Part 14

Oracle PL/SQL
Oracle PL/SQL

Oracle’s PL/SQL is used throughout the Oracle system in the various modules available, including:

- SQL*Plus
- Oracle Forms
- Oracle Reports
- SQL*Module
- Oracle Graphics
- Oracle Glue
- Developer 2000 suite

PL/SQL is essentially a programming language containing SQL commands. As such, it has all the features of a regular programming language including:

- Arithmetic operators (+, -, *, /, **)  
- Relational operators (<> ^= != < > =)
- List enclosures and list item separators ((, ), ,)
  ('Bill', 'Mary', 'Sue')
- Item separators (.)  
- select ename from student1.emp
- Character string enclosures (')
- Assignment (=)     
- Concatenation (||)
- End-of-line comment (--)  
- Open comment/close comment (/*, */)
- Attribute indicator (%)
Variables and Reserved Words

Variables must start with a letter (case insensitive), no longer than 30 characters, may contain 0-9, $, #, _, and cannot be any of the reserved words

Reserved words are used as keywords in the language. There are 185 reserved words

A variable is established in a declare sequence:

```sql
declare
    employee    varchar2(30);
    counter     number; -- but not count
    my_date     date;
    is_eof      Boolean;
```
List of PL/SQL Reserved Words

ABORT
ACCEPT
ACCESS
ADD
ALL
ALTER
AND
ANY
ARRAY
ARRAYLEN
AS
ASC
ASSERT
ASSIGN
AT
AUTHORIZATION
AVG
BASE_TABLE
BEGIN
BETWEEN
BINARY_INTEGER
BODY
BOOLEAN
BY
CASE
CHAR
CHAR_BASE
CHECK
CLOSE
CLUSTER
CLUSTERS
COLAUTH
COLUMNS
COMMIT
COMPRESS
CONNECT
CONSTANT
COUNT
CRASH
CREATE
CURRENT
CURSOR
CURRVAL
DATABASE
DATA_BASE
DATE
DBA
DEBUGOFF
DEBUGON
DECLARE
DECLARATION
DECIMAL
DEFAULT
DEFINITION
DELAY
DELETE
DELTA
DESC
DIGITS
DISPOSE
DISTINCT
DO
DROP
ELSE
ELSIF
END
ENTRY
EXCEPTION
EXCEPTION_INIT
EXISTS
EXIT
FALSE
FETCH
FLOAT
FOR
FORM
FROM
FUNCTION
GENERIC
GOTO
GRANT
GROUP
HAVING
IDENTIFIED
IF
IN
INDEX
INDEXES
INDICATOR
INSERT
INTEGER
INTERSECT
INTO
IS
LEVEL
LIKE
LIMITED
LOOP
MAX
MIN
MINUS
MLSLABEL
MOD
NATURAL
NEW
NEXTVAL
NONE
NOT
NULL
NUMBER
NUMBER_BASE
OF
ON
OPEN
OPTION
OR
ORDER
OTHERS
PL/SQL Components

The basic components of PL/SQL code include the following:

1. A declare section (seen above)
2. Exceptions - a means of dealing with error conditions
3. Control structures, including:
   - program control
   - if logic structures
   - looping structures
4. “Do nothing” construct

The common error conditions to test for are:

- no_data_found: true if retrieved row count = 0
- too_many_rows: true if more than one row retrieved on an “implicit cursor”
- dup_val_on_index: true if attempt made to enter a duplicate value into an index
- value_error: true if value is too long to fit into target field
Logic Structures

If - Then - Else - Elsif - End If

if not (var1 > 10) then
    var2 := var1 + 20;
    var3 := var1 + 30;
elsif var1 between 2 and 4 then
    var2 := var1 + 6;
else
    var3 := var1 + 6;
end if;

Looping
    Loop
    End Loop
    Exit
    Exit When
    While ... Loop
    For ... In ... Loop
Looping Examples

All of the loops below will execute … 100 times, for values of cnt from 1 to 100, inclusive.

cnt := 0;
loop
    cnt := cnt + 1;
    if cnt > 100 then
        exit;
    end if;
    ...
end loop;

cnt := 0;
loop
    cnt := cnt + 1;
    exit when cnt > 100
    ...
end loop;

cnt := 0;
while cnt < 100 loop
    cnt := cnt + 1;
    ...
end loop;

for cnt in 1 .. 100 loop
    ...
end loop;
PL/SQL Cursor Looping Example

drop table temp;
create table temp
    (last_name    char(20),
     sequence_no integer);
declare
    lname    char(20);
    cnt      integer;
cursor mycur is
    select ename
    from emp
    order by ename;
begin
    open mycur;
    loop
        fetch mycur into lname;
        cnt := mycur%rowcount;
        if mycur%found then
            insert into temp
            values (lname, cnt);
        else
            exit;
        end if;
    end loop;
end;
Attribute Indicator

The attribute indicator operator can be used to return the value of a named attribute of the object. The object appears before the % and the attribute name appears after the %.

Some attributes that might be of use:

   For cursors:
      rowcount
      found
      notfound
      isopen

   For variables, constants, and database columns:
      rowtype
      type

Example usage:

declare
   emp_rec     emp%rowtype;
   credit      number(7,2);
   debit       credit%type;
begin
   select * into emp_rec from emp where ...  
      ...
end;
Language Characteristics

1. Case insensitive
2. Block structured
3. ; terminated
4. Procedural
5. SQL-Like
6. Exception-handling
Exception Handling

begin
  ...
exception
  when zero_divide then
    ...
  when cursor_already_open then
    ...
  when no_data_found then
    ...
  ...
  when others then
    ...
end;
Creating a procedure

First, make sure any needed tables, such as countloop in this example, exist and have the correct fields in the correct order.

Then create/replace the procedure in SQL
create or replace procedure countest
as
begin
declare
cnt integer;
begin
delete from countloop;
for cnt in 1 .. 5 loop
insert into countloop values (cnt);
end loop;
end;
end;

If all syntax is correct, you will see the message:
Procedure created.

If there is a syntax error, you will see the message:
Warning: Procedure created with compilation errors.
To see the error messages from compilation, use the SQL*Plus statement:
show error
Executing a Procedure

Once a procedure is in existence, it can be invoked by the SQL*Plus statement:
execute countest

Assuming the stored procedure has been successfully compiled, you will see the message:
PL/SQL procedure successfully completed.

To actually see the result of the procedure, you will have to examine the tables affected by the statements executed in the procedure:
select * from countloop;

<table>
<thead>
<tr>
<th>CT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>