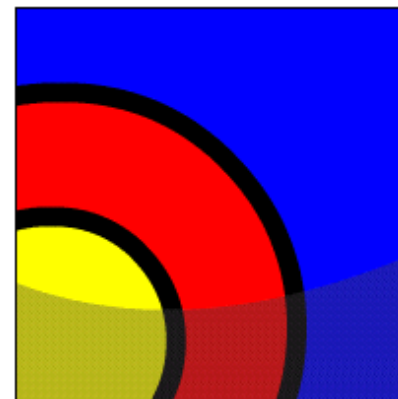


Creating Procedures

What Will I Learn?

In this lesson, you will learn to:

- Differentiate between anonymous blocks and subprograms
- Identify the benefits of subprograms
- Define a stored procedure
- Create a procedure
- Describe how a stored procedure is invoked
- List the development steps for creating a procedure





Why Learn It?

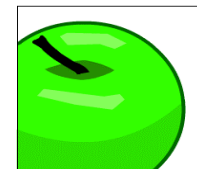
Up to now in this course, you have learned how to write and execute anonymous PL/SQL blocks. Anonymous blocks are written as part of the application program.

In this and the next two sections, you will learn how to create, execute, and manage PL/SQL subprograms. These are stored in the database, giving many benefits, such as shareability, better security, and faster performance.

There are two kinds of PL/SQL subprograms: procedures and functions. In this lesson, you learn how to create and execute stored procedures.



Tell Me / Show Me



Differences Between Anonymous Blocks and Subprograms

Anonymous Blocks

The only kind of PL/SQL blocks that have been introduced in this course so far are anonymous blocks. As the word “anonymous” indicates, anonymous blocks are unnamed executable PL/SQL blocks. Because they are unnamed, they can neither be reused nor stored in the database for later use.

While you can store anonymous blocks on your PC, the database is not aware of them, so no one else can share them.



Tell Me / Show Me

Differences Between Anonymous Blocks and Subprograms

Subprograms

Procedures and functions are named PL/SQL blocks. They are also known as subprograms. These subprograms are compiled and stored in the database. The block structure of the subprograms is similar to the structure of anonymous blocks.

While subprograms can be explicitly shared, the default is to make them private to the owner's schema.

Later subprograms become the building blocks of packages and triggers.



Tell Me / Show Me

Differences Between Anonymous Blocks and Subprograms

Anonymous Blocks

```
DECLARE      (Optional)
    Variables, cursors, etc.;
BEGIN        (Mandatory)
    SQL and PL/SQL statements;
EXCEPTION   (Optional)
    WHEN exception-handling actions;
END;         (Mandatory)
```

Subprograms (Procedures)

```
CREATE [OR REPLACE] PROCEDURE name [parameters] IS|AS
(Mandatory)
    Variables, cursors, etc.; (Optional)
BEGIN      (Mandatory)
    SQL and PL/SQL statements;
EXCEPTION (Optional)
    WHEN exception-handling actions;
END [name]; (Mandatory)
```

 **Tell Me / Show Me****Differences Between Anonymous Blocks and Subprograms
(continued)**

Anonymous Blocks	Subprograms
Unnamed PL/SQL blocks	Named PL/SQL blocks
Compiled on every execution	Compiled only once, when created
Not stored in the database	Stored in the database
Cannot be invoked by other applications	They are named and therefore can be invoked by other applications
Do not return values	Subprograms called functions must return values
Cannot take parameters	Can take parameters



Tell Me / Show Me

Benefits of Subprograms

Procedures and functions have many benefits due to the modularizing of the code:

- Easy maintenance: Modifications need only be done once to improve multiple applications and minimize testing.
- Code reuse: Subprograms are located in one place. When compiled and validated, they can be used and reused in any number of applications.



Tell Me / Show Me

Benefits of Subprograms (continued)

- Improved data security: Indirect access to database objects is permitted by the granting of security privileges on the subprograms. By default, subprograms run with the privileges of the subprogram owner, not the privileges of the user.
- Data integrity: Related actions can be grouped into a block and are performed together (“Statement Processed”) or not at all.



Tell Me / Show Me

Benefits of Subprograms (Continued)

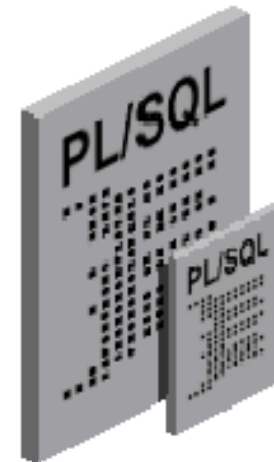
- Improved performance: You can reuse compiled PL/SQL code that is stored in the shared SQL area cache of the server. Subsequent calls to the subprogram avoid compiling the code again. Also, many users can share a single copy of the subprogram code in memory.
- Improved code clarity: By using appropriate names and conventions to describe the action of the routines, you can reduce the need for comments, and enhance the clarity of the code.

Tell Me / Show Me

Procedures and Functions

- Are named PL/SQL blocks
- Are called PL/SQL subprograms
- Have block structures similar to anonymous blocks:
 - Optional parameters
 - Optional declarative section (but the `DECLARE` keyword changes to `IS` or `AS`)
 - Mandatory executable section
 - Optional section to handle exceptions

This section focuses on procedures.





Tell Me / Show Me

What Is a Procedure?

- A procedure is a named PL/SQL block that can accept parameters.
- Generally, you use a procedure to perform an action (sometimes called a “side-effect”).
- A procedure is compiled and stored in the database as a schema object.
 - Shows up in `USER_OBJECTS` as an object type of `PROCEDURE`
 - More details in `USER_PROCEEDURES`
 - Detailed PL/SQL code in `USER_SOURCE`



Tell Me / Show Me

Syntax for Creating Procedures

```
CREATE [OR REPLACE] PROCEDURE procedure_name
  [(parameter1 [mode1] datatype1,
    parameter2 [mode2] datatype2,
    . . .)]
IS|AS
procedure_body;
```

- Parameters are optional
- Mode defaults to `IN`
- Datatype can be either explicit (for example, `VARCHAR2`) or implicit with `%TYPE`
- Body is the same as an anonymous block

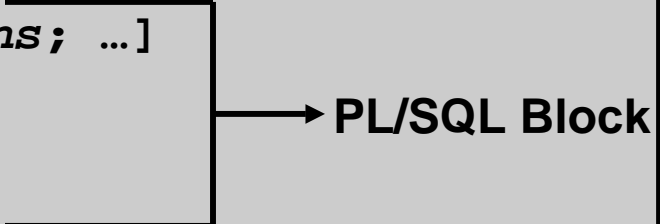


Tell Me / Show Me

Syntax for Creating Procedures (continued)

- Use `CREATE PROCEDURE` followed by the name, optional parameters, and keyword `IS` or `AS`.
- Add the `OR REPLACE` option to overwrite an existing procedure.
- Write a PL/SQL block containing local variables, a `BEGIN`, and an `END` (or `END procedure_name`).

```
CREATE [OR REPLACE] PROCEDURE procedure_name
  [(parameter1 [mode] datatype1,
    parameter2 [mode] datatype2, ...)]
IS|AS
  [local_variable_declarations; ...]
BEGIN
  -- actions;
END [procedure_name];
```





Tell Me / Show Me

Procedure: Example

In the following example, the `add_dept` procedure inserts a new department with the `department_id` 280 and `department_name` ST-Curriculum. The procedure declares two variables, `v_dept_id` and `v_dept_name`, in the declarative section.

```
CREATE TABLE dept AS SELECT * FROM departments;
CREATE OR REPLACE PROCEDURE add_dept IS
    v_dept_id      dept.department_id%TYPE;
    v_dept_name     dept.department_name%TYPE;
BEGIN
    v_dept_id      :=280;
    v_dept_name    := 'ST-Curriculum';
    INSERT INTO dept(department_id,department_name)
        VALUES(v_dept_id,v_dept_name);
    DBMS_OUTPUT.PUT_LINE('Inserted ' || SQL%ROWCOUNT || 'row');
END;
```



Tell Me / Show Me

Procedure: Example (continued)

The declarative section of a procedure starts immediately after the procedure declaration and does not begin with the keyword `DECLARE`. This procedure uses the `SQL%ROWCOUNT` cursor attribute to check if the row was successfully inserted. `SQL%ROWCOUNT` should return 1 in this case.

```
CREATE TABLE dept AS SELECT * FROM departments;
CREATE OR REPLACE PROCEDURE add_dept IS
    v_dept_id      dept.department_id%TYPE;
    v_dept_name     dept.department_name%TYPE;
BEGIN
    v_dept_id      :=280;
    v_dept_name     := 'ST-Curriculum';
    INSERT INTO dept(department_id,department_name)
        VALUES(v_dept_id,v_dept_name);
    DBMS_OUTPUT.PUT_LINE('Inserted ' || SQL%ROWCOUNT || 'row');
END;
```



Tell Me / Show Me

Invoking Procedures

You can invoke (execute) a procedure from:

- An anonymous block
- Another procedure
- A calling application

Note: You CANNOT invoke a procedure from inside a SQL statement such as `SELECT`.



Tell Me / Show Me

Invoking the Procedure from Application Express

To invoke (execute) a procedure in Oracle Application Express, write and run a small anonymous block that invokes the procedure. For example:

```
CREATE OR REPLACE PROCEDURE add_dept IS ...  
  
BEGIN  
    add_dept;  
END;  
  
SELECT department_id, department_name FROM dept  
WHERE department_id=280;
```

The select statement at the end confirms that the row was successfully inserted.



Tell Me / Show Me

Correcting Errors in `CREATE PROCEDURE` Statements

If compilation errors exist, Application Express displays them in the output portion of the SQL Commands window. You must edit the source code to make corrections. The procedure is still created even though it contains errors.

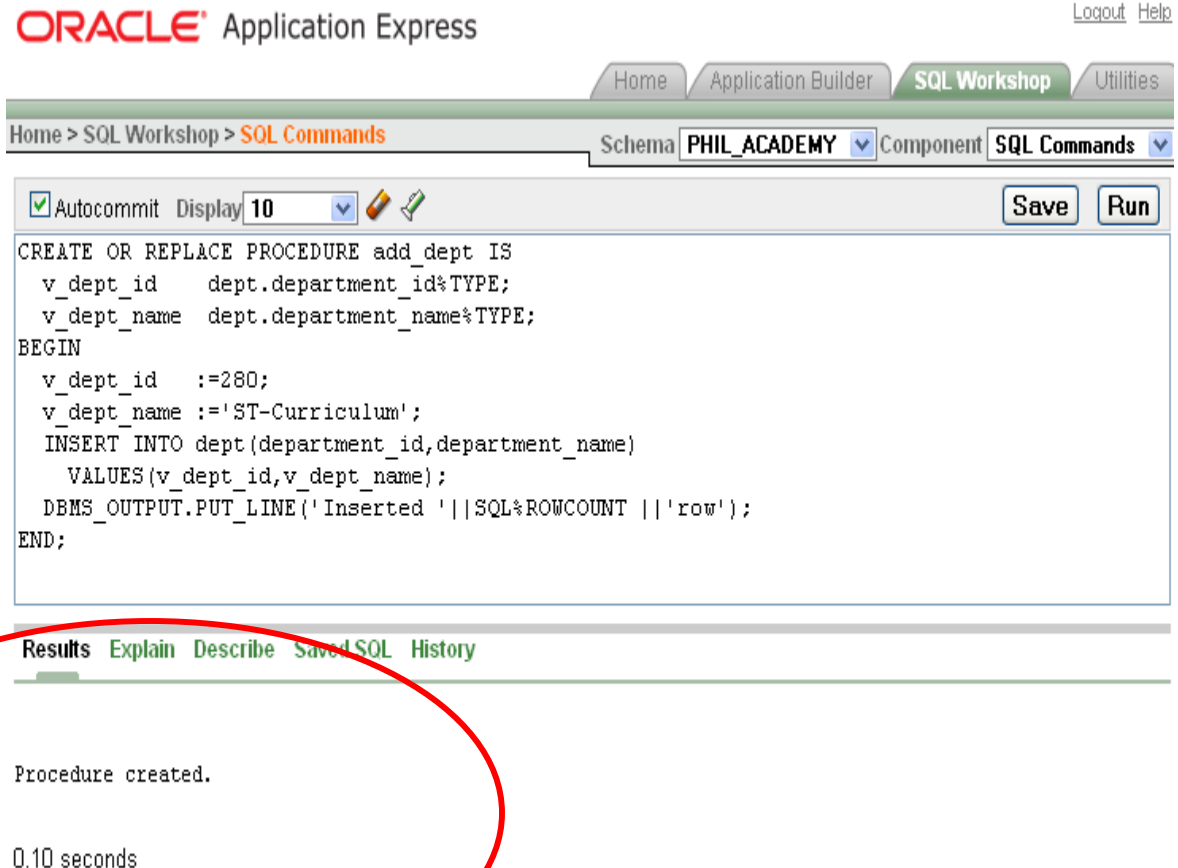
After you have corrected the error in the code, you need to recreate the procedure. There are two ways to do this:

- Use a `CREATE` or `REPLACE PROCEDURE` statement to overwrite the existing code (most common)
- `DROP` the procedure first and then execute the `CREATE PROCEDURE` statement (less common).

Tell Me / Show Me

Saving Your Work

Once a procedure has been created successfully, you should save its definition in case you need to modify the code later.



The screenshot displays the Oracle Application Express SQL Workshop interface. The breadcrumb navigation shows 'Home > SQL Workshop > SQL Commands'. The 'Schema' dropdown is set to 'PHIL_ACADEMY' and the 'Component' dropdown is set to 'SQL Commands'. The 'Autocommit' checkbox is checked, and the 'Display' value is 10. The 'Save' and 'Run' buttons are visible. The SQL editor contains the following code:

```
CREATE OR REPLACE PROCEDURE add_dept IS
  v_dept_id    dept.department_id%TYPE;
  v_dept_name  dept.department_name%TYPE;
BEGIN
  v_dept_id    :=280;
  v_dept_name  := 'ST-Curriculum';
  INSERT INTO dept(department_id,department_name)
    VALUES(v_dept_id,v_dept_name);
  DBMS_OUTPUT.PUT_LINE('Inserted ' || SQL%ROWCOUNT || 'row');
END;
```

Below the editor, the 'Results' tab is selected, showing the output:

```
Procedure created.

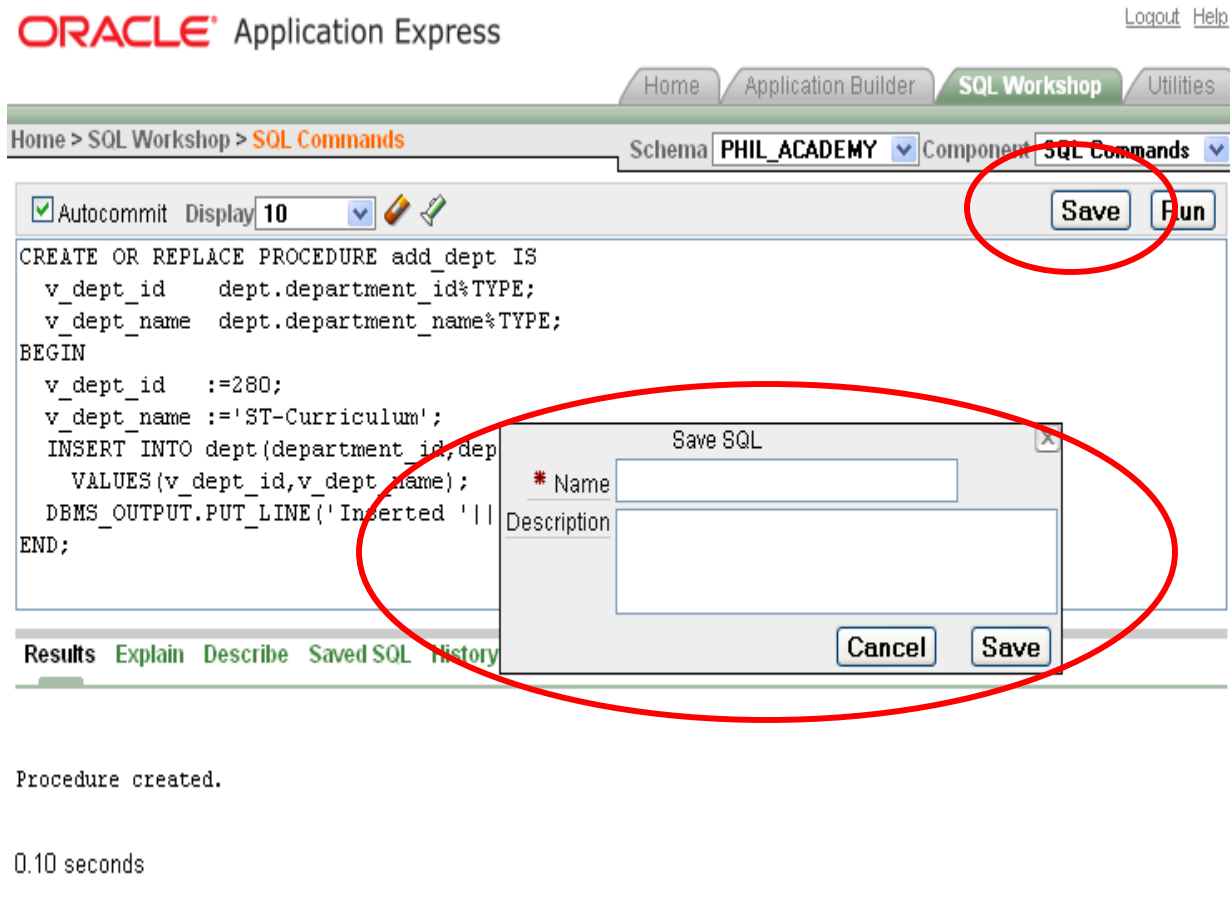
0.10 seconds
```

A red oval highlights the 'Results' section, indicating the successful execution of the procedure.

Tell Me / Show Me

Saving Your Work

In the Application Express SQL Commands window, click the SAVE button and enter a name and optional description for your code.



ORACLE Application Express

Home > SQL Workshop > SQL Commands

Schema: PHIL_ACADEMY Component: SQL Commands

Autocommit: ☒ Display: 10

Save Run

```
CREATE OR REPLACE PROCEDURE add_dept IS
  v_dept_id   dept.department_id%TYPE;
  v_dept_name dept.department_name%TYPE;
BEGIN
  v_dept_id   :=280;
  v_dept_name := 'ST-Curriculum';
  INSERT INTO dept(department_id,dept
    VALUES(v_dept_id,v_dept_name);
  DBMS_OUTPUT.PUT_LINE('Inserted ' ||
END;
```

Save SQL

* Name

Description

Cancel Save

Results Explain Describe Saved SQL History

Procedure created.

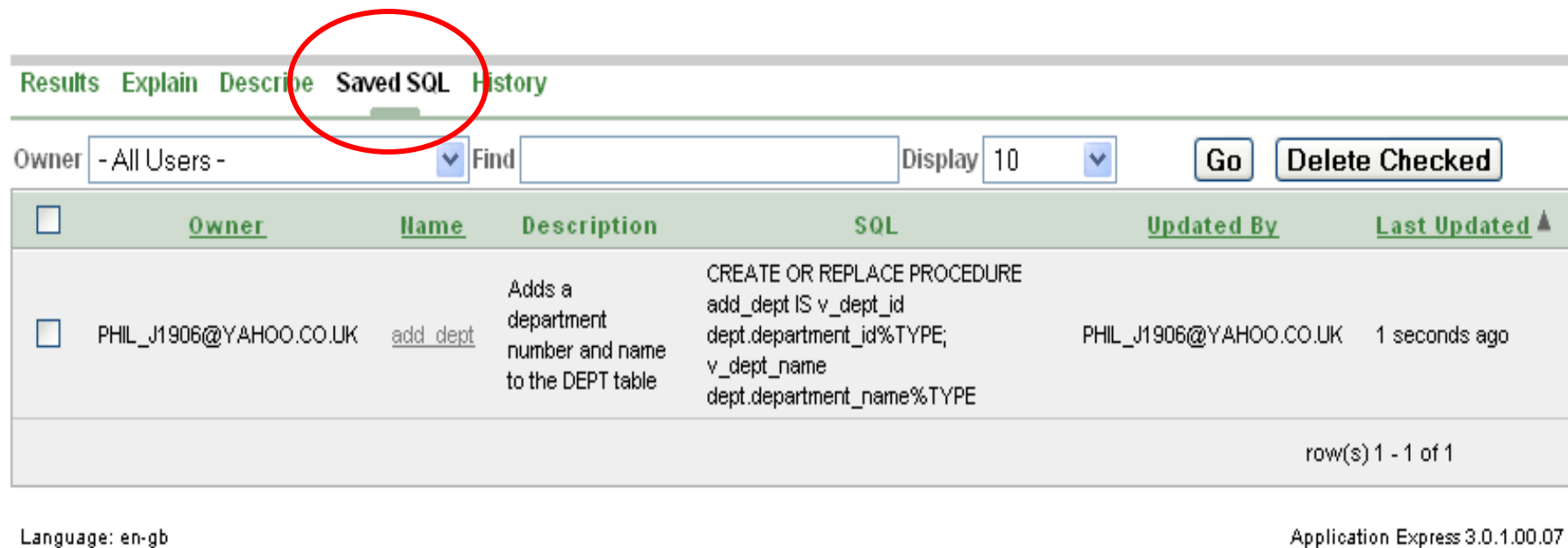
0.10 seconds



Tell Me / Show Me

Saving Your Work

You can view and reload your code later by clicking on the Saved SQL button in the SQL Commands window.



Results Explain Describe **Saved SQL** History

Owner: - All Users - Find: Display: 10 Go Delete Checked

<input type="checkbox"/>	Owner	Name	Description	SQL	Updated By	Last Updated
<input type="checkbox"/>	PHIL_J1906@YAHOO.CO.UK	add_dept	Adds a department number and name to the DEPT table	CREATE OR REPLACE PROCEDURE add_dept IS v_dept_id dept.department_id%TYPE; v_dept_name dept.department_name%TYPE	PHIL_J1906@YAHOO.CO.UK	1 seconds ago

row(s) 1 - 1 of 1

Language: en-gb Application Express 3.0.1.00.07

Tell Me / Show Me

Alternative Tools for Developing Procedures

If you end up writing PL/SQL procedures for a living, there are other free tools that can make this process easier. For instance, Oracle tools, such as SQL Developer and JDeveloper assist you by:

- Color-coding **commands** vs **variables** vs **constants**
- Highlighting matched and mismatched **((parentheses)**
- Displaying errors more graphically
- Enhancing code with standard indentations and capitalization
- Completing commands when typing
- Completing column names from tables



Tell Me / Show Me

Terminology

Key terms used in this lesson include:

Anonymous blocks

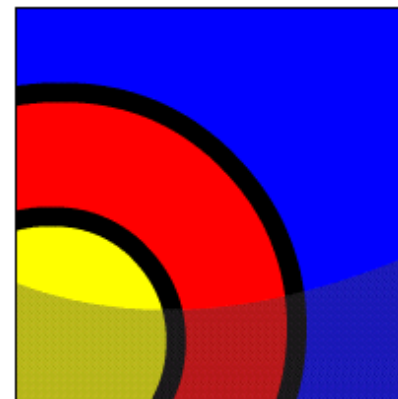
Subprograms

Procedures

Summary

In this lesson, you learned to:

- Differentiate between anonymous blocks and subprograms
- Identify the benefits of subprograms
- Define a stored procedure
- Create a procedure
- Describe how a stored procedure is invoked
- List the development steps for creating a procedure





Try It / Solve It

The exercises in this lesson cover the following topics:

- Differentiating between anonymous blocks and subprograms
- Identifying the benefits of subprograms
- Defining a stored procedure
- Creating a procedure
- Describing how a stored procedure is invoked
- Listing the development steps for creating a procedure

