Database Programming

5.Stored Procedures

Sagara Samarawickrama

CSD 4203 – 2024W

5.Stored Procedures - 1

All the PL/SQL that you have written up to this point has been anonymous blocks that were run as scripts and compiled by the database server at run time. Now you will begin to use modular code. Modular code is a methodology to build a program from distinct parts (modules), each of which performs a specific function or task toward the final objective of the program. Once modular code is stored on the database server, it becomes a database object, or subprogram, that is available to other program units for repeated execution. To save code into the database, the source code needs to be sent to the server so that it can be compiled into p-code and stored in the database.

Benefits of Modular Code

A PL/SQL module is any complete logical unit of work. There are five types of PL/SQL modules:

(1) anonymous blocks that are run with a text script (the type you have used until now),

(2) procedures

5.Stored Procedures - 1

(3) functions,

(4) packages,

and (5) triggers

There are two main benefits to using modular code:

(1) It is more reusable and

(2) it is more manageable

Block Structure

The same block structure is used for all the module types. The block begins with a header (for named blocks only), which consists of

(1) the name of the module and

(2) a parameter list (if used).

5.Stored Procedures - 1

The declaration section defines variables, cursors, and sub-blocks.

The main part of the module is the execution section, where all of the calculations and processing are performed. This will contain executable code such as IF-THEN-ELSE statements, loops, calls to other PL/SQL modules, and so on

The last section of the module is an optional exception handler, which contains the code to handle exceptions.

Anonymous Blocks

Until this lesson, we have written only anonymous blocks. Anonymous blocks are very much like modules, except that anonymous blocks do not have headers. There are important distinctions, though. As the name implies, anonymous blocks have no names and, therefore, cannot be called by another block. They are not stored in the database and must be compiled and then run each time the script is loaded.

5.Stored Procedures - 1

Creating Procedures

A procedure is a module performing one or more actions; it does not need to return any values. The syntax for creating a procedure is as follows

```
CREATE OR REPLACE PROCEDURE name
  [(parameter[, parameter, …])]
AS
  [local declarations]
BEGIN
  executable statements
[EXCEPTION
  exception handlers]
END [name];
```

Every procedure has two parts: (1) the header portion, which comes before the AS (or sometimes IS—they are interchangeable) keyword and contains the procedure name and the parameter list, and (2) the body, which is everything after the AS (IS) keyword. The word REPLACE is optional

5.Stored Procedures - 1

Creating Procedures

The following script demonstrates the syntax for creating a procedure. When this script is run, it creates a procedure named Discount that is compiled into p-code and stored in the database for later execution

```
CREATE OR REPLACE PROCEDURE Discount
AS
  CURSOR c_group_discount
  IS
    SELECT distinct s.course_no, c.description
      FROM section s, enrollment e, course c
     WHERE s.section_id = e.section_id
       AND c.course_no = s.course_no
     GROUP BY s.course_no, c.description,
              e.section_id, s.section_id
    HAVING COUNT(*) >=8;
BEGIN
    FOR r_group_discount IN c_group_discount
    LOOP
        UPDATE course
           SET cost = cost * .95
         WHERE course_no = r_group_discount.course_no;
        DBMS_OUTPUT.PUT_LINE
          ('A 5% discount has been given to '||
           r_group_discount.course_no||' '||
           r_group_discount.description
          );
    END LOOP;
END;
```

5.Stored Procedures - 1

To execute the stored procedure Discount, the following syntax is used:

```
EXECUTE Procedure_name
```

Executing the Discount procedure yields the following result:

```
5% discount has been given to 25  Adv. Word Perfect
…. (through each course with an enrollment over 8)
PL/SQL procedure successfully completed.
```

There is no COMMIT in this procedure, which means the procedure will not update the database. A COMMIT command needs to be issued after the procedure is run, if you want the changes to be made. Alternatively, you can enter a COMMIT command either before or after the end loop

What is the difference ? explain

5.Stored Procedures - 1

To execute the stored procedure Discount, the following syntax is used:

```
EXECUTE Procedure_name
```

Executing the Discount procedure yields the following result:

```
5% discount has been given to 25  Adv. Word Perfect
…. (through each course with an enrollment over 8)
PL/SQL procedure successfully completed.
```

There is no COMMIT in this procedure, which means the procedure will not update the database. A COMMIT command needs to be issued after the procedure is run, if you want the changes to be made. Alternatively, you can enter a COMMIT command either before or after the end loop

What is the difference ? explain

5.Stored Procedures - 1

Querying the Data Dictionary for Information on Procedures

Two main views in the data dictionary provide information on stored code: the USER_OBJECTS view, which gives information about the objects, and the USER_SOURCE view, which gives the text of the source code. The data dictionary also has ALL_ and DBA_ versions of these views.

The following SELECT statement gets pertinent information from the USER_OBJECTS view about the Discount procedure you just wrote:

```
SELECT object_name, object_type, status
  FROM user_objects
 WHERE object_name = 'DISCOUNT';
```

The result would be the following, assuming the only object in the database is the new Discount procedure:

```
OBJECT_NAME      OBJECT_TYPE    STATUS
------ -----  --
DISCOUNT         PROCEDURE      VALID
```

The status indicates where the procedure was compiled successfully. An invalid procedure cannot be executed.

5.Stored Procedures - 1

The following SELECT statement displays the source code from the USER_SOURCE view for the Discount procedure:

```
SELECT TO_CHAR(line, 99)||'>', text
  FROM user_source
 WHERE name = 'DISCOUNT'
```

Stored procedures in the database can also be seen in Oracle SQL Developer. If you expand the nodes under the appropriate database connection, you will see under the Procedure node all procedures in the database for the user specified in the database connection. The node will show both valid and invalid procedures.

See the picture in the next slide →

5.Stored Procedures - 1

Oracle SQL Developer : Procedure STUDENT.DISCOUNT@student

File Edit View Navigate Run Versioning Tools Help

Connections | Rep... | DISCOUNT

Code | Grants | Dependencies | References | Errors | Details | Profiles

- Connections
  - student
    - Tables (Filtered)
    - Views
    - Editioning Views
    - Indexes
    - Packages
    - Procedures
      - DISCOUNT
    - Functions
    - Queues
    - Queues Tables
    - Triggers
    - Crossedition Triggers
    - Types

```
create or replace
PROCEDURE Discount
AS
  CURSOR c_group_discount
  IS
    SELECT distinct s.course_no, c.description
      FROM section s, enrollment e, course c
     WHERE s.section_id = e.section_id
       AND c.course_no = s.course_no
     GROUP BY s.course_no, c.description,
              e.section_id, s.section_id
     HAVING COUNT(*) >=8;
BEGIN
  FOR r_group_discount IN c_group_discount
  LOOP
```

5.Stored Procedures - 1

Lets consider a example:

We will try to insert a record to the following table using stored procedure

First Create the following table structure

Welcome Page | Exconn | CPCM_STUDENT

0.054 seconds

Worksheet | Query Builder

```
CREATE TABLE CPCM_STUDENT (
  STUDENT_ID      NUMBER (5)    NOT NULL,
  STUDENT_NAME    VARCHAR2 (20) NOT NULL,
  CREATED_BY      VARCHAR2 (20) NOT NULL,
  CREATED_DATE    DATE          NOT NULL,
  PRIMARY KEY ( STUDENT_ID )
);
```

Next we have to create a stored procedure to insert data to the above table

Explain the steps in the class

5.Stored Procedures - 1

```
CREATE or REPLACE PROCEDURE insertSTUDENT(
    p_studentid IN CPCM_STUDENT.STUDENT_ID%TYPE,
    p_studentname IN CPCM_STUDENT.STUDENT_NAME%TYPE,
    p_createdby IN CPCM_STUDENT.CREATED_BY%TYPE,
    p_date IN CPCM_STUDENT.CREATED_DATE%TYPE)
IS
BEGIN
 INSERT INTO CPCM_STUDENT ("STUDENT_ID", "STUDENT_NAME", "CREATED_BY", "CREATED_DATE")
 VALUES (p_studentid, p_studentname,p_createdby, p_date);

COMMIT;

END;
```

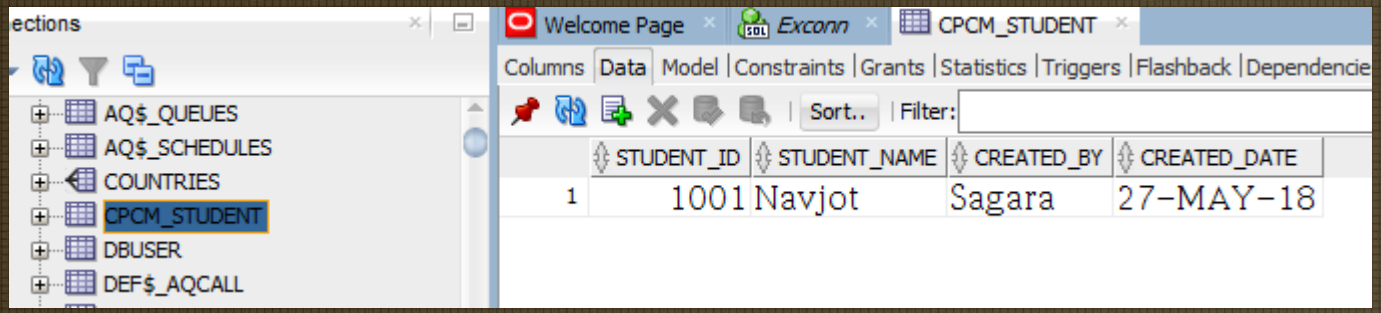
Now call the stored procedure

Welcome Page | Excon | CPCM_STUDENT

Worksheet | Query Builder

```
1  BEGIN
2    insertSTUDENT(1001,'Navjot','Sagara',SYSDATE);
3  END;
```

5.Stored Procedures - 1



Calling the stored procedure with the given parameters will insert the above record to the CPCM_STUDENT table.

Study and practice the steps for creating and calling a stored procedure

Do the lab activities uploaded for this topic

5.Stored Procedures - 1

Lets consider another example:

We will try to select a record from a table based on the given parameter

We use the same table we used in the previous example

Now create the following stored procedure

Worksheet | Query Builder

```
CREATE OR REPLACE PROCEDURE getDBUSERByUserId(
      p_studentid IN CPCM_STUDENT.STUDENT_ID%TYPE,
      o_studentname OUT CPCM_STUDENT.STUDENT_NAME%TYPE,
      o_createdby OUT  CPCM_STUDENT.CREATED_BY%TYPE,
      o_date OUT CPCM_STUDENT.CREATED_DATE%TYPE)
IS
BEGIN

  SELECT STUDENT_NAME , CREATED_BY, CREATED_DATE
  INTO o_studentname, o_createdby,  o_date
  from  CPCM_STUDENT WHERE STUDENT_ID = p_studentid;

END;
```

5.Stored Procedures - 1

Now Call the stored procedure

Welcome Page | Exconn | CPCM_STUDENT

Worksheet | Query Builder

```
DECLARE
   o_studentname CPCM_STUDENT.STUDENT_NAME%TYPE;
   o_createdby DBUSER.CREATED_BY%TYPE;
   o_date DBUSER.CREATED_DATE%TYPE;
BEGIN

   getDBUSERByUserId(1001,o_studentname,o_createdby,o_date);

   DBMS_OUTPUT.PUT_LINE('username :  ' || o_studentname);
   DBMS_OUTPUT.PUT_LINE('createdby :  ' || o_createdby);
   DBMS_OUTPUT.PUT_LINE('createddate :  ' || o_date);

END;
```

It will retrieve the record of the student with the student id 1001

5.Stored Procedures - 1

It will display the following out put

Script Output | Script Output 1

Task completed in 0.037 seconds

```
username :  Navjot
createdby :  Sagara
createddate :  27-MAY-18


PL/SQL procedure successfully completed.
```

Next Week :

Passing Parameters IN and OUT of Procedures