# Oracle PL/SQL

# PL/SQL Comments

```
DECLARE
   v_salary number(9,2) := 40000;
BEGIN
   /* this is a multi-line comment that
      will be ignored by the pl/sql
      interpreter  */
   v_salary := v_salary * 2; -- nice raise
END; -- end of program
```

# Triggers

- PL/SQL code executed automatically in response to a database event, typically DML.
- Like other stored procedures, triggers are stored in the database.
- Often used to:
  - enforce complex constraints, especially multi-table constraints. Financial posting is an example of this.
  - Trigger related actions
  - implement auditing "logs"
  - pop a sequence when creating token keys
- Triggers do not issue transaction control statements (such as commit). Triggers are part of the SQL transaction that invoked them.
- USER_TRIGGERS provides a data dictionary view of triggers.

# TRIGGERS

- You can associate up to 12 database triggers with a given table. A database trigger has three parts: a **triggering event**, an **optional trigger constraint**, and a **trigger action**.

- When an event occurs, a database trigger is fired, and an predefined PL/SQL block will perform the necessary action.

# Triggers

- Associated with a particular table
- Automatically executed when a particular event occurs
  - Insert
  - Update
  - Delete
  - Others

# Triggers vs. Procedures

- Procedures are explicitly executed by a user or application
- Triggers are implicitly executed (fired) when the triggering event occurs
- Triggers should not be used as a lazy way to invoke a procedure as they are fired every time the event occurs

# Triggers

```
CREATE TRIGGER TriggerName
BEFORE [AFTER] event[s] ON TableName
[FOR EACH ROW]
DECLARE
   Declaration of any local variables
BEGIN
   Statements in Executable section
EXCEPTION
   Statements in optional Exception section
END;
/
```

# Triggers

- The trigger specification names the trigger and indicates when it will fire
- The trigger body contains the PL/SQL code to accomplish whatever task(s) needs to be performed

# TRIGGERS

**SYNTAX:**

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE|AFTER} triggering_event ON table_name
[FOR EACH ROW]
[WHEN condition]
DECLARE
Declaration statements
BEGIN
Executable statements
EXCEPTION
Exception-handling statements
END;
```

# Triggers Timing

- A triggers timing has to be specified first
  - Before (most common)
    - Trigger should be fired before the operation
      - i.e. before an insert
  - After
    - Trigger should be fired after the operation
      - i.e. after a delete is performed

# Trigger Events

- Three types of events are available
  - DML events
  - DDL events
  - Database events

# DML Events

- Changes to data in a table
  - Insert
  - Update
  - Delete

# DDL Events

- Changes to the definition of objects
  - Tables
  - Indexes
  - Procedures
  - Functions
  - Others
    - Include CREATE, ALTER and DROP statements on these objects

# Database Events

- Server Errors
- Users Log On or Off
- Database Started or Stopped

# Trigger DML Events

- Can specify one or more events in the specification
  - i.e. INSERT OR UPDATE OR DELETE
- Can specify one or more columns to be associated with a type of event
  - i.e. BEFORE UPDATE OF SID OR SNAME

# Table Name

- The next item in the trigger is the name of the table to be affected

# Trigger Level

- Two levels for Triggers
  - Row-level trigger
    - Requires FOR EACH ROW clause
      - If operation affects multiple rows, trigger fires once for each row affected
  - Statement-level trigger
    - A statement trigger, however, is fired only once for the triggering statement, regardless of the number of rows affected by the triggering statement
  - DML triggers should be row-level
  - DDL and Database triggers should not be row-level

# Event Examples

```
Example 1:
CREATE TRIGGER NameChange
BEFORE UPDATE OF STUDENT_FIRST_NAME, STUDENT_LAST_NAME ON STUDENT
FOR EACH ROW

Example 2:

CREATE TRIGGER AlterStudent
AFTER INSERT OR UPDATE OR DELETE ON STUDENT
FOR EACH ROW

Example 3:

CREATE TRIGGER ErrorLog
AFTER SERVERERROR ON DATABASE

Example 4:

CREATE Trigger TrackChanges
AFTER CREATE ON SCHEMA
```

# Triggers

- Conditions Available So Multiple Operations Can Be Dealt With In Same Trigger
  - Inserting, Updating, Deleting
- Column Prefixes Allow Identification Of Value Changes
  - New, Old

# TYPES OF TRIGGERS

## Example: statement trigger

```
CREATE OR REPLACE TRIGGER mytrig1 BEFORE DELETE OR
      INSERT OR UPDATE ON employee
BEGIN
IF (TO_CHAR(SYSDATE, 'day') IN ('sat', 'sun')) OR
      (TO_CHAR(SYSDATE,'hh:mi') NOT BETWEEN '08:30' AND
      '18:30') THEN          RAISE_APPLICATION_ERROR(-20500,
      'table is secured');
END IF;
END;
/
```

The above example shows a trigger that limits the DML actions to the employee table to weekdays from 8.30am to 6.30pm. If a user tries to insert/update/delete a row in the EMPLOYEE table, a warning message will be prompted.

# Example: ROW Trigger

```
CREATE OR REPLACE TRIGGER mytrig2
AFTER DELETE OR INSERT OR UPDATE ON employee
FOR EACH ROW
BEGIN
IF DELETING THEN
INSERT INTO xemployee (emp_ssn, emp_last_name,emp_first_name,
      deldate)
VALUES (:old.emp_ssn, :old.emp_last_name,:old.emp_first_name,  sysdate);
ELSIF INSERTING THEN
 INSERT INTO nemployee (emp_ssn, emp_last_name,emp_first_name,
      adddate)
VALUES (:new.emp_ssn, :new.emp_last_name,:new.emp_first_name,
      sysdate);
 ELSIF UPDATING('emp_salary') THEN
INSERT INTO cemployee (emp_ssn, oldsalary, newsalary, up_date)
VALUES (:old.emp_ssn,:old.emp_salary,  :new.emp_salary, sysdate);    ELSE
INSERT INTO uemployee (emp_ssn, emp_address, up_date)
VALUES (:old.emp_ssn, :new.emp_address, sysdate);
END IF;
END;
/
```

# TYPES OF TRIGGERS

## Example: ROW Trigger

- The previous trigger is used to keep track of all the transactions performed on the employee table. If any employee is deleted, a new row containing the details of this employee is stored in a table called xemployee. Similarly, if a new employee is inserted, a new row is created in another table called nemployee, and so on.

- Note that we can specify the old and new values of an updated row by prefixing the column names with the :OLD and :NEW qualifiers.

# TYPES OF TRIGGERS

```
SQL>  DELETE FROM  employee WHERE
      emp_last_name = 'Joshi';
1 row deleted.
SQL> SELECT * FROM xemployee;

EMP_SSN   EMP_LAST_NAME   EMP_FIRST_NAME DELDATE
---------- ---------------------  ------------------------- ----------------
999333333 Joshi                  Dinesh                    02-MAY-03
```

# ENABLING, DISABLING, DROPPING TRIGGERS

SQL>ALTER TRIGGER trigger_name DISABLE;

SQL>ALTER TABLE table_name DISABLE ALL TRIGGERS;

**To enable a trigger, which is disabled, we can use the following syntax:**

SQL>ALTER TABLE table_name ENABLE trigger_name;

**All triggers can be enabled for a specific table by using the following command**

SQL> ALTER TABLE table_name ENABLE ALL TRIGGERS;

SQL> DROP TRIGGER trigger_name

# Triggers Exceptions

- EXCEPTION Data Type Allows Custom Exceptions
- RAISE Allows An Exception To Be Manually Occur
- RAISE_APPLICATION_ERROR Allows Termination Using A Custom Error Message
  - Must Be Between -20000 and -20999
  - Message Can Be Up to 512 Bytes

# UpperCase Trigger Example

```
CREATE OR REPLACE TRIGGER UPPERCASE
   BEFORE INSERT OR UPDATE ON STUDENTS
   FOR EACH ROW
DECLARE
BEGIN
   :new.lastname:=UPPER(:new.lastname);
   :new.firstname:=UPPER(:new.firstname);
END UPPERCASE;
/
```

# Another Trigger Example

(no employee can make more than 10 times as much as the lowest paid employee)

```
CREATE OR REPLACE TRIGGER SalaryTrig
  BEFORE INSERT ON Employees
  FOR EACH ROW
DECLARE
  v_upper_sal_limit NUMBER(10,2);
  v_lower_sal_limit NUMBER(10,2);
BEGIN
  SELECT MIN(salary)*10 INTO v_upper_sal_limit
    FROM employees;
  SELECT MAX(salary)/10 INTO v_lower_sal_limit
    FROM employees;
  IF :new.salary NOT BETWEEN v_lower_sal_limit AND v_upper_sal_limit THEN
    RAISE_APPLICATION_ERROR(-20001,'salary out of allowed range');
  END IF;
END SalaryTrig;
/
```

Notes: Application error number is a parameter between -20,000 and -20,999.

You could also stop the insert by "poisoning" it, changing a :new buffer value to one that you know will not pass constraint evaluation.