# SB101

Day-11

# SQL PART-III
# Joins & Subqueries

# SQL Queries for Foreign Key

**To add foreign key using create table**

```
CREATE TABLE table-name(
  col-name data-type(size) constraint,
  col-name data-type(size) constraint,
  ..
  col-name data-type(size) constraint,
  CONSTRAINT constraint-name FOREIGN KEY(col-name)
  REFERENCES  parent-table-name(referenced-column-name));
```

**To add foreign key using ALTER TABLE**

```
ALTER TABLE table-name ADD CONSTRAINT constraint-name FOREIGN KEY
(col-name) REFERENCES parent-table(column-name)
```

**To delete foreign key using ALTER TABLE**

```
ALTER TABLE table-name DROP FOREIGN KEY constraint-name;
```

# JOIN Operation

**Used to fetch data from more than one table using a single SQL query.**

**Joins are of multiple types**

**Cross Join  Inner Join  Left Outer Join  Right Outer Join  Self Join**

## Cross Join

**Father of all join operations**

**In the result of cross join, Degree of both tables is added and cardinality of both tables is multiplied**

```
SELECT projection FROM left-table CROSS JOIN right-table;
                        OR
SELECT projection FROM left-table, right-table;
```

***You can specify column list to limit the number of columns and you can pick the columns of your choice from both tables. One special case of concern is when both tables have column with same name then in this case ambiguity (confusion) occurs such that DBMS unable to decide to fetch column from which table. To overcome this ambiguity we have to use table-name along with the column-name. To make the syntax short we can alias the table-name also.***

# JOIN Operation

## Inner Join

- Used to fetch the records that are common for both the tables i.e. the records that are participating in the relationship
- This is most commonly used join operation

```
SELECT projection FROM left-table
INNER JOIN right-table ON left-table.col-name = right-table.col-name;
                                    OR
SELECT projection FROM left-table, right-table
WHERE left-table.col-name = right-table.col-name;
```

- Both the syntaxes will produce same result but the former query is having better performance because in the latter query cross join will be performed then WHERE clause will filter the record so all that will take lot of time than the former syntax. Use of former syntax is preferred
- You can add additional conditions in join using AND operator; also Join operation can be applied on any number of tables

# JOIN Operation

## Left Outer Join

**Used to fetch the all records from the left table + records that are common for both the tables i.e. the non participating records from the left table will be part of the result but only participating records from right table will be part of the result**

```
SELECT projection FROM left-table
LEFT JOIN right-table ON left-table.col-name = right-table.col-name;
```

## Right Outer Join

**Used to fetch the all records from the right table + records that are common for both the tables i.e. the non participating records from the right table will be part of the result but only participating records from left table will be part of the result**

```
SELECT projection FROM left-table
RIGHT JOIN right-table ON left-table.col-name = right-table.col-name;
```

# JOIN Operation

## Self Join

- The same table is working as left and right table both
- Used when table has self referencing foreign key

# Subquery

- **Subquery is query inside another query such that the subquery can be written in SELECT, FROM, WHERE and HAVING clause**
- **Subquery must be in the parenthesis**
- **Subquery is executed first and its result will be used in the outer query.**
- **Subquery into the SELECT clause should return only one column**
- ***1. When subquery is used with the FROM clause then a temporary table will be generated and this temporary table is not having any name that's why it is mandatory to alias with the result of subquery with FROM clause. This table alias may or may not be used with the column names in outer query.***
  ***2. If column aliasing is done in the inner query then aliased columns names has to be use with the outer query.***
- **Subquery can be used with DML statement. Aliasing is not required if outer query and inner query are applied on the different tables but if they are applied on same table then it is necessary to use alias in the sub query.**

# Subquery

- The subquery that returns only one row is called single row subquery. The result of Single Row Subquery can be compared using operator like >, <, <=, >=, =, !=, BETWEEN.. AND... etc.
- The subquery that returns only multiple rows is called multi row subquery. The result of Multi Row Subquery can be compared using operator IN, ANY and ALL

# Transaction Management

Say A has received a cheque of INR 1000/- from B. A presented that cheque to bank and then he thought soon 1000/- will be deducted from B's account and credited to A's account. But suddenly A thought that what happened if 1000/- deducted from B's account and then power failure takes place. Is A right? How DBA will answer this questions? Answer is transaction.

A transaction is a sequential group of database manipulation operations, which is performed as if it were one single work unit.

In other words, a transaction will never be complete unless each individual operation within the group is successful. If any operation within the transaction fails, the entire transaction will fail.

A transaction has ACID (Atomicity, Consistency, Isolation, Durability) properties, Take a look at these properties in brief

# ACID Property

**Atomicity**: Atomicity refers to the ability of the database to guarantee that either all of the tasks of a transaction are performed or none of them are.

**Consistency**: Consistency property ensures that the database remains in a consistent state before the start of the transaction and after the transaction is over (whether successful or not).

**Isolation**: It refers to the requirement that other operations cannot access or see the data in an intermediate state during a transaction.

**Durability**: Durability refers to the guarantee that once the user has been notified of success, the transaction will persist, and not be undone. This means it will survive system failure, and that the database system has checked the integrity constraints and won't need to abort the transaction.

# Start Transaction, Commit & RollBack

To start a transaction, use following command

```
START TRANSACTION;
```

Start Transaction commits he current transaction and start a new transaction. It tells MySql that a new transaction is beginning and statements followed by transaction must be treated as unit. To save changes made during the transaction use following command

```
COMMIT or COMMIT WORK
```

| Here work keyword is optional.

Tip: Say you forgot to place commit statement after update queries then these queries will not have any effect on database.

During a transaction if any error takes place then the entire transaction will be can be undone using ROLLBACK statement. The ROLLBACK statement cancels the entire transaction and put the database at the beginning point of the transaction. ROLLBACK statement has following

```
ROLLBACK Or ROLLBACK WORK
```

| Here work keyword is optional.

# SAVEPOINT & AUTOCOMMIT

Sometimes it is not require to cancel the entire transaction it is sufficient to ROLLBACK only a small portion of a transaction, to do so we have to mark SAVEPOINT in transaction. A SAVEPOINT is a marker in a transaction that allow to ROLLBACK a database to the marked point. All changes made to the database after the SAVEPOINT are discarded and changes made prior to the transaction remain unchanged. To insert a SAVEPOINT use statement like

```
SAVEPOINT <savepoint-name>
```

To ROLLBACK to the marked SAVEPOINT use following syntax

```
ROLLBACK TO SAVEPOINT <savepoint-name>
```

By default AUTOCOMMIT mode is on in MySql, It means all SQL statements are automatically committed by MySql. To set MySql mode off use following syntax

```
SET AUTOCOMMIT=0;
```

Making AUTOCOMMIT to off work only for a single session. Staring new session automatically set AUTOCOMMIT to on.

# SAVEPOINT & AUTOCOMMIT

**To explicitly set AUTOCOMMIT mode to on just use following syntax**

```
SET AUTOCOMMIT=1;
```

**Tip: Making AUTOCOMMIT off has no impact over DDL statements. They are always committed by database automatically.**

QUERY?