GALGOTIAS UNIVERSITY

School of Computing Science and Engineering

Program: BCA - IOP

Course Code: BCAS3031

Course Name: PL/SQL & Cursors and Triggers

Dr. T. Poongodi

Associate Professor

PL/SQL Transaction Commit, Rollback, Savepoint, Autocommit

- Oracle PL/SQL transaction oriented language. Oracle transactions provide a data integrity.
- PL/SQL transaction is a series of SQL data manipulation statements that are work logical unit.
- Transaction is an atomic unit all changes either committed or rollback.
- At the end of the transaction that makes database changes, Oracle makes all the changes permanent save or may be undone.
- If the program fails in the middle of a transaction, Oracle detect the error and rollback the transaction and restoring the database.

Use the COMMIT, ROLLBACK, SAVEPOINT, and SET TRANSACTION command to control the transaction.

- COMMIT: COMMIT command to make changes permanent save to a database during the current transaction.
- ROLLBACK: ROLLBACK command execute at the end of current transaction and undo/undone any changes made since the begin transaction.
- SAVEPOINT: SAVEPOINT command save the current point with the unique name in the processing of a transaction.
- AUTOCOMMIT: Set AUTOCOMMIT ON to execute COMMIT Statement automatically.
- SET TRANSACTION: PL/SQL SET TRANSACTION command set the transaction properties such as read-write/read only access.

Commit

- The COMMIT statement to make changes permanent save to a database during the current transaction and visible to other users.
- Commit comments are only supported for backward compatibility.

Commit Syntax

```
SQL>COMMIT
```

```
SQL>  BEGIN
      UPDATE emp_information
      SET emp_dept='Web Developer’
      WHERE emp_name='Saulin’;
      COMMIT;
      END; /
```

Rollback

- The ROLLBACK statement ends the current transaction and undoes 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 data.
- If you cannot finish a transaction because an exception is raised or a SQL statement fails, a rollback lets you take corrective action and perhaps start over.

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

```
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;
      /
```

Exception raised because eno = 1 is already so DUP_ON_INDEX exception rise and rollback to the dup_found savepoint named.

Savepoint

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

```
SQL>SAVEPOINT SAVEPOINT_NAME;
```

```
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 COMMIT statement every time.
- Set AUTOCOMMIT ON to execute COMMIT Statement automatically.
- It's automatic execute for each DML statement. set auto commit on using following statement,

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

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

Set Transaction

- SET TRANSACTION statement is use to set transaction are read-only or both read write. you can also assign transaction name.

SET TRANSACTION Syntax

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

Set transaction name using the SET TRANSACTION [...] NAME statement before you start the transaction.

SET TRANSACTION Example

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

```
DECLARE
    rollno student.sno%type;
    snm student.sname%type;
    s_age student.age%type;
    s_cr student.course%type;
BEGIN
    rollno := &sno;
    snm := '&sname';
    s_age := &age;
    s_cr := '&course';
    INSERT into student values(rol
    dbms_output.put_line('One reco
```

- There is a table called **STUDENT** in the database with columns **sno** as number, **sname** as varchar2, **age** as number and **course** as varchar2.
- Executed an `INSERT` statement and then used the `COMMIT` statement to commit or permanently save the changes into the database.
- Instead of the `COMMIT` statement, if we use the `ROLLBACK` statement there, then even though the `INSERT` statement executed successfully, still, after the execution of the PL/SQL block if you will check the Student table in the database, you will not find the new student entry because we executed the `ROLLBACK` statement and it rolled back the changes

PL/SQL Code Example with Savepoint and Rollback

- Add two insert statement in the above code and put a savepoint in between them and then use the ROLLBACK command to revert back changes of one insert statement.
- After execution of the above code, we will have **one entry** created in the Student table, while the second entry will be rolled back.

```
set serveroutput on;
DECLARE
rollno student.sno%type;
snm student.sname%type;
s_age student.age%type;
s_cr student.course%type;
BEGIN
      rollno := &sno;
      snm := '&sname';
      s_age := &age;
      s_cr := '&course';
      INSERT into student
      values(rollno,snm,s_age,s_cr);
      dbms_output.put_line('One record inserted');
      COMMIT;
```

```
-- adding savepoint
SAVEPOINT savehere;
-- second time asking user for input
rollno := &sno;
snm := '&sname';
s_age := &age;
s_cr := '&course';
INSERT into student values(rollno,snm,s_age,s_cr);
dbms_output.put_line('One record inserted');
ROLLBACK TO savehere;
END;
```


Thank You