# **In-Class Exercise: Triggers**

## **AFTER** Trigger

**1. The AFTER trigger can be very useful to form audit trails.** Generally, we, as business analysts, accountants, DBAs, etc., want to be able to keep track of changes being made to our important tables in our database. Run the following script and then try to figure out exactly what is going on. First, you are going to create the table that will be used to track the changes made to the invoices table. Then, you will run the trigger.

--Create audit table

CREATE TABLE invoices\_audit

(

vendor\_id NUMBER NOT NULL,

invoice\_number VARCHAR2(50) NOT NULL,

invoice\_total NUMBER NOT NULL,

action\_type VARCHAR2(50) NOT NULL,

action\_date DATE NOT NULL

);

--Create audit trigger for invoice table changes

CREATE OR REPLACE TRIGGER invoices\_after\_dml

AFTER INSERT OR UPDATE OR DELETE

ON invoices

FOR EACH ROW

BEGIN

IF INSERTING THEN

INSERT INTO invoices\_audit VALUES

(:new.vendor\_id, :new.invoice\_number,

:new.invoice\_total, 'INSERTED', SYSDATE);

ELSIF UPDATING THEN

INSERT INTO invoices\_audit VALUES

(:old.vendor\_id, :old.invoice\_number,

:old.invoice\_total, 'UPDATED', SYSDATE);

ELSIF DELETING THEN

INSERT INTO invoices\_audit VALUES

(:old.vendor\_id, :old.invoice\_number,

:old.invoice\_total, 'DELETED', SYSDATE);

END IF;

END;

/

1. **Now run the following insert and delete statements and check out your invoices\_audit table to see how well this trigger works!**

--An INSERT statement that fires the trigger

INSERT INTO invoices VALUES

(115, 34, 'ZXA-080', '30-AUG-14', 14092.59, 0, 0, 3,

'30-SEP-14', NULL);

--A statement that retrieves the audit table rows

SELECT \* FROM invoices\_audit;

--A DELETE statement that fires the trigger

DELETE FROM invoices WHERE invoice\_number = 'ZXA-080';

--A statement that retrieves the audit table rows

SELECT \* FROM invoices\_audit;

1. **If you get done and feel comfortable, check what happens when you update invoice\_due\_date for all invoices to one month later after the invoice\_due\_date .**

update invoices set payment\_date = payment\_date + 30;

**Use the previous elect statement to check your invoices\_audit table.**

--Once you’re done, here’s how you can drop trigger and table

drop table invoices\_audit;

drop trigger invoices\_after\_dml;

## **BEFORE Trigger**

1. **Run the following script:**

--Create trigger to enforce data consistency

CREATE OR REPLACE TRIGGER vendors\_before\_update\_state

BEFORE INSERT OR UPDATE OF vendor\_state

ON vendors

FOR EACH ROW

WHEN (NEW.vendor\_state != UPPER(NEW.vendor\_state))

BEGIN

:NEW.vendor\_state := UPPER(:NEW.vendor\_state);

END;

/

* 1. **Run this update statement for the vendors table that updates the vendor\_state to ‘tx’ for vendor\_id = 1.**

--An UPDATE statement that fires the trigger

UPDATE Vendors

Set vendor\_state = 'tx'

where vendor\_id = 1;

* 1. **Run the following select statement to see the update that you made. Do you see the difference between what you inserted and what the vendor\_state is listed as? Try it again and change state to ‘az’ in lowercase and see if the trigger allows this. This is a where triggers can be used for data validation and integrity.**

--A SELECT statement that shows the new row

SELECT vendor\_name, vendor\_state

FROM vendors

WHERE vendor\_id = 1;

-- Drop trigger when you’re done

drop trigger vendors\_before\_update\_state;

1. **The trigger below has been written to check the format of an inserted or updated phone number.**

--Create trigger to enforce data consistency for phone number

CREATE OR REPLACE TRIGGER vendor\_before\_insert\_phone

BEFORE INSERT OR UPDATE OF vendor\_phone

ON vendors

FOR EACH ROW

WHEN (length(NEW.vendor\_phone) != 14)

BEGIN

RAISE\_APPLICATION\_ERROR(-20001,

'Insert phone number in the format (###) ###-####.');

END;

/

* 1. **Try to update a phone number for vendor 1 and use the format “888-999-4422”. See whether the trigger works.**

update vendors set vendor\_phone = '888-999-4422' where vendor\_id = 1;

* 1. **Now try to insert a phone number with the proper format: (888) 999-4422**

update vendors set vendor\_phone = '(888) 999-4422' where vendor\_id = 1;

* 1. **Is there a different (actually better) way to ensure/enforce proper phone number format? The answer is yes, so think on this and be ready to tell me what it is!**

**Advanced (if you are done with everything else – Make sure you go check out the AFTER Trigger first!):**

1. **Just for fun, try to figure out a way to use a trigger to put phone numbers in the correct format. See the trigger code below and try to figure out how it works!**

CREATE OR REPLACE TRIGGER vendor\_before\_insert\_phone

BEFORE INSERT OR UPDATE OF vendor\_phone

ON vendors

FOR EACH ROW

WHEN (length(NEW.vendor\_phone) != 14)

BEGIN

IF length(:NEW.vendor\_phone) = 10 THEN

:NEW.vendor\_phone :=

'('||substr(:NEW.vendor\_phone,1,3) ||') '||

SUBSTR(:NEW.vendor\_phone,4,3)||'-'||SUBSTR(:NEW.vendor\_phone,7,4);

ELSIF length(:NEW.vendor\_phone) = 12 THEN

:NEW.vendor\_phone :=

'('||substr(:NEW.vendor\_phone,1,3) ||') '||

SUBSTR(:NEW.vendor\_phone,5,8);

END IF;

END;

/

1. **Use the different update statements to check whether the trigger works**

UPDATE vendors

SET vendor\_phone = '677-123-7456'

WHERE vendor\_id = 1;

UPDATE vendors

SET vendor\_phone = '6771237456'

WHERE vendor\_id = 2;

UPDATE vendors

SET vendor\_phone = '(677) 777-7777'

WHERE vendor\_id = 3;

select vendor\_id, vendor\_phone

FROM vendors;

**Super cool!**