Debugging an SQL Stored Procedure

Version 2

Contents

Before You Begin

Welcome to the **Debugging an SQL stored procedure** tutorial.

Objective:

This tutorial's objective is to guide you through the process of debugging an SQL stored procedure using the integrated SQL debugger that is provided with DB2® Development Center. After you have worked your way through this brief tutorial, you should have the basic understanding required to debug your own stored procedures.

Duration:

This tutorial should take 30 to 60 minutes for you to complete.

Prerequisites:

You need to have the following prerequisites to complete this tutorial:

- Development Center version 8.2 or later
- DB2 Universal Database™ for Linux™, Unix, and Windows® version 8.2 or later (Other DB2 servers will work, but they will require different options)

Overview

The `loop.db2` stored procedure used in this tutorial accesses the SAMPLE database. This procedure fetches rows from the EMPLOYEE table, searching for the first occurrence of an employee without a middle initial

In this stored procedure, the output variable counter is defined to return the index of the first row with no middle initial

This tutorial guides you through the steps required to import, build, and debug the `loop.db2` stored procedure. Follow the eight steps in order because later steps depend on tasks completed in earlier steps.

Conventions used in this tutorial

This tutorial uses typographical conventions in the text to help you distinguish between the names of controls and text that you type. For example:

- Menu items are in boldface font:

  Click **Menu —> Menu choice**.
- Text that you type is in example font on a new line:

  ```
  This is the text that you type.
  ```
- File or directory names are in example font. Bold italics indicates customizable text:

  ...\ProgramFiles\SQLLIB\spb\projects***example**.spp.

Step 1: Verify the status of the DB2 KEEPFENCED keyword

For stored procedures, the DB2 keyword KEEPFENCED has the default value `yes`. This keeps the stored procedure process alive. The setting KEEPFENCED=″yes″ is required to debug SQL stored procedures.

Note: This keyword was known as KEEPDARI in previous versions of DB2 Universal Database.

To verify the value of the KEEPFENCED keyword:

1. From a DB2 Command Line Prompt window, type `get dbm cfg`.
2. Scroll down to the KEEPFENCED keyword to verify its setting.
3. To update the value of the KEEPFENCED keyword, type `update dbm cfg using KEEPFENCED yes`. The database must be restarted for this command to take effect.

For more information on the DB2 KEEPFENCED keyword, see the DB2 Information Center topic called *Updating the database manager configuration file*.

Step 2. Create a new Development Center project and a connection to the SAMPLE database

To create a new Development Center project and a connection to the SAMPLE database:

1. Start Development Center.
2. Click **Project —>New Project** from the main menu.
3. In the window that opens, specify a name for the new project (for example, `debug_tutorial`), and click **OK**. Your project appears in the Project view.
4. Create a connection to the SAMPLE database:
   a. Right click the **Database Connections** folder under your project, and click **Add Connection**.
   b. On the Connection Type page of the Add Database Connection wizard, select **Online** and click **Next**.
   c. On the Connection page, complete the following steps:
      1) Select SAMPLE from the **Database** list.
      2) Select the **Use your current user ID and password** check box, or enter a valid user ID and password in the appropriate fields.
   d. Click **Finish**. The connection appears in the Project view.

Step 3. Specify debug options

Before you create and build the SQL stored procedure for this tutorial, you can specify debug options that are used for all SQL stored procedures on the DB2 instance you are connected to.

To set up the debug options:

1. Highlight the project you just created, and click **Project —> Environment Settings** from the Development Center main menu.
2. Expand the **Environment Settings** folder, and click **Debugger**.
3. Keep the default value of 60 seconds in the **Time-out in seconds** field. This sets the timeout number of seconds before stored procedures run to completion if there is no user action.
4. Select the **Log debugger flows on the server** check box. This option saves communications between the debugger client and the server to log files on the server. These files are named `db2psmds.log` and `db2psmdr.log` on the server.

   Note: If you are debugging a DB2 UDB for z/OS® stored procedure, you can also modify z/OS debug settings by expanding the **Build options** folder, clicking **SQL**, clicking the **Specify settings for debugging** check box, and modifying settings as needed.
5. Click **OK** to save your changes and close the Environment Settings window.

Step 4. Import a sample SQL stored procedure

To import the `loop.db2` stored procedure:

1. Right-click the **Stored Procedures** folder in the project tree view, and click **Import**.
2. In the Import window that opens, select **File system** and select **Source file**, then click **OK**. The Import wizard opens.
3. On the Source File page, click the ellipsis [...] button.
4. Navigate to the *DB2Install*`\SQLLIB\samples\sqlproc\loop.db2` file, and click **Choose**.
5. In the Import wizard, click **Next**.

   On the Entry Points page, you can view the source code for the stored procedure in the **Source** field. This procedure fetches rows from the EMPLOYEE table, searching for the first occurrence of an employee without a middle initial.
6. Click **Next**.

   On the Parameters page, you can view information about the stored procedure parameters. For this stored procedure, the output variable counter is defined to return the index of the first row with no middle initial.
7. Click **Next**.

   On the Name page, you can view the name of the stored procedure.
8. Click **Next**.
9. On the Options page, clear the **Build** check box, and click **Finish**.

   The stored procedure appears in the **Stored Procedures** folder of your project.

Step 5: Build the stored procedure for debug

To build the stored procedure for debug:

1. Right-click the stored procedure in the Project view.
2. Click **Build for Debug.**

Step 6: Set and modify line and variable break points

Using the Development Center's SQL debugging support, you can modify breakpoints and variables, or change the view to see the state at different points on the call stack. The editor window highlights the current line. Line breakpoints cause execution to pause when the indicated line is reached. Line breakpoints can be set, modified, or removed by using either the editor window, the breakpoint window on the debug tab, the toolbar, or the Debug menu items.

To set a line break point:

1. Double-click the stored procedure you created to open it in the editor window.
2. Place the cursor next to a line in the editor window.
3. Double-click to set a line break point at the desired line, click the **Add breakpoint** toolbar icon, or select **Debug –> Breakpoint > Add Breakpoint**.

   Note: Line breakpoints are only valid on the first line of a statement.

To enable, disable, or remove a line break point: Click the break point icon in the editor window, place the cursor at the desired line in the editor window and use the toolbar icons, or click **Debug –> Breakpoint —> Toggle Breakpoint** or **Debug —> Remove Breakpoint**.

Variable breakpoints cause execution to pause when the indicated variables value changes. Variable breakpoints can be set, modified, or removed by using either the variables or breakpoint window on the debug tab, the toolbar, or the debug menu items.

Note: Variable breakpoints can only be set or modified while you are in a debug session.

To set a variable breakpoint: Select the variable in the variable area of the debug tab, and use either the toolbar or the debug menu. Or, you can double-click in the cell next to the variable.

Step 7: Run in debug mode

To start your debugging session:

1. In the Project view, click the SQL stored procedure that you imported.
2. Click the **Debug** toolbar button icon or, in the editor, click **Selected —> Debug**.
3. In the editor, click **Debug —> Command —> Run in Debug**.

   Your debug session begins. If there are any breakpoints set, execution pauses at the first breakpoint. If there are no breakpoints set, execution stops at the first line.

   Note: You can also start the debugging session by using a **Step Into** command. This starts your debugging session and pauses execution at the first line. Select **Debug —> Step Into**, or click the toolbar icon.

Toolbar and menu items allow you to control the execution of your debugging session. You can pause your session, continue execution or run to completion, ignoring all breakpoints. You can control the flow by selecting a variety of ways to step through your code including, **Step Into**, **Step Over**, **Step Out**, **Step Return**, and **Step to Cursor**. These commands can be invoked through the Debug menu or from the toolbar.

Stepping Commands

Step Into/Step Over
: If there are no call statements in the stored procedure, these two commands both step to the next line in the code. If there is a call statement in the stored procedure, **Step Into** steps to the first line in the nested call, and **Step Over** steps to the first line after the call statement in the main procedure.

Step Return
: If you are in a nested call, this command steps back to the line following the nested call in the main procedure. If you are not in a nested call, this command takes you out of the session.

Step to Cursor
: This command takes you to the line where the cursor is placed. Using this command allows you to stop execution at a particular line without setting a line breakpoint.

To run your debug session to completion ignoring breakpoints: click **Debug —> Command —> Run to Completion**, or click the toolbar icon.

To pause your debug session: click **Debug –> Command —> Pause Debug**, or click the toolbar icon.

Step 8: Revert back to release mode by compiling in release (non-debug) mode

To revert back to release mode:

1. Select the stored procedure in the project tree view.
2. Click either the **Build** toolbar button icon or **Selected —> Build.**
3. Click either the **Run** toolbar button icon, or **Selected —> Run.**

Summary

Congratulations! You have finished the tutorial.

By going through this tutorial you completed the following tasks:

- You verified and updated the status of the DB2 KEEPFENCED keyword.
- You created a new Development Center project, specified build options, and imported a sample SQL stored procedure.
- You built the stored procedure for debug and ran the stored procedure in debug mode, setting and modifying line and variable breakpoints.
- You reverted back to release mode by compiling in release (non-debug) mode.

Additional information

The following resources can provide more information on this topic:

- *DB2 Information Center* . This online information center has detailed information about using IBM® DB2 Development Center to debug SQL stored procedures. It also has DB2 SQL Reference and application development information.

  http://publib.boulder.ibm.com/infocenter/db2help/index.jsp
- DB2 Integrated Development Add-ins Web site

  http://www.ibm.com/software/data/db2/udb/ide/
- Application development with DB2 Universal Database

  http://www.ibm.com/software/data/db2/udb/ad/index.html