Exercises for Chapter 14: Mutating Tables and Compound Triggers

The Labs below provide you with exercises and suggested answers with discussion related to how those answers resulted. The most important thing to realize is whether your answer works. You should figure out the implications of the answers here and what the effects are from any different answers you may come up with.

Lab 14.1 Mutating Tables

Answer the following questions:

Mutating Table

a) What is a mutating table?

Answer: A table having A DML statement issues against it is called mutating table. For a trigger, it is the table on which this trigger is defined.

b) What causes a mutating table error?

Answer: If a trigger tries to read or modify a table on which it is defined, it causes a mutating table error. For example, if trigger is defined on the `STUDENT` table and it tries to read from it as well, it will cause a mutating table error. Note that the trigger in this case must be a row level trigger.

c) Is it possible to detect a mutating table error at the time of trigger compilation?

Answer: No, as a mutating table error is a runtime error. The trigger will compile successfully and will cause a mutating table error at the time when it fires.

Resolving Mutating Table Issues

In this exercise, you modify a trigger that causes a mutating table error when an `INSERT` statement is issued against the ENROLLMENT table. Create the following trigger:

For Example *ch14_5a.sql*

```
CREATE OR REPLACE TRIGGER enrollment_biu
BEFORE INSERT OR UPDATE ON enrollment
FOR EACH ROW
DECLARE
   v_total NUMBER;
   v_name  VARCHAR2(30);
BEGIN
   SELECT COUNT(*)
     INTO v_total
     FROM enrollment
    WHERE student_id = :NEW.student_id;

   -- check if the current student is enrolled into too
   -- many courses
   IF v_total >= 3
   THEN
      SELECT first_name||' '||last_name
        INTO v_name
        FROM student
       WHERE student_id = :NEW.STUDENT_ID;

      RAISE_APPLICATION_ERROR
         (-20000, 'Student, '||v_name||', is registered for 3 courses already');
   END IF;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      RAISE_APPLICATION_ERROR (-20001, 'This is not a valid student');
END;
```

Issue the following `INSERT` and `UPDATE` statements:

```
INSERT INTO enrollment
   (student_id, section_id, enroll_date, created_by, created_date, modified_by
   ,modified_date)
VALUES (184, 98, SYSDATE, USER, SYSDATE, USER, SYSDATE);

INSERT INTO enrollment
   (student_id, section_id, enroll_date, created_by, created_date, modified_by
   ,modified_date)
VALUES (399, 98, SYSDATE, USER, SYSDATE, USER, SYSDATE);

UPDATE enrollment
   SET student_id = 399
 WHERE student_id = 283;
```

Answer the following questions:

a) What output is produced after the `INSERT` and `UPDATE` statements are issued?

Answer: Once the trigger is created, the `INSERT` and `UPDATE` statements issued against the `ENROLLMENT` table produce the following output:

```
INSERT INTO ENROLLMENT
   (student_id, section_id, enroll_date, created_by, created_date, modified_by
```

```
   ,modified_date)
VALUES (184, 98, SYSDATE, USER, SYSDATE, USER, SYSDATE);

ORA-20000: Student, Salewa Zuckerberg, is registered for 3 courses already
ORA-06512: at "STUDENT.ENROLLMENT_BIU", line 19
ORA-04088: error during execution of trigger 'STUDENT.ENROLLMENT_BIU'

INSERT INTO ENROLLMENT
   (student_id, section_id, enroll_date, created_by, created_date, modified_by
   ,modified_date)
VALUES (399, 98, SYSDATE, USER, SYSDATE, USER, SYSDATE);

1 rows inserted.

UPDATE enrollment
   SET student_id = 399
WHERE student_id = 283;

ORA-04091: table STUDENT.ENROLLMENT is mutating, trigger/function may not see it
ORA-06512: at "STUDENT.ENROLLMENT_BIU", line 5
ORA-04088: error during execution of trigger 'STUDENT.ENROLLMENT_BIU'
```

b) Explain why two of the statements did not succeed.

Answer: The `INSERT` statement does not succeed because it tries to create a record in the `ENROLLMENT` table for a student that is already registered for three courses.

The IF statement

```
IF v_total >= 3
THEN
   SELECT first_name||' '||last_name
     INTO v_name
     FROM student
    WHERE student_id = :NEW.STUDENT_ID;

   RAISE_APPLICATION_ERROR
      (-20000, 'Student, '||v_name||', is registered for 3 courses already');
END IF;
```

in the body of the trigger evaluates to TRUE, and as a result the `RAISE_APPLICATION_ERROR` statement raises a user-defined exception.

The `UPDATE` statement does not succeed, because a trigger tries to read data from the mutating table. The `SELECT INTO` statement

```
SELECT COUNT(*)
  INTO v_total
  FROM enrollment
 WHERE student_id = :NEW.STUDENT_ID;
```

is issued against the `ENROLLMENT` table that is being modified and therefore is mutating.

c) Modify the trigger so that it does not cause a mutating table error when an `UPDATE` statement is issued against the `ENROLLMENT` table.

Answer: First, create a package to hold the student's ID and name as follows:

```
CREATE OR REPLACE PACKAGE student_pkg
AS
   g_student_id   student.student_id%TYPE;
   g_student_name varchar2(50);
END;
```

Next, modify the existing trigger, ENROLLMENT_BIU as follows:

For Example *ch14_5b.sql*

```
CREATE OR REPLACE TRIGGER enrollment_biu
BEFORE INSERT OR UPDATE ON enrollment
FOR EACH ROW
BEGIN
   IF :NEW.student_id IS NOT NULL
   THEN
      BEGIN
    student_pkg.g_student_id := :NEW.student_id;

         SELECT first_name||' '||last_name
           INTO student_pkg.g_student_name
           FROM student
          WHERE student_id = student_pkg.g_student_id;
      EXCEPTION
         WHEN NO_DATA_FOUND
         THEN
            RAISE_APPLICATION_ERROR (-20001, 'This is not a valid student');
      END;
   END IF;
END;
```

Finally, create a new statement-level trigger on the ENROLLMENT table as follows:

For Example *ch14_6a.sql*

```
CREATE OR REPLACE TRIGGER enrollment_aiu
AFTER INSERT OR UPDATE ON enrollment
DECLARE
   v_total INTEGER;
BEGIN
   SELECT COUNT(*)
     INTO v_total
     FROM enrollment
    WHERE student_id = student_pkg.g_student_id;

   -- check if the current student is enrolled into too
   -- many courses
   IF v_total >= 3
   THEN
      RAISE_APPLICATION_ERROR
         (-20000, 'Student, '||student_pkg.g_student_name||
          ', is registered for 3 courses already ');
   END IF;
```

```
END;
```

Once the package and two triggers are created, the `UPDATE` statement does not cause a mutating table error. However, the `UPDATE` statement

```
UPDATE enrollment
   SET student_id = 399
WHERE student_id = 283;
```

causes different kind of error:

```
ORA-02292: integrity constraint (STUDENT.GR_ENR_FK) violated - child record found
```

Note that this error does not relate to the trigger implementation, rather it is based on the foreign key constraint defined between the `GRADE` and `ENROLLMENT` tables.

Lab 14.2 Compound Triggers

Answer the following questions:

Compound Trigger

a) What is a compound trigger?

Answer: A compound trigger allows you to combine different types of triggers into one trigger.

b) What types of triggers may be combined into a compound trigger?

Answer: You may combine the following triggers into a compound trigger:

- Statement trigger that fires before the firing statement
- Row Trigger that fires before each row that the firing statement affects
- Row Trigger that fires after each row that the firing statement affects
- Statement trigger that fires after the firing statement

c) What are some of the restrictions on the compound triggers?

Answer: Some of the restrictions on the compound triggers are:

- A compound trigger may be defined on a table or a view only.
- A triggering event of a compound trigger is limited to the DML statements.
- A compound trigger may not contain an autonomous transaction. In other words, its declaration portion cannot include `PRAGMA AUTOTONOMOUS_TRANSACTION`.
- An exception that occurs in one executable section must be handled within that section. For example, if an exception occurs in the `AFTER EACH ROW` section it cannot propagate to the `AFTER STATEMENT` section. It must be handled in the `AFTER EACH ROW` section.

Resolving Mutating Table Issues with Compound Triggers

In this exercise, you modify trigger created in exercise section of Lab 14.1 that causes a mutating table error when an `UPDATE` statement is issued against the `ENROLLMENT` table.

Before starting this exercise it is suggested that you drop triggers and package created in the exercise section of Lab 14.1 and deleted records added and/or updated in the ENROLLMENT table as follows:

```
DROP TRIGGER enrollment_biu;
DROP TRIGGER enrollment_aiu;
DROP PACKAGE student_pkg;

DELETE FROM enrollment
 WHERE student_id = 399;
COMMIT;
```

Recall ENROLLMENT_BIU trigger created in the previous Lab:

For Example *ch14_5a.sql*

```
CREATE OR REPLACE TRIGGER enrollment_biu
BEFORE INSERT OR UPDATE ON enrollment
FOR EACH ROW
DECLARE
   v_total NUMBER;
   v_name  VARCHAR2(30);
BEGIN
   SELECT COUNT(*)
     INTO v_total
     FROM enrollment
    WHERE student_id = :NEW.student_id;

   -- check if the current student is enrolled into too
   -- many courses
   IF v_total >= 3
   THEN
      SELECT first_name||' '||last_name
        INTO v_name
        FROM student
       WHERE student_id = :NEW.STUDENT_ID;

      RAISE_APPLICATION_ERROR
         (-20000, 'Student, '||v_name||', is registered for 3 courses already');
   END IF;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      RAISE_APPLICATION_ERROR (-20001, 'This is not a valid student');
END;
```

Recall the following INSERT and UPDATE statements and the errors they produced:

```
INSERT INTO ENROLLMENT
   (student_id, section_id, enroll_date, created_by, created_date, modified_by
   ,modified_date)
VALUES (184, 98, SYSDATE, USER, SYSDATE, USER, SYSDATE);

ORA-20000: Student, Salewa Zuckerberg, is registered for 3 courses already
ORA-06512: at "STUDENT.ENROLLMENT_BIU", line 19
```

```
ORA-04088: error during execution of trigger 'STUDENT.ENROLLMENT_BIU'

INSERT INTO ENROLLMENT
   (student_id, section_id, enroll_date, created_by, created_date, modified_by
   ,modified_date)
VALUES (399, 98, SYSDATE, USER, SYSDATE, USER, SYSDATE);

1 rows inserted.

UPDATE enrollment
   SET student_id = 399
WHERE student_id = 283;

ORA-04091: table STUDENT.ENROLLMENT is mutating, trigger/function may not see it
ORA-06512: at "STUDENT.ENROLLMENT_BIU", line 5
ORA-04088: error during execution of trigger 'STUDENT.ENROLLMENT_BIU'
```

Answer the following questions:

a) Create a new compound trigger so that it does not cause a mutating table error when an `UPDATE` statement is issued against the `ENROLLMENT` table.

Answer: The newly created compound trigger should look similar to the following:

For Example *ch14_7a.sql*

```
CREATE OR REPLACE TRIGGER enrollment_compound
FOR INSERT OR UPDATE ON enrollment
COMPOUND TRIGGER
   v_student_id   STUDENT.STUDENT_ID%TYPE;
   v_student_name VARCHAR2(50);
   v_total        INTEGER;

BEFORE EACH ROW IS
BEGIN
   IF :NEW.student_id IS NOT NULL
   THEN
      BEGIN
         v_student_id := :NEW.student_id;

         SELECT first_name||' '||last_name
           INTO v_student_name
           FROM student
          WHERE student_id = v_student_id;
      EXCEPTION
         WHEN NO_DATA_FOUND
         THEN
            RAISE_APPLICATION_ERROR (-20001, 'This is not a valid student');
      END;
   END IF;
END BEFORE EACH ROW;

AFTER STATEMENT IS
BEGIN
   SELECT COUNT(*)
```

```
      INTO v_total
      FROM enrollment
     WHERE student_id = v_student_id;

   -- check if the current student is enrolled into too
   -- many courses
   IF v_total >= 3
   THEN
      RAISE_APPLICATION_ERROR
         (-20000, 'Student, '||v_student_name||
          ', is registered for 3 courses already ');
   END IF;
END AFTER STATEMENT;

END enrollment_compound;
```

In the trigger created above, you declare variables to record student ID and name that were previously declared in the package `STUDENT_PKG`. You also declared variable `v_total` which was previously declared in the `ENROLLMENT_AIU` trigger. Next, you create `BEFORE EACH ROW` and `AFTER STATEMENT` sections in the body of the trigger. Note that the statements in those sections are copies from executable sections of the `ENROLLMENT_BIU` and `ENROLLMENT_AIU` triggers respectively.

b) Run the `UPDATE` statement listed in the exercise text again. Explain the output produced.

Answer: The output should look as follows:

```
UPDATE enrollment
   SET student_id = 399
WHERE student_id = 283;


ORA-02292: integrity constraint (STUDENT.GR_ENR_FK) violated - child record found
```

Note that the error generated by the `UPDATE` statement is not a mutating table error. This error refers to the integrity constraint violation because there is a child record in the `GRADE` table with student ID of 283.

c) Modify compound trigger so that values for the `CREATED_BY`, `CREATED_DATE`, `MODIFIED_BY`, `MODIFIED_DATE` columns are populated by the trigger.

Answer: The new version of the trigger should look similar to the following. Changes are highlighted in bold.

For Example *ch14_7b.sql*

```
CREATE OR REPLACE TRIGGER enrollment_compound
FOR INSERT OR UPDATE ON enrollment
COMPOUND TRIGGER
   v_student_id   STUDENT.STUDENT_ID%TYPE;
   v_student_name VARCHAR2(50);
   v_total        INTEGER;
   v_date         DATE;
```

```
   v_user         STUDENT.CREATED_BY%TYPE;

BEFORE STATEMENT IS
BEGIN
   v_date := SYSDATE;
   v_user := USER;
END BEFORE STATEMENT;

BEFORE EACH ROW IS
BEGIN
   IF INSERTING
   THEN
      :NEW.created_date := v_date;
      :NEW.created_by   := v_user;
   ELSIF UPDATING
   THEN
      :NEW.created_date := :OLD.created_date;
      :NEW.created_by   := :OLD.created_by;
   END IF;
   :NEW.MODIFIED_DATE := v_date;
   :NEW.MODIFIED_BY   := v_user;

   IF :NEW.STUDENT_ID IS NOT NULL
   THEN
      BEGIN
         v_student_id := :NEW.STUDENT_ID;

         SELECT first_name||' '||last_name
           INTO v_student_name
           FROM student
          WHERE student_id = v_student_id;
      EXCEPTION
         WHEN NO_DATA_FOUND
         THEN
            RAISE_APPLICATION_ERROR (-20001, 'This is not a valid student');
      END;
   END IF;
END BEFORE EACH ROW;

AFTER STATEMENT IS
BEGIN
   SELECT COUNT(*)
     INTO v_total
     FROM enrollment
    WHERE student_id = v_student_id;

   -- check if the current student is enrolled into too
   -- many courses
   IF v_total >= 3 THEN
      RAISE_APPLICATION_ERROR
         (-20000, 'Student, '||v_student_name||
          ', is registered for 3 courses already ');
   END IF;
END AFTER STATEMENT;
```

```
END enrollment_compound;
```

In this version of the trigger, you defined two new variables `v_date` and `v_user` in the declaration section of the trigger. You added a `BEFORE STATEMENT` section to initialize these variables. You also modified `BEFORE EACH ROW` section where you now initialize `CREATED_BY`, `CREATED_DATE`, `MODIFIED_BY`, and `MODIFIED_DATE` columns. Note that the `ELSIF` statement

```
IF INSERTING THEN
   :NEW.CREATED_DATE := v_date;
   :NEW.CREATED_BY   := v_user;
ESLIF UPDATING THEN
   :NEW.created_date := :OLD.created_date;
   :NEW.created_by   := :OLD.created_by;
END IF;
```

checks whether the current operation is an `INSERT` or `UPDATE` in order to determine how to populate the `CREATED_DATE` and `CREATED_BY` columns. For the `INSERT` operation, these columns are assigned values based on the `v_date` and `v_user` variables. For the `UPDATE` operation `CREATED_BY` and `CREATED_DATE` columns do not change their values, and as a result, the values are copied from the `OLD` pseudorecord. Since `MODIFIED_BY` and `MODIFIED_DATE` columns are always populated with the new values, there is no need to evaluate whether current record is being inserted or updated.

This version of the trigger may be tested as follows. Note that in this case the values of student and section IDs have been changed to allow trigger to execute successfully:

```
INSERT INTO enrollment
   (student_id, section_id, enroll_date, final_grade)
VALUES (102, 155, sysdate, null);

ORA-20000: Student, Fred Crocitto, is registered for 3 courses already
ORA-06512: at "STUDENT.ENROLLMENT_COMPOUND", line 55
ORA-04088: error during execution of trigger 'STUDENT.ENROLLMENT_COMPOUND'

INSERT INTO enrollment
   (student_id, section_id, enroll_date, final_grade)
VALUES (103, 155, sysdate, null);

1 rows inserted.

UPDATE enrollment
   SET final_grade = 85
 WHERE student_id = 105
   AND section_id = 155;

1 rows updated.

ROLLBACK;

rollback complete.
```

It is important to note that when the `CREATED_DATE` and `CREATED_BY` columns are not initialized to any values in the body of the trigger for the `UPDATE` operation, the trigger causes

NOT NULL constraint violation. In other words, the CREATED_DATE and CREATED_BY columns should be reinitialized to their original values explicitly in the BEFORE EACH ROW section of the trigger. Consider modified version of the BEFORE EACH ROW section that causes NOT NULL constraint violation error for the UPDATE operation. In this code fragment, the ELSIF portion of the IF statement has been omitted (modified portion is highlighted in bold):

```
BEFORE EACH ROW IS
BEGIN
   IF INSERTING THEN
      :NEW.created_date := v_date;
      :NEW.created_by   := v_user;
   END IF;
   :NEW.modified_date := v_date;
   :NEW.modified_by   := v_user;

    IF :NEW.STUDENT_ID IS NOT NULL THEN
       BEGIN
          v_student_id := :NEW.student_id;

          SELECT first_name||' '||last_name
            INTO v_student_name
            FROM student
           WHERE student_id = v_student_id;
       EXCEPTION
          WHEN NO_DATA_FOUND THEN
             RAISE_APPLICATION_ERROR
               (-20001, 'This is not a valid student');
       END;
    END IF;
END BEFORE EACH ROW;
```

This version of the trigger causes the following error when an UPDATE is issued against the ENROLLMENT table:

```
ORA-01407: cannot update ("STUDENT"."ENROLLMENT"."CREATED_DATE") to NULL
```

Try It Yourself

The projects in this section are meant to have you use all of the skills that you have acquired throughout this chapter. Here are some exercises that will help you test the depth of your understanding.

1) Create a compound trigger on the INSTRUCTOR table that fires on the INSERT and UPDATE statements. The trigger should not allow insert on the INSTRUCTOR table during off hours where off hours are times of day outside the 9:00 am–5:00 pm window and weekends. The trigger should also populate INSTRUCTOR_ID, CREATED_BY, CREATED_DATE, MODIFIED_BY, MODIFIED_DATE columns with their default values.

   Answer: The trigger should look similar to the following:

   For Example *ch14_8a.sql*

```
CREATE OR REPLACE TRIGGER instructor_compound
FOR INSERT OR UPDATE ON instructor
COMPOUND TRIGGER
```

```
   v_date DATE;
   v_user VARCHAR2(30);

BEFORE STATEMENT IS
BEGIN
   IF RTRIM(TO_CHAR(SYSDATE, 'DAY')) NOT LIKE 'S%' AND
      RTRIM(TO_CHAR(SYSDATE, 'HH24:MI')) BETWEEN '09:00' AND '17:00'
   THEN
      v_date := SYSDATE;
      v_user := USER;
   ELSE
      RAISE_APPLICATION_ERROR
         (-20000, 'A table cannot be modified during off hours');
   END IF;

END BEFORE STATEMENT;

BEFORE EACH ROW IS
BEGIN
   IF INSERTING
   THEN
      :NEW.instructor_id := INSTRUCTOR_ID_SEQ.NEXTVAL;
      :NEW.created_by    := v_user;
      :NEW.created_date  := v_date;

   ELSIF UPDATING
   THEN
      :NEW.created_by    := :OLD.created_by;
      :NEW.created_date  := :OLD.created_date;
   END IF;

   :NEW.modified_by   := v_user;
   :NEW.modified_date := v_date;

END BEFORE EACH ROW;

END instructor_compound;
```

The compound trigger created above has two executable sections, `BEFORE STATEMENT` and `BEFORE EACH ROW`. The `BEFORE STATEMENT` portion prevents any updates to the `INSTRUCTOR` table during off hours. In addition, it populates `v_date` and `v_user` variables that are used to populate the `CREATED_BY`, `CREATED_DATE`, `MODIFIED_BY`, `MODIFIED_BY` columns. The `BEFORE EACH ROW` section populates the above specified columns. In addition, it assigns value to the `INSTRUCTOR_ID` column from the `INSTRUCTOR_ID_SEQ`.

Note the use of the `INSERTING` and `UPDATING` functions in the `BEFORE EACH ROW` section. The `INSERTING` function is used because `INSTRUCTOR_ID`, `CREATED_BY`, and `CREATED_DATE` columns are populated with new values only if record is being inserted in the `INSTRUCTOR` table. This is not so when a record is being updated. In this case, the `CREATE_BY` and `CREATED_DATE` columns are populated with the values copied from the

OLD pseudorecord. However, MODIFIED_BY and MODIFIED_DATE columns need to be populated with the new values regardless of the INSERT or UPDATE operation.

The newly created trigger may be tested as follows:

```
DECLARE
   v_date VARCHAR2(20);
BEGIN
   v_date := TO_CHAR(SYSDATE, 'DD/MM/YYYY HH24:MI');
   DBMS_OUTPUT.PUT_LINE ('Date: '||v_date);

   INSERT INTO instructor
      (salutation, first_name, last_name, street_address, zip, phone)
   VALUES
      ('Mr.', 'Test', 'Instructor', '123 Main Street', '07112',
       '2125555555');

   ROLLBACK;
END;

Date: 17/11/2014 11:10

DECLARE
   v_date VARCHAR2(20);
BEGIN
   v_date := TO_CHAR(SYSDATE, 'DD/MM/YYYY HH24:MI');
   DBMS_OUTPUT.PUT_LINE ('Date: '||v_date);

   UPDATE instructor
      SET phone = '2125555555'
    WHERE instructor_id = 101;

   ROLLBACK;
END;

Date: 16/11/2014 11:50

ORA-20000: A table cannot be modified during off hours
ORA-06512: at "STUDENT.INSTRUCTOR_COMPOUND", line 17
ORA-04088: error during execution of trigger 'STUDENT.INSTRUCTOR_COMPOUND'
ORA-06512: at line 7
```

2) Create a compound trigger on the ZIPCODE table that fires on the INSERT and UPDATE statements. The trigger should populate MODIFIED_BY, MODIFIED_DATE columns with their default values. In addition it should record in the STATISTICS table type of the transaction, name of the user who issued the transaction, date of the transaction, and number of records affected by the transaction. Assume the STATISTICS table has the following structure:

```
Name                           Null?    Type
------------------------------ -------- ----
TABLE_NAME                              VARCHAR2(30)
TRANSACTION_NAME                        VARCHAR2(10)
TRANSACTION_USER                        VARCHAR2(30)
TRANSACTION_DATE                        DATE
```

Answer: The trigger should look similar to the following:

For Example *ch14_9a.sql*

```
CREATE OR REPLACE TRIGGER zipcode_compound
FOR INSERT OR UPDATE ON zipcode
COMPOUND TRIGGER

   v_date DATE;
   v_user VARCHAR2(30);
   v_type VARCHAR2(10);

BEFORE STATEMENT IS
BEGIN
   v_date := SYSDATE;
   v_user := USER;
END BEFORE STATEMENT;

BEFORE EACH ROW IS
BEGIN
   IF INSERTING
   THEN
      :NEW.created_by   := v_user;
      :NEW.created_date := v_date;

   ELSIF UPDATING
   THEN
      :NEW.created_by   := :OLD.created_by;
      :NEW.created_date := :OLD.created_date;
   END IF;

   :NEW.modified_by   := v_user;
   :NEW.modified_date := v_date;

END BEFORE EACH ROW;

AFTER STATEMENT IS
BEGIN
   IF INSERTING
   THEN
      v_type := 'INSERT';

   ELSIF UPDATING
   THEN
      v_type := 'UPDATE';
   END IF;

   INSERT INTO statistics
      (table_name, transaction_name, transaction_user, transaction_date)
   VALUES ('ZIPCODE', v_type, v_user, v_date);

END AFTER STATEMENT;
```

```
END zipcode_compound;
```

This trigger may be tested as follows:

```
UPDATE zipcode
   SET city = 'Test City'
 WHERE zip  = '01247';

1 rows updated.

SELECT *
  FROM statistics
 WHERE transaction_date >= TRUNC(sysdate);

TABLE_NAME TRANSACTION_NAME TRANSACTION_USER TRANSACTION_DATE
---------- ---------------- ---------------- ----------------
ZIPCODE    UPDATE           STUDENT          11/17/2014 11:28

ROLLBACK;

rollback complete.
```