Trigger Examples
Eric J. Schwabe
CSC 355 Winter 2020

(This document is for study and review purposes only. Copying any part of the examples in this document into a submitted assignment constitutes plagiarism.)

=====

```
-- Workers.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

-- First, set up table of workers' names (just letters),
-- department numbers (just digits), and salaries, and
-- display both the table and the total of the salaries

DROP TABLE WORKER CASCADE CONSTRAINTS;

CREATE TABLE WORKER
(
      Name        CHAR(1),
      DeptNumber  NUMBER(1,0),
      Salary      NUMBER(8,2),

      CONSTRAINT PK_WORKER
            PRIMARY KEY (Name)
);

INSERT INTO WORKER VALUES('A', 5, 30000);
INSERT INTO WORKER VALUES('B', 2, 45000);
INSERT INTO WORKER VALUES('C', 5, 70000);
INSERT INTO WORKER VALUES('D', 2, 55000);
INSERT INTO WORKER VALUES('E', 1, 25000);
INSERT INTO WORKER VALUES('F', 3, 90000);
INSERT INTO WORKER VALUES('G', 4, 90000);
INSERT INTO WORKER VALUES('H', 2, 39000);
INSERT INTO WORKER VALUES('I', 1, 36000);
INSERT INTO WORKER VALUES('J', 3, 60000);
INSERT INTO WORKER VALUES('K', 5, 76000);
INSERT INTO WORKER VALUES('L', 3, 40000);
INSERT INTO WORKER VALUES('M', 4, 85000);
INSERT INTO WORKER VALUES('N', 1, 39000);
INSERT INTO WORKER VALUES('O', 2, 42000);

SELECT * FROM WORKER;

SELECT SUM(Salary) FROM WORKER;

-- This trigger tests if inserting another employee
-- will yield a total of more than one million dollars
-- in salaries, and if it will, it raises an error
-- message and cancels the insert


CREATE OR REPLACE TRIGGER SalaryCap AFTER INSERT OR UPDATE ON WORKER

DECLARE
      total INTEGER;
```

```
BEGIN
      SELECT SUM(Salary)
      INTO total
      FROM WORKER;

      DBMS_OUTPUT.PUT_LINE(total);

      IF (total > 1000000) THEN
            RAISE_APPLICATION_ERROR(-20001, 'Million Dollar Limit Exceeded');
      END IF;
END;
/

=====
```

```
=====

-- Salaries.sql
-- Eric J. Schwabe
-- CSC 355 Winter 2020

-- Trigger example modified from Elmasri&Navathe, Sec 26.1

-- First, build the EMPLOYEE and DEPARTMENT tables

DROP TABLE EMPLOYEE CASCADE CONSTRAINTS;
DROP TABLE DEPARTMENT CASCADE CONSTRAINTS;

CREATE TABLE DEPARTMENT
(
      DNo         NUMBER(1, 0),
      DName       VARCHAR2(20),
      TotalSalary NUMBER(9, 2),

      PRIMARY KEY (Dno)
);

CREATE TABLE EMPLOYEE
(
      EID         CHAR(3),
      Name        VARCHAR2(20),
      DNo         NUMBER(1, 0)      NOT NULL,
      Salary      NUMBER(9,2),

      PRIMARY KEY (EID),

      FOREIGN KEY (DNo)
            REFERENCES DEPARTMENT(DNo)
);


-- This trigger updates the total salary for a department
-- when an employee is added to that department

CREATE OR REPLACE TRIGGER AddingEmployee
      AFTER INSERT ON EMPLOYEE
FOR EACH ROW

BEGIN
      DBMS_OUTPUT.PUT_LINE('Adding new employee');

      UPDATE DEPARTMENT
      SET TotalSalary = TotalSalary + :new.Salary
      WHERE Dno = :new.Dno;

      INSERT INTO LOGFILE VALUES ('INSERT', NULL, :new.EID, sysdate());
END;
/


-- This trigger updates the total salary for a department
-- when an employee in that department has their salary changed

CREATE OR REPLACE TRIGGER ChangingSalary
      AFTER UPDATE OF Salary ON EMPLOYEE
```

```
FOR EACH ROW

BEGIN
      DBMS_OUTPUT.PUT_LINE('Changing employee salary');

      UPDATE DEPARTMENT
      SET TotalSalary = TotalSalary + :new.Salary - :old.Salary
      WHERE Dno = :new.Dno;

      INSERT INTO LOGFILE VALUES ('UPDATE', 'Salary', :new.EID, sysdate());
END;
/


-- This trigger updates the total salary for two departments
-- when a employee is moved from one department to another

CREATE OR REPLACE TRIGGER ChangingDepartment
      AFTER UPDATE OF Dno ON EMPLOYEE
FOR EACH ROW

BEGIN
      DBMS_OUTPUT.PUT_LINE('Changing employee department');

      UPDATE DEPARTMENT
      SET TotalSalary = TotalSalary + :new.Salary
      WHERE Dno = :new.Dno;

      UPDATE DEPARTMENT
      SET TotalSalary = TotalSalary - :old.Salary
      WHERE Dno = :old.Dno;

      INSERT INTO LOGFILE VALUES ('UPDATE', 'DNo', :new.EID, sysdate());
END;
/


-- This trigger updates the total salary for a department
-- when a employee is deleted

CREATE OR REPLACE TRIGGER RemovingEmployee
      AFTER DELETE ON EMPLOYEE
FOR EACH ROW

BEGIN
      DBMS_OUTPUT.PUT_LINE('Removing employee');

      UPDATE DEPARTMENT
      SET TotalSalary = TotalSalary - :old.Salary
      WHERE Dno = :old.Dno;

      INSERT INTO LOGFILE VALUES ('DELETE', NULL, :old.EID, sysdate());
END;
/


-- All of the above triggers include logging of operations on EMPLOYEE
-- to the table LOGFILE(Operation, OpType, OpRow, OpDate)...

-- If we try to add a record to EMPLOYEE where the DNo is not for an existing
```

```
-- department, add a record to DEPARTMENT first so that referential integrity
-- will be maintained

CREATE OR REPLACE TRIGGER InsureReferentialIntegrity
      BEFORE INSERT ON EMPLOYEE
FOR EACH ROW

DECLARE
      Flag INTEGER;
BEGIN
      DBMS_OUTPUT.PUT_LINE('Checking referential integrity...');

      SELECT COUNT(DNo)
      INTO Flag
      FROM DEPARTMENT
      WHERE DNo = :new.DNo;

      IF (Flag = 0) THEN
            INSERT INTO DEPARTMENT
                  VALUES (:new.DNo, NULL, 0);
            DBMS_OUTPUT.PUT_LINE (' Placeholder Department ' || :new.DNo ||
                  ' added, DName needs to be updated!');
      END IF;
END;
/

-- Now, with the triggers in place, populate the tables

INSERT INTO DEPARTMENT VALUES
      ('1', 'Accounting', 0);

INSERT INTO DEPARTMENT VALUES
      ('2', 'Marketing', 0);

INSERT INTO EMPLOYEE VALUES
      ('990', 'Clark Kent', '1', 150000);

INSERT INTO EMPLOYEE VALUES
      ('454', 'Bruce Wayne', '1', 120000);

INSERT INTO EMPLOYEE VALUES
      ('197', 'Diana Prince', '1', 90000);

INSERT INTO EMPLOYEE VALUES
      ('660', 'Tony Stark', '2', 180000);

INSERT INTO EMPLOYEE VALUES
      ('823', 'Natasha Romanoff', '2', 125000);

INSERT INTO EMPLOYEE VALUES
      ('123', 'Steve Rogers', '2', '100000');


SELECT * FROM EMPLOYEE;

SELECT * FROM DEPARTMENT;

=====
```