Rungta International Journal of Computer Science and Information Technology
3 (1&2), 2018, 9-14
http://www.publishingindia.com/ijcsit

Triggering Security Features of Oracle

Ankur N. Shah
Assistant Professor, CE Department, Babaria Institute of Technology, Vadodara, Gujarat, India.
Email : ankur11586@gmail.com

Abstract: Database trigger is a PL/SQL block that is executed on an event in the database. The event is related to a particular data manipulation of a table such as inserting, deleting or updating a row of a table [8]. Triggers are automatically fired when condition written for trigger is matched. Thus it provides security to database by avoiding harmful operations on database. The triggers are also cascading, so trigger can be called by another trigger. There are also various types of triggers available. Even there are various advantages of using trigger, the excessive use of triggers can result in complex interdependences, which may be difficult to maintain in a large application. So it advisable to used trigger when they are really needed.

Keywords: Procedure, Security, Trigger.

Nomenclature:

SQL - Structure Query Language

PL/SQL - Procedural Language / Structural Query Language

DML - Data Manipulation Language

I. Introduction to Trigger

Database triggers are procedures that are stored in the database and are implicitly executed (fired) when the contents of a table are changed. Triggers are executed when an insert, update or delete is issued against a table from SQL or through an application. The major point that make these triggers stand alone is that they are fired implicitly (i.e. internally) by Oracle itself and not explicitly called by the user, as done in normal procedures [9].

For example, Fig. 1 shows a database application with some SQL statements that implicitly fire several triggers stored in the database [12].

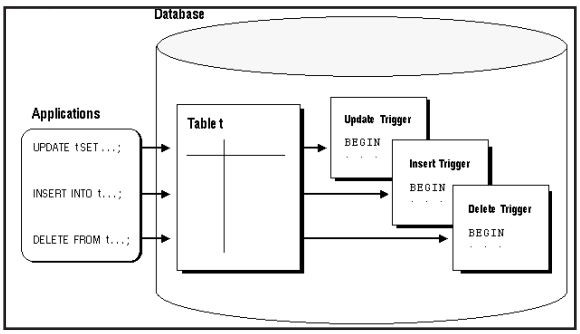

Fig. 1: Triggers

Triggers are stored in the database separately from their associated tables. Triggers can be defined only on tables, not on views. However, triggers on the base table(s) of a view are fired if an INSERT, UPDATE, or DELETE statement is issued against a view [12].

II. How Triggers are Used

In many cases, triggers supplement the standard capabilities of Oracle to provide a highly customized database management system. For example, a trigger can permit DML operations against a table only if they are issued during regular business hours. The standard security features of Oracle, roles and privileges, govern which users can submit DML statements against the table. In addition, the trigger further restricts DML operations to occur only at certain times during weekdays. This is just one way that you can use triggers to customize information management in an Oracle database [12].

In addition it can be used to: automatically generate derived column values, prevent invalid transactions, enforce complex security authorizations, enforce referential integrity across nodes in a distributed database, enforce complex business rules, provide transparent event logging, provide sophisticated auditing, maintain synchronous table replicates, gather statistics on table access etc. [12].

III. Benefits of Trigger

- A trigger can permit DML statements against a table only if they are issued during regular business hours or on predetermined weekdays.
- A trigger can also be used to keep an audit trail of a table (i.e. store the modified and deleted records of the table) along with the operation performed and the time on which the operation was performed.
- It can be used to prevent invalid transactions.
- Enforce complex security authorizations [9].

IV. Safety Measures of Trigger

- When a trigger is fired, a SQL statement inside the trigger can also fired the same or some other trigger, called cascading, which must be considered [9].
- Unnecessary use of triggers for customizing the database can result in complex interdependencies between the triggers, which may be difficult to maintain in a large applications [9].

V. Difference between Procedure and Trigger

- We can execute a stored procedure whenever we want with the help of the exec command, but a trigger can only be executed whenever an event (insert, delete, and update) is fired on the table on which the trigger is defined.
- Stored procedures are used for performing tasks. Stored procedures are normally used for performing user specified tasks. They can have parameters and return multiple results sets. While the Triggers for auditing work: Triggers normally are used for auditing work. They can be used to trace the activities of table events.

VI. Parts of Trigger

- *A Triggering Event or Statement* - It is a SQL statement that causes a trigger to be fired. It can insert, update or delete statement for a specific table. A triggering statement can also specify multiple DML statements.
- *A Trigger Restriction* - A trigger restriction specifies a Boolean expression that must be TRUE for the trigger to fire. It is an option available for triggers that are fired for each row. Its function is to conditionally control the execution of a trigger. A trigger restriction is specified using a WHEN clause.
- *A Trigger Action* - A trigger action is the procedure that contains the SQL statements and PL/SQL code to be executed when a triggering statement is issued and the trigger restriction evaluated to TRUE. It can contain SQL and PL/SQL statements; can define PL/SQL language constructs and can call stored procedures. Additionally, for row triggers, the statements in a trigger action have access to column values (new and old) of the current row being processed.

VII. Syntax of Trigger

Create or replace trigger trigger-name {before, after}{delete, insert, update} on table-name

[referencing {OLD as old, NEW as new}]

[for each row / statement [when condition]]

declare

variable declaration;

constant declaration;

begin

pl/sql subprogram body;

exception

exception pl/sql block;

end;[9]

Following Fig. 2 is useful to understand create trigger statement [13].

OR REPLACE recreates the trigger if it already exists. You can use this option to change the definition of an existing trigger without dropping it.

Trigger-name is the name of the trigger to be created.

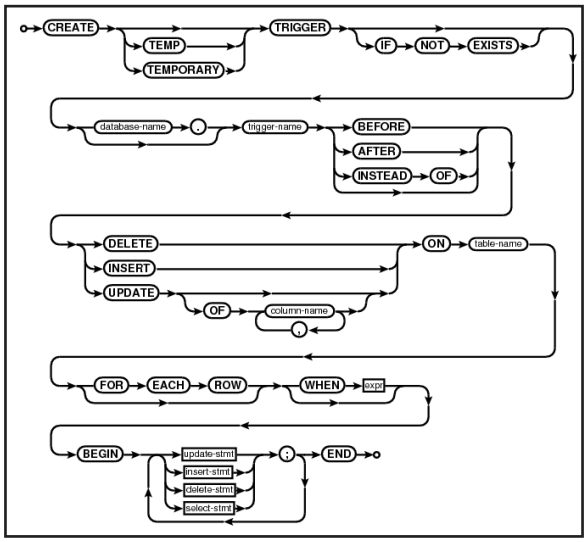

Fig. 2: Create Trigger Statement

VIII. Keywords and Parameters

BEFORE indicates that Oracle fires the trigger before executing the triggering statement.

AFTER indicates that Oracle fires the trigger after executing the triggering statement.

DELETE indicates that Oracle fires the trigger whenever a DELETE statement removes a row from the table.

INSERT indicates that Oracle fires the trigger whenever an INSERT statement adds a row to the table.

UPDATE indicates that Oracle fires the trigger whenever an UPDATE statement changes a value in any column of the table.

ON Specifies the name of the table on which the trigger is to be created.

REFERENCING specifies correlation names. You can use correlation names in the PL/SQL block and WHEN clause of a row triggers to refer specifically to old and new values of the current row. The default correlation names are OLD and NEW. If your row trigger is associated with a table named *OLD or NEW*, you can use this clause to specify different correlation names to avoid confusion between table name and the correlation name.

FOR EACH ROW designates the trigger to be a row trigger. Oracle fires a row trigger once for each row that is affected by the triggering statement and meets the optional trigger constraint defined in the when clause. If you omit this clause, the trigger is a statement trigger.

WHEN Specifies the trigger restriction. The trigger restriction contains a SQL condition that must be satisfied for Oracle to fire the trigger. This condition must contain correlation names and cannot contain a query. You can specify trigger restriction

only for the row triggers. Oracle evaluates these conditions for each row affected by the triggering statement.

PL/SQL block is the PL/SQL block that oracle executes to fire the trigger [9].

IX. Types of Triggers

- *Row Triggers and Statement Triggers* - When you define a trigger, you can specify the number of times the trigger action is to be run: Once for every row affected by the triggering statement, such as a trigger fired by an UPDATE statement that updates many rows. Once for the triggering statement, no matter how many rows it affects.
- *Row Triggers* - A row trigger is fired each time the table is affected by the triggering statement. For example, if an UPDATE statement updates multiple rows of a table, a row trigger is fired once for each row affected by the UPDATE statement. If a triggering statement affects no rows, a row trigger is not run. Row triggers are useful if the code in the trigger action depends on data provided by the triggering statement or rows that are affected.
- *Statement Triggers* - A statement trigger is fired once on behalf of the triggering statement, regardless of the number of rows in the table that the triggering statement affects, even if no rows are affected. For example, if a DELETE statement deletes several rows from a table, a statement-level DELETE trigger is fired only once. Statement triggers are useful if the code in the trigger action does not depend on the data provided by the triggering statement or the rows affected. For example, use a statement trigger to: Make a complex security check on the current time or user. Generate a single audit record. *BEFORE and AFTER Triggers* - When defining a trigger, you can specify the trigger timing - whether the trigger action is to be run before or after the triggering statement. BEFORE and AFTER apply to both statement and row triggers. BEFORE and AFTER triggers fired by DML statements can be defined only on tables, not on views. However, triggers on the base tables of a view are fired if an INSERT, UPDATE, or DELETE statement is issued against the view. BEFORE and AFTER triggers fired by DDL statements can be defined only on the database or a schema, not on particular tables.
- *BEFORE Triggers* - BEFORE triggers run the trigger action before the triggering statement is run. This type of trigger is commonly used in the following situations: When the trigger action determines whether the triggering statement should be allowed to complete. Using a BEFORE trigger for this purpose, you can eliminate unnecessary processing of the triggering statement and its eventual rollback in cases where an exception is raised in the trigger action. To derive specific column values before completing a triggering INSERT or UPDATE statement.
- *AFTER Triggers* - AFTER triggers run the trigger action after the triggering statement is run.
- *INSTEAD OF Triggers* - INSTEAD OF triggers provide a transparent way of modifying views that cannot be modified directly through DML statements (INSERT, UPDATE, and DELETE). These triggers are called INSTEAD OF triggers because, unlike other types of triggers, Oracle fires the trigger instead of executing the triggering statement.

X. Trigger Type Combinations

We can also combine trigger type like, before statement trigger, before row trigger etc.

XI. Cascading Trigger

When trigger is fired, SQL statements within a trigger have ability to fire another trigger. It is shown in Fig. 3. When a statement in a trigger body causes another trigger to be fired, the triggers are said to be cascading [12].

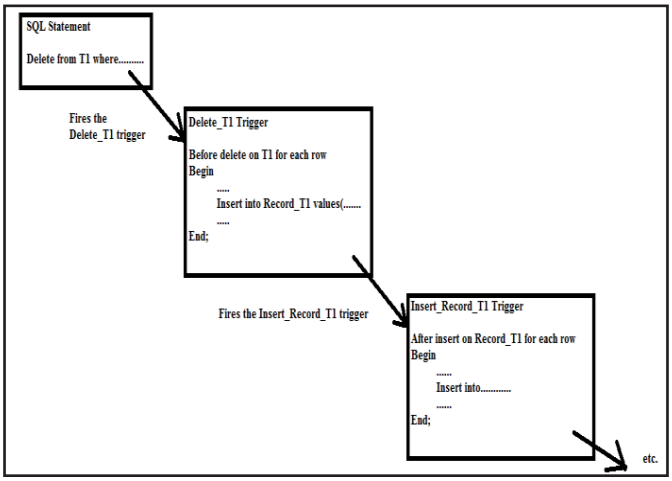

Fig. 3: Cascading Triggers

While triggers are useful for customizing a database, you should only use triggers when necessary. The excessive use of triggers can result in complex interdependences, which may be difficult to maintain in a large application [12].

XII. Examples of Trigger

First we create sailor table as shown below.

create table sailor(sid number(3), sname varchar2(30), rating number(2), age number(3));

Now we insert few records in sailor table.

insert into sailor values(1,'veer', 5, 20);

insert into sailor values(2,'vinay', 7, 22);

insert into sailor values(3,'raja', 15, 18);

Now if we look sailor table it is look as below.

select * from sailor;

SID	SNAME	RATING	AGE
1	Veer	5	20
2	Vinay	7	22
3	Raja	15	18

Now we create one another table sailor_history as shown below.

```
create table sailor_history(sid number(3), sname varchar2(30));
```

Now we create one trigger known as sailor_delete which used to keep history of sailors those are deleted from sailor table. We keep these records into sailor_history table.

```
Create or replace trigger sailor_delete  after delete on sailor for each row
begin
        insert into sailor_history values(:OLD.SID, :OLD.SNAME);
        dbms_output.put_line('Record deleted successfully from sailor table and kept in sailor_history table successfully');
end;
```

Now we delete record from sailor table as shown below.

```
delete from sailor where sid=1;
```

Due to trigger sailor_delete we get following output and two tables look as below.

Record deleted successfully from sailor table and kept in sailor_history table successfully

1 row(s) deleted.

Now,

select * from sailor; gives us,

SID	SNAME	RATING	AGE
2	Vinay	7	22
3	Raja	15	18

Now,

select * from sailor_history; gives us,

SID	SNAME
1	Veer

Following are some other examples of triggers.

- Write a database trigger before insert for each row on the table route_detail not allowing transaction on Saturday and Sundays.

Table: route_detail (rid, from, to, shipping_date)

```
create or replace trigger sun_trig before insert or update or delete on route_detail.
declare
        shipping_date char;
begin
        shipping_date := to_char(sysdate, 'dy');
        if shipping_date in ('sat', 'sun') then
                raise_application_error(-20001, 'try on any weekdays');
        end if;
end;
```

- Write a trigger that is fired before the DML statement's execution on the employee table. The trigger checks the day based on sysdate. If the day is Sunday, the trigger does not allow the DML statement's execution and raises an exception. Write the appropriate message in the exception handling section.

Tables: employees (employeeid, lname, fname, positioned, supervisor, hiredate, salary, commission, deptid, qualid)

dept (deptid, deptname, location, employeeid)

```
create or replace trigger day_chk before insert or update or delete or select on employee
declare
                msg exception;
begin
        if to_char (sysdate,'day') = 'sunday' or
          to_char (sysdate,'hh:mm a.m.') < '8:00 a.m.' or
          to_char (sysdate,'hh:mm p.m.') >= '5:00 p.m.'
        then
                raise msg;
        end if;
exception
        when msg then
                dbms_output.put_line (''changes to employee table allowed only during business hours');
end;
```

- Write a database before / insert or update / row trigger to check distance between the origin and destination would not be greater than 2000 km.

Tables: route_header (route_id number (5), route_no number (5), destination varchar2 (20), fare number(5,2))

```
create or replace trigger dis_chk before insert or update on route_header for each row
declare
        msg exception;
        d route_header.distance % type;
        rid route_header.route_id % type;
begin
        select distance into d from route_header where route_id=&rid;
        if d > 2000then
                raise msg;
        end if;
```

```
exception
        when msg then
        dbms_output.put_line ('you cannot insert or update
        values for distance which is greater than 2000 km.');
end;
```

XIII. Execution Sequence of Database Triggers

The following Fig. 4 shows the execution sequence of database triggers.

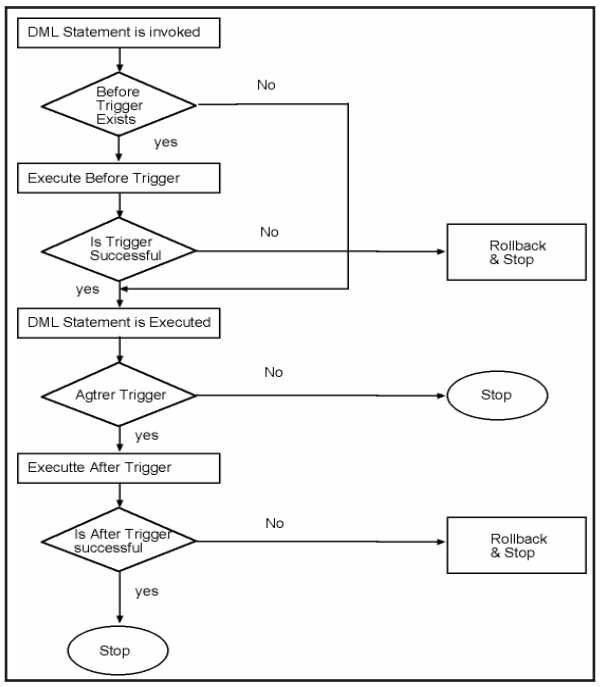

Fig. 4: Execution Sequence for Database Triggers

XIV. Enabling and Disabling Triggers

A database trigger is in either enable state or disable state. Disabling trigger may improve performance when large amount of table data is to be modified because trigger code is not executed [8].

For example, Update emp set name = upper(name);

Run faster if all triggers fired by update on emp table are disabled. Using alter command we can enable or disable trigger, as shown below,

```
alter trigger sun_trig disable;
```

This command will disable sun_trig trigger.

Use following command to enable any trigger.

```
alter trigger sun_trig enable;
```

Following command will disable all triggers on emp table.

```
alter table emp disable all triggers;
```

XV. Remove Triggers

When trigger is no longer needed it can be removed using drop trigger command as shown below [8].

```
drop trigger sun_trig;
```

XVI. View Information about Triggers

Following example shows how to view information about trigger.

```
SELECT Trigger_type, Triggering_event, Table_name FROM
USER_TRIGGERS WHERE Trigger_name = 'PART';
```

TRIGGER_TYPE	TRIGGERING_EVENT	TABLE_NAME
AFTER STATEMENT	INSERT	PART

```
SELECT Trigger_body FROM USER_TRIGGERS WHERE
Trigger_name= 'PART';
```

Result:

```
TRIGGER_BODY
----------------------------------------------------
BEGIN
        DBMS_OUTPUT.PUT_LINE('The insert in part
        table is successful ');
END;
```

1 row selected [11].

XVII. Conclusion

Database triggers are used to implement complex business rules that cannot be implemented using integrity constraints. Triggers can also be used for logging and to take actions that should be automatically performed. Trigger may be fired before the DML operation or after DML operation. Trigger may be fired for each row affected by the DML operation or for the entire statement. INSTEAD-OF trigger are called instead of executing the given DML command. They are defined on relational-views and object-views. A trigger can be disabled to increase the performance when a lot of data manipulations are to be done on the table. Trigger is also used to provide security as condition is violated then it is automatically fired.

References

[1] A. Behrend, C. Dorau, and R. Manthey, "SQL triggers reacting on time events: An extension proposal," in J. Grundspenkis, T. Morzy, and G. Vossen, (eds), Advances in Databases and Information Systems (ADBIS 2009),

Lecture Notes in Computer Science, vol. 5739, Springer, Berlin, Heidelberg, 2009.

[2] L. Pamulaparty, T. P. Kumar, and P. V. B. Varma, "A survey: Security perspectives of ORACLE and IBM-DB2 databases," *International Journal of Scientific and Research Publications*, vol. 3, no. 1, January 2013.

[3] D. Lee, W. Mao, H. Chiu, and W. W. Chu, "Designing triggers with trigger-by-example," *Knowledge and Information Systems*, vol. 7, no. 1, pp. 110-134, 2005.

[4] J. J. King, "Oracle 11g/10g for developers: What you need to know," Session S300195, King trainging resources, 2008.

[5] An Oracle white paper, "Oracle Database Security Checklist," June 2008.

[6] A. Kornbrust, "Best Practices for Oracle Databases Hardening Oracle 10.2.0.3 / 10.2.0.4," N. D.

[7] I. Bayross, *SQL, PL/SQL the Programming Language of Oracle*, N. D.

[8] Oracle for beginners, "Ch-20 Database Triggers," N. D. srikanthtechnologies.com

[9] A. N. Shah "RDBMS," Mahajan Publication, 2011.

[10] "Oracle," Atmiya Infotech Pvt Ltd.

[11] "PL/SQL Triggers,". Available: http://docs.oracle.com/cd/E11882_01/appdev.112/e25519/triggers.htm

[12] "Database Triggers - Oracle7 Server Concepts Manual,". Available: docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch15.htm?

[13] "Create Trigger,". Available: http://www.sqlite.org/lang_createtrigger.html