Agenda

- Constraints
- Alter
- ~~PL-SQL~~

Surrogate Primary Key

- These are the keys that are autogenerated
- Mysql has such key called as auto_increment
- ORACLE has such key called as sequences
- MS-SQL has such key called as Identity

```sql
CREATE TABLE items(
id INT PRIMARY KEY AUTO_INCREMENT,
name CHAR(30),
price DECIMAL(5,2)
);

INSERT INTO items(name,price) VALUES('A', 10);
INSERT INTO items(name,price) VALUES('B', 15);
INSERT INTO items(name,price) VALUES('C', 20);
SELECT * FROM items;

ALTER TABLE items AUTO_INCREMENT = 100;

INSERT INTO items(name,price) VALUES('X', 50);
INSERT INTO items(name,price) VALUES('Y', 55);
SELECT * FROM items;

INSERT INTO items(id,name,price) VALUES (1000, 'P', 30);
SELECT * FROM items;

INSERT INTO items(name,price) VALUES('Q', 60);
SELECT * FROM items;
```

4. Foreign Key

- It determines parent child relationship
- Here the primary key from parent table is added as foreign key in child table

```sql
SELECT * FROM emps;
DROP TABLE emps;
DROP TABLE depts;

CREATE TABLE depts (deptno INT PRIMARY KEY, dname VARCHAR(20));
INSERT INTO depts VALUES (10, 'DEV');
INSERT INTO depts VALUES (20, 'QA');
INSERT INTO depts VALUES (30, 'OPS');
INSERT INTO depts VALUES (40, 'ACC');
DESCRIBE depts;

CREATE TABLE emps (empno INT PRIMARY KEY, ename VARCHAR(20), deptno INT, mgr INT,
FOREIGN KEY (deptno) REFERENCES depts(deptno));
INSERT INTO emps VALUES (1, 'Amit', 10, 4);
INSERT INTO emps VALUES (2, 'Rahul', 10, 3);
INSERT INTO emps VALUES (3, 'Nilesh', 20, 4);

INSERT INTO emps VALUES (4, 'Nitin', 50, 5);
-- error: a foreign key constraint fails
INSERT INTO emps VALUES (5, 'Sarang', 50, NULL);
-- error: a foreign key constraint fails

INSERT INTO emps VALUES (6, 'Vishal', NULL, 3);
-- FK can be NULL

DELETE FROM depts WHERE deptno=40;
DELETE FROM depts WHERE deptno=30;
-- error: a foreign key constraint fails

DROP TABLE depts;
-- error: Cannot drop table 'depts' referenced by a foreign key constraint
```

- Cannot add/update in child row, if corresponding row is absent in parent table.
- Cannot delete parent row, if corresponding rows are present in child table.
- Create the foreign key as ON DELETE CASCADE ON UPDATE CASCADE

```
DROP TABLE emps;
DROP TABLE depts;

CREATE TABLE depts (deptno INT PRIMARY KEY, dname VARCHAR(20));
INSERT INTO depts VALUES (10, 'DEV');
INSERT INTO depts VALUES (20, 'QA');
INSERT INTO depts VALUES (30, 'OPS');
INSERT INTO depts VALUES (40, 'ACC');

CREATE TABLE emps (empno INT, ename VARCHAR(20), deptno INT, mgr INT, FOREIGN KEY
(deptno) REFERENCES depts(deptno) ON DELETE CASCADE ON UPDATE CASCADE);

INSERT INTO emps VALUES (1, 'Amit', 10, 4);
INSERT INTO emps VALUES (2, 'Rahul', 10, 3);
INSERT INTO emps VALUES (3, 'Nilesh', 20, 4);

DELETE FROM depts WHERE deptno = 20;
-- ON DELETE CASCADE: If parent row is deleted, corresponding child rows will be
deleted automatically.

UPDATE depts SET deptno=100 WHERE dname='DEV';
-- ON UPDATE CASCADE: If parent row (primary key) is updated, corresponding child
```

```
rows (foreign key) will be updated automatically.

DROP TABLE depts;
-- error: Cannot drop table 'depts' referenced by a foreign key constraint
```

- If PK is Composite primary key, the Foreign key can be Composite key.

```
CREATE TABLE students(
email CHAR(40),
password CHAR(40),
name CHAR(40),
course_code CHAR(20),
PRIMARY KEY(course_code, email)
);

CREATE TABLE marks(
id INT PRIMARY KEY,
subject CHAR(20),
marks INT,
course_code CHAR(20),
email CHAR(40),
FOREIGN KEY (course_code,email) REFERENCES students(course_code, email);
);
```

- Foreign key internally creates index on the table. It also helps in faster searching.
- Foreign key constraint can be disabled temporarily in some cases (like backup/restore).

```
SELECT @@foreign_key_checks;

CREATE TABLE dept_backup(deptno INT, dname CHAR(40), loc CHAR(40), PRIMARY
KEY(deptno));

CREATE TABLE emp_backup(empno INT, ename CHAR(40), sal DECIMAL(8,2), deptno
INT, PRIMARY KEY(empno), FOREIGN KEY (deptno) REFERENCES dept_backup(deptno));

INSERT INTO dept_backup SELECT * FROM dept;
SELECT * FROM dept_backup;

SET @@foreign_key_checks=0;

INSERT INTO emp_backup(empno,ename,sal,deptno) SELECT empno,ename,sal,deptno
FROM emp;
-- insert is fast, bcoz FK is disabled.

INSERT INTO emp_backup VALUES(1000, 'JOHN', 2000, 60);
-- allowed, bcoz FK checks are disabled -- but wrong

SET @@foreign_key_checks=1;
-- FK check is enabled -- further DML ops.
```

```
INSERT INTO emp_backup VALUES(1001, 'JACK', 2200, 60);
-- error: FK checks are enabled

SELECT * FROM emp_backup;
```

- Foreign key can be for the same table.
- It is called as "self-referencing" FK.

```
CREATE TABLE emps(
empno INT,
ename CHAR(40),
mgr INT,
PRIMARY KEY(empno),
FOREIGN KEY(mgr) REFERENCES emps(empno)
);
```

5. Check

- Arbitrary conditions (application specific) to be applied on the column.
- Do not work in MySQL version <= 8.0.15

```
CREATE TABLE employees(
id INT PRIMARY KEY,
ename CHAR(40) CHECK (LENGTH(ename) > 1),
age INT NOT NULL CHECK (age > 18),
sal DECIMAL(7,2) CHECK (sal > 1000),
);

INSERT INTO employees VALUES (1, 'e', 20, 2000);
-- error: LENGTH(ename) > 1

INSERT INTO employees VALUES (1, 'e1', 16, 2000);
-- error: age > 18

INSERT INTO employees VALUES (1, 'e1', 20, 900);
-- error: sal > 1000

INSERT INTO employees VALUES (1, 'e1', 20, 1100);
```

Constraint names

```
CREATE TABLE employees(
id INT,
ename CHAR(40),
age INT NOT NULL,
sal DECIMAL(7,2),
deptno INT,
PRIMARY KEY(id),
FOREIGN KEY(deptno) REFERENCES departments(deptno)
);
-- names of constraints are given auto by db

CREATE TABLE employees(
id INT,
ename CHAR(40),
age INT NOT NULL,
sal DECIMAL(7,2),
deptno INT,
CONSTRAINT pk_empid PRIMARY KEY(id),
CONSTRAINT fk_deptno FOREIGN KEY(deptno) REFERENCES departments(deptno)
);

-- to display the Constraints
SHOW CREATE TABLE emps;
```

ALTER Table

- Add column, Remove column, Change column data type/name, Add/Remove constraint
- Not recommeded in production database.
- After alteration table storage become unefficient.

```
DESCRIBE emp_backup;

ALTER TABLE emp_backup ADD COLUMN job CHAR(20);
SELECT * FROM emp_backup;
UPDATE emp_backup e SET e.job = (SELECT job FROM emp WHERE empno = e.empno);
DESCRIBE emp_backup;

ALTER TABLE emp_backup MODIFY job VARCHAR(40);
-- can change data type to compatible data type
DESCRIBE emp_backup;

ALTER TABLE emp_backup MODIFY job INT;
-- error: cannot change data type to incompatible.

ALTER TABLE emp_backup CHANGE ename name CHAR(30);
DESCRIBE emp_backup;

ALTER TABLE emp_backup DROP COLUMN sal;
DESCRIBE emp_backup;
```

- Alter table to add constraints

```sql
ALTER TABLE emp ADD PRIMARY KEY (empno);
ALTER TABLE emp ADD UNIQUE(ename);

SHOW CREATE TABLE emp;

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

- Alter table to delete constraints

```sql
ALTER TABLE emp DROP PRIMARY KEY;

SHOW CREATE TABLE emp;

ALTER TABLE emp DROP CONSTRAINT ename;
ALTER TABLE emp DROP CONSTRAINT emp_ibfk_1;
```