Lecture 6: Review/Introduction of SQL and Oracle Proc*C/C++

(03-25-2020)

(Reading: Lecture Notes)

Lecture Outline

```
1. Introduction
2. Using Oracle Pro*C/C++ Pre-compiler in UNIX and
   Windows platforms
3. Fundamentals
4. Dynamic SQL
5. Examples
```

1. Introduction

(1) Motivations

a. SQL

(a) Benefits: DML and DDL

(b) Problems: no control structures

b. High-level programming languages

(a) Benefits: powerful programming mechanism

(b) Problems: no access to databases

c. PL/SQL

(a) Benefits: extending SQL to a full PL

(b) Problems: restricted to PL/SQL alone

(2) Basic ideas of Oracle Pro* C/C++

a. The role of the pre-compiler

b. The necessary supporting elements

(a) Header files that contain type, constant, and variable declarations needed in declaring embedded SQL elements;

(b) SQL library that contains object code for each function/procedure that is generated by the pre-compiler

(3) All supported functionalities of Oracle Pro*C/C++:

(4) An example: sample1.pc

2. Fundamentals

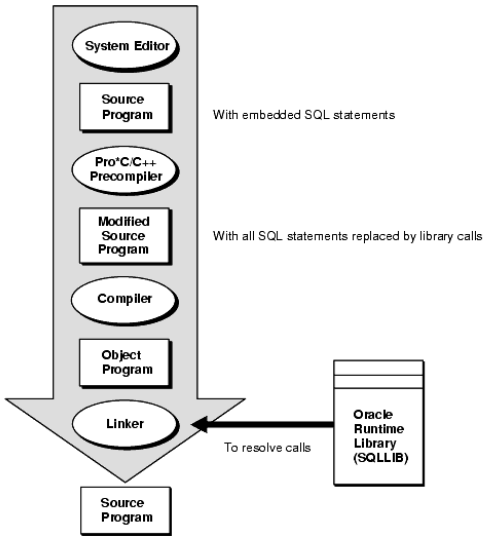

Figure 16: Illustration of Embedded SQL

(1) Key concepts of embedded SQL programming

a. Embedded SQL statements: Two types of embedded SQL statements – executable statements and declaration directives

(a) Declaration directives (Table 1)

(b) Executable statements (Table 2)

b. Syntax of embedded SQL statements

(a) Executable SQL statements are defined by prefixing embedded SQL statements with the keywords *EXEC SQL* that end with a semicolon (";")

(b) SQL types are defined by using *EXEC SQL TYPE* keyword sequences and ended with a semicolon (";")

(c) Host variables are defined using C/C++ variable declarations inside the pair of keyword sequences *EXEC SQL BEGIN DECLARE SECTION;* and *EXEC SQL END DECLARE SECTION;*

DIRECTIVE	PURPOSE
ARRAYLEN*	To use host arrays with PL/SQL
BEGIN DECLARE SECTION*	To declare host variables (optional)
END DECLARE SECTION*	
DECLARE*	To name Oracle schema objects
INCLUDE*	To copy in files
TYPE*	To equivalence datatypes
VAR*	To equivalence variables
WHENEVER*	To handle runtime errors
*Has no interactive counterpart	

Table 3: Declaration Directives

EXECUTABLE STATEMENT	PURPOSE
ALLOCATE*	To define and control Oracle data
ALTER	
ANALYZE	
DELETE	DML
INSERT	
SELECT	
UPDATE	
COMMIT	To process transactions
ROLLBACK	
SAVEPOINT	
SET TRANSACTION	
DESCRIBE*	To use dynamic SQL
EXECUTE*	
PREPARE*	
ALTER SESSION	To control sessions
SET ROLE	
*Has no interactive counterpart	

Table 4: Executable statements

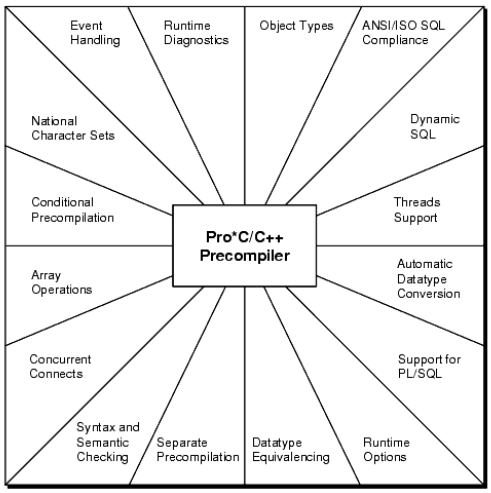

Figure 17: Embedded SQL features and benefits

c. Embedded PL/SQL blocks

d. Static vs dynamic SQL statements

   (a) When the needed information (such as the names of SQL commands, the names of tables involved, and the names of the table columns used) of an executable SQL statement is known (hence fixed) before run-time, the SQL statement is called a static SQL statement

   (b) Dynamic SQL programming allows the needed information of a SQL statement to be supplied at run-time. It also allows construction and subsequent execution of of SQL statements at run-time.

e. Host and indicator variables

   (a) Host variables, arrays, and structures.

   Host variables are used to exchange info between the host program and the referred database.

When a variable from a host language such as C/C++ is used inside an embedded SQL statement, it is called a host variable.

For convenience host arrays and structures are also allowed.

Host variables are prefixed with a colon char (":") to differentiate them from native database objects.

(b) Indicator variables, arrays, and structures

An indicator variable is associated with a corresponding host variable to indicate the value or condition of its associated host variable.

Use of indicator variables are optional. However, when a host variable receives value from a column of a table that has a NULL value, indicator variable is the only way to know that fact.

f. Oracle datatypes: Oracle recognizes two kinds of datatypes: internal datatypes and external datatypes

(a) Internal datatypes: to represent and specify how data to be stored in database columns

(b) External datatypes: to represent and specify how data to be stored in host variables

g. Private SQL areas, cursors, and active sets

(a) Before processing each SQL statement Oracle opens a work area called private SQL area that is used to store the info needed to execute the statement.

(b) A cursor is an identifier that can be associated with a SQL statement. A cursor can be used to access information in the private SQL area and can also to some degree control the processing of the SQL statement.

(c) The set of rows returned from a SQL query is called an *active set*.

(d) Implicit and explicit cursors. For static SQL statements, an implicit cursor is declared by Oracle for each SQL statement. Explicit cursors are needed if explicit control of how to fetch rows in an active set is desired.

h. Transactions

(a) A transaction is a sequence of logically related SQL statements that must be executed *atomically* – either all of the statements succeed and their effects are made permanent or none of them leave any effects.

(b) Oracle Pro*C/C++ allows use of SQL statement COMMIT, ROLLBACK, and SAVEPOINT to achieve transaction control.

i. Errors and warnings

(a) SQLCA (SQL communcation area): is a data structure that can be included in host programs. It defines program variables used by Oracle to pass runtime status info to the program. By inspecting SQLCA a host program can check the type of errors that occurred and decide the appropriate actions.

(b) WHENEVER statement: allows a host program to specify actions to be taken in response to the type of errors in given WHENEVER statement.

(2) Steps in Pro* C/C++ application development

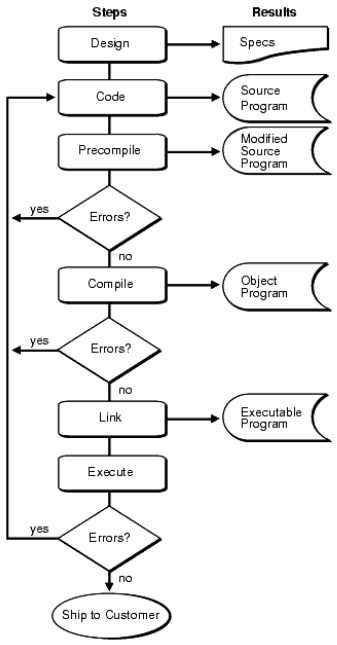

Figure 18: Embedded SQL Application Development Process

3. Using Oracle Pro*C/C++ Pre-compiler in UNIX and Windows platforms

(1) UNIX platform

a. Structure of a UNIX Oracle server

b. The samples and the *demo_proc.mk* make file

c. The file *env_precomp.mk* that sets up environments for Pro C/C++ pre-compiler and C/++ compiler

(2) Windows

a. Structure of a windows Oracle server

b. The Pre C/C++ compiler

   (a) The main window

   (b) Pre-compiler options

c. Visual C++ in Microsoft Visual Studio

   (a) The main window

   (b) Changing compiler options

   (c) Adding Oracle library to be used for linking

4. Dynamic SQL

(1) Motivations

a. Limitation of conventional embedded SQL

b. What is dynamic SQL programming?

(2) Types of dynamic SQL

a. Method 1: the program accepts or builds a dynamic SQL statement and then immediately executes it using the EXECUTE IMMEDIATE command. Restrictions:

   (a) The SQL statement must not be a query (SELECT) statement;

   (b) It is not allowed to contain any placeholders for input host variables.

   With method 1, the SQL statement is parsed everytime it is executed. Example host strings:

   'DELETE FROM EMP WHERE DEPTNO = 20'
   'GRANT SELECT ON EMP TO scott'

b. Method 2: the program accepts or builds a dynamic SQL statement and then processes it using the PREPARE and EXECUTE commands. Restrictions:

   (a) The SQL statement must not be a query (SELECT) statement;

   (b) Placeholders for input host variables are allowed. However the number of placeholders for input host variables and the datatypes of the input host variables must be known at precompiler time.

   With method 2, the SQL statement is parsed just once, but can be executed many times with different values for the host variables.

   SQL DDL statements such as CREATE and GRANT are executed when they are PREPAREd (no need to execute it). Example host strings:

'INSERT INTO EMP (ENAME, JOB) VALUES (:emp_name,:job_title)'
'DELETE FROM EMP WHERE EMPNO = :emp_number

c. Method 3: the program accepts or builds a dynamic SQL statement and then processes it using the PREPARE command with the DECLARE, OPEN, FETCH, and CLOSE cursor commands. Restrictions:

   (a) Query (SELECT) statements are allowed. However, the number of select-list items, the number of placeholders for input host variables and the datatypes of the input host variables must be known at precompiler time.

   With method 3, the SQL statement is parsed just once, but can be executed many times with different values for the host variables. Example host strings:

   'SELECT DEPTNO, MIN(SAL), MAX(SAL) FROM EMP GROUP BY DEPTNO'
   'SELECT ENAME, EMPNO FROM EMP WHERE DEPTNO = :dept_number'

d. Method 4: the program accepts or builds a dynamic SQL statement and then processes it using descriptors.

   (a) Most SQL statements are allowed.

   (b) The number of select-list items, the number of placeholders for input host variables and the datatypes of the input host variables may be unknown until run-time.

   Example host strings:

   'INSERT INTO EMP (<unknown>) VALUES (<unknown>)'
   'SELECT <unknown> FROM EMP WHERE DEPTNO = 20'

   The *<unknown>* portion is filled dynamically at run-time.

(3) Guidelines

a. All four methods require storing the dynamic SQL statement in a host string variable or in a quoted literal. Construct such host string is done in the host language.

b. Each succeeding method imposes fewer constraints on the way the method can be used, but involves deeper SQL knowledge to code. Hence, as a rule of thumb, try to use the simplest possible method.

c. With Methods 2 and 3 the number of placeholders and datatypes of input host variables must be known at precompile time.

d. Method 4 provides maximum flexibility and least restrictions. But full understanding of dynamic SQL concepts and advanced SQL data structures have to be used.

5. Examples

(1) Example 1: dynamic SQL type 1 (sample6.pc).

   a. It creates a table, inserts a row into the newly created table, and then drops the table.

   b. Notice the way the host variables *username, password, dynstmt1, dynstmt2, dynstmt3* are declared.

(2) Example 2: dynamic SQL type 2 (sample7.pc). It inserts two rows into the EMP table, and then deletes them.

(3) Example 3: dynamic SQL type 3 (sample8.pc)

   a. It retrieves the names of all employees in a given department from the EMP table.

   b. The deptno is provided as initial value. In practice it can be obtained from stdin or a file.

(4) Example 4: dynamic SQL type 4 (sample10.pc)

   a. Applicable to situations when the number of columns in the select-list or the number of placeholders for input host variables is unknown until run-time.

   b. The need of a special Oracle data structure called SQLDA.

      (a) The program has to place the info of select-list into a data structure called SQL Descriptor Area (SQLDA). It is also called a *select descriptor*. The program must issue the *DESCRIBE SELECT LIST* command to do so.

      (b) To handle the unknown number of placeholders of input variables and their data types, the host-variable list is declared in another SQLDA data structure called *bind descriptor*. The program must issue the *DESCRIBE BIND VARIABLES* command to do so.

   c. The program sample10.pc allows to enter any legal SQL statement (in regular SQL syntax, not embedded SQL syntax), which will be handled using dynamic SQL method 4.

(5) More examples of dynamic SQL method 4: see the demo program.