

4.1: Database Triggers

#### Concept of Database Triggers

#### Database Triggers:

- Database Triggers are procedures written in PL/SQL, Java, or C that run (fire) implicitly:
  - · whenever a table or view is modified, or
  - · when some user actions or database system actions occur
- They are stored subprograms.

#### <u>Database Triggers</u>:

A Trigger defines an action the database should take when some database related event occurs.

Anonymous blocks as well as stored subprograms need to be explicitly invoked by the user.

Database Triggers are PL/SQL blocks, which are implicitly fired by ORACLE RDBMS whenever an event such as INSERT, UPDATE, or DELETE takes place on a table.

Database triggers are also stored subprograms.

# Concept of Database Triggers



- You can write triggers that fire whenever one of the following operations occur:
  - · User events:
    - DML statements on a particular schema object
  - DDL statements issued within a schema or database
  - user logon or logoff events
  - System events:
    - server errors
    - database startup
    - instance shutdown

# **Usage of Triggers**

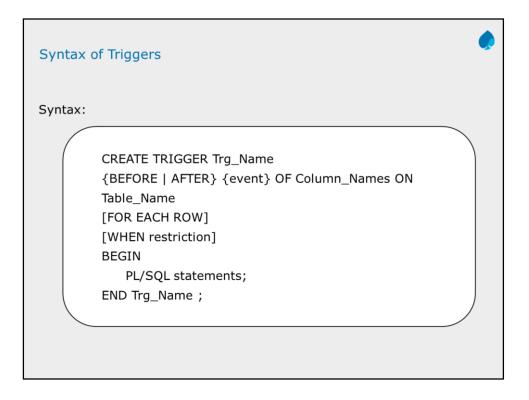
- Triggers can be used for:
- maintaining complex integrity constraints.
- auditing information, that is the Audit trail.
- automatically signaling other programs that action needs to take place when changes are made to a table.

## **Database Triggers (contd.)**:

Triggers can be used for:

- Maintaining complex integrity constraints. This is not possible through declarative constraints that are enabled at table creation.
- Auditing information in a table by recording the changes and the identify of the person who made them. This is called as an audit trail.
- Automatically signaling other programs that action needs to take place when changes are made to a table.

contd.



# **Database Triggers (contd.)**:

To create a trigger on a table you must be able to alter that table. The slide shows the syntax for creating a table.

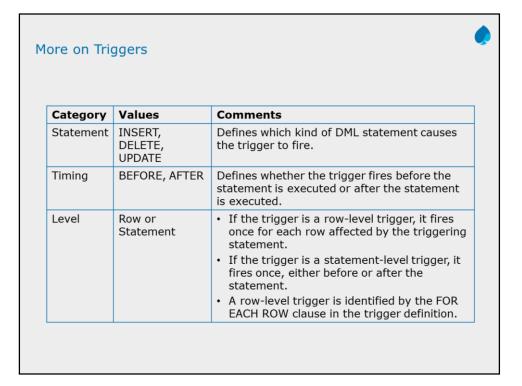
contd.

4.2Types of Triggers

## Types of Triggers

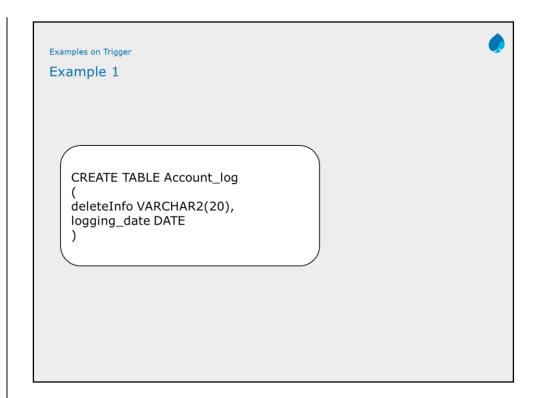
Type of Trigger is determined by the triggering event, namely:

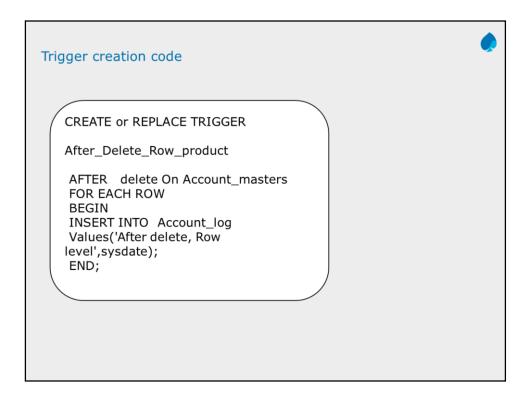
- INSERT
- UPDATE
- DELETE
- Triggers can be fired:
  - · before or after the operation.
  - · on row or statement operations.
- Trigger can be fired for more than one type of triggering statement.



#### Note:

- A trigger can be fired for more than one type of triggering statement. In case of multiple events, OR separates the events.
- From ORACLE 7.1 there is no limit on number of triggers you can write for a table.
  - Earlier you could write maximum of 12 triggers per table, one of each type.





4.4 Restrictions on Triggers

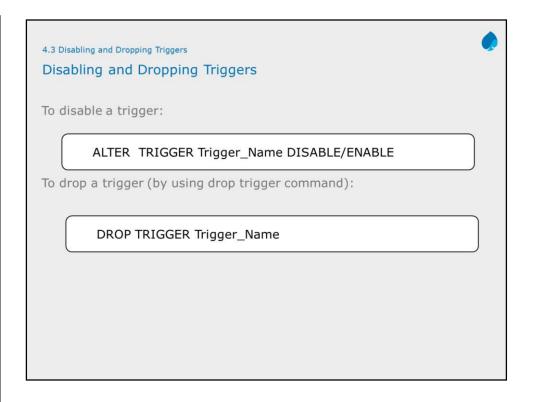
#### Restrictions on Triggers

The use of Triggers has the following restrictions:

- Triggers should not issue transaction control statements (TCL) like COMMIT, SAVEPOINT.
- Triggers cannot declare any long or long raw variables.
- :new and :old cannot refer to a LONG datatype.

#### Restrictions on Triggers:

- A Trigger may not issue any transaction control statements (TCL) like COMMIT, SAVEPOINT.
  - Since the triggering statement and the trigger are part of the same transaction, whenever triggering statement is committed or rolled back the work done in the trigger gets committed or rolled back, as well.
- Any procedures or functions that are called by the Trigger cannot have any transaction control statements.
- Trigger body cannot declare any long or long raw variables. Also :new and :old cannot refer to a LONG datatype.
- There are restrictions on the tables that a trigger body can access depending on type of "triggers" and "constraints" on the table.



## Note:

An UPDATE or DELETE statement affects multiple rows of a table. If the trigger is fired once for each affected row, then FOR EACH ROW clause is required. Such a trigger is called a ROW trigger. If the FOR EACH ROW clause is absent, then the trigger is fired only once for the entire statement. Such triggers are called STATEMENT triggers

4.5 Order of Trigger Firing

#### Order of Trigger Firing

## Order of Trigger firing is arranged as:

- Execute the "before statement level" trigger.
- For each row affected by the triggering statement:
- · Execute the "before row level" trigger.
- · Execute the statement.
- · Execute the "after row level" trigger.
- Execute the "after statement level" trigger.

#### Note:

- As each trigger is fired, it will see the changes made by the earlier triggers, as well as the database changes made by the statement.
- The order in which triggers of same type are fired is not defined.
- If the order is important, combine all the operations into one trigger.

4.6 Using :Old & :New values in Triggers

Using :Old & :New values in Triggers

Note: They are valid only within row level triggers and not in statement level triggers.

Triggering statement	:Old	:New
INSERT	Undefined – all fields are null.	Values that will be inserted when the statement is complete.
UPDATE	Original values for the row before the update.	New values that will be updated when the statement is complete.
DELETE	Original values before the row is deleted.	Undefined - all fields are NULL.

#### Using :old and :new values in Row Level Triggers:

- Row level trigger fires once per row that is being processed by the triggering statement.
- Inside the trigger, you can access the row that is currently being processed.
  This is done through keywords :new and :old (they are called as pseudo records). The meaning of the terms is as shown in the slide.

#### Note:

- The pseudo records are valid only within row level triggers and not in statement level triggers.
  - :old values are not available if the triggering statement is INSERT.
  - :new values are not available if the triggering statement is DELETE.
- Each column is referenced by using the notation :old.Column\_Name or :new.Column\_Name.
- If a column is not updated by the triggering update statement, then :old and :new values remain the same.

4.7 When clause

### Using WHEN clause

Use of WHEN clause is valid for row-level triggers only.

Trigger body is executed for rows that meet the specified condition.

## **Using WHEN Clause:**

- The WHEN clause is valid for row-level triggers only. If present, the trigger body will be executed only for those rows that meet the condition specified by the WHEN clause.
- The :new and :old records can be used here without colon.

contd.

```
CREATE TABLE Account_masters

(
account_no NUMBER(6) PRIMARY KEY,
cust_id NUMBER(6) ,
account_type CHAR(3) CONSTRAINT chk_acc_type
CHECK(account_type IN ('SAV','SAL')) ,
Ledger_balance NUMBER(10)
)
```

# Trigger creation code CREATE OR REPLACE TRIGGER trg\_acc\_master\_ledger before INSERT OR UPDATE OF Ledger\_balance ,account\_type ON Account\_masters FOR EACH ROW WHEN (NEW.account\_type = 'SAV') DECLARE v\_led\_bal NUMBER(10); BEGIN v\_led\_bal:=:NEW.Ledger\_balance; IF v\_led\_bal < 5000 THEN :NEW.Ledger\_balance := 5000; END if; END trg\_acc\_master\_ledger;

### Summary

Database Triggers are procedures written in PL/SQL, Java, or C that run (fire) implicitly: Database Triggers are procedures written in PL/SQL, Java, or C that run (fire) implicitly:



- whenever a table or view is modified, or
- when some user actions or database system actions occur

There are three types of triggers:

- Statement based triggers
- Timing based triggers
- Level based triggers

# Summary

Disabling and Dropping triggers can be done instead of actually removing the triggers

Order of trigger firing is decided depending on the type of triggers used in the sequence



Answers to Review Questions Question 1: True Question 2:False

# Review - Questions

Question 1: Triggers should not issue Transaction Control Statements (TCL).

True / False



Question 2: The :new and :old records must be used in WHEN clause with a colon.

Answers to Review Questions Question 3: Mutating Question 4: Statement level

# Review - Questions

Question 3: A \_\_\_\_ is a table that is currently being modified by a DML statement.

Question 4: A \_\_\_\_ is fired once on behalf of the triggering statement, regardless of the number of rows in the table that the triggering statement affects.

