Stored Procedures and Triggers (PL/SQL)

Introduction

This unit describes how to use the Oracle PL/SQL language to create stored procedures and triggers. A stored procedure is an SQL command, or multiple commands that a user can execute over and over. PL/SQL has its own syntax, and is compiled and stored on the server, similar to programs from other languages. It can contain variable/parameter definitions, SQL commands, logic structures, exception handling and output commands, and is very powerful. It is in fact a complete working program. It is submitted to the system by the user. A trigger is similar to a stored procedure; the difference is a trigger is executed by the system not the user, usually in response to an update to the database. As an example, when an order is placed for an item, it would trigger a procedure to run that would decrease the quantity on hand in the item table.

Stored procedures and triggers are created in script files first, then submitted to the DBMS for compiling. If no errors are found, the DBMS creates an object on the server. This object can then be invoked whenever the user wishes, or ready for the system to invoke when needed.

Retrieving a single row

If the query in the procedure returns a single row, then no special handling is required. However, if the query returns more than row, then a special storage area called a cursor is required. We will discuss cursors later, so for now we will concentrate on a procedure whose SQL command returns a single row. The surest way to make sure a single row is returned is to search a table using the primary key.

The basic form of a stored procedure goes as follows:

```
CREATE OR REPLACE PROCEDURE procedurename (arguments, if needed) AS
        Local variable definitions, if needed
        BEGIN
                SQL statement (SELECT, INSERT, UPDATE, DELETE)
                Output statements to print messages or returned rows, if applicable
        END;
/
```

Once this procedure is created as a script file and executed, the procedure gets compiled and checked for syntax errors, and if error free, is placed in the Oracle catalog and can be ran when needed.

To call or run a cataloged procedure, do the following in the SQL window:

```
BEGIN
        procedurename
END;
/
```

The results from the procedure should appear in the results window, similar to any other SQL command. The backslash (/) at the end of procedure definitions and calls is optional in some Oracle installations, and required in others, so it's best to code it just in case.

Refer to Chapter 8 in your text book for good examples on creating and running a stored procedure.

Error Handling

As with any program, the system may throw an exception if certain situations arise. A good procedure should use exception handling to catch these problems and react accordingly. A classic example would be when the user searches for a particular value, and no rows match that condition. Oracle would set its no data found flag. From a user perspective, it would be nice to see a message displayed, such as "No rows match your search criteria", instead of the procedure "blowing up".

Update Procedures

Procedures typically contain SELECT commands to query tables and return rows. However, they can also be used to make changes to table data, using INSERT, UPDATE or DELETE commands. The syntax is very similar. The power of this type of procedure gets realized when multiple commands are issued within a single procedure. For example, an INSERT command could be followed by an UPDATE command. Or, if we realized an order was invalid, we might need to DELETE an order row, then the corresponding order line rows that go with that order.

Cursor processing for multiple row queries

If a procedure contains a SELECT statement that brings back more than a single row, special processing must be added to the procedure to handle this situation. Multiple rows returned from the query must be placed in a special memory area called a cursor. The cursor can then be opened, looped thru and closed, similar to processing a sequential file.

The process goes as follows: declare a cursor with a name that contains the SELECT command; open the cursor when ready (this actually runs the SELECT and places the rows in the cursor object); set up a loop structure, similar to an EOF loop; inside the loop, issue a FETCH command, which acts like a read statement and gets each row from the cursor; after the FETCH, code commands necessary to process each row; after the loop ends, close the cursor.

The SELECT statement in the cursor can be as simple or as complex as necessary to accomplish the task.

Using Triggers

Triggers are similar to procedures in that they are coded, compiled and stored as objects in the Oracle database that can be executed when necessary. Where they differ is that a trigger gets executed automatically by the system in response to a database operation, typically an INSERT, UPDATE or DELETE statement.

An example would be when a customer places an order for an item, we would insert a row into the orders table and a row into the order line table (or multiple rows if the order is for more than one item). In response to these inserts, it could trigger a procedure that would then do an update to the item table to deduct the number ordered from the quantity on hand. This allows the system to do what a person would have to do manually.

The basic form of a trigger goes as follows:

```
CREATE OR REPLACE TRIGGER triggername
        Command that does the triggering (AFTER INSERT ……)
        BEGIN
                Command to be performed in response
        END;
```

The NEW operator is often used to reference the column in the table that is currently being inserted or updated, and the OLD operator is used to reference the value in a column before the change takes place.

For example, when doing an order and inserting a row in the order line table for an item, we might use: SET Num_On_Hand = Num_On_Hand - :NEW.Num_Ordered
This assumes Num_On_Hand is in the Item table and Num_Ordered is in the OrderLine table. This would subtract number ordered from the current on hand.

Again, refer to Chapter 8 of the textbook for Trigger examples.
