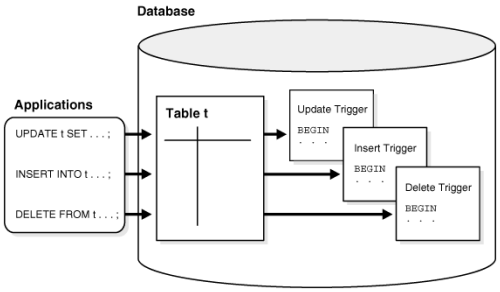
Database Triggers
Mila's Introduction

(MORE ON TRIGGEROMANIA)



Topics:

- **One for all?** Combining triggers (one trigger for multiple statements)
- **Stop that update!** Error conditions (stopping the modification inside the trigger)
- **Where did it go?** HELP! Where to find triggers in your database?
- **Who goes first?** Order of firing
- **How do tables mutate?** Some problems with selecting(updating) table from inside trigger for the same table ☹ Mutating tables problem (try to avoid if you can).

```
CREATE [OR REPLACE] TRIGGER trigger_name
        BEFORE | AFTER | INSTEAD OF triggering_event ON table_name
        [FOR EACH ROW | STATEMENT [WHEN trigger_condition]]
        trigger_body;
```

1. Automatically adding the next sequence value

Trigger to add next value for a sequence (sequence is used for ORDER_ID in ORDERS table).

```
CREATE OR REPLACE TRIGGER generate_ord_id_trigger
        BEFORE INSERT ON ORDERS
        FOR EACH ROW
BEGIN
        SELECT s_ord_id.NEXTVAL
          INTO :new.ORDER_ID
          FROM DUAL;
END;
```

From Oracle 11g the SELECT… FROM DUAL can be replaced by:

```
CREATE OR REPLACE TRIGGER generate_ord_id_trigger
        BEFORE INSERT ON ORDERS
        FOR EACH ROW
BEGIN

:new.ORDER_ID   :=  s_ord_id.NEXTVAL;

END;
```

2. Example for Audit using Triggers

```
CREATE TABLE Enrollment
  (
    S_Id      INTEGER,
    Course_Id INTEGER,
    Grade     CHAR(2),
    CONSTRAINT Enrollment_Pk PRIMARY KEY (S_Id,Course_Id)
  );
```

Create a Trigger that audits the ENROLLMENT table.

a. Create an audit table by typing in the following command

```
CREATE TABLE Enrollment_Audit
  (
    Student_Id  INTEGER NOT NULL,
    COURSE_ID   INTEGER NOT NULL,
    Action      VARCHAR2(40),
    Action_Date TIMESTAMP,
    User_Id     Varchar2(20)
  );
```

b. Create a trigger that fires after an update, insert and delete on the ENROLLMENT table

c. If the action is Update, insert the data and the string “Record Updated” in the action column.

d. If the action is Insert, insert the data and the string “Record Inserted” in the action column.

e. If the action is Delete, insert the data and the string “Record Deleted” in the action column.

```
CREATE TRIGGER ENROLMENT_AUDIT_UIDR
AFTER INSERT OR DELETE OR  UPDATE ON ENROLLOMENT
FOR EACH ROW
BEGIN
  IF INSERTING THEN
     _____________________________________________

   _____________________________________________;
 ELSIF  UPDATING THEN
  _____________________________________________;

 ELSE
  _____________________________________________;

 END IF;
END;
```

Explanation: Trigger predicates: INSERTING, DELETING, UPDATING allow to check the type of triggering statement inside the trigger.

```
CREATE OR REPLACE TRIGGER Enrollment_Uidr AFTER
  INSERT OR
  DELETE OR
  UPDATE ON Enrollment FOR EACH Row
BEGIN
IF Inserting THEN
  INSERT
  INTO Enrollment_Audit VALUES
 ( :New.S_Id, :New.Course_Id, 'inserting new row', Systimestamp, USER
);
ELSIF Updating THEN
  INSERT
  INTO Enrollment_Audit VALUES
    ( :New.S_Id, :New.Course_Id, 'updating row', Systimestamp, USER );
ELSE
  INSERT
  INTO Enrollment_Audit VALUES
    ( :old.S_Id, :old.Course_Id, 'deleting row', Systimestamp, USER );
END IF;
END;
```

3. Defining Business Constraints using Triggers (No updates on Saturdays and Sundays)

Let's write a trigger:

The table ENROLLMENT should not be updated on Sundays and Saturdays. We will write a trigger that stops updates to GRADE column on weekends.

```
CREATE TRIGGER enrollment_bef_upd
BEFORE UPDATE OF GRADE ON ENROLLMENT
DECLARE
  e_weekend_update EXCEPTION;
BEGIN
  IF TO_CHAR(SYSDATE, 'DY')  IN ('SAT','SUN') THEN
      RAISE e_weekend_update;
  END IF;
EXCEPTION
   WHEN e_weekend_update THEN
   RAISE_APLICCATION_ERROR (-20001, 'Grade updates not allowed on weekends');
END;
```

Explanation:

Note that this is a BEFORE statement level trigger. The RAISE_APPLICATION_ERROR procedure takes two parameters: the error number (must be between -20001 and -20999), and the error message to be displayed. This exception will stop the UPDATE statement from executing!

Where is my trigger? HELP!

Where to find triggers in your database.
So great, you have created a trigger or two … now how we can find them? Triggers are stored in the data dictionary (same as stored procedures or functions).

Data dictionary Views! Check USER_TRIGGERS (it has trigger body, WHEN clause, triggering table, and trigger type)

```
SELECT trigger_type, table_name, triggering_event
    FROM user_triggers;
```

I'm done with that trigger! You may have enough of triggers, then … You can **DROP** triggers (same as stored programs)

```
DROP TRIGGER triggername;
```

Less drastic measure is to just disable them. You can **disable** triggers by:

```
ALTER TRIGGER triggername {DISABLE| ENABLE};
```

Who goes first?

You can only imagine how the pure triggers feel just before firing --- so there is an order in this madness:

1. BEFORE statement-level triggers (yes, you can have more than one trigger), if present
2. For each row affected by the statement
   - BEFORE row-level trigger(s)
   - Statement itself
   - After row-level trigger(s)
3. AFTER statement-level trigger, if present

If you manage to create more than one trigger of the same type and level, the order of execution is not defined. SO… if the order is important, you should combine all the operations into one trigger (for that type and level). ☺

Mutating table problems

A ***mutating table*** is a table that is currently being modified by the DML statement. Inside the **row-level** trigger, you cannot read from or modify the mutating table. You can SELECT from mutating table from a statement -level trigger body. There is a way (as always) around the problem: when you have to select from inside the trigger --- we can create two triggers : statement (you can select from mutating table) and row-level. You can pass data between these triggers using a PL/SQL table (an array); however, this programatic table must be global (available for both triggers). … and that brings us to PACKAGES as a place to define global (scope of the package) variables. The mutating table error is famous (or infamous)
ORA-04091 and this will be your run-time error (not syntactical error). We will probably not create this error in our famous lab (you can try on your own) and we can discuss the solution in the advanced database class.