Database Programming

Triggers – Part II

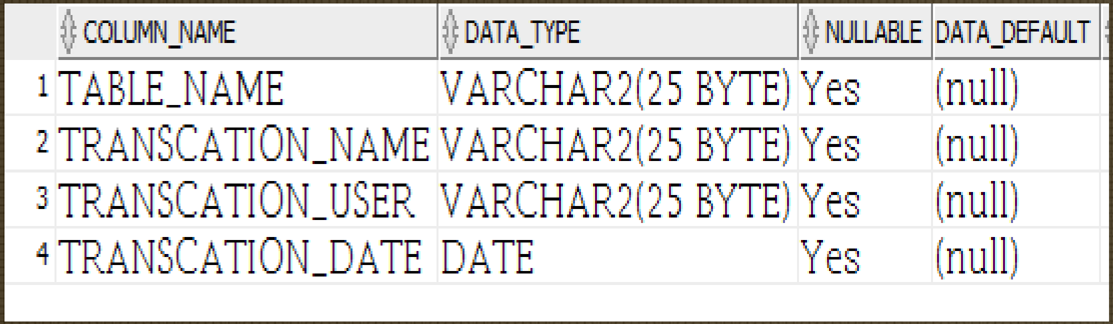
Sagara Samarawickrama

CSD 4204 - CPCM GP 1,2 & 3

Triggers

AFTER Triggers

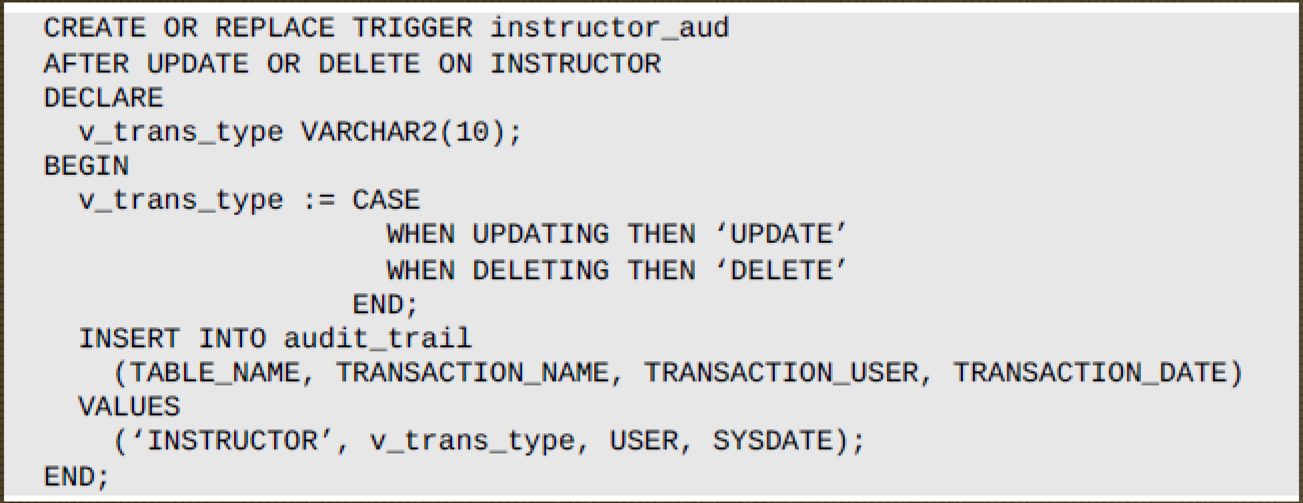
Assume there is a table called AUDIT_TRAIL having the structure shown in table. This table is used to collect user access information on different tables in the STUDENT schema. For example, you can record who deleted records from the INSTRUCTOR table and when they were deleted.



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT
1	TABLE_NAME	VARCHAR2(25 BYTE)	Yes	(null)
2	TRANSCATION_NAME	VARCHAR2(25 BYTE)	Yes	(null)
3	TRANSCATION_USER	VARCHAR2(25 BYTE)	Yes	(null)
4	TRANSCATION_DATE	DATE	Yes	(null)

To accomplish this, you would need to create a trigger on the INSTRUCTOR table, as shown in the following example.

Triggers



```
CREATE OR REPLACE TRIGGER instructor_aud
AFTER UPDATE OR DELETE ON INSTRUCTOR
DECLARE
  v_trans_type VARCHAR2(10);
BEGIN
  v_trans_type := CASE
                    WHEN UPDATING THEN ‘UPDATE’
                    WHEN DELETING THEN ‘DELETE’
                  END;
  INSERT INTO audit_trail
    (TABLE_NAME, TRANSACTION_NAME, TRANSACTION_USER, TRANSACTION_DATE)
  VALUES
    (‘INSTRUCTOR’, v_trans_type, USER, SYSDATE);
END;
```

This trigger fires after an UPDATE or DELETE statement is issued on the INSTRUCTOR table. The body of the trigger contains two Boolean functions, UPDATING and DELETING. The function UPDATING evaluates to TRUE if an UPDATE statement is issued on the table, and the function DELETING evaluates to TRUE if a DELETE statement is issued on the table.

Triggers

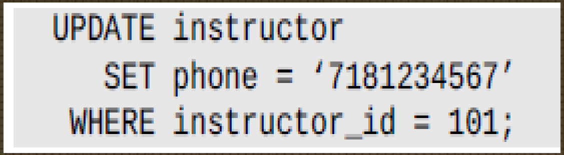
This trigger inserts a record into the AUDIT_TRAIL table when an UPDATE or DELETE operation is issued against the INSTRUCTOR table. First, it determines which operation was issued against the INSTRUCTOR table via the CASE statement. The result of this evaluation is then assigned to the v_trans_type variable. Next, the trigger adds a new record to the AUDIT_TRAIL table.

Once this trigger is created on the INSTRUCTOR table, any UPDATE or DELETE operation causes the creation of new records in the AUDIT_TRAIL table. Furthermore, this trigger may be enhanced by calculating how many rows were updated or deleted from
the INSTRUCTOR table.

Triggers

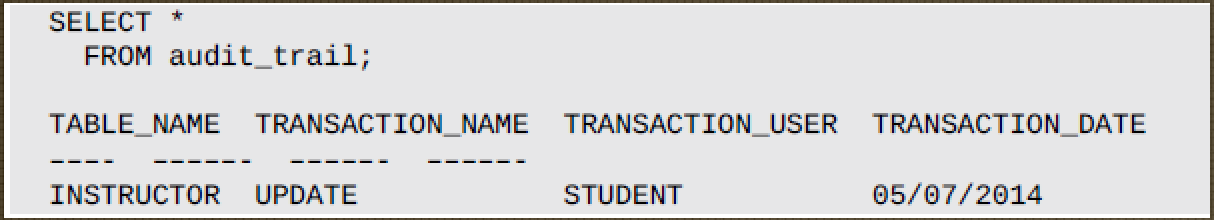
Autonomous Transcation

As stated previously, when a trigger fires, all operations performed by the trigger become part of a transaction. When this transaction is committed or rolled back, the operations performed by the trigger are committed or rolled back as well. Consider an UPDATE statement against the INSTRUCTOR table as shown below



```
UPDATE instructor
   SET phone = ‘7181234567’
 WHERE instructor_id = 101;
```

When this UPDATE statement is executed, the INSTRUCTOR_AUD trigger fires and adds a single record to the AUDIT_TRAIL table as shown below



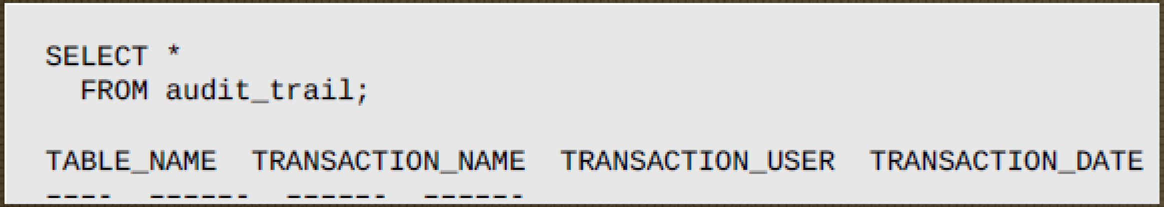
```
SELECT *
  FROM audit_trail;

TABLE_NAME  TRANSACTION_NAME  TRANSACTION_USER  TRANSACTION_DATE
----  ------  ------  ------
INSTRUCTOR  UPDATE            STUDENT           05/07/2014
```

Triggers

Next, consider rolling back the UPDATE statement just issued. In this case, the record inserted in the AUDIT_TRAIL table is rolled back as well, as shown below

ROLLBACK;

Then



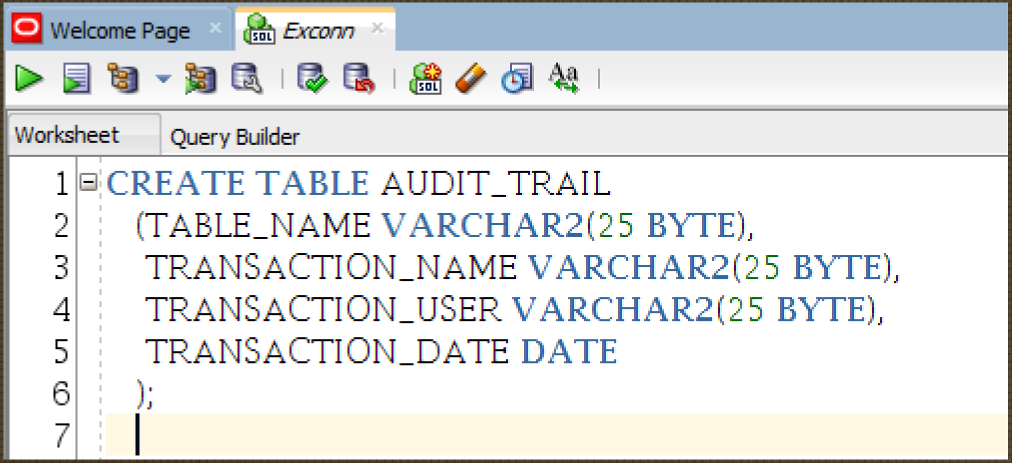
```
SELECT *
  FROM audit_trail;

TABLE_NAME  TRANSACTION_NAME  TRANSACTION_USER  TRANSACTION_DATE
----  ------  ------  ------
```

As you can see, the AUDIT_TRAIL table no longer contains any records. To circumvent such behavior, you may choose to employ autonomous transactions. An autonomous transaction is an independent transaction started by another transaction that is usually referred to as the main transaction. In other words, an autonomous transaction may issue various DML statements and commit or roll them back, without committing or rolling back the DML statements issued by the main transaction.

Triggers

Next, we will see an example.

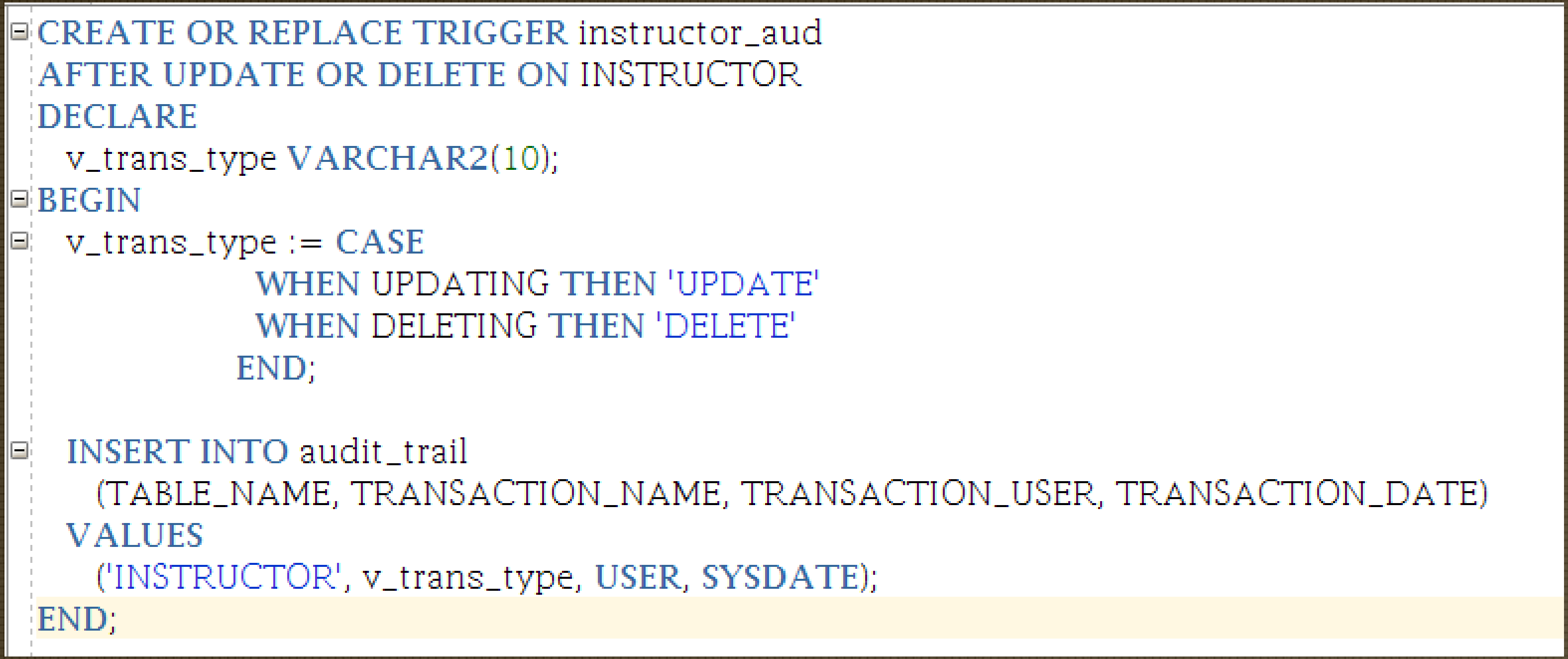
First create the following table



```
CREATE TABLE AUDIT_TRAIL
  (TABLE_NAME VARCHAR2(25 BYTE),
   TRANSACTION_NAME VARCHAR2(25 BYTE),
   TRANSACTION_USER VARCHAR2(25 BYTE),
   TRANSACTION_DATE DATE
  );
```

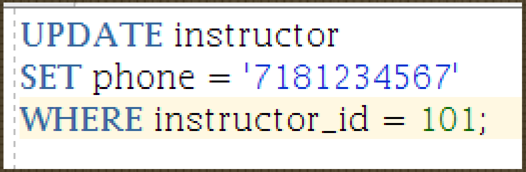
Now we will create a AFTER trigger to activate when user try to update and delete records

Triggers



```
CREATE OR REPLACE TRIGGER instructor_aud
AFTER UPDATE OR DELETE ON INSTRUCTOR
DECLARE
  v_trans_type VARCHAR2(10);
BEGIN
  v_trans_type := CASE
                    WHEN UPDATING THEN 'UPDATE'
                    WHEN DELETING THEN 'DELETE'
                  END;

  INSERT INTO audit_trail
    (TABLE_NAME, TRANSACTION_NAME, TRANSACTION_USER, TRANSACTION_DATE)
  VALUES
    ('INSTRUCTOR', v_trans_type, USER, SYSDATE);
END;
```

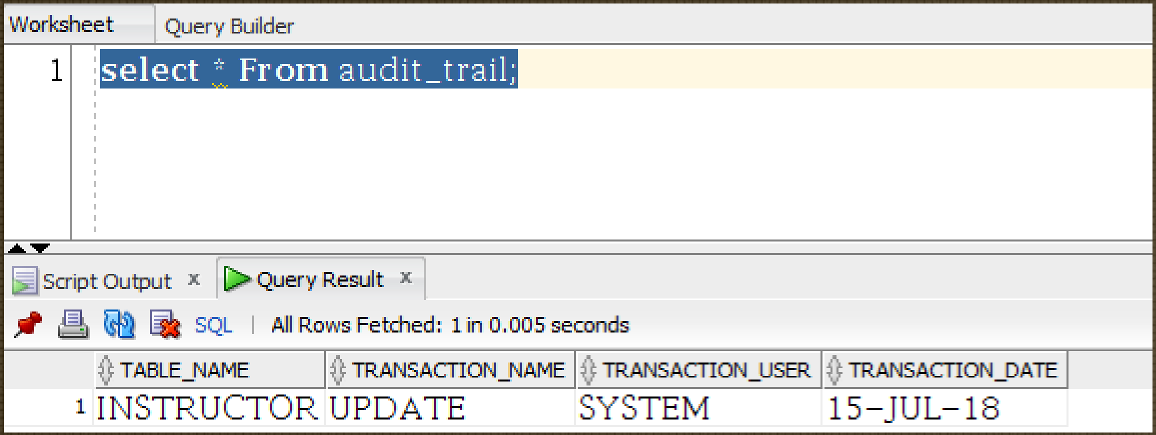
Now issue the following update statement



```
UPDATE instructor
SET phone = '7181234567'
WHERE instructor_id = 101;
```

Triggers

Now we can check the audit trail table

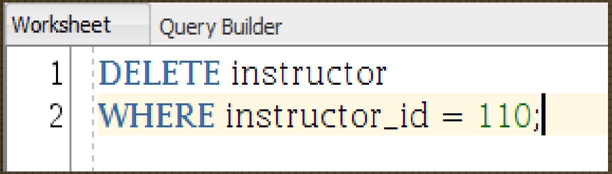


```
select * From audit_trail;
```

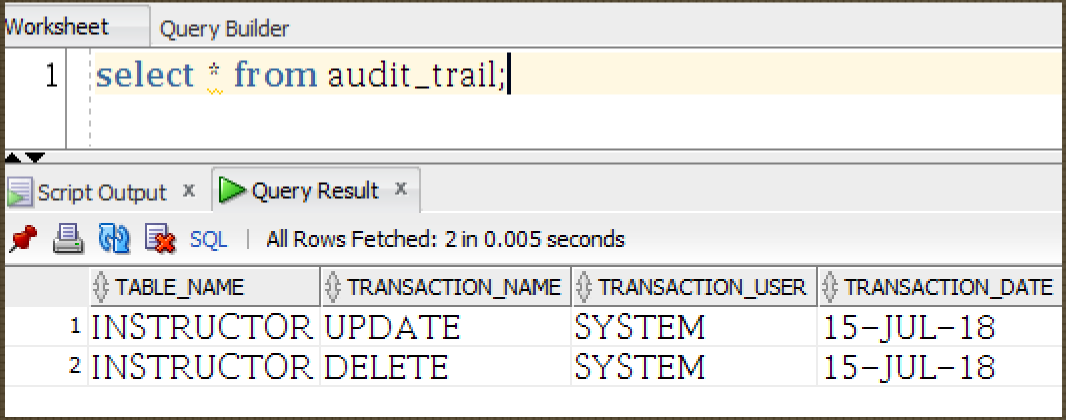
All Rows Fetched: 1 in 0.005 seconds

	TABLE_NAME	TRANSACTION_NAME	TRANSACTION_USER	TRANSACTION_DATE
1	INSTRUCTOR	UPDATE	SYSTEM	15-JUL-18

Now issue the following sql statement to delete the instructor with id 110



```
DELETE instructor
WHERE instructor_id = 110;
```



```
select * from audit_trail;
```

All Rows Fetched: 2 in 0.005 seconds

	TABLE_NAME	TRANSACTION_NAME	TRANSACTION_USER	TRANSACTION_DATE
1	INSTRUCTOR	UPDATE	SYSTEM	15-JUL-18
2	INSTRUCTOR	DELETE	SYSTEM	15-JUL-18

Q
&
A