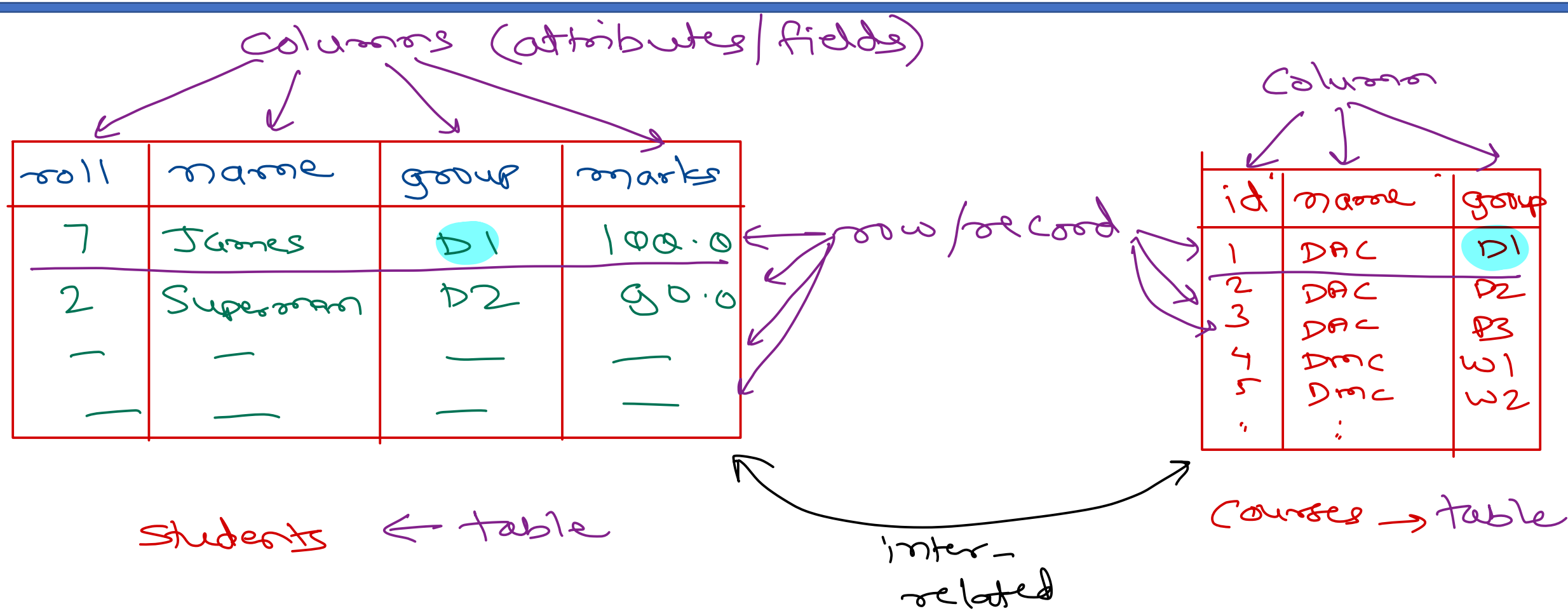
RDBMS & SQL Introduction

Trainer: Mr. Nilesh Ghule

DBMS

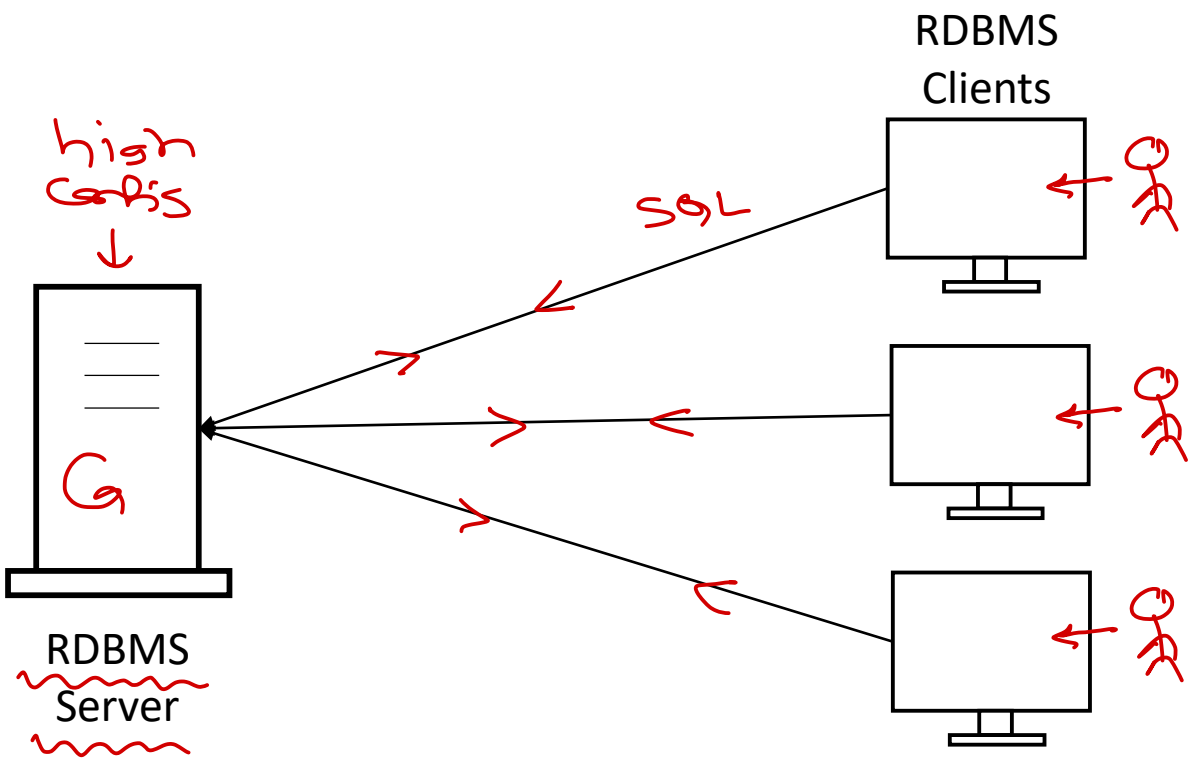
- Any enterprise application need to manage data.
- In early days of software development, programmers store data into files and does operation on it. However data is highly application specific.
- Even today many software manage their data in custom formats e.g. Tally, Address book, etc.
- As data management 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 table

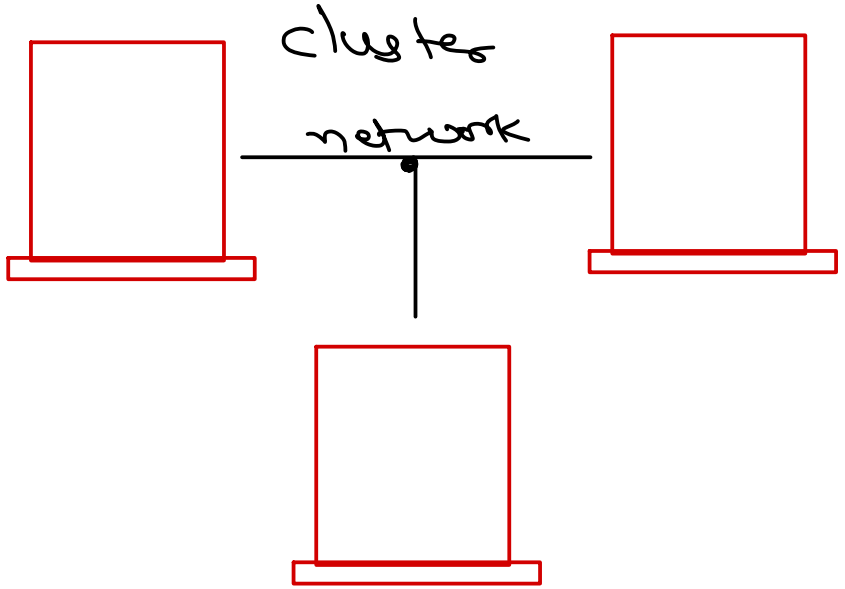

Columns (attributes/fields)
roll
name
group
marks
7
James
D1
100.0
2
Superman
D2
90.0
Students ← table
row/record
Column
id
name
group
1
DAC
D1
2
DAC
D2
3
DAC
B3
4
DMC
W1
5
DMC
W2
Courses → table
inter-related

RDBMS

- 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, …
- RDBMS design is based on Codd's rules developed at IBM (in 1970).



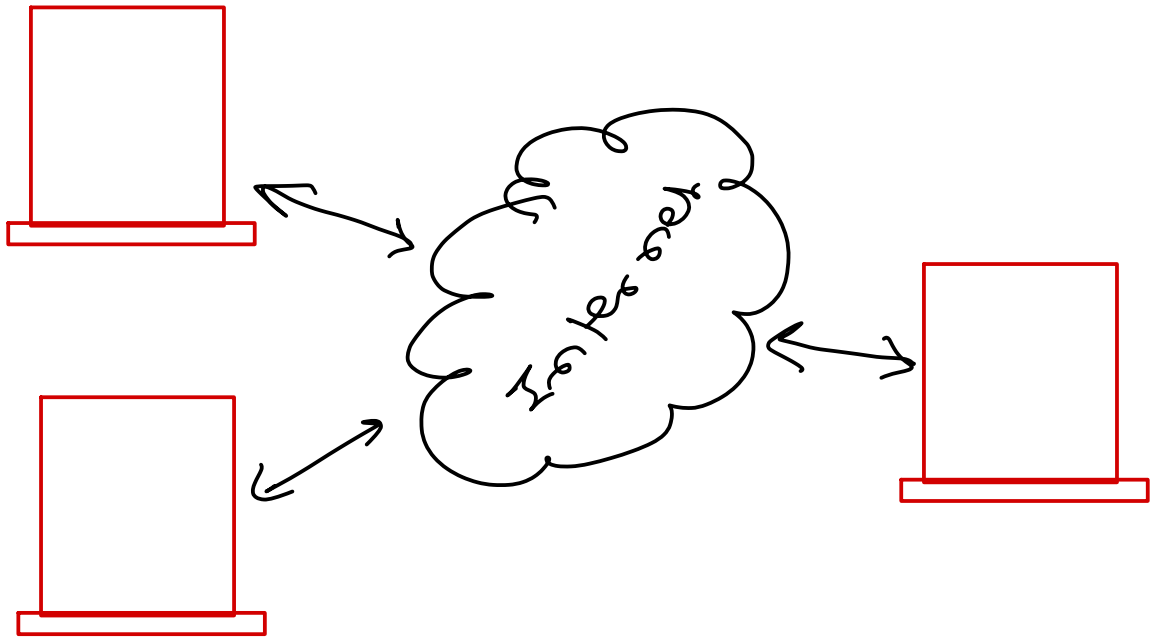
clustering - distributed systems.



eg. database cluster
webserver cluster.

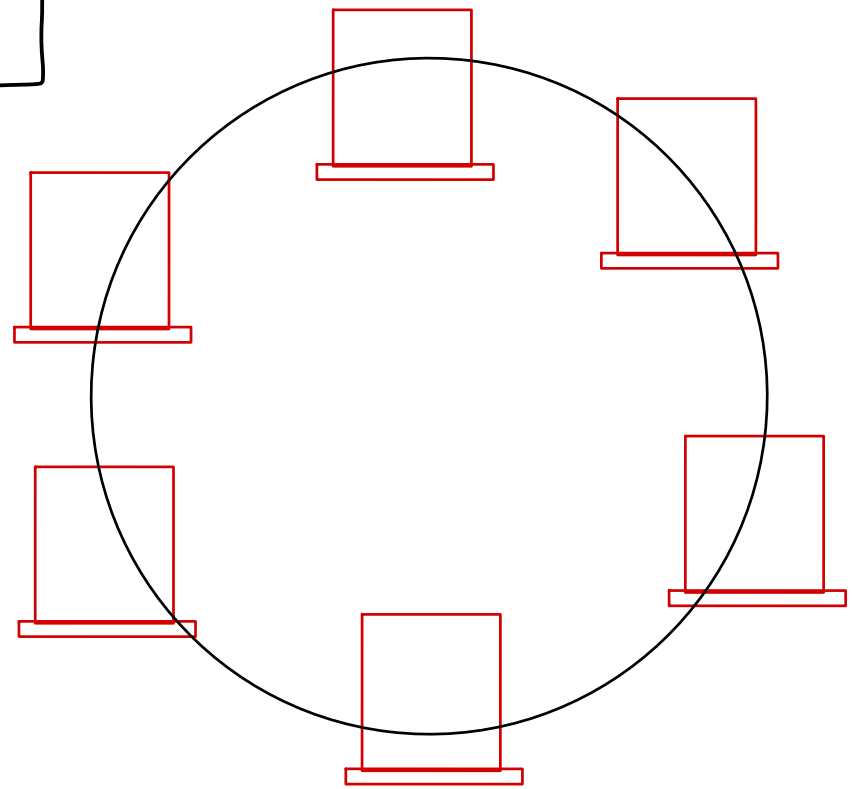
cluster is set of computers connected in a network for a specific task.

Internet



Internet

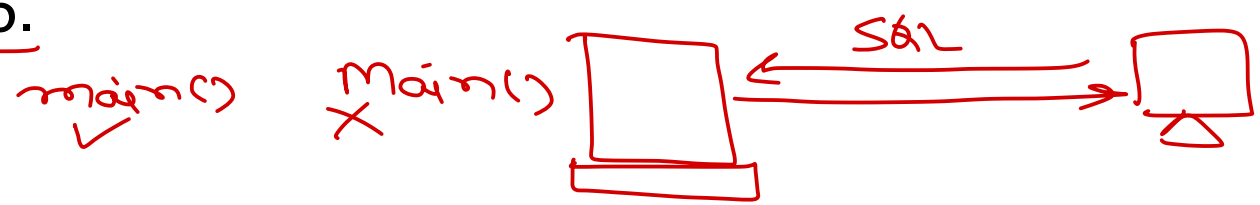
Star
Ring
Bus



LAN, MAN, WAN

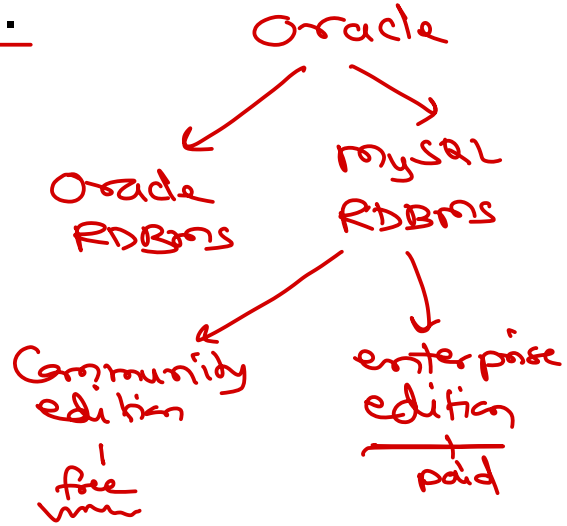
SQL - Structured Query Language.

- 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 - RDBMS Software → (of Oracle)

- 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

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

RDBMS & SQL Introduction

Trainer: Mr. Nilesh Ghule

developed in C/C++

mysql -u root -h localhost -p ← Password
- u: user
- h: host/server

mysql CLI

MySQL server
mysqld

Socket → IP + Port
Common endpoint in a network

mysql

mysql shell

mysql workbench

1) accept SQL query
2) process query
   - read data from disk
   - process data
   - create result
3) return result

port = 3306

port = 33060

disk

mysql daemon
↓
no GUI ← run in background

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

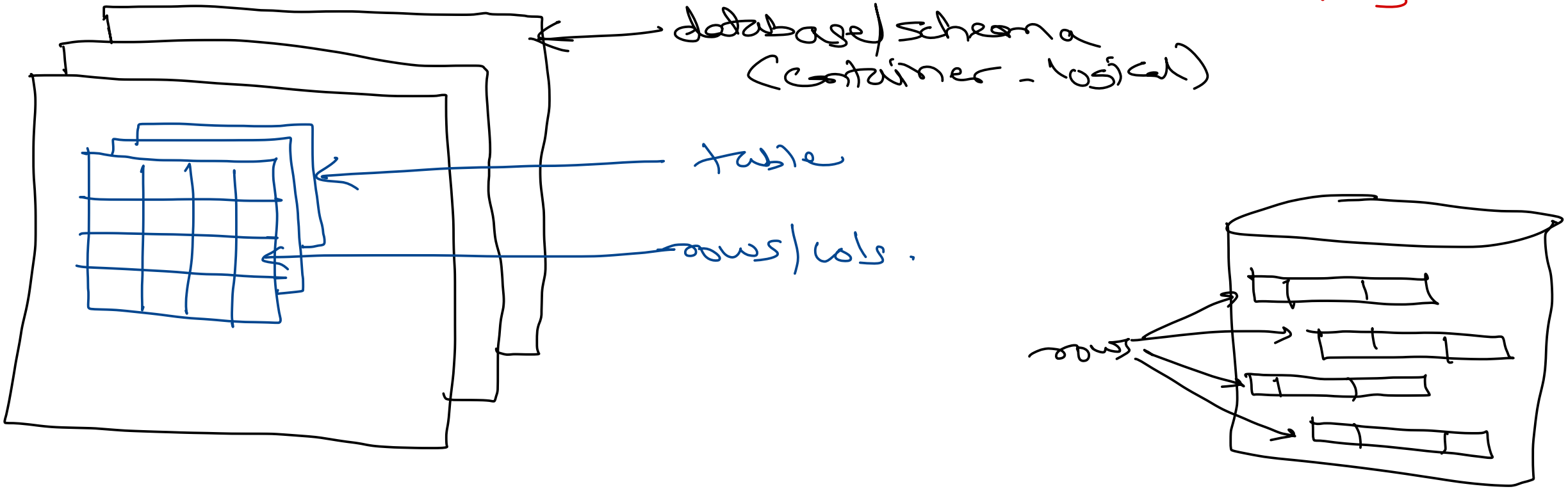
- root login can be used to perform CRUD as well as admin operations.
- It is recommended to create users for performing non-admin tasks.
  - mysql> CREATE DATABASE db;
  - mysql> SHOW DATABASES;
  - mysql> CREATE USER dbuser@localhost IDENTIFIED BY 'dbpass';
  - mysql> SELECT user, host FROM mysql.user;
  - 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;

Database logical layout

- 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).

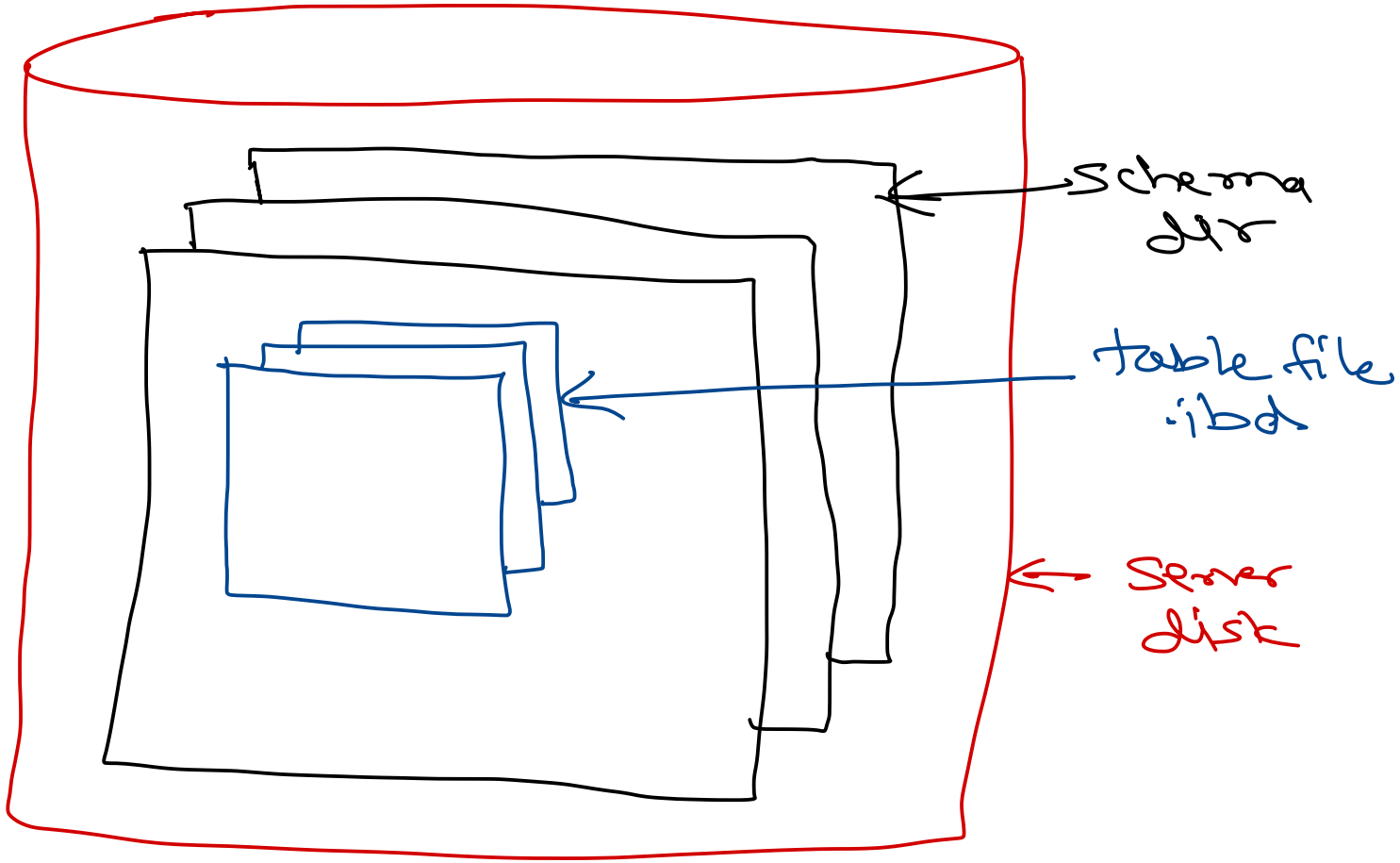
logical

physical



Database physical layout

- In MySQL, the data is stored on disk in its data directory i.e. /var/lib/mysql
- 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.


Schema dir
table file .ibd
Server disk

Thank you!

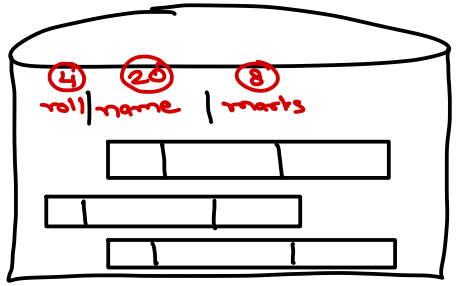
Nilesh Ghule <nilesh@sunbeaminfo.com>

RDBMS & SQL Introduction

Trainer: Mr. Nilesh Ghule

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) I DECIMAL(m, n) – exact precision
  - Date/Time types
    - DATE, TIME, DATETIME, TIMESTAMP, YEAR(1)
  - 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



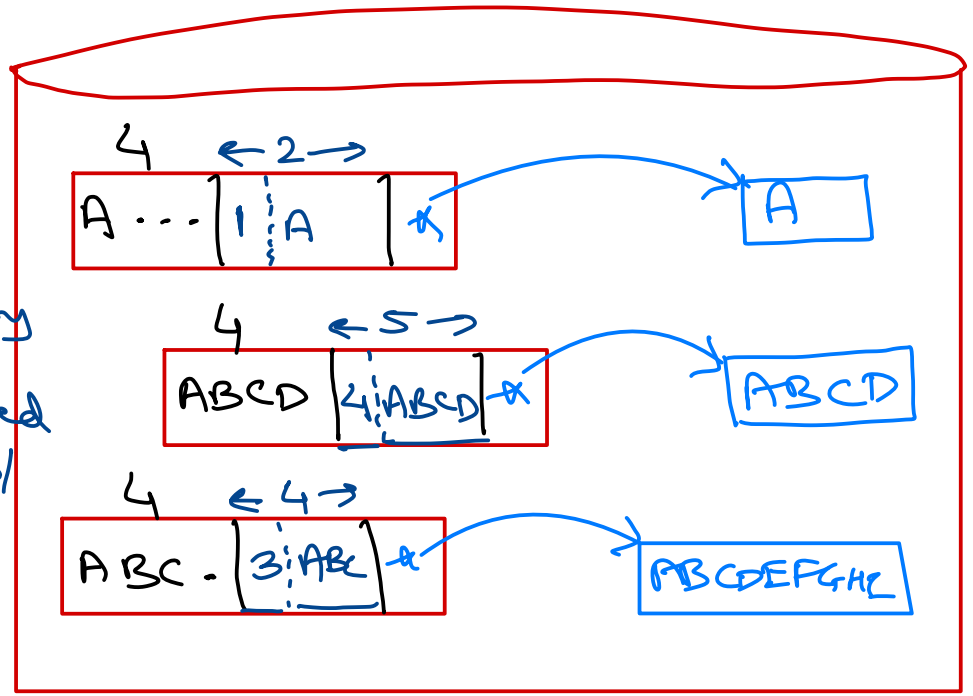
CHAR vs VARCHAR vs TEXT

String / date/time } data must be in '____';

- 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. → to record.
  - Very slow access.
  - Not ideal for indexing. → we will learn in indexes lecture.

char encoding

each char → how many bytes
→ how stored in mem/disk?



Server disk

- CREATE TABLE temp(c1 CHAR(4), c2 VARCHAR(4), c3 TEXT(4));
- DESC temp;
- INSERT INTO temp VALUES('abcd', 'abcd', 'abcdef');

Numerical data is not enclosed in '____'.

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.

Thank you!

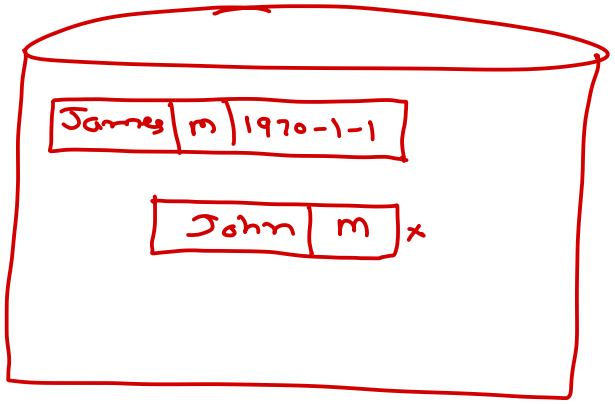
Nilesh Ghule <nilesh@sunbeaminfo.com>

RDBMS & SQL Introduction

Trainer: Mr. Nilesh Ghule

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;

SELECT – DQL

- 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;

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

RDBMS & SQL Introduction

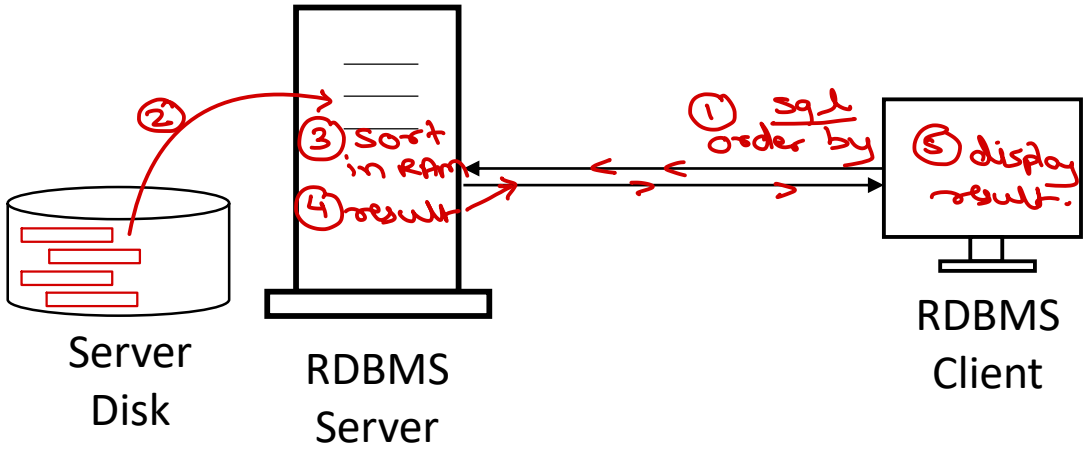
Trainer: Mr. Nilesh Ghule

SELECT – DQL

- Distinct values in column. → unique values.
  - 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;



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

operators
1) between
2) in
3) like
} monday lecture.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

RDBMS & SQL Introduction

Trainer: Mr. Nilesh Ghule

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.

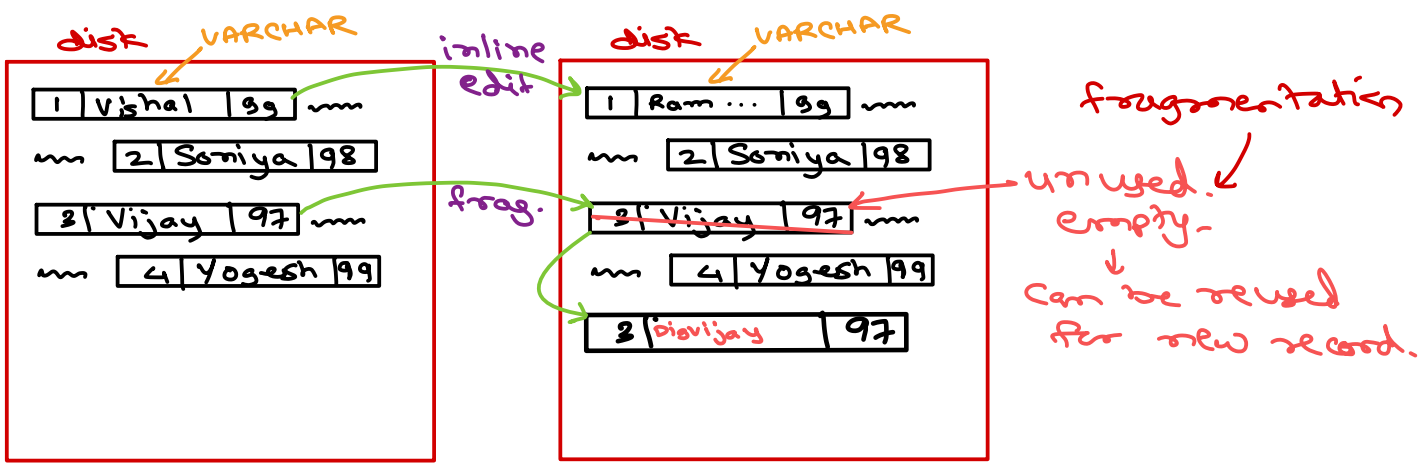
NOT BETWEEN

NOT IN

NOT LIKE

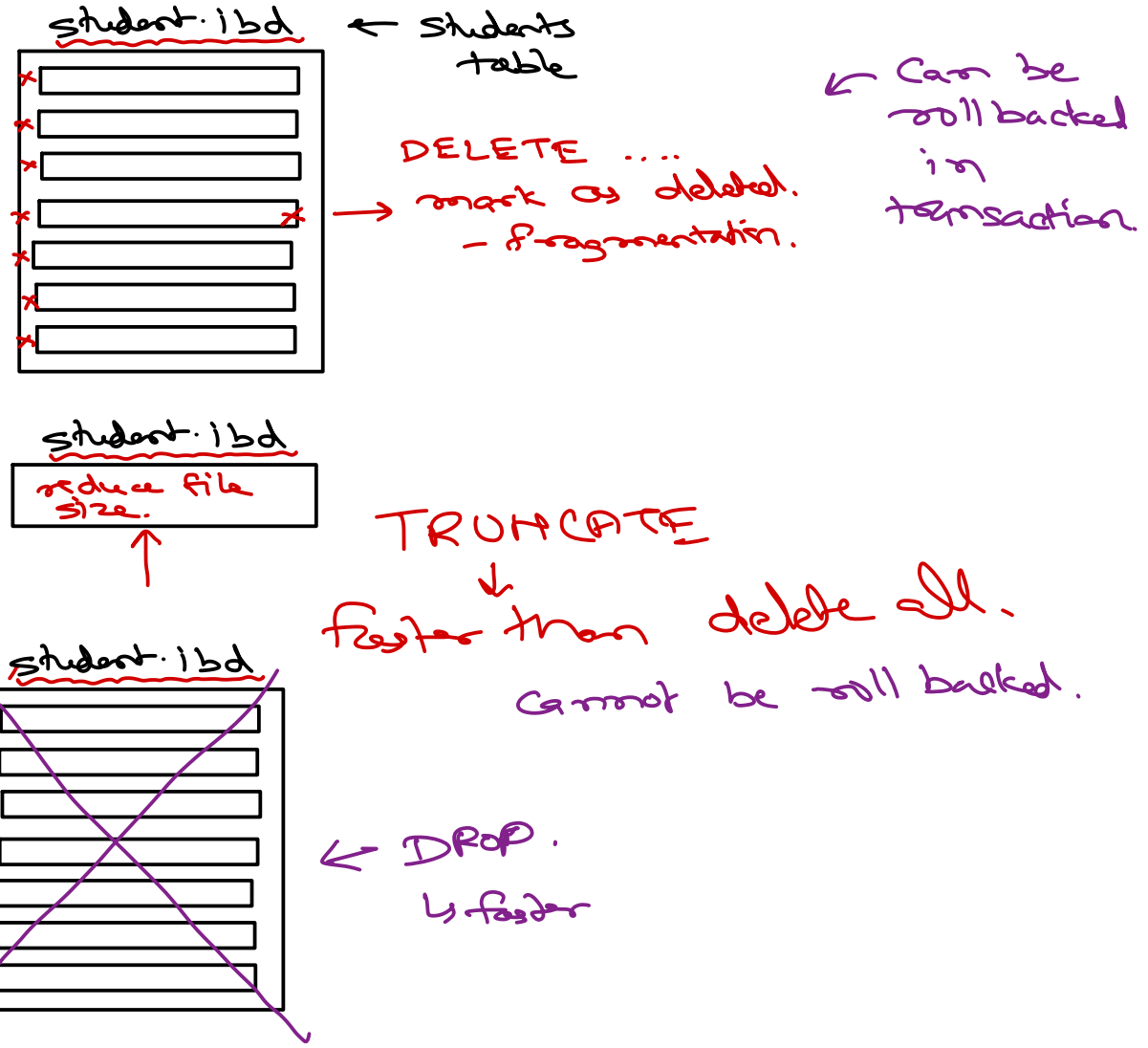
UPDATE – DML

- 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;



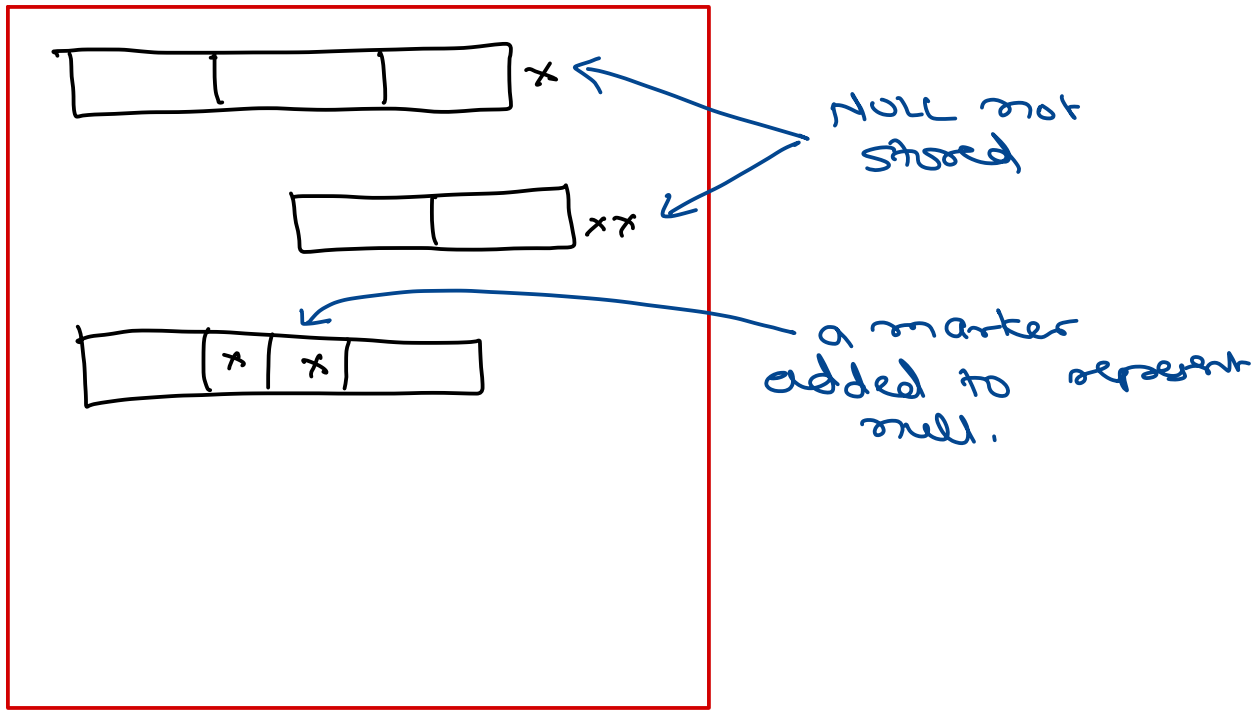
Thank you!

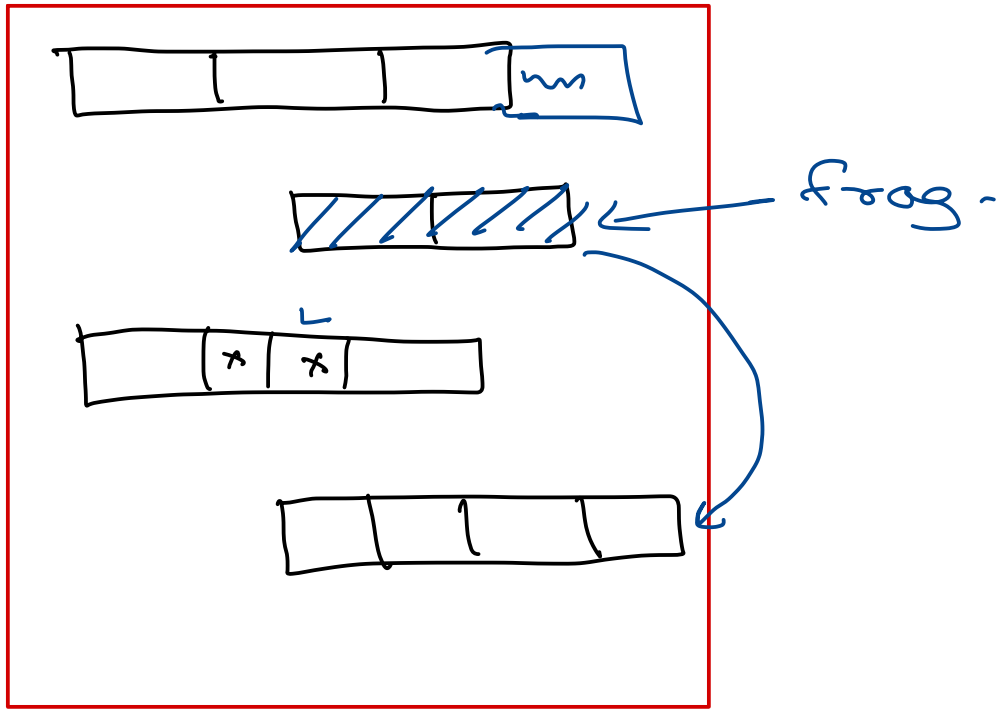
Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL RDBMS

Trainer: Mr. Nilesh Ghule

NULL





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

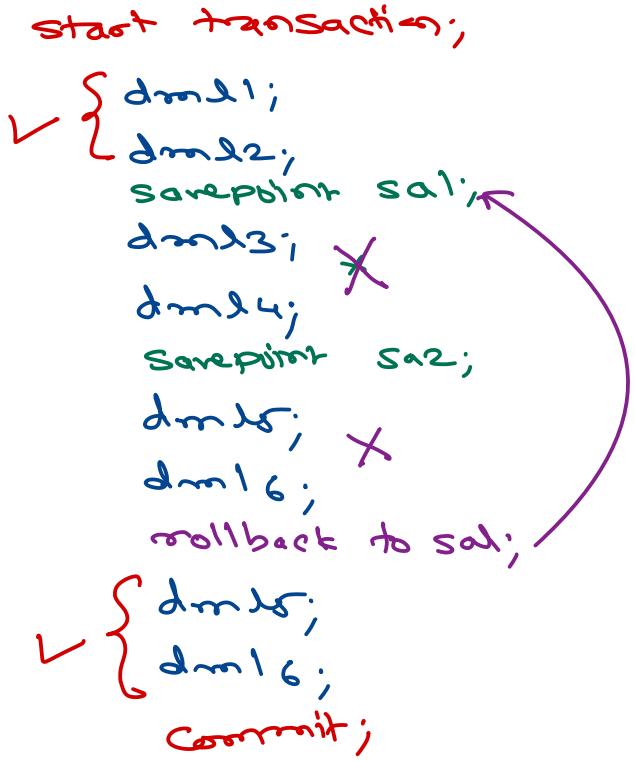
- Transaction management
  - START TRANSACTION;
  - …
  - COMMIT WORK;
  - START TRANSACTION;
  - …
  - ROLLBACK WORK;

WORK keyword optional.

- In MySQL autocommit variable is by default 1. So each DML command is auto-committed into database. → each query is one tx – auto commited.
  - 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".

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.
- 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.

Thank you!

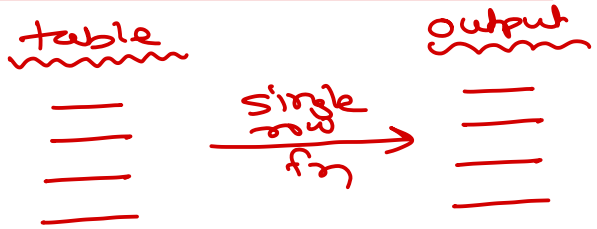
Nilesh Ghule <nilesh@sunbeaminfo.com>

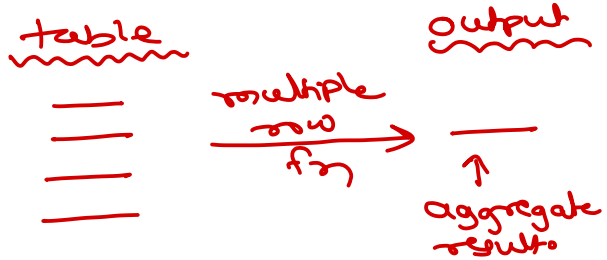
MySQL RDBMS

Trainer: Mr. Nilesh Ghule

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
    - Values from multiple rows are aggregated to single value.
    - e.g. SUM(), MIN(), MAX(), …





- 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)

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL RDBMS

Trainer: Mr. Nilesh Ghule

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()

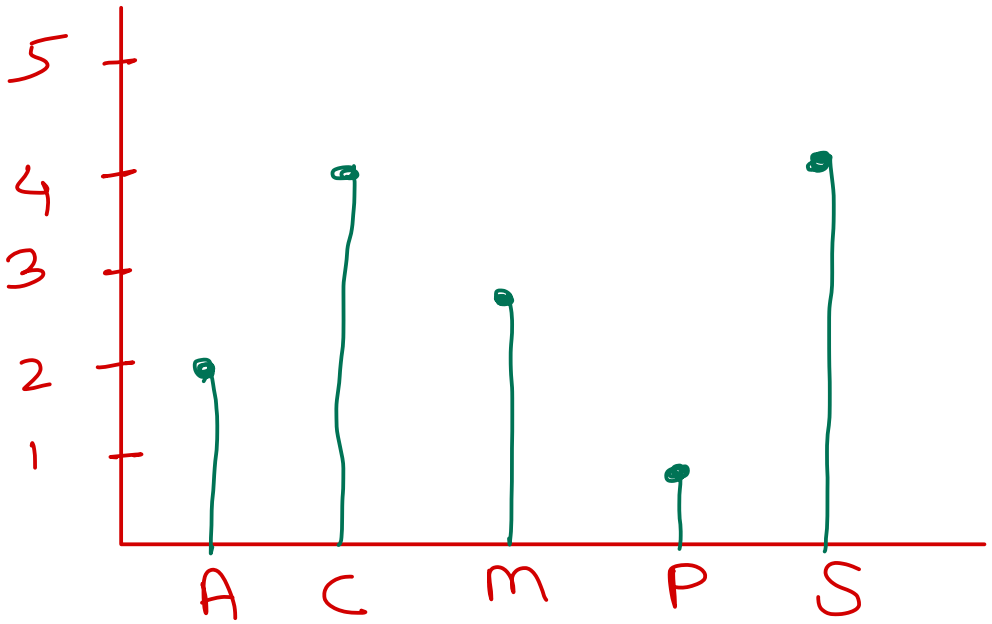
- IF(condition, true-value, false-value)

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.

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.


5
4
3
2
1
A
C
M
P
S

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL RDBMS

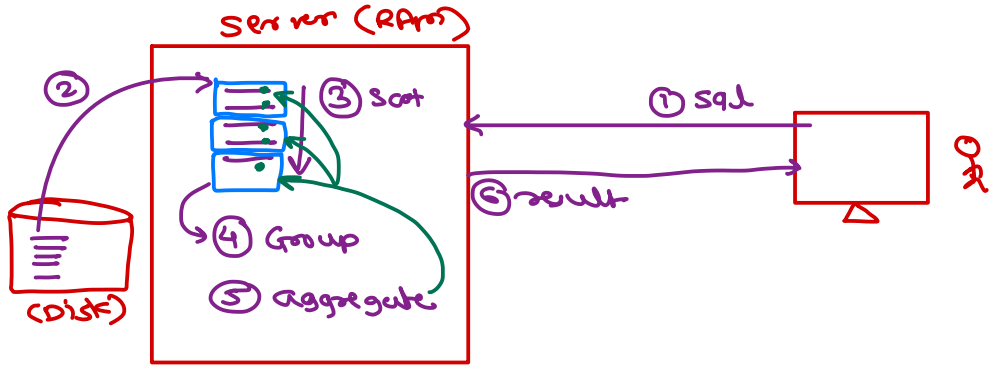
Trainer: Mr. Nilesh Ghule

GROUP BY clause

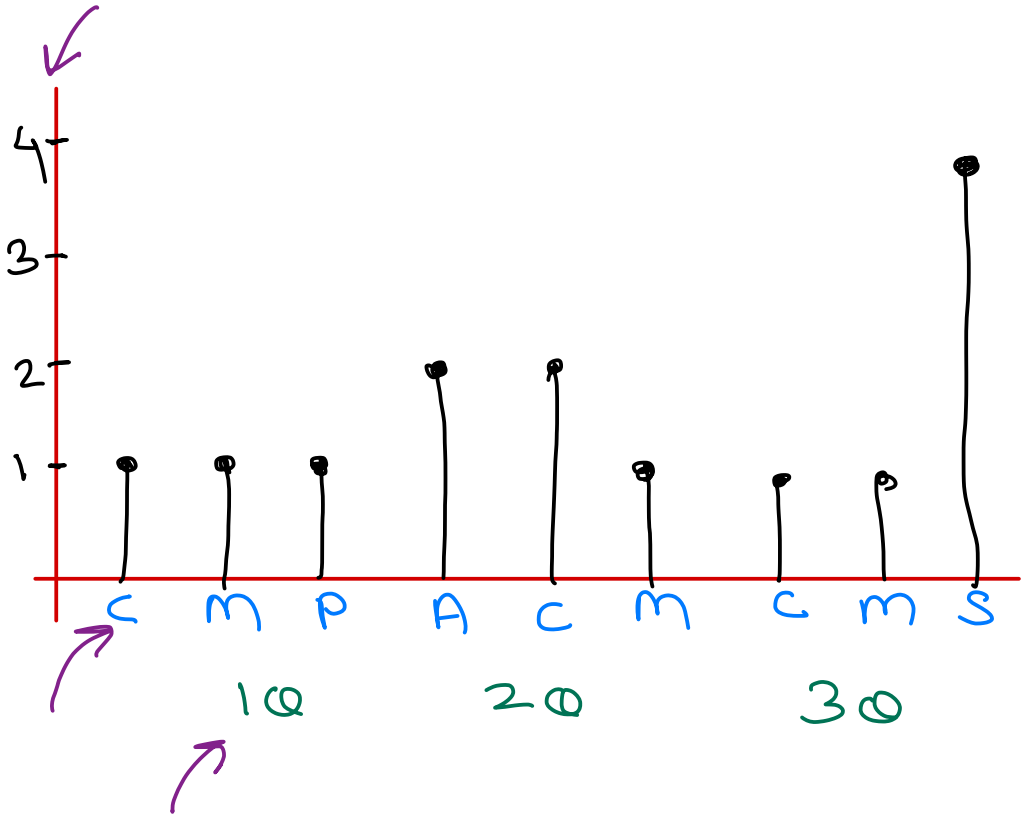
- GROUP BY is used for analysis of data i.e. generating reports & charts.
- When GROUP BY single column, generated output can be used to plot 2-D chart. When GROUP BY two column, generated output can be used to plot 3-D chart and so on.
- GROUP BY queries are also called as Multi-dimensional / Spatial 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.

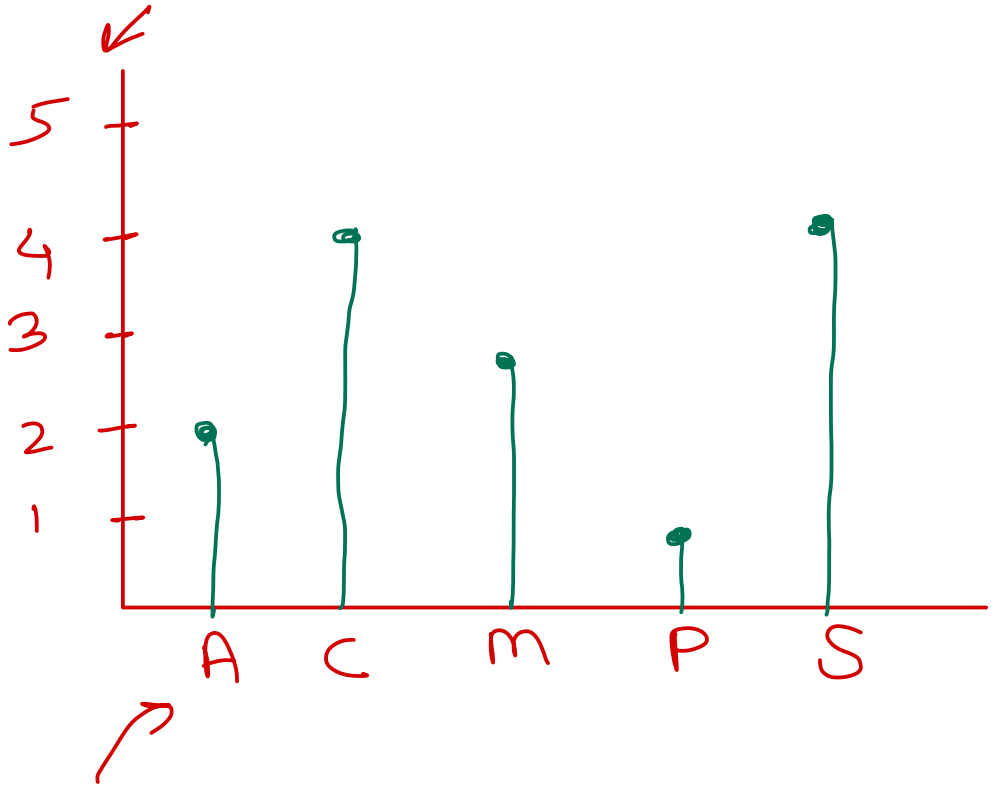
```
select job, count(empno) from emp group by job
having job in (M, A);
```

- 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 job, count(empno) from emp where job in (M, A)
group by job;
```


4
3
2
1
C M P A C M C M S
100
200
300


5
4
3
2
1
A C M P S

HAVING clause

- HAVING clause cannot be used without GROUP BY clause.
- HAVING clause is used to specify condition on aggregate values/fns.
- Examples:
  - SELECT deptno, SUM(sal) FROM EMP GROUP BY deptno HAVING SUM(sal) > 9000;
- Syntactical Characteristics:
  - WHERE clause executed for each record; while HAVING is executed for each group.
  - HAVING clause can be used to specify condition on group fn or grouped columns.
  - However using HAVING to specify condition of group col reduce the performance. Use WHERE clause for the same.
- Examples:
  - SELECT deptno, SUM(sal) FROM EMP GROUP BY deptno HAVING deptno = 20;
  - SELECT deptno, SUM(sal) FROM EMP WHERE deptno = 20 GROUP BY deptno; ← efficient.
- We may use GROUP BY with WHERE, ORDER BY & LIMIT.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL RDBMS

Trainer: Mr. Nilesh Ghule

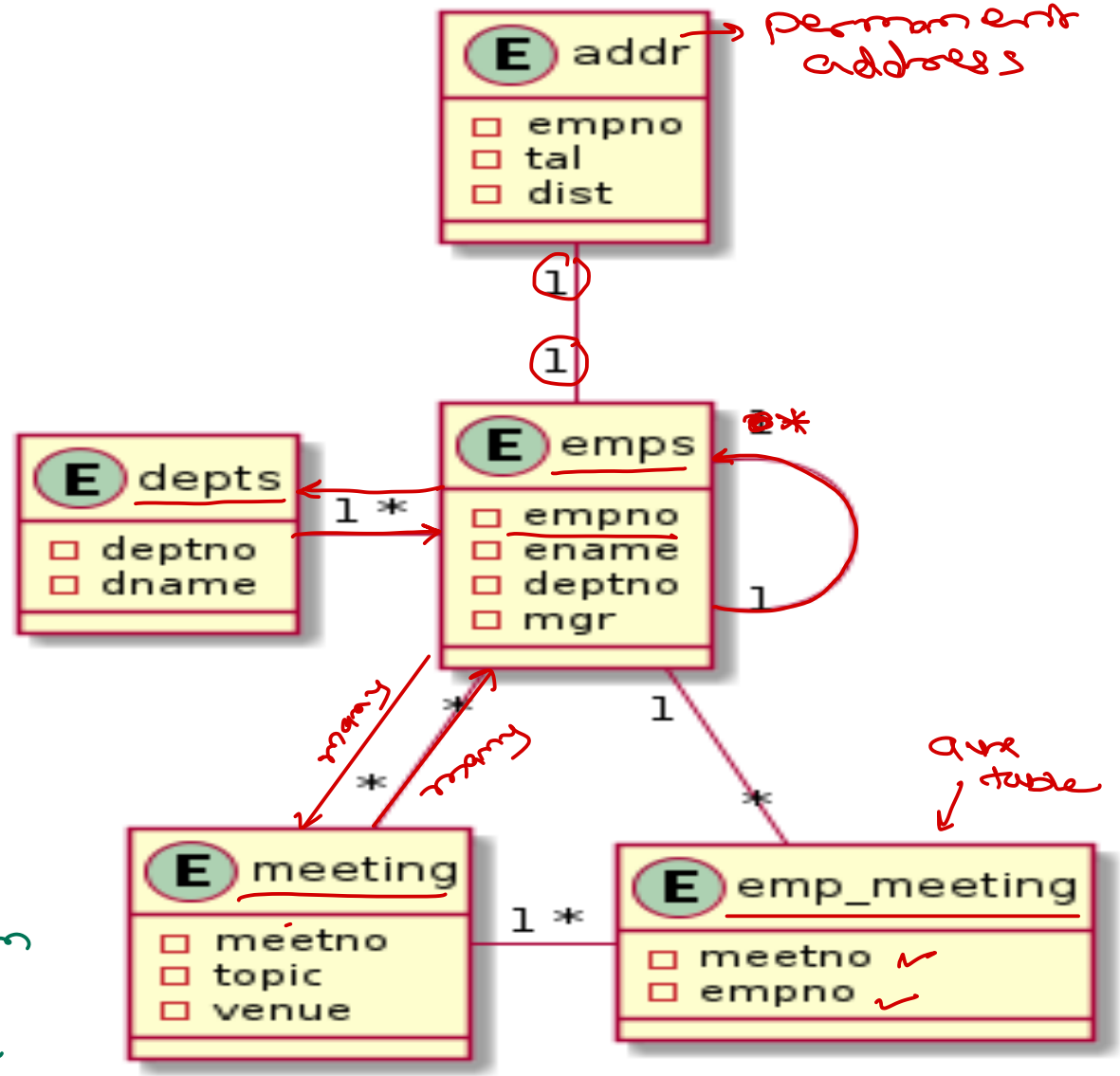
Entity Relations

ER - diagram

- 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
  - One to Many
  - Many to One
  - Many to Many
- Entity relations is outcome of Normalization process.

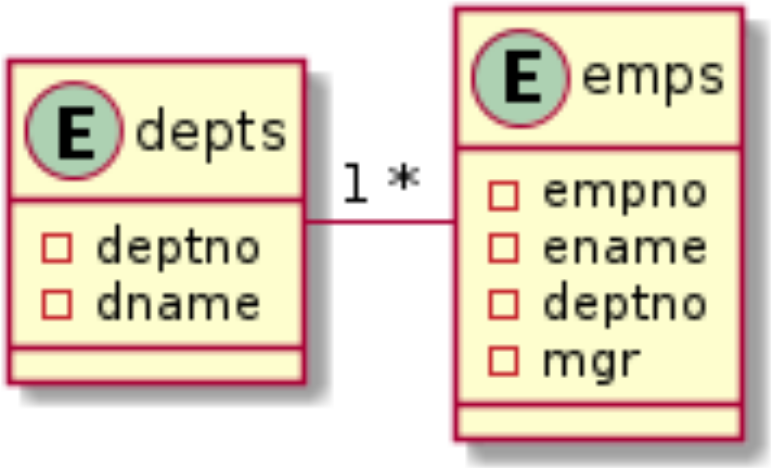
part of application design

1. req analysis
2. design — ui design, dto design, oo design
3. implementation
4. testing / maintence



SQL Joins

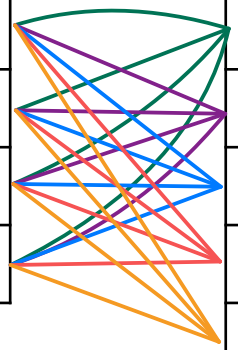
- Join statements are used to SELECT data from multiple tables using single query.
- Typical RDBMS supports following types of joins:
  - Cross Join
  - Inner Join
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join
  - Self join



Cross Join

depts (4)

deptno	dname
10	DEV
20	QA
30	OPS
40	ACC



(5) emps

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

outer (depts)

```
for (emp e : emps) {
    for (dept d : depts) {
        print (e.ename, d.dname);
    }
}
```

inner (emps)

```
select e.ename, d.dname from emps e
cross join depts d;
```

- 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 → common rows

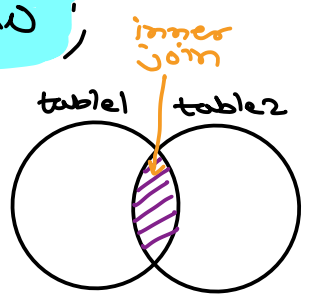
deptno	dname
10	DEV
20	QA
30	OPS
40	ACC

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

```
for(emp e : emps) {
    for(dept d : depts) {
        if(e.deptno == d.deptno)
            print(e.ename, d.dname);
    }
}
```

Amit DEV
Rahul DEV
Nilesh QA

```
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.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL RDBMS

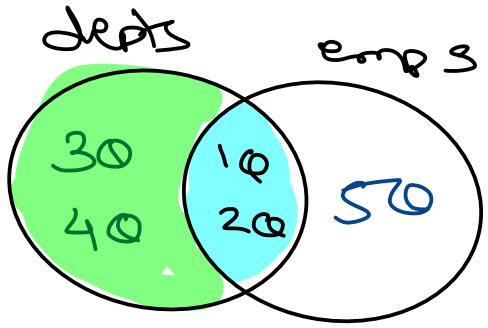
Trainer: Mr. Nilesh Ghule

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

```
for (dept d : depts) {
    found = 0;
    for (emp e : emps) {
        if (d.deptno == e.deptno) {
            print(e.ename, d.dname);
            found = 1;
        }
    }
    if (found == 0)
        print(NULL, d.dname);
}
```




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

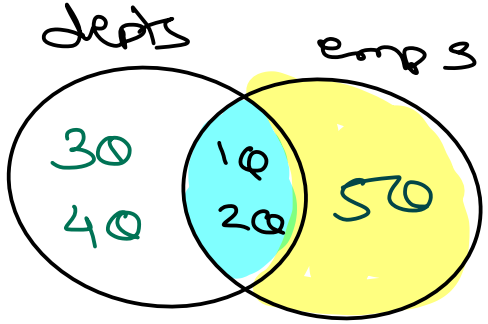
- Left outer join is used to return matching rows from both tables along with additional rows in left table.
- Corresponding to additional rows in left table, right table values are taken as NULL.
- OUTER keyword is optional.

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

```
for(emp e : emps) {
  found = 0;
  for (dept d : depts){
    if(d.deptno == e.deptno) {
      print(e.ename, d.dname);
      found = 1;
    }
  }
  if (found == 0)
    print(e.ename, NULL);
}
```




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

- 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

Left Join

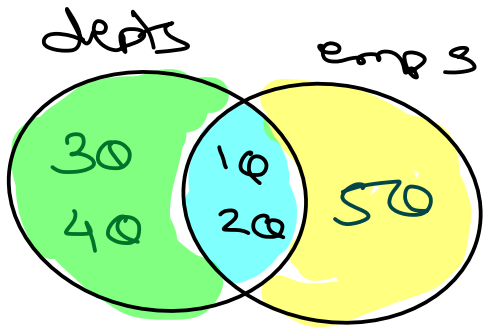
Amit Dev
Rahul Dev
Nilesh QA
X OPS
X ACC

Right Join

Amit Dev
Rahul Dev
Nilesh QA
Nitin X
Sarang X

Full Join

Amit Dev
Rahul Dev
Nilesh QA
X OPS
X ACC
Nitin X
Sarang X



- 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 – Combine output of two queries.

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

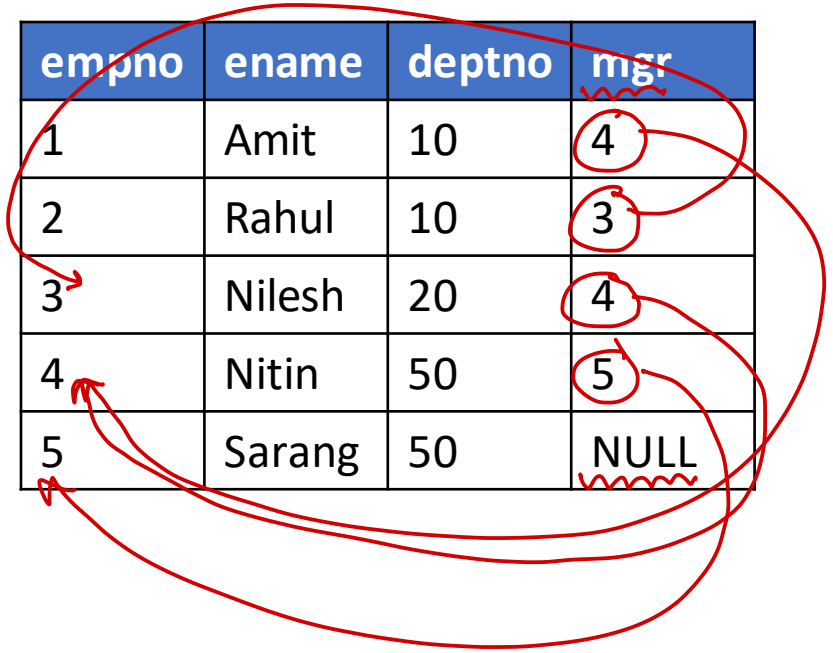
ename	dname
Amit	DEV
Rahul	DEV
Nilesh	QA
Nitin	NULL
Sarang	NULL

- 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



mgr column - represent empno of manager of current emp.

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

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

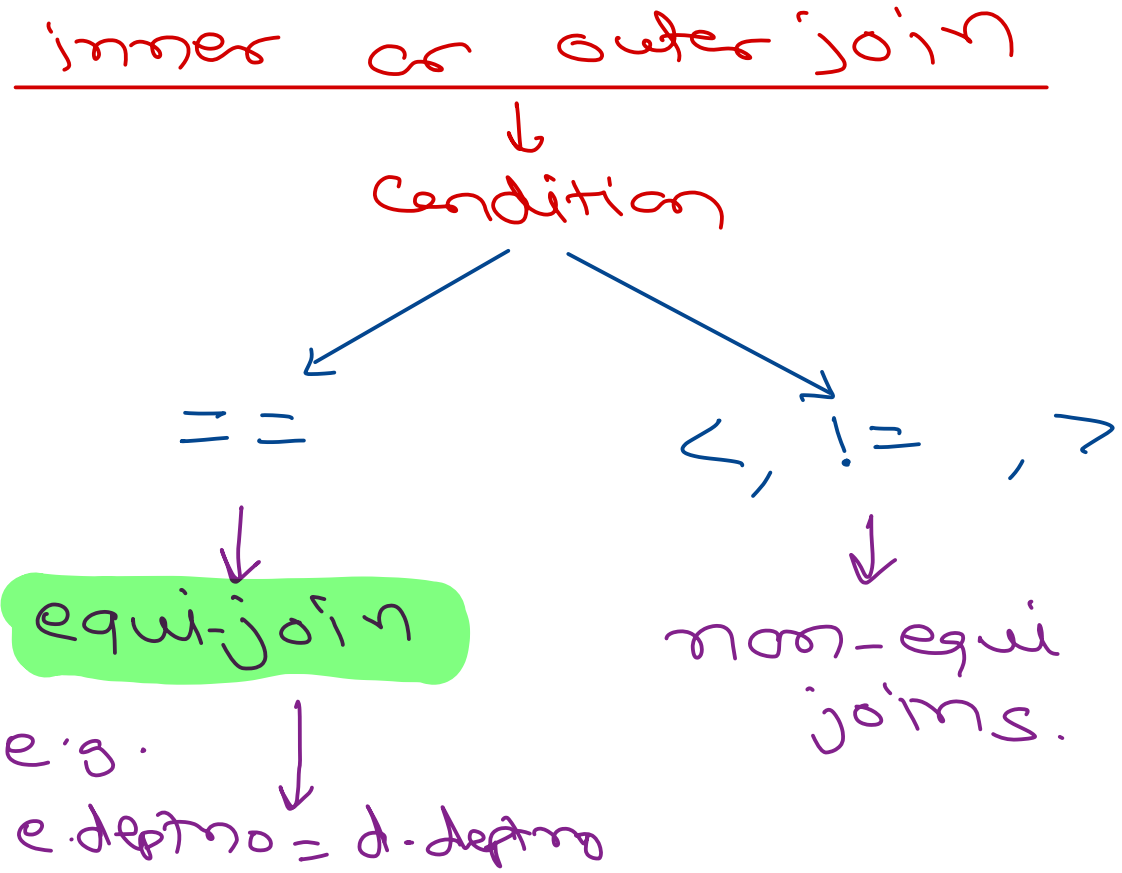
empno	ename	deptno	mgr
1	Amit	10	4
2	Rahul	10	3
3	Nilesh	20	4
4	Nitin	50	5
5	Sarang	50	NULL

e m
Amit, Nitin
Rahul, Nilesh
Nilesh, Nitin
Nitin, Sarang

e m
Amit, Nitin
Rahul, Nilesh
Nilesh, Nitin
Nitin, Sarang
Sarang, x

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

Joins




```
select e.ename, d.dname from
depts d inner join emps e
on e.deptno = d.deptno;
```

```
select e.ename, d.dname from
depts d join emps e
using (deptno);
```

→ col name is same in both tables.

Joins – Inner join

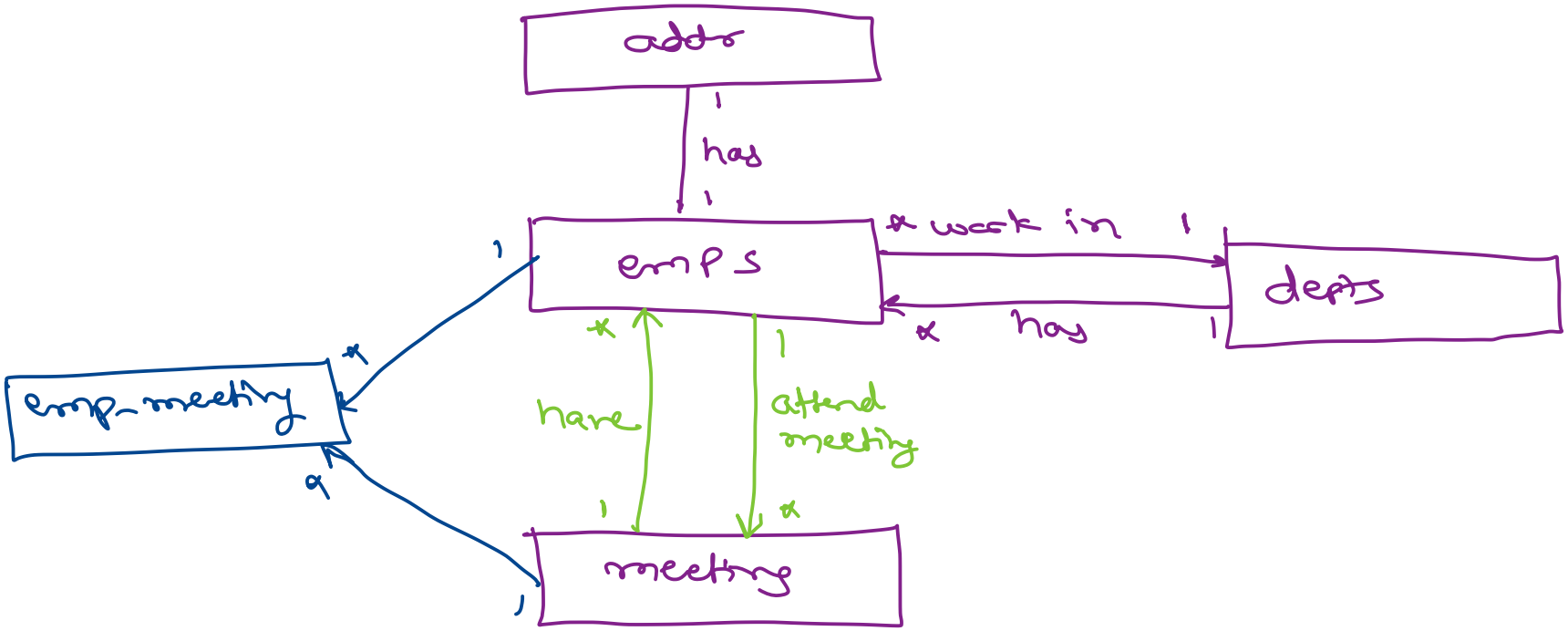
standard

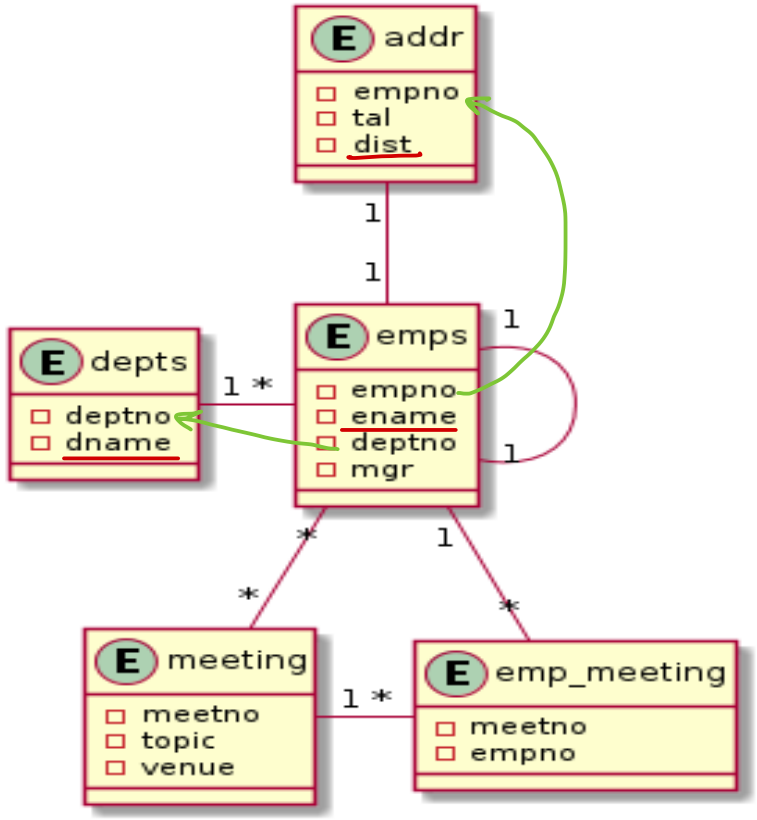
```
select e.ename, d.dname from
emps e inner join depts d
on e.deptno = d.deptno;
```

non-standard

```
select e.ename, d.dname from
emps e, depts d
where e.deptno = d.deptno;
```

Joins


addr
has
emps
work in
depts
has
emp_meeting
have
attend meeting
meeting


addr
empno
tal
dist
emps
empno
ename
deptno
mgr
depts
deptno
dname
meeting
meetno
topic
venue
emp_meeting
meetno
empno

Thank you!

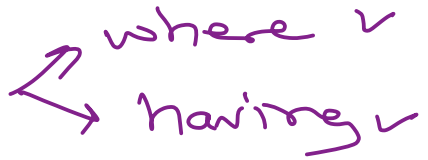
Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL - RDBMS

Trainer: Mr. Nilesh Ghule

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 of value.
    - ANY operator compares with one of the result from sub-queries. → like OR
    - ALL operator compares with all the results from sub-queries. → like AND

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL - RDBMS

Trainer: Mr. Nilesh Ghule

Sub queries

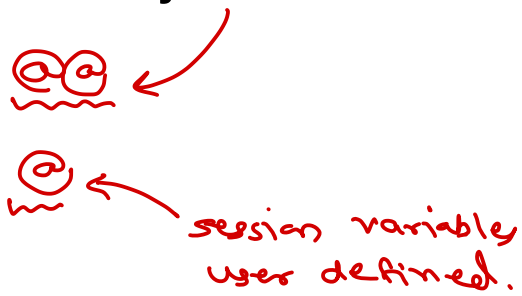
- Correlated sub-query
  - If number of results from sub-query are reduced, query performance will increase.
  - This can be done by adding criteria (WHERE clause) in sub-query based on outer query row.
  - Typically correlated sub-query use IN, ALL, ANY and EXISTS operators.

Sub query

- Sub queries with UPDATE and DELETE are not supported in all RDBMS.
- In MySQL, Sub-queries in UPDATE/DELETE is allowed, but sub-query should not SELECT from the same table, on which UPDATE/DELETE operation is in progress.

Query performance → RDBMS specific.

- 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';



Thank you!

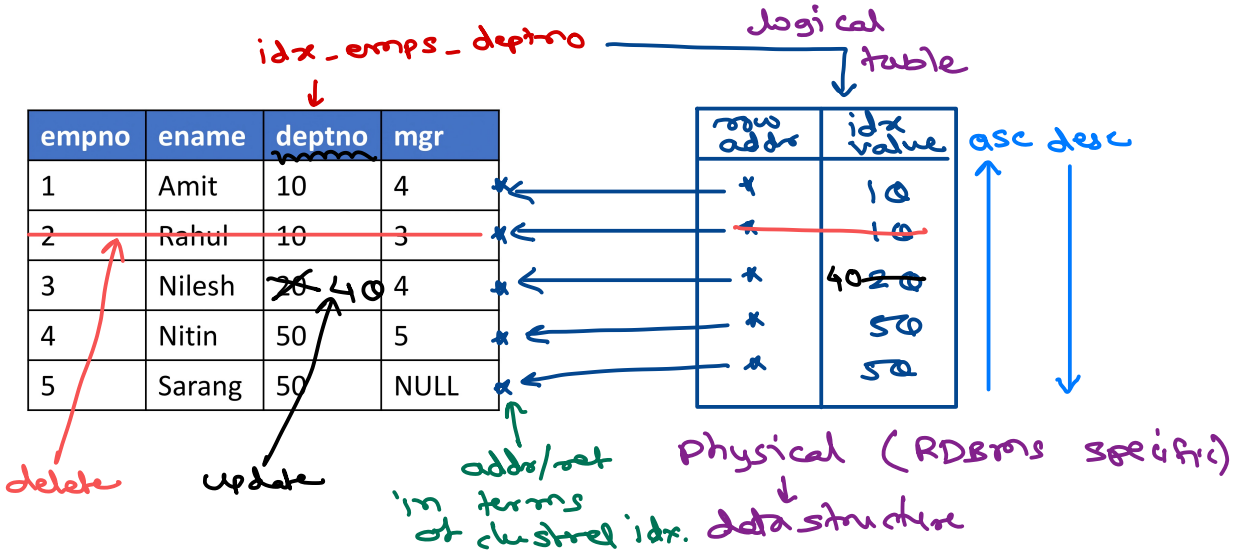
Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL - RDBMS

Trainer: Mr. Nilesh Ghule

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

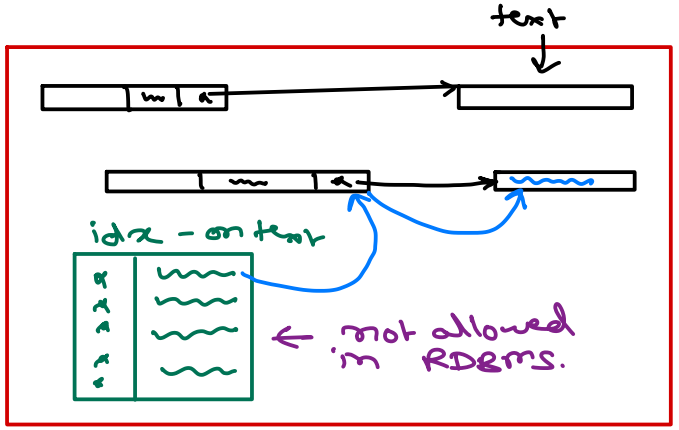
- Index can be ASC or DESC.
  - It cause storage of key values in respective order (MySQL 8.x onwards).
  - ASC/DESC index is used by optimizer on ORDER BY queries.
- There are four types of indexes:
  - Simple index
    - CREATE INDEX idx_name ON table(column [ASC|DESC]);
  - Unique index
    - CREATE 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.

hidden column added by RDBMS

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;

index on first 10 bytes. i.e. only first 10 bytes used for searching with indexes.

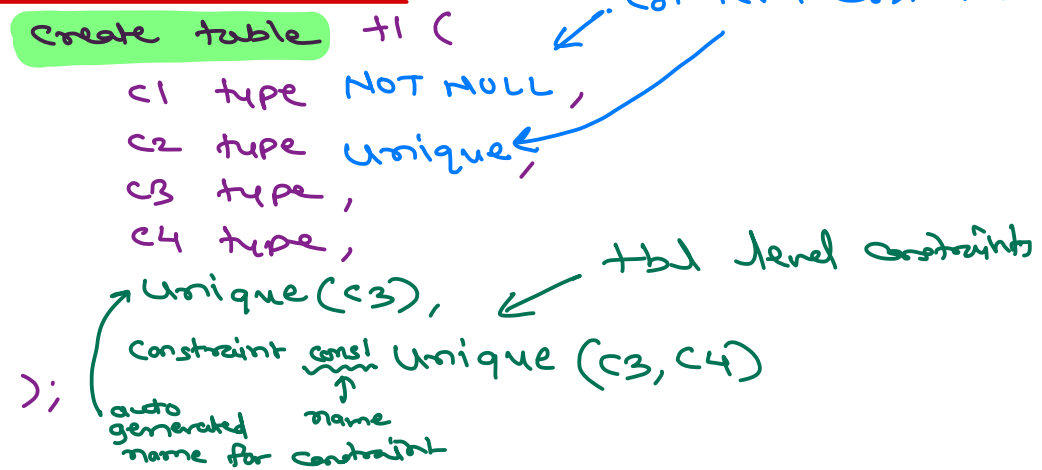


- When table is dropped, all indexes are automatically dropped.
- Indexes should not be created on the columns not used frequent search, ordering or grouping operations.
- Columns in join operation should be indexed for better performance.

emp(deptno) → ON e.deptno = d.deptno dept(deptno)

Constraints

- Constraints are restrictions imposed on columns.
- There are five constraints
  - NOT NULL → col level
  - UNIQUE → col or tbl level
  - PRIMARY KEY → col or tbl level
  - FOREIGN KEY → col or tbl level
  - CHECK → col or tbl level



- 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.

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));

```
create table students (
    std INT not null,
    roll INT not null,
    name CHAR(20),
    unique (std, roll)
);
```

internally create unique composite index.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL - RDBMS

Trainer: Mr. Nilesh Ghule

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));

```
Create table students (
    std INT not null,
    roll INT not null,
    name CHAR(20),
    unique (std, roll)
);
```

internally create unique composite index.

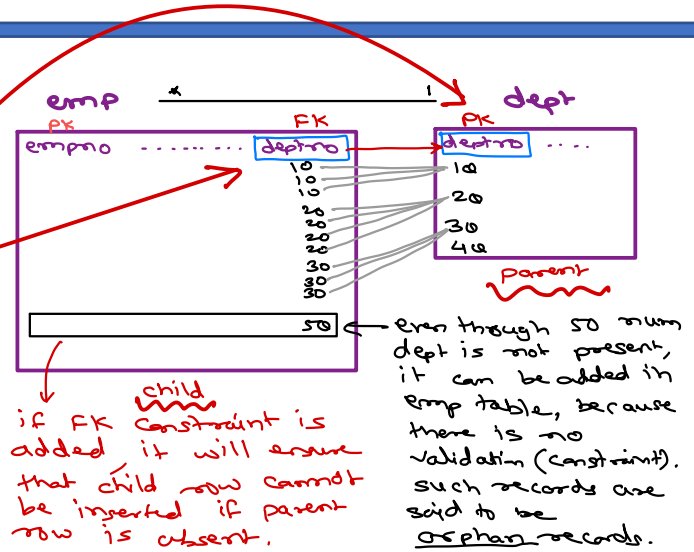
one table may have multiple unique keys.

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.
  - 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".
  - FK can have duplicate values as well as null values.
  - FK constraint is applied on column of child table (not on parent table).
  - parent rows cannot be deleted, until child 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))



preferred

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);

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

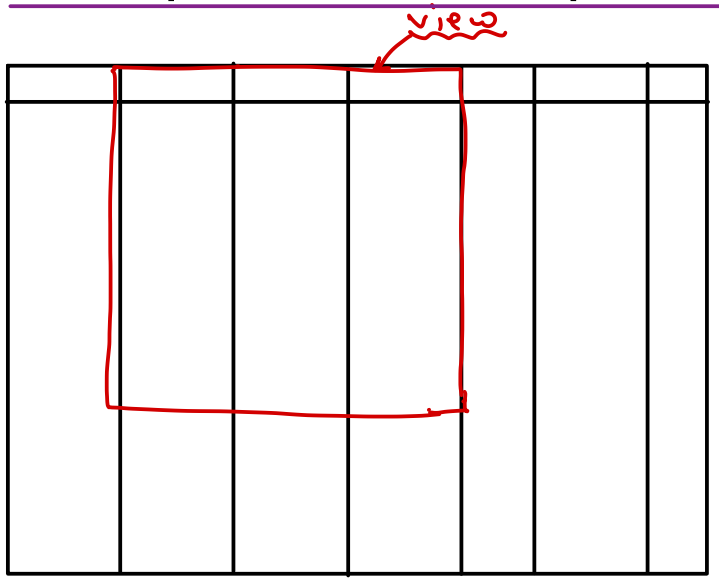
MySQL - RDBMS

Trainer: Mr. Nilesh Ghule

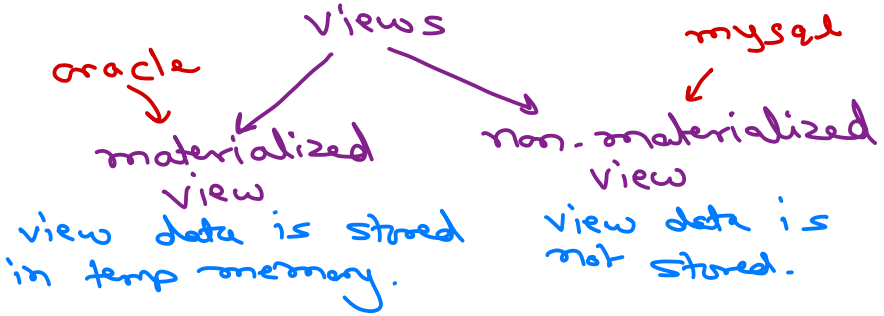
Views

- RDBMS view represents view (projection) of the data.
- View is based on SELECT statement.
- Typically it is restricted view of the data (limited rows or columns) from one or more tables (joins and/or sub-queries) or summary of the data (grouping).
- Data of view is not stored on server hard-disk; but its SELECT statement is stored in compiled form. It speed up execution of view.

→ each query on view will 'internally execute select query first. & process on top of it.




```
Create view view_name
AS SELECT ..... ;
```



Views

- Views are of two types: Simple view and Complex view
- Usually if view contains computed columns, group by, joins or sub-queries, then the views are said to be complex. DML operations are not supported on these views.
- DML operations on view affects underlying table.
- View can be created with CHECK OPTION to ensure that DML operations can be performed only the data visible in that view.

View

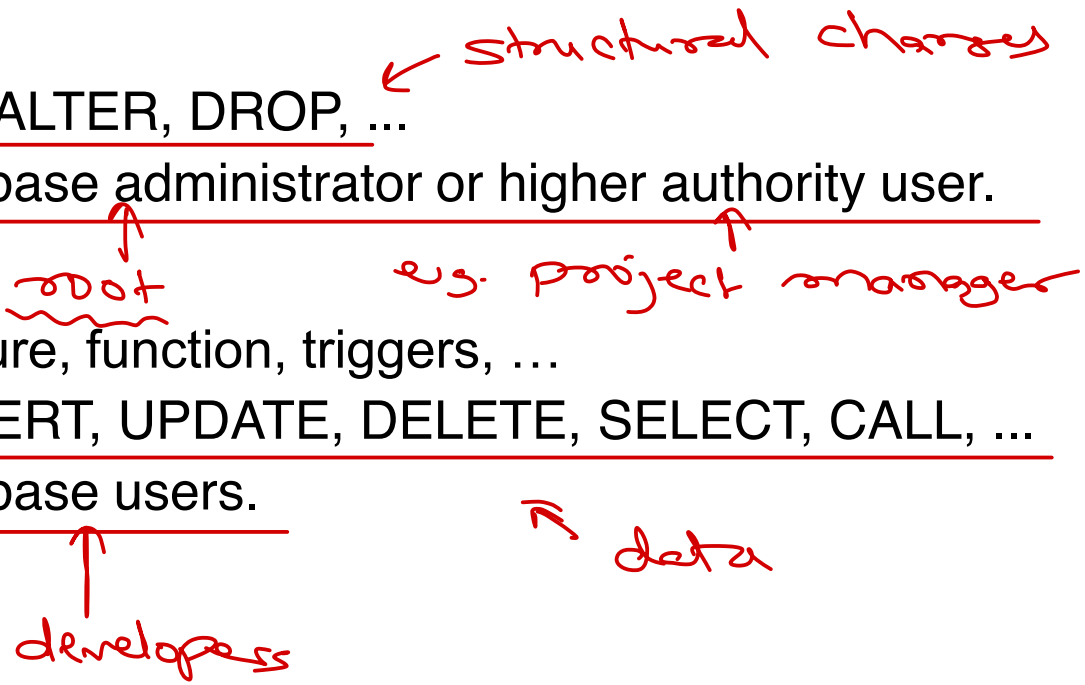
- Views can be differentiated with: SHOW FULL TABLES.
- Views can be dropped with DROP VIEW statement.
- View can be based on another view.

create view view2 as
select * from view1 where ---;

- Applications of views
  - Security: Providing limited access to the data.
  - Hide source code of the table.
  - Simplifies complex queries.

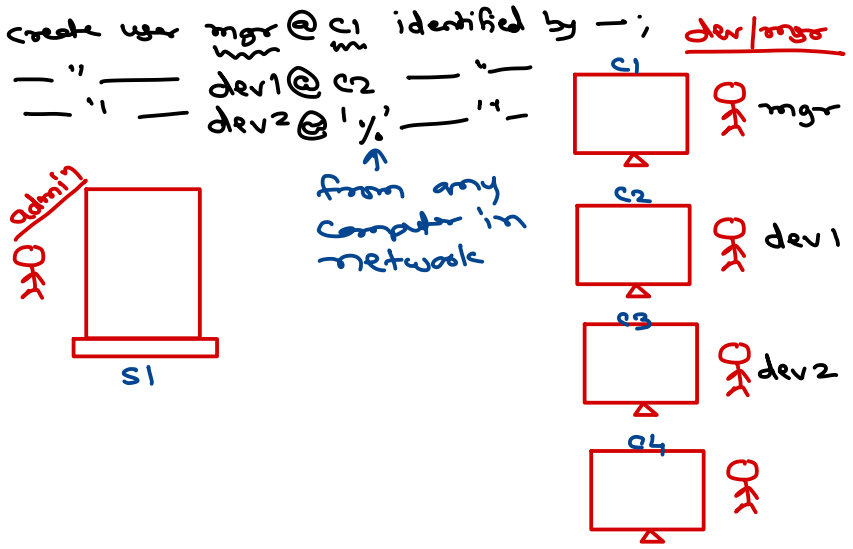
Data Control Language

- Security is built-in feature of any RDBMS. It is implemented in terms of permissions (a.k.a. privileges).
- There are two types of privileges.
- System privileges
  - Privileges for certain commands i.e. CREATE, ALTER, DROP, ...
  - Typically these privileges are given to the database administrator or higher authority user.
- Object privileges
  - RDBMS objects are table, view, stored procedure, function, triggers, …
  - Can perform operations on the objects i.e. INSERT, UPDATE, DELETE, SELECT, CALL, ...
  - Typically these privileges are given to the database users.



User Management

- User management is responsibility of admin (root).
- New user can be created using CREATE USER.
  - CREATE USER user@host IDENTIFIED BY 'password';
  - host can be hostname of server, localhost (current system) or '%' for all client systems.
- Permissions for the user can be listed using SHOW GRANTS command.
  - SHOW GRANTS FOR user@host;
- Users can be deleted using DROP USER.
  - DROP USER user@host;
- Change user password.
  - ALTER USER user@host IDENTIFIED BY 'new_password';
  - FLUSH PRIVILEGES;



Data Control Language

- Permissions are given to user using GRANT command.
  - GRANT CREATE ON db.* TO user@host;
  - GRANT CREATE ON *.* TO user1@host, user2@host;
  - GRANT SELECT ON db.table TO user@host;
  - GRANT SELECT, INSERT, UPDATE ON db.table TO user@host;
  - GRANT ALL ON db.* TO user@host;
- By default one user cannot give permissions to other user. This can be enabled using WITH GRANT OPTION.
  - GRANT ALL ON *.* TO user@host WITH GRANT OPTION;
- Permissions assigned to any user can be withdrawn using REVOKE command.
  - REVOKE SELECT, INSERT ON db.table FROM user@host;
- Permissions can be activated by FLUSH PRIVILEGES.
  - System GRANT tables are reloaded by this command. Auto done after GRANT, REVOKE.
  - Command is necessary if GRANT tables are modified using DML operations.

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;

PSM - Agenda - Persistent Storage Module.

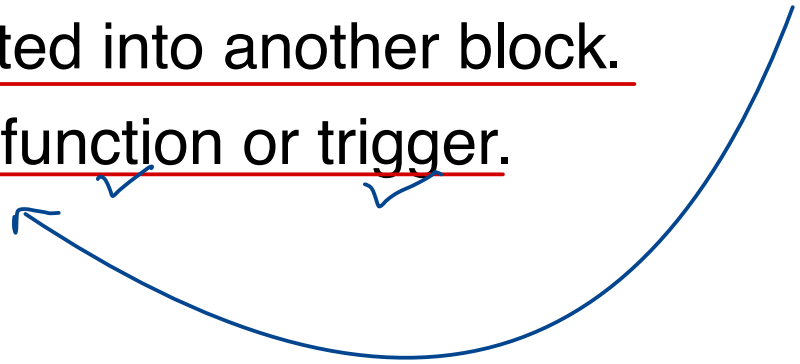
- MySQL Programming
- Stored procedure
- Exceptions
- Function
- Trigger

program → set of instructions

MySQL program → set of sql stmts.
along with programming constructs e.g.
loops, if-else, ...

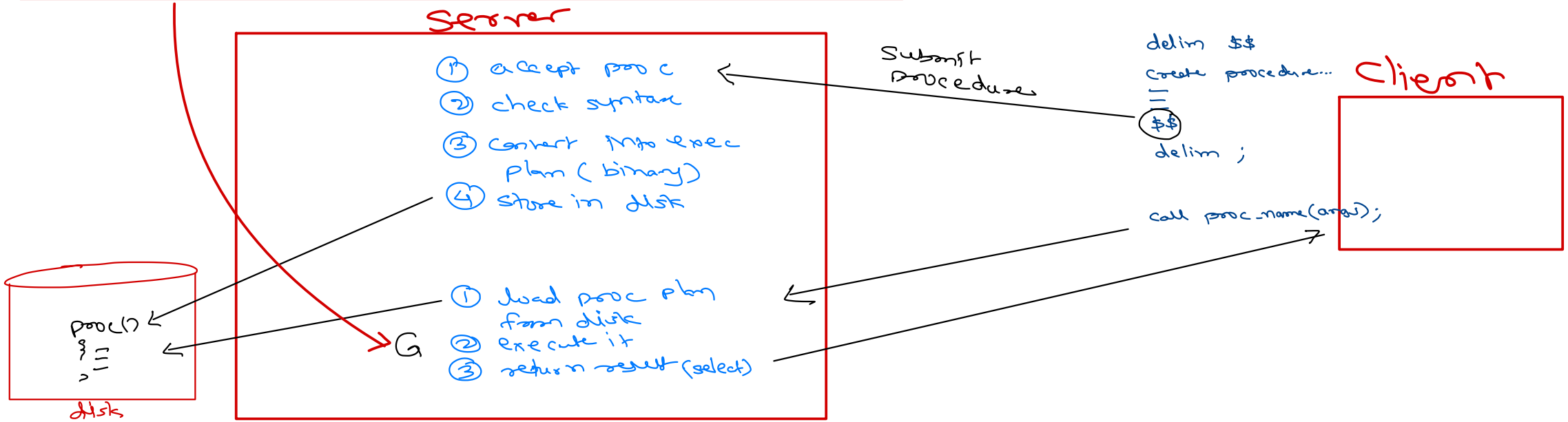
MySQL Programming

- RDBMS Programming is an ISO standard – part of SQL standard – since 1992.
- SQL/PSM stands for Persistent Stored Module.
- Inspired from PL/SQL - Programming language of Oracle.
- PSM allows writing programs for RDBMS. The program contains set of SQL statements along with programming constructs e.g. variables, if-else, loops, case, ...
- PSM is a block language. Blocks can be nested into another block.
- MySQL program can be a stored procedure, function or trigger.



MySQL Programming

- MySQL PSM program is written by db user (programmers).
- It is submitted from client, server check syntax & store them into db in compiled form.
- The program can be executed by db user when needed.
- Since programs are stored on server in compiled form, their execution is very fast.
- All these programs will run in server memory.



Stored Procedure

- Stored Procedure is a routine. It contains multiple SQL statements along with programming constructs.
- Procedure doesn't return any value (like void fns in C).
- Procedures can take zero or more parameters.
- Procedures are created using CREATE PROCEDURE and deleted using DROP PROCEDURE.
- Procedures are invoked/called using CALL statement.
- Result of stored procedure can be
  - returned via OUT parameter.
  - inserted into another table.
  - produced using SELECT statement (at end of SP).
- Delimiter should be set before writing SQL query.

Stored Procedure

```
CREATE TABLE result(v1 DOUBLE, v2 VARCHAR(50));

DELIMITER $$

CREATE PROCEDURE sp_hello()
BEGIN
    INSERT INTO result VALUES(1, 'Hello World');
END;
$$

DELIMITER ;

CALL sp_hello();

SELECT * FROM result;
```

```
-- 01_hello.sql (using editor)
DROP PROCEDURE IF EXISTS sp_hello;
DELIMITER $$
CREATE PROCEDURE sp_hello()
BEGIN
    SELECT 1 AS v1, 'Hello World' AS v2;
END;
$$
DELIMITER ;


SOURCE /path/to/01_hello.sql

CALL sp_hello();
```

Stored Procedure – PSM Syntax

VARIABLES

```
DECLARE varname DATATYPE;
DECLARE varname DATATYPE DEFAULT init_value;
SET varname = new_value;
SELECT new_value INTO varname;
SELECT expr_or_col INTO varname FROM table_name;
```

PARAMETERS

```
CREATE PROCEDURE sp_name(PARAMTYPE p1 DATATYPE)
BEGIN
    …
END;

-- IN param: Initialized by calling program.
-- OUT param: Initialized by called procedure.
-- INOUT param: Initialized by calling program and
modified by called procedure
-- OUT & INOUT param declared as session variables.

CREATE PROCEDURE sp_name(OUT p1 INT)
BEGIN
    SELECT 1 INTO p1;
END;

SET @res = 0;
CALL sp_name(@res);
SELECT @res;
```

IF-ELSE

```
IF condition THEN
     body;
END IF;
-----------------------
IF condition THEN
     if-body;
ELSE
     else-body;
END IF;
-----------------------
IF condition THEN
     if1-body;
ELSE
     IF condition THEN
            if2-body;
     ELSE
            else2-body;
     END IF;
END IF;
-----------------------
IF condition THEN
     if1-body;
ELSEIF condition THEN
     if2-body;
ELSE
     else-body;
END IF;
```

LOOPS

```
WHILE condition DO
    body;
END WHILE;
------------------
REPEAT
    body;
UNTIL condition
END REPEAT;
------------------
label: LOOP
IF condition THEN
        ...
        LEAVE label;
    END IF;
    ...
END LOOP;
```

CASE-WHEN

```
CASE
WHEN condition THEN
      body;
WHEN condition THEN
      body;
ELSE
       body;
END CASE;
```

SHOW PROCEDURE

```
SHOW PROCEDURE STATUS
LIKE ‘sp_name’;

SHOW CREATE PROCEDURE sp_name;
```

DROP PROCEDURE

```
DROP PROCEDURE
IF EXISTS sp_name;
```

Thank you!

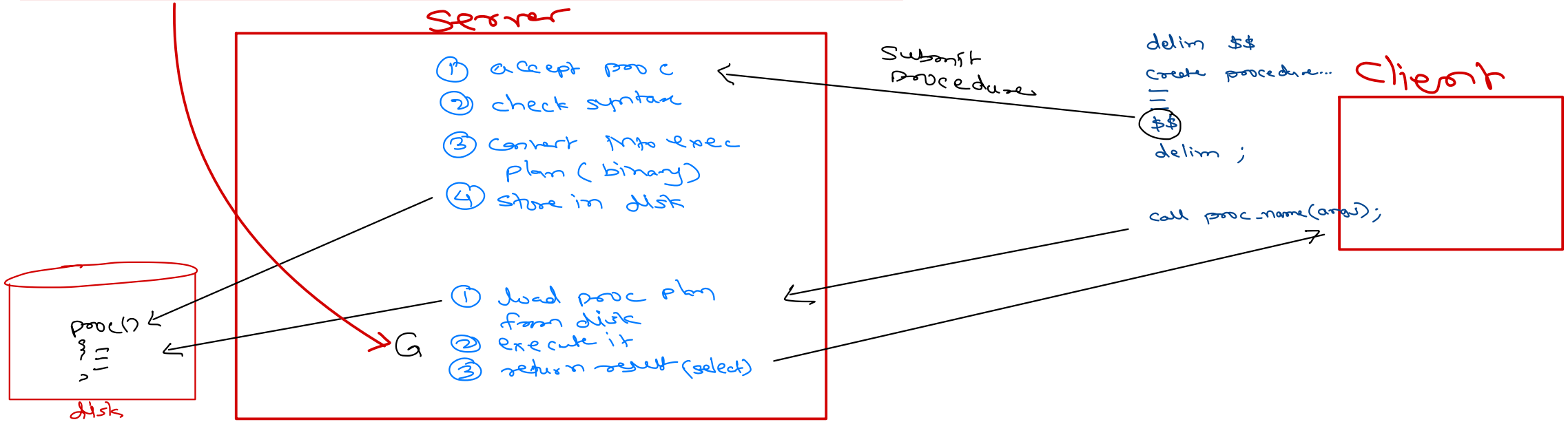
Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL - RDBMS

Trainer: Mr. Nilesh Ghule

MySQL Programming

- MySQL PSM program is written by db user (programmers).
- It is submitted from client, server check syntax & store them into db in compiled form.
- The program can be executed by db user when needed.
- Since programs are stored on server in compiled form, their execution is very fast.
- All these programs will run in server memory.



Stored Procedure

- Stored Procedure is a routine. It contains multiple SQL statements along with programming constructs.
- Procedure doesn't return any value (like void fns in C).
- Procedures can take zero or more parameters.
- Procedures are created using CREATE PROCEDURE and deleted using DROP PROCEDURE.
- Procedures are invoked/called using CALL statement.
- Result of stored procedure can be
  - returned via OUT parameter.
  - inserted into another table.
  - produced using SELECT statement (at end of SP).
- Delimiter should be set before writing ~~SQL query~~ procedure.

like for definition

like for call.

→ eg. results table

→ only last select stmt output will be displayed.

Stored Procedure

```
CREATE TABLE result(v1 DOUBLE, v2 VARCHAR(50));

DELIMITER $$

CREATE PROCEDURE sp_hello()
BEGIN
    INSERT INTO result VALUES(1, 'Hello World');
END;
$$

DELIMITER ;

CALL sp_hello();

SELECT * FROM result;
```

← see the result.

```
-- 01_hello.sql (using editor)
DROP PROCEDURE IF EXISTS sp_hello;
DELIMITER $$
CREATE PROCEDURE sp_hello()
BEGIN
    SELECT 1 AS v1, 'Hello World' AS v2;
END;
$$
DELIMITER ;

SOURCE /path/to/01_hello.sql

CALL sp_hello();
```

forward slash

no space in whole path.

output will be displayed here.(CLI).

Stored Procedure – PSM Syntax

VARIABLES

```
DECLARE varname DATATYPE;
DECLARE varname DATATYPE DEFAULT init_value;
SET varname = new_value;
SELECT new_value INTO varname;
SELECT expr_or_col INTO varname FROM table_name;
```

PARAMETERS

```
CREATE PROCEDURE sp_name(PARAMTYPE p1 DATATYPE)
BEGIN
    …
END;

-- IN param: Initialized by calling program.
-- OUT param: Initialized by called procedure.
-- INOUT param: Initialized by calling program and
modified by called procedure
-- OUT & INOUT param declared as session variables.

CREATE PROCEDURE sp_name(OUT p1 INT)
BEGIN
    SELECT 1 INTO p1;
END;

SET @res = 0;
CALL sp_name(@res);
SELECT @res;
```

IF-ELSE

```
IF condition THEN
     body;
END IF;
-----------------------
IF condition THEN
     if-body;
ELSE
     else-body;
END IF;
-----------------------
IF condition THEN
     if1-body;
ELSE
     IF condition THEN
            if2-body;
     ELSE
            else2-body;
     END IF;
END IF;
-----------------------
IF condition THEN
     if1-body;
ELSEIF condition THEN
     if2-body;
ELSE
     else-body;
END IF;
```

LOOPS

```
WHILE condition DO
    body;
END WHILE;
------------------
REPEAT
    body;
UNTIL condition
END REPEAT;
------------------
label: LOOP
IF condition THEN
        ...
        LEAVE label;
    END IF;
    ...
END LOOP;
```

CASE-WHEN

```
CASE
WHEN condition THEN
      body;
WHEN condition THEN
      body;
ELSE
       body;
END CASE;
```

SHOW PROCEDURE

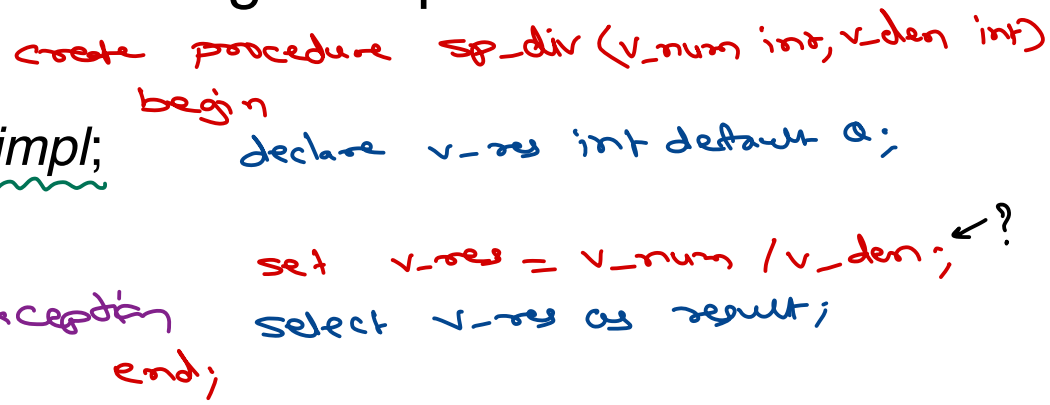
```
SHOW PROCEDURE STATUS
LIKE ‘sp_name’;

SHOW CREATE PROCEDURE sp_name;
```

DROP PROCEDURE

```
DROP PROCEDURE
IF EXISTS sp_name;
```

MySQL Exceptions / Error Handling

- Exceptions are runtime problems, which may arise during execution of stored procedure, function or trigger.
- Required actions should be taken against these errors.
- SP execution may be continued or stopped after handling exception.
- MySQL error handlers are declared as:
  - DECLARE *action* HANDLER FOR *condition handler_impl*;
- The *action* can be: CONTINUE or EXIT.
- The *condition* can be:
  - MySQL error code: e.g. 1062 for duplicate entry.
  - SQLSTATE value: e.g. 23000 for duplicate entry, NOTFOUND for end-of-cursor.
  - Named condition: e.g. DECLARE duplicate_entry CONDITION FOR 1062;
- The *handler_impl* can be: Single liner or PSM block i.e. BEGIN … END;



Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

MySQL - RDBMS

Trainer: Mr. Nilesh Ghule

MySQL Stored Functions

- Stored Functions are MySQL programs like stored procedures.
- Functions can be having one or more parameters. MySQL allows only IN params.
- Functions must return some value using RETURN statement.
- Function entire code is stored in system table.
- Like procedures, functions allows statements like local variable declarations, if-else, case, loops, etc. One function can invoke another function/procedure and vice-versa. The functions can also be recursive.
- There are two types of functions: DETERMINISTIC and NOT DETERMINISTIC.

CREATE FUNCTION

```
CREATE FUNCTION fn_name(p1 TYPE)
RETURNS TYPE
[NOT] DETERMINISTIC
BEGIN
   body;
   RETURN value;
END;
```

SHOW FUNCTION

```
SHOW FUNCTION STATUS LIKE 'fn_name';

SHOW CREATE FUNCTION fn_name;
```

DROP FUNCTION

```
DROP FUNCTION IF EXISTS fn_name;
```

MySQL Triggers

- Triggers are supported by all standard RDBMS like Oracle, MySQL, etc.
- Triggers are not supported by WEAK RDBMS like MS-Access.
- Triggers are not called by client's directly, so they don't have args & return value.
- Trigger execution is caused by DML operations on database.
  - BEFORE/AFTER INSERT, BEFORE/AFTER UPDATE, BEFORE/AFTER DELETE.
- Like SP/FN, Triggers may contain SQL statements with programming constructs. They may also call other SP or FN.
- However COMMIT/ROLLBACK is not allowed in triggers. They are executed in same transaction in which DML query is executed.

CREATE TRIGGER

```
CREATE TRIGGER trig_name
AFTER|BEFORE dml_op ON table
FOR EACH ROW
BEGIN
  body;
    -- use OLD & NEW keywords
    -- to access old/new rows.
    -- INSERT triggers - NEW rows.
    -- DELETE triggers - OLD rows.
END;
```

SHOW TRIGGERS

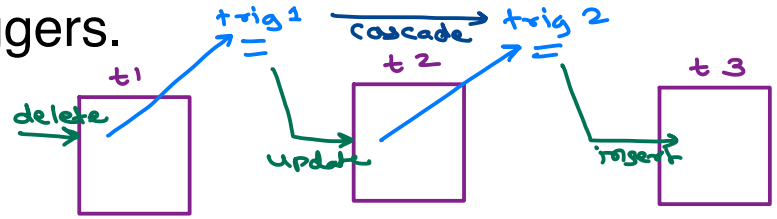
```
SHOW TRIGGERS FROM db_name;
```

DROP TRIGGER

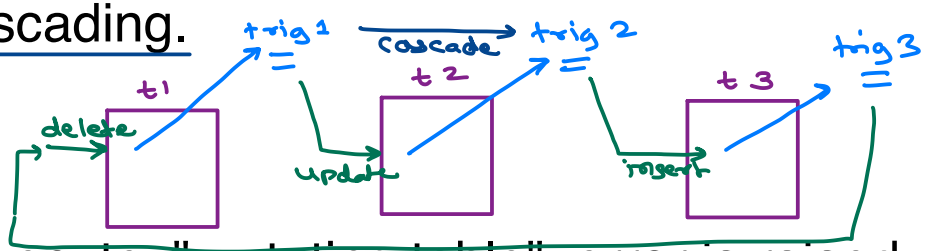
```
DROP TRIGGER trig_name;
```

MySQL Triggers

- Applications of triggers:
  - Maintain logs of DML operations (Audit Trails).
  - Data cleansing before insert or update data into table. (Modify NEW value).
  - Copying each record AFTER INSERT into another table to maintain "Shadow table".
  - Copying each record AFTER DELETE into another table to maintain "History table".
  - Auto operations of related tables using cascading triggers.



- Cascading triggers
  - One trigger causes execution of 2nd trigger, 2nd trigger causes execution of 3rd trigger and so on.
  - In MySQL, there is no upper limit on number of levels of cascading.
  - This is helpful in complicated business processes.



- Mutating table error
  - If cascading trigger causes one of the earlier trigger to re-execute, "mutating table" error is raised.
  - This prevents infinite loop and also rollback the current transaction.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>

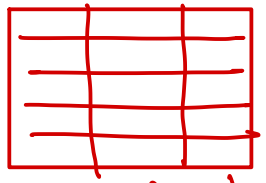
all RDBMS → SQL

↓

Relational

↓

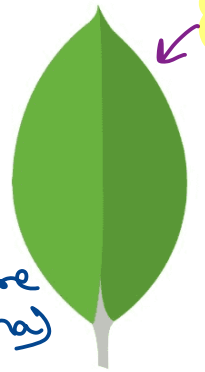
tabular [T1 ↔ T2]



Rows & cols

↑

- fixed structure (schema)
- 100s of GB
- high consistency / transactions.



mongo

↑

cassandra

neo 4j

↑

redis

- flexible schema
- 100s of TB / PB (scaling)
  - data
  - clients
- economical
- eventual consistency

NoSQL Databases

→ no standard query language X

Trainer: Mr. Nilesh Ghule

Not Only SQL

Document oriented databases

Java Script Object Notation

- Document contains data as key-value pair as JSON or XML.
- Document schema is flexible & are added in collection for processing.
- RDBMS tables → Collections
- RDBMS rows → Documents
- RDBMS columns → Key-value pairs in document
- Examples: MongoDb, CouchDb, …

b1 JSON → Java Script Object Notation.

```
{
  "id" : 1,        → int
  "title" : "Let us C",   → string
  "author" : "Kanetkar",
  "price" : 240.4   → double
}
```

r1 → JSON document

```
{ id : 1,
  name : "Nilesh",
  age : 38,
  hobbies : ["Program", "Reading", ...]
  addr : { area : "Katraj", city : "Pune", pin : 411046 },
  political : false,
  height : 5.9,
  bloodgroup : null
}
```

MongoDb Databases

Trainer: Mr. Nilesh Ghule

Mongo Db

- Developed by 10gen in 2007
- Publicly available in 2009
- Open-source database which is controlled by 10gen
- Document oriented database → stores JSON documents
- Stores data in binary JSON. → (BSON) → faster access.
- Design Philosophy
  - MongoDB wasn't designed in a lab and is instead built from the experiences of building large scale, high availability, and robust systems.

PBs 24x7

JSON

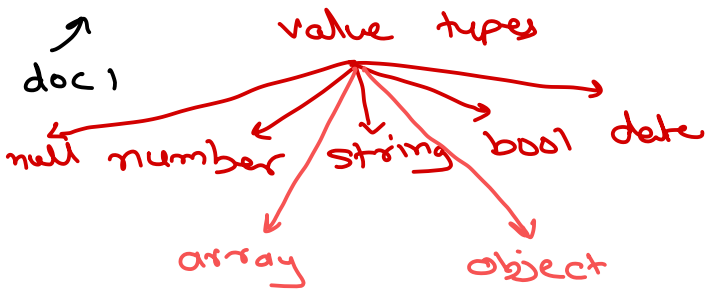
- Java Script Object Notation
- Hierarchical way of organizing data
- Mongo stores JSON data into Binary form. → data type internally identified by a number.

people → collection

doc 1

```
{
    "name" : "Nitin",
    "mobile" : "9881208115",
    "pin" : 411037,
    "sal" : 50000.00
}
```

key, value

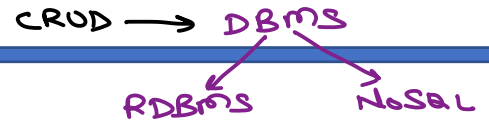


doc 2

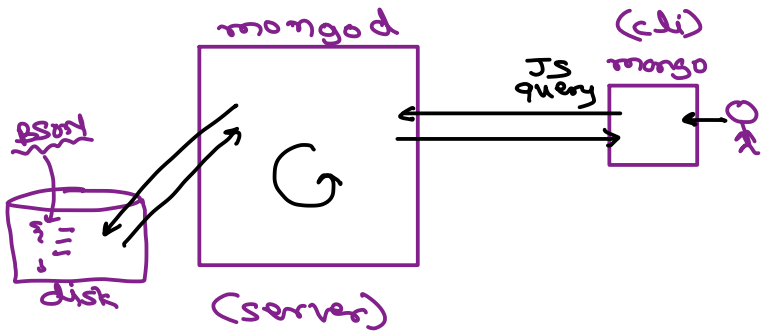
```
{
    "name" : "Nilesh",
    "addr" : {
        "area" : "katraj",
        "city" : "pune",
        "pin" : 411046
    },
    "hobbies" : ["programming", "reading", "cooking"],
    "age" : 37,
    "sal" : 300000.0,
    "political" : false,
    "rating" : null
}
```

string, json object, array, number, boolean, null

Mongo Server and Client



- MongoDb server (mongod) is developed in C, C++ and JS.
- MongoDb data is accessed via multiple client tools
  - mongo : client shell (JS).
  - mongofiles : stores larger files in GridFS.
  - mongoimport / mongoexport : tools for data import / export.
  - mongodump / mongorestore : tools for backup / restore.



- MongoDb data can be accessed in application through client drivers available for all major programming languages e.g. Java, Python, Ruby, PHP, Perl, …
- Mongo shell is follows JS syntax and allow to execute JS scripts.

MongoDb: Data Types

data	bson	values
null	10	
boolean	8	true, false
number	1 / 16 / 18	123, 456.78, NumberInt(“24”), NumberLong(“28”)
string	2	“....”
date	9	new Date(), ISODate(“yyyy-mm-ddThh:mm:ss”)
array	4	[…, …, …, …]
object	3	{ … }

Mongo - INSERT

- show databases;
- use database;
- db.contacts.insert({name: "nilesh", mobile: "9527331338"});
- db.contacts.insertMany([

  {name: "nilesh", mobile: "9527331338"},

  {name: "nitin", mobile: "9881208115"}

  ]);
- Maximum document size is 16 MB.
- For each object unique id is generated by client (if _id not provided).
  - 12 byte unique id :: [counter(3) | pid(2) | machine(3) | timestamp(4)]

Mongo – QUERY

- db.contacts.find(); → returns cursor on which following ops allowed:
  - hasNext(), next(), skip(n), limit(n), count(), toArray(), forEach(fn), pretty()
- Shell restrict to fetch 20 records at once. Press "it" for more records.
- db.contacts.find({ name: "nilesh" });
- db.contacts.find({ name: "nilesh" }, { _id:0, name:1 });
- Relational operators: $eq, $ne, $gt, $lt, $gte, $lte, $in, $nin
- Logical operators: $and, $or, $nor, $not
- Element operators: $exists, $type
- Evaluation operators: $regex, $where, $mod
- Array operators: $size, $elemMatch, $all, $slice

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>