MySQL - RDBMS

Agenda

- Lab Exam Pattern
- Indexes
- Constraints
- ~~ALTER~~
- ~~Views~~
- ~~DCL~~

Internal assesments

- Rapid fire sheet -- approx "15" questions
  - Study from class notes/lecture.
  - Hand written (organized) answers on your notebook
  - Ensure that your name is written on each page (in your handwriting)
  - Scan (Adobe scanner) and upload as per instructions
  - This sheet is helpful during campus for quick revision.
- Interview question video.
  - 5 mins single video.
  - max 3 questions.
  - Improving presentation skills.
- Assignments/Queries
  - 20 queries
  - Understading problem statement is part of assesments
  - Write the best answer

Lab Exam Pattern

- 90 to 120 mins -- proctored exam
- There will be different database tables.
- SQL queries - SELECT, INSERT, UPDATE, DELETE, GROUP BY, ORDER BY, JOINS, SUB-QUERIES, etc.
- PSM programs - Stored Procedure, Function, Trigger.
- MySQL HELP command is allowed.
- Mock lab exam -- to understand exam submission process.

Indexes

Simple Index

```
SELECT * FROM emps;

SELECT * FROM emps WHERE deptno = 10;

EXPLAIN FORMAT=JSON
```

```
SELECT * FROM emps WHERE deptno = 10;
-- 0.75

CREATE INDEX idx_emps_deptno ON emps(deptno);
-- by default indexes are ascending

SELECT * FROM emps WHERE deptno = 10;

EXPLAIN FORMAT=JSON
SELECT * FROM emps WHERE deptno = 10;
-- 0.70

CREATE INDEX idx_emps_ename ON emps(ename DESC);

SELECT * FROM emps WHERE ename = 'Nitin';

SHOW INDEXES FROM emps;

DESCRIBE emps;
```

Unique Index

```
CREATE UNIQUE INDEX idx_emps_empno ON emps(empno);

SHOW INDEXES FROM emps;
-- non-unique = 0

DESCRIBE emps;

SELECT * FROM emps;

INSERT INTO emps VALUES (3, 'John', 30, 1);
-- error: empno 3 already exists

INSERT INTO emps VALUES (7, 'Motu', 30, 1);
```

Composite Index

- Index on combination of two or more columns.
- WHERE clause on multiple columns execute faster.

```
SELECT * FROM emp;

-- find all CLERK working in dept 20.
SELECT * FROM emp WHERE deptno = 20 AND job = 'CLERK';

EXPLAIN FORMAT=JSON
SELECT * FROM emp WHERE deptno = 20 AND job = 'CLERK';
-- 1.65
```

```
CREATE INDEX idx_emo_deptno_job ON emp(deptno ASC, job ASC);

SHOW INDEXES FROM emp;
DESCRIBE emp;

SELECT * FROM emp WHERE deptno = 20 AND job = 'CLERK';

EXPLAIN FORMAT=JSON
SELECT * FROM emp WHERE deptno = 20 AND job = 'CLERK';
-- 0.70
```

```
-- roll no are unique in each std.
CREATE TABLE students(roll INT, std INT, name CHAR(30), marks DECIMAL(5,2));

INSERT INTO students VALUES (1, 1, 'Soham', 90.0);
INSERT INTO students VALUES (2, 1, 'Sakshi', 92.0);
INSERT INTO students VALUES (3, 1, 'Prisha', 94.0);
INSERT INTO students VALUES (1, 2, 'Madhura', 95.0);
INSERT INTO students VALUES (2, 2, 'Om', 96.0);

SELECT * FROM students;

CREATE UNIQUE INDEX idx_students ON students(roll,std);

INSERT INTO students VALUES (3, 1, 'Rachana', 99.0);
-- error: roll 3 in std 1 already exists

SHOW INDEXES FROM students;

DROP INDEX idx_students ON students;

SHOW INDEXES FROM students;
```

Constraints

NOT NULL and UNIQUE

```
DROP TABLE IF EXISTS contacts;

CREATE TABLE contacts(
    name VARCHAR(40) NOT NULL,
    phone CHAR(14) UNIQUE NOT NULL, -- column level constraint
    email VARCHAR(40),
    UNIQUE(email) -- table level constraint (name is auto-generated)
);

INSERT INTO contacts VALUES('Nilesh', '9527331338', 'nilesh@sunbeaminfo.com');
INSERT INTO contacts VALUES(NULL, '9876543210', NULL);
```

```
-- error: name cannot be NULL
INSERT INTO contacts(email,phone) VALUES('james.bond@london.com', '9876543210');
-- error: name cannot be NULL
INSERT INTO contacts VALUES('Abhishek', '9822012345', NULL);
INSERT INTO contacts VALUES('Amitabh', '9822012346', NULL);

INSERT INTO contacts VALUES('Jaya', '9822012345', NULL);
-- error: phone cannot be duplicated
INSERT INTO contacts VALUES('Jaya', NULL, NULL);
-- error: phone cannot be NULL

SELECT * FROM contacts;

DESCRIBE contacts;

SHOW INDEXES FROM contacts;
```

- To delete unique constraint one can delete unique index created with it.
- Constraints can be modified after creating table using ALTER statement.

```
DROP INDEX idx_emo_deptno_job ON emp;

DESCRIBE emp;

ALTER TABLE emp MODIFY ename VARCHAR(40) NOT NULL;

DESCRIBE emp;

INSERT INTO emp(empno, sal) VALUES(1000, 3000.0);
-- error: ename cannot be NULL

ALTER TABLE emp ADD CONSTRAINT unique_empno UNIQUE(empno);

DESCRIBE emp;

INSERT INTO emp(empno, ename) VALUES(7900, 'JOHN');
-- error: ename cannot be duplicated

ALTER TABLE emp DROP CONSTRAINT unique_empno;

DESCRIBE emp;
```

Primary Key

```
-- primary key
CREATE TABLE customers(
    name VARCHAR(40),
    email VARCHAR(30) PRIMARY KEY, -- column level
    phone CHAR(14),
```

```
    password VARCHAR(20),
    addr VARCHAR(80),
    birth DATE
);

-- composite primary key
CREATE TABLE students(
    std INT,
    roll INT,
    name VARCHAR(40),
    marks DECIMAL(5,2),
    PRIMARY KEY (std, roll) -- must be on table level
);

CREATE TABLE students(
    std INT PRIMARY KEY,
    roll INT PRIMARY KEY,
    name VARCHAR(40),
    marks DECIMAL(5,2)
);
-- error: only one primary key is allowed

-- surrogate primary key
CREATE TABLE products(
    id INT PRIMARY KEY,
    name VARCHAR(40),
    category CHAR(20),
    price DECIMAL(7,2),
    quantity INT,
    rating DECIMAL(2,1)
);
-- id is extra column added into table to use as primary key -- surrogate primary
key
-- mostly surrogate PK are auto-generated (serial number)

CREATE TABLE products(
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(40),
    category CHAR(20),
    price DECIMAL(7,2),
    quantity INT,
    rating DECIMAL(2,1)
);

DESCRIBE products;

INSERT INTO products(name,price) VALUES('Notebook', 40);
INSERT INTO products(name,price) VALUES('Pencil', 4);
INSERT INTO products(name,price) VALUES('File', 25);

SELECT * FROM products;

ALTER TABLE products AUTO_INCREMENT = 100;
```

```
INSERT INTO products(name,price) VALUES('Pen', 10);
INSERT INTO products(name,price) VALUES('Eraser', 5);

SELECT * FROM products;
```

- Oracle have a concept of SEQUENCE for auto-generated values.

Foreign Key

```
INSERT INTO emp(empno, ename, deptno) VALUES (1000, 'Motu', 50);
-- allowed because no FK constraint present. due to this Motu is orphan record.

DELETE FROM emp WHERE empno = 1000;

SELECT * FROM emp;

ALTER TABLE dept ADD PRIMARY KEY (deptno);

DESCRIBE dept;

SHOW INDEXES FROM dept;

ALTER TABLE emp ADD FOREIGN KEY(deptno) REFERENCES dept(deptno);

DESCRIBE emp;

SHOW INDEXES FROM emp;

SELECT e.ename, d.dname FROM emp e
JOIN dept d ON e.deptno = d.deptno;

EXPLAIN FORMAT=JSON
SELECT e.ename, d.dname FROM emp e
JOIN dept d ON e.deptno = d.deptno;

INSERT INTO emp(empno, ename, deptno) VALUES (1000, 'Motu', 50);
-- error: PK is not present in dept
```

```
DROP TABLE IF EXISTS depts;
DROP TABLE IF EXISTS emps;

CREATE TABLE depts(deptno INT, dname VARCHAR(20), PRIMARY KEY(deptno));

CREATE TABLE emps(empno INT, ename VARCHAR(20), deptno INT, mgr INT,
FOREIGN KEY (deptno) REFERENCES depts(deptno));
```

```
DROP TABLE IF EXISTS depts;
DROP TABLE IF EXISTS emps;

CREATE TABLE depts(deptno INT, dname VARCHAR(20), PRIMARY KEY(deptno));

CREATE TABLE emps(empno INT, ename VARCHAR(20), deptno INT, mgr INT,
PRIMARY KEY (empno),
FOREIGN KEY (deptno) REFERENCES depts(deptno),
FOREIGN KEY (mgr) REFERENCES emps(empno));

-- parent rows should be inserted first
INSERT INTO depts VALUES (10, 'DEV');
INSERT INTO depts VALUES (20, 'QA');
INSERT INTO depts VALUES (30, 'OPS');
INSERT INTO depts VALUES (40, 'ACC');

-- emps must be in a order to handle self-referencing fk.
INSERT INTO emps VALUES (5, 'Sarang', 40, NULL);
INSERT INTO emps VALUES (4, 'Nitin', 40, 5);
INSERT INTO emps VALUES (1, 'Amit', 10, 4);
INSERT INTO emps VALUES (3, 'Nilesh', 20, 4);
INSERT INTO emps VALUES (2, 'Rahul', 10, 3);
```

```
DROP TABLE IF EXISTS emps;
DROP TABLE IF EXISTS depts;

CREATE TABLE depts(deptno INT, dname VARCHAR(20), PRIMARY KEY(deptno));

CREATE TABLE emps(empno INT, ename VARCHAR(20), deptno INT, mgr INT,
PRIMARY KEY (empno),
FOREIGN KEY (deptno) REFERENCES depts(deptno),
FOREIGN KEY (mgr) REFERENCES emps(empno));

-- disable fk checks temporarily so that data insertion can be faster.
SELECT @@foreign_key_checks;
SET @@foreign_key_checks = 0;
SELECT @@foreign_key_checks;

INSERT INTO emps VALUES (4, 'Nitin', 40, 5);
INSERT INTO emps VALUES (5, 'Sarang', 40, NULL);
INSERT INTO emps VALUES (2, 'Rahul', 10, 3);
INSERT INTO emps VALUES (1, 'Amit', 10, 4);
INSERT INTO emps VALUES (3, 'Nilesh', 20, 4);

INSERT INTO depts VALUES (10, 'DEV');
INSERT INTO depts VALUES (20, 'QA');
INSERT INTO depts VALUES (30, 'OPS');
INSERT INTO depts VALUES (40, 'ACC');

SELECT * FROM depts;
SELECT * FROM emps;
```

```
-- enable fk checks after data insertion, so that future inserts/updates will be
validated.
SET @@foreign_key_checks = 1;
SELECT @@foreign_key_checks;

INSERT INTO emps VALUES (6, 'Patlu', 70, 5);
-- error: fk check
```

```
-- Cannot delete parent rows until all child rows are deleted (because it will
make child rows orphan).
DELETE FROM depts WHERE deptno = 40;
-- error
DELETE FROM depts WHERE deptno = 30;
-- allowed: no child rows in emp table for deptno=30

DROP TABLE depts;
-- error: not allowed because all rows in emp will become orphan.

UPDATE depts SET deptno = 60 WHERE deptno = 10;
-- error: not allowed because child rows in emp for dept=10 will become orphan.
```

```
DROP TABLE IF EXISTS emps;
DROP TABLE IF EXISTS depts;

CREATE TABLE depts(deptno INT, dname VARCHAR(20), PRIMARY KEY(deptno));

CREATE TABLE emps(empno INT, ename VARCHAR(20), deptno INT, mgr INT,
PRIMARY KEY (empno),
FOREIGN KEY (deptno) REFERENCES depts(deptno) ON UPDATE CASCADE ON DELETE CASCADE
);
-- this feature is mysql rdbms only.

SET @@foreign_key_checks = 0;

INSERT INTO emps VALUES (4, 'Nitin', 40, 5);
INSERT INTO emps VALUES (5, 'Sarang', 40, NULL);
INSERT INTO emps VALUES (2, 'Rahul', 10, 3);
INSERT INTO emps VALUES (1, 'Amit', 10, 4);
INSERT INTO emps VALUES (3, 'Nilesh', 20, 4);

INSERT INTO depts VALUES (10, 'DEV');
INSERT INTO depts VALUES (20, 'QA');
INSERT INTO depts VALUES (30, 'OPS');
INSERT INTO depts VALUES (40, 'ACC');

SET @@foreign_key_checks = 1;

SELECT * FROM depts;
```

```
SELECT * FROM emps;

UPDATE depts SET deptno = 60 WHERE deptno = 10;
-- ON UPDATE CASCADE (in CREATE TABLE emps)
-- when parent is updated 10 --> 60, cascade child updatation 10 --> 60

SELECT * FROM depts;
SELECT * FROM emps;

DELETE FROM depts WHERE deptno = 40;
-- ON DELETE CASCADE (in CREATE TABLE emps)
-- where  parent (40) is deleted, cascade child deletion (emps with dept 40)

SELECT * FROM depts;
SELECT * FROM emps;
```

CHECK

```
CREATE TABLE checked_emp(
empno INT(4),
ename VARCHAR(40),
job VARCHAR(40) CHECK (BINARY job = BINARY UPPER(job)),
mgr INT(4),
hire DATE,
sal DECIMAL(8,2) CHECK (sal BETWEEN 500 AND 5000),
comm DECIMAL(8,2),
deptno INT(4),
CHECK (sal + IFNULL(comm,0.0) >= 800)
);

INSERT INTO checked_emp(empno, ename, job, sal, comm) VALUES (1001, 'JOHN',
'ANALYST', 2900.0, NULL);

INSERT INTO checked_emp(empno, ename, job, sal, comm) VALUES (1002, 'MOTU',
'Director', 4000.0, NULL);
-- error: job must be in upper case

INSERT INTO checked_emp(empno, ename, job, sal, comm) VALUES (1003, 'PATLU',
'SWEEPER', 400.0, NULL);
-- error: sal between 500 and 5000

INSERT INTO checked_emp(empno, ename, job, sal, comm) VALUES (1003, 'PATLU',
'SWEEPER', 800.0, -200.0);
-- error: sal + comm must >= 800
```