"DATABASE" MySQL

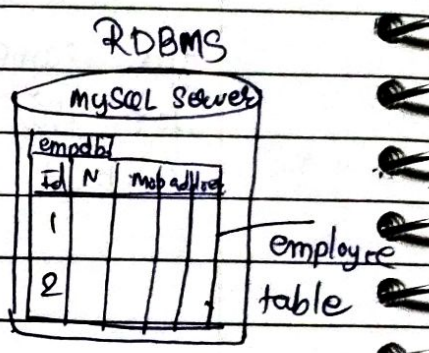
How to Connect java application to database?

* Database ⇒ mySQL. (RDBMs)

MySQL ⇒ It is a RDBMs data structure.

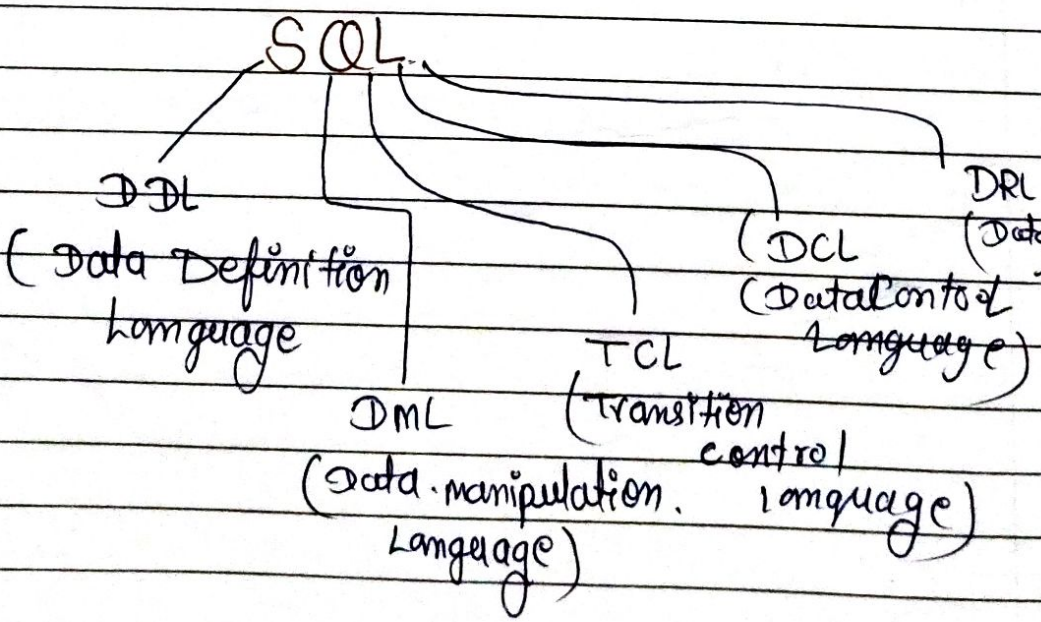
Database ⇒ Collection of data.

"To represent any realworld entity in Database, than we have to create table for that entity."



Database → Table → record | Row | Tuple.

SQL ⇒ Structured Query Language.
[to communicate with RDBMs, to perform any operation on RDMS., used SQL]



(1) DDL :- Data Definition Language.
used to define or modify the structure of database objects (like tables, schemas, indexes etc)

Commands :-

(1) CREATE :- To create Table / structure

(2) ALTER. :- modifies structure of an existing object

(3) DROP. :- Deletes an object (table) from the database

(4) TruncCate. :- Removes all the ~~objects~~ records from a table but keeps the structure

(2) DML :- Data Manipulation Language.
Used to ~~define or modify the structure~~ manipulate data.

Commands :-

(1) INSERT :- Add new data (rows) into a table.

(2) DELETE :- Removes existing records

(3) UPDATE :- Modifies existing data.

(3) TCL :- Transaction Control Language.
used to manage transaction to ensure data integrity

(1) Commit :- Save all the changes made in the transcation

(2) Rollback. :- Undoes changes made in the current transcation.

when to handle transcation, when we do update, Insert, delete operation on records.

APP

insert select upadte delete
create retrieval
CRUD → most using operation.

USER :- root
Password :- ....
Port :- 3306

(4) DCL :- Data Control Language.
when granting the permission, or revoke the permission.

Commands

(1) grant :- Grant the permission.

(2) Revoke :- degrant the permission.

(5) DRL :- Data Retrieval Language
used to query (retrieve) data from the database

Command :-

(1) SELECT :- To select any record.

Command cmd.
→ mySql -u root -p
Enter password:

Database listing :-
Show database
Create Database :- create database empdb;
Drop database :- Drop database empdb;
Using Database :- use empdb;
Create table :- create table employee(
name varchar(100), mobile varchar(10), age int(3),
Salary int(10, 2));

Datatypes (mySQL) :- int, tinyint char, varchar, text, float, date, blob, clob

blob → binary large object

database engine → engine ← InnoDb. / MyISAM.

PAGE NO.:

char vs varchar vs text ⇒ String data.

char	varchar	text
↓	↓	↓
fixed-length String	variable-length String	Long text
(0-255 char)	(0 - 65,535 char)	(upto 65,533)

desc employee; :— Description of employee.
Show tables; :— Show table

Constraints :— Restriction / Rules.
(1) Unique, (2) check, (3) Default, (4) not null,
(5) Primary key (unique + not null). (only one in one table)
(6) unique key (unique + can be null) ~~one~~ (in one table, can be multiples unique key)

(7) Auto_INCREMENT
commonly automatically increases commonly numeric values (int)
(Primary key only).

(8) FOREIGN key :— Relation between two tables.
(9) INDEX

Constraints	Description	Example.
NOT NULL	Prevents Null value	age INT NOT Null.
UNIQUE	No duplicates values	email VARCHAR(50) unique.
PRIMARY KEY	UNIQUE + Not Null	Id INT PRIMARY KEY
FOREIGN KEY	LINKS TABLE	FOREIGN KEY (courseid)
CHECK	validates Condition	check (age>0)
Default.	sets default value	status VARCHAR(10) default 'Active'.

PAGE NO.:

Create table with Constraints :-

* CREATE Table Employee(id int primary key auto-increment, name varchar(100) not null, Salary int(10). default 10000, mobile varchar(10) unique key, age int(3) check (age > 17));

Insert values into Table :-

* Insert into employee (name, salary, mobile, age) values('Atul', 95000, '9117825678', 20));

SELECT Table :-

Select * from employee;

Show create table employee; (To know the description..)

Select id, name, salary from employee; selecting.

[Projection of database :- presenting some specific Column)]

Filter the records :- WHERE Clause

(1) # Select id, name, salary from employee where Salary > 80,000;

(2) ⇒ where salary >= 80,000 and Salary =< 100000;

(3) Select id, name, salary where salary between 80000 and 100000;

Select id, name, salary where age = 28 or age = 30;

Select id, name, salary where age in (28, 30);

where (salary >= 80000 and salary =< 100000 or age = 28;

Select id, name, age, salary from employee where salary is Null;

!= where salary is not Null;

Wild Card Characters :-

% :- Zero or any number of Occurrence.

_ :- Position.

Select id, name, salary, age where name like '%a';
Select id, name, salary, age where name like '%K%';
Select id, name, salary, age where name like '%S__';

Functions :-

Aggregate function :-
It returns single value;
Sum(), count(), max(), min(), avg().

SELECT Count(*) from employee;
select max(salary) from employee;
select min(salary) from employee;
select avg(salary) from employee;

Count cannot count null value

Fetch the employee whose salary is maximum.

Nested query :-

Select id, name salary from employee where Salary = (select max(salary) from employee);

Sorting ⇒ :-

(1) Order By Clause :-

→ asc ; Select * from employee order by Salary asc;

→ desc ; select * from employee order by Salary desc;

~~int~~ (numeric, string)

Limit [5,6,7 ...]; Clause :- Top 5 / record.

→ Select * from employee order by desc limit 5;

→ Select * from employee order by desc limit 3,2 ;

skip

Imp • Second max salary ⇒

① SELECT id, name, salary from employee where salary = (Select max(salary) from employee where salary != (select max (salary) from employee));

• third max salary ⇒

② select id, name, salary from employee where salary = (select max(salary) from employee where salary < (select max(salary) from employee where

Having Clause is used with Aggregate function.

salary < (select max(salary) from employee));

To add columns in existing table.

⇒ Alter table employee add(skill varchar(100));

⇒ Update employee set skill = 'Java' where id in (3,4);

Group By Clause :-

⇒ Select count(*), skill from employee group by skill;

⇒ Select Count(*), skill from employee where age > 25 group by skill;

Select Count(*), skill from employee group by skill having avg(age) > 25;

Having used with aggreate function.

- **Relationship** :- A relationship defines how two or more tables are connected to each other using keys (mainly primary and foreign key).

(1) One To One. (1:1) Relationship.

T_1 — T_2

Ex :- one person has one password

(2) One to Many T_1 —< T_2 (1:N) Relationship

Ex :- one department has many employee

(3) Many to Many T_1 >— <T_2 (M:N) relationship)

Ex :- ~~One department has many employee~~

(1) one to one :-

(1) If the Record of T_1 is associated with only one record of T_2.

(2) one to many

(2) the one record of T_1 is associated with multiple record of T_2.

(3) many to many.

(3) the multiple record of T_1 is associated with multiple record of T_2.

Ex :- one student can enroll in many courses and one course can have many students.

Examples :-

(1) [1:1] Relationship.	[1:N] relationship	[M:N] relationship
1) Each person have one adharcard.	1) one department has many employees.	1) Students can enroll in many courses, courses have many students.
2) Each user has one password	2) one teacher can teach many students.	2) Actors can act in many movies and movies can have many actors
3) Each car has one registration number	3) one author can write many books.	3) An order can include many products and a product can appear in many orders.
4) Each employee is assigned one laptop.	4) one customer can place many orders.	
5) Each ~~password~~ citizen has one password.		

To establish relationship between two tables:-

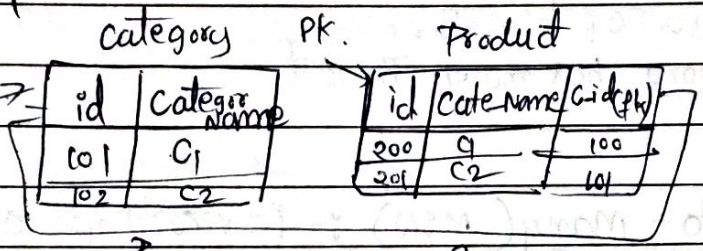
yes ⇒ in a table can, we create multiple foreign keys?

yes ⇒ In a foreign key, column can you insert null in foreign key column

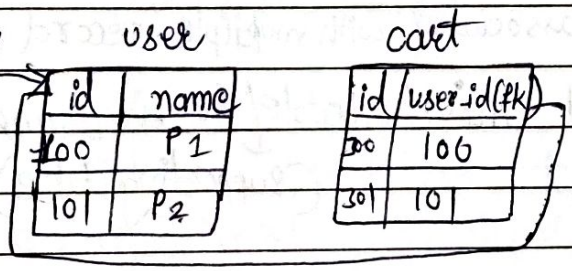
Because null means no data...

In a

- O2M ⇒

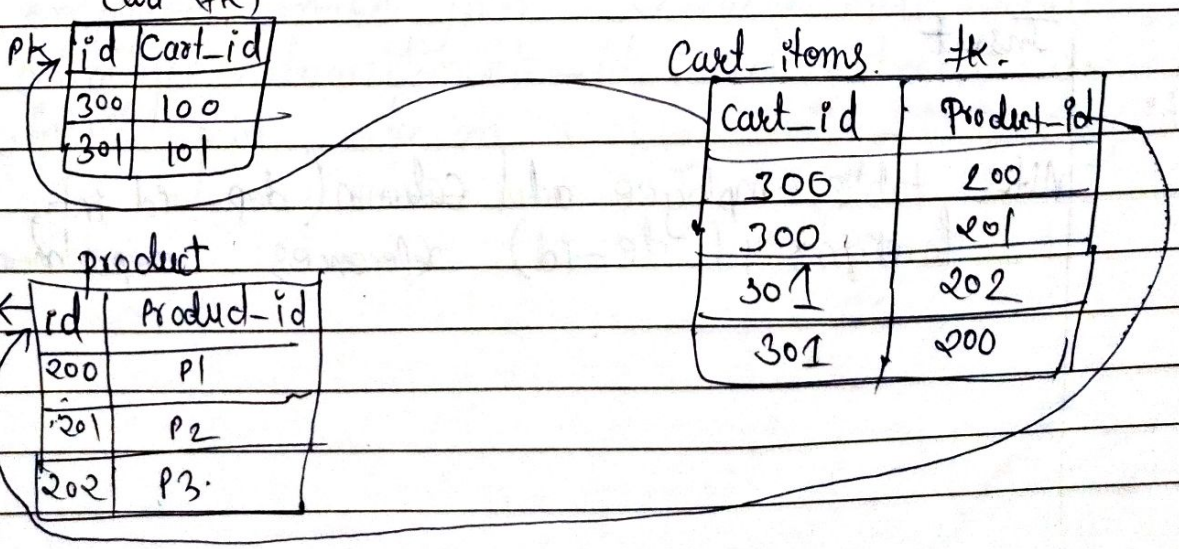


- OTO ⇒



- M2M ⇒

for many-to-many, we have to create the third table.



PAGE NO.:

Relationship / Mapping - Cardinalities / Association.

(1) One-to-one :- (1:1)
If record of Table-A is only associated with exactly one record of Table-B.

eg:- User has one Card.

(2) One-to-many (1:N) :- If record of table-A is associated with multiple record of table-B.
eg:- Category has many Product.

(3) Many-to-many (M:N) :- If record of table-A is associated with multiple record of Table-B. and record of Table-B is also associated with multiple record of Table-A.

Note :- need third table to implement many to many. (Supporting table).

Practical Implementation :-

Create table department (id int primary key auto_increment, department_name varchar(10));

Insert

Alter table employee add column (dep_id int, foreign key (dep_id) references department (id));

PAGE NO.:

Join :- where you want to fetch the data from multiple tables :-

- → inner join → Matching record in Two tables.
- → Left outer join → (Emp) LOJ Dep (matching)
- → Right outer join → Dep ROJ (Emp Dept)
- → self join → [ALIAS] when we compare record of same Table
- → Cross Join | cartisian product (All possible Comparison)

Inner join :-

employee (department_id) department (id),

on :- mysql ko batayega ke, two table ko kaise connect karna

```
Select employee.id, employee.name, department.departmentname
from employee inner join department on
employee.department_id = department.id;
```

Joining three Tables :-

```
Select employee.id, employee.name, department.
department_name, city.city_name from
employee inner join department on
employee.department_id = department.id
inner join city on employee.city_id = city.id;
```

Truncate ⇒ Reset Auto-Increment
Delete ⇒ It will not reset auto-Increment.

PAGE NO.:

CROSS join :-

Select * from employee, department ;

Self join :-

⇒ when we Compare same record of same table.

Select u1.name as name, u2.name as friend-name from users as u2 inner join user as u1.id = u2.friend_id;

Delete Query :-

Delete from employee where id = 11 ;

Update Query :-

Update employee set salary = 89000, mobile = '9826246327' where id = 10;

View :-

Create view Employee-Details as join-Query ;
But write join query atleast one time.