Dynamic SQL

**Static SQL**

Any SQL statement which does not change during the runtime and remain fixed is called a static SQL statement.

Biggest advantage of static SQL statement is that we hardcode these statements into our application thus we can tune them for optimal performance.

**Native Dynamic SQL**

Any SQL statement which is constructed at the runtime is called dynamic SQL.

As these statements are built on the fly thus they cannot be hardcoded into the application which in turn increases the flexibility.

Using Dynamic SQL we can perform DDL operations inside the pl/sql.

Similar to dynamic SQL, the process of constructing pl/sql code at runtime is called dynamic pl/sql.

How do we use dynamic SQL

Execute immediate statement and

Open-For, fetch and close block.

Bulk fetch

Bulk execute immediate

Bulk forall and

Bulk collection into statement

**Execute immediate**

Using execute immediate we can parse and execute any SQL statement or a pl/sql block dynamically in oracle database.

It is best suited for those SQL statements which returns a single row of results.

**Syntax**

Execute immediate dynamic\_query [into user\_defined\_var1,user\_Defined\_var2]

[using bind\_Argument-1.bind\_argumant-2]

[returning |return into caluse];

Example

Declare

Sql\_query varchar2(200);

Emp\_tot number;

Begin

Sql\_query :=’select count(\*) from emp’;

Execute immediate Sql\_query into Emp\_tot ;

Dbms\_output.put\_line(‘totla emp:’ || Emp\_tot );

End ;

/

2. create (DDL statements)

Create

Alter

Drop

Declare

Ddl\_query varchar2(3000);

Begin

Ddl\_query:= ‘create table tut\_89(col1 number(3), col2 varchar2(30))’;

Execute immediate Ddl\_query;

End;

/

Declare

Ddl\_query varchar2(3000);

Begin

Ddl\_query:= ‘create table tut\_89(‘||’col1 number(3),’||’ col2 varchar2(30)’||’)’;

Execute immediate Ddl\_query;

End;

/

Declare

Ddl\_query varchar2(4000);

Begin

Ddl\_query:= ‘alter table tut\_89 add col3 date’;

Execute immediate Ddl\_query;

End;

/

Declare

Ddl\_query varchar2(4000);

Begin

Ddl\_query:= ‘drop table tut\_89’;

Execute immediate Ddl\_query;

End;

/

Bind variables

In order to use bind variables in dynamic SQL we need to take the help of using clause of execute immediate statement.

Specifying values for all bind variables is must.

Few important point

It is mandatory for us to specify the values for all the bind variables in the list of USING clause.

How the values from the list of using clause gets assigned to the values.

We cannot use the bind varaibles for specifying any schema object name in native dynamic SQL.

Need of bind variables

Security against SQL injection

Performance enhancements by reducing hard parsing.

Example

Declare

sql\_query varchar2(4000);

Begin

sql\_query:= ‘insert into tut\_89 values(:col1,:col2)’;

Execute immediate Ddl\_query using 1,’aaa’;

End;

/

Declare

sql\_query varchar2(4000);

Begin

sql\_query:= ‘update tut\_89 set col2=:var1 where col2=:var2’;

Execute immediate Ddl\_query using ‘bbb’,’aaa’;

End;

/

Example using table name in using clause

Declare

sql\_query varchar2(4000);

Begin

sql\_query:= ‘update :tablenaem set col2=:var1 where col2=:var2’;

Execute immediate Ddl\_query using ‘tut\_89 ‘,‘bbb’,’aaa’;

End;

/

Ora error: -invalid table name

Bulk collect with execute immediate

Set serveroutput on;

Declare

Type nt\_name is table of varchar2(60);

Fname nt\_name;

Sql\_query varchar2(4000);

Begin

Sql\_query:= select ename bulk collect into fname from emp’; // is wrong of using

Sql\_query:= ‘select ename from emp’;

Execute immediate sql\_query bulk collect into fname;

For I in 1..fname.count

Loop

Dbms\_output.put\_line(‘emp name’||fname(i));

End loop;

End;

/

Plsql block with execute immediate

Declare

Pl\_ssqlblock varchar2(4000);

Begin

Pl\_ssqlblock :=’

Declare

Var\_name varchar2(20);

Begin

Select user into var\_name from dual;

Dbms\_output.put\_line(‘’current user’’||var\_name);

End;’;

End;

/

/