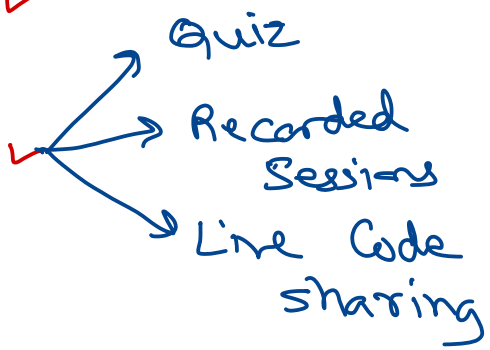
DAY 0

Agenda

- Welcome
- Zoom Profile/Attendance
- Zoom Channel
- Student Portal
- Schedules
- C Programming Setup
- Introduction to Database

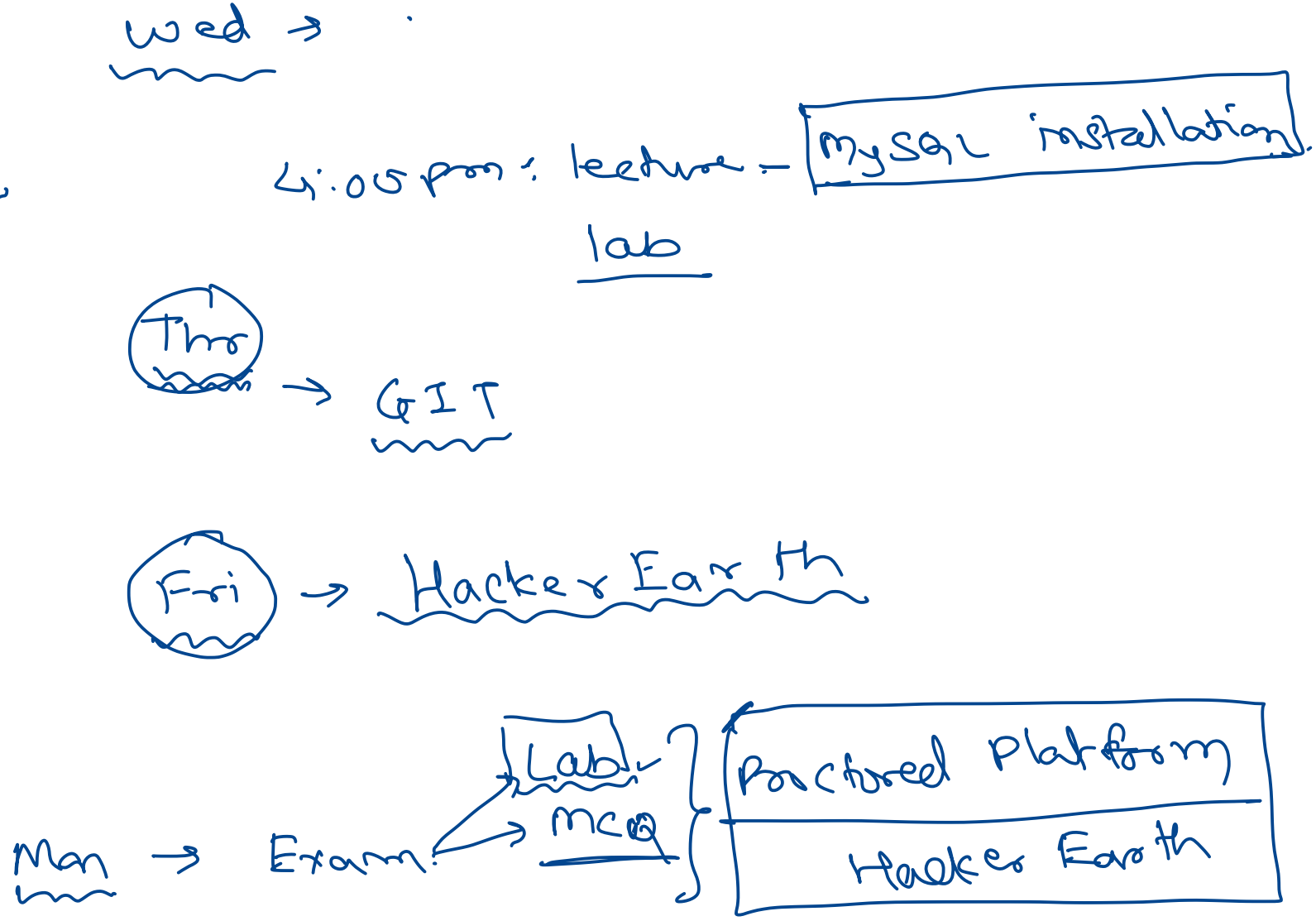


Zoom → Meetings → online classes
→ channels → lab/tech support
↓
① Faculty send notice/info/question.
- Reply

② Student send errors/problems.
- Text
- Screenshot.
- File
reply

③ Call with video
- Screen share
- Remote Control.

- Welcome
- Zoom Profile/Attendance
- Zoom Channel
- Student Portal
- Schedules
- C Programming Setup
- Introduction to Database



Database Technologies

Data mgmt → DBMS (generic records)
- C - Create / Insert / Add
- R - Retrieve / Select
- U - Update / Modify
- D - Delete / Remove

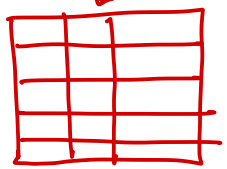
DAC → enterprise appln dev

DBMS
- RDBMS (SQL) (Relational)
- NoSQL

SQL : Structured Query Language

RDBMS : MySQL 8.x

NoSQL : Mongo DB



DAY 1

Agenda / Syllabus

- DBMS vs RDBMS
- MySQL: Introduction, Installation, …
- SQL
  - CREATE TABLE, MySQL data types
  - SELECT with LIMIT, ORDER, WHERE, GROUP BY, HAVING
  - INSERT, UPDATE, DELETE
  - Joins, Sub-queries
  - Transaction & Locking
  - GRANT & REVOKE
- MySQL programming (PSM) / PL-SQL
  - Stored procedure
  - Cursors
  - Functions
  - Triggers

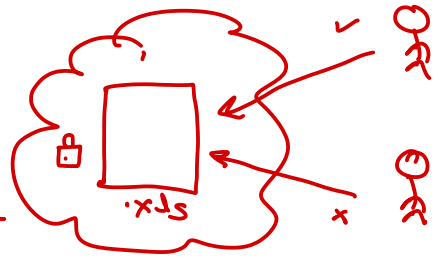
Syllabus : ① RDBMS : MySQL
② NoSQL : Mongo

Evaluation = 100 marks
Theory = 40 → CCEE (course end) MCQ
Lab = 40 → proctored lab exam, Lab assignments (module end)
Internals = 20 → MCQ

Interview Preparations
Interview Questions
- Rapid Fire
- Hot Seat

DBMS Database Management System.

- Any enterprise application need to manage data.
- In early days of software development, programmers store data into files and does operation on it. However data is highly application specific.
- Even today many software manage their data in custom formats e.g. Tally, Address book, etc.
- As data management became more common, DBMS systems were developed to handle the data. This enabled developers to focus on the business logic e.g. FoxPro, DBase, Excel, etc.
- At least CRUD (Create, Retrieve, Update and Delete) operations are supported by all databases.
- Traditional databases are file based, less secure, single-user, non-distributed, manage less amount of data (MB), complicated relation management, file-locking and need number of lines of code to use in applications.

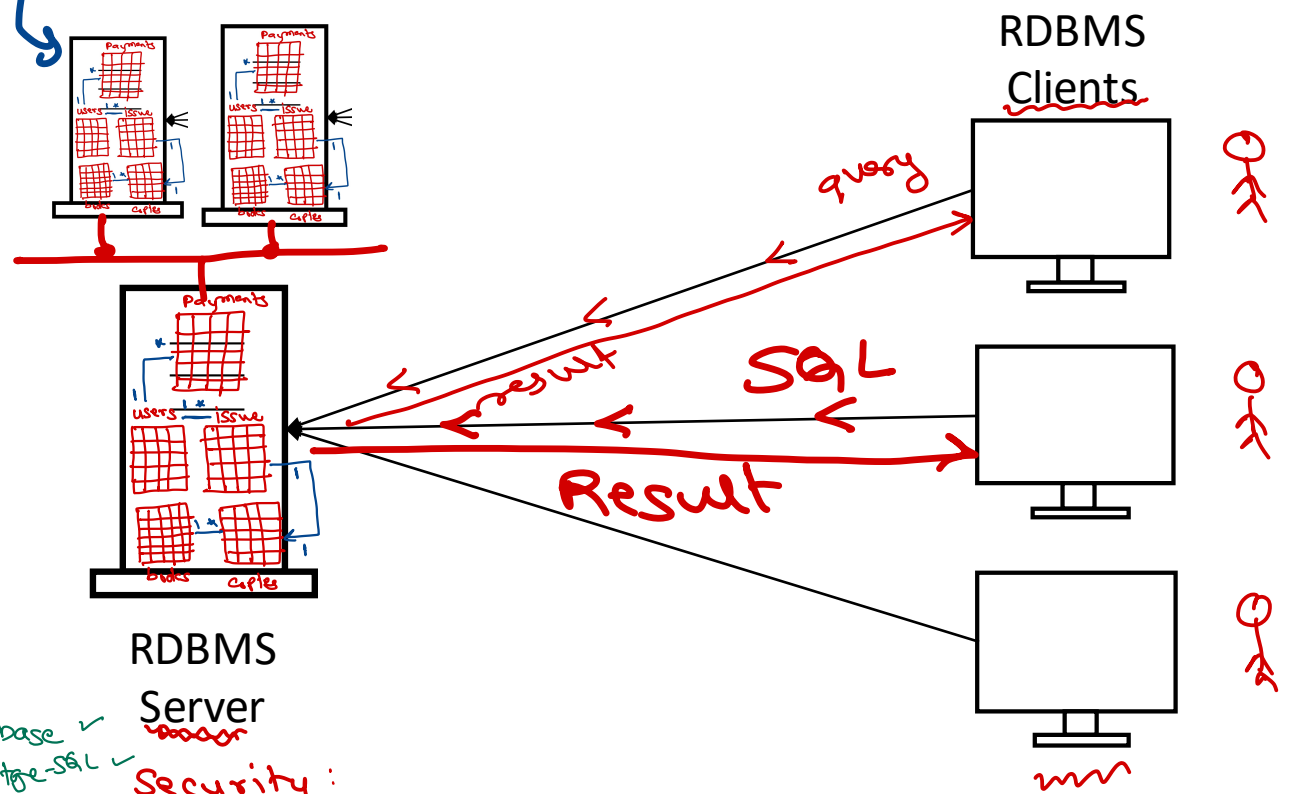


RDBMS

CRUD

DB cluster — Number of computers connected in a network for dedicated task.

- RDBMS is relational DBMS.
- It organizes data into Tables, rows and columns. The tables are related to each other.
- RDBMS follow table structure, more secure, multi-user, server-client architecture, server side processing, clustering support, manage huge data (TB), built-in relational capabilities, table-locking or row-locking and can be easily integrated with applications.
- e.g. DB2, Oracle, MS-SQL, MySQL, MS-Access, SQLite, … (mainframe, enterprise apps, open source; file based RDBMS (small data))
- RDBMS design is based on Codd's rules developed at IBM (in 1970). → Mathematician = 1 + 12 rules → based on set theory

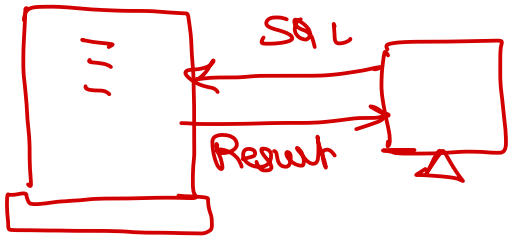


Cybase, Postgre-SQL, Informix, ...

Security:
OS + DB authentication
+ object level privileges
+ database/table/OPS.

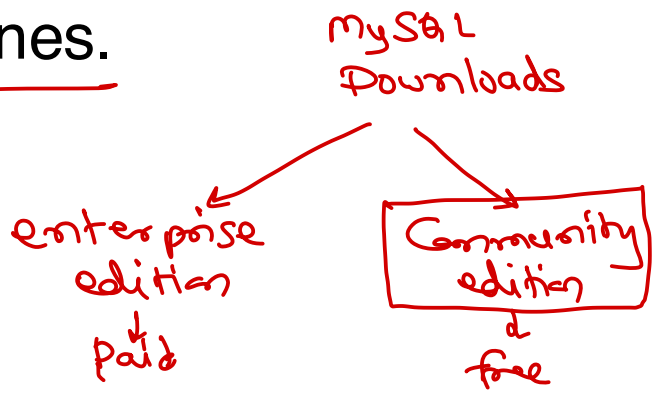
SQL : Structured Query Language

- Clients send SQL queries to RDBMS server and operations are performed accordingly.
- Originally it was named as RQBE (Relational Query By Example).
- SQL is ANSI standardised in 1987 and then revised multiple times adding new features. Recent revision in 2016.
- SQL is case insensitive.
- There are five major categories:
  - DDL: Data Definition Language e.g. CREATE, ALTER, DROP, RENAME.
  - DML: Data Manipulation Language e.g. INSERT, UPDATE, DELETE.
  - DQL: Data Query Language e.g. SELECT.
  - DCL: Data Control Language e.g. CREATE USER, GRANT, REVOKE.
  - TCL: Transaction Control Language e.g. SAVEPOINT, COMMIT, ROLLBACK.
- Table & column names allows alphabets, digits & few special symbols.
- If name contains special symbols then it should be back-quotes.
- e.g. Tbl1, `T1#`, `T2$` etc. Names can be max 30 chars long.



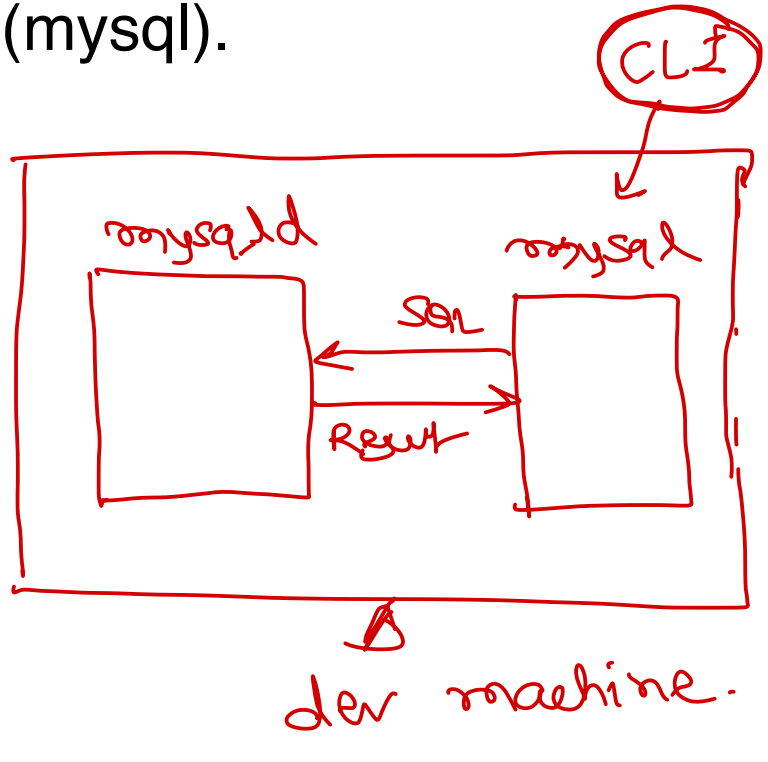
MySQL

- Developed by Michael Widenius in 1995. It is named after his daughter name Myia.
- Sun Microsystems acquired MySQL in 2008.
- Oracle acquired Sun Microsystem in 2010.
- MySQL is free and open-source database under GPL. However some enterprise modules are close sourced and available only under commercial version of MySQL.
- MariaDB is completely open-source clone of MySQL.
- MySQL support multiple database storage and processing engines.
- MySQL versions:
  - < 5.5: MyISAM storage engine
  - 5.5: InnoDb storage engine
  - 5.6: SQL Query optimizer improved, memcached style NoSQL
  - 5.7: Windowing functions, JSON data type added for flexible schema
  - 8.0: CTE, NoSQL document store.
- MySQL is database of year 2019 (in database engine ranking).



MySQL installation on Ubuntu/Linux

- terminal> sudo apt-get install mysql-community-server mysql-community-client
- This installs MySQL server (mysqld) and MySQL client (mysql).
- MySQL Server (mysqld)
  - Run as background process.
  - Implemented in C/C++.
  - Process SQL queries and generate results.
  - By default run on port 3306.
  - Controlled via systemctl.
    - terminal> sudo systemctl start|stop|status|enable|disable mysql
- MySQL client (mysql)
  - Command line interface
  - Send SQL queries to server and display its results.
  - terminal> mysql –u root -p
- Additional MySQL clients
  - MySQL workbench
  - PHPMyAdmin



Getting started

- root login can be used to perform CRUD as well as admin operations.
- It is recommended to create users for performing non-admin tasks.
  - mysql> CREATE DATABASE db;
  - mysql> SHOW DATABASES;
  - mysql> CREATE USER dbuser@localhost IDENTIFIED BY 'dbpass';
  - mysql> SELECT user, host FROM mysql.user;
  - mysql> GRANT ALL PRIVILEGES ON db.* TO dbuser@localhost;
  - mysql> FLUSH PRIVILEGES;
  - mysql> EXIT;
- terminal> mysql –u dbuser –pdbpass
  - mysql> SHOW DATABASES;
  - mysql> SELECT USER(), DATABASE();
  - mysql> USE db;
  - mysql> SHOW TABLES;
  - mysql> CREATE TABLE student(id INT, name VARCHAR(20), marks DOUBLE);
  - mysql> INSERT INTO student VALUES(1, 'Abc', 89.5);
  - mysql> SELECT * FROM student;

DAY 2

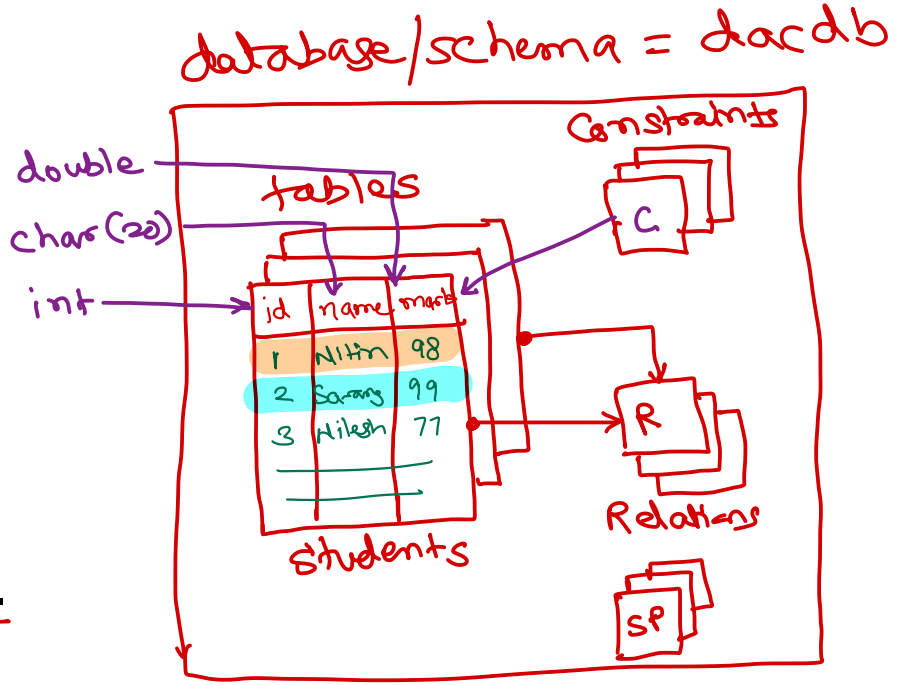
Getting started

- root login can be used to perform CRUD as well as admin operations.
- It is recommended to create users for performing non-admin tasks.
  - mysql> CREATE DATABASE db;
  - mysql> SHOW DATABASES;
  - mysql> CREATE USER dbuser@localhost IDENTIFIED BY 'dbpass';
  - mysql> SELECT user, host FROM mysql.user;
  - mysql> GRANT ALL PRIVILEGES ON db.* TO dbuser@localhost;
  - mysql> FLUSH PRIVILEGES;
  - mysql> EXIT;
- terminal> mysql –u dbuser –pdbpass
  - mysql> SHOW DATABASES;
  - mysql> SELECT USER(), DATABASE();
  - mysql> USE db;
  - mysql> SHOW TABLES;
  - mysql> CREATE TABLE student(id INT, name VARCHAR(20), marks DOUBLE);
  - mysql> INSERT INTO student VALUES(1, 'Abc', 89.5);
  - mysql> SELECT * FROM student;

Database logical layout

DESCRIBE students; → Show table structure (metadata).

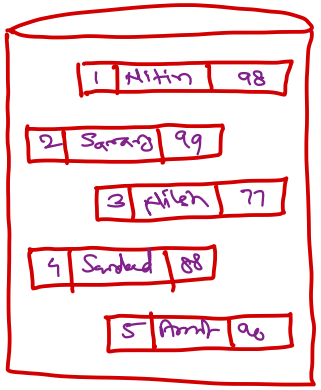
- Database/schema is like a namespace/container that stores all db objects related to a project.
- It contains tables, constraints, relations, stored procedures, functions, triggers, ...
- There are some system databases e.g. mysql, performance_schema, information_schema, sys, ... They contains db internal/system information.
  - e.g. SELECT user, host FROM mysql.user;
- A database contains one or more tables.
- Tables have multiple columns.
- Each column is associated with a data-type.
- Columns may have zero or more constraints.
- The data in table is in multiple rows.
- Each row have multiple values (as per columns).



Database physical layout

As a db developer, we must understand logical layout of the database. Physical layout understanding is only for info/GK.

- In MySQL, the data is stored on disk in its data directory i.e. /var/lib/mysql (Linux)
- Each database/schema is a separate sub-directory in data dir.
- Each table in the db, is a file on disk.
- e.g. student table in current db is stored in file /var/lib/mysql/db/student.ibd.
- Data is stored in binary format.
- A file may not be contiguously stored on hard disk.
- Data rows are not contiguous. They are scattered in the hard disk.
- In one row, all fields are consecutive.
- When records are selected, they are selected in any order.



SQL scripts

- SQL script is multiple SQL queries written into a .sql file.
- SQL scripts are mainly used while database backup and restore operations.
- SQL scripts can be executed from terminal as:
  - terminal> mysql –u user –ppassword db < /path/to/sqlfile
- SQL scripts can be executed from command line as:
  - mysql> SOURCE /path/to/sqlfile
- Note that SOURCE is MySQL CLI client command.
- It reads commands one by one from the script and execute them on server.

DAY 3

MySQL data types

- RDBMS have similar data types (but not same).
- MySQL data types can be categorised as follows
  - Numeric types (Integers)
    - TINYINT (1 byte), SMALLINT (2 byte), MEDIUMINT (3 byte), INT (4 byte), BIGINT (8 byte), BIT(n bits)
    - integer types can signed (default) or unsigned.
  - Numeric types (Floating point)
    - approx. precision – FLOAT (4 byte), DOUBLE (8 byte) | DECIMAL(m, n) – exact precision
  - Date/Time types
    - DATE, TIME, DATETIME, TIMESTAMP, YEAR
  - String types – size = number of chars * size of char
    - CHAR(1-255) – Fixed length, Very fast access.
    - VARCHAR(1-65535) – Variable length, Stores length + chars.
    - TINYTEXT (255), TEXT (64K), MEDIUMTEXT (16M), LONGTEXT (4G) – Variable length, Slower access.
  - Binary types – size = number of bytes
    - BINARY, VARBINARY, TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB
  - Miscellaneous types
    - ENUM, SET

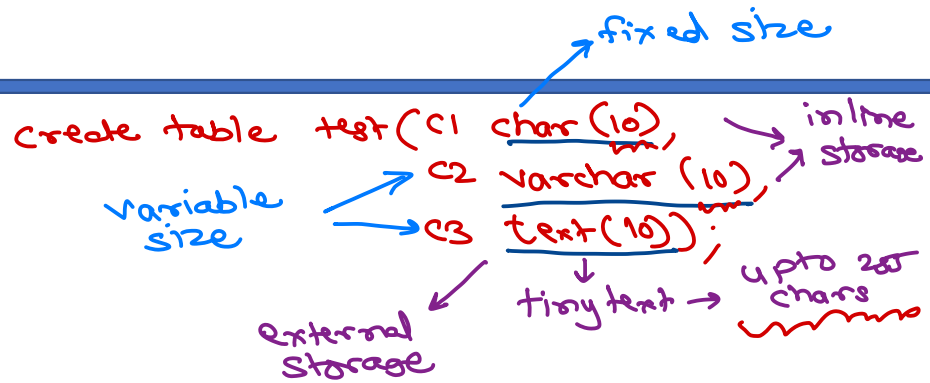
MySQL : INT, DECIMAL, FLOAT
Oracle : NUMBER
DERBY : INTEGER

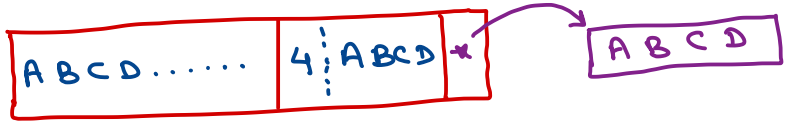
±2^15 ±2^23 ±2^31 ±2^63
±2^7
0 to 2^16 0 to 2^32
create table t1(c1 smallint unsigned, c2 int unsigned);
IEEE-754 format
create table t2(col decimal(8,3));
1 2 3 4 5.6 7 8
8

images, pdf, docs, ...
like text.

Size of char
depend on charset
1) ASCII → 1 byte
2) Unicode → 2 bytes
3) EBCDIC → 4 bytes (MBCS)

CHAR vs VARCHAR vs TEXT





- CHAR
  - Fixed inline storage.
  - If smaller data is given, rest of space is unused.
  - Very fast access.
- VARCHAR
  - Variable inline storage.
  - Stores length and characters.
  - Slower access than CHAR.
- TEXT
  - Variable external storage.
  - Very slow access.
  - Not ideal for indexing.
- CREATE TABLE temp(c1 CHAR(4), c2 VARCHAR(4), c3 TEXT(4));
- DESC temp;
- INSERT INTO temp VALUES('abcd', 'abcd', 'abcdef');

INSERT – DML

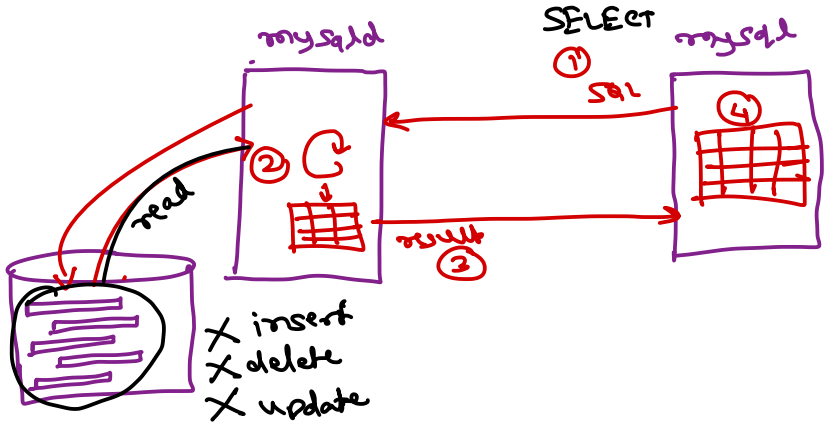
- Insert a new row (all columns, fixed order).
  - INSERT INTO table VALUES (v1, v2, v3);
- Insert a new row (specific columns, arbitrary order).
  - INSERT INTO table(c3, c1, c2) VALUES (v3, v1, v2);
  - INSERT INTO table(c1, c2) VALUES (v1, v2);
  - Missing columns data is NULL.
  - NULL is special value and it is not stored in database.
- Insert multiple rows.
  - INSERT INTO table VALUES (av1, av2, av3), (bv1, bv2, bv3), (cv1, cv2, cv3).
- Insert rows from another table.
  - INSERT INTO table SELECT c1, c2, c3 FROM another-table;
  - INSERT INTO table (c1,c2) SELECT c1, c2 FROM another-table;

DAY 4

SELECT – DQL

order in which cols added in table while creation.

- Select all columns (in fixed order).
  - SELECT * FROM table;
- Select specific columns / in arbitrary order.
  - SELECT c1, c2, c3 FROM table;
- Column alias
  - SELECT c1 AS col1, c2 col2 FROM table;
- Computed columns.
  - SELECT c1, c2, c3, expr1, expr2 FROM table;
  - SELECT c1,
  - CASE WHEN condition1 THEN value1,
  - CASE WHEN condition2 THEN value2,
  - …
  - ELSE valuen
  - END
  - FROM table;

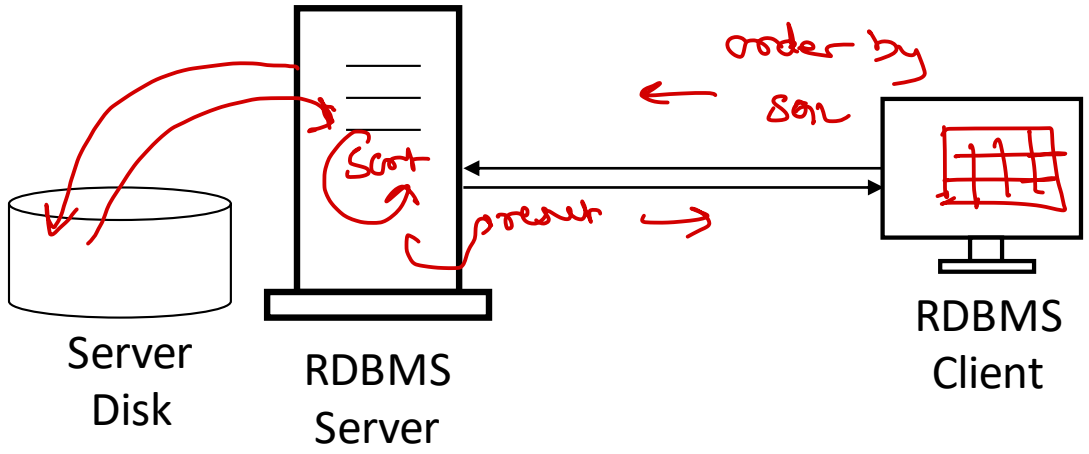


SELECT – DQL

- Distinct values in column.
  - SELECT DISTINCT c1 FROM table;
  - SELECT DISTINCT c1, c2 FROM table;
- Select limited rows.
  - SELECT * FROM table LIMIT n;
  - SELECT * FROM table LIMIT m, n;

SELECT – DQL – ORDER BY

- In db rows are scattered on disk. Hence may not be fetched in a fixed order.
- Select rows in asc order.
  - SELECT * FROM table ORDER BY c1;
  - SELECT * FROM table ORDER BY c2 ASC;
- Select rows in desc order.
  - SELECT * FROM table ORDER BY c3 DESC;
- Select rows sorted on multiple columns.
  - SELECT * FROM table ORDER BY c1, c2;
  - SELECT * FROM table ORDER BY c1 ASC, c2 DESC;
  - SELECT * FROM table ORDER BY c1 DESC, c2 DESC;
- Select top or bottom n rows.
  - SELECT * FROM table ORDER BY c1 ASC LIMIT n;
  - SELECT * FROM table ORDER BY c1 DESC LIMIT n;
  - SELECT * FROM table ORDER BY c1 ASC LIMIT m, n;



To sort too many records and/or sort on many columns will slow down execution of query.

SELECT – DQL – WHERE

- It is always good idea to fetch only required rows (to reduce network traffic).
- The WHERE clause is used to specify the condition, which records to be fetched.
- Relational operators
  - <, >, <=, >=, =, != or <>
- NULL related operators
  - NULL is special value and cannot be compared using relational operators.
  - IS NULL or <=>, IS NOT NULL.
- Logical operators
  - AND, OR, NOT

DAY 5

SELECT – DQL – WHERE

- It is always good idea to fetch only required rows (to reduce network traffic).
- The WHERE clause is used to specify the condition, which records to be fetched.
- Relational operators
  - <, >, <=, >=, =, != or <>
- NULL related operators
  - NULL is special value and cannot be compared using relational operators.
  - IS NULL or <=>, IS NOT NULL.
- Logical operators
  - AND, OR, NOT

SELECT – DQL – WHERE

- BETWEEN operator (include both ends)
  - c1 BETWEEN val1 AND val2
- IN operator (equality check with multiple values)
  - c1 IN (val1, val2, val3)
- LIKE operator (similar strings)
  - c1 LIKE ‘pattern’.
  - % represent any number of any characters.
  - _ represent any single character.

DAY 6

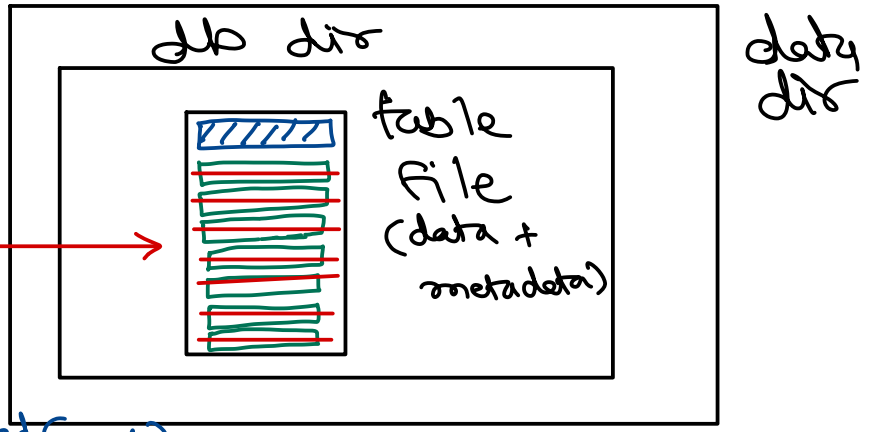
UPDATE – DML

- To change one or more rows in a table.
- Update row(s) single column.
  - UPDATE table SET c2=new-value WHERE c1=some-value;
- Update multiple columns.
  - UPDATE table SET c2=new-value, c3=new-value WHERE c1=some-value;
- Update all rows single column.
  - UPDATE table SET c2=new-value;

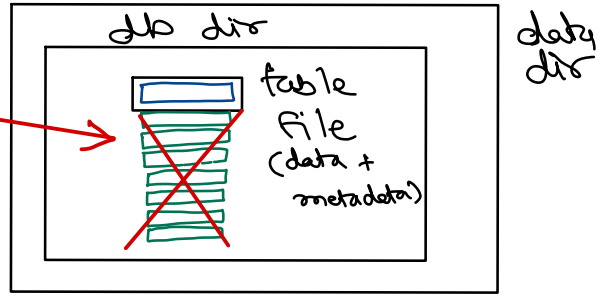
DELETE – DML vs TRUNCATE – DDL vs DROP – DDL

- DELETE
  - To delete one or more rows in a table.
  - Delete row(s)
    - DELETE FROM table WHERE c1=value;
  - Delete all rows
    - DELETE FROM table
- TRUNCATE
  - Delete all rows.
    - TRUNCATE TABLE table;
  - Truncate is faster than DELETE.
- DROP
  - Delete all rows as well as table structure.
    - DROP TABLE table;
    - DROP TABLE table IF EXISTS;
  - Delete database/schema.
    - DROP DATABASE db;

Query impl change from RDBMS to RDBMS.
In general - RDBMS.



① mark all rows as deleted.
② space occupied by them can be overwritten/reused by new records.
③ actual table file size is not changed (much).
④ DML query - roll backed.



① truncate file size so that only structure is kept.
② all rows space is released.
③ much faster operation for huge table.
④ DDL query - can never be rollbacked.

① delete table file struct + data.
② DDL - no rollback.

Synonym for DATABASE ← SCHEMA → dbname

Seeking HELP

C R U D

- HELP is client command to seek help on commands/functions.
  - HELP SELECT;
  - HELP Functions;
  - HELP SIGN;

DAY 7

Seeking HELP

C R U D

- HELP is client command to seek help on commands/functions.
  - HELP SELECT;
  - HELP Functions;
  - HELP SIGN;

DUAL table → First used in oracle. Two rows table. Later it made single row & col table.

- A dummy/in-memory a table having single row & single column.
- It is used for arbitrary calculations, testing functions, etc.
  - SELECT 2 + 3 * 4 FROM DUAL;
  - SELECT NOW() FROM DUAL;
  - SELECT USER(), DATABASE() FROM DUAL;
- In MySQL, DUAL keyword is optional.
  - SELECT 2 + 3 * 4;
  - SELECT NOW();
  - SELECT USER(), DATABASE();

Numeric & String functions

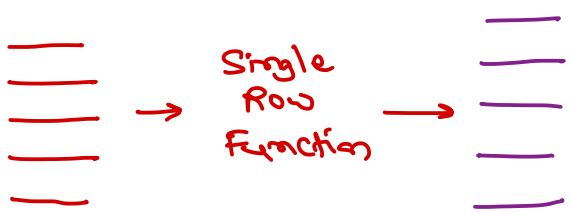
- ABS()
- POWER()
- ROUND(), FLOOR(), CEIL()

- ASCII(), CHAR()
- CONCAT()
- SUBSTRING()
- LOWER(), UPPER()
- TRIM(), LTRIM(), RTRIM()
- LPAD(), RPAD()
- REGEXP_LIKE()

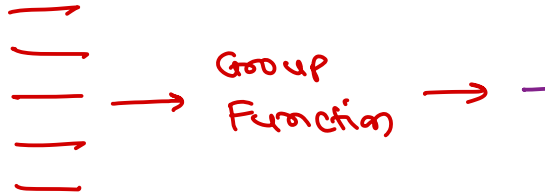
DAY 8

SQL functions

- RDBMS provides many built-in functions to process the data.
- These functions can be classified as:
  - Single row functions
    - One row input produce one row output.
    - e.g. ABS(), CONCAT(), IFNULL(), …
  - Multi-row or Group functions or aggregate functions
    - Values from multiple rows are aggregated to single value.
    - e.g. SUM(), MIN(), MAX(), …



function work on one or more values/columns from same row of the table.



function work on multiple values from multiple rows

- These functions can also be categorized based on data types or usage.
  - Numeric functions
  - String functions
  - Date and Time functions
  - Control flow functions
  - Information functions
  - Miscellaneous functions

Numeric & String functions

- ABS()
- POWER()
- ROUND(), FLOOR(), CEIL()

- ASCII(), CHAR()
- CONCAT()
- SUBSTRING()
- LOWER(), UPPER()
- TRIM(), LTRIM(), RTRIM()
- LPAD(), RPAD()
- REGEXP_LIKE()

Date-Time and Information functions

- VERSION()
- USER(), DATABASE()

- MySQL supports multiple date time related data types
  - DATE (3), TIME (3), DATETIME (5), TIMESTAMP (4), YEAR (1)
- SYSDATE(), NOW()
- DATE(), TIME()
- DAYOFMONTH(), MONTH(), YEAR(), HOUR(), MINUTE(), SECOND(), …
- DATEDIFF(), DATE_ADD(), TIMEDIFF()
- MAKEDATE(), MAKETIME()

Control and NULL and List functions

- NULL is special value in RDBMS that represents absence of value in that column.
- NULL values do not work with relational operators and need to use special operators.
- Most of functions return NULL if NULL value is passed as one of its argument.
- ISNULL()
- IFNULL()
- NULLIF()
- COALESCE()

- GREATEST(), LEAST()

- IF(condition, true-value, false-value)

Group functions

- Work on group of rows of table.
- Input to function is data from multiple rows & then output is single row. Hence these functions are called as "Multi Row Function“ or "Group Functions“.
- These functions are used to perform aggregate ops like sum, avg, max, min, count or std dev, etc. Hence these fns are also called as "Aggregate Functions".
- Example: SUM(), AVG(), MAX(), MIN(), COUNT().
- NULL values are ignored by group functions.
- Limitations of GROUP functions:
  - Cannot select group function along with a column.
  - Cannot select group function along with a single row fn.
  - Cannot use group function in WHERE clause/condition.
  - Cannot nest a group function in another group fn.

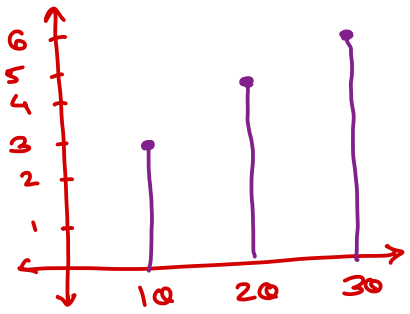
DAY 9

Group functions

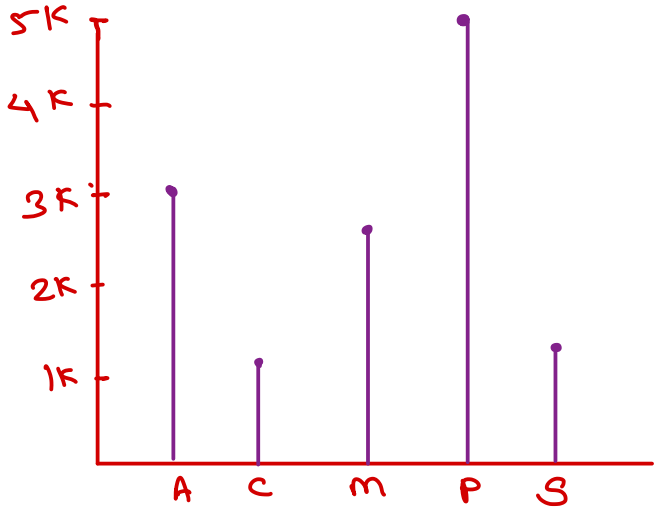
- Work on group of rows of table.
- Input to function is data from multiple rows & then output is single row. Hence these functions are called as "Multi Row Function“ or "Group Functions“.
- These functions are used to perform aggregate ops like sum, avg, max, min, count or std dev, etc. Hence these fns are also called as "Aggregate Functions".
- Example: SUM(), AVG(), MAX(), MIN(), COUNT().
- NULL values are ignored by group functions.
- Limitations of GROUP functions:
  - Cannot select group function along with a column.
  - Cannot select group function along with a single row fn.
  - Cannot use group function in WHERE clause/condition.
  - Cannot nest a group function in another group fn.

GROUP BY clause

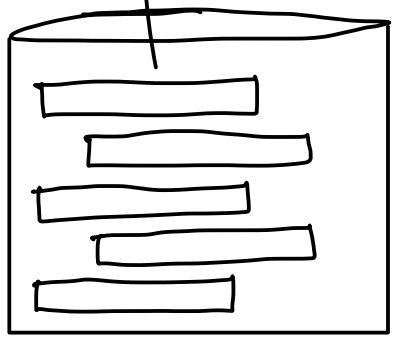
- GROUP BY is used for analysis of data i.e. generating reports & charts.
- When GROUP BY single column, generated output can be used to plot 2-D chart.
  When GROUP BY two column, generated output can be used to plot 3-D chart and so on.
- GROUP BY queries are also called as Multi-dimensional / Spatial queries.
- Syntactical Characteristics:
  - If a column is used for GROUP BY, then it may or may not be used in SELECT clause.
  - If a column is in SELECT, it must be in GROUP BY.
- When GROUP BY query is fired on database server, it does following:
  - Load data from server disk into server RAM.
  - Sort data on group by columns.
  - Group similar records by group columns.
  - Perform given aggregate ops on each column.
  - Send result to client.

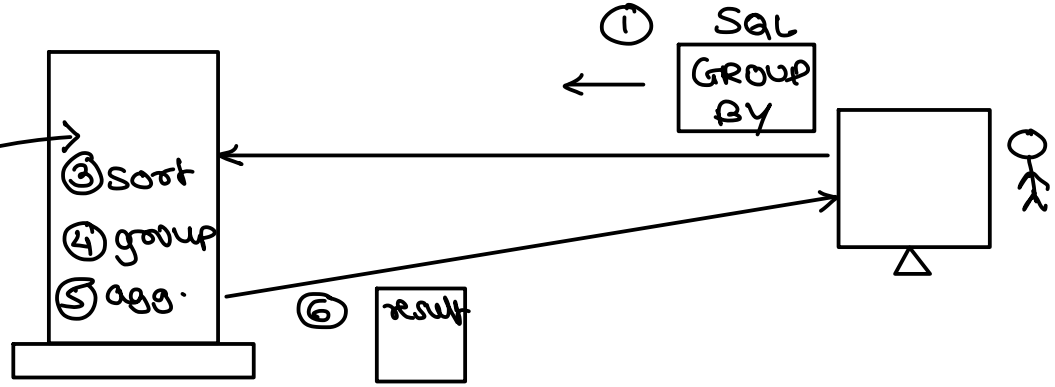



```
SELECT deptno, COUNT(empno)
FROM emp
GROUP BY deptno;
```




```
SELECT job, AVG(sal)
FROM emp
GROUP BY job;
```





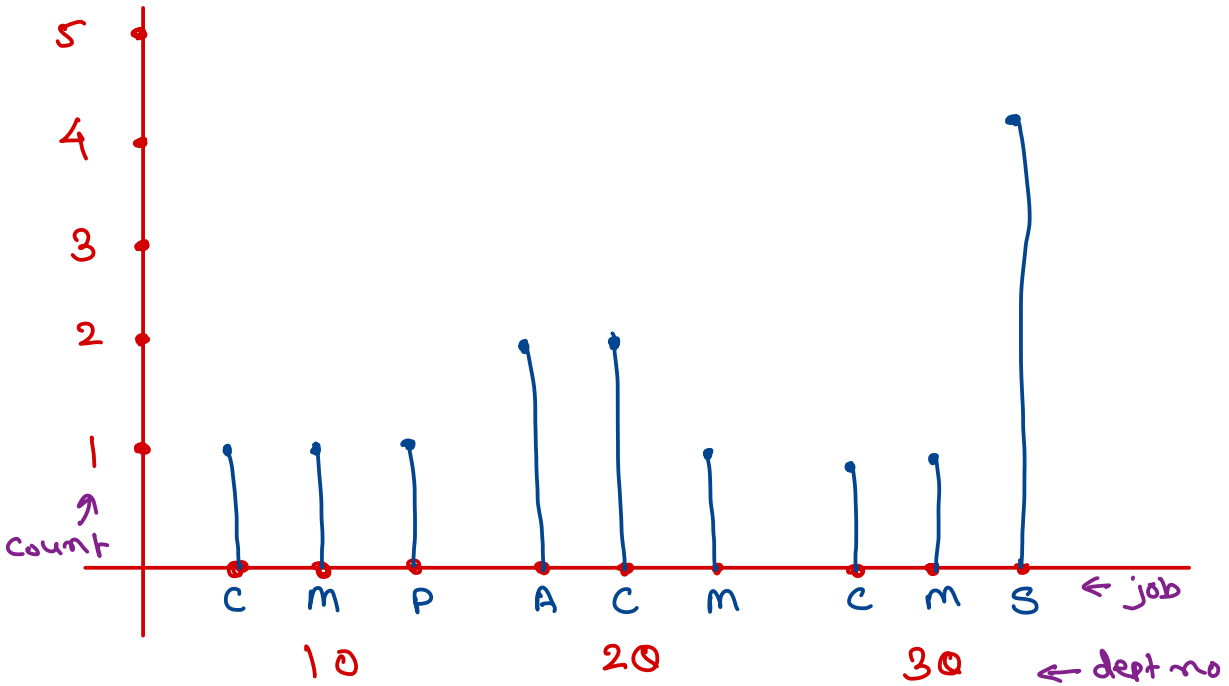
Group by on multiple columns and/or too many rows slow down execution.

DAY 10

GROUP BY clause

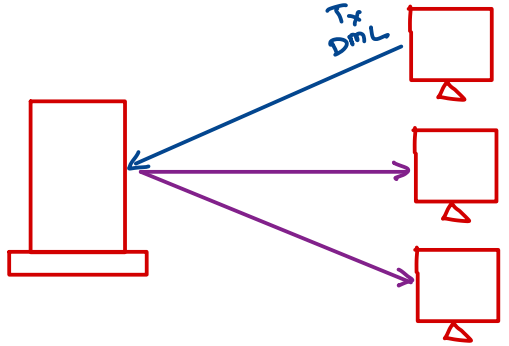
- GROUP BY is used for analysis of data i.e. generating reports & charts.
- When GROUP BY single column, generated output can be used to plot 2-D chart. When GROUP BY two column, generated output can be used to plot 3-D chart and so on.
- GROUP BY queries are also called as Multi-dimensional / Spatial queries.
- Syntactical Characteristics:
  - If a column is used for GROUP BY, then it may or may not be used in SELECT clause.
  - If a column is in SELECT, it must be in GROUP BY.
- When GROUP BY query is fired on database server, it does following:
  - Load data from server disk into server RAM.
  - Sort data on group by columns.
  - Group similar records by group columns.
  - Perform given aggregate ops on each column.
  - Send result to client.

```
select deptno, job, count(empno)
from emp
group by deptno, job;
```

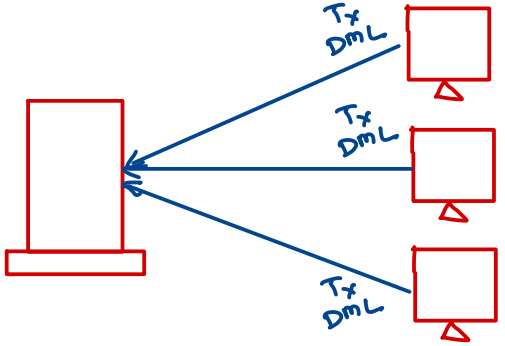


Transaction

- Transaction is set of DML queries executed as a single unit.
- Transaction examples
  - accounts table [id, type, balance]
  - UPDATE accounts SET balance=balance-1000 WHERE id = 1;
  - UPDATE accounts SET balance=balance+1000 WHERE id = 2;



- RDBMS transaction have ACID properties.
  - Atomicity
    - All queries are executed as a single unit. If any query is failed, other queries are discarded.
  - Consistency
    - When transaction is completed, all clients see the same data.
  - Isolation
    - Multiple transactions (by same or multiple clients) are processed concurrently.
  - Durable
    - When transaction is completed, all data is saved on disk.



Transaction

- Transaction management
  - START TRANSACTION;
  - … dml1 dml2 dml3
  - COMMIT WORK; → final save

  - START TRANSACTION;
  - … dml1; dml2; dml3;
  - ROLLBACK WORK; → discard
- In MySQL autocommit variable is by default 1. So each DML command is auto-committed into database.
  - SELECT @@autocommit;
- Changing autocommit to 0, will create new transaction immediately after current transaction is completed. This setting can be made permanent in config file.
  - SET autocommit=0;

DAY 11

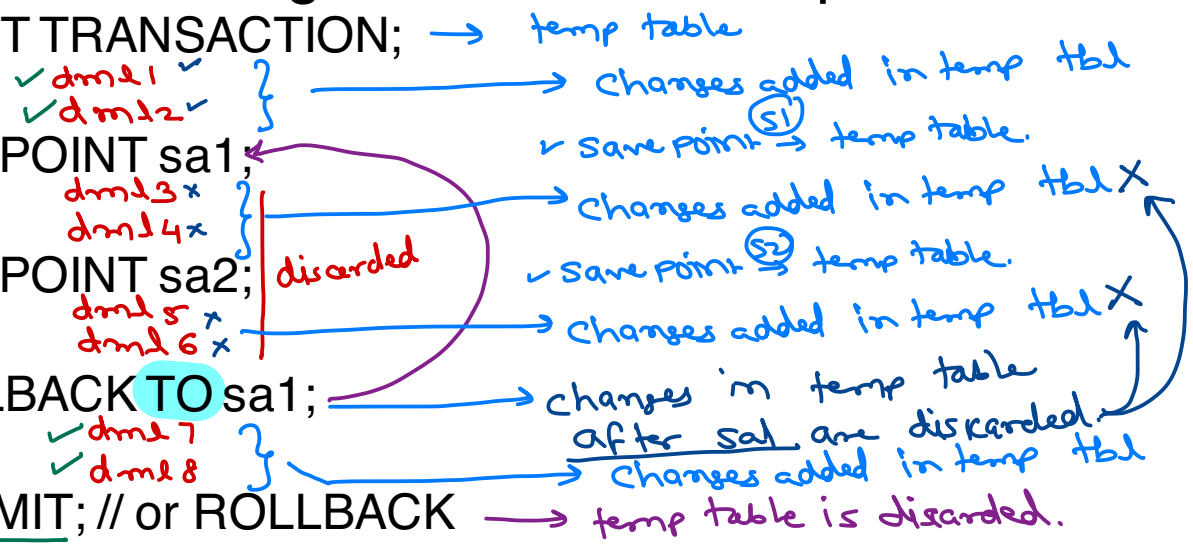
Transaction

- Transaction management
  - START TRANSACTION;
  - … dml 1 dml 2 dml 3
  - COMMIT WORK; → final save

  - START TRANSACTION;
  - … dml 1; dml 2; dml 3;
  - ROLLBACK WORK; → discard
- In MySQL autocommit variable is by default 1. So each DML command is auto-committed into database.
  - SELECT @@autocommit;
- Changing autocommit to 0, will create new transaction immediately after current transaction is completed. This setting can be made permanent in config file.
  - SET autocommit=0;

Transaction

- Save-point is state of database tables (data) at the moment (within a transaction).
- It is advised to create save-points at end of each logical section of work.
- Database user may choose to rollback to any of the save-point.
- Transaction management with Save-points
  - START TRANSACTION;
  - …
  - SAVEPOINT sa1;
  - …
  - SAVEPOINT sa2;
  - …
  - ROLLBACK TO sa1;
  - …
  - COMMIT; // or ROLLBACK



- Commit always commit the whole transaction.
- ROLLBACK or COMMIT clears all save-points.

Transaction

- Transaction is set of DML statements.
- If any DDL statement is executed, current transaction is automatically committed.
- Any power failure, system or network failure automatically rollback current state.
- Transactions are isolated from each other and are consistent.

Row locking

- When an user update or delete a row (within a transaction), that row is locked and becomes read-only for other users.
- The other users see old row values, until transaction is committed by first user.
- If other users try to modify or delete such locked row, their transaction processing is blocked until row is unlocked.
- Other users can INSERT into that table. Also they can UPDATE or DELETE other rows.
- The locks are automatically released when COMMIT/ROLLBACK is done by the user.
- This whole process is done automatically in MySQL. It is called as "OPTIMISTIC LOCKING".

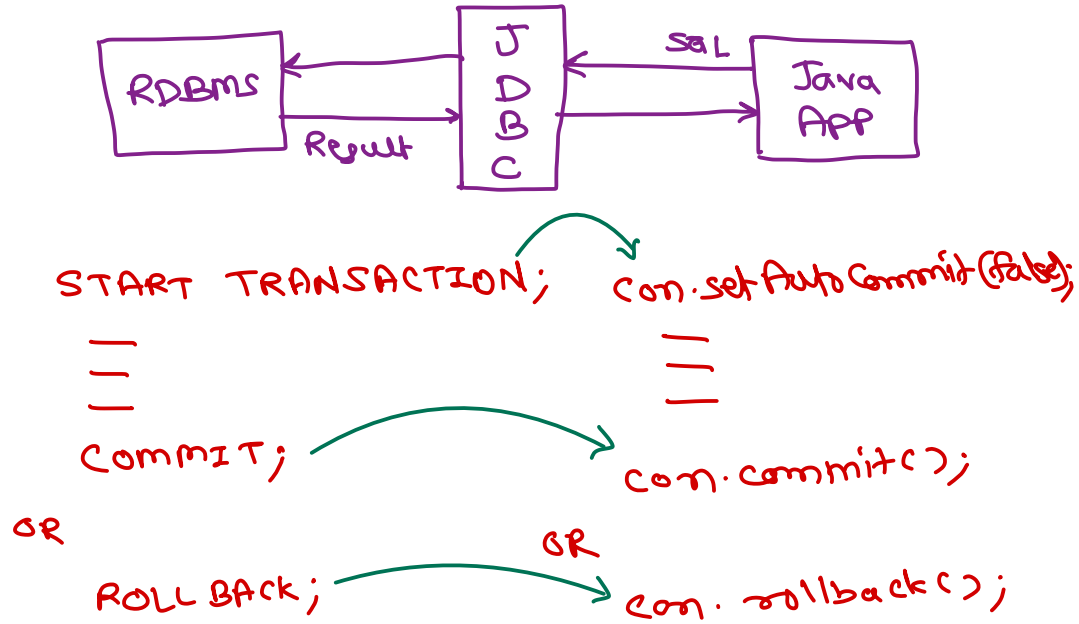
DAY 12

Row locking

- When an user update or delete a row (within a transaction), that row is locked and becomes read-only for other users.
- The other users see old row values, until transaction is committed by first user.
- If other users try to modify or delete such locked row, their transaction processing is blocked until row is unlocked. → or timeout.
- Other users can INSERT into that table. Also they can UPDATE or DELETE other rows.
- The locks are automatically released when COMMIT/ROLLBACK is done by the user. → first.
- This whole process is done automatically in MySQL. It is called as "OPTIMISTIC LOCKING".

Row locking

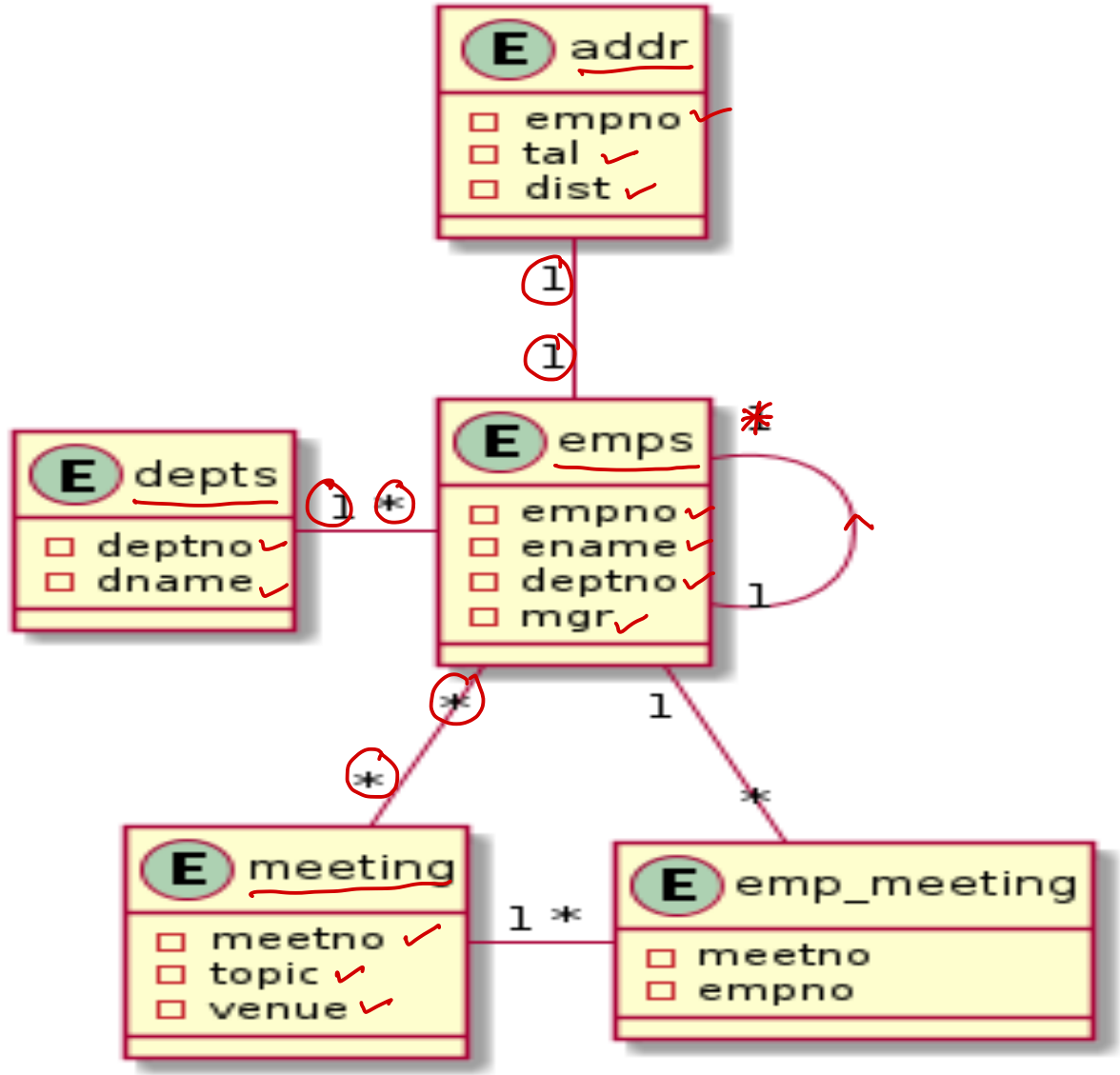
- Manually locking the row in advanced before issuing UPDATE or DELETE is known as "PESSIMISTIC LOCKING".
- This is done by appending FOR UPDATE to the SELECT query.
- It will lock all selected rows, until transaction is committed or rollbacked. (or timeout)
- If these rows are already locked by another users, the SELECT operation is blocked until rows lock is released.



- By default MySQL does table locking. Row locking is possible only when table is indexed on the column.

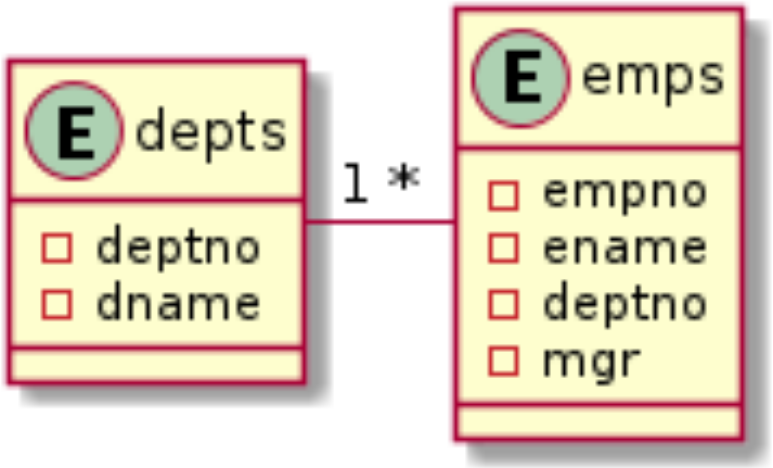
Entity Relations

- To avoid redundancy of the data, data should be organized into multiple tables so that tables are related to each other.
- The relations can be one of the following
  - One to One → emps → addr
  - One to Many → depts → emps
  - Many to One → emps → depts
  - Many to Many → emps ↔ meeting
- Entity relations is outcome of Normalization process.



SQL Joins

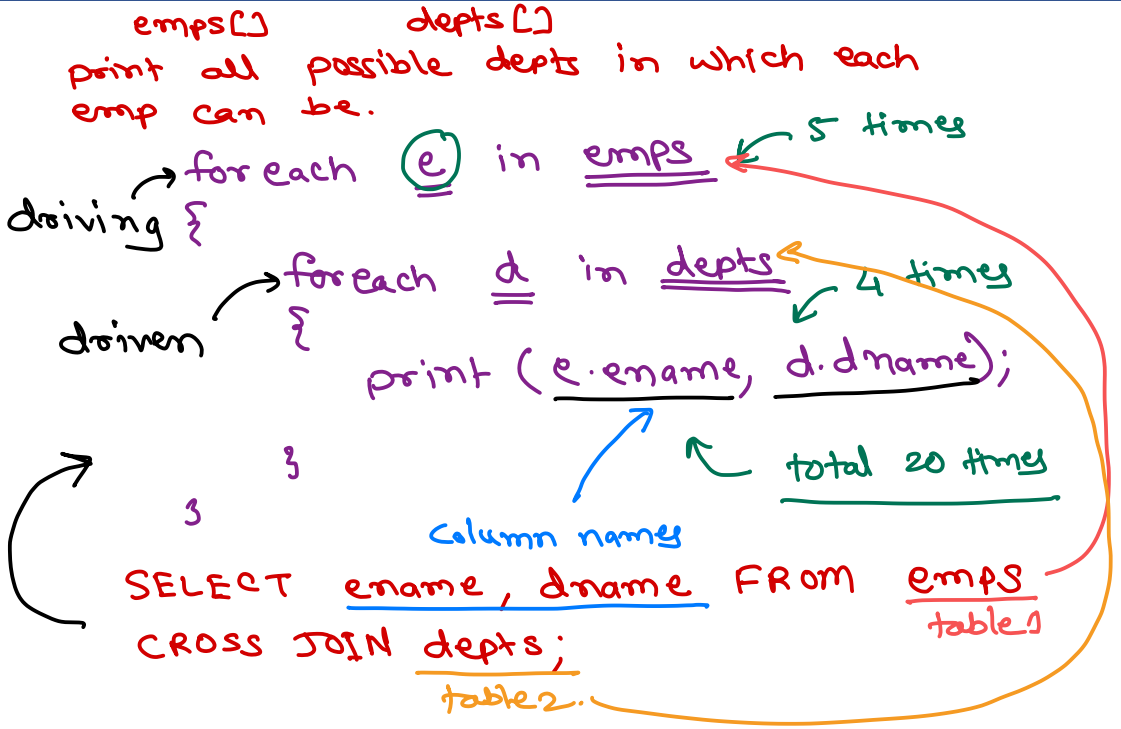
- Join statements are used to SELECT data from multiple tables using single query.
- Typical RDBMS supports following types of joins:
  - Cross Join
  - Inner Join
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join
  - Self join



Cross Join

deptno	dname
10	DEV
20	QA
30	OPS
40	ACC

empno	ename	deptno
1	Amit	10
2	Rahul	10
3	Nilesh	20
4	Nitin	50
5	Sarang	50

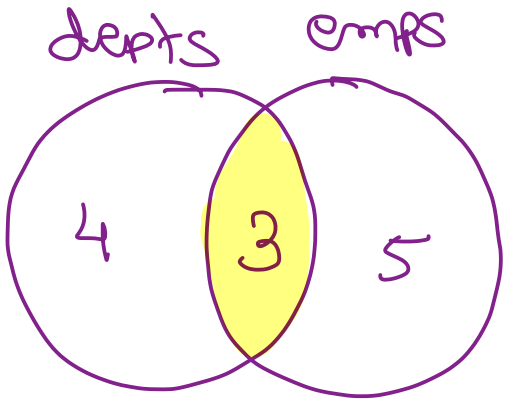


- Compares each row of Table1 with every row of Table2.
- Yields all possible combinations of Table1 and Table2.
- In MySQL, The larger table is referred as "Driving Table", while smaller table is referred as "Driven Table". Each row of Driving table is combined with every row of Driven table.
- Cross join is the fastest join, because there is no condition check involved.

Inner Join

deptno	dname
10	DEV
20	QA
30	OPS
40	ACC

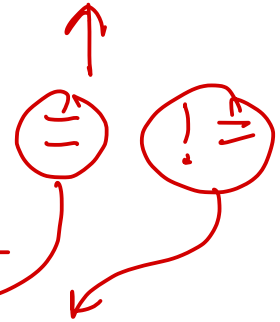
empno	ename	deptno
1	Amit	10
2	Rahul	10
3	Nilesh	20
4	Nitin	50
5	Sarang	50




```
foreach e in emps
{
    foreach d in depts
    {
        if (e.deptno == d.deptno)
            print (e.ename, d.dname);
    }
}
```

Columns

```
SELECT e.ename, d.dname FROM emps e
INNER JOIN depts d ON e.deptno = d.deptno;
```



- The inner JOIN is used to return rows from both tables that satisfy the join condition.
- Non-matching rows from both tables are skipped.
- If join condition contains equality check, it is referred as equi-join; otherwise it is non-equi-join.

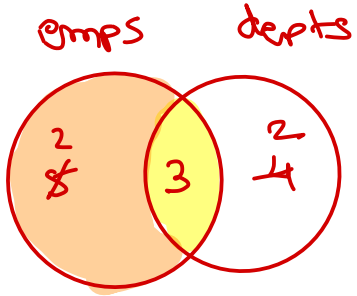
DAY 13

Left Outer Join

deptno	dname
10	DEV
20	QA
30	OPS
40	ACC

empno	ename	deptno
1	Amit	10
2	Rahul	10
3	Nilesh	20
4	Nitin	50
5	Sarang	50

```
foreach e in emps
{
    found = false;
    foreach d in depts
    {
        if (e.deptno == d.deptno) {
            print (e.ename, d.dname);
            found = true;
        }
    }
    if ( found == false)
        print (e.ename, NULL);
}
```




```
Select e.ename, d.dname from emps e
left outer join depts d on e.deptno = d.deptno;
```

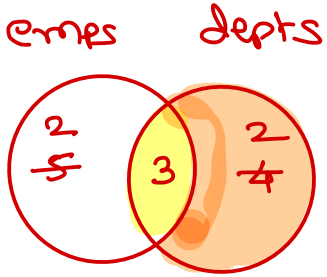
(left = emps e; right = depts d)

- Left outer join is used to return matching rows from both tables along with additional rows in left table.
- Corresponding to additional rows in left table, right table values are taken as NULL.
- OUTER keyword is optional.

Right Outer Join

deptno	dname
10	DEV
20	QA
30	OPS
40	ACC

empno	ename	deptno
1	Amit	10
2	Rahul	10
3	Nilesh	20
4	Nitin	50
5	Sarang	50




```
Select e.ename, d.dname from emps e
right outer join depts d on e.deptno=d.deptno;
```

left

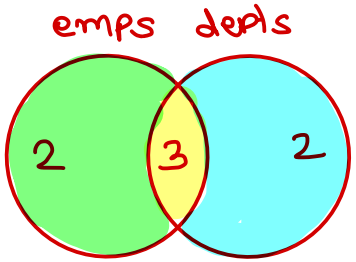
right

- Right outer join is used to return matching rows from both tables along with additional rows in right table.
- Corresponding to additional rows in right table, left table values are taken as NULL.
- OUTER keyword is optional.

Full Outer Join

deptno	dname
10	DEV
20	QA
30	OPS
40	ACC

empno	ename	deptno
1	Amit	10
2	Rahul	10
3	Nilesh	20
4	Nitin	50
5	Sarang	50

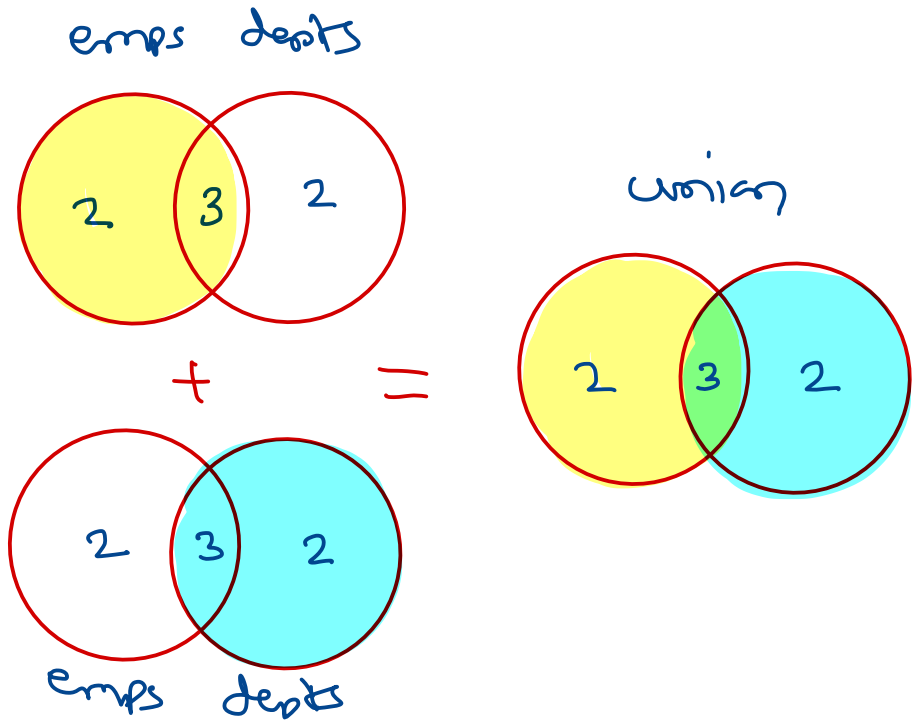


- Full join is used to return matching rows from both tables along with additional rows in both tables.
- Corresponding to additional rows in left or right table, opposite table values are taken as NULL.
- Full outer join is not supported in MySQL, but can be simulated using set operators.

Set operators

ename	dname
Amit	DEV
Rahul	DEV
Nilesh	QA
NULL	OPS
NULL	ACC

ename	dname
Amit	DEV
Rahul	DEV
Nilesh	QA
Nitin	NULL
Sarang	NULL



- UNION operator is used to combine results of two queries. The common data is taken only once. It can be used to simulate full outer join.
- UNION ALL operator is used to combine results of two queries. Common data is repeated.

Self Join

- When join is done on same table, then it is known as "Self Join". The both columns in condition belong to the same table.
- Self join may be an inner join or outer join.

empno	ename	deptno	mgr
1	Amit	10	4
2	Rahul	10	3
3	Nilesh	20	4
4	Nitin	50	5
5	Sarang	50	NULL

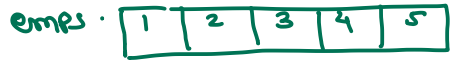
empno	ename	deptno	mgr
1	Amit	10	4
2	Rahul	10	3
3	Nilesh	20	4
4	Nitin	50	5
5	Sarang	50	NULL

emps e

Amit - Nitin
Rahul - Nilesh
Nilesh - Nitin
Nitin - Sarang

mgrs m

```
Select e.ename, m.ename from emps e
inner join emps m on e.mgr = m.empno;
```




```
foreach e in emps
{
    foreach m in emps
    {
        if(e.mgr == m.empno)
            print(e.ename, m.ename);
    }
}
```

DAY 14

DDL – ALTER statement

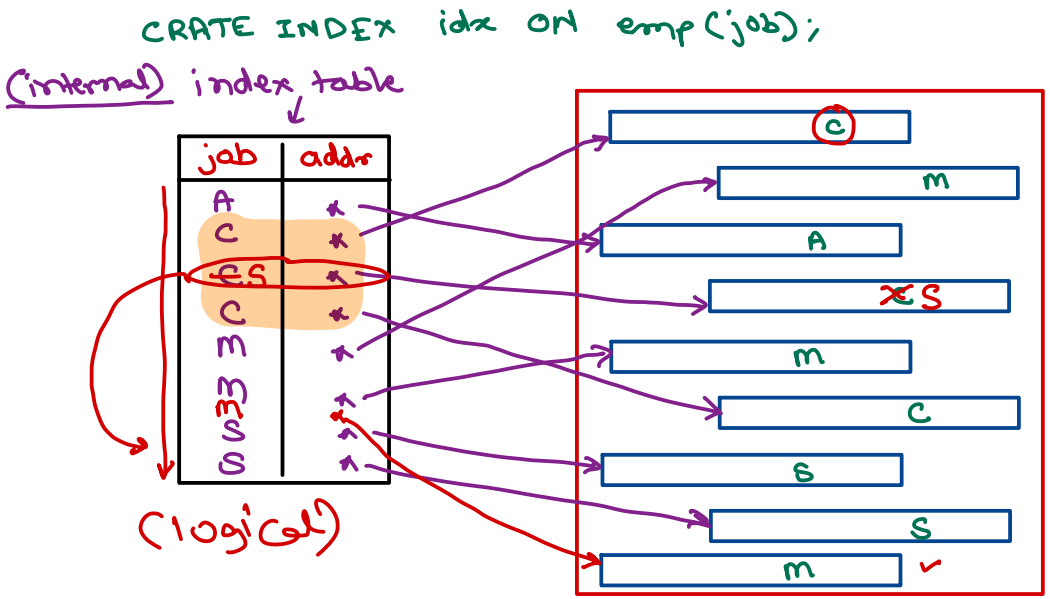
- ALTER statement is used to do modification into table, view, function, procedure, …
- ALTER TABLE is used to change table structure.
- Add new column(s) into the table.
  - ALTER TABLE table ADD col TYPE;
  - ALTER TABLE table ADD c1 TYPE, c2 TYPE;
- Modify column of the table.
  - ALTER TABLE table MODIFY col NEW_TYPE;
- Rename column.
  - ALTER TABLE CHANGE old_col new_col TYPE;
- Drop a column
  - ALTER TABLE DROP COLUMN col;
- Rename table
  - ALTER TABLE table RENAME TO new_table;

Using ALTER TABLE on production db is bad practice / strictly prohibited.

Index

- Index enable faster searching in tables by indexed columns.
  - CREATE INDEX idx_name ON table(column);
- One table can have multiple indexes on different columns/order.
- Typically indexes are stored as some data structure (like BTREE or HASH) on disk.
- Indexes are updated during DML operations. So DML operation are slower on indexed tables.

automatically



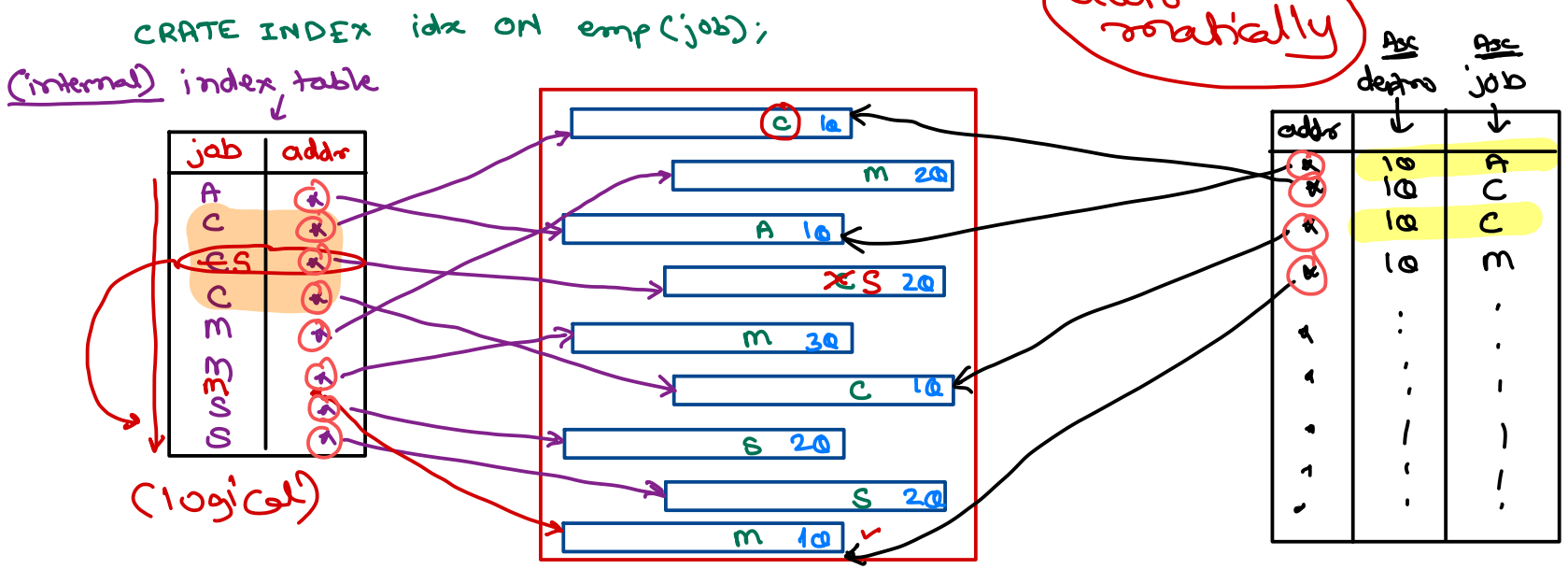
Query performance

- Few RDBMS features ensure better query performance.
  - Index speed up execution of SELECT queries (search operations).
  - Correlated sub-queries execute faster.
- Query performance can observed using EXPLAIN statement.
  - EXPLAIN FORMAT=JSON SELECT …;
- EXPLAIN statement shows
  - Query cost (Lower is the cost, faster is the query execution).
  - Execution plan (Algorithm used to execute query e.g. loop, semi-join, materialization, etc).
- Optimizations can be enabled or disabled by optimizer_switch system variable.
  - SELECT @@optimizer_switch;
  - SET @@optimizer_switch='materialization=off';

DAY 15

Index

- Index enable faster searching in tables by indexed columns.
  - CREATE INDEX idx_name ON table(column);
- One table can have multiple indexes on different columns/order.
- Typically indexes are stored as some data structure (like BTREE or HASH) on disk.
- Indexes are updated during DML operations. So DML operation are slower on indexed tables.



Index

- Index can be ASC or DESC.
  - It cause storage of key values in respective order (MySQL 8.x onwards).
  - ASC/DESC index is used by optimizer on ORDER BY queries.
- There are four types of indexes:
  - Simple index
    - CREATE INDEX idx_name ON table(column [ASC|DESC]);
  - Unique index
    - CREATE UNIQUE INDEX idx_name ON table(column [ASC|DESC]);
    - Doesn't allow duplicate values.
  - Composite index
    - CREATE INDEX idx_name ON table(column1 [ASC|DESC], column2 [ASC|DESC]);
    - Composite index can also be unique. Do not allow duplicate combination of columns.
  - Clustered index
    - PRIMARY index automatically created on Primary key for row lookup.
    - If primary key is not available, hidden index is created on synthetic column.
    - It is maintained in tabular form and its reference is used in other indexes.

Index

- Indexes should be created on shorter (INT, CHAR, …) columns to save disk space.
- Few RDBMS do not allow indexes on external columns i.e. TEXT, BLOB.
- MySQL support indexing on TEXT/BLOB up to n characters.
  - CREATE TABLE test (blob_col BLOB, …, INDEX(blob_col(10)));
- To list all indexes on table:
  - SHOW INDEXES ~~ON~~ FROM table;
- To drop an index:
  - DROP INDEX idx_name ON table;
- When table is dropped, all indexes are automatically dropped.
- Indexes should not be created on the columns not used frequent search, ordering or grouping operations.
- Columns in join operation should be indexed for better performance.

Query performance

- Few RDBMS features ensure better query performance.
  - Index speed up execution of SELECT queries (search operations).
  - Correlated sub-queries execute faster.
- Query performance can observed using EXPLAIN statement.
  - EXPLAIN FORMAT=JSON SELECT …;
- EXPLAIN statement shows
  - Query cost (Lower is the cost, faster is the query execution).
  - Execution plan (Algorithm used to execute query e.g. loop, semi-join, materialization, etc).
- Optimizations can be enabled or disabled by optimizer_switch system variable.
  - SELECT @@optimizer_switch;
  - SET @@optimizer_switch='materialization=off';

Constraints → DDL → Integrity/Validity of data.

- Constraints are restrictions imposed on columns. → restricts values to be added in the column.
- There are five constraints
  - NOT NULL → Col
  - UNIQUE → Col, Tbl
  - PRIMARY KEY → Col, Tbl
  - FOREIGN KEY → Col, Tbl
  - CHECK → Col, Tbl

  OR given while creating table. / given later using ALTER TABLE.
- Few constraints can be applied at either column level or table level. Few constraints can be applied on both.
- Optionally constraint names can be mentioned while creating the constraint. If not given, it is auto-generated.
- Each DML operation check the constraints before manipulating the values. If any constraint is violated, error is raised. → slower.

Constraints

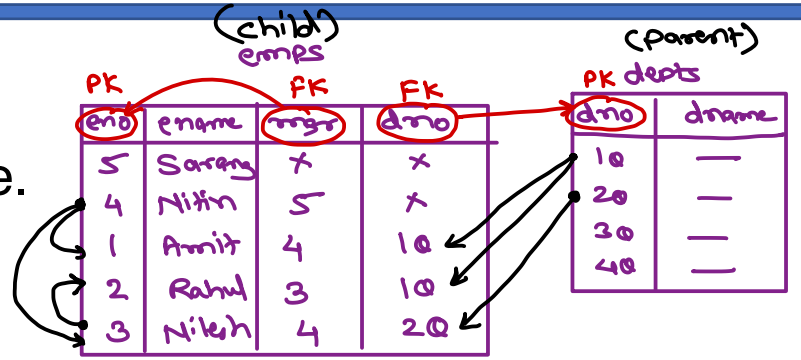
- NOT NULL
  - NULL values are not allowed.
  - Can be applied at column level only.
  - CREATE TABLE table(c1 TYPE NOT NULL, …);
- UNIQUE
  - Duplicate values are not allowed.
  - NULL values are allowed.
  - Not applicable for TEXT and BLOB.
  - UNIQUE can be applied on one or more columns.
  - Internally creates unique index on the column (fast searching).
  - Can be applied at column level or table level.
    - CREATE TABLE table(c1 TYPE UNIQUE, …);
    - CREATE TABLE table(c1 TYPE, …, UNIQUE(c1));
    - CREATE TABLE table(c1 TYPE, …, CONSTRAINT constraint_name UNIQUE(c1));

Constraints

- PRIMARY KEY
  - Column or set of columns that uniquely identifies a row.
  - Only one primary key is allowed for a table.
  - Primary key column cannot have duplicate or NULL values.
  - Internally index is created on PK column. → Primary Index (Clustered)
  - TEXT/BLOB cannot be primary key.
  - If no obvious choice available for PK, composite or surrogate PK can be created.
  - Creating PK for a table is a good practice.
  - PK can be created at table level or column level.
  - CREATE TABLE table(c1 TYPE PRIMARY KEY, …);
  - CREATE TABLE table(c1 TYPE, …, PRIMARY KEY(c1));
  - CREATE TABLE table(c1 TYPE, …, CONSTRAINT constraint_name PRIMARY KEY(c1));
  - CREATE TABLE table(c1 TYPE, c2 TYPE, …, PRIMARY KEY(c1, c2));

Constraints

- FOREIGN KEY
  - Column or set of columns that references a column of some table.
  - If column belongs to the same table, it is "self referencing".
  - Foreign key constraint is specified on child table column.
  - FK can have duplicate values as well as null values.
  - FK constraint is applied on column of child table (not on parent table).
  - Child rows cannot be deleted, until parent rows are deleted.
  - MySQL have ON DELETE CASCADE clause to ensure that child rows are automatically deleted, when parent row is deleted. ON UPDATE CASCADE clause does same for UPDATE operation.
  - By default foreign key checks are enabled. They can be disabled by
    - SET @@foreign_key_checks = 0;
  - FK constraint can be applied on table level as well as column level.
  - CREATE TABLE child(c1 TYPE, …, FOREIGN KEY (c1) REFERENCES parent(col))

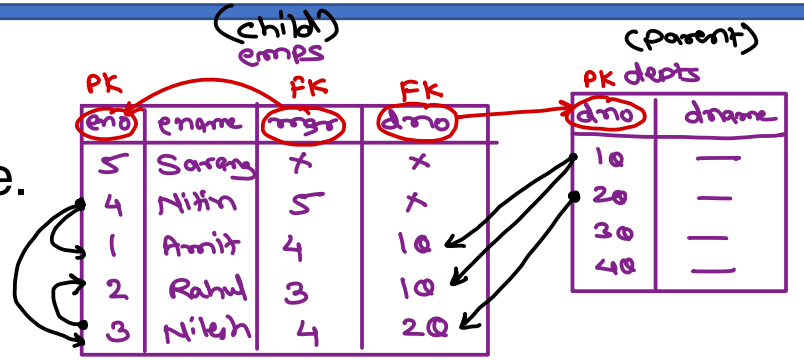


DAY 16

Constraints

Data Integrity

- FOREIGN KEY
  - Column or set of columns that references a column of some table.
  - If column belongs to the same table, it is "self referencing".
  - Foreign key constraint is specified on child table column.
  - FK can have duplicate values as well as null values.
  - FK constraint is applied on column of child table (not on parent table).
  - Child rows cannot be deleted, until parent rows are deleted.
  - MySQL have ON DELETE CASCADE clause to ensure that child rows are automatically deleted, when parent row is deleted. ON UPDATE CASCADE clause does same for UPDATE operation.
  - By default foreign key checks are enabled. They can be disabled by
    - SET @@foreign_key_checks = 0; → No checks will be performed while DML. So parent records can be inserted, update & deleted directly. This is done only in special cases like DB backup.
  - FK constraint can be applied on table level as well as column level.
  - CREATE TABLE child(c1 TYPE, …, FOREIGN KEY (c1) REFERENCES parent(col))



Constraints

- CHECK
  - CHECK is integrity constraint in SQL.
  - CHECK constraint specifies condition on column.
  - Data can be inserted/updated only if condition is true; otherwise error is raised.
  - CHECK constraint can be applied at table level or column level.
  - CREATE TABLE table(c1 TYPE, c2 TYPE CHECK condition1, …, CHECK condition2);

Sub queries

- Sub-query is query within query. Typically it work with SELECT statements.
- Output of inner query is used as input to outer query.
- If no optimization is enabled, for each row of outer query result, sub-query is executed once. This reduce performance of sub-query.
- Single row sub-query
  - Sub-query returns single row.
  - Usually it is compared in outer query using relational operators.

Sub queries

- Multi-row sub-query
  - Sub-query returns multiple rows.
  - Usually it is compared in outer query using operators like IN, ANY or ALL.
    - IN operator checks for equality with results from sub-queries.
    - ANY operator compares with one of the result from sub-queries.
    - ALL operator compares with all the results from sub-queries.

Sub queries

- Correlated sub-query
  - If number of results from sub-query are reduced, query performance will increase.
  - This can be done by adding criteria (WHERE clause) in sub-query based on outer query row.
  - Typically correlated sub-query use IN, ALL, ANY and EXISTS operators.

doesn't Compare inner query result with outer row.

it only checks if inner query is returning any row. (cnt > 0).