Index - SQL

topic + practise syntax

sdet →

- DBMS, RDBMS
- Create, Insert, Retrive data
- Filter Rows (where clause)
- DDL
- SQL Functions
- Group By Having & order By
- Union & Union All
- SQL Joins & Sub-Query
- Integrity Constraints
- SQL views & Index
- TCL
- JDBC, ODBC, CLT

Naveen

- create / Insert / select
- Order By & AND / OR / NOT
- Like / is Null / is NOT Null
- Joins

SQL Tutorial

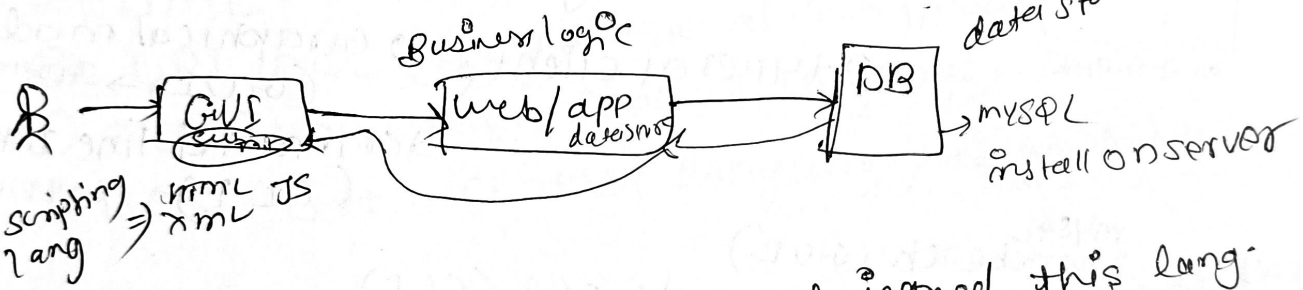
on file cannot perform operation then excel also only some sort of operation can be done.

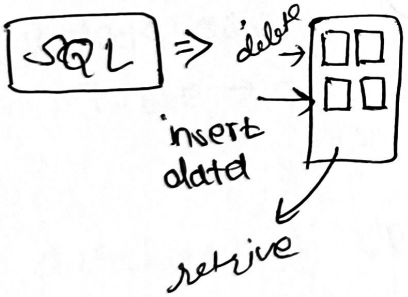
DBMS ⇒ data will be stored in the form of tables.
table means having rows & colms

Dbase, FoxPro, Ms-access etc.

RDBMS ⇒ Relational DBMS. → oracle, MS-SQL server, mySQL, DB2, ms-access
→ we can retrive data easily. } supporting RDBMS features
→ can store huge data.

DBMS ⇒ data storage area



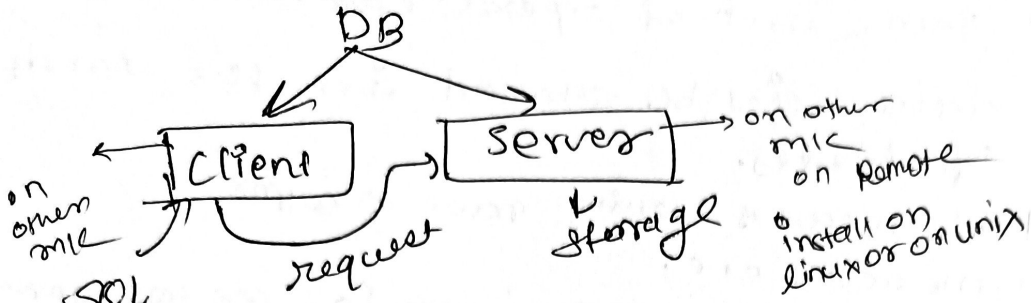


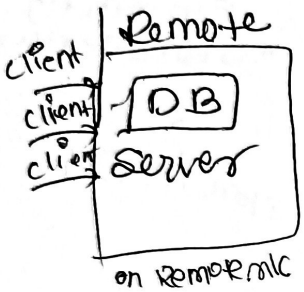
SQL ⇒ ANSI designed this lang.
⇒ same for all ~~language~~ DB
⇒ It is language, which ~~is used for~~ we by can communicate with DB, to perform this operation.

Structured query language → used to communicate to the database to perform different kinds of operations.

Database components

① Client ② Server





Server ⇒ Data is actually stored in DB Server (Remote m/c)

Client ⇒ we can send SQL commands using client s/w.

2 types of client
- Graphical mode (GUI) → SQL Developer
- command line Interface (CLI) → SQL PLUS

MYSQL ⇒ mysql workbench (GUI)
mysql ~~.exe~~ command line tool (CLI)

Install MYSQL

① go to mysql official website
② MYSQL Installer
MYSQL communicate server
https://dev.mysql.com

MYSQL Workbench ⇒ GUI client
MYSQL command line ⇒ CLI client

→ database → connect to db.

→ every database is having port No.

SQL - structured query language

Database → it contains lot of object
It contains lot of schemas.

Schema → It is subfolder, also stored db object there.
for some s/w schema it is considered as a table.

SQL Language ⇒

① DDL (Data Definition Language) - db designer
② DML (Data Manipulation Language) - developer
③ DRL/DQL (Data Retrieval lang/Data Query lang)
④ TCL (Transaction Control language) - developer
⑤ DCL (Data control language) → administrator and

① database
Table → It is object, main source, where data will stored.

database ⇒ CREATE DATABASE mydb; # create.
DROP DATABASE mydb; # delete

OR

CREATE SCHEMA mydb;
DROP SCHEMA mydb;

CREATE DATABASE IF NOT EXISTS mydb3

create table <Table Name> (col1 datatype,
col2 datatype, col3 datatype...)

eg. use mydb;
Create table STUDENT (SNO INT(5), SNAME VARCHAR(15), MARKS INT(3));

metadata ⇒ data about data.

DESCRIB STUDENT # description of table

INSERT INTO STUDENT VALUES (101, 'Karan', 80);
SELECT * from STUDENT;
pass value as per defined

INSERT INTO STUDENT (SNAME, SNO, MARKS)
VALUES ('RAM', 102, 60); # can insert value as per our self

INSERT INTO STUDENT (103, 'Ram', NULL);
insert NULL when don't know data.

① Created database
② created Table Student
③ Inserted data in a Student table.
④ Select / retrieve data.

SQL data types

1. Numeric
2. text
3. Date/time

with the help of SQL we can interact with db.
SQL is common/standard language to interact with db.

SQL is very imp to know
SQL used for SQL based db
SQL is applicable for all SQL based db.
eg. mySQL, oracle, SQL server, db2, Access

1. select ⇒ Retrive the data
2. Insert ⇒ create the data
3. Update
4. Delete

jdoodle.com ⇒ for practice SQL query

```
⇒ create table Employee
  {
    EmpID varchar(255),
    EmpName varchar(255),
    Age int,
    PhoneNumber int,
    EmailID varchar(255),
    Address varchar(255)
};
insert into Employee value(1,'Rohini',25,9096,ro@g.com,LA)
select * from Employe; # return all row)
```

sdet

SQL commands

① DDL (Data Definition language)
- CREATE, ALTER, DROP, TRUNCATE, RENAME

② DML (Data manipulation language)
- Insert, Update, Delete

③ DRL/DQL (Data Retrieval / Data Query language)
- Select

④ TCL (transaction control language)
- Commit, Rollback, Save point

⑤ DCL (data control language)
- Grant, Revoke

Where clause ⇒ filter data (conditional based retrieval of data)
filtering records based on conditions (filtering rows using where condition)

Select * from Employee; # get all employee from table

Select * from Employee where SALARY <= 5000;

to check NULL colm are need to use is operator
↑ or is not

Select * from Employee where Email is null;

distinct ⇒ It is used to retrieve unique records from a table)
avoid duplication.

SELECT DISTINCT * from EMPLOYEES;

Arithmetic operators ⇒ >, <, <=, >=

Logical Operators (AND, OR, NOT)

Use hr;

SET Name=0;
Select * from employee; # display all colm

Select * from employees

AND => If both condn match then only that row will be displayed.

OR => one of condn is match then only the row will be displayed.

NOT => if not is there then that will not display in table.

select * from employees where salary > 15000
AND Job_ID = 'AD_VP';

select * from employees where salary > 15000 OR Job_ID = "key";
select * from employees where NOT first_Name = "David";

Between & IN operator

① Between => used to display the rows which is following in the range of values.

② IN => In operator returns the rows when the values are matching in the list.

select * from table where salary BETWEEN 1000 and 12000;
select * from table where salary NOT BETWEEN 100 and 200;
Select * from table where salary = 340 or salary = 500 OR salary = 480;
select * from table where salary IN (3400, 2000, 3000);
Select * from table where salary NOT IN (8400, 9500, 1000);

If you want to extract more no. of rows then use IN

Pattern matching operators (wild card characters)
(Regular Expression)

% ⟶ many characters

_ ⟶ single character

select first_Name from table where first_Name like 'S%';

Select first Name which is start S from s
% ⇒ after s there can be many character.

select first_Name from table where first_Name like '%s';

select first name where s is last character
before s can be any character.
(end character is s)

%m% ⇒ many character m many character
m will be in between, before m or after m can be anything.

Not like 'S%' ⇒ # all the name whose character will not start with S character

'%e_' ⇒ many character e, after e there will be only one character.

'___' ⇒ retrieve firstname with 3 character only.

Reg exp. use only with like operator

Built in functions in MySQL

In function, we can pass some parameter, it will return some response/o/p.

MYSQL functions

① String functions - operate on string data types
② Numeric functions - operate on numeric data types
③ Date functions - operate on date data types
④ Aggregate functions - operate on all of the data types and produce summarized result sets.

① String functions ⇒

upper() → converts into upper case letters.

```
SELECT UPPER (first_name) from employees;
```

convert any character into upper case character

```
SELECT UPPER ('smith');   # SMITH
```

Lower() → converts into lower case letters.

```
SELECT LOWER (first_name) from employees;
```

Length() → return the length of string.

```
SELECT LENGTH ('oracle');
SELECT * from EMPLOYEES where length(first_name)=4;
```

Trim() → Removes the specified characters from both sides

```
Select Trim(' Oracle ') from DUAL;   # Oracle
SELECT TRIM('z' from 'zzoraclez') from dual;   # Oracle
```

trim from both left & right side

INSTR() → Returns the position of the character within a string. ⇒ `SELECT INSTR('ORACLE','E');`

SELECT INSTR('WELCOME', 'O')
↑ pair of this?

5

Steing functions ⇒

① SUBSTR()/SUBSTRING() ⇒
Returns the substeing of the steing
SELECT SUBSTR('ORACLE', 2, 3) # RAC
↑ 3 character

② concat() →
To Join two steings.
SELECT CONCAT('ORACLE', 'TRAINING');
ORACLETRAINING
Select concat(first_Name, last_name) from table;

Numeric function ⇒

① Select ABS(-40); # return absolute value
② SQRT(25) # return sq root
③ select MOD(10,3) # 1 return remainder
④ select POWER(2,5); #32
TRUNCATE() → function truncates a number to the specified no. of decimal places.

Select TRUNCATE(40.1234,3) # 40.123
Select TRUNCATE(40.1234,2) # 40.12
Select TRUNCATE(6876, -1) # 6870 (last digit become zero)
Select TRUNCATE(6876, -2) # 6800

- greatest() & least() ⇒ returns greatest, least values in the provided values.

Select greatest (100, 200, 500); # 500

Select least (100, 5, 3); # 3

SQL Update

Update statement is used to modify the existing records in a table.

```
UPDATE tablename
Set colm1 = value1, colm2 = value2, ....
where condition;
```

DDL Command

1. CREATE
2. ALTER
3. DROP
4. TRUNCATE
5. RENAME

1. **Create** ⇒ is used to create database objects [Database, Table, Views, synonymes etc)

ALTER ⇒ 1. Adding a new column
2. Dropping the existing column
3. modifying the existing column [increase/decrease size of the column & change the data type of colm]
4. Renaming colm

```
Use mydb;#
Drop Table Student; # drop complete table data with structure

create table student
(SID int(4), SNAME varchar(15)); # creat table
Describe student # display created table
Insert into student values (101, 'David'); # insert
insert into student values (102, 'Smith');   data into
insert into student values (103, 'Scott');   table
commit;
select * from student; # retrieve data
```

② ALTER ⇒ At table + colm level

ALTER TABLE Student Add(Grade varchar(2));

add colm grade.

describe student;

We update cmd to add data in colm it is used in definath level

ALTER table Student DROP COLUMN Grade;

delete existing colm

Update

Alter table student modify column sname varchar(20);

increase size of colm (update it) but if you want to decrease the size of colm you need remove data first.

modify existing colm

Alter table student Rename column Sname TO STNAME;

Rename the colm

③ DROP ⇒ ① used to remove table/db
② completely table is gone with data definim

ddl ↗

Drop table student # complete table will delete data along with structure.

④ TRUNCATE ⇒ ① data is deleted but table is there
ddl
table structure is available but data is present.
② if you deleted data, you cannot rollback the data

⑤ Delete ⇒ ① remove only data but still table structure is there.
DML
② But in delete command, data deleted temporarily.
Work on only data not on definition level
③ data can be rollback.

set autocommit = 0; # enable this feature
Commit ⇒ permit action
autocommit by default value is 1.
so, whenever perform any action permanently save in database

while inserting some data in table.
Commit; # set permanently store होता db में
must execute commit cmd to make permanent changes in db.

DELETE from student;

set autocommit=0;
set SQL_SAFE_UPDATES = 0; # only then allows us to update the table

permanently perform operation → commit;
select * from student;
delete from student;
Rollback; # when delete the data there is chance to get back the data by using Rollback

Rollback works with DML commands (insert, delete, update)

In DDL, we can't rollback

commit ⇒ permanently do changes in db.

TRUNCATE ⇒

```
Insert into ~~table~~ Student values (101, 'smith');
# insert data into Student table
Commit ; # permanantly store in table
truncate table student ; # delete data permenently
                           can't ~~remove~~ rollback
                           using Rollback.
```

Rename a table

```
USE HR;
Rename table jobs To designations ;  # table structure remain same just rename the table name.
```

Union operator

① Set operators → union, union All, Intersect, minus
use to join ~~~~ multiple data or retrieve/extract data from multiple table.

Union ⇒ ① remove duplicate data

union All ⇒ ① will give you duplicate data as well.

Intersect ⇒ ① will give common data from both table.

minus ⇒ ① give you all data from first table which is not in 2nd table

① UNION ⇒

→ The union operator is used to combine the result-set of two or more SELECT statements.
→ each SELECT statement within UNION must have the same number of columns.
→ The columns must also have similar data types.
→ The column in each select statement must also be in the same order. (colm name)

select * from A
select * from B
select Num from A union select num from B;
select num from A union All Select num from B;
satisfied all condn

SQL Joins

→ Joins help retrieving data from two or more database table.

→ The tables are mutually related using primary & foreign keys.

→ types of Joins

① Equi Join / Inner Join / Simple Join
② Right Join
③ left Join
④ full Join
⑤ Self Join

if you want to establish some connectn bt? 2 or more table so we should have one comm which is common in more than one table so that we can do joins / connect those tables

① Inner Join ⇒ Retrieve the common data from both the tables

(*) Retrieve common data btn two tables

```
Select * from tab1 inner join tab2
on tab1.numid = tab2.numid # 11, 12
```

tab1

numID
12
14
10
11

tab2

NumID
13
15
11
12

got only matched records

Left outer join ⇒ give datas should be in left table no that datas should not be right (and) table

```
select * from tab1 left outer join tab2
on tab1.numid = tab2.numid # [10,14,11,12]
```

returns matched records + unmatched from ~~right~~ left table tab1

Right outer join ⇒ (from both table) give datas from right table only

```
select * from tab1 right outer join tab2
on tab1.numid = tab2.numid #[11,12,13,15]
```

returns matched records + unmatched from ~~left~~ right table

Full outer join ⇒

matched Records from both table + unmatched record from left table + unmatched record from right table

self join ⇒

* Retrieve datas from same table
join with a table with the same table.

Query ⇒ print Employees details whom is manager of other employees.

```
select E.Employee_ID, M.manager_ID, E.First_Name,
E.JOB_ID, M.First_Name from Employees E, Alias
Employees M where E.Employee_ID = M.manager_ID;
   Alias
```

eid	Nam	mid
101	–	103
102	–	–
103	–	–

E.id = M.id

eid E — M mid → Alias of E

① Group By - each & every

→ The group by clause groups records into summary rows.

→ Group By returns one records for each group.

→ Group by typically also involves aggregates :- COUNT, MAX, SUM, AVG etc.

→ Group By can group by one or more columns

Select Department_ID, SUM(Salary) from Employees Group By Department_ID; # display dept ID & sum of Salary from employee table where dept ID में जो same होगा वो group में जाएगा।

Select Department_ID, AVG(salary) from Employee Group by Department_ID;

display dept ID & avg of sal of dept ID display for each & every how much avg salary is? grouped them by using department ID

↓

select Department_ID, MAX(salary), min(salary) from Employees Group By Department_ID;

display dept ID, max salary, min salary from employ table which will group dept ID

↓ इसे भी group by apply होगा

Select JOB_ID, count(*) from Employees Group By JobID;

↑ how many

display JobID, each & every Job ID how many employes are working

* apply group by on multiple columns
group result based on 2 colm

```
select Job_ID, Department_ID, count(*)
from Employees group by Job_ID;
```

Specific Job ID how many employees are working in each job ID
& same for dept ID

group by ⇒ grouping results from table.

Having ⇒ apply filter on results which is obtain by group by clause.
↙
apply on group result for filter

```
select Job_ID, count(*) from employees
Group By Job_ID Having count(*) > 20;
```

Having clause is used to filter the output from the group by clause.

```
Select Department_ID, sum(salary) from employees
Group By Department_ID Having Sum(Salary) > 20000;
```

where ⇒ Group by ⟶ having

```
select department_ID, sum(salary) from employees
where Department_ID <> 50 Group by department_ID
having sum(salary) > 10000;
```

Order By ⇒ show Asc, desc order keyword
by default, Ascending order

select * from Employees order by salary desc;

Order of Execution
where → Groupby → Having → Order By

```
select   column_names
    from  table-name
      where condition
        Group By column-names
          Having condition
            order By column names
```

Select department_ID, sum(Salary) from employees
Group by department_ID Having sum(salary) > 20000
order by sum(salary) desc;

Subquery

→ Subquery is a query within a query.

→ Sub query contains 2 parts.

1) Outer Query
2) Inner Query

→ The output of inner query is become input of outer query.

→ 2 types of Sub queries :-

1) Single row sub query , <=, >=, !=
2) multi row sub query , IN, ANY, All

inner query output will become input for outer query.

```
select salary from employees where salary <
    (select salary from employee where first_name='Ali');
```

2nd max salary from employee

```
select max(salary) from employees where
    Salary < (select max(salary) from Employee);
```

Multiple row subquery ⇒ if the inner query row returns multiple value then it is multiple row subquery.

If the inner query row returns single value then it is Single row subquery.

Integrity Constraints => condn which will apply on colm.

Single Row Sub Queries

Find the salary of employees whose salary is greater than the salary of employee whose Employee_ID 150

select salary from employee where salary > (select salary from employee where employee_ID = 150);

display the employees who all are earning the highest salary.

select * from Employee where salary = (select max(salary) from Employee);

multiple row subquery - (IN, ANY, ALL)

Display salary whose salary is equal to the salary of the at least one employee in department id 30.

Select * from employee where salary (IN) → compare with all OR & print (select salary from employees where department_ID=30));

return none rows

In ⇔ <Any ⇒ less than any of the salary

>Any ⇒ greater than all the employee salary where ID = 30

<All ⇒ all of the values

>All ⇒

Auto Increment

It is a function that operates on numeric data types. It automatically generates sequential numeric values every time that a record is inserted into a table for a field defined as auto increment.

```
Create table student
(sno INTC5) primary key AUTO_INCREMENTS;
  sname varchar(15), marks int (5));
```

by default = 1 (val)

starting value

```
ALTER TABLE student AUTO_INCREMENT = 100;
# if data delete & if eg. 2 then insert from 3 if you want insert from 2 only the above line
insert into student (sname, marks) values ('x',60);
insert into student (sname, marks) values ('y',45);
insert into student (sname, marks) values ('z',105);

Select * from student;
delete from student where sno=3;
```

Limit ⇒

```
Select * from employees limit 5;
# retrieve limited rows from table
Select * from employees limit 5,10;
          # start from 5th row to 10th row  (display)
```

Views and Index

→ A View is a virtual table based on the result-set of an SQL statement.

→ A view contains rows and columns, just like a real table. The fields in a view are fields ~~are~~ from one or more real tables in a database.

→ You can add SQL functions, WHERE and JOIN statements to a view & present the data as if the data were coming from one single data.

→ view is a logical object not having physical structure.

→ result store in view & give access to other user

→ view have logical data (used security purpose)

```
User hr;
Select * from Employees;
```

creating or view

```
CREATE VIEW Employees_VI AS select Employee_ID, first_name,
salary from EMPLOYEES;

Select * from Employees_VI;
```

dropping view

```
DROP VIEW EMPLOYEES_VI;
```

Index ⇒

select * from employees;

Indexes are used to retrieve data from the database very fast.

The users cannot see the indexes, they are just used to speed up searches | queries.

creating Index ⇒

CREATE INDEX idx_employees ON Employees (First_Name);

dropping index ⇒

drop index idx_employees on employees;

TCL – commit & Rollback

Commit⇒

DML → insert, update, delete } transaction which is not permanent → on → db

→ bydefault it will become permanent

but above write commit after cmd then permanently perform that transaction

Autocommit = 1

Rollback⇒ once commit delete then perform Rollback still will not ~~get~~ Rollback

```
SET autocommit = 0;
use mydb;
drop ~~student~~ table student;
create table student (sid int(3), sname varchar(15));
insert into student values(101, 'abc');
insert into student values(102, 'xyz');
delete from student where sid=103; # delete records jaspon jaisy
select * from student;
Rollback; # Rollback record
Select * from student;
Commit; # commited delete
Rollback;
Select * from student; # cannot get deleted record after commit
```

Database Testing

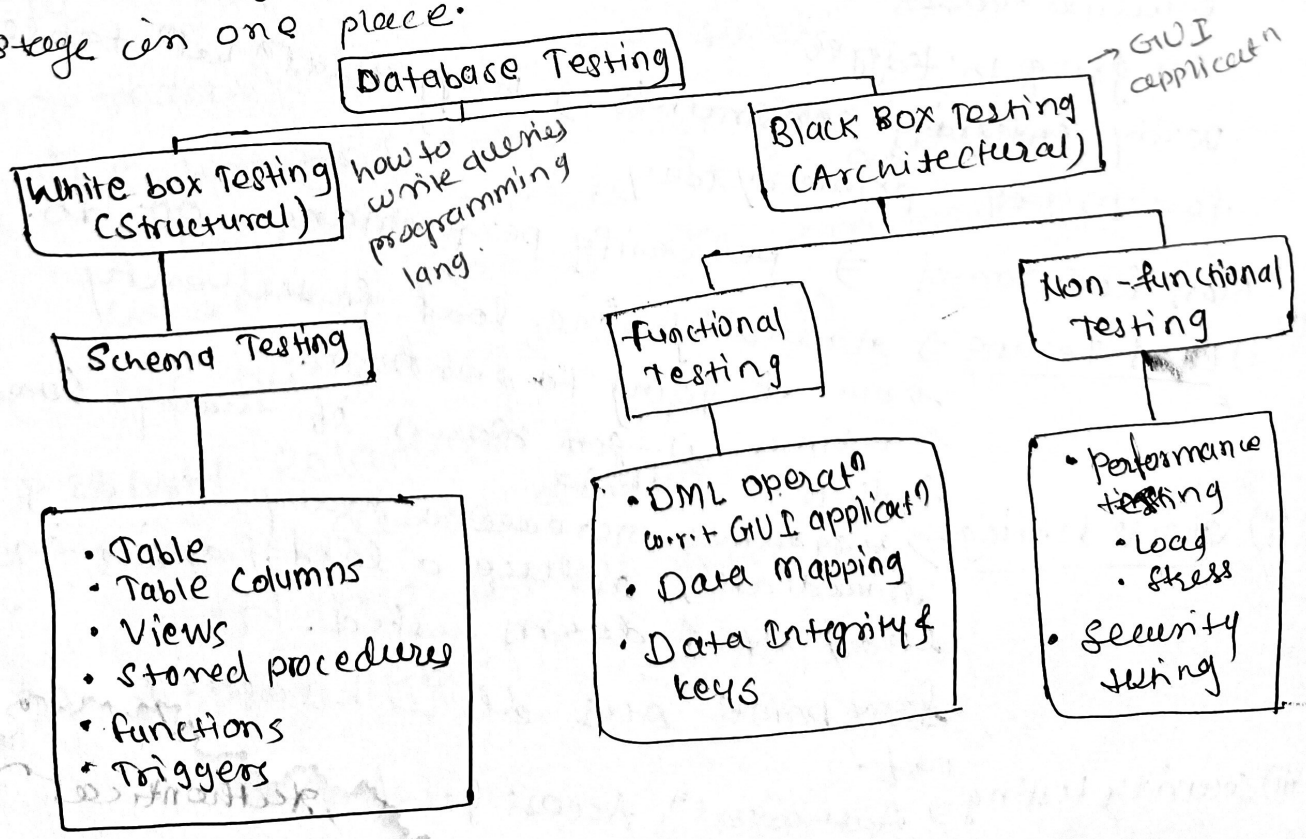
UI testing mainly focus on User interface according to customer requirement or not.

DB Testing mainly focus on db components like data and database objects like table, views, Indexes & so on.

Database Testing → db objects, db schemas & data
↳ It contains tables, views, stored procedures, functn properly working or not, white box testing, black box testing.

extract data from diff. geographical places, from diff servers, from diff. db, extract it apply some transform rules & push data in single server basically called data warehous.

ETL testing perform on data warehouse from
↳ data extracted from diff. server, from diff db, from diff. geographical locatn or OLTP systems stage in one place.



① DML operatn w.r.t. GUI Applicatn ⇒
function testing means test behaviour of a database
test DML operatn, data manipulation, selection, insertion,
update, delete. this type of operations are properly working
or not.
inserting customer data, manupulating data, updating
data or delete data
we do various operatn from GUI & we will test
same operatn reflecting on db objects or not

Data mapping ⇒ whatever data we are sending from
client application the same data is storing in proper
related column or not.

Data Integrity & keys ⇒
there is so many tables or objects are there. as
so now the objects are internally communicating
send some data from applicatn there on multiple tables
when we retrive the data from db coming from
multiple tables internally stored procedures, functn
are going to trigger
verify integrity constraints ⇒ verify relatn betn tables,
foreign key, primary key.

Non functional ⇒ per verify performance of db.
i) Load Testing ⇒ gradually inc load & verify
how its going to perform.
sometimes it gets down if heavy load
is there. db is
ii) stress testing ⇒ sometimes increase a heavy loads &
immediately reduce a load, again inc
load ups & down tested.
some points our db is breaking or
not.
iii) Security testing ⇒ Authorizatn, Access level, Authenticat

Data mapping Testing [CRUD]

⇒ we will perform this operation from front end application & we will see the data is exactly reflecting corresponding tables in db or not.

⇒ focused on → ① Data Existence - data exist or not (check)
② Data Correctness - data is correct or not
③ Data Completeness - data is completely stored in db or not

⇒ we should know the functionality of the application we will try to update, delete, create or retrive the data. through UI

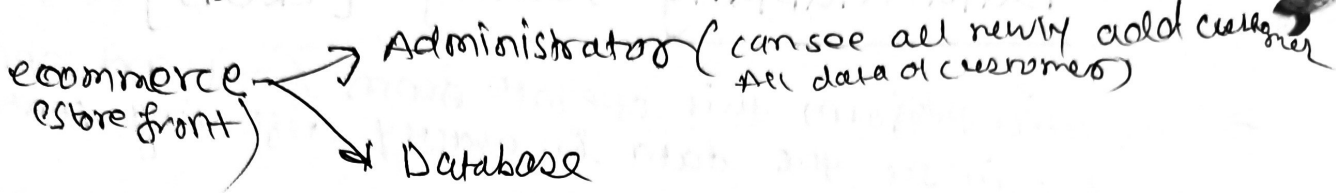
⇒ gray box testing ⇒ UI + Backend

⇒ mainly focus on data which is inserting, updating or deleting from front end some data reflecting in db table or not.

⇒ prerequisite ⇒ ① what are table present in db
② Submiting data from front end then on which table it is reflecting in db.
what are tables are impacting.
③ for this we need to refer ER model
Entity Relationship model.
Database design document.

for every field for all there should be store in mapping table.

⇒ according we will write test cases



Test cases for Data mapping testing ⇒ CRUD

① As per my I have performed DB testing for data mapping.

Database validator can do manually by executing SQL command.

Data Integrity testing

mainly focus on integrity constraints.

⇒ Basically SQL constraints are specified a rules a data in a table.

⇒ Normally database contains multiple tables table contains multiple columns. we can set multiple no. of rules on those colm. & those rules called as a Integrity constraints.

⇒ when you are trying to inserting data into table then those constraint going with check whether it is properly inserting data into table or not.

⇒ basically applying some rules on the table or table colm basically called as integrity constraints

⇒ mainly focus on testing constraint of a table. or constraints of colm.

SQL constraints are used to specify rules for data in a table.

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.

SQL constraints

① NOT NULL ⇒ (constraints on table)
Ensure that a column cannot have a NULL value.

② UNIQUE ⇒
Ensures that all values in a col^m are different.

③ Primary key ⇒
A combination of a NOT NULL and UNIQUE.
uniquely identified each row in a table.

④ foreign key ⇒ can link multiple tables [if you create col^m as FK then that particular col^m referring to some other col^m / table]
uniquely identifies a row/record in another table (establish rel^n betn two tables)

⑤ check ⇒ apply some validat^n on col^m. when inserting data, it'll then check constraint will check data is valid or not according to table
Ensures that all values in a col^m satisfies a specific cond^n.
if valid it will update data. if not then throw except^n

⑥ Default ⇒
Sets a default value for a col^m when no value is specified.

Constraints validat^n for each & every table once it is created then we validated whether those constraint property triggered or not by doing updat^n, insert^n & deletion operat^n.

* tables, rel^n bet^n them, we should know data modal

① first of all we need to understand tables available in db then constraint available.

② then write test cases for constraints that properly working or not by +ve, -ve testing

we need to write SQL queries.

Insert query ને validate કરવાનું
mysql માં run કરવાનું |

we need to test relat^n bet^n two tables.

③ when testing child table ⇒ then its ID should be available in parent table then only insert can occur.

deletion ⇒ જ્યારે child table અને parent table માં record હોય
રહ્યા છે તો parent table ના record [illegible] child table
ના delete કરી [illegible]
ને record delete કરી શકાય |

whenever you will delete record from child table then only you can delete record from parent table otherwise you can't.

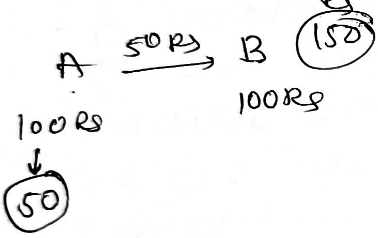
if u use ⇒ on delete cascade ⇒ then you can delete parent table record along with child table record

ACID properties testing - mainly focus on transaction.

when ↓
whenever you applicatⁿ perform some operatⁿ/ betⁿ source & target (specially banking applicatⁿ) transatn

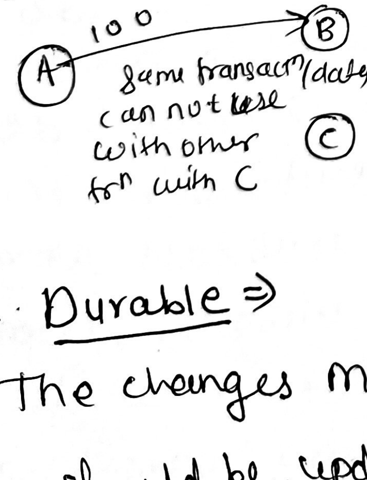
① Atomic ⇒
All changes to the data must be performed successfully or not at all.
But it should not perform the partial transaction.
But it should not perform the partial transaction or execute partially
→ The tr should not break in betⁿ or execute partially
→ reffered as "All or nothing" rule.

② Consistent ⇒
data must be in a consistent state before & after the transaction. the data should always be correct. any change is made in db it should reserved always.



③ Isolated ⇒
No other process can change the data while the transaction

if there are 2 account A & B then if A is doing tran with B then A cannot perform transactⁿ with C until complete the transactⁿ with B.



④ - Durable ⇒
The changes made by a transaction must persist.

tr should be properly updated in db. table. success → changes occur in db

A → B

because of some operatⁿ failure system get shut down betⁿ tr then data should not be loss it should go back & recover system

Isolation ⇒

→ In most of time, we do concurrent transactⁿ / ‖el for we initiate multiple transactⁿ at a same time.

1st tr होने के साथ ही दूसरा tr होना चाहिए

→ one tr should not impact on any other tr

→ Independent executⁿ

Durability ⇒ Recovery testing

Ensures that the data is not lost in case of a system failure or restart & is present in the same state as it was before the system failure or restart.

What is stored procedure and advantages of stored procedure?

- Its block of SQL statements, if we want to execute same query multiple times (again & again)

eg. Select * from customers ; # return data from table

hit the db & get the data.

wrap multiple/single SQL statements. it saves as file. & whenever you want to invoke those statements, we just call the stored procedure. Stored procedure executes that block of statements.

- A stored procedure is block of SQL statements
- we can save stored procedure & can be reuse multiple times.
- we can also pass parameters to a stored procedure.

Advantage ⇒

① Reduce N/w traffic ⇒

whenever you run multiple queries from application against db.

instead of executing complex queries betn Applicatn & mysql db, simply create stored procedure & invoke stored procedure from the applicatn & that stored procedure will talk to the db.

So it Reduce N/w traffic betn applicatn & mysql db.

stored procedures help reduce the N/w traffic betn application & MySQL Server. Because instead of sending multiple lengthy SQL statements, applicatn have to send only the name & parameters of stored procedures.

② centralize business logic in the database:-
Sometimes we implement a logic in stored procedure then call it from applicatn. We can reuse this business logic in multiple bases in our application.
We can use the stored procedures to implement business logic that is reusable by multiple applications.
The stored procedures help reduce the efforts of duplicating the same logic in many applications & make your database more consistent.

③ make database more secure ⇒
If your applicatn directly hit the db then that is not secure..
Stored procedure internally execute SQL queries & talk to db objects.
The database administrator can grant appropriate privileges to applicatn that only access specific stored procedures without giving any privileges on the underlying tables.
As a user we just invoke the stored procedure & then we will get resource.
All db level manage.

How can we create stored procedure?

Sometimes it takes some parameters or may not take any parameters or takes single/multiple parameters. sometimes it returns some data that is stored in o/p parameters.

Stored procedures syntax can be vary from db to db.

mysql syntax ⇒

How to implement in db?

→ open SQL editor

syntax ⇒ create our own delimeter → ##, $$, //

```
Begin
;  ← sql statement
end
```

It is considered as seperate statement

Stored proc # difinition

```
Begin
---;
---
---
End //
```

execute entire statement in one

① set delimeter

```
delimiter //
create procedure select AllCustomers ()   (In mycity varchar(s))
                                  ↑ no argument
Begin
select * from customers where city = mycity;
                # also can write query can retrieve
                  data from multiple table
End //
delimeter ;
```

How to call/invoke stored procedure?

```
call selectAllCustomers ();  ('singapore')
```

can call this multiple times in multiple place

SQL is not case sensitive

2nd Stored procedure - take 2 i/p parameter

```
delimeter //
create procedure selectAllCustomerByCityAndPin(In mycity varchar(50), In pcode varchar(50))
Begin
    select * from customers where city = mycity and
                postalcode = pcode;
end //
delimeter ;

call selectAllCustomerByCityAndPin ('singapore','440008');
```

(3) Accept customer No. returns total No. of orders were shipped, canceld, resolved or dispute

```
delimeter //
create procedure get_order_by_cust(
```