ADC UNIT 3

TRIGGERS

Triggers-A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

Component of trigger

1.Triggering SQL Statements-SQL DML(Insert Update & Delete) statement that execute and implicitly called trigger to execute.

2.Triggering Restriction-We can specify the condition inside the trigger to when the trigger is to be fired.

3.Trigger Action-When the triggering SQL statement is executed trigger automatically calls trigger action.

Types of trigger-

1.BEFORE TRIGGER-A BEFORE triggered action executes before the triggering statement , that is, before the occurrence of the trigger event.

2.AFTER TRIGGER-An AFTER triggered action executes after the action of the triggering statement is complete.

3.ROW-LEVEL TRIGGER-A row-level trigger fires once for each row that is affected by a triggering event.

4.STATEMENT-LEVEL TRIGGER-A statement-level trigger fires only once for each statement.

5.COMBINATION TRIGGER-

1. BEFORE STATEMENT TRIGGER-Trigger fire only once for each statement before the triggering DML statement.
2. BEFORE ROW TRIGGER-Trigger fire for each and every record before the trigger DML statement.
3. AFTER STATEMENT TRIGGER- Trigger fire only once for each statement after the triggering DML execution.
4. AFTER ROW TRIGGER-Trigger fire for each and every record after the triggering DML execution.

Create[or replace] trigger triggername

{BEFORE|AFTER|INSTEAD OF}

{INSERT[OR]|UPDATE[OR]|DELETE}

[of col_name]

ON table_name

[REFRENCING OLD as NEW as N]

[for each ROW]

when (condition)

Begin

- - sql statements

End;

PACKAGES

Packages are schema objects that groups logically related PL/SQL types, variables, and subprograms.

A package will have two mandatory parts –

- Package specification
- Package body or definition

Package Specification-The specification is the interface to the package. It just DECLARES the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package.

```
CREATE PACKAGE cust_sal AS
  PROCEDURE find_sal(c_id customers.id%type);
END cust_sal;
/
```

Package Body-The package body has the codes for various methods declared in the package specification and other private declarations, which are hidden from the code outside the package.

```
CREATE OR REPLACE PACKAGE BODY cust_sal AS
```

```
  PROCEDURE find_sal(c_id customers.id%TYPE) IS
  c_sal customers.salary%TYPE;
  BEGIN
    SELECT salary INTO c_sal
    FROM customers
    WHERE id = c_id;
    dbms_output.put_line('Salary: '|| c_sal);
  END find_sal;
END cust_sal;
/
```

TRANSACTION

Transaction- A transaction is a sequence of operations performed (using one or more SQL statements) on a database as a single logical unit of work.

ACID properties

1. ATOMIC-Either all actions are performed or none are. Not to worry about incomplete transaction.
2. CONSISTENCY-DBMS assures that the consistency holds for each transaction.
3. ISOLATION-Transactions are isolated or protected from the effects of concurrently scheduling other transactions.
4. DURABILITY-The effect of transaction is present if DBMS informs the user successful execution.

Schedule-A list of actions from a set of transactions seen by DBMS.

1. SERIAL SCHEDULE-Schedule does not interleave the actions of different transactions.
2. EQUIVALENT SCHEDULE-The effect of executing the first schedule is identical to the effect of executing the second schedule.
3. SERIALIZABLE SCHEDULE-A schedule that is equivalent to same serial execution of the transactions on any consistent database instance .

DEADLOCKS- A deadlock occurs when two (or more) processes lock the separate resource. Under these circumstances, each process cannot continue and begins to wait for others to release the resource.

CRASH RECOVERY

ARIES ALGORITHM

Algorithm for Recovery and Isolation Exploiting Semantics (ARIES) is based on the Write Ahead Log (WAL) protocol.

1. Undo-only log record: Only the before image is logged. Thus, an undo operation can be done to retrieve the old data.
2. Redo-only log record: Only the after image is logged. Thus, a redo operation can be attempted.
3. Undo-redo log record: Both before images and after images are logged.

The recovery process actually consists of 3 phases:

1. Analysis: The recovery subsystem determines the earliest log record from which the next pass must start. It also scans the log forward from the checkpoint record to construct a snapshot of what the system looked like at the instant of the crash.
2. Redo: Starting at the earliest LSN, the log is read forward and each update redone.
3. Undo: The log is scanned backward and updates corresponding to loser transactions are undone.

LOG BASED RECOVERY

- The log is a sequence of records. Log of each transaction is maintained in some stable storage so that if any failure occurs, then it can be recovered from there.
- If any operation is performed on the database, then it will be recorded in the log.
- But the process of storing the logs should be done before the actual transaction is applied in the database.

Let's assume there is a transaction to modify the City of a student. The following logs are written for this transaction.

- When the transaction is initiated, then it writes 'start' log.
    1. <T1, Start>

- When the transaction modifies the City from 'Noida' to 'Bangalore', then another log is written to the file.
    1. <T1, City, 'Noida', 'Bangalore' >
- When the transaction is finished, then it writes another log to indicate the end of the transaction.
    1. <T1 Commit>

WRITE-AHEAD LOG PROTOCALL

Write-ahead logging (WAL) is a family of techniques for providing atomicity and durability (two of the ACID properties) in database systems. A write ahead log is an append-only auxiliary disk-resident structure used for crash and transaction recovery.

REDO AND UNO PHASE

Redo: Repeats all actions, starting from an appropriate point in the log, and restores the database state to what it was at the time of the crash.

Undo: Undoes the actions of transactions that did not commit, so that the database reflects only the actions of committed transactions.