(id) values(5);

insert into emp values(6,null,null);

insert into emp(name,id,salary) values('Dinesh',7,22000);

**Delete Query**

delete from tableName; : all records

**delete from emp;**

Delete with where clause

Delete from tableName where clause

**delete from emp where id=1;**

**delete from emp where name like ‘Ravi’;**

**delete from emp where salary between 2000 and 5000;**

**update Query**

syntax

update tableName set columnName = value;

update emp set salary = 20000; update salary for all employee

update with where clause

update tableName set columnName = value where clause

update emp set salary = 20000 where id=2;

update emp set name=’Ravi Kumar’ where name like ‘Ravi’;

update emp set salary = 22000 where salary = 18000;

DDL

Create

Drop

Drop table tableName ;

drop table sample;

**drop table sample;**

it will remove all records as well as table structure.

Truncate

Truncate table tableName;

It remove all records from table but maintains the table structure.

**truncate table sample**

delete

1. It is a part of DML
2. Using delete we can use where clause
3. Without where clause delete all records but maintains table structure.
4. We can use TCL commands.

truncate

1. It is part of DDL
2. We can’t use where clause
3. Delete all records but maintains table structure.
4. We can’t use TCL commands.

drop

1. It is part of DDL
2. Remove all records as well as table structure.

Alter command

1. Add column

alter table emp add desg varchar(2);

1. Modify column

alter table emp modify desg varchar(10);

1. Rename the column

alter table emp rename column desg to designation;

1. Change the tableName

alter table emp rename to empdetails;

1. Drop column

alter table empdetails drop column designation;

DDL

DML

Online shopping

Online Examination

custId,custName,age,phnumber,accnumber,typeofaccount,amount,pid,productName,price,mgrId,managerName,phnumber etc

**Single table**

**Super keys**

**Any one**

custId PK

accNumber, unique

phNumber, unique

mgrid unique

pid unique Candidate keys

CustId,CustName : super key but not candidate key

CustName not super key not a candidate key

CustId : is super key as well as candidate key

cusId,mgrId,

custId,Pid

custId,Pdi,mgrId

custId,phNumber

custid,phNumber,pid,

etc

**Candidate keys**

custid,phNumber,pid

phNumber,Pid

Pid CK

PhNumber CK

CustId,CustName :Super key

CustName :not CK

CustId :CK

MySQL

auto\_increment : generate the numbers one by one

Oracle

sequence: using this they generate auto increment number.

**🡨-PK-🡪(ProductId,customerId):**

**ProductId, CustomerId Price**

A X 10

A Y 20

B X 20

B Y 40

A X 50 Error

Composite primary key

**Constraints :** It is use to restrict the user to insert the invalid data in table.

1. Not null
2. Unique
3. Default
4. enum
5. Primary key
6. Foreign key
7. Check constraints : it doesn’t support.

EmpDetails

PK not null male/female unique IBM

EmpId FName Age gender phnumber Company

create table empdetails(

empid int primary key,

fname varchar(10) not null,

age int,

gender enum('male','female'),

phnumber int unique,

company varchar(10) default 'IBM');

**Relationship : Using Primary key and foreign key**

4 types

One means primary key and many foreign key

1. One – to – many : Trainer -- Student
2. Many – to – one : Employee -- Project
3. One – to one : Person -- Passport
4. Many – to – Many : Students -- Technologies

ER Diagram : Entity Relationship Diagram

One – to – Many

Trainer

PK

TId TName Tech

Column Level

**create table trainer(tid int primary key,tname varchar(10) not null,tech varchar(10) not null);**

Table Level

**create table trainer(tid int, tname varchar(10) not null, tech varchar(10) not null, constraint t\_pk primary key(tid));**

Student

PK FK

Sid SName Age TSId

**create table student(sid int primary key,sname varchar(10) not null,age int,tsid int references trainer(tid)); Not consider as FK in mysql but consider as In Oracle.**

Table Level

**create table student(sid int,**

**sname varchar(10) not null,**

**age int, tsid int,**

**constraint s\_pk primary key(sid),**

**constraint ts\_fk foreign key(tsid) references trainer(tid));**

**create table trainer(tid int,**

**tname varchar(10) not null,**

**tech varchar(10) not null,**

**constraint t\_pk primary key(tid));**

**Drop Foreign key**

alter table student drop foreign key ts\_fk;

**Add foreign key on student table after table created with records**

alter table student add constraint ts\_fk foreign key(tsid) references trainer(tid);

**Drop the primary key**

alter table student drop primary key;

**Add the primary key after table created with records (records must be unique)**

alter table student add constraint s\_pk primary key(sid);

**Keys**

Constraints

SQL

NoSQL

MongoDB

JSON

Document

Id,name,age

Id,phnumber

Id,image

Info