07/09/25

Training @ RKIT

PAGE NO.:
DATE :

* Week 1: MySQL:-

→ current version 8.0.43

→ Connect with tcp/ip Port NO: 3306

→ On installation wizard have downloaded:
Custom installation → server
workbench
examples &
documentation

→ samples & examples were loaded for the further training (HELPFUL)

→ Add Path var for mysql bin to Path system variable so it can be Accessed by any location.

→ check version:- mysql --version
↳ to show the version.

Login:- mysql -u [username] -p
↓
will Ask the password: ******

→ to see avilable dbs: show databases;
query → server
cmd-line ↵

source filename.sql
→ to load the sample database.

→ Sql:-

→ mysql workbench:-

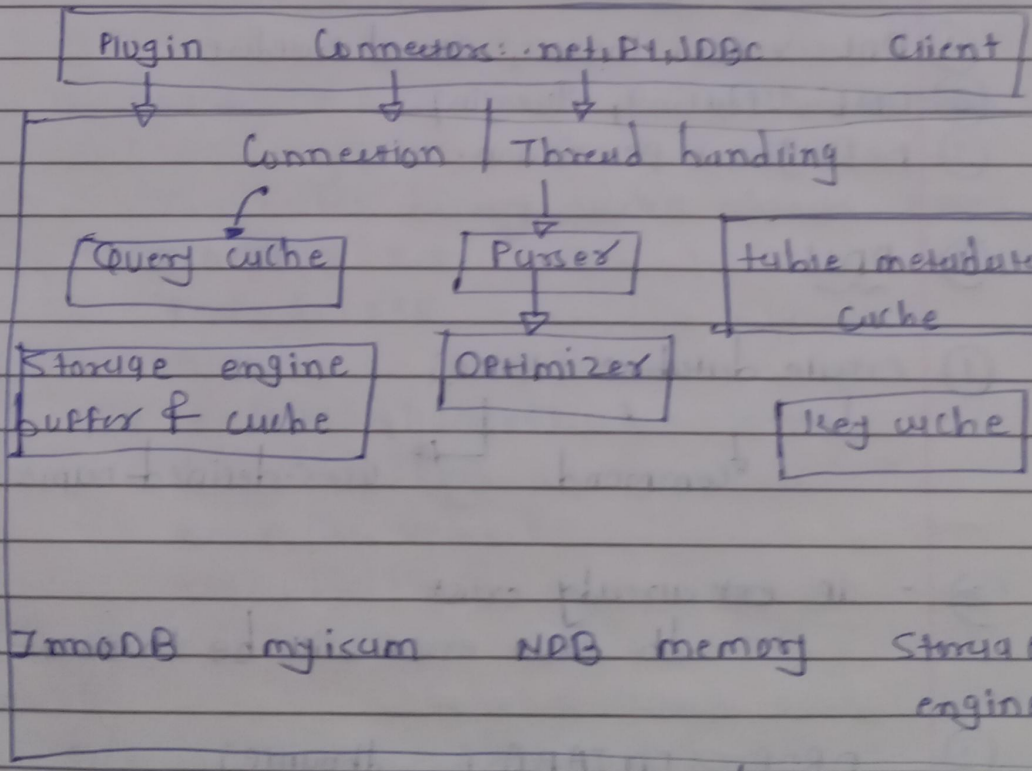
→ what is mysql:-

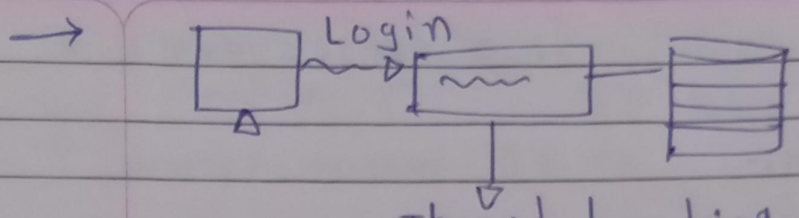
→ open-source Relation database
dev begin in 1994 by swedish devs.

→

Sun micro sys. Acquired mysql AB

Architecture:-





Thread handling.

{All queries will be handled}

Parser

Optimizer → caching hit or miss

{Success Optimizing
Show Result}

→ table metadata cache:-

→ SQL includes:-

1. DDL: views, triggers, Stored Procedure
2. DML: Updating, Querying
3. DCL: grant permission.

1. DDL:-

1. create database Name;

Command — user-defined name

2. - if already exist
db exist error will be thrown.

2. DROP DATABASE dbname;

3. use db;

Float (size, d): d no after point
Float (P): 0-24 float above → double
Decimal (size, d)

- Date:

Date: 1000-01-01 to 9999-12-31
YYYY-MM-DD
Datetime: ↓ + hh:mm:ss
timestamp: No of sec till unix dep epoch
Unix epoch: 1970-01-01 00:00:01

Default-current-timestamp
on-update ——

Year:

→ Create:- As

Create table tbname
as select
field1, field2 ...
from exist table
[where conditions];
{structure + data}

* Drop:-

Drop table tbname;
└> delete data + structure

truncate:
{delete the data but reserved
datatype, constraint schema}

- deallocates Pages
- Resetting indexes
- x Activate delete trigger

* Alter table:-

→ Alter table to Add, delete, modify columns in exist table
Add, drop Constraint

- Add:-

Alter table tb
// add (c1 datatype;
// add (c1, d1, c2 d2, ...)

- Drop

Alter table tb
Drop column / Constraint (c1, c2, c3);

* modify:-

Alter table tb-name
modify column col-name data-type;

* Insert, Update, delete rows:-

→ insert into tb name ()
values ();
↳ inserting null values or default.

→ insert into tbname (col1, col2, col3...)
values (v1, v2, ...);

A tbname (c1, c2...)
Select c1, c2
from tbname
[where condition];

→ Updates

→ Update tbname
set c1=v1, c2=v2
where condition;

* delete:

- delete from tbname
where condition;
↳ entire row will be deleted.

- delete from table;
Remove all rows.

→ mysql workbench Allow graphical
Result grid: Json, form data

* Constraint:-

→ create table tbname
(cname datatype constraint);

→ Constraints:

Not null :
Unique :
Primary key :
Foreign key :
check :
default :
create index :

→ naming a constraint will be useful instead of giving it system generated name.

→ Alter table tbname
Add constraint [name]
unique (columns)

→ modify col datatype Not null;

* Foreign key :-

→ primary key of 1 table will become foreign key for other table.

Constraint name foreign-key (col)
References table (col);

* Select:

- Select what to select
from table
where condition;

AND:
OR:

* Order by:
Col1, col2
Desc or Asc

multiple columns can be done with.

* date function:-

→ SQL inbuilt function
TimestampDiff (format,
start date
end-date)

* extraction or Part or date
month()
date()
year()
dayofmonth()

* String pattern matching:-

→ '%' Any no of char
'_' exact no of char.
Field
Regexp '^exp'
'$' end of string.

→ call:
lock:
explain:

→ Dcl:- Create Alter Permission of users

Grant: allowing Permission

grant [Commands, DML, DDL]
on [database]
To
User;

→ create user [ ]@[network]
identified by 'Passwd';

* Join:-

(1) to querying data from 2 or more tables.

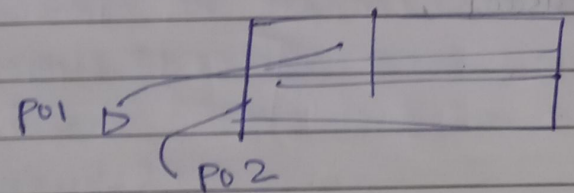
(1) Select columns
from t1
Inner join
t2
on t1.c1 = t2.c1;

→ got columns duplicate which has the c1 column same in table t2;

* outer join:-
—x—

[* insert into table
values (..)
ON duplicate key → primary key / unique
Col1 = val1,
Col2 = val2;] → Upsert in NoSQL.

* Partitioning:-



{divde large table into
smaller partition
row-wise}
easily manageable table
logically one table.
{Parallel Processing.}

Part by types:-

Range
list
hash

key
range column / list columns:-

Create table — ( )
Partition by [type] ( ) (

Partition P-name values [filters on type],
);

→ each table have same Partition strategy.
P-key must include Partioning column.
foreign-key not support

→ drop Partition instead of deletes

- when data evenly distributed
hash will work easy search on columns.

* ctes:-

→ Common table expression: Temp resultset.
- use as temp views / table that can be refered with another query till stat exe.
- used as alternative of subqueries or nested queries.
- named query reusable.

→ [with cte_name As (
// select statt filters
) main query using cte.]

Subquery vs cte vs view

less readable | clean view | permanent
Redundant | temporary | Stored in db.

* Pagination:-

→ Limit Page-to-skip * size, page-size;
x , y

* case if-else like functions:-

— [case
when (condition) Then result
when " Then ",
Else else-result
end As column-name
from tbl;]

* Cascade:-

→ Parent-PK ——— Child-FK
Reaction
ON delete/update cascade; Refer to the child table. (whatever done in Parent)

→ SetNull:- Child ke become null; /delete/update

Restrict:- Need to Remove Rleutionchip manually needs to be deleted child ~~row~~ Row first;

* nested queries:-

⇒ query1 (query2 (...))

used when
- Where
- From
- In
- between (q1) And (q2)
- Select.

[ ] {inner-query first}

Result computed. Returned to outer query.

* Correlated Subquery:-

→ Select outer-query
Where (
qery based on field of outer query
)

Any, All
In, exsits
From
Where

Correlated Subqueries

Stu_id
other author_id Not IN C
select Dist id NOT NULL)

- nested | correlated ✓

* Set operation:-

→ RESULT1 [operation] RESULT2

→ Union:
Removes duplicates.
Union All:
Join Result keep dup.

* trigger:-

-> stored Program Run automatic on events
() effect → other table views
timing: before, After
event: insert update delete
Scope: Row level, statment

Create trigger name
{ Before | After } { in|up|del }
on table
For each row
Begin
--SQl
End;

new: befor insert/update
old: after delete/update

business rules, auditing,

* Variables:-

(1) User defined
Select @var_name = val ;
:= val;
Select @var; display.

set @var = val;

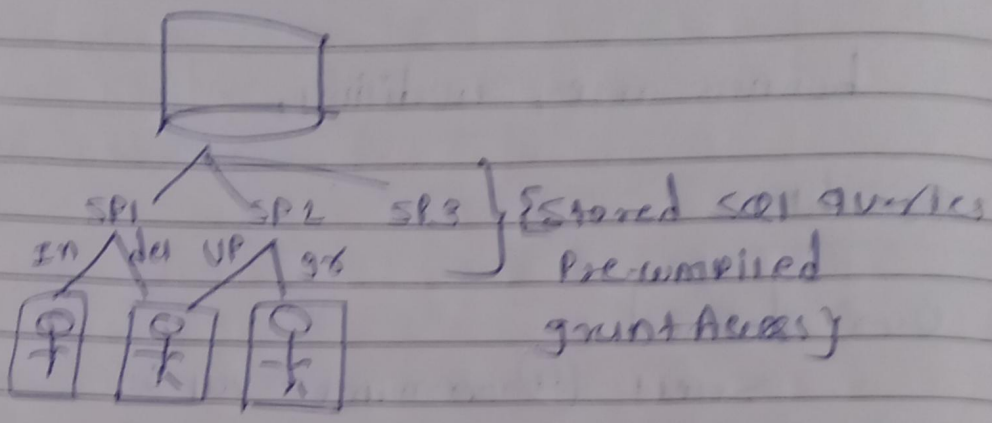
(2) local var.
Strongly typed var
No need of @

declare

→ insert trigger new/update bef
After delete/update.

* Stored Procedures

→



→ Set of SQL statment, faster Acess, Reusable function As SQL

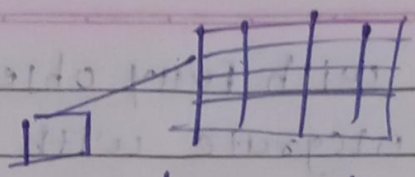
Delimeter //
Create Procedure name (IN param dtype,
OUT Pname type)
Begin
INOUT
--
END
Delimeter ;

→ Call ProcedureName (param)

* When:-

→ Frequ used queries
batch insert/updates
Reports

Views:-



↳ data abstraction
data hiding, multiple user.

Query ∫ - saved Result : multiple use
write once.

virtual table of query.
doesn't store data by default.

Types:-

Simple view: single tbl
complex: Joins, agg
updatable views:
Read-only view:

Syntax:-

[create view name As
Select tbl-columns
from tbl
group/where/having
joins;]