PL/SQL and Security: Chapters 7 and 12

CIS 399, Week 4
Intro to PL/SQL

Provides pre-compiled query logic
Looping, selection, variables, error handling, object-oriented design

Benefits:
• Handling Exceptions
• Iteration of Rows
• Securing through Encryption
• Stored code in the database
• Decision making and conditional branching
PL/SQL block types

Anonymous blocks

Subprograms:
  Procedures
  Functions
Anonymous Blocks

Declarative section
  Variables
  Constants

Executable section
  Actual code

Exception handling section
  Gracefully handle any errors that occur

[DECLARE]
BEGIN
[EXCEPTION]
END;
Declarative section

C_date CONSTANT DATE := SYSDATE;
V_first Agents.FirstName%TYPE;
V_yearsonjob NUMBER (4,2);

<variable name> [CONSTANT] <datatype> [NOT NULL] [{ := | DEFAULT} <value or expression>
Executable section

All PL/SQL statements between “BEGIN” and “END” statements

Any SELECT statements open a “cursor”
  Implicit cursor
  Explicit cursor

```
SELECT FirstName, LastName (c_date - HireDate)/365.25 INTO v_first, v_last, v_yearsonjob FROM Agents
WHERE LastName = '&AgentLastName';
```
Exception Handling

/* this will raise an exception */
SELECT 9/0 INTO v_bogus FROM DUAL;

/* handling exceptions: */
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE(“Divide by zero error.”);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
Full Examples

see ‘/space/ordata/classfiles/Chapter07/Chapter/AnonBlockOne.sql’
and ‘/space/ordata/classfiles/Chapter07/Chapter/AnonBlockTwo.sql’
and ‘/space/ordata/classfiles/Chapter07/Chapter/AnonBlockThree.sql’

– the last two will have errors…
Loops

Simple Loop

LOOP

   statements

END LOOP;

WHILE Loop

WHILE <condition> LOOP

   statements

END LOOP;

FOR Loop

FOR <loop variable> IN [REVERSE] <lower bound>..<upper bound> LOOP

   statements

END LOOP;
Simple loop example

/* loop 10 times */

v_counter := 1;
LOOP
    DBMS_OUTPUT.PUT_LINE(v_counter);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 10;
END LOOP;
While loop example
/* loop 10 times */

v_counter := 1;
WHILE v_counter <= 10 LOOP
  DBMS_OUTPUT.PUT_LINE(v_counter);
  v_counter := v_counter + 1;
END LOOP;
For loop examples

/* loop 10 times */
FOR v_counter IN 1..10 LOOP
    DBMS_OUTPUT.PUT_LINE(v_counter);
END LOOP;

/* loop 10 times, backwards! */
FOR v_counter IN REVERSE 1..10 LOOP
    DBMS_OUTPUT.PUT_LINE(v_counter);
END LOOP;
Declaring Cursor Variables

CURSOR cv_TeamCursor IS
SELECT Team, Avg(BaseSalary)
FROM AgentsHR
GROUP BY Team ORDER BY Team;

Syntax:

CURSOR <cursor name> [ <parameter list> ]
RETURN <return type> IS <query> [FOR UPDATE [OF (<column list>)[NOWAIT]]];
Opening, Fetching, Closing

OPEN cv_TeamCursor;
LOOP
    FETCH cv_TeamCursor INTO v_TeamNo, v_AvgSalary;
    EXIT WHEN cv_TeamCursor%NOTFOUND;
    ...
END LOOP;
CLOSE cv_TeamCursor;
Putting it all together…

See ‘/space/ordata/classfiles/Chapter07/Chapter/ExplicitCursor.sql’
Using FOR Loops for cursors

No open, no close, no declaring variables…

FOR <record name> IN <cursor name>
  LOOP
    ...
  END LOOP;

See ‘/space/ordata/classfiles/Chapter07/Chapter/ForCursor.sql’
IF-THEN-ELSE Statements

IF <condition> THEN
  ...
{[ELSIF <condition> THEN
    ...]}
[ELSE
  ...
] END IF;
IF-THEN-ELSE example

DECLARE

v_count INTEGER := 0;
BEGIN

SELECT COUNT(*) INTO v_count FROM Agents WHERE LastName = 'Smith' AND FirstName = 'Joe';
IF v_count = 0 then
    INSERT ...
ELSE
    UPDATE ...
END IF;
END;
Another IF-THEN-ELSE example

See ‘/space/ordata/classfiles/Chapter07/Chapter/ExplicitCursor.sql’