**TRIGGERS**

**Trigger #1**

|  |  |
| --- | --- |
| **Trigger Name** | BEFORE\_WORKCENTERS\_INSERT |
| **Trigger Type** | Before Insert |
| **Tables** | WORKCENTERS  WORKCENTERSTATS |
| **Description** | Updates the total capacity in the Workcenterstats table before a new work center is inserted into Workcenter table |

**Tables Creation:**

DROP TABLE IF EXISTS WORKCENTERS;

DROP TABLE IF EXISTS WORKCENTERSTATS;

CREATE TABLE WORKCENTERS (

ID INT AUTO\_INCREMENT PRIMARY KEY,

NAME VARCHAR(100) NOT NULL,

CAPACITY INT NOT NULL

);

CREATE TABLE WORKCENTERSTATS(

TOTALCAPACITY INT NOT NULL

);

**Trigger Creation:**

DELIMITER //

CREATE TRIGGER BEFORE\_WORKCENTERS\_INSERT BEFORE INSERT ON WORKCENTERS FOR EACH ROW

BEGIN

DECLARE ROWCOUNT INT;

SELECT COUNT(\*) INTO ROWCOUNT FROM WORKCENTERSTATS;

IF ROWCOUNT > 0 THEN

UPDATE WORKCENTERSTATS SET TOTALCAPACITY = TOTALCAPACITY + NEW.CAPACITY;

ELSE

INSERT INTO WORKCENTERSTATS(TOTALCAPACITY) VALUES(NEW.CAPACITY);

END IF;

END //

DELIMITER ;

**Trigger Testing:**

|  |  |  |
| --- | --- | --- |
| Step # | Step Description | Expected result |
| 1 | Insert a new row into Workcenters table  INSERT INTO WORKCENTERS(NAME, CAPACITY) VALUES(‘Mold machine’, 100); | - |
| 2 | Query data from the Workcenters table  SELECT \* FROM WORKCENTERS; | TOTALCAPACITY in Workcenterstats table should be 100 |
| 3 | Insert a new work center  INSERT INTO WORKCENTERS(NAME, CAPACITY) VALUES(‘PACKING’, 200); | - |
| 4 | Finally, query data from Workcenterstats table  SELECT \* FROM WORKCENTERSTATS; | TOTALCAPACITY in Workcenterstats table should be 300 |

**Trigger #2**

|  |  |
| --- | --- |
| **Trigger Name** | AFTER\_MEMBERS\_INSERT |
| **Trigger Type** | After Insert |
| **Tables** | MEMBERS  REMINDERS |
| **Description** | Inserts a reminder into the Reminders table if the birthdate in the Members table is null |

**Tables Creation:**

DROP TABLE IF EXISTS MEMBERS;

DROP TABLE IF EXISTS REMINDERS;

CREATE TABLE MEMBERS (

ID INT AUTO\_INCREMENT,

NAME VARCHAR(100) NOT NULL,

EMAIL VARCHAR(255),

BIRTHDATE DATE,

PRIMARY KEY (ID)

);

CREATE TABLE REMINDERS (

ID INT AUTO\_INCREMENT,

MEMBERID INT,

MESSAGE VARCHAR(255) NOT NULL,

PRIMARY KEY (ID, MEMBERID)

);

**Trigger Creation:**

DELIMITER //

CREATE TRIGGER AFTER\_MEMBERS\_INSERT AFTER INSERT ON MEMBERS FOR EACH ROW

BEGIN

IF NEW.BIRTHDATE IS NULL THEN

INSERT INTO REMINDERS(MEMBERID, MESSAGE) VALUES(NEW.ID, CONCAT('HI ', NEW.NAME, ', PLEASE UPDATE YOUR DATE OF BIRTH'));

END IF;

END //

DELIMITER ;

**Trigger Testing:**

|  |  |  |
| --- | --- | --- |
| Step # | Step Description | Expected result |
| 1 | Insert two rows into Members table  INSERT INTO MEMBERS(NAME, EMAIL, BIRTHDATE) VALUES('JOHN', 'JOHN@EXAMPLE@COM', NULL)  INSERT INTO MEMBERS(NAME, EMAIL, BIRTHDATE) VALUES('KIM', 'KIM@EXAMPLE@COM', '2012-05-03') | - |
| 2 | Query data from the Members table  SELECT \* FROM MEMBERS; | Members table should contain two records. One of them should have birthdate value as null |
| 3 | Query data from Reminders table  SELECT \* FROM REMINDERS; | Reminders table should have one row with memberid value as 1 |

**Trigger #3**

|  |  |
| --- | --- |
| **Trigger Name** | BEFORE\_SALES\_UPDATE |
| **Trigger Type** | Before update |
| **Tables** | SALES |
| **Description** | If the new value in quantity column is three times greater than the old value, then the trigger raises an error |

**Tables Creation:**

CREATE TABLE SALES(

ID INT AUTO\_INCREMENT,

PRODUCT VARCHAR(100) NOT NULL,

QUANTITY INT NOT NULL DEFAULT 0,

FISCALMONTH TINYINT NOT NULL,

FISCALYEAR SMALLINT NOT NULL,

CHECK(FISCALMONTH >= 1 AND FISCALMONTH <= 12),

CHECK(FISCALYEAR BETWEEN 2000 AND 2050),

CHECK(QUANTITY > 0),

UNIQUE(PRODUCT, FISCALMONTH, FISCALYEAR),

PRIMARY KEY (ID)

);

INSERT INTO SALES(PRODUCT, QUANTITY, FISCALYEAR, FISCALMONTH) VALUES

('2003 HARLEY-DAVIDSON EAGLE DRAG BIKE', 120, 2020, 1),

('1969 CORVAIR MONZA', 150, 2020, 1),

('1970 PLYMOUTH HEMI CUDA', 200, 2020, 1);

**Trigger Creation:**

DELIMITER //

CREATE TRIGGER BEFORE\_SALES\_UPDATE BEFORE UPDATE ON SALES FOR EACH ROW

BEGIN

DECLARE ERRORMESSAGE VARCHAR(255);

SET ERRORMESSAGE = CONCAT('THE NEW QUANTITY ', NEW.QUANTITY, ' CANNOT BE THREE TIMES GREATER THAN THE OLD QUANTITY ', OLD.QUANTITY);

IF NEW.QUANTITY > OLD.QUANTITY \* 3 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = ERRORMESSAGE;

END IF;

END //

DELIMITER ;

**Trigger Testing:**

|  |  |  |
| --- | --- | --- |
| Step # | Step Description | Expected result |
| 1 | Update quantity to 150 for row with id 1  UPDATE SALES SET QUANTITY = 150  WHERE ID = 1; | No error |
| 2 | Verify the update  SELECT \* FROM SALES; | Row with id = 1 should have quantity value as 150 |
| 3 | Update quantity to 500 for row with id 1  UPDATE SALES SET QUANTITY = 500  WHERE ID = 1; | Error Code: 1644. The new quantity 500 cannot be three times greater than the old quantity 150. |

**Trigger #4**

|  |  |
| --- | --- |
| **Trigger Name** | AFTER\_SALES\_UPDATE |
| **Trigger Type** | After update |
| **Tables** | SALES, SALESCHANGES |
| **Description** | If a value in quantity column of Sales table is updated, then the trigger inserts a new row to Saleschanges table to log the updation |

**Tables Creation:**

CREATE TABLE SALESCHANGES (

ID INT AUTO\_INCREMENT PRIMARY KEY,

SALESID INT,

BEFOREQUANTITY INT,

AFTERQUANTITY INT,

CHANGEDAT TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP

);

**Trigger Creation:**

CREATE TRIGGER AFTER\_SALES\_UPDATE AFTER UPDATE ON SALES FOR EACH ROW

BEGIN

IF OLD.QUANTITY <> NEW.QUANTITY THEN

INSERT INTO SALESCHANGES(SALESID, BEFOREQUANTITY, AFTERQUANTITY)

VALUES(OLD.ID, OLD.QUANTITY, NEW.QUANTITY);

END IF;

END //

DELIMITER ;

**Trigger Testing:**

|  |  |  |
| --- | --- | --- |
| Step # | Step Description | Expected result |
| 1 | Update the quantity to 350 for row with id 1 of Sales table  UPDATE SALES SET QUANTITY = 350  WHERE ID = 1; | No error |
| 2 | Query data from Saleschanges  SELECT \* FROM SALESCHANGES; | A row should be inserted to Saleschanges table |
| 3 | Update quantity by increasing 10% for all three rows of Sales table  UPDATE SALES SET QUANTITY = CAST(QUANTITY \* 1.1 AS UNSIGNED); | No error |
| 4 | Query data from Saleschanges  SELECT \* FROM SALESCHANGES; | Three more rows should be inserted to Saleschanges table |

**Trigger #5**

|  |  |
| --- | --- |
| **Trigger Name** | BEFORE\_SALARIES\_DELETE |
| **Trigger Type** | Before delete |
| **Tables** | Salaries, Salaryarchives |
| **Description** | Inserts a new row to Salaryarchives table before a row from Salaries table is deleted |

**Tables Creation:**

CREATE TABLE SALARIES(

EMPLOYEENUMBER INT PRIMARY KEY,

VALIDFROM DATE NOT NULL,

SALARY DECIMAL(12, 2) NOT NULL DEFAULT 0

);

INSERT INTO SALARIES(EMPLOYEENUMBER, VALIDFROM, SALARY)

VALUES

(1002, '2000-01-01', 50000),

(1056, '2000-01-01', 60000),

(1076, '2000-01-01', 70000);

CREATE TABLE SALARYARCHIVES(

ID INT PRIMARY KEY AUTO\_INCREMENT,

EMPLOYEENUMBER INT,

VALIDFROM DATE NOT NULL,

SALARY DEC(12, 2) NOT NULL DEFAULT 0,

DELETEDAT TIMESTAMP DEFAULT NOW()

);

**Trigger Creation:**

DELIMITER //

CREATE TRIGGER BEFORE\_SALARIES\_DELETE BEFORE DELETE ON SALARIES FOR EACH ROW

BEGIN

INSERT INTO SALARYARCHIVES(EMPLOYEENUMBER, VALIDFROM, SALARY)

VALUES(OLD.EMPLOYEENUMBER, OLD.VALIDFROM, OLD.SALARY);

END //

DELIMITER ;

**Trigger Testing:**

|  |  |  |
| --- | --- | --- |
| Step # | Step Description | Expected result |
| 1 | Delete a row from Salaries table  DELETE FROM SALARIES WHERE EMPLOYEENUMBER = 1002; | - |
| 2 | Query data from SalaryArchives table  SELECT \* FROM SALARYARCHIVES; | A new row should be inserted in Salaryarchives table |
| 3 | Delete all rows from Salaries table  DELETE FROM SALARIES; | - |
| 4 | Query data from SalaryArchives table  SELECT \* FROM SALARYARCHIVES; | Two new rows should be inserted in Salaryarchives table |

**Trigger #6**

|  |  |
| --- | --- |
| **Trigger Name** | AFTER\_SALARIES\_DELETE |
| **Trigger Type** | After delete |
| **Tables** | Salaries, Salarybudgets |
| **Description** | Updates the total salary in Salarybudgets table after a row is deleted from Salaries table |

**Tables Creation:**

CREATE TABLE SALARIES(

EMPLOYEENUMBER INT PRIMARY KEY,

VALIDFROM DATE NOT NULL,

SALARY DECIMAL(12, 2) NOT NULL DEFAULT 0

);

INSERT INTO SALARIES(EMPLOYEENUMBER, VALIDFROM, SALARY)

VALUES

(1002, '2000-01-01', 50000),

(1056, '2000-01-01', 60000),

(1076, '2000-01-01', 70000);

CREATE TABLE SALARYBUDGETS(TOTAL DECIMAL(15, 2) NOT NULL);

INSERT INTO SALARYBUDGETS(TOTAL) SELECT SUM(SALARY) FROM SALARIES;

**Trigger Creation:**

CREATE TRIGGER AFTER\_SALARIES\_DELETE AFTER DELETE ON SALARIES FOR EACH ROW

UPDATE SALARYBUDGETS SET TOTAL = TOTAL - OLD.SALARY;

**Trigger Testing:**

|  |  |  |
| --- | --- | --- |
| Step # | Step Description | Expected result |
| 1 | Delete a row from Salaries table  DELETE FROM SALARIES WHERE EMPLOYEENUMBER = 1002; | - |
| 2 | Query data from SalaryBudgets table  SELECT \* FROM SALARYBUDGETS; | 'Total' should reduce by deleted salary |
| 3 | Delete all rows from Salaries table  DELETE FROM SALARIES; | - |
| 4 | Query data from SalaryBudgets table  SELECT \* FROM SALARYBUDGETS; | 'Total' should be 0 |

\*\*\*\*\*\*