Oracle PL/SQL

PL/SQL Transaction and Cursors

Agenda

PL/SQL Transaction

Transaction in PL/SQL

- **PL/SQL is a transaction-oriented language**
- **It provides data integrity**
- **A series of SQL data manipulation statements that are work logical unit**
- **An atomic unit of all changes either committed or rollback**

COMMIT	ROLLBACK	SAVEPOINT	AUTOCOMMIT	SET TRANSACTION
COMMIT command is used to make changes permanently saved to a database during the current transaction	ROLLBACK command executes at the end of the current transaction and does undo/undone any change made since the beginning of the transaction	SAVEPOINT command saves the current point w ith the unique name in the processing of a transaction	Set AUTOCOMMIT ON to execute the COMMIT statement automatically	PL/SQL SET TRANSACTION command sets the transaction properties such as read-w rite/read-only access

COMMIT

- **The COMMIT statement is used to make changes permanently saved to a database during the current transaction and visible to other users**

Syntax

```
SQL>COMMIT
[COMMENT
"comment text"];
```

Example

```
SQL>BEGIN
   UPDATE emp_information SET
emp_dept='XXX Developer'
      WHERE emp_name='ABC';
   COMMIT;
END;
/
```

ROLLBACK

- **The ROLLBACK statement ends the current transaction and does undo any changes made during that transaction. If you make a mistake, such as deleting the wrong row from a table, a rollback restores the original**

Syntax

```
SQL>ROLLBACK
[To SAVEPOINT_NAME];
```

Example

```
SQL>DECLARE
  emp_id  emp.empno%TYPE;
BEGIN
  SAVEPOINT  dup_found;
  UPDATE  emp SET eno=1
      WHERE  empname = 'Forbs ross'
EXCEPTION
  WHEN  DUP_VAL_ON_INDEX  THEN
     ROLLBACK TO dup_found;
END;
/
```

SAVEPOINT

- **SAVEPOINT savepoint_names marks the current point in the processing of a transaction. SAVEPOINT lets you rollback part of a transaction instead of the whole transaction**

Syntax

```
SQL>SAVEPOINT
SAVEPOINT_NAME;
```

Example

```
SQL>DECLARE
  emp_id  emp.empno%TYPE;
BEGIN
  SAVEPOINT  dup_found;
  UPDATE  emp SET eno=1
      WHERE  empname = 'Forbs ross'
EXCEPTION
  WHEN  DUP_VAL_ON_INDEX  THEN
     ROLLBACK TO dup_found;
END;
/
```

AUTOCOMMIT

- **No need to execute the COMMIT statement every time.**
- **You just set AUTOCOMMIT ON to execute the COMMIT statement automatically. It automatically executes for each DML statement.**

Example

```
SQL>SET AUTOCOMMIT ON;
```

Example

```
SQL>SET AUTOCOMMIT OFF;
```

SET TRANSACTION

- **SET TRANSACTION statement is used to set the transaction as read-only or both read and write. You can also assign the transaction name using this statement.**

Syntax

```
SQL>SET TRANSACTION [
READ ONLY | READ WRITE ]
      [ NAME 'transaction_name' ];
```

Example

```
SQL>SET TRANSACTION READ WRITE NAME
'tran_exp';
```

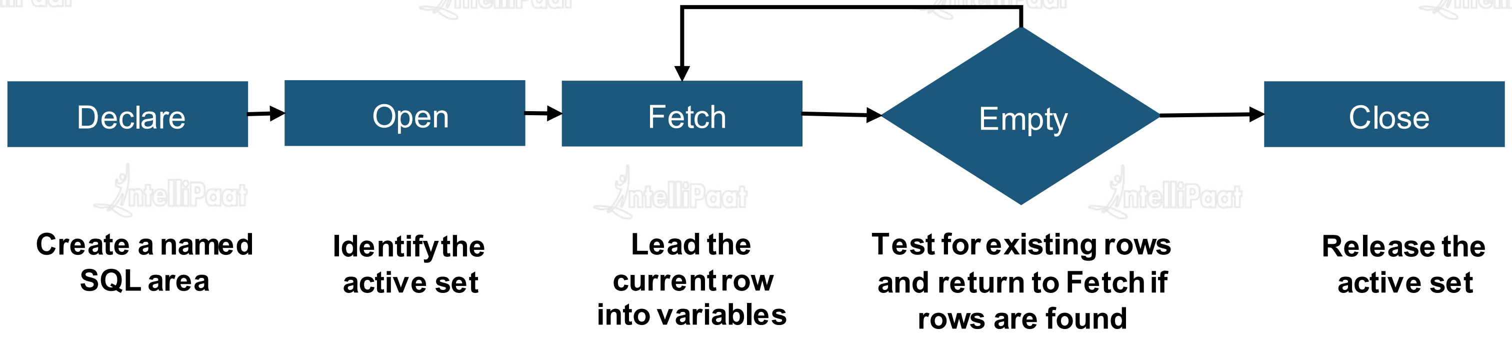
PL/SQL Cursors

Cursors in PL/SQL

- Every SQL statement executed by Oracle Server has an individual cursor associated with it.

Implicit Cursor	Explicit Cursor
•Declared for all DML and PL/SQL SELECT statements	•Declared and named by the programmer

- Use CURSOR to individually process each row returned by a multiple-row SELECT statement
- The set of rows returned by a multiple-row query is called an active set.


Declare
Open
Fetch
Empty
Close
Create a named SQL area
Identify the active set
Lead the current row into variables
Test for existing rows and return to Fetch if rows are found
Release the active set

Declare a Cursor

- **A cursor is a SELECT statement that is defined within the declaration section of your PL/SQL code.**

Cursor without parameters

```
CURSOR cursor_name
IS
  SELECT_statement;
```

Example

```
CURSOR c1
IS
  SELECT course_number
  FROM courses_tbl
  WHERE course_name =
name_in;
```

Cursor with parameters

```
CURSOR cursor_name
(parameter_list)
IS
  SELECT_statement;
```

Example

```
CURSOR c2 (subject_id_in IN
varchar2)
IS
  SELECT course_number
  FROM courses_tbl
  WHERE subject_id = subject_id_in;
```

Cursor with a return clause

```
CURSOR cursor_name
RETURN field%ROWTYPE
IS
  SELECT_statement;
```

Example

```
CURSOR c3
RETURN courses_tbl%ROWTYPE
IS
  SELECT *
  FROM courses_tbl
  WHERE subject = 'Mathematics';
```

OPEN Statement

- **Once you've declared your cursor, the next step is to use the OPEN statement to open the cursor**

Syntax

```
OPEN cursor_name;
```

Example

```
OPEN c1;
```

SELECT FOR UPDATE Statement

- **SELECT FOR UPDATE statement allows you to lock records in the cursor result set.**
- **You are not required to make changes to the records in order to use this statement.**
- **The record locks are released when the next commit or rollback statement is issued.**

Syntax

```
CURSOR cursor_name
IS
  select_statement
  FOR UPDATE [OF column_list]
[NOWAIT];
```

Example

```
CURSOR c1
IS
  SELECT course_number,
instructor
  FROM courses_tbl
  FOR UPDATE OF instructor;
```

WHERE CURRENT OF Statement

- If you plan to update or delete records that have been referenced by a SELECT FOR UPDATE statement, you can use the WHERE CURRENT OF statement.

Syntax

```
UPDATE  table_name
  SET set_clause
  WHERE  CURRENT  OF
cursor_name;
```

```
DELETE FROM table_name
WHERE  CURRENT  OF
cursor_name;
```

Example

```
UPDATE  courses_tbl
        SET instructor = 'SMITH'
        WHERE  CURRENT  OF c1;
```

```
 DELETE FROM courses_tbl
        WHERE  CURRENT  OF c1;
```

Fetching Data from the Cursor

- **Retrieve the first 10 employees one by one**

```
SET SERVEROUTPUT  ON
DECLARE
          v_empno employees.employee_id%TYPE;
          v_ename employees.last_name%TYPE;
                    CURSOR  emp_cursor IS
                    SELECT employee_id, last_name FROM employees;
BEGIN
          OPEN emp_cursor;
                    FOR i IN  1..10 LOOP
                    FETCH emp_cursor INTO  v_empno, v_ename;
                    DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_empno)||'   '||v_ename);
          END LOOP;
CLOSE emp_cursor;
END;
```

Explicit Cursor Attributes

cname%ROWCOUNT	Number	Evaluates to the total number of rows returned so for
cname%FOUND	Boolean	Evaluates to TRUE if the most recent fetch returns a row
cname%NOTFOUND	Boolean	Evaluates to TRUE if the most recent fetch does not return a row
cname%ISOPEN	Boolean	Evaluates TRUE if the cursor is open

```
IF NOT emp_cursor%ISOPEN THEN
            OPEN emp_cusor;
END IF;
LOOP
 FETCH emp_cursor....
```

```
LOOP
            FETCH c1 INTO my_ename,
my_sal;

            EXIT WHEN c1%NOTFOUND;
.. ..
END LOOP;
```

```
LOOP
            FETCH c1 INTO my_deptno;
            IF c1%ROWCOUNT > 10 THEN
                  .. ..
            END IF;
 .. ..
END LOOP;
```

Explicit Cursor Attributes: Example

- **Retrieve the first 10 employees one by one using attributes**

```
SET SERVEROUTPUT  ON
DECLARE
            v_empno employees.employee_id%TYPE;
            v_ename employees.last_name%TYPE;
            CURSOR emp_cursor IS
                        SELECT employee_id, last_name FROM employees;
BEGIN
            OPEN emp_cursor;
            LOOP
                        FETCH emp_cursor INTO  v_empno, v_ename;
                        EXIT  WHEN  emp_cursor%ROWCOUNT  > 10 OR
emp_cursor%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(TO_CHAR(v_empno)||'  '||v_ename);
      END  LOOP;
  CLOSE emp_cursor;
END;
```

Cursor and Records

- **Process rows of the active set by fetching values into the PL/SQL record**
- Populate to the table temp_list

```
DECLARE
          CURSOR emp_cursor IS
          SELECT employee_id, last_name FROM employees;
        emp_record emp_cursor%ROWTYPE;
BEGIN
        OPEN emp_cursor;
          LOOP
              FETCH emp_cursor INTO emp_record;
              EXIT WHEN emp_cursor%NOTFOUND;
                INSERT INTO temp_list (emp_id, ename)
                VALUES (emp_record.employee_id, emp_record.last_name);
        END LOOP;
        COMMIT;
   CLOSE emp_cursor;
END;
```

Cursor FOR Loop

- **Implicit Open, Fetch, and Close occur here**
- **The record is implicitly declared**
- Retrieve employees one by one who are working in department 80

```
SET SERVEROUTPUT  ON
DECLARE
            CURSOR emp_cursor IS
            SELECT last_name, department_id FROM employees;
BEGIN
            FOR emp_record IN emp_cursor LOOP
                        -- implicit open fetch occur
              IF emp_record.department_id  = 80 THEN
              DBMS_OUTPUT.PUT_LINE('Employee  ' || emp_record.last_name || ' works in the Sales Dept.');
              END IF;
            END LOOP; -- implicit close
END;
```

Cursor FOR Loop

- **No need to declare the cursor, if FOR loop is used**
- **Same result as the previous slide**
- Retrieve employees one by one who are working in department 80

```
SET SERVEROUTPUT  ON
BEGIN
      FOR emp_record  IN
      (SELECT last_name,  department_id  FROM employees)  LOOP
                                  -- implicit open fetch occur
        IF emp_record.department_id  = 80 THEN   DBMS_OUTPUT.PUT_LINE('Employee  ' || emp_record.last_name  || ' works in the Sales  Dept.');
        END  IF;
      END LOOP;                   -- implicit close
END;
```

Cursor with Parameters

- **Pass parameter values to a cursor using WHERE and open an explicit cursor different times, each time with a different active set.**

```
SET SERVEROUTPUT  ON
    DECLARE
                CURSOR emp_cursor  (p_dno  NUMBER)  IS
                                SELECT employee_id,  last_name  FROM  employees
                                WHERE department_id  = p_dno;
BEGIN
                FOR emp_record  IN emp_cursor(50)  LOOP
                      DBMS_OUTPUT.PUT_LINE(‘Employee   ‘ || emp_record.employee_id||’   ‘|| emp_record.last_name  ||‘ works in 50‘);
                END LOOP;
                FOR emp_record  IN emp_cursor(60)  LOOP
                     DBMS_OUTPUT.PUT_LINE(‘Employee   ‘ || emp_record.employee_id||’   ‘|| emp_record.last_name  ||‘ works  in 60‘);
                END LOOP;
END;
/
```

Advanced Explicit Cursor

FOR UPDATE Clause

- Use explicit locking to deny access for the duration of a transaction
- Lock the rows before update or delete
- NOWAIT keyword tells not to wait if the requested rows have been locked by another user

WHERE CURRENT OF cursor

- To reference the current row from an explicit cursor

```
DECLARE
        CURSOR sal_cursor IS
          SELECT e.department_id, employee_id, last_name, salary
          FROM employees e, departments d
          WHERE d.department_id = e.department_id AND d.department_id=60
        FOR UPDATE OF salary NOWAIT;
BEGIN
        FOR emp_record IN sal_cursor
        LOOP
           IF emp_record.salary < 5000 THEN
              UPDATE employees
              SET salary = emp_record.salary * 1.10
              WHERE CURRENT OF sal_cursor ;
           END IF;
        END LOOP;
END;
```

Quiz

Quiz 1

A transaction ends when

A	A COMMIT or a ROLLBACK statement is issued.
B	A DDL statement, like CREATE TABLE statement, is issued; because in that case a COMMIT is automatically performed.
C	A DCL statement, such as a GRANT statement, is issued; because in that case a COMMIT is automatically performed.
D	All of the above

Answer 1

A transaction ends when

A. A COMMIT or a ROLLBACK statement is issued.

B. A DDL statement, like CREATE TABLE statement, is issued; because in that case a COMMIT is automatically performed.

C. A DCL statement, such as a GRANT statement, is issued; because in that case a COMMIT is automatically performed.

D. All of the above

Quiz 2

When a user creates an object without a TABLESPACE clause, where will Oracle store the segment?

A	Users tablespace
B	System tablespace
C	Default tablespace for the user
D	Default tablespace for the user

Answer 2

When a user creates an object without a TABLESPACE clause, where will Oracle store the segment?

A Users tablespace

B System tablespace

[✓ C Default tablespace for the user]

D Default tablespace for the user

Quiz 3

Which Oracle access method is the fastest way for Oracle to retrieve a single row?

A	Access via unique index
B	Full table scan
C	Primary key access
D	Table access by ROWID

Answer 3

Which Oracle access method is the fastest way for Oracle to retrieve a single row?

A Access via unique index

B Full table scan

C Primary key access

[✓ D Table access by ROWID]

Quiz 4

The following code tries to fetch some information from all the rows in a table named customers for use in a PL/SQL block. What is wrong in the following code?

A It need not use a cursor.

B The cursor is not opened.

C It will not print information from all the rows.

D There is nothing wrong in the code.

```
DECLARE
  c_id customers.id%type;
  c_name customers.name%type;
  c_addr customers.address%type;
  CURSOR c_customers is
    SELECT id, name, address FROM customers;
BEGIN
  LOOP
    FETCH c_customers into c_id, c_name, c_addr;
    EXIT WHEN c_customers%notfound;
    dbms_output.put_line(c_id || ' ' || c_name || ' ' ||
c_addr);
  END LOOP;
  CLOSE c_customers;
END;
```

Answer 4

The following code tries to fetch some information from all the rows in a table named customers for use in a PL/SQL block. What is wrong in the following code?

A It need not use a cursor.

B The cursor is not opened.

C It will not print information from all the rows.

D There is nothing wrong in the code.

```
DECLARE
  c_id customers.id%type;
  c_name customers.name%type;
  c_addr customers.address%type;
  CURSOR c_customers is
    SELECT id, name, address FROM customers;
BEGIN
  LOOP
    FETCH c_customers into c_id, c_name, c_addr;
    EXIT WHEN c_customers%notfound;
    dbms_output.put_line(c_id || '' || c_name || '' ||
c_addr);
  END LOOP;
  CLOSE c_customers;
END;
```

Quiz 5

What does a COMMIT statement do to a CURSOR?

A Open the Cursor

B Fetch the Cursor

C Close the Cursor

D None of the above

Quiz 5

What does a COMMIT statement do to a CURSOR?

A Open the Cursor

B Fetch the Cursor

C Close the Cursor

D None of the above

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor