Agenda

- views
- Indexes
- Constraints

Views

- It is just a projection of data.
- CREATE VIEW viewname AS SELECT ...;
- In MySQL views are non-materialized i.e. output of SELECT statement (of view) is not saved in server disk.
- Only SELECT query is saved in server disk.

```
SELECT empno, ename, sal, IF(sal<=2000,"POOR","RICH") AS category FROM emp;

CREATE VIEW v_empcategory AS SELECT empno, ename, sal,
IF(sal<=2000,"POOR","RICH") AS category FROM emp;

SHOW TABLES;
SHOW FULL TABLES;
SELECT * FROM v_empcategory;

SELECT category, COUNT(empno) FROM v_empcategory GROUP BY category;
```

- Their are two types of views
    - 1. Simple
        - DQL + DML
    - 2. Complex
        - DQL

```
CREATE VIEW v_empsal AS SELECT empno, ename, sal FROM emp;
SELECT * FROM v_empsal;
INSERT INTO v_empsal VALUES(1001,"e1",1000);

CREATE VIEW v_empjobsummary AS
SELECT job, SUM(sal) salsum, AVG(sal) salavg, MAX(sal) salmax, MIN(sal)
salmin
FROM emp GROUP BY job;

SELECT * FROM v_empjobsummary;

-- we cannot perform DML Operation on v_empjobsummary


CREATE VIEW v_richemp AS SELECT * FROM emp WHERE sal > 2500;
SELECT * FROM v_richemp;
```

```
INSERT INTO v_richemp(empno,ename,sal) VALUES(1002,"e2",2600);
SELECT * FROM v_richemp;

INSERT INTO v_richemp(empno,ename,sal) VALUES(1003,"e3",2200);
SELECT * FROM v_richemp;

SELECT * FROM emp;

ALTER VIEW v_richemp AS SELECT * FROM emp WHERE sal > 2500 WITH CHECK
OPTION;

INSERT INTO v_richemp(empno, ename, sal) VALUES(1004, "e4", 2100);
--error check option failed

CREATE VIEW v_richemp2 AS SELECT empno,ename,sal,comm FROM v_richemp;

SELECT * FROM v_richemp2;

SHOW CREATE VIEW v_richemp2;

DROP VIEW v_richemp;

SELECT * FROM v_richemp2;
-- error: invalid table/view.

DROP VIEW v_richemp2;
```

Indexes

- It is used for faster Searching
- Types of Index
  - Simple Index
  - Unique Index
  - Composite Index
  - Clustered Index

Simple Index

```
SELECT * FROM books;
EXPLAIN FORMAT=JSON
SELECT * FROM books WHERE subject = 'C Programming';

CREATE INDEX idx_books_subject ON books(subject);

EXPLAIN FORMAT=JSON
SELECT * FROM books WHERE subject = 'C Programming';

DESCRIBE books;
SHOW INDEXES FROM books;
```

```
CREATE INDEX idx_books_author ON books(author DESC);
DESCRIBE books;
SHOW INDEXES FROM books;
```

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

CREATE INDEX idx1 ON emps(deptno);
CREATE INDEX idx2 ON depts(deptno);

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

Unique Index

- Duplicate values are not allowed inside the column on which unique index is created.

```
CREATE UNIQUE INDEX idx3 ON emps(ename);
DESCRIBE emps;
SELECT * FROM emps;
SELECT * FROM emps WHERE ename = 'Nitin';

INSERT INTO emps VALUES (6, 'Rahul', 70, 1);
-- error: Duplicate entry

INSERT INTO emps VALUES (7, NULL, 50, 5);
-- (multiple) NULL value is allowed, but duplicate is not allowed.

CREATE UNIQUE INDEX idx4 ON emps(mgr);
-- error
```

Composite Index

```
SELECT * FROM emp;
DESCRIBE emp;

EXPLAIN FORMAT=JSON SELECT * FROM emp WHERE deptno = 20 AND job =
'ANALYST';
CREATE INDEX idx_dj ON emp(deptno ASC, job ASC);

DESCRIBE emp;
SHOW INDEXES FROM emp;
```

Unique Composite Index

```
CREATE TABLE students(std INT, roll INT, name CHAR(30), marks
DECIMAL(5,2));
INSERT INTO students VALUES (1, 1, 's1', 99);
INSERT INTO students VALUES (1, 2, 's2', 96);
INSERT INTO students VALUES (1, 3, 's3', 98);
INSERT INTO students VALUES (2, 1, 's4', 97);
INSERT INTO students VALUES (2, 2, 's5', 95);

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

INSERT INTO students VALUES (1, 2, 's6', 99);
-- error: duplicate combination of std+roll not allowed

SELECT * FROM students;

INSERT INTO students VALUES (3, 1, 's6', 99);
DESCRIBE students;
```

Clustered Index

- Clustered index is auto-created on Primary key.
- It is internally a unique index that is used to lookup rows quickly on server disk.
- If Primary key is not present in the table, then a hidden (synthetic) column is created by RDBMS and Clustered index is created on it.

Drop Index

```
SHOW INDEXES FROM books;
DROP INDEX idx_books_author ON books;
DESCRIBE books;
```

Constraints

- Their are 5 types of constraints
    - 1. Not Null
    - 2. Unique
    - 3. Primary Key
    - 4. Foreign Key
    - 5. Check
- Based on syntax constraints are classified as
    - 1. Column Level constraints
        - NOT NULL,UNIQUE,PRIMARY, FOREIGN, CHECK

```
CREATE TABLE customers(
    id INT PRIMARY KEY,
    name CHAR(30) NOT NULL,
    email CHAR(40) UNIQUE NOT NULL,
    mobile CHAR(12) UNIQUE,
    addr VARCHAR(100)
);
```

- 2. Table Level constraints
  - UNIQUE,FOREIGN,PRIMARY,CHECK

```
    CREATE TABLE customers(
        id INT,
        name CHAR(30) NOT NULL,
        email CHAR(40) NOT NULL,
        mobile CHAR(12),
        addr VARCHAR(100),
        PRIMARY KEY(id),
        UNIQUE(email),
        UNIQUE(mobile)
);
```

1. NOT NULL

- Null values are not allowed in this column
- can be given at column level only

```
CREATE TABLE temp1(c1 INT, c2 INT, c3 INT NOT NULL);
DESCRIBE temp1;
INSERT INTO temp1 VALUES (1, 1, 1);
INSERT INTO temp1 VALUES (NULL, 2, 2);
INSERT INTO temp1(c1,c3) VALUES (3,3);

SELECT * FROM temp1;

INSERT INTO temp1 VALUES (4, 4, NULL);
-- error: c3 cannot be NULL

INSERT INTO temp1(c1,c2) VALUES (5,5);
-- error: c3 cannot be NULL

SHOW INDEXES FROM temp1;
```

2. Unique

- Cannot have duplicate value in the column.

- However NULL value(s) allowed.
- Unique constraint internally creates unique index.
- Unique constraint on combination of multiple columns internally creates Composite Unique index.
- Must be at table level -- UNIQUE(c1,c2).

```
CREATE TABLE temp2(c1 INT, c2 INT, UNIQUE(c1));
-- CREATE TABLE temp2(c1 INT UNIQUE, c2 INT);

CREATE TABLE students(std INT,
roll INT,
name CHAR(30),
marks DECIMAL(5,2),
UNIQUE(std,roll)
);
```