SQL Triggers Notes

Triggers in DBMS:

DBMS:

> DATABASE

> DATA

> SOFTWARE

> TECHNOLOGY

> SYSTEM MANAGEMENT

A Trigger is a user-defined SQL command that is invoked automatically in response to an event such as INSERT, DELETE, or UPDATE.

Syntax:

```
CREATE TRIGGER trigger_name trigger_time trigger_event
ON table_name FOR EACH ROW
BEGIN
    -- Trigger Logic
END;
```

Trigger Time: BEFORE, AFTER

Trigger Event: INSERT, UPDATE, DELETE

```
CREATE DATABASE triggers;
USE triggers;
SHOW TABLES;
```

```
# BEFORE INSERT Trigger
CREATE TABLE customers (
    cust_id INT,
    age INT,
    name VARCHAR(30)
);

DELIMITER //
CREATE TRIGGER age_verify
BEFORE INSERT ON customers
FOR EACH ROW
BEGIN
    IF NEW.age < 0 THEN
        SET NEW.age = 0;
    END IF;
END;
//

INSERT INTO customers VALUES
(101, 27, "James"),
(102, -40, "Ammy"),
(103, 32, "Ben"),
(104, -39, "Angela");

# INSERT Trigger
CREATE TABLE customers1 (
    id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    name VARCHAR(25) NOT NULL,
    email VARCHAR(30),
    birthdate DATE
);

CREATE TABLE message (
    id INT AUTO_INCREMENT,
    message_id INT,
    message VARCHAR(300) NOT NULL,
    PRIMARY KEY(id, message_id)
);

DELIMITER //
CREATE TRIGGER check_null_dob
AFTER INSERT
ON customers1
FOR EACH ROW
BEGIN
    IF NEW.birthdate IS NULL THEN
        INSERT INTO message (message_id, message)
        VALUES (NEW.id, CONCAT('Hi ', NEW.name, ', please update your birthday.'));
    END IF;
END;
//

INSERT INTO customers1 (name, email, birthdate)
VALUES ('Nancy', 'nancy223.com', NULL),
```

```
        ('Ronald', 'ronald@257.com', '1999-09-12');

SELECT * FROM message;

# BEFORE UPDATE Trigger
CREATE TABLE employees (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(25),
    age INT,
    salary FLOAT
);

INSERT INTO employees VALUES
(101, "Jimmy", 35, 70000),
(102, "Shane", 30, 55000),
(103, "Marry", 28, 62000),
(104, "Dwayne", 37, 57000),
(105, "Sara", 32, 72000),
(106, "Ammy", 35, 80000),
(107, "Jack", 40, 100000);

DELIMITER //
CREATE TRIGGER upd_trigger
BEFORE UPDATE
ON employees
FOR EACH ROW
BEGIN
```

```
    IF NEW.salary = 10000 THEN
        SET NEW.salary = 85000;
    ELSEIF NEW.salary < 10000 THEN
        SET NEW.salary = 72000;
    END IF;
END;
//

UPDATE employees SET salary = 8000;

# BEFORE DELETE Trigger
CREATE TABLE salary (
    eid INT PRIMARY KEY,
    valid_from DATE NOT NULL,
    amount FLOAT NOT NULL
);

INSERT INTO salary VALUES
(101, '2005-05-01', 55000),
(102, '2007-09-01', 75000),
(103, '2006-09-01', 75000);

SELECT * FROM salary;

CREATE TABLE salarydel (
    id INT PRIMARY KEY AUTO_INCREMENT,
    eid INT,
```

```
    valid_from DATE NOT NULL,
    amount FLOAT NOT NULL,
    deleted_at TIMESTAMP DEFAULT NOW()
);

DELIMITER $$
CREATE TRIGGER salary_delete
BEFORE DELETE
ON salary
FOR EACH ROW
BEGIN
    INSERT INTO salarydel(eid, valid_from, amount)
    VALUES (OLD.eid, OLD.valid_from, OLD.amount);
END $$
DELIMITER ;

DELETE FROM salary WHERE eid = 103;

SELECT * FROM salarydel;
```

SQL Concepts:

- Subqueries in SQL
- Stored Procedures
- Triggers in SQL
- Views in SQL
- Window Functions in SQL

```
# Example Subquery

USE sql_intro;

SELECT * FROM employees;


SELECT emp_name, dept, salary

FROM employees

WHERE salary > (SELECT AVG(salary) FROM employees);


Database Name: classicmodels


USE classicmodels;


# Example Queries

SELECT emp_name, gender, dept, salary

FROM employees

WHERE salary > (SELECT salary FROM employees WHERE emp_name = 'John');


SELECT * FROM products;

SELECT * FROM orderdetails;


SELECT productcode, productname, msrp

FROM products

WHERE productcode IN (SELECT productcode FROM orderdetails);
```