IBM

Data Studio and DB2 for z/OS Stored Procedures

- Understand IBM Data Studio V2.2.1 packaging
- Use Data Studio with DB2 10 for z/OS stored procedures
- Take advantage of the Unified Debugger

Paolo Bruni
Marichu Scanlon

ibm.com/redbooks

Redpaper

International Technical Support Organization

Data Studio and DB2 for z/OS Stored Procedures

March 2011

REDP-4717-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (March 2011)

This edition applies to IBM DB2 Version 10.1 for z/OS (program number 5605-DB2), IBM InfoSphere Optim Development Studio Version 2.2.1 (program number 5724-X83), and IBM Data Studio Version 2.2.1, available from: http://www.ibm.com/developerworks/downloads/im/data/index.html

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
DataPower®
DB2 Connect™
DB2®
developerWorks®
DRDA®
IBM®
ILOG®
Informix®
InfoSphere™
iSeries®
Language Environment®
MVS™
Optim™
OS/390®
Passport Advantage®
pureScale™
QMF™
Query Management Facility™
RACF®
Rational®
Redbooks®
Redpaper™
Redbooks (logo) ®
System z®
System/390®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

Stored procedures can provide major benefits in the areas of application performance, code re-use, security, and integrity. DB2® has offered ever-improving support for developing and operating stored procedures.

This IBM® Redpaper™ publication is devoted to tools that can be used for accelerating the development and debugging process, in particular to the stored procedure support provided by the latest and fastest evolving IBM product: Data Studio.

We discuss topics related to handling stored procedures across different platforms. We concentrate on how to use tools for deployment of stored procedures on z/OS®, but most considerations apply to the other members of the DB2 family.

This paper is a major update of Part 6, “Cool tools for an easier life,” of the IBM Redbooks® publication *DB2 9 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7604.

The team who wrote this paper

This paper was produced by a team of specialists working at the Silicon Valley Lab, San Jose, California.

Paolo Bruni is a DB2 Information Management Project Leader at the International Technical Support Organization based in the Silicon Valley Lab. He has authored several Redbooks publications about DB2 for z/OS and related tools and has conducted workshops and seminars worldwide.

Marichu Scanlon is an Advisory Software Engineer for IBM Application Development Tooling organization, working at the Silicon Valley Laboratory in San Jose, CA, and is a member of the development team for IBM Data Studio. She has over 28 years of experience in the application and software development field. She holds a degree in electrical engineering from the University of the Philippines and an MBA from Ateneo University, Philippines. Her areas of expertise include stored procedures and application development tooling in all platforms. She has given presentations and demonstrations on application tooling at IDUG, regional DB2 Users Groups, and Information Management conferences. She has also co-authored IBM Redbooks publications and written several articles in DeveloperWorks on application tooling.

Thanks to the following people for their contributions to this project:

Moira Casey
Clifford Chu
Gary Lazzotti
Hung P Le
Lakshman Sakaray
Mark Taylor
Emily Zhang
Ruiming Zhou
IBM Data Studio, SVL

Emma Jacobs
IBM ITSO

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks
- Send your comments in an email to:

  redbooks@us.ibm.com
- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806
- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm
- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

The IBM Data Studio V2.2.1

IBM Data Studio is a tool that simplifies database development and administration for IBM data servers. In this chapter we provide information about the packaging, functions, and setup of the tool for use with DB2 for z/OS.

This chapter contains the following:

- Introduction
- Understanding the Data Studio packaging
- Eclipse Workbench common terminology
- Installation, configuration, and setup
- Navigating through the Data Studio workspace

1.1 Introduction

In DB2 V7.2 for Linux®, UNIX®, and Windows®, IBM introduced tooling support for stored procedures via the Stored Procedure Builder product. IBM enhanced this tooling with the follow-on tool, Development Center, in DB2 V8.1 for Linux, UNIX, and Windows. With DB2 9 for Linux, UNIX, and Windows, the Developer Workbench was introduced. Developer Workbench is based on Eclipse technology. The stored procedure tooling for DB2 databases is also consistent with the tooling delivered in WebSphere® Application Developer and Rational® Application Developer. On October 30, 2007, IBM announced Data Studio, which builds upon the tooling support of Developer Workbench and other IBM tooling products.

As of the publication of this paper, there are three products referred to as Data Studio (Table 1-1).

Table 1-1 Data Studio products

Product	COMPID
Data Studio stand-alone V2.2.1 (no charge version)	5724-DST00
Data Studio V2.2.1 (no charge version)	5724-DST04
Optim™ Development Studio V2.2.1 (charge version)	5724-DST01

Data Studio is a comprehensive data management solution that empowers you to effectively design, develop, deploy, and manage your data, databases, and database applications throughout the entire application development life cycle utilizing a consistent and integrated user interface. Included in this tooling suite are the tools for developing and deploying DB2 for z/OS stored procedures. Unlike Development Center, which was included in the DB2 V8.1 for Linux, UNIX, and Windows Application Development Client (ADC) component, Data Studio is independent of any other product offering and does not require a DB2 client to be installed.

1.2 Understanding the Data Studio packaging

Data Studio is made available in two no charge forms:

- A light weight Rich Client Platform (RCP) Eclipse executable version, called Data Studio stand-alone or RCP
- An Eclipse Interactive Development Environment (IDE) version, called Data Studio IDE.

In addition to these two products, a third fee version of Data Studio is also available, which contains more functions. This third product is Optim Development Studio. Throughout this document, we refer to all three product forms as *Data Studio*. Features that are specific to Optim Development Studio are pointed out when needed.

Data Studio stand-alone executes as a desktop application on Windows and Linux. Data Studio uses an installer, called Installation Manager, to install, update, and uninstall the product. Both Data Studio IDE and Optim Development Studio sit on top of the Eclipse Framework. Thus, both products have a similar *look and feel*. Certain versions of products in the InfoSphere™ Optim and InfoSphere Rational, brands also sit on top of the Eclipse framework. As a result, these products are able to share the Eclipse base (*shell sharing*).

A description of the supported products that can shell-share with Data Studio or Optim Development Studio is provided in the technote “Shell sharing with InfoSphere Data Architect, Optim Development Studio, and Optim Database Administrator” available from:

http://www.ibm.com/developerworks/data/library/techarticle/dm-0811khatri/index.html?ca=drs-

Data Studio supports the entire family of DB2 servers, as well as Informix®. Support for the Oracle data server is available in Optim Development Studio.

The latest version, Data Studio V2.2.1, supports:

- Version 9.x of DB2 for Linux, UNIX, and Windows
- Versions 8, 9, and 10 of DB2 for z/OS
- Versions 5.3 and later of DB2 for iSeries®
- Informix Data Servers

The suite of servers and functions that the Data Studio products support are summarized in Table A-1 on page 136 and Table A-2 on page 137.

For more information about Data Studio, see:

http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/IBM_Data_Studio

1.3 Eclipse Workbench common terminology

In this section we introduce the basic Data Studio concepts and terminology.

Data Studio is based on the open and extensible framework of the Eclipse Workbench.

The Eclipse Workbench is an Integrated Development Environment. It is the major delivery of a consortium of companies, called *eclipse.org*, and is an open source development platform. This consortium was initiated by IBM. Eclipse is currently the most successful open source project judging by the number of contributing participants. Eclipse defines extensibility interfaces and integration points, so other projects can contribute, extend, or consume “pieces” of the code, called *plug-ins*. Several specialized projects extend the basic IDE, such as Web Tools Project, Data Tools Project, and so on.

The Eclipse Workbench, or Workbench for short, consists of:

- Workspace
- Resources
- Perspectives
- Views
- Editors
- Wizards
- Task Launcher

1.3.1 Workspace

When you open the Workbench, you are asked to choose a workspace. All your resources and settings are saved in this workspace. Only one workspace is active at any given time. You can open a different workspace each time that you open the Workbench. You can also switch workspaces by clicking **File** → **Switch Workspace**.

1.3.2 Resources

A resource is a collective term for the projects, folders, and files that you created in a workspace. Typically, resources are viewed in a hierarchical format and can be opened for editing. There are three basic types of resources that exist in the Workbench:

- Files

  A file in Eclipse is comparable to files in the workstation. Each resource in Eclipse is associated with a file. Eclipse persists or saves all resources in the workspace as files in the file system.

- Folders

  Folders in the Eclipse workspace are comparable to directories in a file system. In the workspace, folders are containers for the various resources that can be created, viewed, or manipulated by the tooling.

- Projects

  In Eclipse, development is contained in projects. Projects contain folders, which in turn can contain either a set of objects or another folder. Projects have one or more nature associated with them, meaning that the contents and tasks that can be done on the objects within the project depend on the kind of project created.

  A project is either open or closed. When a project is closed, it cannot be changed in the Workbench. The resources of a closed project do not appear in the Workbench, but the resources still reside on the local file system. When a project is open, the structure and contents of the project can be viewed and modified.

1.3.3 Perspectives

A perspective is a group of views and editors in the Workbench window. One or more perspectives can exist in a single Workbench window. Each perspective contains one or more views and editors. Each perspective can have a different set of views, but all perspectives share the same set of editors.

Data perspective

The Data perspective is the default perspective for Data Studio IDE. The Data perspective provides a set of functions and specialized views for displaying, creating, deploying, and managing application development objects and database objects.

Database Administration perspective

The Database Administration perspective displays a set of views that the database administrator can use to manage databases and complete database administration tasks.

IBM Query Tuning perspective

The Query Tuning perspective is the default perspective for Data Studio stand-alone. This perspective provides the Administration Explorer view which displays the database objects in a hierarchical form.

Debug perspective

The Debug perspective is the primary perspective used by Data Studio when debugging an SQL or Java™ stored procedure. This perspective is used by other products, such as the Rational Developer V7 for System z®.

Java perspective

The Java perspective is the primary perspective used by Data Studio when developing pureQuery applications. This perspective is also by other products, such as the Rational Application Developer.

Other perspectives used by Data Studio are the Team Perspective and the Resource Perspective.

1.3.4 Views

A view is a visual component within the Workbench that is used to display a hierarchy of resources in the Workbench, display properties of a resource, and perform tasks on the resource. Modifications made in a view are saved immediately. Only one instance of a particular type of view can exist within a Workbench window.

In any of the perspectives you can:

- Open or show a view.
- Move a view to a different area of the workspace.
- Reset views.
- Minimize or maximize a view.

Several views can share an area of the workspace as in the case when multiple objects are opened in the Editor. The Output view also shows multiple types of output (for example, error log, problems, data output, and so on).

The views used by a database application developer differ slightly from the views used by a database administration developer. Commonly used views are the Data Source Explorer, Data Project Explorer, Administration Explorer, and Data Output views. More information about these views in Data Studio is given in 1.5, “Navigating through the Data Studio workspace” on page 26.

1.3.5 Editors

An editor is a visual component within the Workbench that is used to edit or browse a resource. Modifications made in the editor follow an open-save-close lifecycle model. An editor can be specialized for a function. Multiple instances of specialized editors can exist within a Workbench window.

1.3.6 Wizards

A wizard is a visual component within the Workbench that is used to step a user through a series of tasks related to a resource. The purpose of the wizard is to make a task easy for the user.

1.3.7 Task Launcher

Available in Data Studio IDE and Optim Development Studio only, the Task Launcher is used to simplify determining which perspective and views should be used when performing a task. Data Studio launches the proper perspective and default views based on what the user wants to do.

1.4 Installation, configuration, and setup

For Data Studio to communicate with the z/OS server, ensure that both the client and the server are properly set up. Here we describe the setup steps and areas of consideration for both the client and the z/OS Server.

- Connectivity and the JDBC driver selection
- Client setup
- DB2 for z/OS setup
- Unicode support
- Setup for SQL and Java stored procedures
- WLM application environments and procedures
- Data Studio actual costs setup
- Data Studio and JDBC driver selection
- Java SDK used by Data Studio
- Overview of routine development with Data Studio

1.4.1 Connectivity and the JDBC driver selection

Data Studio uses the JDBC driver specified in the Connection wizard in the following areas:

- The server connection
- The generated stored procedure (Only Java SQLJ requires the Driver significance[1], which is included in the `*.ser` file.)
- The runtime environment for Java stored procedures, both SQLJ and JDBC, which are determined by the WLM SPAS //JAVAENV DD statement
- The runtime environment for pureQuery applications

Data Studio V2.2.1 is shipped with the IBM Data Server Driver for JDBC and SQLJ V3.61. Data Studio and Optim Development Studio requires the IBM Data Server Driver for JDBC and SQLJ V3.57 or later for JDBC 3.0 applications, and IBM Data Server Driver for JDBC and SQLJ V4.7 or later for JDBC 4.0 applications.

For a detailed discussion about DB2 drivers, see *DB2 9 for z/OS: Distributed Functions*, SG24-6952.

Build process generates Java stored procedure

The build process performed by Data Studio generates the connection in the Java stored procedure using the default syntax *jdbc:default:connection*. This eliminates any JDBC Driver significance in the stored procedure source. Instead, the JDBC Driver significance is included in the *.ser file for Java SQLJ stored procedures during the customization process. No driver significance is included in generated Java JDBC stored procedures. The `*.ser` file is created during customization for Java SQLJ stored procedures and is performed by Data Studio using DB2SQLJCUSTOMIZE.

Runtime JDBC driver

The JDBC driver used at runtime for both Java stored procedures is determined by the JCC_HOME environment variable in the //JAVAENV DD statement of the WLM address space executing the stored procedure. For more information see “Runtime determines JDBC driver” on page 20.

[1] The driver significance pertains to the set of information regarding the JDBC driver being used.

1.4.2 Client setup

The client setup is made up of the following steps:

1. Obtaining the product.
2. Installing Data Studio.
3. Installing DB2 Connect.
4. Binding JDBC packages.

Obtaining the product

Unlike IBM Development Center, Data Studio is a separate installable product. Both the stand-alone and IDE versions are available from the IBM Support website:

http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/IBM_Data_Studio

Data Studio is also available as a downloadable feature from the DB2 for z/OS web page:

http://www.ibm.com/software/data/db2/zos/downloads/

You can also obtain Data Studio from Passport Advantage® or developerWorks®.

Installing Data Studio

After downloading the Data Studio package, follow the steps below for installing Data Studio or Optim Development Studio.

Installing Data Studio stand-alone

To do this:

1. Extract the product package to a directory in your computer (for example, `C:\temp\datastudio`).
2. Optional: Data Studio ships with a JRE at level 1.6. To use this JRE, update the JAVA_HOME and PATH environment variables to *<the directory where you extracted the produce>*`\eclipse\jre`.
3. From the directory where you extracted Data Studio, run the executable file:
   - Windows: `install.exe`
   - Linux: `./install.bin`
4. Follow the installation wizard steps to complete the installation.

Installing Data Studio IDE or Optim Development Studio

The downloaded image has the following layout:

1. Go to the *<drive>*`:\windows\InstallerImage_win32\install.exe`.
2. Double-click the **`install.exe`** file. The IBM Installation Manager wizard is launched.
3. Click the **Data Studio** and the **Version x.x.x** check boxes. Click **Next**.
4. Select the radio button for **I accept the terms in the license agreement**. Click **Next**.
5. In the Shared Resourced Directory, select **C:\Program Files\IBM\SDPShared**. Click **Next**.
6. In the Installation Directory text box, select **C:\Program Files\IBM\DS**. Click **Next**.
7. Do not extend the Eclipse IDE. Click **Next**.
8. The default setting is English. Select your language, then click **Next**.
9. A default list of features is preselected for you. Deselect or select features that you want installed. Click **Next**.

10. Verify that the disk space that you have can accommodate the installation size. Click **Install**.
11. Check for a success message, then click **Finish**.

Shell share: Data Studio IDE and Optim Development Studio can share a package group with other compatible products that have been installed with IBM Installation Manager. So in step 7 on page 7, you can opt to install the same package group or to create a new one. You can also opt to extend the Eclipse IDE you currently installed in your system. Several InfoSphere products, Optim or Rational, include the Eclipse IDE in their installation. You can significantly reduce your storage requirements if you opt to shell share or share the base Eclipse plug-ins of these products with Data Studio.

The technote “Information about which IBM Software products can be installed together so together so that they share a common environment” lists the compatible products that can shell-share with Data Studio. It is available from:

`http://www.ibm.com/support/docview.wss?rs=2042&uid=swg21279139`

Data Studio is installed into two default directories:

- `C:\Program Files\IBM\SDPShared`
- `C:\Program Files\IBM\Data Studio`

Installing DB2 Connect

This step is optional. If you are using Data Studio for development purposes, then you do not need to have DB2 Connect™ installed.

Data Studio ships with the IBM Data Server driver for JDBC and SQLJ[2], also called the Universal driver. These include the license jars, `db2jcc.jar` and `db2jcc_license_cisuz.jar`, needed to connect to DB2 for z/OS servers.

You can opt to use the license jars supplied by DB2 Connect. In the Driver Properties of the database connection, specify these jars and their location in the file system (see 2.1.3, “Editing the connection” on page 54). Note, however, that certain features and functions in Data Studio, and in particular Optim Development Studio, require specific levels of `db2jcc.jar`. You need to verify whether the version of the DB2 Connect license jars is equal to or greater than the version shipped with Data Studio. To verify this, type the command shown in Example 1-1 on a Windows command prompt, where *classpath* is the directory where `db2jcc.jar` is located.

Example 1-1 How to verify the JCC version

```
java -cp <classpath> com.ibm.db2.jcc.DB2Driver -version
```

[2] Also called, the IBM Universal Driver for JDBC and SQLJ

Binding JDBC packages

The JDBC packages in the IBM Data Server driver for JDBC and SQLJ need to be bound to the server. The DB2Binder utility performs this task. This task needs to be done only once per server per collection ID. The DB2Binder utility can be executed from the server side or the client side. Example 1-2 shows the **DB2Binder** command executed from Windows.

Example 1-2 Connecting and binding the JDBC packages

```
set
CLASSPATH=c:\Progra~1\IBM\sqllib\java\db2jcc.jar;c:\Progra~1\IBM\sqllib\java\db2jc
c_license_cisuz.jar;c:\Progra~1\IBM\sqllib\java\db2jcc_license_cu.jar;%$CLASSPATH%

java com.ibm.db2.jcc.DB2Binder -url jdbc:db2://utec730.vmec.svl.ibm.com:446/STLEC1
-user ADMF001 -password CODESHOP  -collection DSNJDBC

cd C:\Progra~1\IBM\SQLLIB\bnd
db2 connect to EC730V10 user ADMF001 using CODESHOP
db2 bind @ddcsmvs.lst BLOCKING ALL SQLERROR CONTINUE GRANT PUBLIC
db2 connect reset
```

In Example 1-2, *-url* points to the domain:port//location of the DB2 for z/OS server that you want to connect to. The *user* is the TSO logon ID, and *password* is the TSO logon ID password. The bind might need to be repeated after applying a FixPak to Data Studio. If the DB2Binder is not run, and you do not re-execute the bind, you receive -805 at the workstation when trying to connect to the server from Data Studio.

Note: Both the location and collection IDs should be in uppercase when submitting the **DB2Binder** command from the client. To continue typing a long line to the next line, type a backslash (\), followed by a space, then continue typing on the next line.

1.4.3 DB2 for z/OS setup

Data Studio requires the following minimum prerequisites for:

- SQL and Java stored procedures:
  - Language Environment®
  - Workload Manager
  - Resource Recovery Services
  - REXX language
  - Unicode support
- External SQL stored procedures
  - C compiler
- Java stored procedures
  - IBM Data Server Driver for JDBC and SQLJ V3.57
  - IBM SDK for z/OS, Java 2 Technology Edition V1.4 (SDK1.4.2, 5655-I56)

Data Studio interacts with the DB2 for z/OS subsystems using several DB2-supplied stored procedures. In DB2 10, these stored procedures are installed and configured during installation. Installation job DSNTIJRT installs and configures the DB2-supplied stored procedures that were previously installed by the following customization jobs:

- DSNTIJSD
- DSNTIJRX

- DSNTIJTM
- DSNTIJMS
- DSNTEJ6W
- DSNTIJSG
- DSNTIJCC

Data Studio authorization setup

General authorities and privileges for using Data Studio follow the regular rules for DB2 for z/OS and are listed in Table 1-2.

Table 1-2 General authorities and privileges for all platforms using Data Studio

Task	Authorities and privileges
Access target databases.	CONNECT
Register stored procedures with a database server.	CREATE PROCEDURE Also requires one of the following privileges: ► SYSADM or DBADM. ► CREATEIN for the schema if the schema name of the stored procedure refers to an existing schema. ► IMPLICIT_SCHEMA authority on the database if the implicit or explicit schema name of the stored procedure does not exist. IMPLICIT_SCHEMA allows you to implicitly create an object with a CREATE statement and specify a schema name that does not already exist. SYSIBM becomes the owner of the implicitly created schema, and PUBLIC is given the privilege to create objects in this schema. ► CREATE IN privilege on desired collection ID.
Retrieve rows from a table or view.	SELECT
Create a view on a table.	SELECT
Run the EXPORT utility.	SELECT
Insert an entry in a table or view, and run the IMPORT utility.	UPDATE
Change an entry in a table, a view, or one or more specific columns in a table or view.	UPDATE
Delete rows from a table or view.	DELETE
To use the Data Studio Unified Debugger.	DEBUGSESSION
Test a stored procedure.	SYSADM or DBADM or EXECUTE or CONTROL for the package associated with the stored procedure (for SQL stored procedures or Java stored procedures with embedded SQL)
Drop a stored procedure.	You must have ownership of the stored procedure and at least one of the following: ► DELETE privilege ► DROPIN privilege for the schema or all schemas ► SYSADM or SYSCTRL authority
Update a stored procedure.	You must have ownership of the stored procedure and at least one of the following: ► UPDATE privilege ► ALTERIN privilege for the schema or all schemas ► SYSADM or SYSCTRL authority

Data Studio accesses a number of DB2 system catalog tables on z/OS.

Table 1-3 lists the privileges required to view the objects in the Data Source Explorer. The privileges can be held by any authorization ID of the process, either the primary authorization ID or any secondary authorization ID.

Table 1-3 Privileges required to view the objects in the Data Source Explorer in Data Studio

Additional required privileges	**EXECUTE privilege on DSNTPSMP**
SELECT privilege on:	SYSIBM.SYSCOLAUTH SYSIBM.SYSCOLUMNS SYSIBM.SYSDATABASE SYSIBM.SYSDBAUTH SYSIBM.SYSINDEXES SYSIBM.SYSJAROBJECTS SYSIBM.SYSPACKAGE SYSIBM.SYSPACKAUTH SYSIBM.SYSPACKDEP SYSIBM.SYSPARMS SYSIBM.SYSPLAN SYSIBM.SYSPLANAUTH SYSIBM.SYSRESAUTH SYSIBM.SYSROUTINEAUTH SYSIBM.SYSROUTINES SYSIBM.SYSSCHEMAAUTH SYSIBM.SYSSYNONYMS SYSIBM.SYSTABAUTH SYSIBM.SYSTABLES SYSIBM.SYSUSERAUTH SYSIBM.SYSVIEWS

Data Studio also accesses the catalog and non-catalog tables listed in Table 1-4 when creating external SQL stored procedures.

Table 1-4 DB2 system catalog tables accessed when creating SQL stored procedures

SELECT privilege on:	SYSIBM.SYSDUMMY1 SYSIBM.SYSROUTINES SYSIBM.SYSPARMS
SELECT, INSERT, UPDATE, and DELETE privilege on:	SYSIBM.SYSROUTINES_SRC SYSIBM.SYSROUTINES_OPTS SYSIBM.SYSPSM SYSIBM.SYSPSMOPTS
ALL on the global temporary table	SYSIBM.SYSPSMOUT

Data Studio accesses the catalog tables listed in Table 1-5 when creating JAVA stored procedures.

Table 1-5 DB2 system catalog tables accessed when creating Java stored procedures

SELECT privilege on:	SYSIBM.SYSROUTINES SYSIBM.SYSDUMMY1 SYSIBM.SYSPARMS SYSIBM.SYSJARCONTENTS SYSIBM.SYSJAROBJECTS SYSIBM.SYSJAVAOPTS

The user connecting to DB2 for z/OS must hold privileges listed in Table 1-3 on page 11 and Table 1-4 on page 11 for SQL stored procedures, and Table 1-5 for Java stored procedures.

1.4.4 Unicode support

Data Studio users creating SQL and Java Stored Procedures experience incorrect codepage translation when Unicode Conversion Services (UCS) is not set up. For more information, see *Support for Unicode: Using Conversion Services*, SC33-7050, and the following website:

http://www.ibm.com/servers/s390/os390/bkserv/latest/v2r10unicode.html

To determine whether UCS is active, issue **'D UNI,ALL'** from an SDSF screen. If the support is installed, you receive output with the actual CCSID entries that have been defined. If UCS is not installed, the following message is returned:

```
CUN2029S CONVERSION ENVIRONMENT IS NOT AVAILABLE
```

The installation of Unicode Conversion Services requires:

- Updating SYS1.PARMLIB member IEASYSxx with UNI=xx
- Adding SYS1.PARMLIB member CUNUNIxx
- Defining Conversion Table with CCSID entries in SYS1.PARMLIB (CUNIMGxx)
- IPLing the system
- De-activating or activating the conversion table

Without Unicode Conversion Services set up, you can initially create, view, and modify a Java stored procedure. However, restoring the source from the database of a previously created Java stored procedure returns the source as a single line with red blocks interspersed, which represent line feeds that have not been translated correctly. The Data Studio support for SQL stored procedures handles the code conversion, and UCS is not required.

1.4.5 Setup for SQL and Java stored procedures

Data Studio uses DB2-supplied stored procedures to build external SQL and Java stored procedures. In DB2 10 these routines are installed by installation job, DSNTIJRT.

The DB2-supplied stored procedures listed in Table 1-6 are used by Data Studio.

Table 1-6 DB2-supplied stored procedures used by Data Studio

Procedure name	Description
SYSPROC.DBG_%MANAGER	Server routines required by the Unified Debugger to enable debug interaction with Data Studio.
DSNTPSMP	Used by Data Studio for creating, modifying, or dropping an external SQL stored procedures.

Procedure name	Description
SYSPROC.SQL%	Server routines used by the SQL Wizard when creating SQL Statements within the New Stored procedure wizard in Data Studio. These are also used by the Administrator Explorer view.
SQLJ.DB2_%_JAR SQLJ.DB2_UPDATE_JARINFO SQLJ.ALTER_JAVA_PATH	Server routines used by Data Studio on behalf of the developer, to update the catalog tables with the Java stored procedure jar, class, and other information.
WLM_REFRESH[a]	Used by Data Studio on behalf of the developer to perform a refresh of a specific WLM application environment after a stored procedure is re-deployed to ensure that the latest changes are in effect.

a. This stored procedure requires RACF® permissions using an authorization ID that has MVS™ command authority.

Additionally, the RACF class DSNR needs to be activated prior to calling this stored procedure. This is done using the RACF panels listed in Table 1-7.

Table 1-7 Activate class DSNR

RACF panel	Option to select
RACF Services Option Menu	5.
RACF System Security Options Menu	3.
RACF Set Class Options Menu, panel 1	Enter YES in To CHANGE options for SPECIFIC CLASSES field.
RACF Set Class Options Menu, panel 2	Enter DSNR in CLASS field and YES in ACTIVE field.

Setup specific to SQL stored procedures only

The following setup is needed for Data Studio to create external SQL stored procedures.

1. Configure DSNTPSMP.

   The WLM AE used by DSNTPSMP needs to be configured with NUMTCB=1. Create the procedure that runs in this WLM application environment using <hlq>.SDSNSAMP(DSN8WLMP).

   When DSNTPSMP executes, it creates a compiled SQL load module using the C compiler in the data set referenced by //SQLLMOD in its WLM AE. This same data set needs to be included in STEPLIB in the WLM procedure where the user's SQL stored procedure created by DSNTPSMP runs.

2. Optional: Configure CFGTPSMP.

   You can optionally define the data set for the //CFGTPSMP DD statement. This is a DD statement that can be included in the WLM procedure that executes DSNTPSMP. This is a configuration file that externalizes some settings for DSNTPSMP. It is expected that additional options will be added to this data set in future releases.

   Example 1-3 lists the definition of the configuration file that we used on DB9A.

Example 1-3 Sample CFGTPSMP configuration data set

```
;-THE CONFIGURATION KEYWORDS AND VALUES ARE AS FOLLOWS:
;-
;-                    .-CBCDRVR--.
;-C_COMPILER--=--+----------+--------------------------|
```

```
;-                    |-CCNDRVR--|
;-                    |-CBC320PP-|
;-                    '-EDCDC120-'
;-THE NAME OF THE C COMPILER TO USE. ADJUSTMENT OR ADDITIONAL
;-CONFIGURATION OF THE WLM ENVIRONMENT IS USUALLY REQUIRED
;-WHEN CHANGING THE C COMPILER.
;-
VALIDATE_BIND = DEFAULT
;- DEFAULT, PERMIT, ENFORCE
;-SPECIFIES INSTALLATION CONTROL FOR ALL BUILDS OVER THE
;-USAGE OF THE BIND PACKAGE OPTION VALIDATE(BIND). CHANGING
;-THE DEFAULT MAY PROVIDE A PERFORMANCE IMPROVEMENT.
ISOLATION_DEFAULT = CS
;-    CS  OR RR
;-SPECIFIES INSTALLATION CONTROL OVER THE DEFAULT VALUE FOR
;-THE BIND PACKAGE OPTION ISOLATION.  CHANGING THE DEFAULT
;-MAY PROVIDE A PERFORMANCE IMPROVEMENT.
;
CURRENTDATA_DEFAULT = YES
;-SPECIFIES INSTALLATION CONTROL OVER THE DEFAULT VALUE FOR
;-THE BIND PACKAGE OPTION CURRENTDATA.  CHANGING THE DEFAULT
;-MAY PROVIDE A PERFORMANCE IMPROVEMENT.
;
DSNTPSMP_TRACELEVEL= LOW
;- OFF, LOW, MEDIUM, HIGH
;-CONTROLS THE LEVEL OF DSNTPSMP TRACE DATA WRITTEN OUT TO
;-THE DD:SYSTSPRT DATASET IN THE WLM ADDRESS SPACE. SETTING
;-THE VALUE TO OFF WILL MINIMIZE, NOT ELIMINATE, LOG RECORDS
;-WRITTEN TO DD:SYSTSPRT IN THE WLM-SPAS.
;
```

Setup specific to Java stored procedures only

The following setup is needed for Data Studio to create Java stored procedures.

1. Install JDBC drivers.

   The DB2 JDBC driver needs to be set up in your environment through SMP/E. The *DB2 10 for z/OS Program Directory*, GI10-8829-00, describes the installation for the JDBC driver in the Receive Sample job DSNRECV3 for ODBC/JDBC/SQLJ.

2. Create the JAVAENV data set.

   The WLM procedure where Java stored procedures execute requires a JAVAENV DD. This data set defines the Java environment variables.

   This JAVAENV data set should have the characteristics shown in Table 1-8.

Table 1-8 JAVAENV definition

JAVAENV data set characteristics		
LRECL	255	This maximum is limited by LE. 245 bytes usable. If more than 245 bytes are included, unpredictable results occur.
RECFM	VB	
ORGANIZATION	PS	

Example 1-4 shows an example of the contents of the JAVAENV data set.

Example 1-4 Contents of JAVAENV - DB9AU.JAVAENV file

```
ENVAR("JAVA_HOME=/usr/lpp/java/J5.0",
"JCC_HOME=/usr/lpp/db2/db9a/db2910_jdbc",
"CLASSPATH=/usr/lpp/db2/db9a/db2910_jdbc/userproc",
"DB2_BASE=/usr/lpp/db2/db9a/db2910_base",
"RESET_FREQ=-1"),
XPLINK(ON)
```

All the environment variables need to be included in this file. Ensure that the total length of all the entries does not exceed 245 bytes (exclude the blanks)[3]. In case your entries exceed the 245-byte limit, you need to take a different approach. Example 1-5 shows an alternate form of JAVAENV definitions.

Example 1-5 Contents of JAVAENV having _CEE_ENVFILE variable

```
ENVAR("_CEE_ENVFILE=/usr/lpp/db2/db9a/evnfile.txt",
"JAVA_HOME=/usr/lpp/java/J5.0",
"DB2_BASE=/usr/lpp/db2/db9a/db2910_base",
"RESET_FREQ=-1"),
XPLINK(ON)
```

The _CEE_ENVFILE variable points to an HFS file that contains most of the environment variables, because this file has no limitation of size. The JAVA_HOME variable must be defined in the JAVAENV data set, and not in the HFS file corresponding to `_CEE_ENVFILE`. Example 1-6 shows the contents of the `_CEE_ENVFILE` file. This is a standard UNIX file where each line must start in column 1 and the continuation character is a backslash (`\`).

Example 1-6 Contents of the _CEE_ENVFILE - /usr/lpp/db2/db9a/envfile.txt

```
JCC_HOME=/usr/lpp/db2/db9a/db2910_jdbc
CLASSPATH=/usr/lpp/db2/db9a/db2910_jdbc/userproc
```

You can use _CEE_ENVFILE to overcome the 245-byte limit when specifying other environmental variables that tend to be long or transitory in nature, such as JITC_COMPILING and JITC_COMPILEOPT.

Environment variables in the JAVAENV data set

Table 1-9 provides a description of the various environment variables that need to be defined in the JAVAENV data set.

Table 1-9 Description of JAVAENV variables

Environment variable	Description
JCC_HOME[a]	This environment variable is set to the location of the JCC driver, for example, `JCC_HOME=/usr/lpp/db2/db9a/db2910_jdbc`.
JAVA_HOME	This environment variable indicates to DB2 that the WLM environment is for Java routines. The value of JAVA_HOME is the highest-level directory in the set of directories that contain the Java SDK, for example, `JAVA_HOME=/usr/lpp/java/IBM/J15`.
DB2_BASE	The DB2 base directory defaults to `/usr/lpp/db2910_base`. If you did not use the default location, set this environment variable to the directory where your DB2 base directory is installed.

[3] The 245-byte limit is a restriction of the Language Environment.

Environment variable	Description
CLASSPATH	The directory where you place your compiled stored procedures. A detailed discussion of CLASSPATH can be found in 13.7, “Making the stored procedure class files available to DB2” on page 201.
JVMPROPS	Here you can optionally specify the name of a USS file that contains JVM startup options. See “JVMPROPS” on page 190.
RESET_FREQ	Specify a value of -1 to indicate that the JVM is to be started in its non-resettable mode and is never reset.
HEAP	When debugging with JDK 1.4[a], set this to (8M,2M,ANYWHERE,KEEP). Otherwise, you can code (NONE) or remove this variable.
WORK_DIR	Optional: If you do not code a JAVAOUT or JAVAERR DD card in your WLM, you can set this environment variable to a valid HFS directory. This will be the default directory for STDOUT and STDERR. The default file names will be `server_stdout.txt` and `server_stderr.txt`.

a. If the DB2_HOME environment variable is coded in the JAVAENV data set, DB2 for z/OS V8 ignores this. However, in DB2 9 for z/OS, this causes an error when the WLM is started.

Heap size: When using JVM 1.4.2, the default native heap size is insufficient for debugging Java stored procedures with the IBM Data Studio. The JVM 1.5 and 1.6 default native heap size is sufficient. The JDK1.4.2 heap size therefore has to be increased to (8M,2M,ANYWHERE,KEEP). Do not put this environment variable in the _CEE_ENVFILE. For example, in your JAVAENV file, you can code:

```
MSGFILE(JSPDEBUG,,,,ENQ),
XPLINK(ON),
HEAP(8M,2M,ANYWHERE,KEEP),
ENVAR("_CEE_ENVFILE=/u/oeusr05/CEEOPTIONS.txt")
```

DB2 9 and 10 for z/OS flag the presence of the environment variable DB2_HOME as an error. DB2 for z/OS V8 ignores this variable.

JVMPROPS

JVMPROPS is the environment variable that specifies the name of a z/OS UNIX System Services file that contains startup options for the JVM in which the stored procedure runs. JVMPROPS is the Java stored procedures environment mechanism to set the -Xoptionsfile option.

Example 1-7 shows the contents of the HFS file.

Example 1-7 Contents of the JVMPROPS file

```
# Properties file for JVM for Java stored procedures
# Sets the initial size of middleware heap within non-system heap
-Xms64M
# Sets the maximum size of nonsystem heap
-Xmx128M
#initial size of system heap
-Xinitsh512K
```

For information about JVM startup options, see the *BM 31-bit and 64-bit SDKs for z/OS, Java 2 Technology Edition, Version 5 SDK and Runtime Environment User Guide,* available at:

http://www.ibm.com/servers/eserver/zseries/software/java/pdf/sdkguide.zos.pdf

Note: To enable class sharing in JDK 1.5, code the -Xshareclasses option in this file.

1.4.6 WLM application environments and procedures

The DB2-supplied stored procedures needed by Data Studio and installed in DSNTIJRT, require a WLM environment for running them. Installation job, DSNTIJRT defines a set of core WLM environments which can be used by the DB2-supplied stored procedures. These DB2 core WLM environments are intended as a starting point. Over time, you can optimize these environments or create additional WLM environments that are more suited to the routines you run.

Installation job DSNTIJMV creates an address space procedure for each of the core WLM environments. The WLM environment address space names have the form *ssnm*WLM*suffix*, where *ssnm* is the DB2 subsystem name specified on panel DSNTIPM in the SUBSYSTEM NAME field, and *suffix* is a character that differentiates the various WLM environments. Running DSNTIJMV adds these procedures to the system PROCLIB.

Table 1-10 lists the DB2 core WLM environments and their corresponding procedures used with Data Studio.

Table 1-10 DB2 core WLM environments and their corresponding procedures

WLM environment name	WLM procedure name	Description/usage
DSNWLM_GENERAL	DSNWLMG	For DB2-supplied stored procedures that support JDBC.
DSNWLM_JAVA	DSNWLMJ	For executing Java stored procedures. This procedure includes a DD statement for the JAVAENV file. See “Create the JAVAENV data set.” on page 14 for details about the JAVAENV file.
DSNWLM_DEBUGGER	DSNWLMD	For executing the DB2-supplied stored procedures used by the Unified Debugger.
DSNWLM_PGM_CONTROL	DSNWLMP	For executing the DB2-supplied stored procedure, WLM_REFRESH.
DSNWLM_REXX	DSNWLMR	For executing the DB2 SQL Procedure Processor routine, DSNTPSMP. However, it can be shared with the other DB2-supplied REXX stored procedures, DSNTBIND and ADMIN_COMMAND_DSN_%/.
DSNWLM_XML	DSNWLMX	For executing the DB2-supplied XML processing routines used in Data Studio’s XML support. These environments require large amounts of memory.

Creating multiple versions of DSNTPSMP

Multiple versions of DSNTPSMP for external SQL stored procedures are needed if there are different resource requirements for the same stored procedure needed on the same DB2 system during the stored procedure build process. That is, there might be a test version and a production version where the WLM procedure needs different data sets in either STEPLIB for external SQL stored procedures.

When multiple versions of either of these DB2-supplied stored procedures are required, first register a new copy of DSNTPSMP with a new schema (SYSPROC is the default). In that registration, specify a new WLM procedure where this copy of DSNTPSMP executes and complete other similar steps for SQL stored procedures as described in “Setup specific to SQL stored procedures” on page 14.

Selecting a different version of DSNTPSMP

SQL stored procedures can specify a different schema value, and optionally a different build utility than the DB2-supplied DSNTPSMP REXX stored procedure. Specify the default build utility schema that is used by clicking **Window** → **Preferences** → **Data Management** → **SQL Development** → **Routines** → **Deploy Options**. Figure 1-1 shows an example for setting the build utility, DSNTPSMP, with a different schema value from the SYSPROC default.

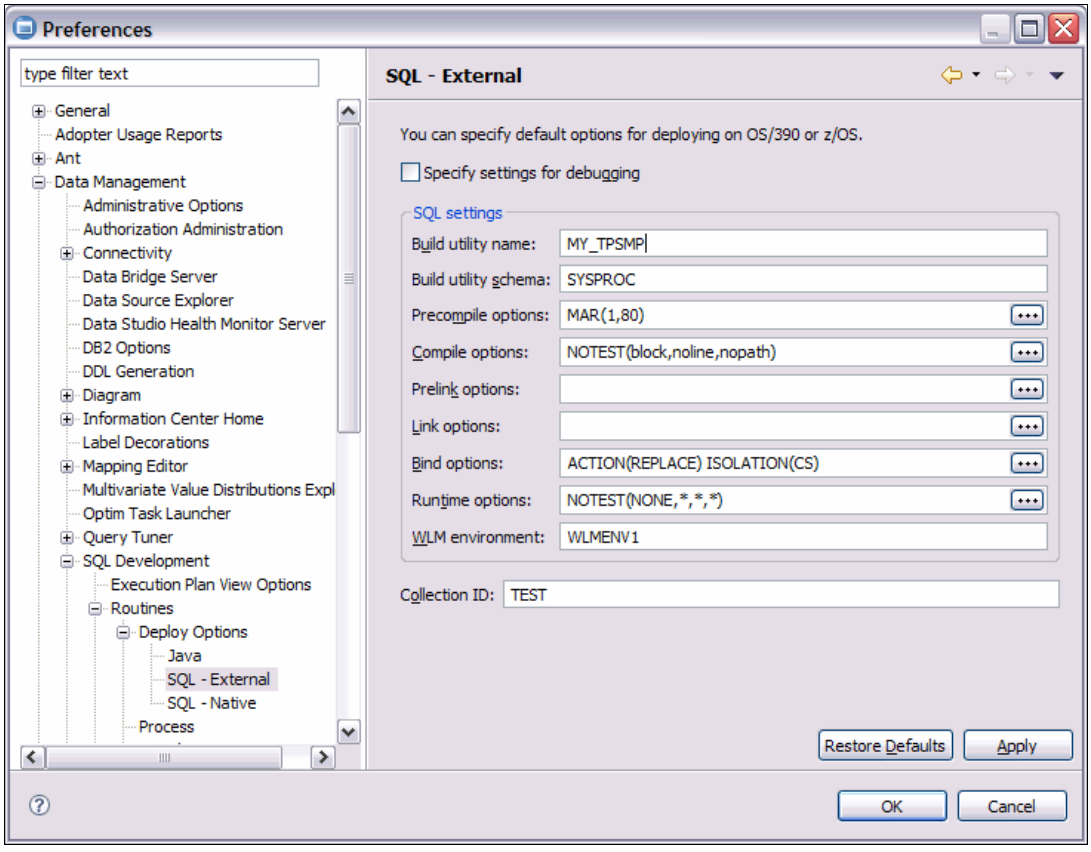

Figure 1-1 DSNTPSMP setting with different schema

When deploying a new SQL or Java stored procedure using the Deploy wizard, a different build schema and utility (external SQL only) can be selected from the **Routine options** page → **Deploy Options tab** → **Build utility** field (Figure 1-2).

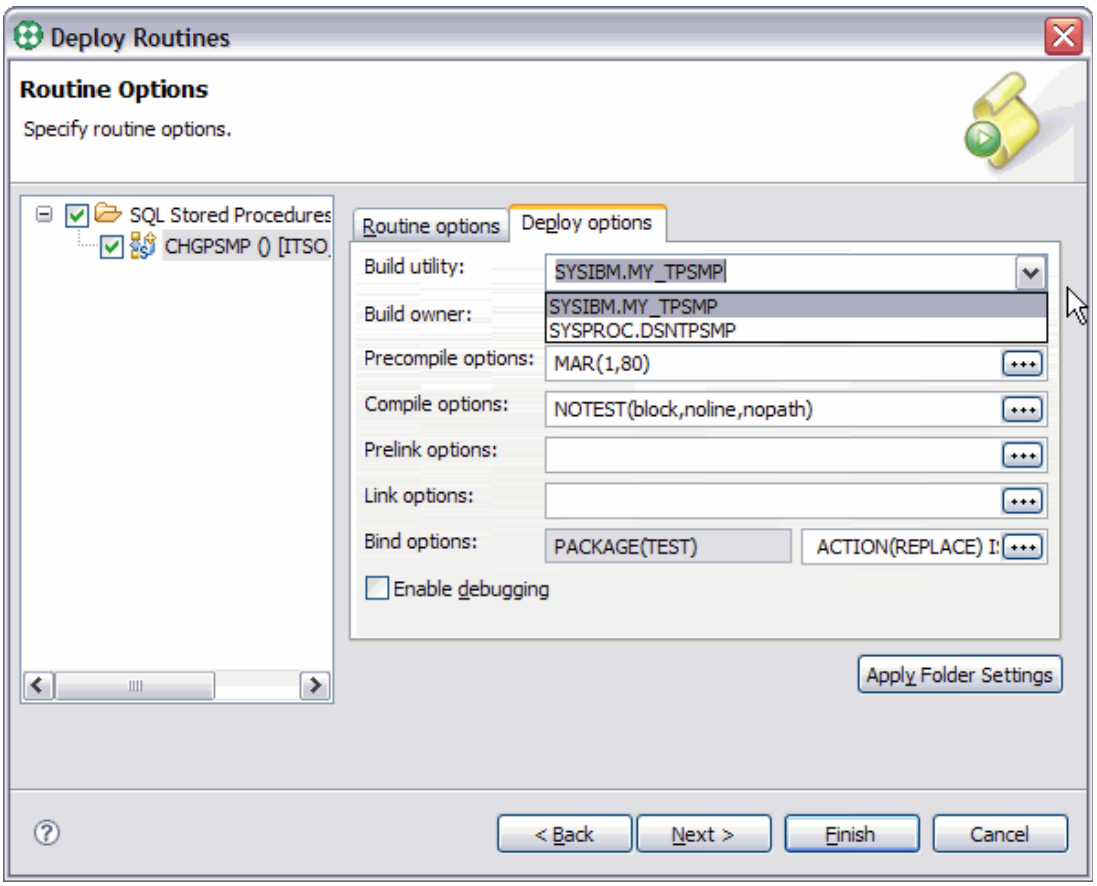

Figure 1-2 Multiple versions of schema

1.4.7 Data Studio actual costs setup

SYSPROC.DSNWSPM is a DB2-supplied stored procedure called by Data Studio from the Routine Editor when initially creating a SQL stored procedure for z/OS. DSNWSPM measures the following areas of a specific SQL statement in your stored procedure:

- CPU time
- Latch/lock wait time
- Getpages
- Read I/Os
- Write I/Os

The installation job DSNTIJRT installs and configures DSNWSPM. A final setup step for actual costs is to ensure that the DB2 accounting trace is running. If it is not, issue the following command to start it:

```
-START TRACE(ACCTG) CLASS(1,2,3)
```

1.4.8 Data Studio and JDBC driver selection

Data Studio uses the JDBC driver specified in the Connection wizard in the following three areas:

- The server connection profile
- The generated stored procedure (Only Java SQLJ requires the Driver significance[4], which is included in the `*.ser` file.)
- The runtime environment for Java stored procedures, both SQLJ and JDBC, which are determined by the WLM SPAS //JAVAENV DD statement

The default JDBC driver that Data Studio uses is the IBM Data Server Driver for JDBC and SQLJ.

Build process generates Java stored procedure

The build process performed by Data Studio generates the connection in the Java stored procedure using the default syntax, jdbc:default:connection. This eliminates any JDBC Driver significance in the stored procedure source. Instead, the JDBC Driver significance is included in the `*.ser` file for Java SQLJ stored procedures during the customization process. No Driver significance is included in generated Java JDBC stored procedures. The `*.ser` file is created during customization for Java SQLJ stored procedures and is performed by Data Studio using DB2SQLJCUSTOMIZE.

Runtime determines JDBC driver

The JDBC driver used for run time for both Java SQLJ and Java JDBC stored procedures is determined by the //JAVAENV DD statement in the WLM SPAS where the stored procedure executes. The JCC_HOME environment variable in this file specifies the location in UNIX System Services (USS) where the Universal JDBC driver is installed.

The default directory structure for JCC_HOME is `/usr/lpp/db2/db2a10/jcc`. For more details on the //JAVAENV DD statement, see “Create the JAVAENV data set” on page 13.

For more information about setting up the IBM Data Server Driver for JDBC and SQLJ, see the section “Installing the IBM DB2 Driver for JDBC and SQLJ” on the DB2 for z/OS Information Center website:

http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc.inst/db2z_installjccintro.htm

1.4.9 Java SDK used by Data Studio

When connected to DB2 for z/OS, Data Studio uses the client-side Java Software Development Kit (JDK) to compile and (optionally) translate and customize a Java stored procedure.

When Data Studio “calls” a Java stored procedure, the Java Virtual Machine (JVM) and Java Runtime Environment, both included with the server-side JDK, are used to execute the program.

Java methods used at compile time must be available at run time in the Java Runtime Environment (JRE), otherwise an execution error indicating a mismatch occurs in the WLM SPAS. For instance, calling a Java stored procedure that included JDK v1.6 methods fails if it executes in a WLM SPAS referencing a JRE v1.5.

[4] The Driver significance pertains to the set of information regarding the JDBC driver being used.

Data Studio IDE ships with a JDK 1.6 library. This is the default JDK used when developing Java stored procedures and applications. The JDK is found in <*Data Studio install directory*>\jdk.

Data Studio stand-alone does not ship any JDK, as Java development is not available in the stand-alone version. However, a Java Runtime Environment is shipped with Data Studio stand-alone in <*Data Studio stand-alone directory*>\jre, which allows Java stored procedures and applications to execute.

The user can opt to compile with a lower-level JDK by setting the JDK level in three places:

- In the workspace Preferences
- In the Project's Properties
- In the stored procedure's Deploy options

Overriding the workspace's default JDK

Overriding the default JDK used by the Data Studio client is done through the workspace Preferences (Figure 1-3).

1. Click **Window** → **Preferences** → **Data Management** → **SQL Development** → **Routines** → **Deploy Options** → **SQL -External**.
2. Click **Browse**. Point the file browser to the directory where the overriding JDK is located.

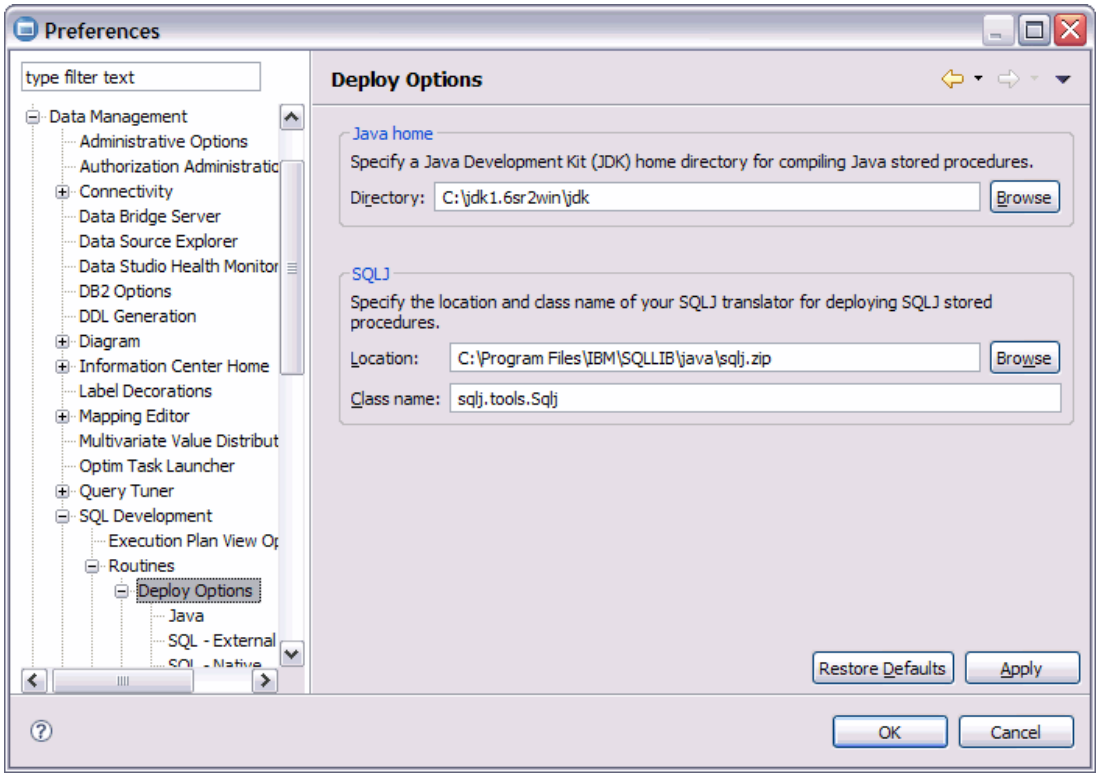

Figure 1-3 Setting preferences

JDK at run time for a Java stored procedure

The JDK used at run time is determined by the //JAVAENV DD statement in the WLM SPAS where the stored procedure executes. Specifically, the JAVA_HOME environment variable included in this DD statement determines the JDK that is selected.

DB2 10 for z/OS supports JDK 1.4 or later. JDBC 4.0 functions require Java Technology Edition V6 (JDK 1.6) or later.

When using JDK 1.4.1 or later, the XPLINK(ON) parameter is required in the //JAVAENV DD statement. When the XPLINK(ON) parameter is not included in a //JAVAENV DD statement that specifies JDK 1.4.1, the WLM SPAS does not initialize and the following error message is included in the WLM SPAS joblog:

```
+DSNX961I  DSNX9WLJ ATTEMPT TO PERFORM JNI FUNCTION CreateJavaVM  421
 FAILED FOR STORED PROCEDURE . .  SSN= DB8A     PROC= DB8AJAV1 ASID=
 008E  CLASS=  METHOD=  ERROR INFO= DSNX9WLS ESTAE ENTERED
```

If JSPDEBUG is turned on in the same //JAVAENV DD statement, information like the following is also included, indicating that a call was made from a NOXPLINK-compiled application to an XPLINK-compiled exported function in DLL libjvm.so, and the XPLINK(ON) runtime option was not specified:

```
CEE3501S The module libjvm.so was not found.
         From entry point initjvm at compile unit offset +000014A0 at entry off
```

Runtime JAVAENV DD statement example

Example 1-8 shows using JDK 1.4 and the JCC driver in DB2 10.

Example 1-8 Universal JDBC driver - JDK 1.4.1

```
XPLINK(ON),
ENVAR("JCC_HOME=usr/lpp/db2a10/jdbc/",
"JAVA_HOME=/usr/lpp/java/IBM/J1.4"),
MSGFILE(JSPDEBUG,,,,ENQ)
```

1.4.10 Overview of routine development with Data Studio

When the Data Studio is launched, it asks for a directory to be used for the tooling's workspace. A default directory is provided, but the user can change this to another directory. Data Studio then checks whether there is a DB2 database directory in the client's workstation. If there is one, it creates Connection objects for each of the DB2 aliases in the DB2 database directory. Figure 1-4 shows the basic user interface flow to get you started using Data Studio.

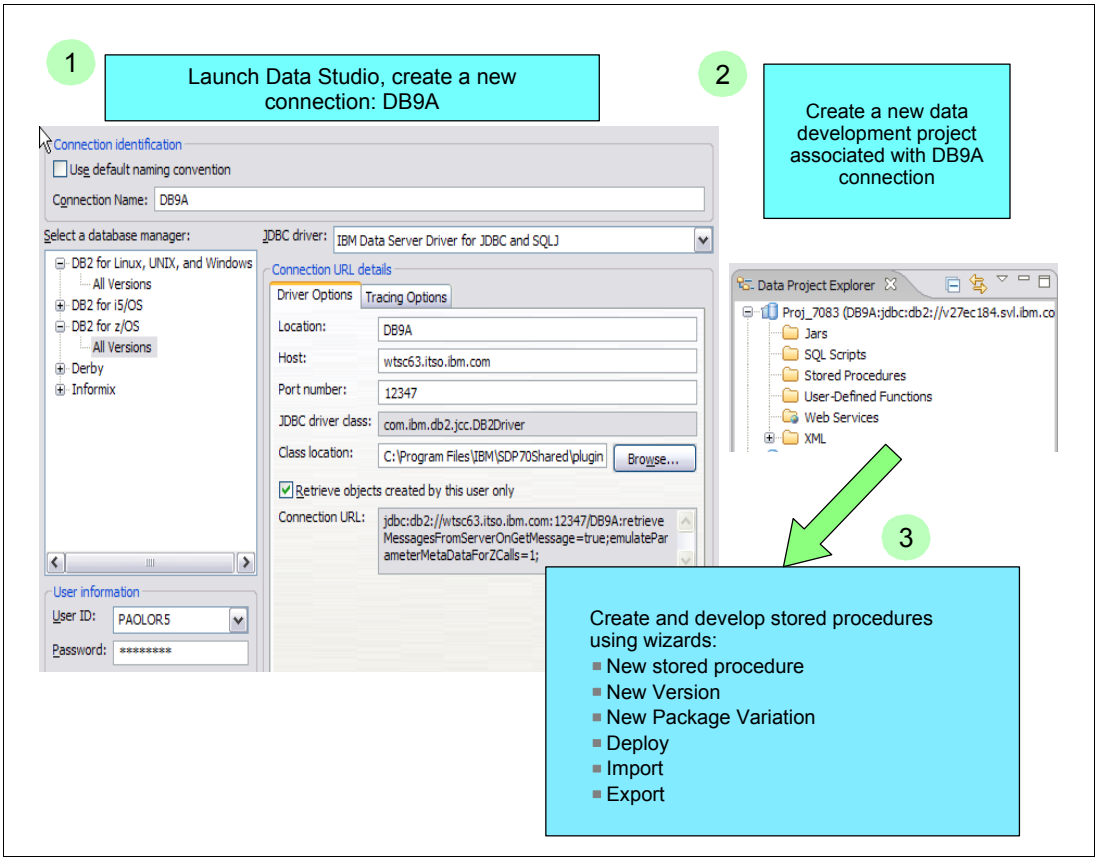

Figure 1-4 Starting Data Studio

Data Studio creates external SQL and Java stored procedures on z/OS using multiple DB2-supplied stored procedures. The main DB2-supplied stored procedures that perform the processing on z/OS for Data Studio are:

- DSNTPSMP for SQL stored procedures
- SQLJ.DB2_INSTALLJAR
- SQLJ.DB2_REPLACEJAR
- SQLJ.DB2_UPDATEJARINFO

When connected to DB2 10 for z/OS, Data Studio can create both external SQL stored procedures and Native stored procedures that no longer require DSNTPSMP. The tooling identifies the type of SQL stored procedure by appending **"**`(external)`**"** or **"**`(native)`**"** to the stored procedure name displayed in the Data Project Explorer.

External SQL stored procedures built with DSNTPSMP

DSNTPSMP is a REXX DB2-supplied stored procedure that builds External SQL stored procedures on DB2 for z/OS. Multiple *build* functions are supported by this stored procedure. Table 1-11 lists the functions that are supported.

Table 1-11 DSNTPSMP supported functions

Type	Name	Function
Basic	BUILD	Creates a new SQL procedure only.
	ALTER	Updates (most) stored procedure options.
	REBIND	Performs Bind Package again (not REBIND command).
	DESTROY	Removes an existing SQL procedure.
	REBUILD	Builds an SQL procedure. Destroys the existing one first. Best for changing parameter-declarations.
Modify	ALTER_REBIND	Performs ALTER of collection ID.
	ALTER_REBUILD	Performs procedure-body updates.
	REBUILD_DEBUG	Basic function plus Debugger support.
	ALTER_REBUILD_DEBUG	Hybrid function plus Debugger support.
Identification	QUERYLEVEL	Returns interface and service level of DSNTPSMP.

Figure 1-5 describes how the Data Studio creates SQL stored procedures on z/OS.

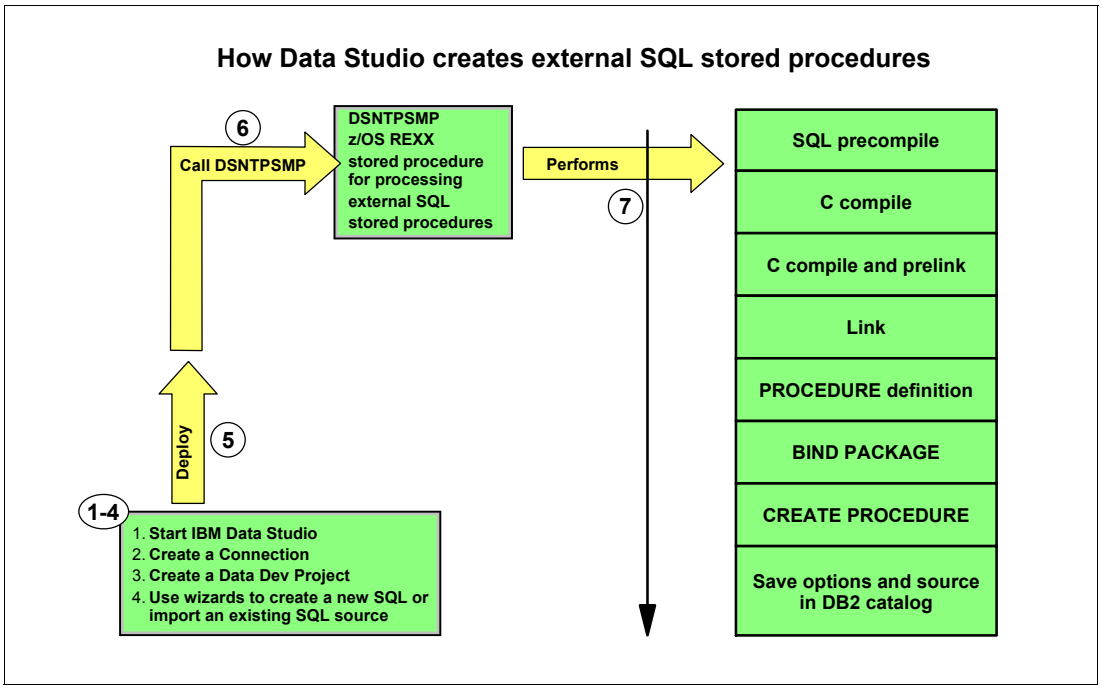

Figure 1-5 How Data Studio creates SQL stored procedures

The steps performed by Data Studio to create external SQL stored procedures are:

1. Launch Data Studio and select a workspace.
2. Create a connection or reconnect to an existing connection to a DB2 server.
3. Create a Data Development Project and set the target connection to the above database server.
4. Create a new SQL stored procedure using the New Stored Procedure wizard. This is a one-page wizard that presents the user with a choice of templates for the new stored procedure.
5. After the stored procedure has been created and edited, the user can choose to deploy the stored procedure using the Deploy wizard.
6. The Deploy wizard calls the DB2-supplied stored procedure, DSNTPSMP.
7. DSNTPSMP performs the following steps to create the SQL stored procedure on z/OS:
   a. SQL precompile
   b. C precompile
   c. C compile and prelink
   d. Link
   e. Bind package
   f. Registers procedure in the DB2 catalog
   g. Saves options

Java stored procedures built on the client

Data Studio builds Java stored procedures on the client side when connected to the DB2 10 server with the IBM Universal Driver. It no longer supports building Java stored procedures on the server side using DSNTJSPP.

Figure 1-6 describes how the Data Studio creates Java stored procedures on z/OS.

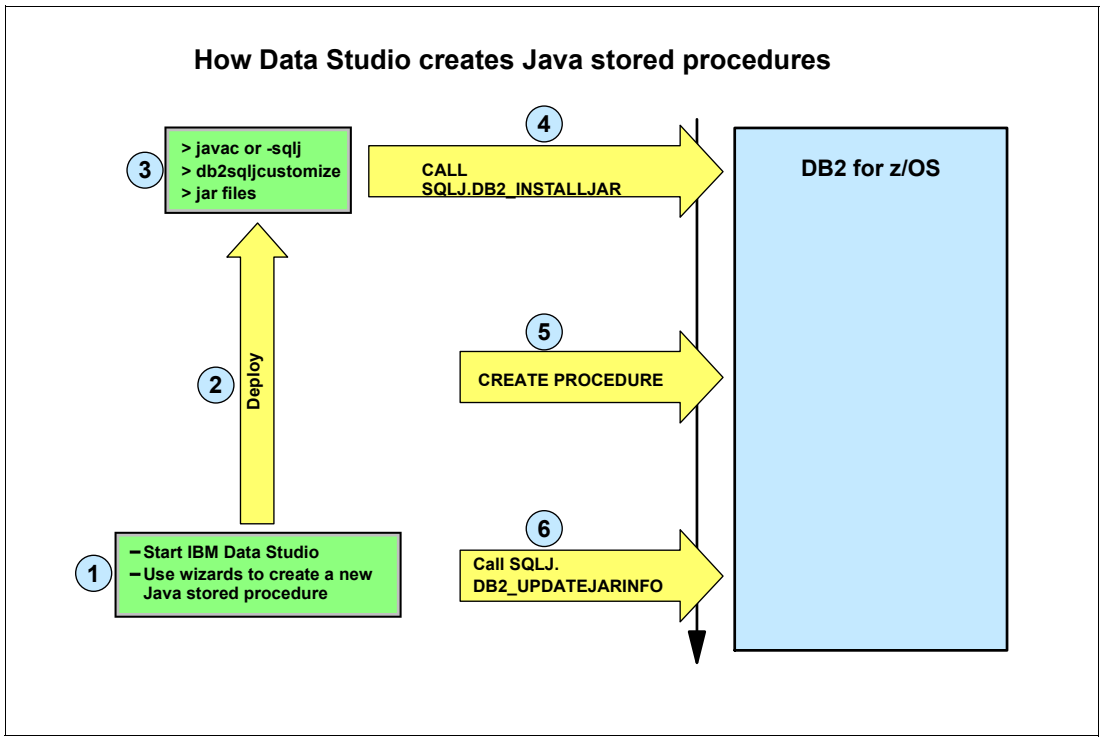

Figure 1-6 How Data Studio creates Java stored procedures

The steps performed by Data Studio to create Java stored procedures are:

1. As in SQL stored procedures, use the New Stored Procedure wizard to create a Java stored procedure. Data Studio generates both the DDL and the Java source.
2. Deploy the stored procedure.

   For Java stored procedures using dynamic (JDBC) SQL, Data Studio issues **`javac`** to compile the stored procedure.

   For Java stored procedures using static (SQLJ) SQL, Data Studio issues **`sqlj`** to translate and compile the stored procedure.

   Data Studio also issues **`db2sqljcustomize`**, which:

   – Updates the `.ser` file created in step 2
   – Optionally binds the stored procedure
3. Then Data Studio issues a **`jar`** command to jar the `.ser`, `.class`, and optionally the `.java` files.
4. Data Studio calls SQLJ.DB2_INSTALL_JAR to install the jar in the server or SQLJ.DB2_REPLACE_JAR to replace the jar when the stored procedure already exists in the server.
5. Data Studio issues the CREATE PROCEDURE DDL to register the stored procedure into the catalog.
6. Data Studio calls SQLJ.DB2_UPDATEJARINFO to copy the Java source into the catalog.

1.5 Navigating through the Data Studio workspace

This section applies to all platforms supported by Data Studio. Data Studio uses the Data Perspective, which contains the following views that can be rearranged or closed. Additional views can also be opened.

- Task Launcher view
- Data Source Explorer view
- Administration Explorer view
- Data Project Explorer view
- Output view
- Editor view

1.5.1 Task Launcher view

The Task Launcher view displays a list of tasks that the user can perform in Data Studio. When a task is selected, Data Studio configures the perspective and views to use for performing the task. Knowing what task you want to do and selecting it is easier than knowing which perspective has the correct views and functions for a specific task (Figure 1-7).

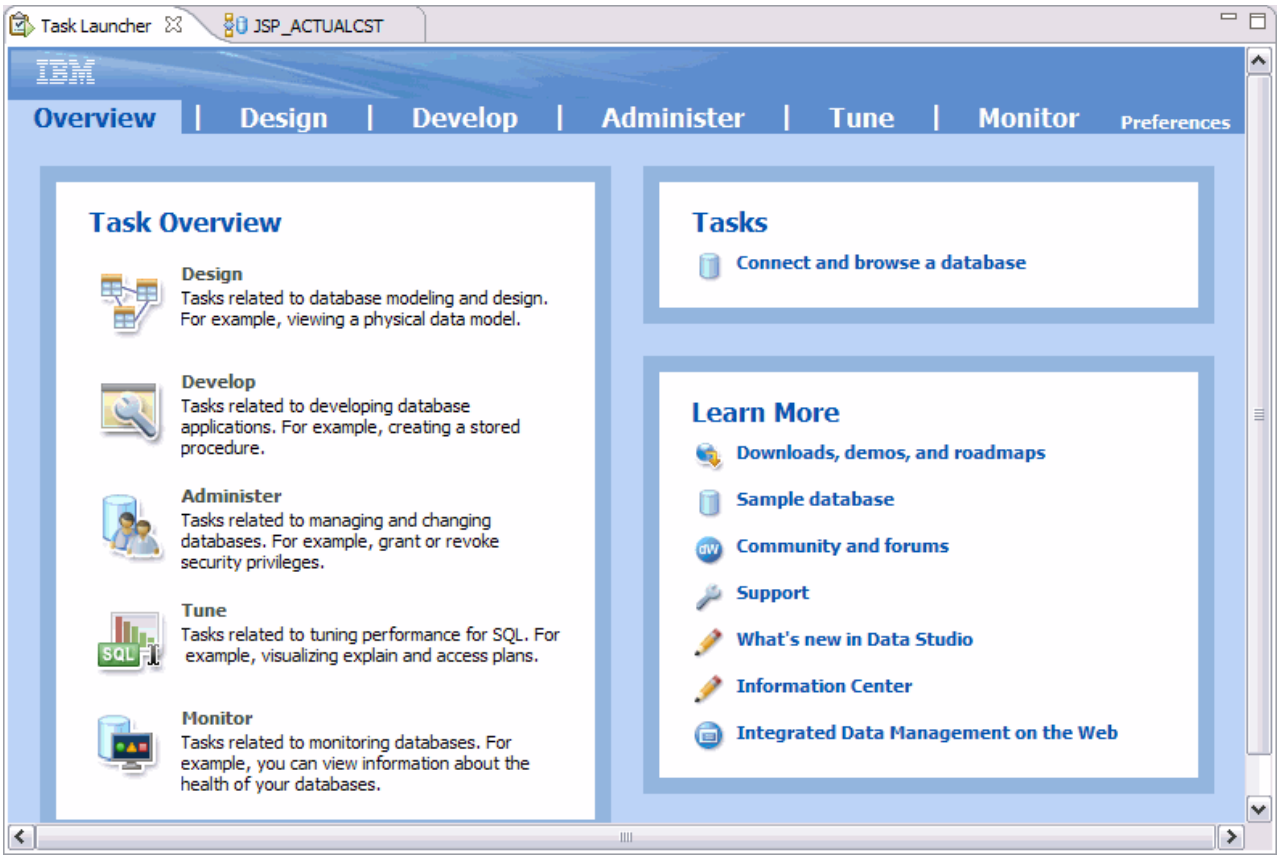

Figure 1-7 Task Launcher view

1.5.2 Data Source Explorer view

The Data Source Explorer displays the connections to database servers that were previously created in the workspace, as well as those in the client DB2's database directory, if a DB2 for LUW is installed. Each connection shows a hierarchical view of the database and the objects within them. The cataloged stored procedures and user-defined functions (UDFs) are listed under a schema folder in this view (Figure 1-8).

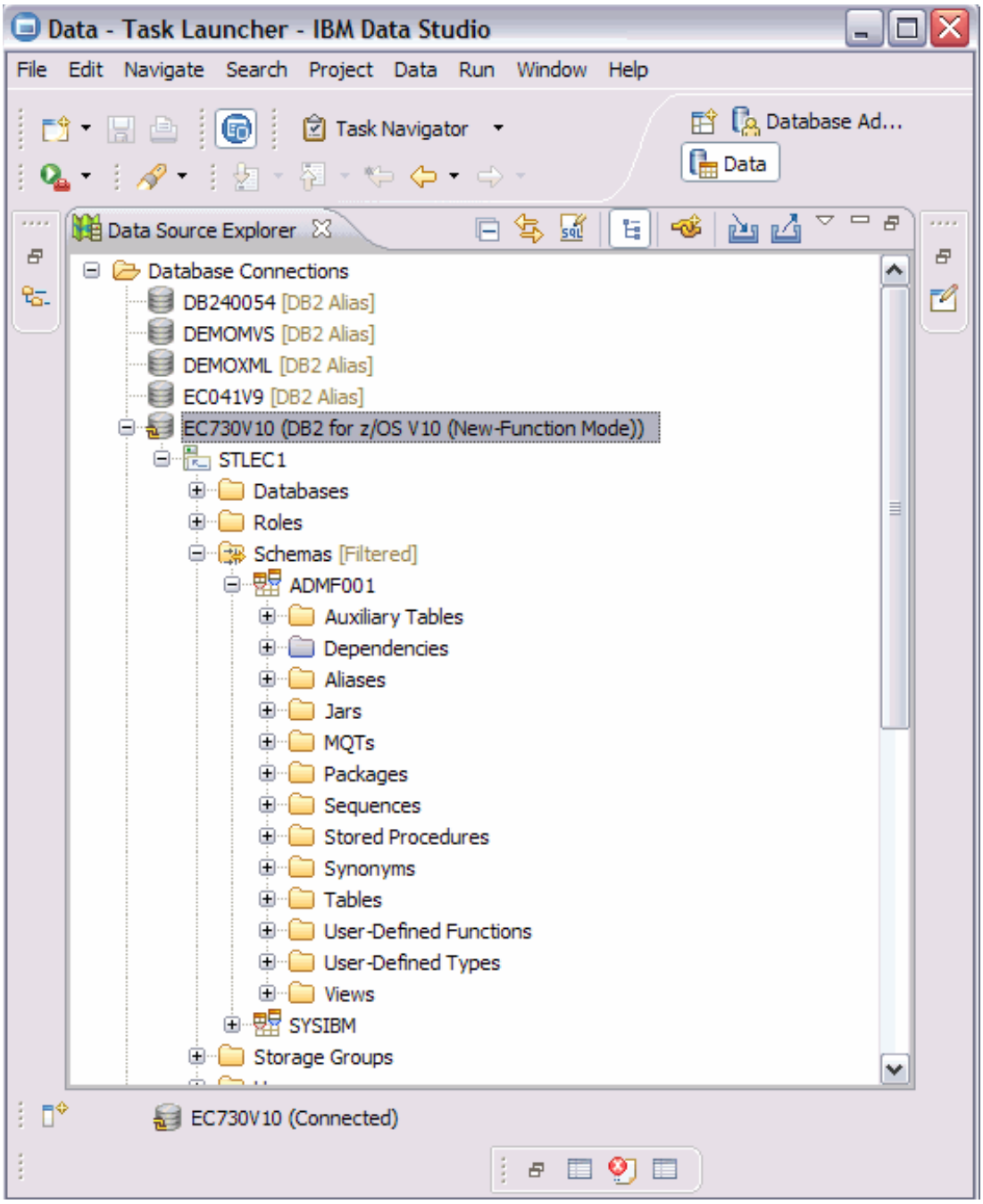

Figure 1-8 Data Source Explorer

In addition to viewing the stored procedures and UDFs that are on the server, the Data Source Explorer allows you to view and work with other database objects such as tables, triggers, views, and so on.

Data Source Explorer menu bar

The Data Source Explorer contains a toolbar at the top of the view with icons for doing the following tasks:

- Collapse all folders.
- Link open editors with content in the Navigator.
- Create a new SQL script.
- Show category.
- Create a New connection profile.
- Export Connections.
- Import Connections.

In addition to the above icons, there is a pull-down menu in the toolbar for additional tasks such as Monitor events and Customize view (Figure 1-9).

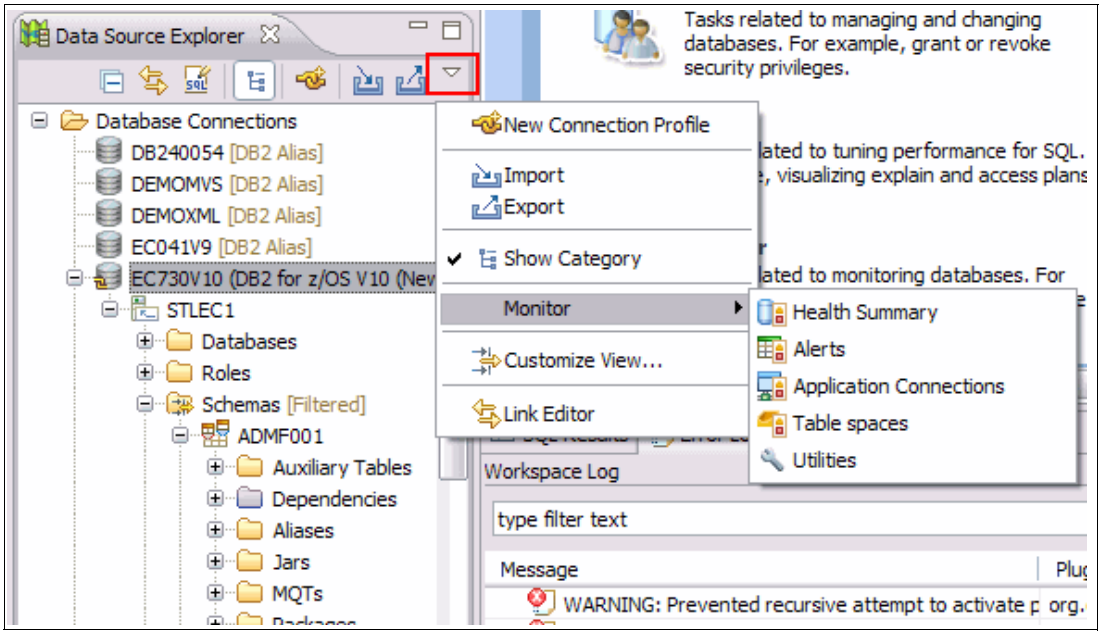

Figure 1-9 Data Source Explorer pull-down actions

In “Using the SQL and XQuery Editor” on page 59, we discuss the Connection Wizard and the SQL Editor in detail.

Context menu actions

Each object in the Data Source Explorer has a context menu that shows the actions that can be performed on this object. To view the actions on a folder or object, right-click the object folder. In this paper, we examine only the context menu actions available for stored procedures. To activate the context menu for stored procedures, expand the database connection and select **Schemas** → ***<your schema>*** → **Stored Procedures**. Right-click the **Stored Procedure** folder and select **Filter** to launch the Filter dialog to view the following context menu actions:

- Filter

  This action allows you to filter what is displayed in the `Stored Procedures` folder in two ways (Figure 1-10).

  - Using an Expression pull-down list allows the user to create clause-like expressions such as NOT LIKE 'AA%'.
  - Select from a list of stored procedure names.

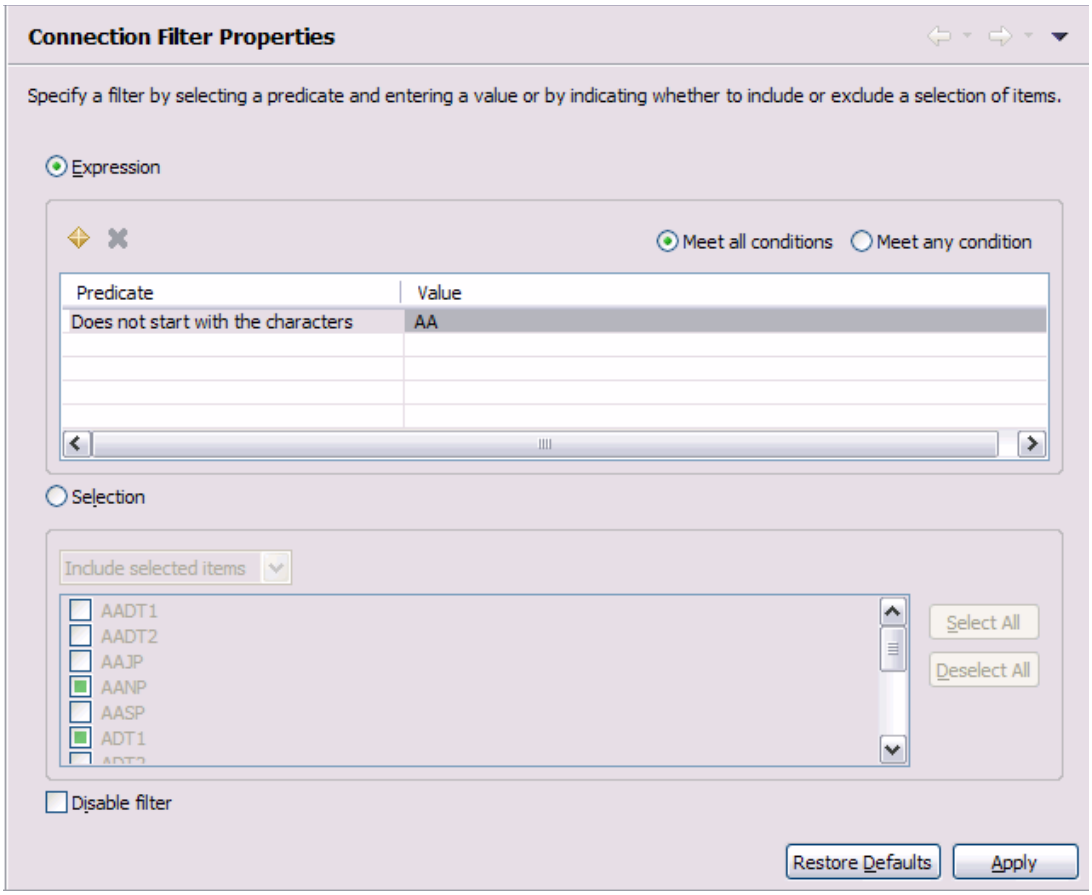

Figure 1-10 Filtering option

- New

  Alternatively to filter, you can select **New**. This action creates a new stored procedure using the SQL Editor. You can only create a new SQL Procedure using this action. Right-click the **Stored Procedures** folder and click **New** → **With SQL Editor** to create a blank editor in the Editor view. You can type your CREATE PROCEDURE statement there. See "Using the SQL and XQuery Editor" on page 59 for details about how to use the SQL Editor.

- Refresh

  This action reloads the latest information for the specific folder, in this case, stored procedures, from the server catalog. Right-click the **Stored Procedures** folder and click **Refresh** to refresh the list of deployed or cataloged stored procedures.

- Deploy

  This action launches the Deploy wizard. Right-click the **Stored Procedures** folder and click **Deploy** to redeploy one, some, or all the stored procedures listed in this folder. If you expand the `Stored Procedures` folder, you can deploy one or mores stored procedures as well. Right-clicking a specific stored procedure and then clicking **Deploy** allows you to redeploy only the selected stored procedure. Pressing Ctrl and clicking several stored procedures and then right-clicking **Deploy** allows you to redeploy the selected stored procedures. Figure 1-11 shows the first page of the Deploy wizard. Details on the Deploy wizard are discussed in 2.5, “Deploying a stored procedure” on page 71.

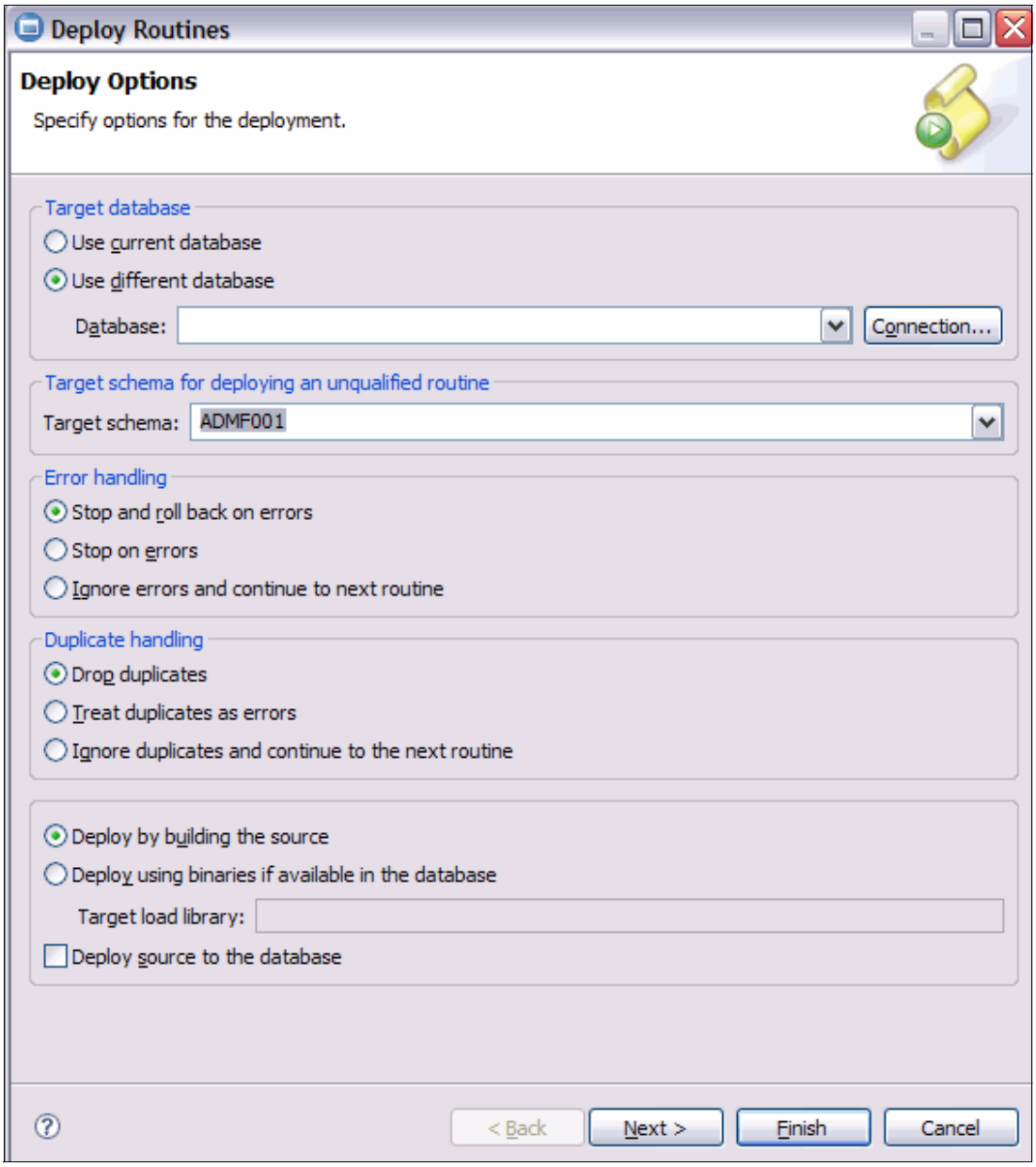

Figure 1-11 Deploy wizard

The context menu for a specific stored procedure shows additional actions that can be taken on the stored procedure. Right-click the stored procedure to see a menu with the following options:

– Run and Run Settings

  These two actions execute the selected stored procedure. Run Settings allows you to preset parameters and execute SQL statements before and after calling the stored procedure.

– Open with SQL Editor

  This action launches the SQL Editor on the Editor view. Data Studio assumes that you are opening the stored procedure for editing. To save your changes, specify a Data Development Project to contain the modified stored procedure.

– Debug

  This action is grayed out if the stored procedure is *not* enabled for debugging.

– Drop

  This action issues a DROP PROCEDURE against the selected stored procedure. A confirmation dialog is displayed before the action is sent to the server.

  Note: This action drops all versions of a Native SQL Stored procedure. To drop a specific version, expand the selected stored procedure and click **Versions**, then select the specific version and activate the Drop action from this version.

– Generate DDL

  This action launches the Generate DDL wizard. In the wizard, you can:

  - Generate the CREATE PROCEDURE DDL.
  - Optional: Generate the associated DROP statement before the CREATE statement.
  - Optional: Generate any associated COMMENT ON and LABEL ON statements.
  - Optional: Generate any GRANT statements based on the current privileges held on this stored procedure.
  - Execute the generated DDL, or save and edit the generated script. You are asked to specify an existing project to contain the generated script. Data Studio looks at the list of existing projects and defaults the project to one that is using the current server or one that is a “best fit” (that is, same operating system, same version).

  Note: For Java stored procedures, Generate DDL does not create a clone of the Java source associated with the Java stored procedure.

Figure 1-12 shows the DDL generated for one of our sample stored procedures.

Figure 1-12 Generated DDL for a stored procedure

– Analyze Impact

  This action examines the SYSROUTINESDEP and SYSPACKAGEDEP catalogs and lists the objects that have dependencies on this stored procedure.

– New version

  This action is only available when the selected stored procedure is a Native SQL stored procedure. This action launches the New Version wizard. This wizard creates a new version of the selected stored procedure and optionally deploys it.

– Refresh

  This action queries the server catalogs and displays all stored procedures in the schema. If the folder is filtered, the filter is applied to the refresh action.

– Query Tuner

  This action launches several sub-actions related to single-query tuning. The target database needs to be configured for tuning either explicitly from the Configure for Tuning menu sub-action or implicitly when you try to Start tuning. We discuss the query tuner support in **3.1, “Additional features in Data Studio” on page 86**.

– Deploy

  This is the same action as in the stored procedure executed against a specific stored procedure.

– Open Visual Explain

  This action browses a SQL stored procedure source for the first SQL Statement and attempts to explain the statement. This action is *not* available for Java stored procedures. Also, this action assumes that the server is configured for single-statement query tuning.

– Copy

  This action creates a clone of the stored procedure model. The corresponding Paste action should be done against a Data Development Project's stored procedure folder.

– Properties

  This action displays and organizes the stored procedure's properties in tabbed pages in the Property Browser.

– Generate pureQuery Code (Optim Development Studio only)

  This action allows you to generate the pureQuery code to call this stored procedure. You need to have an existing Java project and an existing Java program where the pureQuery code is embedded. PureQuery is beyond the scope of this book. However, see 3.2, "Additional features in Optim Development Studio" on page 107, for an overview of Data Studio's pureQuery support.

Details on the above actions are discussed in 2.1, "Getting started with Data Studio stored procedures development" on page 50.

1.5.3 Administration Explorer view

The Administration Explorer view (Figure 1-13) is a hierarchical or flat view of the Data Source Explorer connections.

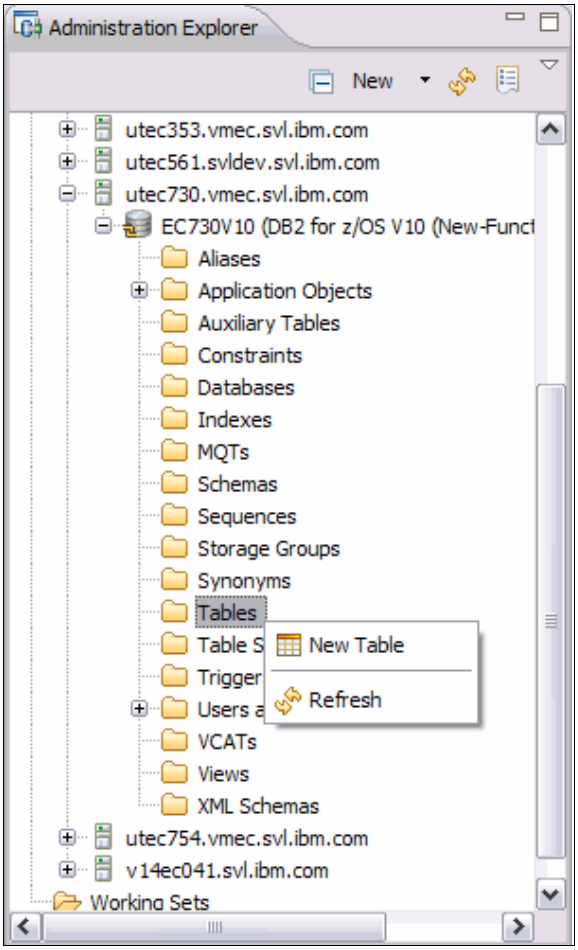

Figure 1-13 Administration Explorer

In the Administration Explorer, against a DB2 for z/OS connection, the subsystem folder expands to a list of object types (a folder) found in the server. In the Administration Explorer, you can do the following types of administration tasks[5]:

- Manage the connection to the entire subsystem.
- For each object type folder (a flat folder), you can create a new object of that type.
- When you click an object type in the Administration Explorer, a list of objects is displayed in the Object Editor.
- In the Object Editor, you can do the following types of administration tasks:
  - Alter, drop, and copy.
  - Generate DDL.
  - Manage privileges.
  - Analyze impact.
  - Add the object to an Overview diagram.

[5] Not all tasks can be performed on all the objects. See Table A-3 on page 139 for specific tasks that you can perform against a specific object.

– For tables, materialized query tables (MQTs), synonyms and views, unload and edit data using the Object Data Editor.

– For tables only, Load data.

The Object Editor

The Object Editor is the main editor used by the Database Administration perspective for managing the database. This editor is launched when an object type (for example, tables) is selected in the Administration Explorer view. The editor then displays an object list of this type.

As the object list can be very large, Data Studio supports filtering of the objects by a specific attribute. For example, in Figure 1-14, the Tables object list can be filtered by schema. The Refresh context menu action refreshes the object list based on the new filters.

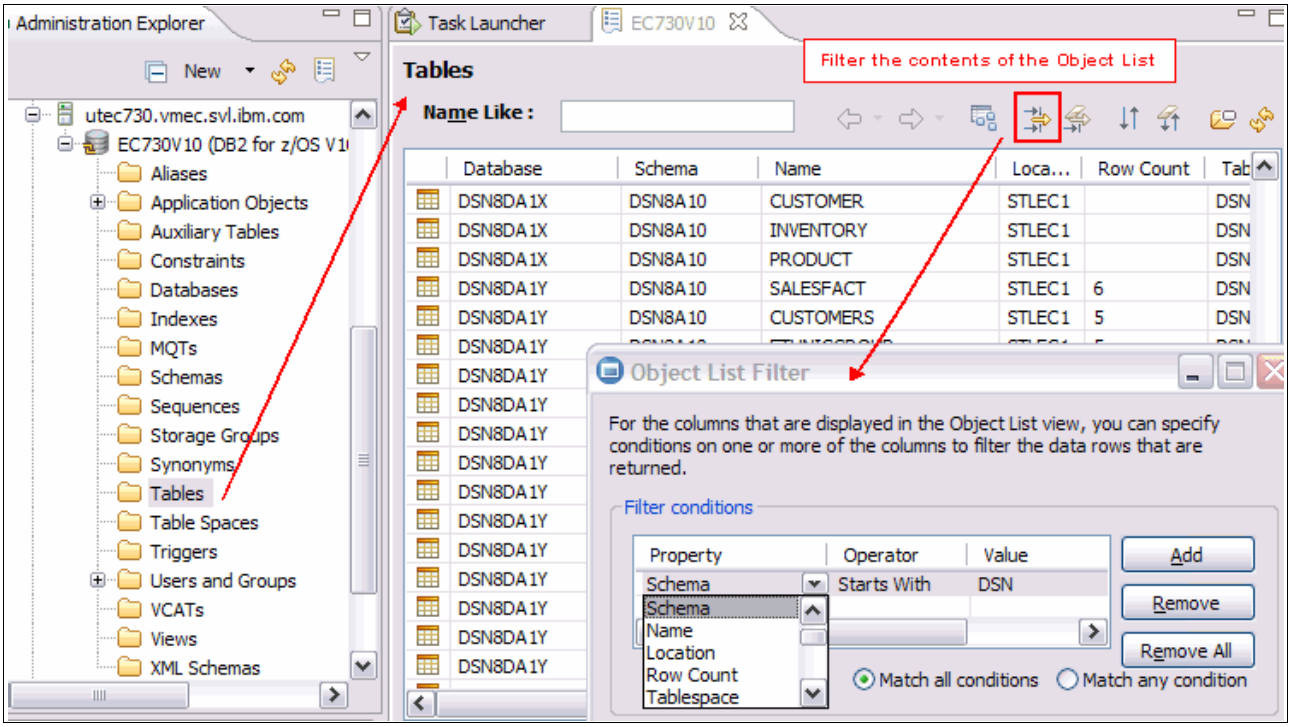

Figure 1-14 Object Editor - Tables object list filtering

1.5.4 Data Project Explorer view

The Data Project Explorer view is the main development view for managing your projects. The main project type for routine and SQL development in the Data Project Explorer is the Data Development Project. In this view, you can:

- Manage multiple projects.
- Target a specific connection to a project.
- Under each project, create and manage objects such as stored procedures, SQL scripts, UDFs, Jars, XML objects, and web services.
- Copy and paste objects from one project to another.
- Share a project between teams.

Each Data Development Project contains the database objects that you can work on, in an object tree structure. Figure 1-15 shows the contents of a Data Development Project.

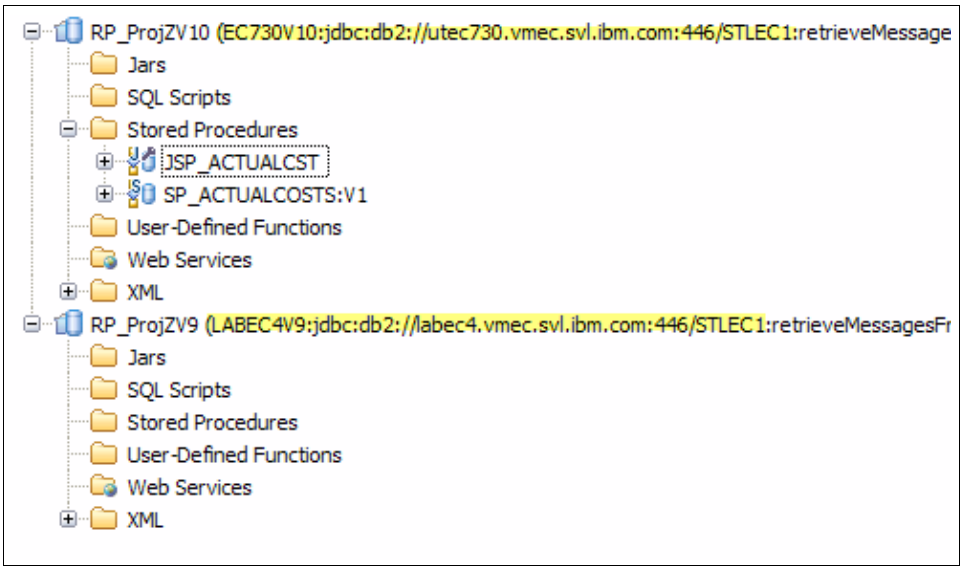

Figure 1-15 Data Project Explorer

Project properties

A set of project properties associated with the Data Development Project can be used to set default values when creating objects within the project. Right-click the project name and select **Properties** to launch the Properties dialog.

- The Database Connection page displays the connection properties of the target connection. In 2.1.2, “Creating a connection profile” on page 51, we discuss how to set the connection properties. This page also allows you to set the current schema (Figure 1-16).
- A discussion of the current schema versus current SQLID is provided in 2.5.1, “The Deploy wizard” on page 71.

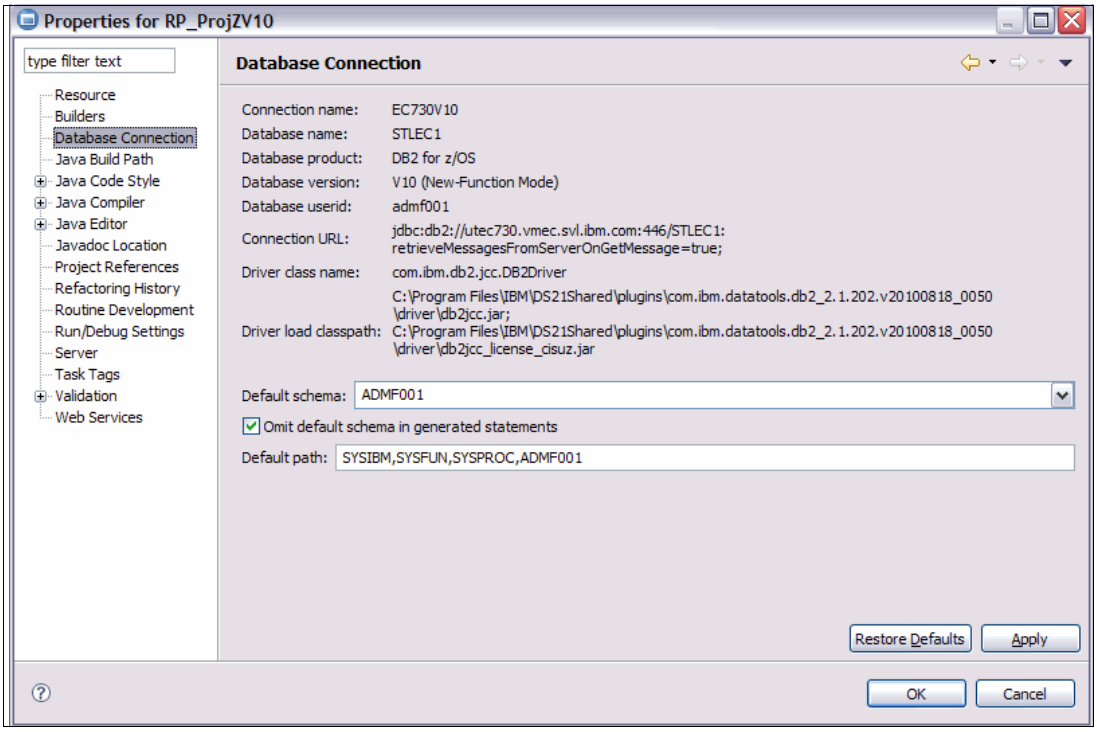

Figure 1-16 Setting current schema

- The Routine Development page allows you to set the JDK level to be used for compiling the Java stored procedures that you create. Additionally, you can set the package owner and build owner for SQL and SQLJ stored procedures on this page (Figure 1-17). Both the package and build owner can be set to secondary authorization IDs.

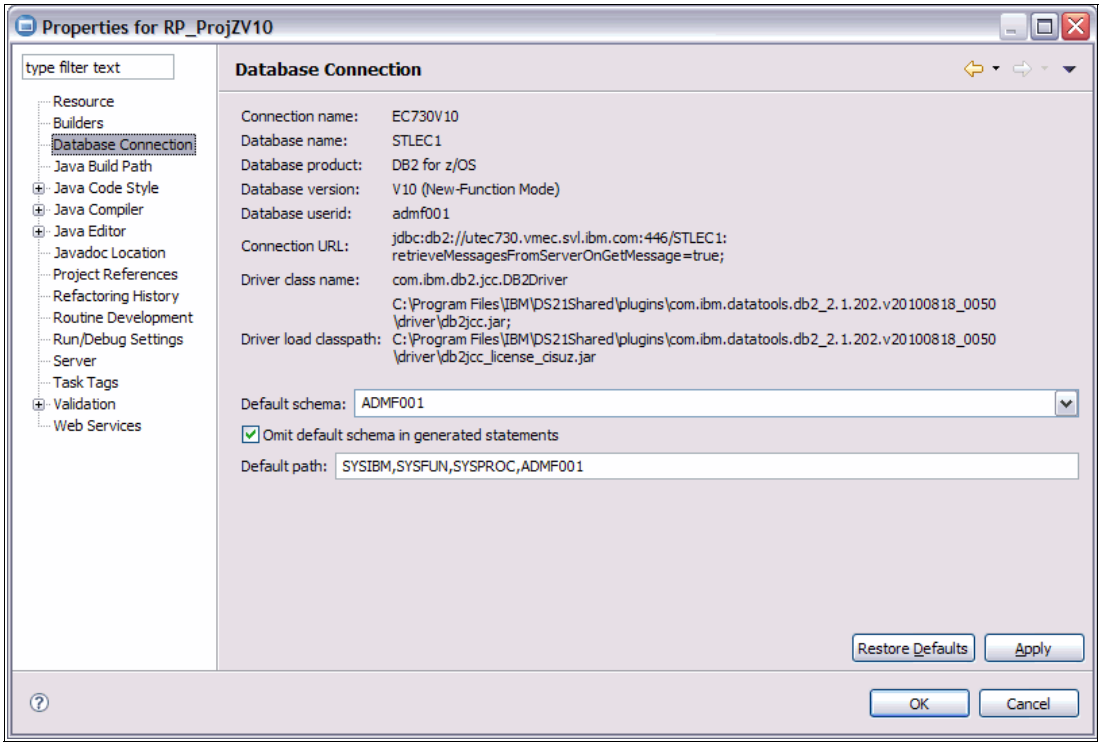

Figure 1-17 Setting package and build owner

1.5.5 Output view

The Output view is used for displaying messages and execution results. Any output-related view in Eclipse is added as a tab to the Output view. We examine five tabs in this view.

Properties view

In the Data Source Explorer, when a specific object is selected, the properties of the object are retrieved and displayed on the Output view's Properties tab (Figure 1-18).

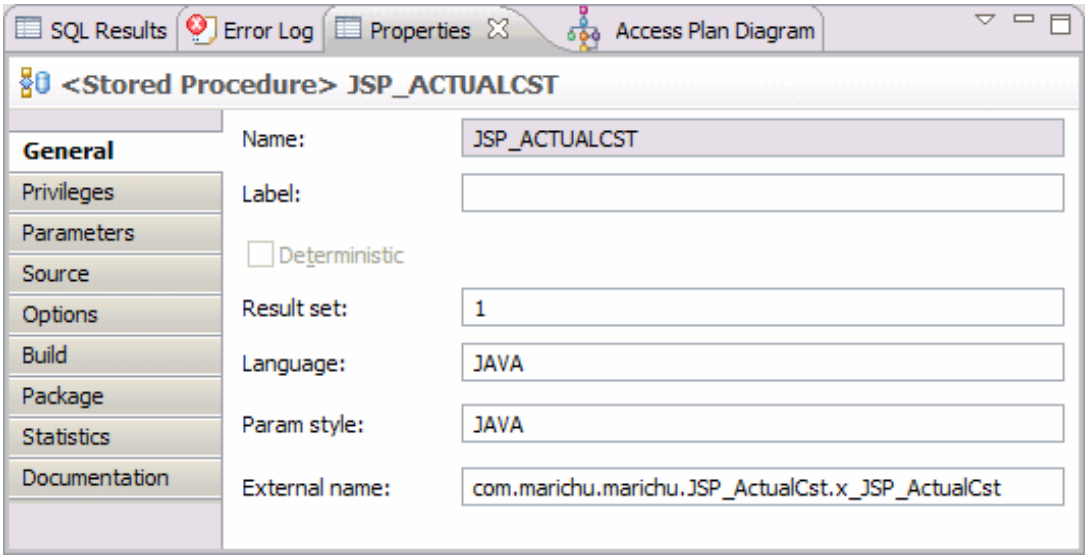

Figure 1-18 Property Browser for stored procedure

The Properties tab, in turn, contains multiple tabs that group the stored procedure's attributes into:

- General
  - Name
  - Label (on)
  - Result sets returned
  - Language
  - Parameter style
  - External name
  - Deterministic/Non Deterministic
- Parameters
  - Parameter type
    - IN
    - OUT
    - INOUT
  - Parameter name
  - Parameter data type
- SQL or Java source
- Privileges
  - Grantee
  - Grantee type, privilege, Grantor, with grant option
- Procedure options
  - Specific name
  - Package ID, data access type
  - Collection ID
  - ASUTIME
  - External Security
  - Stay Resident

  - Program Type
  - Commit on Return
- Build options
  - WLM Environment
  - Build Utility
  - Build Owner
  - Precompile
  - Compile
  - Prelink
  - Link
  - Bind options
- Documentation: The text supplied in the *Comment on* statement for this stored procedure is displayed here

The fields in the Property Browser are READ-ONLY. To modify an SQL or Java stored procedure, you need to *open* the procedure with the SQL Editor, then redeploy it.

Also, the Package and Statistics tabs are not used. To view the package information related to a specific stored procedure, expand the Packages folder for this schema and select the associated package ID from the list. The Properties tab is refreshed with the package information.

The SQL Results view

The SQL Results tab of the Output view is used to report execution status and messages for SQL statements and stored procedures. Figure 1-19 shows an example of the SQL Results view.

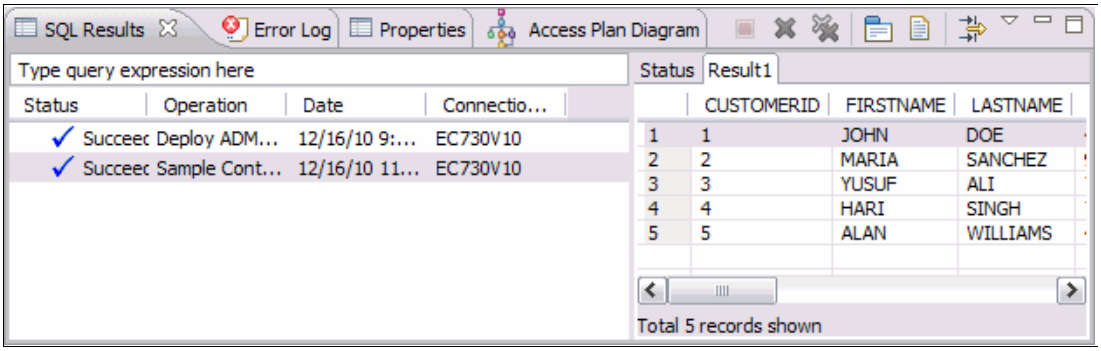

Figure 1-19 SQL Results view

The SQL Results view contains two panes:

- The Status History
- The Current Output

The Status History pane is in turn divided into four columns:

- The Status column shows the current status of each action, such as error or success.
- The Operation column shows the type of action, such as deploy, run, export, import, and so on. If an operation involves multiple steps, as in the case of a script, the a higher level node is displayed. The node can be expanded to see the individual steps or statements executed.
- The Date column shows the timestamp of when the operation was done.
- The Connection column gives the target connection name.

The Status History pane displays individual actions or a group of actions (as in when a SQL script is executed). When a specific action in the Status History pane is selected, the Current Output pane is populated with the output details.

In the Current Output pane, you can select different tabs to view different kinds of output:

- The Status tab displays the overall status of the action.
- The Parameters tab displays the name and values of the parameters after a stored procedure is executed. This tab is only shown when the action is run or debug of a routine.
- The Results tab displays the result sets of an action. This tab is only shown if an action returns one or more result sets. Each result set is a separate tab.

You can remove an entry or all entries from the Status History list by right-clicking the action and clicking **Remove** or **Remove All**.

Access Plan Diagram view

The Access Plan Diagram is added to the Output view when you click **Open Visual Explain** against an SQL statement in a SQL script or SQL stored procedure. Data Studio presents a wizard to determine the default special registers to use for the Explain statement as well as the default schema of the Explain tables to store the results of the Explain.

Problem view

The Problem view displays a list of errors and warnings encountered when building a project and is automatically launched when errors are detected. You can click each error and examine the error details of the error. Double-click the message to locate the error in your Java source.

The Filter button lets you configure the view. You can filter by element, such as the class that you are editing or the working set, the type of problem (such as. java problems, buildfile problems, and so on), and severity.

Error Log view

The Error Log view displays Data Studio and Eclipse exceptions, errors, and warnings. This log captures information about the plug-in that caught the exception. This information is high-level and limited to the plug-in name, the line number in the plug-in, and the timestamp of each successive failure.

The Error Log view is not automatically displayed when Data Studio is first launched. To display this view, click **Window** → **Show View** → **General** → **Error Log**. The Error Log view is persisted in the workspace's `.metadata` folder.

You can manage the entire log or an entry in the Error Log through the context menu actions (Figure 1-20).

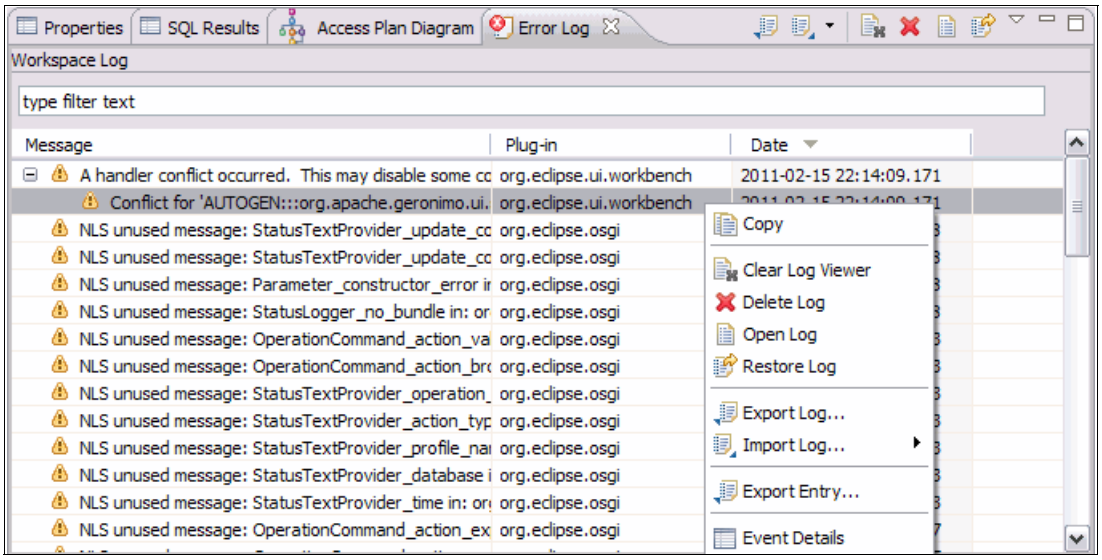

Figure 1-20 Error Log view

1.5.6 Editor view

Data Studio uses this view to present different editors to the user. The editors that are related to routine development are:

- Routine Editor
- Integrated Query Editor or SQL Editor
- Java Editor

All these editors share the same Editor view space. Any object being viewed shows up as a tabbed page in the Editor view.

The Routine Editor

This editor is launched in several ways:

- From the Data Project Explorer, double-click a stored procedure.
- From the Data Project Explorer, right-click a specific stored procedure and select **Open**.
- At the end of the New Stored Procedure or New Version wizards, the Routine Editor is refreshed with the generated stored procedure DDL.

Use this editor for viewing and changing the source code and configuration options of a stored procedure that you are working on in the Data Project Explorer. Figure 1-21 and Figure 1-22 on page 45 show the contents of the Routine Editor's Java and DDL tabs for a Java stored procedure. SQL stored procedures have only one tab, DDL.

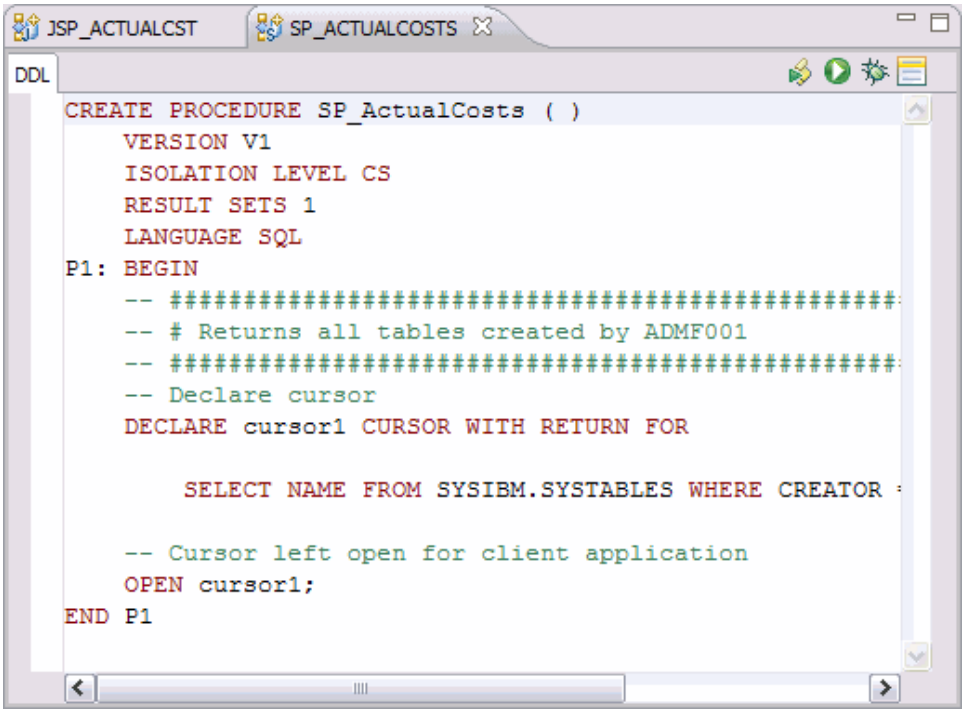

Figure 1-21 Routine Editor - DDL tab

The Routine Editor's DDL tab is a rich editor that supports cut, copy, paste, find and replace, menu and keyboard shortcuts, and syntax highlighting. You can change the default *look and feel* of this page in the Preferences. See 2.1.1, "Starting Data Studio for the first time" on page 50.

To edit the DDL of an SQL stored procedure:

1. In the Data Project Explorer, right-click the routine that you want to modify and click **Open**. The routine DDL displays in the Routine Editor view.
2. Edit the DDL. You can:
   - Change or add SQL statements directly in the editor.
   - Press Ctrl+Space bar to launch Content Assist.
3. To save your changes, you can:
   - Click **File** → **Save Object** or **File** → **Save All**.
   - Click the Save icon (floppy disk image) or press Ctrl+S.

For Java stored procedures, the Routine Editor contains a second tab, the Java tab. The Java source of the Java stored procedure is displayed here. It is also editable. Certain changes to the Java source can impact the DDL. Data Studio V2.2.1 attempts to synchronize changes in the Java source with the DDL, but not vice versa. When the modified Java source is saved, Data Studio requests the user to select a method to synchronize with. Click **Sync** to complete this action.

After saving your changes, Data Studio replaces the persisted resource corresponding to this stored procedure in the workspace. However, to replace the object in the server, you need to

deploy or redeploy the stored procedure. Depending on your deploy options, Data Studio either drops the old routine from the database and creates a routine that reflects the changes that you made or it alters it. Changes to the SQL procedure body rarely cause the procedure to be dropped. Where possible, changes to the source code of SQL stored procedures result in an ALTER command rather than a DROP command.

Finally, the Routine Editor is also used by the Debug Perspective when debugging SQL routines.

Integrated SQL and XQuery Editor

This editor is launched when creating or editing SQL scripts. We discuss this more in 2.1.5, “Creating SQL statements and scripts” on page 57.

Java Editor

The Java Editor can be launched from the Data Perspective, the Debug Perspective, and the Java Perspective.

- From the Data Project Explorer:
  a. Right-click the stored procedure and select **Open**. The stored procedure DDL is displayed in the Routine Editor view.
  b. Click the **java** tab. The Java source is displayed (Figure 1-22).

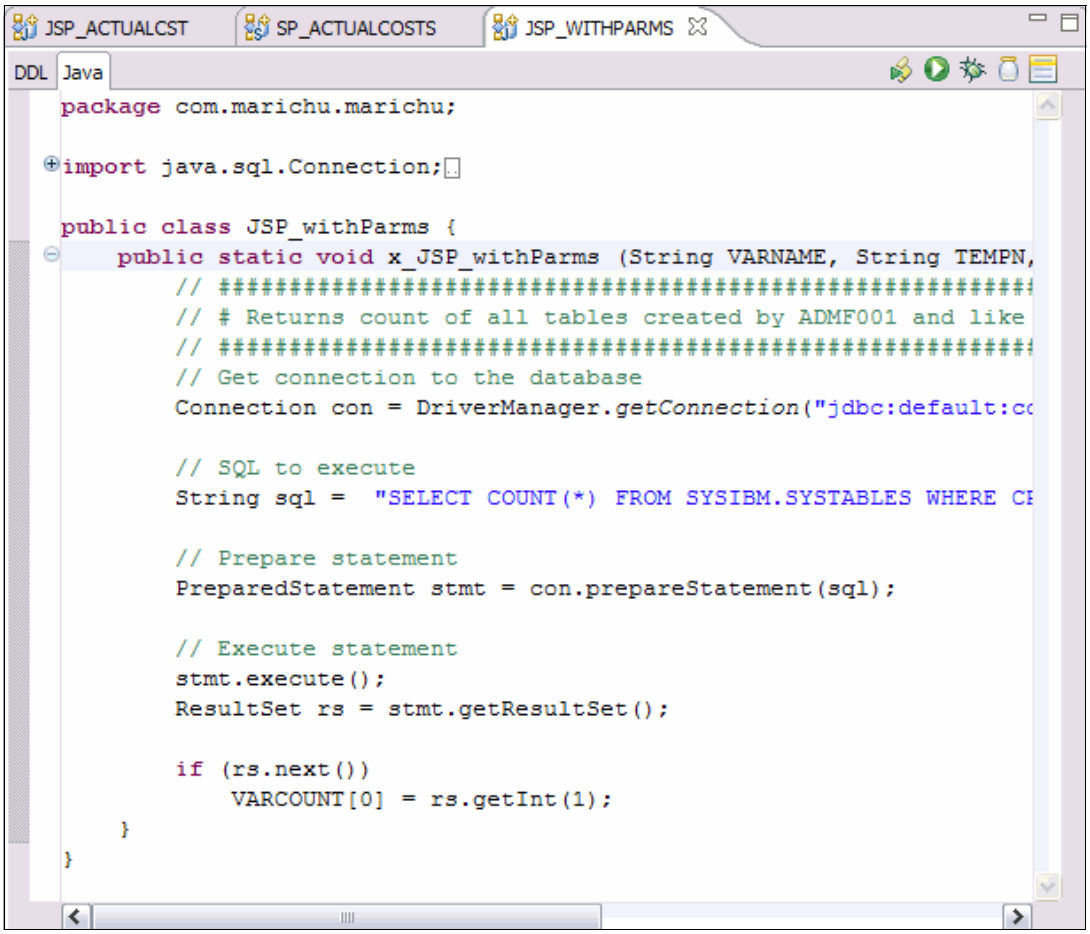

Figure 1-22 Routine Editor - Java tab

- When a Java stored procedure is deployed with debug enabled, you can debug the Java stored procedure from Data Studio IDE or Optim Development Studio. The Debug Perspective is automatically launched and the Java source is displayed in the source area.
- From the Java Perspective, select the Project Explorer:

  a. Open the package or the default package to the `.java` or `.sqlj` file.
  b. Double-click this file.

In Optim Development Studio, when working with pureQuery, the Java editor is launched at the end of the New Java Class wizard.

Java routines built by the Data Studio conform to the SQLJ Routines specification. Java objects are defined in the catalog table with LANGUAGE JAVA and PARAMETER STYLE JAVA. Java objects must follow these rules:

- The method that is mapped to the object must be defined as a public static void method.
- The object must receive input parameters as host variables.
- Output and InOut parameters must be set up as single element arrays.

When editing your source in the Java editor, your changes are dynamically compiled and errors reported immediately. When you add arguments to the .java or .sqlj main method and then save the changes, they are reflected as input parameters in the DDL tab.

As in SQL stored procedures, changes to the source code of Java stored procedures only change the object in the workspace. To replace the object in the server, redeploy the Java stored procedure.

To close any object, open it in the Editor, click **File** → **Close Object** or **File** → **Close All**, or click the **X** next to the procedure name in the Routine Editor.

Export wizard

Use the Export wizard to export routines from your current project to the file system for later deployment. Data Studio supports exporting an entire project or just the stored procedures. You might want to export the entire project to the file system, which can then be imported into another workspace. In this book, we discuss exporting stored procedures only.

You can export a specific stored procedure or several stored procedures at a time. To export routines using the Export wizard:

1. In the Data Project Explorer, right-click the **Stored Procedures** folder and select **Export**.
2. On the Selection page, click the check boxes for the stored procedures that you want to export. You can also click **Select All** to select all stored procedures in this folder. Click **Next**.
3. On the Target and Options page, type the file name and directory where the exported script is sent. You can optionally click **Browse** to launch the file browser.
4. Click **Next** or **Finish**. The wizard exports the selected routines to the file name and directory that you specified.

Note: Optim Development Studio supports enterprise deployment through its implementation of server profiles (see 3.2.3, “Server profiles” on page 110) and deployment groups (see 3.2.4, “Deployment groups” on page 111).

Import wizard

Use the Import wizard to import routines to your project. To open the Import wizard:

1. In the Data Project Explorer, right-click the **Stored Procedures** folder and click **Import**. The Import wizard is launched.
2. On the Import wizard's Source page (Figure 1-23), select the location of the object or file that you want to import. You can import from the file system or from another project in this workspace.

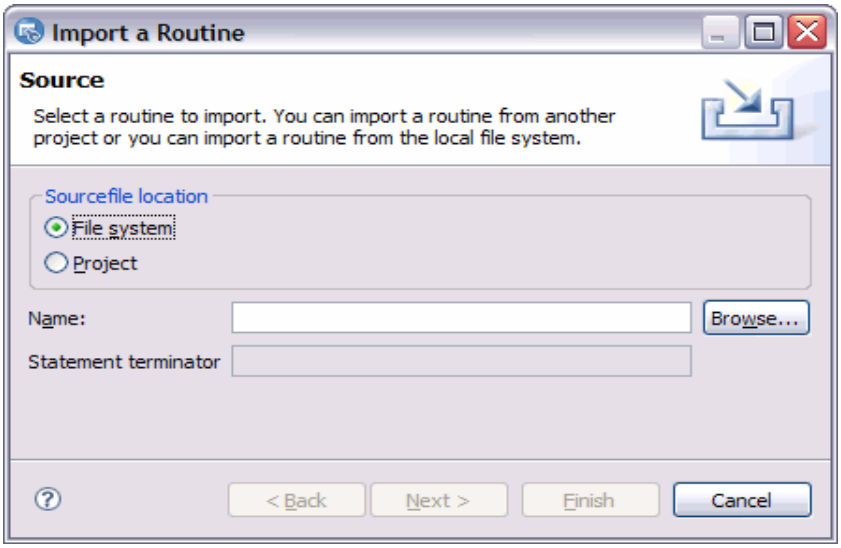

Figure 1-23 Import wizard

3. Click **Browse** to select the directory or project that contains the stored procedure. A file browser is launched. Click **OK** after selecting the stored procedure.
4. If you are importing an SQL stored procedure, you can optionally set the statement terminator used in the imported file. Click **Next**.
5. The next page shows the "discovered" entry points for this stored procedure. You can verify whether the imported stored procedure is correct. Click **Next**.

> **Note:** If there are multiple CREATE PROCEDURE statements in the imported file, only the first CREATE PROCEDURE statement is processed and imported by Data Studio.

6. The next page shows the parameters of the imported stored procedure. You can change the parameter data types of imported Java stored procedures. You cannot change the parameters of imported SQL stored procedures. Click **Next**.
7. The next page of the Import wizard allows you to specify import options. You can opt to replace stored procedures with the same name and parameter signature that already exist in the project. Click **Next** or **Finish**.

Deploy wizard

Use the Deployment wizard to deploy routines to a target database. The target database must be compatible with the database for which the object was created.

The wizard consists of four steps. First, select the target database and enter your user ID and password. Next, select the routines that you want to deploy. Then specify deployment and

error handling options. A summary of the deployment options that you specified in the wizard is displayed in the last page.

To deploy routines to a target database using the Deployment wizard, open the Deployment wizard and:

1. In the Data Project Explorer, select a project → **Stored Procedures** → select a stored procedure.
2. Right-click this stored procedure and click **Deploy**. Alternatively, from the Routine Editor, you can click the Deploy icon.
3. Complete the necessary steps of the wizard.
4. Click **Finish**. The wizard deploys the routines to the target database.

Menu and Task bar

The Data Studio menu bar includes several selections (Figure 1-24).

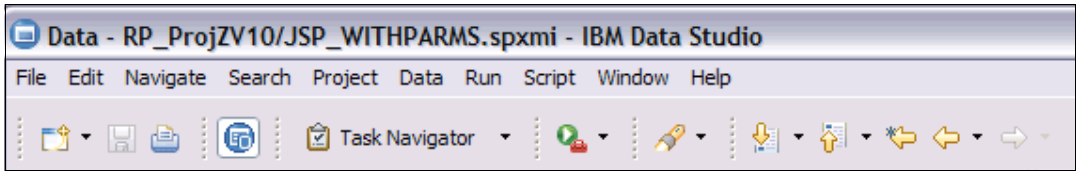

Figure 1-24 Menu and Task Bar

The main selections used in the tooling are:

- File: Use this menu item to save or close objects that you are currently editing.

  You can also create new objects from this menu. Eclipse also allows you to switch to a different workspace from this menu.
- Edit: Use this menu item to work with the object that you are currently editing.
- Project: Use this menu item to open or close a project and to edit the project's properties. See "Project properties" on page 36 for details on what the properties are.
- Script: Use this menu item to issue *Run SQL* against the currently selected script file.
- Window: Use this menu item to:
  - Open additional Data Studio views.
  - Open, customize, or reset the Data Perspective (or any other perspective).
  - Launch the Preferences dialog, where you can set various default items. See "Configuring preferences" on page 51 for more details about setting preferences.
- Help: Use this menu to display online help and product information, and to open the Information Center.

2

Developing stored procedures with Data Studio

In this chapter we show how to use Data Studio for developing stored procedures with DB2 10 for z/OS. We define the environment and then proceed to create a stored procedure, then modify, import, deploy, and execute it.

This chapter contains the following sections:

- Getting started with Data Studio stored procedures development
- Creating a new stored procedure
- Modifying the stored procedure
- Importing a stored procedure
- Deploying a stored procedure
- Executing a stored procedure

2.1 Getting started with Data Studio stored procedures development

This section describes the steps for developing stored procedures with Data Studio. We discuss:

- Starting Data Studio for the first time
- Creating a connection profile
- Creating a Data Development Project
- Creating SQL statements and scripts

In 2.2, “Creating a new stored procedure” on page 60, after we have created our connection, projects, and scripts, we continue our discussion about creating, building, and executing our stored procedure.

2.1.1 Starting Data Studio for the first time

We start Data Studio by selecting **Start** → **Programs** → **IBM Data Studio** → **Data Studio V2.2.1**.

The first time that Data Studio is started, a default workspace directory, named `workspace`, is created for you in `C:\Documents` and `Settings\Administrator\workspace`. You can click **Browse** to launch the File dialog and point to a different directory. The workspace is the main container for all the resources that you use.

Figure 2-1 shows the Workspace Launcher for Data Studio.

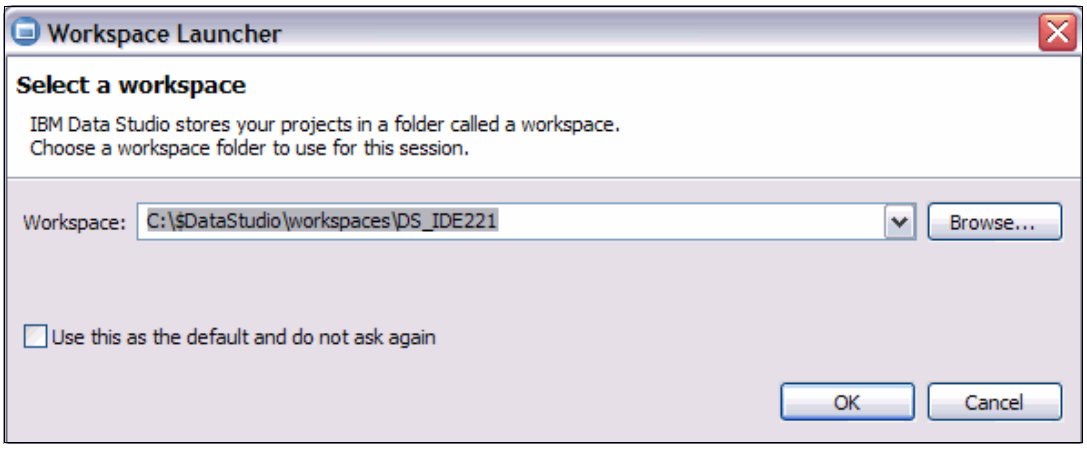

Figure 2-1 Select a workspace

The next window that appears is the Data Studio Task Launcher window. To close the Task Launcher window, click the **X** in the top right corner of the title bar.

The new window presented is the Data Perspective, which is the default perspective when the Data Studio IDE or Optim Development Studio is launched. For Data Studio stand-alone, the default perspective is the Database Administration perspective. You can change the perspective by selecting the title bar **Window** → **Open Perspective**, and selecting from the list presented.

Configuring preferences

Each workspace has a set of preferences that is stored in an Eclipse resource. To view the preferences specific to stored procedures, click **Window** → **Preferences** → **Data Management** → **SQL Development**. Click the following folders to set specific preferences for:

- SQL Results View Options
  - Deploy window rendering/display mode
  - Maximum number of rows to retrieve and maximum number of rows to display
  - Default display for NULL
- Routines → Deploy Options
  - The JDK level used when generating and compiling Java stored procedures
  - The location of the SQLJ translator used for translating SQLJ stored procedures
  - Deploy options for Java, External SQL, and Native SQL stored procedures
- Routines → Process
  - Commit setting.
  - Save files after build.
  - Set tracing on.
- Routines → Templates
  - View or create routine templates.
  - Import a master template.

You can also set the Unified Debugger preferences in the preferences: **Run/Debug** → **Routine Debugger** → **IBM** to decide on the location of the Session Manager: on the server, on the client (built-in), or launched separately in the server or on another workstation.

Session Manager information for the Unified Debugger is in 4.1, “The Unified Debugger” on page 116.

2.1.2 Creating a connection profile

A connection profile is a Data Studio object that describes a specific connection to a server. So, for example, user A creates a connection profile, DB2z10_A, to connect to his DB2 10 subsystem. User B creates another connection profile, DB2z10_B, to connect to the same subsystem. User A has SYSADM authority and can access any of the databases in the subsystem and create objects in any schema. User B has only BINDADD and schema privileges to certain schemas.

The New Connection Profile wizard is launched when you click the New Connection Profile icon in the Data Source Explorer (Figure 1-8 on page 28). The wizard can also be launched by right-clicking **Database Connections folder** and clicking **New**. The wizard is also embedded in the Deploy and New Project wizards. Figure 2-2 shows the first page of the wizard.

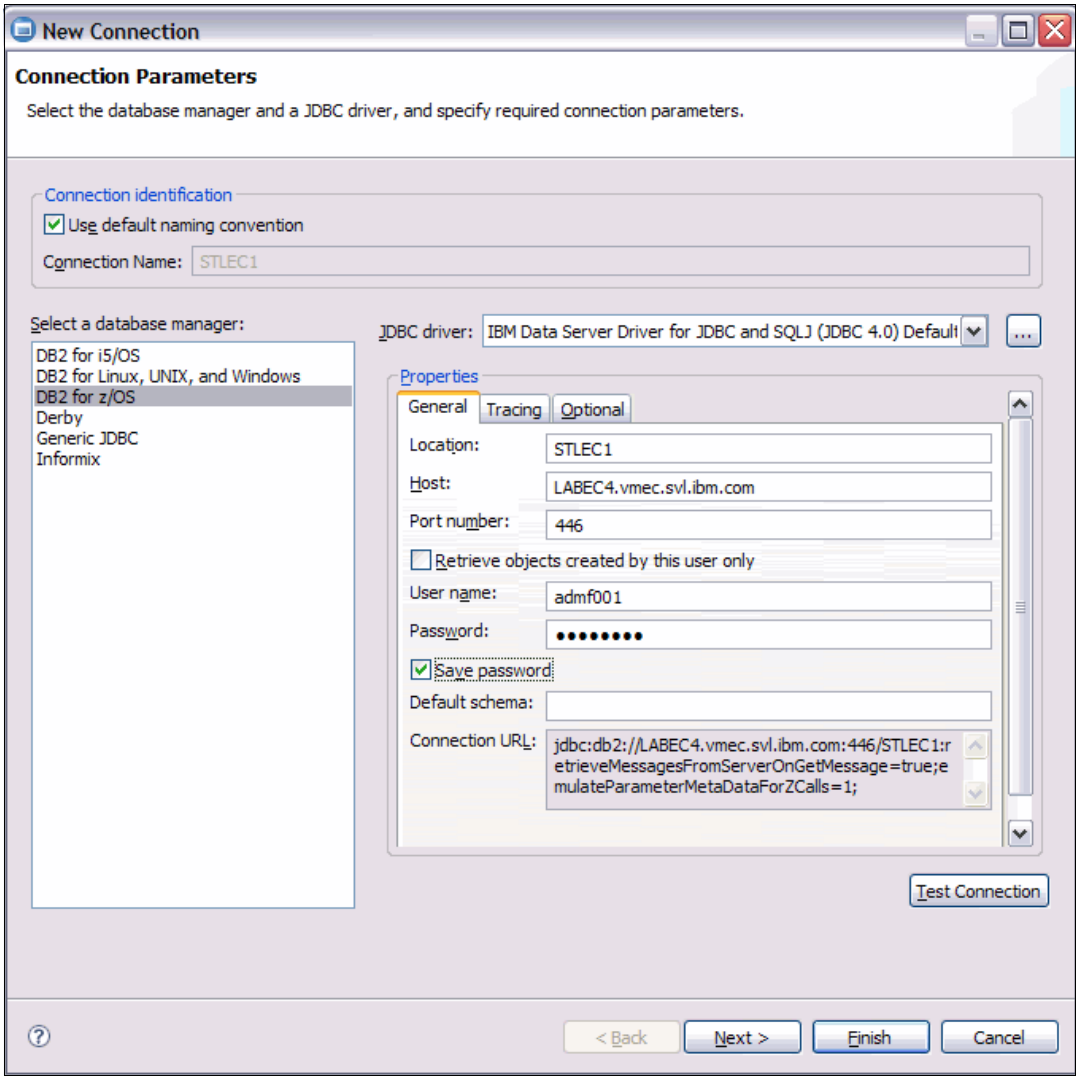

Figure 2-2 New Connection Profile wizard

You fill in the location, host, and port number from the DDF information for this subsystem (Figure 2-3).

```
LOCATION           LUNAME            GENERICLU
STLEC1             USIBMSY.SYEC1DB2  -NONE
TCPPORT=446   SECPORT=0     RESPORT=5001  IPNAME=-NONE
IPADDR=::9.30.222.229
SQL    DOMAIN=UTEC730.vmec.svl.ibm.com
DSNLTDDF CURRENT DDF OPTIONS ARE:
PKGREL = COMMIT
DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Figure 2-3 Display DDF output

The Connection Profile wizard contains three tabs:

- General
- Tracing
- Optional

The General tab

This tab contains the name of the JDBC driver to use to connect to the server. Data Studio is delivered with a version of the IBM Data Server driver for JDBC and SQLJ[1] in <*Install directory*>/plugins/com.ibm.datatools.db2_<*some version*>\driver. The driver and license files are:

- db2jcc.jar and db2jcc_license_cisuz.jar for JDBC 3.0
- db2jcc4.jar for JDBC 4.0

The fields on the first page of the wizard allow you to enter the following information:

- Connection name: The default is the location name.
- JDBC driver: This is a combo box that is initialized with the default driver, IBM Data Server Driver for JDBC and SQLJ (JDBC 4.0). IBM strongly suggests using this driver or the JDBC 3.0 version of this driver. This driver uses the server authentication.

  You can select another kind of driver from the JDBC driver's pull-down list. The current list contains:

  - IBM Data Server driver for JDBC and SQLJ (JDBC 4.0) using Kerberos security
  - IBM Data Server driver for JDBC and SQLJ (JDBC 4.0) using LDAP
  - IBM Data Server driver for JDBC and SQLJ Default
  - IBM Data Server driver for JDBC and SQLJ using Kerberos security Default
  - IBM Data Server driver for JDBC and SQLJ using LDAP
  - Other Default Driver

  If you choose **Other Default Driver**, you need to provide the JDBC driver class name, class location, and connection URL.
- Location: Enter the DB2 for z/OS location ID.
- Host: Enter the domain or FTP address of your DB2 for z/OS server.
- Port number*:* Enter the port number of your DB2 for z/OS server.
- JDBC driver class: When using the IBM Universal driver, this is pre-filled with the value com.ibm.db2.jcc.DB2Driver.
- Class location: When using the IBM Data Server driver for JDBC and SQLJ (JDBC 4.0), this is pre-filled with the location of the license jar files installed with your Data Studio.
- Retrieve objects created by this user only: Check this box if you want to work only with objects that you created.
- User name: Enter your DB2 for z/OS login authorization ID.
- Password: Enter the password associated with the above user ID.
- Save password: Check this box if you want to use this ID and password every time that you connect to this server.
- Connection URL: Data Studio composes this as you enter values for the location, host, and port number. It additionally adds default JDBC properties.

The Test Connection button allows you to test the connection using the fields that you entered.

[1] Also known as the IBM Universal driver.

Tracing

The JDBC tracing options can be set in the connection URL by selecting the trace levels on this tab of the New Connection wizard. For more information about tracing levels and other problem determination tools for Data Studio, see:

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0706scanlon/

Optional

Data Studio allows the user to append certain JDBC connection properties in the connection URL. You need to be familiar with the specific JDBC connection properties that you want to set, as well as the format of the specification.

Click **Finish** to complete creating the connection.

2.1.3 Editing the connection

After you create the connection, you can modify the connection properties by selecting the connection then right-clicking **Properties**. Data Studio opens a dialog with the Driver Properties page selected (Figure 2-4).

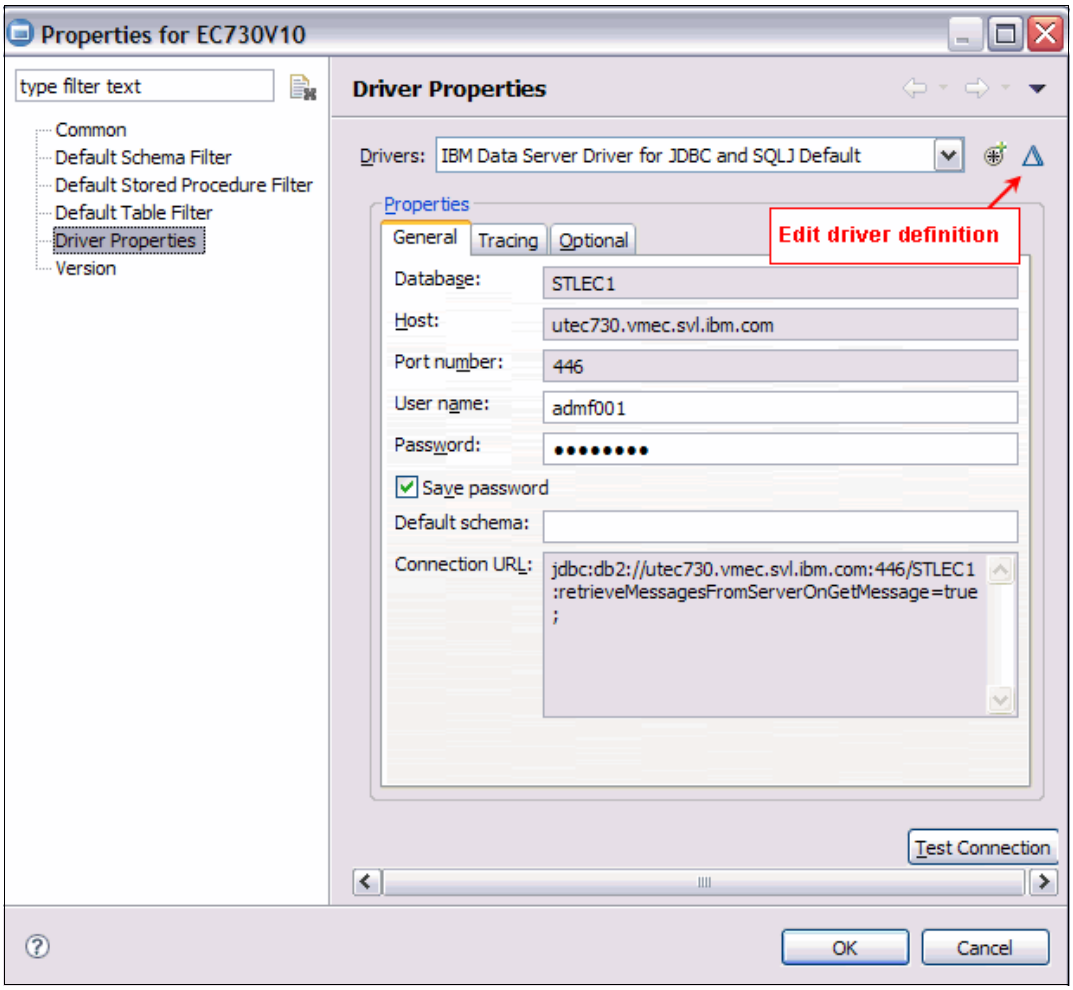

Figure 2-4 Modifying the Driver Properties

When you upgrade Data Studio to a new release, you might need to edit the connection properties on your workspace so that you pick up the latest jars. Figure 2-5 shows a typical error that you might get when connecting.

```
Connection has failed. The following error was reported:
----------------------------------------------------------
java.io.FileNotFoundException:
C:\Program\IBM\SDP70Shared\plugins\com.ibm.datatools.db2_1.0.100.v200707172230\
driver\db2jcc.jar

Do you want to work offline?
----------------------------------------------------------
```

Figure 2-5 Typical connection error

Click the Edit driver definition icon. On the Edit driver definition dialog, click the **Jar List** tab. Click the **Add JAR/zip**, then use the file browser to point to the directory where the JDBC driver jars are. You can also opt to remove or edit the jar files from this dialog.

Filtering the connection

From the connection properties dialog, you can set a default schema filter, a default stored procedure filter, and a default table filter. Filtering the schemas, stored procedure, or table restricts what Data Studio loads into the Data Source Explorer for that connection. You can filter using a where clause-like expression or select from a list. Figure 1-10 on page 30 shows the filter page.

2.1.4 Creating a Data Development Project

Click **File** → **New** → **Data Development Project** to launch the New Data Development Project wizard (Figure 2-6). The wizard contains three pages for setting the following:

- Project name: Enter a meaningful project name. Click **Next**.

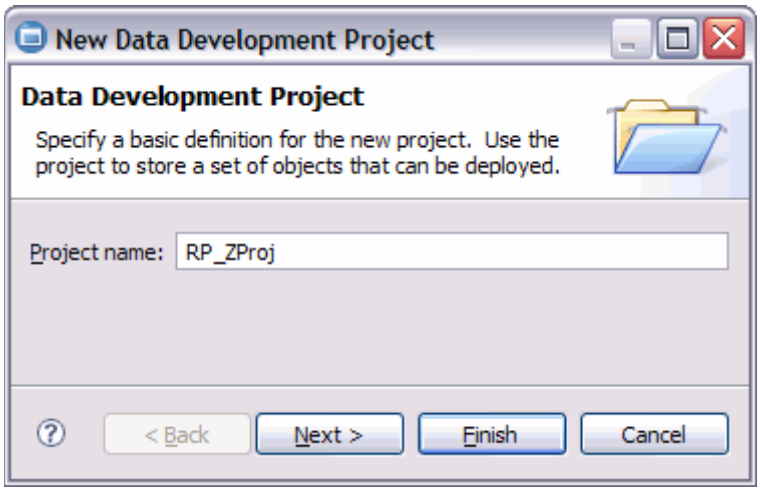

Figure 2-6 New Data Development Project

- Select connection: The wizard's second page lists all the connection profiles available (Figure 2-7). The user can also opt to create a new connection profile by clicking **New**. The New Connection Profile wizard is launched. The properties of the selected connection are shown in the Properties area of this page.

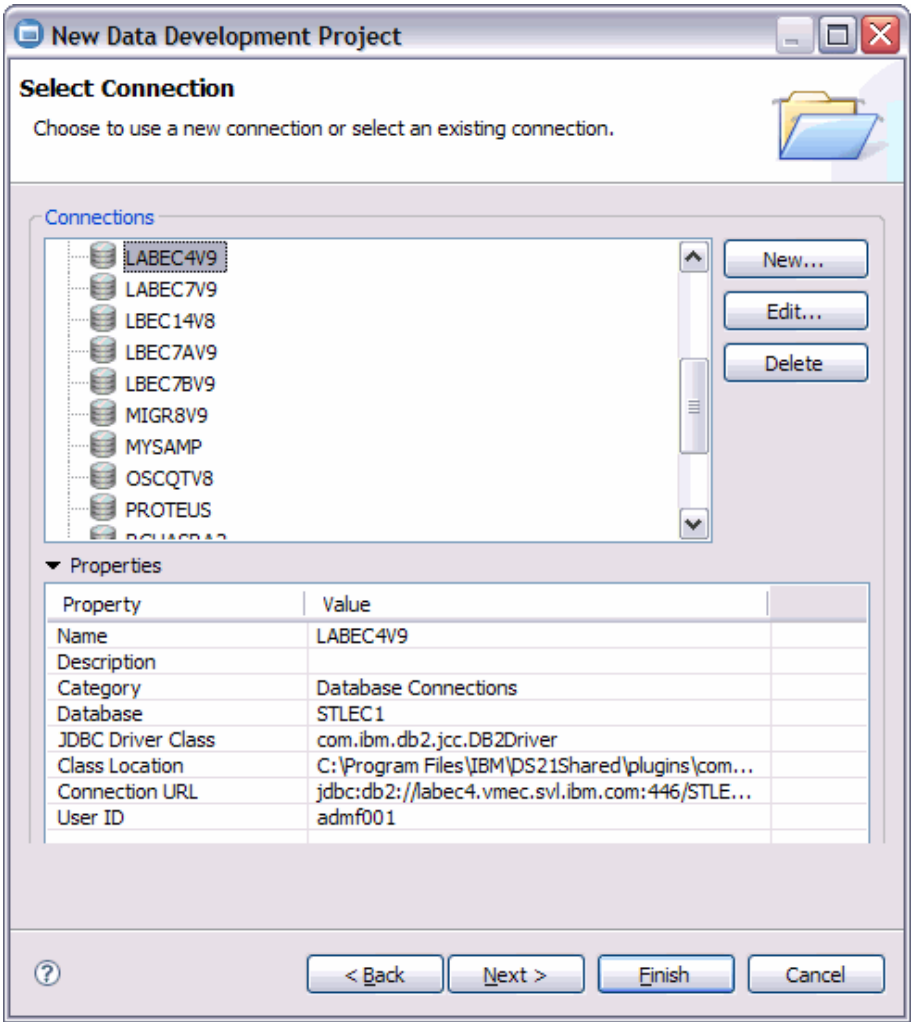

Figure 2-7 New Data Development Project - Select Connection page

The wizard's third page allows the user to specify the authorization ID for package owner and build owner.

- Package Owner: Enter a valid primary or secondary authorization ID as the package owner. If blank, the current login ID is used.
- Build Owner: Enter a valid primary or secondary authorization ID as the owner of the stored procedure created. This is the value in the OWNER column in SYSROUTINES. If blank, the current ID is used.

Click **Finish** in the wizard to create a data development project. Figure 1-15 on page 37 shows you the folders contained in a data development project.

2.1.5 Creating SQL statements and scripts

Before you develop your stored procedures, you can optionally develop your SQL statements first, contain them in a script, test and optionally tune them, and then import them into the stored procedure. An SQL script can contain one or more SQL statements.

Data Studio provides you with three tools for developing SQL statements:

- SQL Query Builder: This is a graphical builder used for creating SELECT, INSERT, UPDATE, DELETE, Full SELECT, and WITH statements.
- SQL and XQuery Editor: This is a rich text editor that can handle both DML and DDL SQL statements, as well as XQuery statements. It has colorization and content assist capabilities.
- SQL Wizard: This is embedded in the New SQL Procedure wizard and is similar to SQL Assist in Development Center.

Using the SQL Query Builder

Right-click the SQL Scripts folder and select **New** → **SQL** or **XQuery** to launch the New SQL Statement wizard. Figure 2-8 shows the dialog that allows you to select whether to create the SQL statement with either the SQL Editor or the SQL Query Builder. Select the latter.

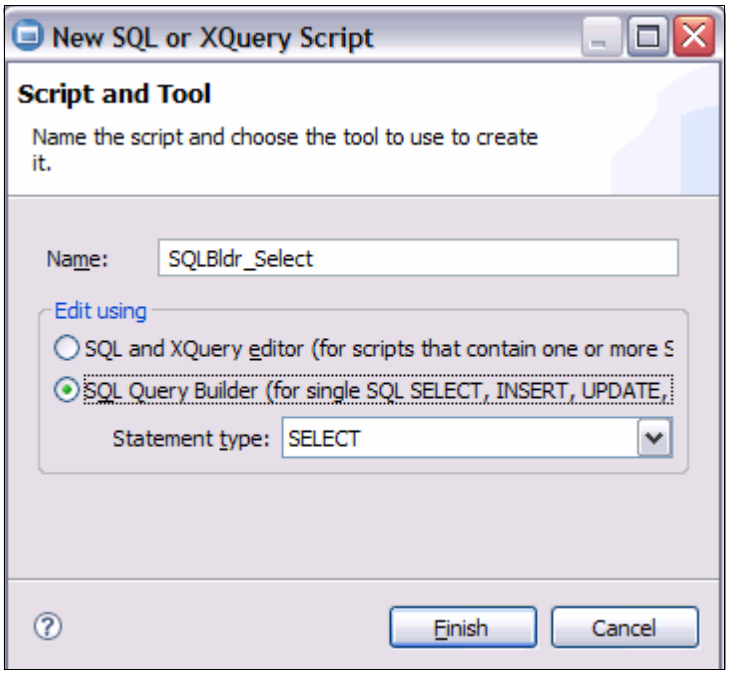

Figure 2-8 New SQL or XQuery Script

The SQL Query Builder populates the Editor view with three panes (the top three icons in Figure 2-9 are the three panes). Choose the the SQL statement view.

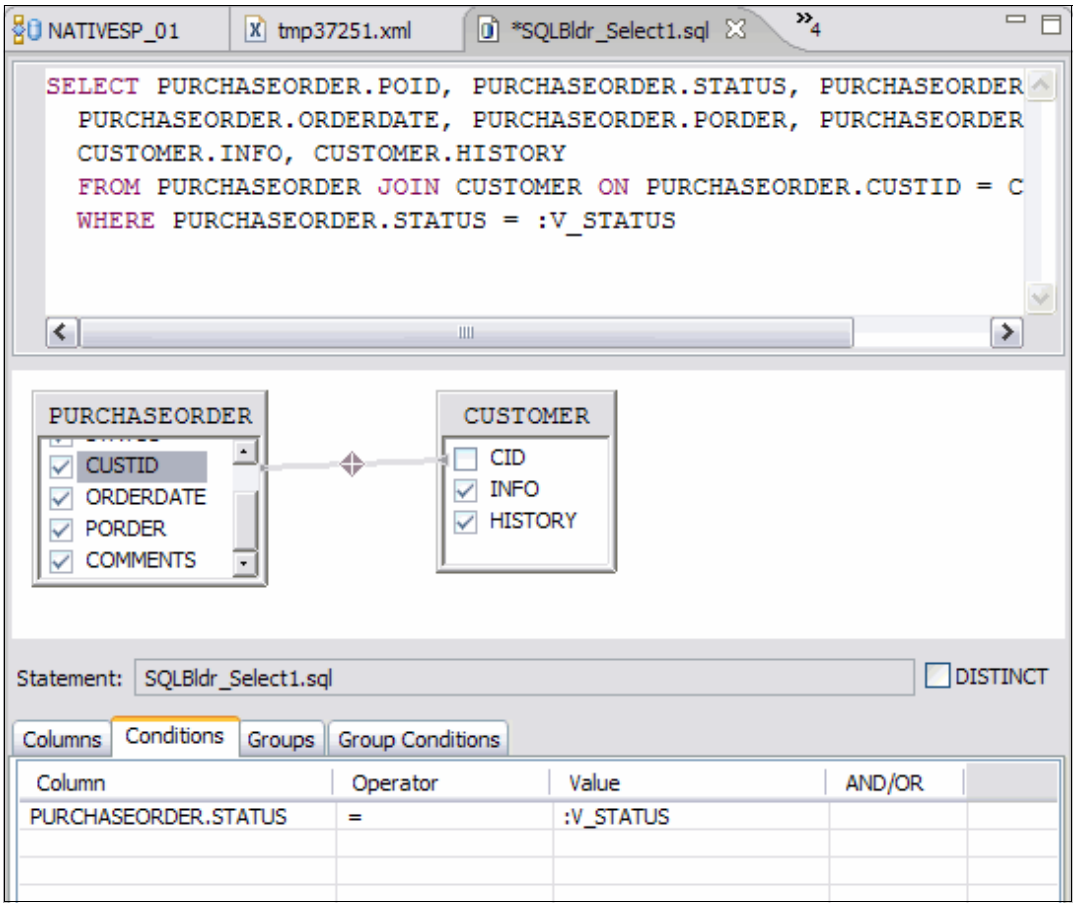

Figure 2-9 SQL Query Builder

To build the illustrated statement in the SQL Query Builder:

1. Right-click the graphical view (middle pane) and select **Add table**. Expand the **DEVL7083** schema and select the **PURCHASEORDER** table. Add the CUSTOMER table in the same way. This adds the selected tables to the FROM clause of the statement.
2. Still in the graphical view, click **CUSTID** in the PURCHASEORDER table and drag ti to CID in the CUSTOMER table. This creates an inner join between the two tables. You can right-click the join line and select **Specify the join type** to select another type of join, such as a left outer join.
3. Click the **POID**, **STATUS**, **CUSTID**, **ORDERDATE**, **PORDER**, and **COMMENTS** check boxes in the PURCHASEORDER table. This adds those columns to the SELECT clause of the statement.
4. Click the **INFO** and **HISTORY** check boxes in the CUSTOMER table.
5. Click the **Conditions** tab. Click the first cell, and a pull-down arrow is displayed. Scroll down the list and select **PURCHASEORDER.STATUS**.
6. Click the **Value** column and type `:V-STATUS`. This creates a variable in the statement. When you run the query, you are prompted to provide a value for this variable.
7. Click **Save**.

Using the SQL and XQuery Editor

In the New SQL or XQuery Statement dialog (Figure 2-8 on page 57), you can opt to use the SQL and XQuery Editor by selecting **SQL and XQuery Editor**. The editor is started with a blank page in which you can enter any SQL statement.

The SQL and XQuery Editor has the following features:

- Multiple statement support

  You can type multiple statements in a `.sql` file and then run them.

- Variable statement terminator

  The statement terminator is actually a statement separator. By default, the SQL editor uses a semicolon (;). You can specify a different statement terminator for the statements that you create in the Enhanced SQL Editor. You do not need to specify a statement terminator for the last (or only) statement in your script.

- Multiple target connection profiles

  You can select the target connection profile where you want the SQL script executed. The combo box contains connections that you have used prior to this execution. Click **Select** to use another connection profile or to create a new connection profile.

- Syntax highlighting

  To aid you in differentiating the elements in an SQL statement, syntax highlighting renders different kinds of elements in the text in unique colors.

- Content assist

  Content assist is an editing tool that provides you with helpful information as you type an SQL or XQuery statement. For example, after you type the dot that follows a schema qualifier in an SQL statement, content assist supplies a list of the tables in the schema.

  As you develop your statement, you can press Ctrl+Spacebar at any time to see a list of available choices in the content assist window. You can filter the list of choices by typing a character, and only the choices beginning with that character are shown.

- Query parsing and validation

  As you type, the parser checks the syntax of both SQL and XQuery expressions, and provides a visual indication of any errors that it detects in the query.

The developerWorks article “Creating scripts more efficiently in the SQL and XQuery editor” provides examples and more details about the SQL and XQuery editor. It is available at:

`http://www.ibm.com/developerworks/data/library/techarticle/dm-1011sqlguidetour/index.html?cmp=dw&cpb=dwinf&ct=dwnew&cr=dwnen&ccy=zz&csr=111110`

In Data Studio, you can also use or create code templates. This allows you to skip typing parts of an SQL statement and tab into input fields within the templates. Figure 2-10 shows an example of a template.

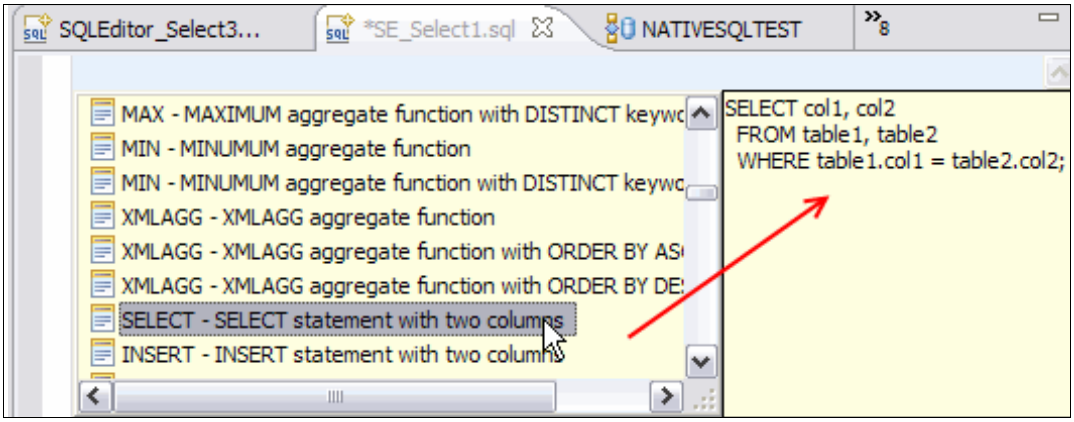

Figure 2-10 SELECT template with two columns

To use this template and create the SQL statement in the Enhanced SQL Editor, we took the following steps:

1. Right-click the SQL Scripts folder and select **New** → **SQL** or **XQuery** script.
2. Type `SE_Select1` for the statement name. Click **Edit using the SQL Editor**.
3. A blank editor page is created. Press Ctrl+Spacebar to activate the Content Assist.
4. Type `S` to see template choices that start with S. Select **SELECT - SELECT statement with two columns template**.
5. In the editor the cursor is placed in col1. Type `PURCHASEORDER.POID`.
6. Tab to col2. Type `CUSTOMER.INFO`.
7. Tab to table1. Type `PURCHASEORDER`.
8. Tab to table2. Type `CUSTOMER`.
9. Press Esc to return to the normal editing mode.

You can create your own SQL statement template in the Preferences of Data Studio. In the menu bar, click **Window** → **Preferences** → **Data Management** → **SQL Development** → **SQL and XQuery Editor** → **Templates**.

Whether you use the SQL Query Builder or the SQL and XQuery Editor, your created scripts are saved in the SQL Scripts folder.

2.2 Creating a new stored procedure

For stored procedures, Data Studio supports:

- Creating a stored procedure from templates
- Copying and pasting a stored procedure from the Data Source Explorer
- Importing a previously written stored procedure from another project or file system

2.2.1 Creating a new stored procedure using templates

Right-click the **Stored Procedures** folder and click **New**. The New Stored Procedure wizard is launched. This is a one-page wizard. Figure 2-11 shows the wizard's page.

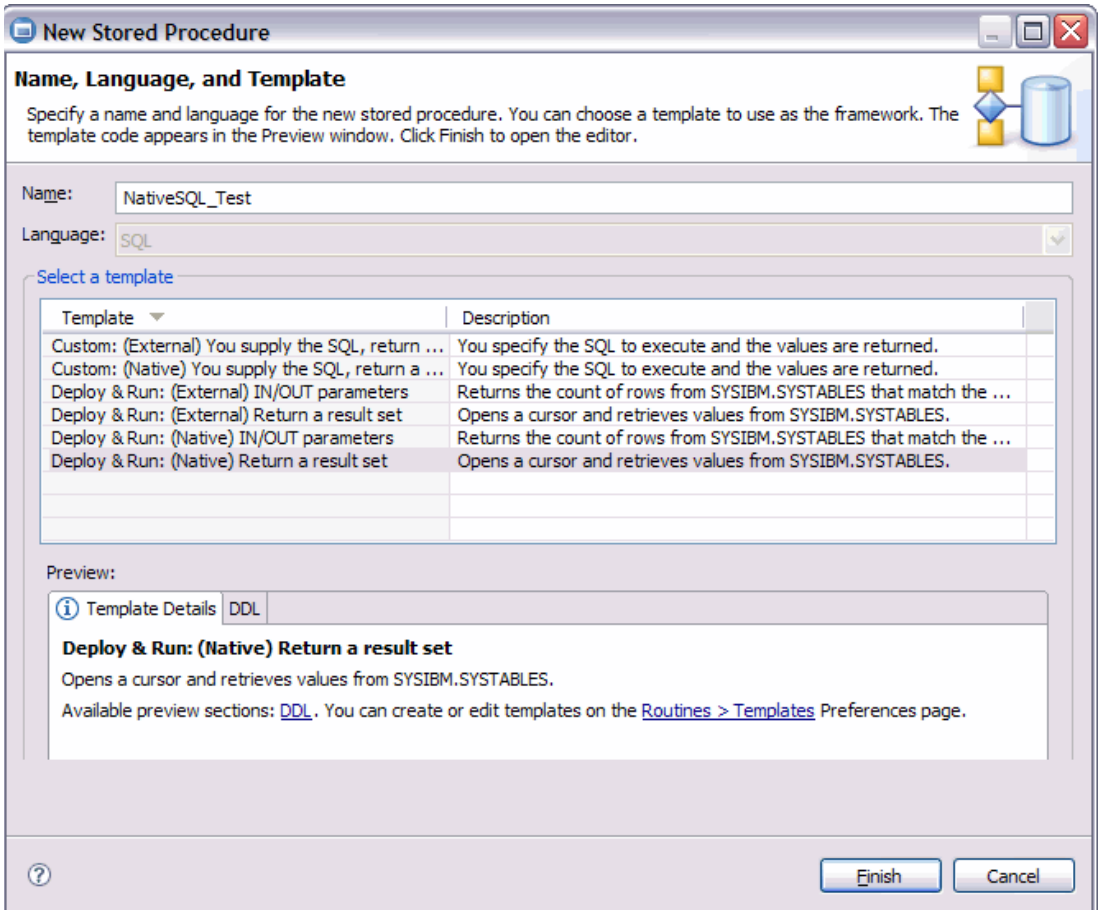

Figure 2-11 New stored procedure wizard

Name

Specify the name of the stored procedure. You might also qualify the stored procedure name with a schema qualifier (for example, MYSCHEMA.MYPROC).

Note: Data Studio accepts uppercase and lowercase schema.procname. However, when the SQL or Java stored procedure is built, both schema and procname are converted to uppercase in the DB2 catalog on z/OS. To enter lowercase, enclose the name in quotation marks, for example, “MyProc”.

Language

Select from the pull-down menu the language for this stored procedure. Data Studio only supports SQL or Java stored procedures. When you select SQL, you are presented with the default templates for both Native and External SQL stored procedures. If you select Java, the templates list is changed to display templates using dynamic Java (JDBC) or static Java (SQLJ).

Note: In DB2 for z/OS, the default version for native SQL procedures is V1.

The template area below the language area contains two tabs describing the template that you have selected.

Default templates

Data Studio provides three basic default templates:

- You specify the SQL statement that returns a result set.
- The stored procedure has two parameters:
  - Input
  - Output
- The stored procedure returns a result set.

Template details icon

A description of the content of this template is shown here. If you create your own templates, the text that you entered to describe the template is what is shown here.

Template DDL

This is a preview of the CREATE PROCEDURE DDL. If you are creating a SQL stored procedure, the SQL Body is also shown in this tab.

When you click **Finish** from the New Stored Procedure wizard, Data Studio generates the DDL and, in the case of Java stored procedures, the Java source. The DDL is rendered in the Routine Editor, whereas the Java source is rendered in a Java Editor.

When generating the DDL, Data Studio uses values from **Preferences** → **Routines** → **Deploy Options**.

- For Java stored procedures, it picks up the bind options and WLM Environment preferences.
- For External SQL stored procedures, it picks up precompile options, compile options, prelink options, link options, bind options, runtime options, WLM environment, and collection ID.
- For Native SQL stored procedures, it picks up procedure options and (optionally) the WLM environment, which can be used when debug is enabled. Consult the SQL Reference for details on the procedure options that you can specify in your native SQL stored procedure.

Note: Although native SQL stored procedures do not require you to specify a WLM environment, you still need to specify a default WLM environment in your DSNZPARM.

2.2.2 Copying and pasting (or dragging and dropping) from the Data Source Explorer

In Data Studio, you can copy and then paste or drag and drop a stored procedure that has been previously deployed from the Data Source Explorer to a Data Project Explorer. You can copy an SQL or Java stored procedure from one server, paste it to a project, modify as needed, and then deploy to another server. This can be between like platforms and servers or different platforms and servers. Some syntax errors might occur if the stored procedure contains non-standard (that is, platform-specific) SQL.

Copying from the Data Source Explorer to a project

To do this:

1. Ensure that the project exists.
2. Select the stored procedure from the Data Source Explorer.
3. Drag and drop (or right-click **Copy** and then click **Paste**) to the `Stored Procedures` folder of the project.

2.2.3 Importing the source of a stored procedure from a file

You can use the Import wizard to copy a SQL stored procedure from the file system or another project to the current project. Section 2.4, “Importing a stored procedure” on page 65, provides more details about the import wizard.

2.3 Modifying the stored procedure

The Routine Editor gives the user total control in modifying the DDL, the SQL body of a SQL stored procedure or the Java source of a Java stored procedure. From the Routine Editor, right-clicking whitespace brings up a menu that allows you to:

- Copy and paste SQL statements and text from another editor.
- Insert from file SQL statements and text contained in files.
- Format SQL statements that you typed into the editor so that it is more readable.
- Validate Syntax of SQL statements that you typed. Data Studio can parse the SQL statement for correctness or validate that the tables exist in the target server.
- Invoke Content Assist while typing the SQL statement. See “Using the SQL and XQuery Editor” on page 59 for details about how to invoke Content Assist.
- Set Terminator (SQL) allows you to set different characters for the statement terminator other than the semicolon.
- Select SQL → Compute Actual Cost, which calls the DB2-supplied stored procedure DSNWPSM and returns a cost value for executing the statement.

  Note: You need to set up this support in DB2 for z/OS beforehand. See 1.4.7, “Data Studio actual costs setup” on page 19.

- Open Visual Explain against a selected SQL statement. A graphical display of the access path is shown in the Access Plan Diagram view of the Outline View.

2.3.1 Copying and pasting, and inserting from file

Data Studio provides menu actions for inserting SQL statements or text into the stored procedure. When you paste in SQL statements, the parser can detect some syntax errors in the SQL statements added. You can copy and paste SQL statements from the `SQL Scripts` folder or any other open editor.

You can also insert files containing SQL statements or text from the file system. In previous versions of Data Studio, you were limited to inserting code fragments in certain areas of the stored procedure. For completeness, we define the popular areas where code fragments are usually inserted.

For SQL and Java stored procedures, examples of code fragments are:

- Header fragment: This can contain text such as prologs, copyright information, author, modification history, and so on.
- Imports fragment (Java): This can contain a list of common imported Java packages.
- Variable / Data declaration fragment: This can contain code for declaring or initializing global variables, local variables that might follow a specific naming convention, and so on.
- Exception handlers fragment (SQL): This can contain code for handling exceptions encountered when executing the stored procedure.
- Pre-return fragment: This can contain code for cleanup, “finalize”-ing, logging, reporting, and other end-of-task code that might be common to all your stored procedures.
- Method fragment: This can contain code for common methods called within a Java stored procedure. You can reference methods in your source that are not coded in the Java stored procedure. Section 3.1.9, “Multiple jar support for Java stored procedures” on page 99, describes how to add a supporting `jar` that contains these methods.

Note: The file browser defaults to presenting you with files that have a `.sql`, `.ddl`, or `.db2` file extension. You can only select one file to insert. You have to position your cursor to the location of where you want the file inserted. When inserting a file containing SQL Statements into the Java source, you are responsible for coding the JDBC APIs and Java-specific code needed to process the SQL statements.

2.3.2 Editing the Java source

The Java Editor also allows the user to modify the Java source, and includes features such as:

- Syntax highlighting and checking
- Content/code assist
- Code formatting
- Import assistance
- Quick fix

A word about parameters

The new stored procedure wizard gives the user the flexibility to add, modify, and remove parameters, including handling of:

- SQL Exception
- SQL Message
- SQLSTATE

However, if you add, remove, or reorder parameters in a Java stored procedure DDL, you need to update the Java source to map the SQL data types of the parameters with the Java data types of the arguments in the Java source’ main method. Table 31, “Mappings of database server data types to Java data types for retrieving data from database server,” in the *DB2 10 for z/OS, Application and Programming Guide for Java* maps the SQL data types to Java data types.

Data Studio now supports XML parameter data types when developing a stored procedure against a DB2 9 or DB2 10 for z/OS server. The SQL data type, XML, maps to `java.sql.SQLXML`.

2.4 Importing a stored procedure

You can import an entire stored procedure into a project using Data Studio's import wizard. You can import from the file system or another project into the current project's stored procedure folder.

2.4.1 Importing an SQL stored procedure

To import a SQL stored procedure from the file system, right-click the **Stored Procedures** folder and then select **Import** to launch the Import wizard. The Import wizard includes the four pages (discussed in this section) for importing a stored procedure.

Source

On the Source page, you have the option to import from an existing project or the file system (Figure 2-12).

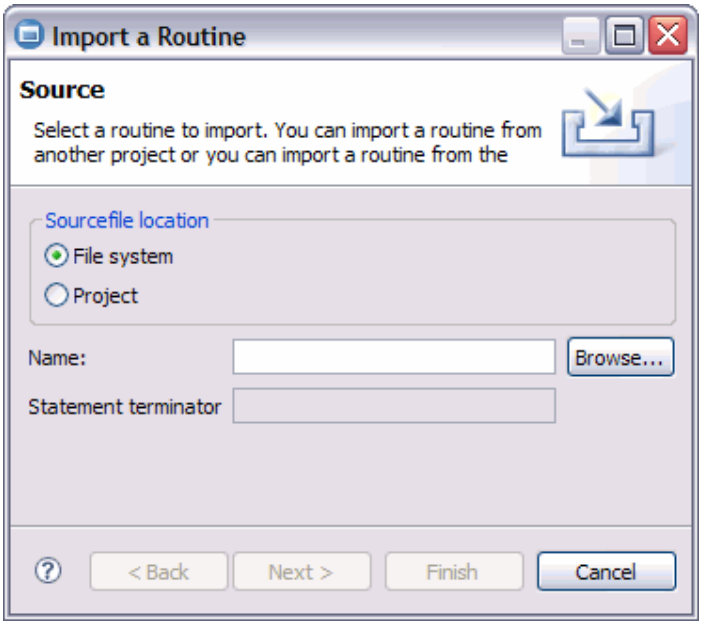

Figure 2-12 Import Wizard, Source page

To import from the file system:

1. Click **Browse** to the right of the Name field. A file browser is launched.
2. Use the file browser to locate the file that contains the stored procedure code.
3. Click **Open** to import this file. The file directory appears in the Name textbox.
4. Specify the statement terminator used in the file. Because SQL stored procedures can include SQL statements in the SQL body that are terminated by a semicolon, the termination of the CREATE PROCEDURE statement itself is signified by a character other than a semi-colon.

To import from a project:

1. Click the **Project** radio button.
2. Click **Browse** to the right of the Name field. You are presented with a tree list of projects and stored procedures (Figure 2-13).

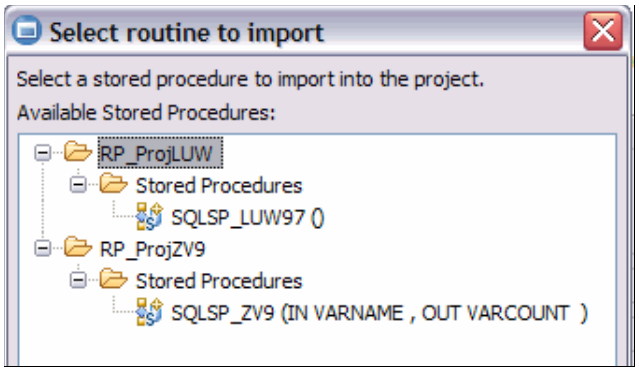

Figure 2-13 Import from a project

3. Select the stored procedure that you want to import. Click **OK** to dismiss this dialog.
4. Click **Next** to go to the next page in the Import wizard.

Entry Points dialog

Data Studio opens and parses the imported file and presents you with possible entry points in the file. If the file contains a SQL stored procedure, the CREATE PROCEDURE name and parameter signature is the entry point (Figure 2-14).

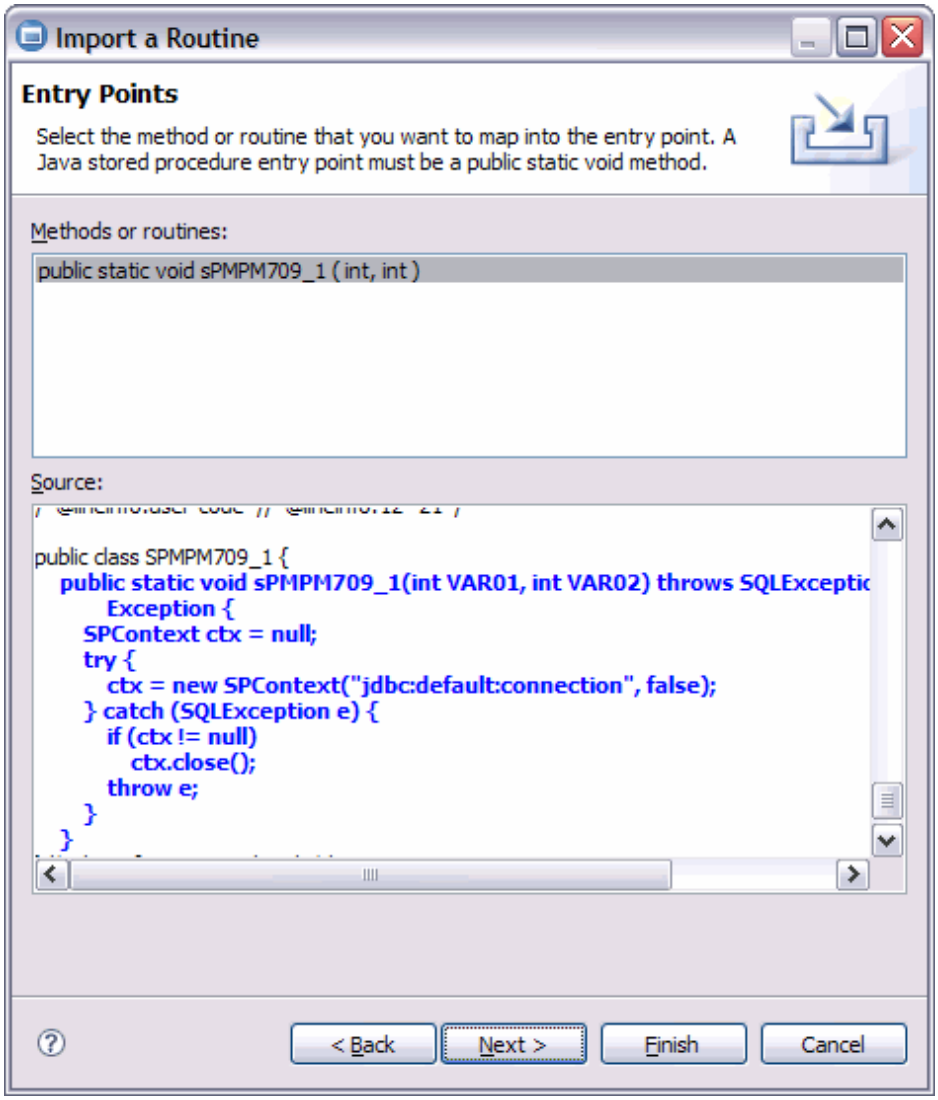

Figure 2-14 Import wizard - SQL stored procedure entry point

Click **Next** to go to the next page in the Import Wizard.

Parameters

Data Studio displays the parameters and their data types on this page. You can edit the SQL data types of the parameters in a Java stored procedure (Figure 2-15).

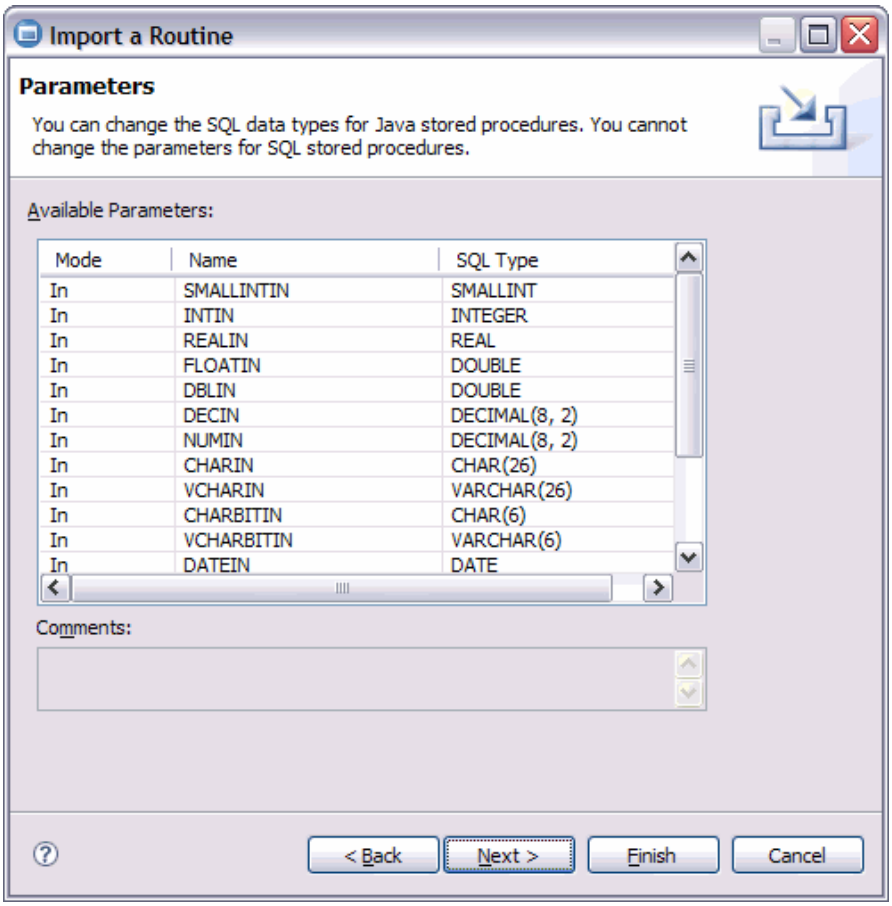

Figure 2-15 Import wizard - Parameters page

Click **Next** to go to the next page.

Options

On this page, you can set the following options needed to deploy the imported stored procedure (Figure 2-16):

- Collection ID
- Replace duplicate routines in the project
- Deploy on Finish (Automatically deploy the imported stored procedure on finish.)
- Enable debugging (when you deploy the imported stored procedure)

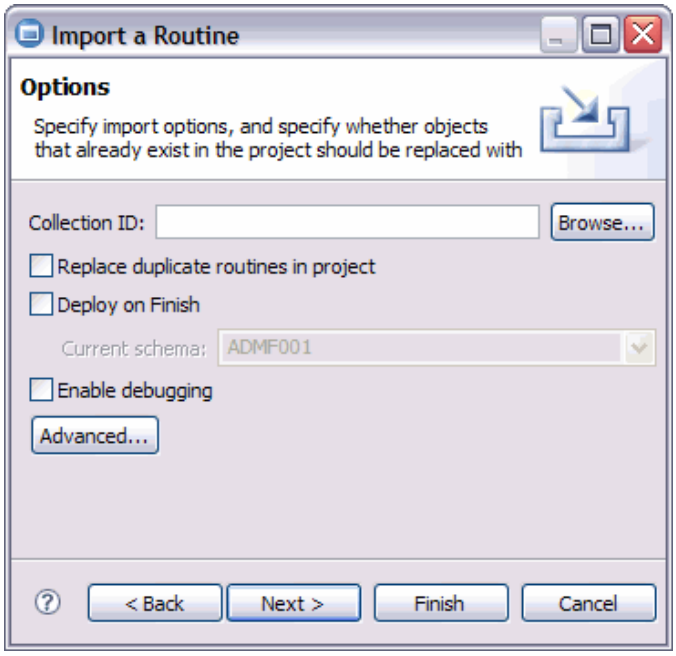

Figure 2-16 Import wizard - Options

Click **Advanced** to specify additional stored procedure and deploy options, similar to those shown in Figure 2-20 on page 76.

Click **Next** to display the Summary page.

Summary

The summary page summarizes the above settings. Click **Finish** to terminate the Import wizard. The imported stored procedure is added to the `Stored Procedures` folder.

Note: When importing an SQL stored procedure from the file system to a project that is targeting a DB2 10 for z/OS, if the imported DDL does not contain the FENCED or EXTERNAL keyword, Data Studio constructs the imported SQL stored procedure as a Native SQL stored procedure.

2.4.2 Importing a Java stored procedure

As mentioned in 2.2.3, “Importing the source of a stored procedure from a file” on page 63, you can import a SQL stored procedure source from the file system using the Import wizard. The steps to import a Java stored procedure are similar to the steps for a SQL stored procedure.

Right-click the **Stored Procedures** folder and then select **Import** to launch the Import wizard. On the Source page, select the location of a `.java` file in the file system or a Java stored procedure from another project.

If the imported stored procedure is a Java stored procedure, Data Studio examines the methods that are included in the file. Data Studio selects as the main entry point the method that is named the same as the file as shown in Figure 2-17.

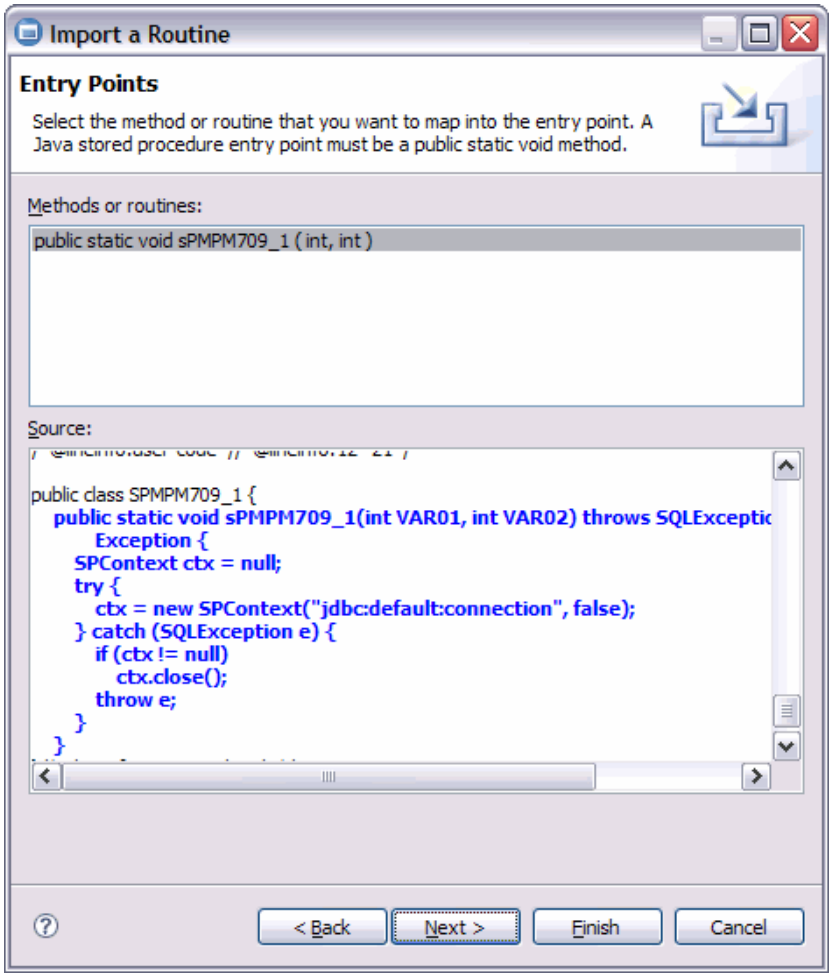

Figure 2-17 Import wizard - Entry points for a Java stored procedure

The Options page when importing a Java stored procedure is slightly different. On this page, you can:

- Enter the JAR ID.
- Enter a collection ID.
- Opt to replace duplicate routines in the project.
- Deploy on finish importing.
- Enable debugging when the imported stored procedure is deployed.

Similar to SQL stored procedures, you can also click **Advanced** to specify additional deploy and routine options, such as the WLM environment name.

2.5 Deploying a stored procedure

To execute or CALL a stored procedure, the stored procedure needs to first be built or deployed to the server. The deploy process differs slightly between external SQL, native SQL, and Java stored procedures. Data Studio assists you in the deploy process through the Deploy wizard.

2.5.1 The Deploy wizard

We can deploy the stored procedure under development to the current server from several spots in the workspace. You can launch the Deploy wizard from the context menu of the following:

- **Data Source Explorer** → **Stored Procedures** folder → **Deploy**
- **Data Source Explorer** → ***a specific stored procedure*** → **Deploy**
- **Data Project Explorer** → **Stored Procedures** folder → **Deploy**
- **Data Project Explorer** → ***a specific stored procedure*** → **Deploy**
- **Routine Editor** → right-click **DDL** tab → **Deploy**
- **Routine Editor** → Deploy icon (Figure 2-18)

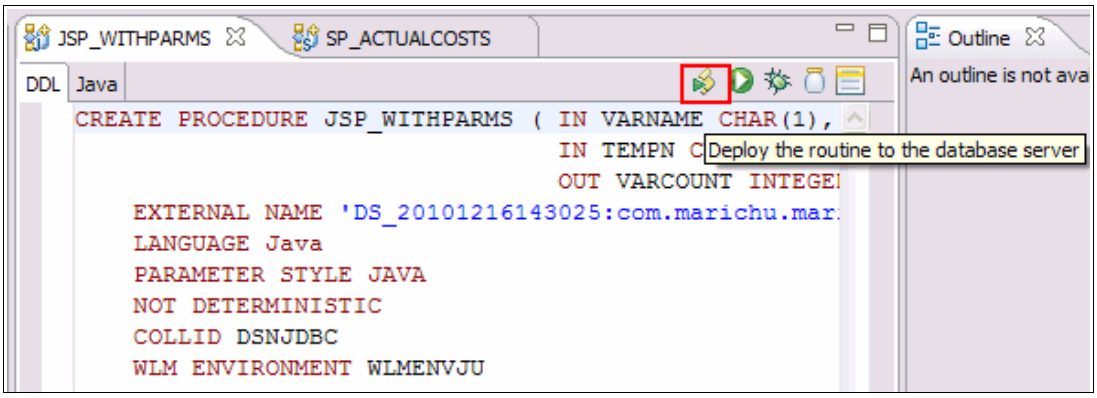

Figure 2-18 Routine Editor - Deploy to server

Let us assume that you clicked the “Deploy the routine to the database server” icon. The Deploy wizard is launched (Figure 2-19).

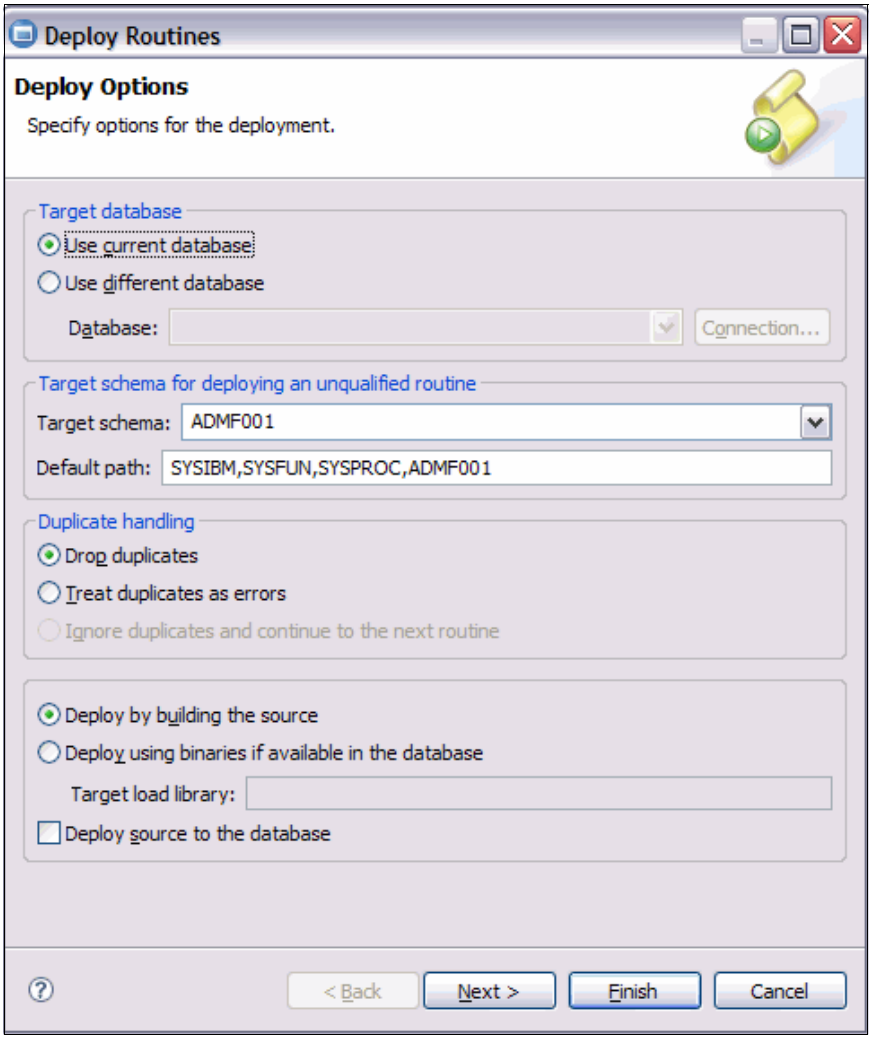

Figure 2-19 Deploy wizard - Deploy Options

2.5.2 Deploy options

The first page of the Deploy wizard presents you with options for setting the target database, handling errors and duplicates, and specifying what is included when deploying the stored procedure.

Setting the target database

Select the **Use current database** radio button to deploy the stored procedure to the database server configured for the project that contains this stored procedure.

Select the **Use different database** radio button if you want to deploy to another database. If you select this radio button, the Database combo box is populated with the list of current active connections. If there are no other active connections, you can click **Connection** to launch the Select Connection dialog. Click **New** to create a new connection profile. Click **Finish** from each dialog to return to the Deploy wizard.

If the stored procedure is unqualified, you can specify the target schema for this routine. The pull-down menu shows you all schema names used so far within the project. The default schema is the login ID.

Error and duplicate handling options

These options specify how you want Data Studio to handle duplicate names or errors when deploying the stored procedure.

- Duplicate handling
  - Drop duplicates.

    If a duplicate is found in the server, issue a DROP against this stored procedure, and then proceed to deploy the new stored procedure. This is the default action for handling duplicates.
  - Treat duplicates as errors.

    If a duplicate is encountered, handle this as an error and follow the action selected under Error Handling.
  - Ignore duplicates and continue to the next routine[2].

    This option is grayed out when deploying a single stored procedure.This option prevents dropping the duplicate stored procedure in the server, and rolls back the deploy of the current stored procedure.
- Error handling
  - Stop and roll back on errors.

    If an error occurs during the deploy, abort the deploy and roll back any previous actions.
  - Stop on errors.

    If an error occurs, stop the current deploy. All previous deploys are not rolled back.
  - Ignore errors and continue to the next routine[3].

    This option only applies when deploying multiple stored procedures. This action skips deploying the current stored procedure and continues to deploy the next stored procedure in the list.

Other deploy options

Other deploy options are:

- Deploy by building source.

  For a first time deploy, this option is selected by default. On subsequent deploys, the user can opt to deploy using binaries.
- Deploy using binaries if available in the database server.

  This option is grayed out for a first time deploy. If an external SQL stored procedure or a Java stored procedure has been deployed successfully, you can opt to redeploy using binaries. See 3.1.6, “Deploying SQL or Java stored procedures without recompiling” on page 94, for details about this option.

[2] When launched from the `Stored Procedures` folder, the Deploy wizard allows you to select multiple stored procedures from the folder. When a stored procedure of the same name exists on the server, this option instructs Data Studio to ignore deployment of this stored procedure. Data Studio continues to process and deploy the other stored procedures in the list. The default is to terminate when a duplicate is found.

[3] When deploying a stored procedure from a list, if the stored procedure has errors (for example, missing objects), then this option instructs Data Studio to ignore these errors and continue deploying the other stored procedures in the list.

- Deploy source to server.

  This option always includes the current Java source when the `jar` file is created for a Java stored procedure. For an external SQL stored procedure, the complete DDL and SQL body is copied into `SYSROUTINES_SRC`.

Click **Next** to display the Routine Options page.

2.5.3 Routine options

The Routine Options page contains 1, 2, or 3 tabs, depending on the stored procedure:

- For native SQL stored procedures, the Routine options is a single page.
- For external SQL and Java stored procedures, you see the Routine options tab and the Deploy Options tab.
- For Java stored procedures, you see a third tab, the Java Path tab.

Routine Options tab

The fields on the Routine Options page depend on the type of stored procedure being deployed.

For External SQL and Java stored procedures, you can set the following options:

- Collection ID
- Runtime options (External SQL stored procedures only)
- WLM Environment
- Procedure options:
  - ASUTIME
  - STAY RESIDENT
  - EXTERNAL SECURITY

For native SQL stored procedures, you can set the options shown in Figure 2-24 on page 80. When the Enable Debugging check box is selected, you are asked to enter a WLM environment that is used when debugging this stored procedure.

Native SQL stored procedures are interpreted rather than compiled like an external SQL stored procedure. Specify both procedure and bind options in the Procedure Options field. See 2.5.7, “Setting the bind options in native SQL stored procedures” on page 80, for information about setting these options.

Note: The Build owner field is the authorization ID that is used to issue the CREATE PROCEDURE statement and hence is the value in the OWNER column of the stored procedure in SYSROUTINES. If there are unqualified objects in the SQL body, these objects are qualified by the value of the CURRENT SQLID, unless the QUALIFIER bind option is specified in the Procedure options.

Deploy Options tab

The fields in this tab also depend on the type of stored procedure being deployed.

For an External SQL stored procedure, you can set the following:

- Build Utility: Specify the SQL Procedure processor to use to build this external SQL stored procedure. The default is the build utility value specified in the preferences.
- Build Owner: Specify the authorization ID that is the owner/definer of this external SQL stored procedure. See 3.1.4, “Package owner and build owner” on page 88, for more information about this field.
- Precompile options: This defaults to `MAR(1,80)`.
- Compile options: This defaults to `NOTEST(block,noline,nopath)`.
- Prelink options: Specify any prelink options (default is blank).
- Link options: Specify any prelink options (default is blank).
- Bind options: The bind options are specified in two parts. The `PACKAGE` area is read-only and defaults to `PACKAGE(collid)`, where collid is the collection ID specified in the Deploy Options preference page.
- Enable debugging: Check this box if you want to use the Unified Debugger later on for debugging this stored procedure.

For a Java stored procedure, you can:

- Click the **Enable debugging** check box to set up the deployed stored procedure for debugging. The compile option -g is automatically generated.
- Specify a different JDK level for the client by clicking **Browse** for the JDK home and pointing the file browser to the directory of your JDK. Data Studio interrogates the JDK that you specified and fills in the JDK version field.
- Add additional bind options in the space to the right of the PACKAGE field. Click the ellipsis to display a text box for typing in your bind options.
- Display all messages generated during the deploy process by clicking **Verbose build**.

Figure 2-20 shows the Deploy Routines dialog.

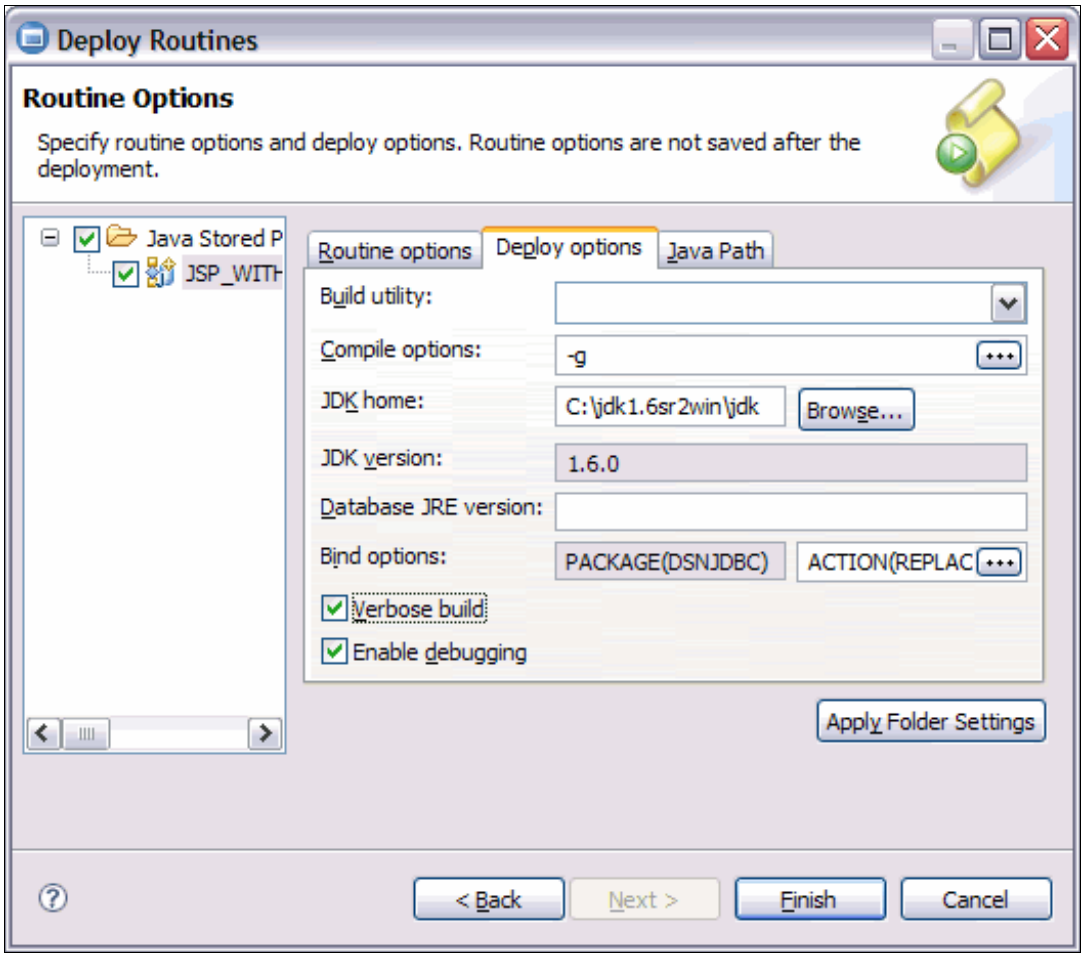

Figure 2-20 Deploy Wizard - Routine Options - Deploy options tab

Java Path tab

On the Java Path tab, you can add jars from other projects to resolve references in your Java stored procedure. Click **Add** and specify the jar name (Figure 2-21).

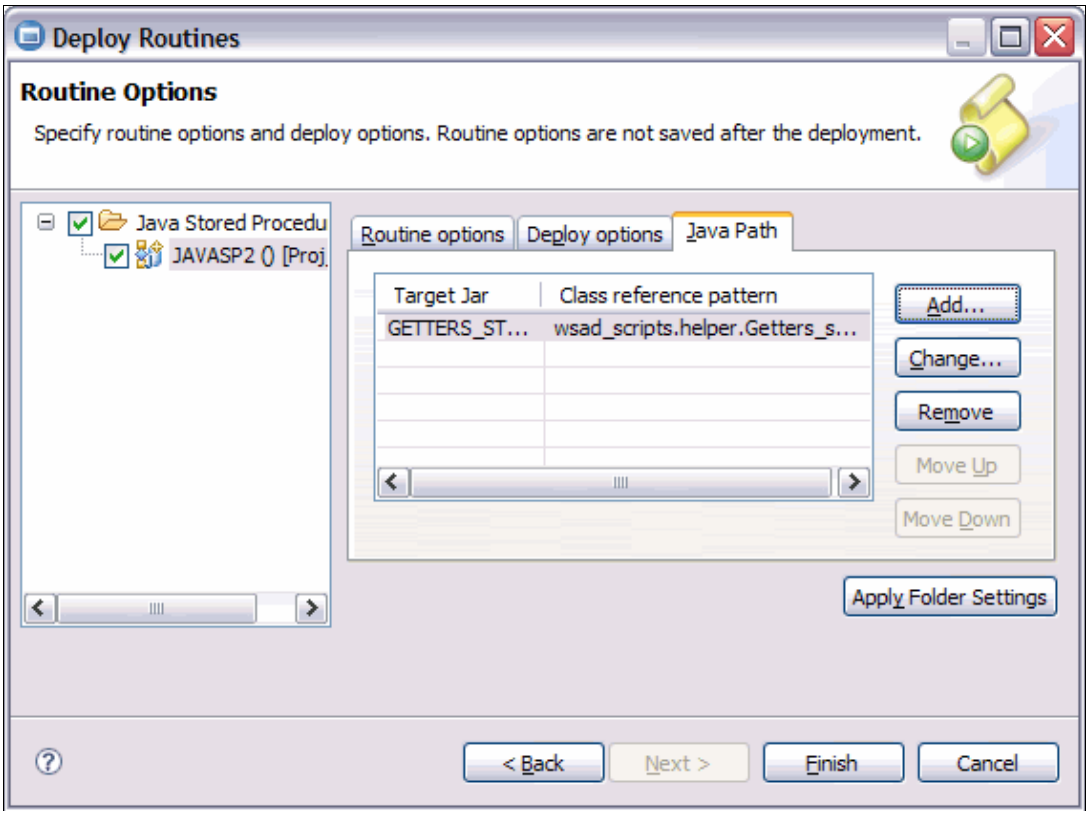

Figure 2-21 Deploy Wizard - Routine Options - Java Path

Click **Finish** to close the Deploy wizard. The Data Output view is refreshed with the status and progress of the deploy.

2.5.4 Deploying to a different server

Data Studio supports deploying to an “unlike” DB2 server (that is, to a DB2 server that is running in another type of operating system). Table 2-1 lists the source and target server combinations supported.

Table 2-1 Deploy source and target server combinations

Source servers	Target servers		
	DB2 for LUW V8.2 and V9	**DB2 for z/OS V8**	**DB2 for z/OS 9 and 10**
DB2 for LUW V8.2, V9	SQL and Java	Java	Native SQL and Java
Derby	Java	N/A	N/A
DB2 for iSeries V5.4	N/A	N/A	N/A
DB2 for z/OS V8	N/A	External SQL and Java	External SQL and Java

Source servers	Target servers		
	DB2 for LUW V8.2 and V9	DB2 for z/OS V8	DB2 for z/OS 9 and 10
DB2 for z/OS 9 and 10	N/A	N/A	External SQL, Native SQL and Java

2.5.5 Deploying nested or dependent stored procedures

You can call stored procedures from within your stored procedures. Deploy the inner or nested stored procedures before the calling stored procedures. In Optim Development Studio, you can specify the order of the deployment using a deployment group. Figure 2-22 shows an example of a deployment group's artifact list.

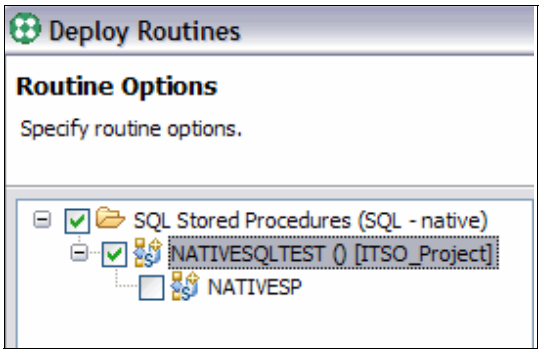

Figure 2-22 Deploying nested stored procedures

2.5.6 Setting the JDK level for Java stored procedures

Data Studio V2.2.1 ships with the Java Development Kit Version 1.6. DB2 10 for z/OS typically ships with JDK 1.5 (although both JDK 1.4 and JDK 1.6 can be installed).

If you compiled your Java stored procedure with a JDK that is higher than what is installed in the DB2 10 for z/OS Server 1.5, then executing this Java stored procedure fails with the following error message:

```
17.38.36 STC00108  DSNX961I  DSNX9WLJ ATTEMPT TO PERFORM OPERATION
     -  FindClass
     - FAILED FOR ROUTINE . .  SSN= V91A     PROC= V91AWMJU ASID= 003A
     - CLASS=  METHOD=  ERROR INFO= java.lang.NoClassDefFoundError:
     - com/ibm/db2/jcc/DB2Driver
```

You can change the JDK level by changing the location of the JDK Home setting in the client. You can set this for all projects, for one project, or for a specific stored procedure.

- Workspace scope: Go to **Window** → **Preferences** → **Data Management** → **SQL Development** → **Stored Procedures & User-Defined Functions** → **Deploy options**.
- Project scope: Right-click the project, then select **Properties** → **Routine Development**.
- Stored procedure scope: When deploying the stored procedures, click **Routine Options** → **Deploy Options** tab.

Figure 2-23 shows how to set the JDK home for a specific stored procedure during deploy. Data Studio issues a UDF to determine the JVM level of the DB2 for z/OS server. The JVM level or version can also be determined from the JAVA_HOME environment variable set in DB2 for z/OS UNIX System Services.

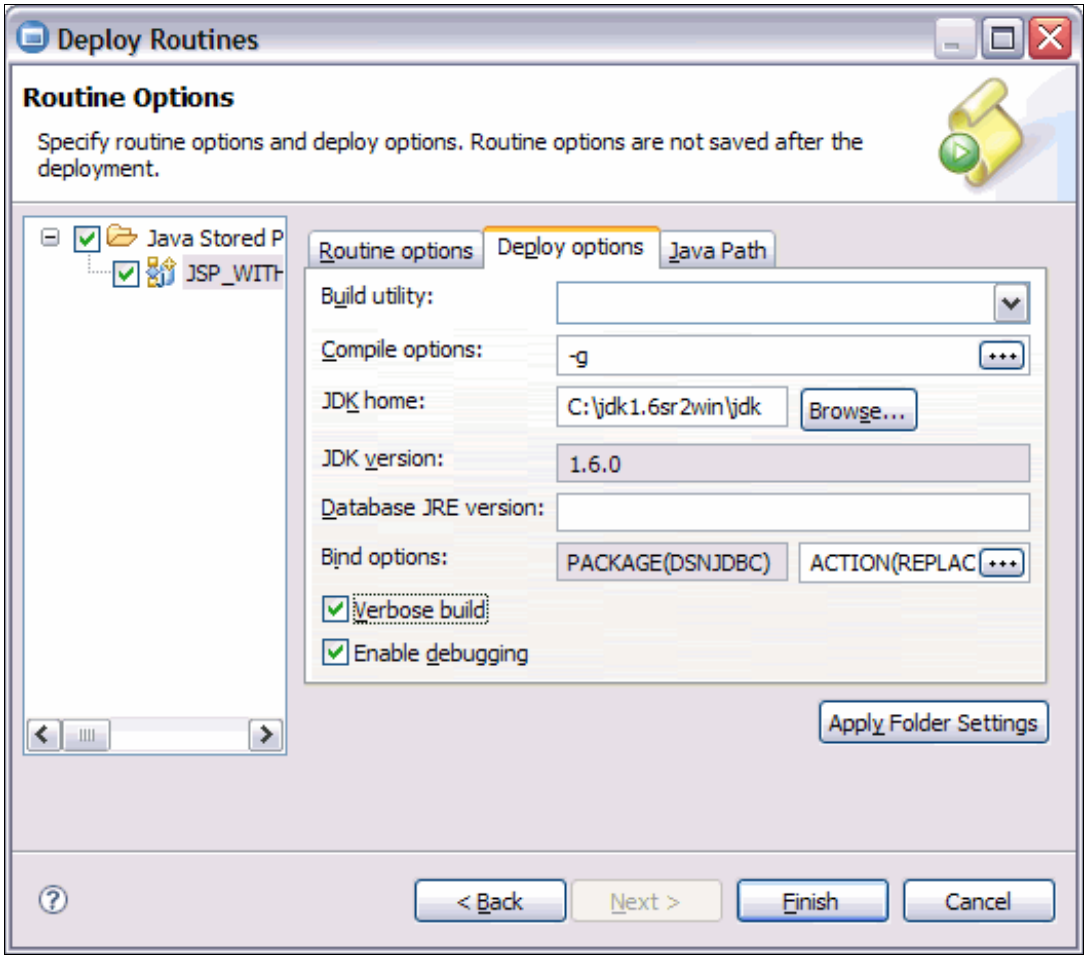

Figure 2-23 Changing the JDK level at deploy time

2.5.7 Setting the bind options in native SQL stored procedures

Native SQL stored procedures process certain bind options as procedure options. The New Stored Procedure wizard allows you to specify non-default bind options in the z/OS Options dialog (Figure 2-24). Click the ellipsis to type the options in a larger text area. These options appear in the CREATE PROCEDURE DDL after the VERSION keyword.

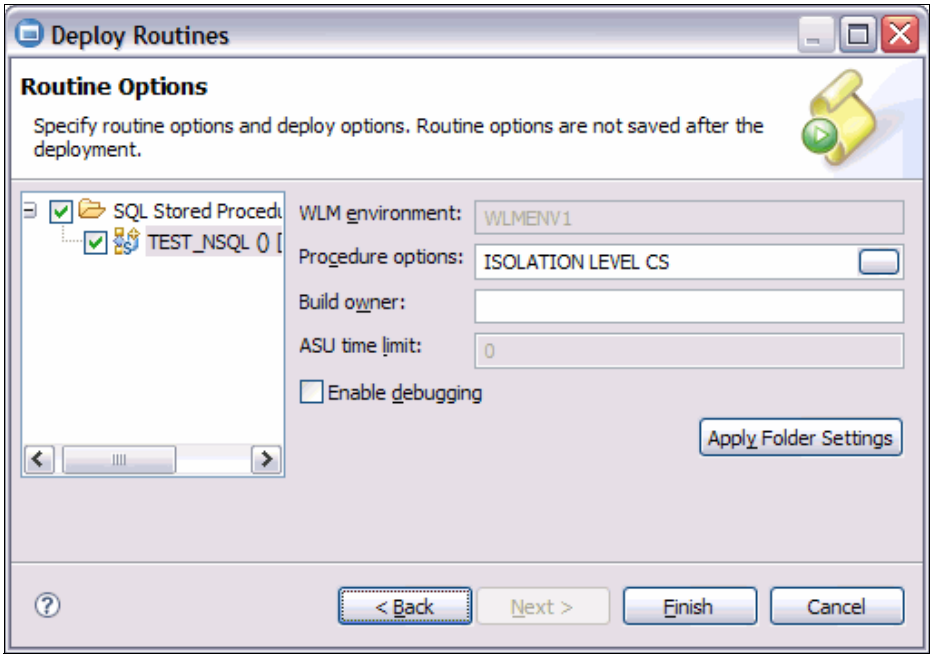

Figure 2-24 Native SQL z/OS options

2.5.8 Enabling debug

Refer to 4.1, “The Unified Debugger” on page 116, for information about how to enable debug using the Unified Debugger.

2.6 Executing a stored procedure

After you have deployed your stored procedure, you can execute it from the following areas:

- **Data Source Explorer** → ***a specific stored procedure*** → **Run**
- **Data Project Explorer** → ***a specific stored procedure*** → **Run**
- **Routine Editor** → right-click white space → **Run**
- **Routine Editor** → click the Run icon (green arrow in Figure 2-18 on page 71)

If there are any input parameters in your stored procedure, the Specify Parameter Values dialog is launched (Figure 2-25).

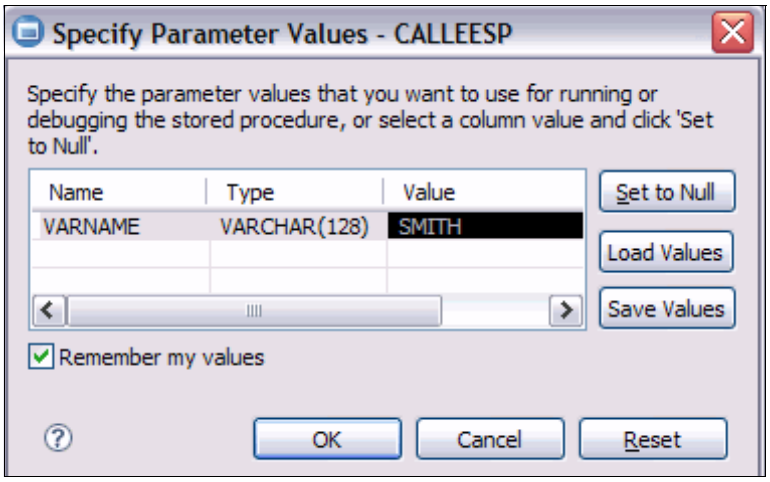

Figure 2-25 Specify parameter values at SP execution

Note that the tooling can recognize that a string was entered even though single quotation marks were not entered. However, double quotation marks are preserved in the input value. To enter FOR BIT DATA, type an `X` in front of a quoted value (for example, `X'F1F2F3'`).

2.6.1 Run Settings dialog

The Run Settings dialog is launched when executing a stored procedure that contains INPUT or INOUT parameters in its declaration. This dialog can also be launched prior to executing a stored procedure from the stored procedure's context menu.

In the Parameter Values tab of this dialog, you can:

- Enter values for each INPUT or INOUT parameter. Data Studio validates the value against the data type of the parameter. An icon appears and a tooltip message is displayed when you click Ctrl+Spacebar.
- Set the value of a parameter to NULL by clicking **Set to Null**.
- Remember the values that you entered from one execution to another.
- Save the values that you entered into a file.
- Load the values from a file that was previously saved.

You can opt to execute SQL statements before and after calling a stored procedure, as when a stored procedure is manipulating a table. You can do this using the Run Settings dialog (Figure 2-26).

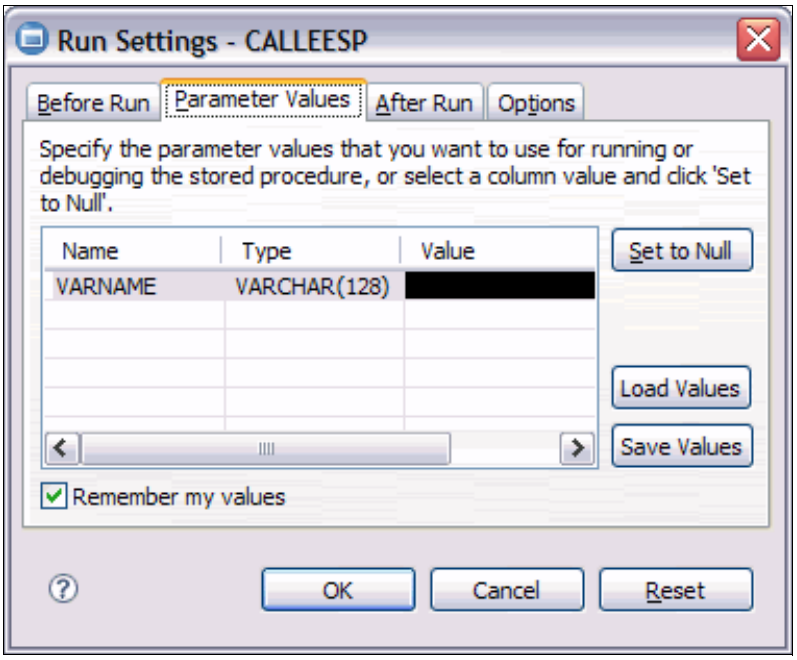

Figure 2-26 Run Settings dialog

On the Parameter Values page, specify the parameters that you want to use for running the routine. If a parameter requires a string value, type the value without string delimiters. If the parameter requires binary input, enter the hex string without delimiters.

2.6.2 Processing information in the Data Output view

In 1.5.5, “Output view” on page 39, we discussed the two areas in the Data Output view. The actions that you can take on these are as follows:

- Status History table
  - Select a cell in the history table. Right-click the cell and then select **Remove / Remove All** entries.
  - Select a cell in the history table. Right-click and select **Save History** to save the status history into a file.
- Status tab

  Right-click the **Status** tab and click **Select All**, then right-click **Copy** to copy the messages to the clipboard.

- Results tab

  If you have multiple result sets, the actions below only apply to the result set page in view. Select a cell or a set of cells (by pressing Ctrl+click). Right-click to see the context menu actions shown in Figure 2-27.

  - Copy Row(s): Saves the selected rows to the clipboard.
  - Save ... → Current Result / All Results: Saves the result sets into a file within a project. In the Save Result set dialog, you can select the project container of the exported result set, the file type, character encoding, and format of the saved result set (Figure 2-28 on page 84).
  - Export ... → Current Result / All Results: Saves the results into a file in the file system. Use the Browse button to launch the file browser. You also have the option to set the file type, character encoding, and format of the exported file.
  - Print ... → Current Result / All Results: Prints the results to a connected printer. Results are printed in a table format.
  - Convert row(s) to hexadecimal: Converts the column data to hexadecimal.

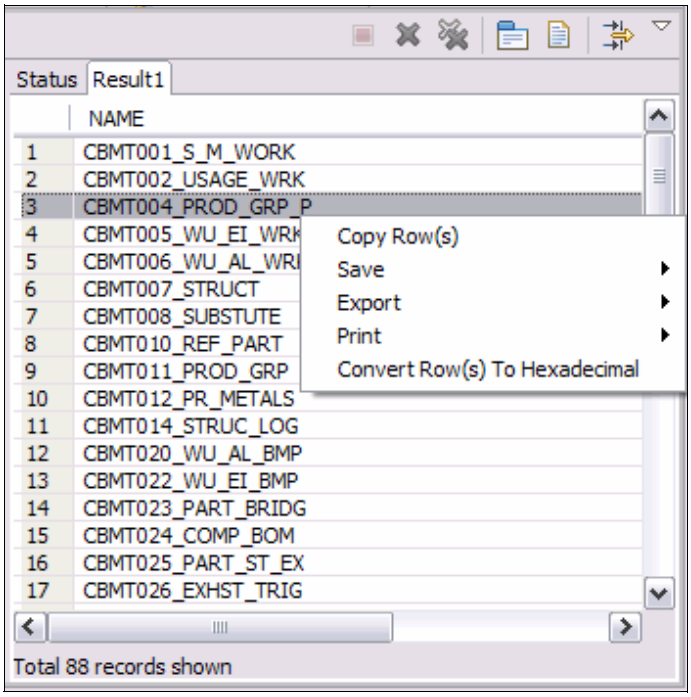

Figure 2-27 Output view - Results tab context menu

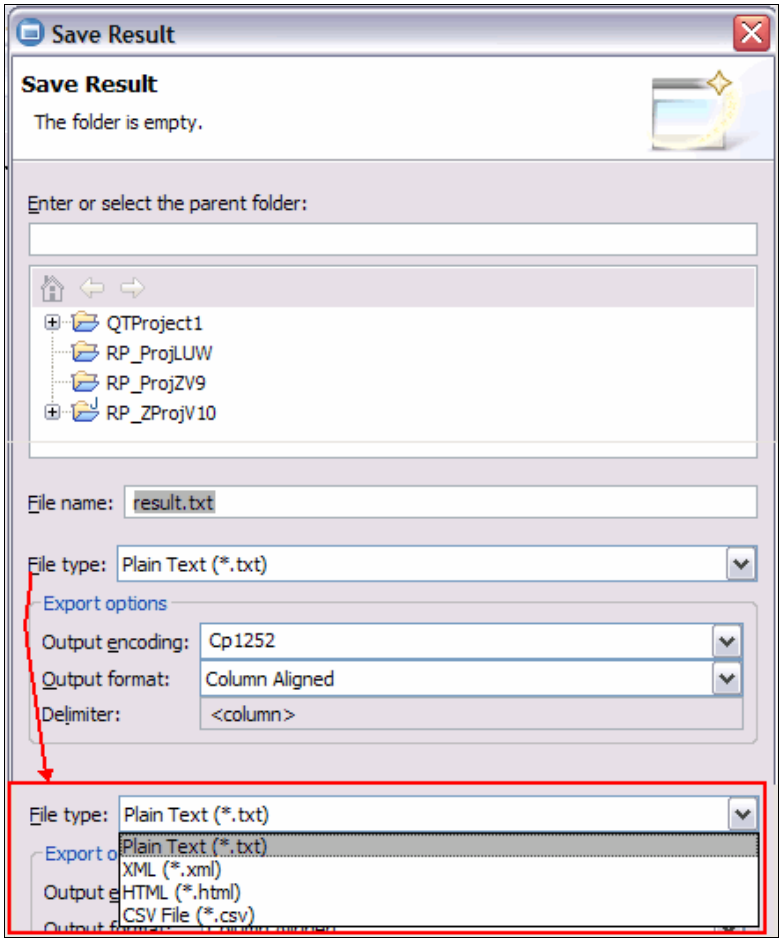

Figure 2-28 Output view - Save Results

3

Additional development features in the Data Studio products

In this chapter we provide a description of additional features used for stored procedures development offered by the Data Studio and Optim Development Studio products.

This chapter contains the following sections:

- Additional features in Data Studio
- Additional features in Optim Development Studio

3.1 Additional features in Data Studio

The features in Data Studio discussed in this section do not fall into the mainstream stored procedure development, and are discussed here for completeness.

3.1.1 Native SQL procedure new version

You can create new versions of native SQL stored procedures that are still under development in the Data Project Explorer or that have been previously deployed and listed in the Data Source Explorer. From the Data Project Explorer, you can also set which version of the stored procedure is the active version. The active version is the version that DB2 for z/OS executes if a version number is not specified.

Right-click a stored procedure and select the **New Version** action menu item. The New Version wizard is launched. This is a one-page wizard that allows you to name the version (Figure 3-1).

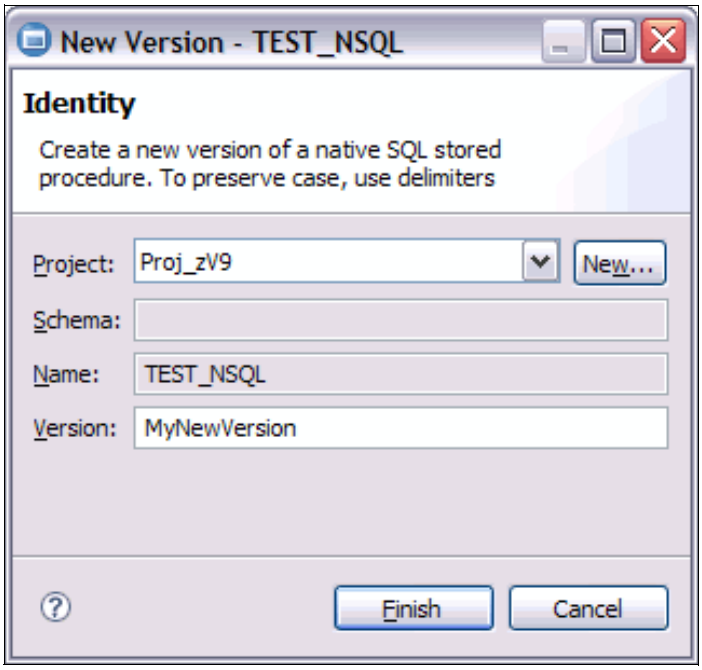

Figure 3-1 New Version wizard

3.1.2 Developing templates

Data Studio V2.2.1 ships with a set of default templates for external SQL, native SQL, and Java stored procedures. You can use these templates as the basis for more complicated templates. To create or modify an existing template, go to **Window** → **Preferences** → **Data Management** → **SQL Development** → **Routines** → **Templates**. Click **New** to launch the New Template dialog (Figure 3-2).

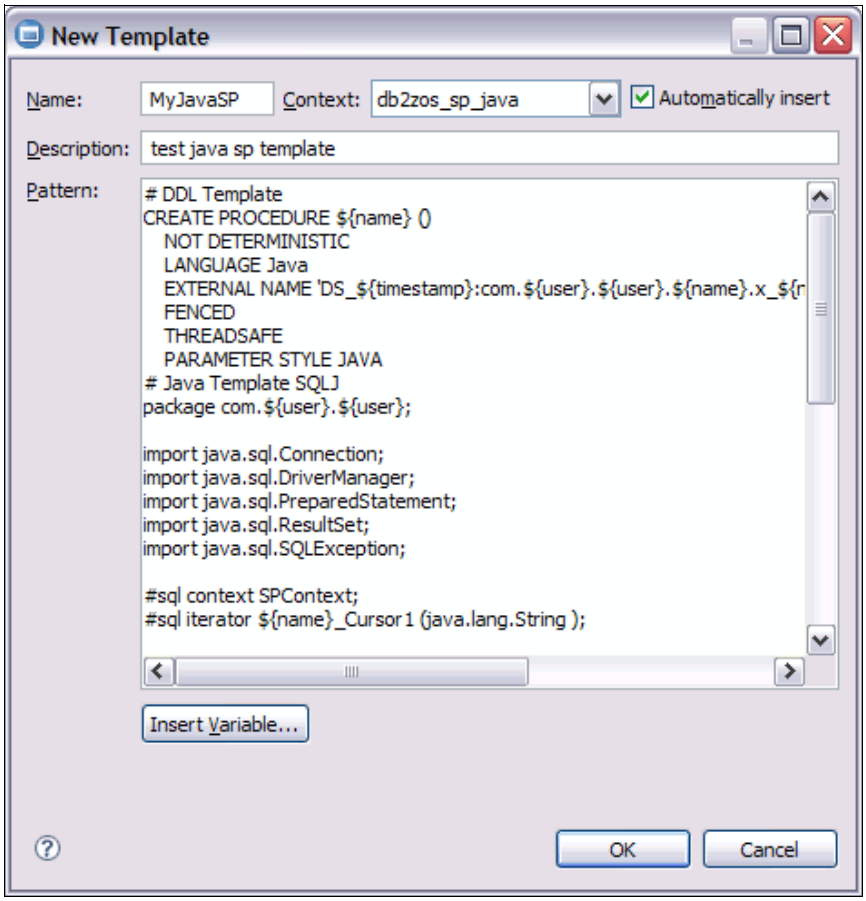

Figure 3-2 New Template dialog

Type or copy and paste the CREATE PROCEDURE DDL for either a SQL or Java stored procedure. For Java stored procedures, type or copy and paste the Java source as well.

Data Studio allows you to insert variables into the template. To insert a variable, position the cursor to where you want the variable inserted, then click **Insert Variable**. Data Studio allows automatic insertion of the variables listed in Table 3-1.

Table 3-1 Insertion variables

Variable name	Description
collid	Collection name (z/OS only)
date	Current date
name	Routine name
runOpts	Runtime options (z/OS only)
schemaName	Schema name

time	Current time
timestamp	Current timestamp
user	User name
wlmEnvironment	WLM environment (z/OS only)
year	Current year

3.1.3 CURRENT SQLID and CURRENT SCHEMA usage in Data Studio

The CURRENT SCHEMA value can be predefined in the project's properties. The value of this special register is used as the default target schema for:

- A stored procedure deployed without a fully qualified name
- An unqualified table used in a SQL script

In the Data Project Explorer, right-click the stored procedure and select **Properties** → **Database Connections** → **Default Schema** to set the CURRENT SCHEMA.

Table 3-2 summarizes setting the current schema in the project's properties during various tasks in Data Studio. Assume that the connection login ID is PAOLOR5.

Table 3-2 Current schema behavior

Current schema[a]	**Name specified in New SP wizard**	**SYSROUTINES SCHEMA column value**
blank	PROC1	PAOLOR5
blank	DEVL4717.PROC1	DEVL4717
DEVL4717	PROC1	DEVL4717
DEVL4717	PAOLOR1.PROC1	PAOLOR1

a. This is the value set in the project properties.

3.1.4 Package owner and build owner

In Data Studio, the package owner and build owner can also be predefined in your project's properties. In the Data Project Explorer, right-click the stored procedure and then select **Properties** → **Routine Development** → **Package owner** and **Build owner** to set these values.

The Package Owner field is used to define the authorization ID of the owner of the bound packages for a SQL or Java (SQLJ) stored procedure during deployment. The package owner can be overridden during stored procedure deployment by specifying the OWNER bind option in the bind options field. See "Deploy Options tab" on page 74.

When the Build Owner field is set in the project properties, the routine deployment uses this build owner's authority instead of the connection login IDs. Basically, the CURRENT SQLID is set to the value of build owner before the deployment starts. Therefore, the connection login ID must be able to issue SET CURRENT SQLID to this build owner. Binary deployment is a special case. It cannot use the build owner's authorization. In this scenario, the connection login ID itself must have proper privilege to create the stored procedure in the remote target server.

3.1.5 Export and deploy stored procedures

Data Studio allows you to export one or more stored procedures to the file system and deploy these stored procedures outside the tooling to a target server. You can optionally zip the files exported, then port the zip file to another workstation where you can issue the deploy.

You can export native SQL, external SQL, and Java stored procedures. Note, however, that deploying native SQL stored procedures to another DB2 9 or 10 for z/OS server outside of Data Studio, requires either a DB2 Connect or a DB2 on LUW database system running on the client issuing the deploy.

Exporting stored procedures

To export stored procedures:

1. In the Data Project Explorer, right-click the **Stored Procedures** folder and then click **Export**. The Export Wizard is launched. The first page is the Selection page.
2. Select the stored procedures that you want to export (Figure 3-3). Click **Next**.

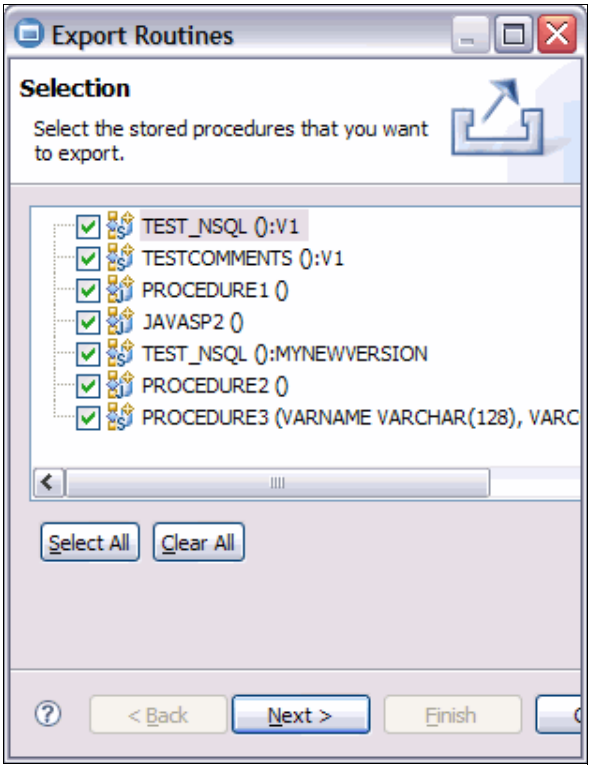

Figure 3-3 Export Wizard Selection page

3. On the Target and Location page, enter a file name for the exported `xml` file. Enter a directory location or click **Browse** to choose a directory.

4. Click the **Include DROP statements** check box. Figure 3-4 shows the completed page.
5. Click **Finish**.

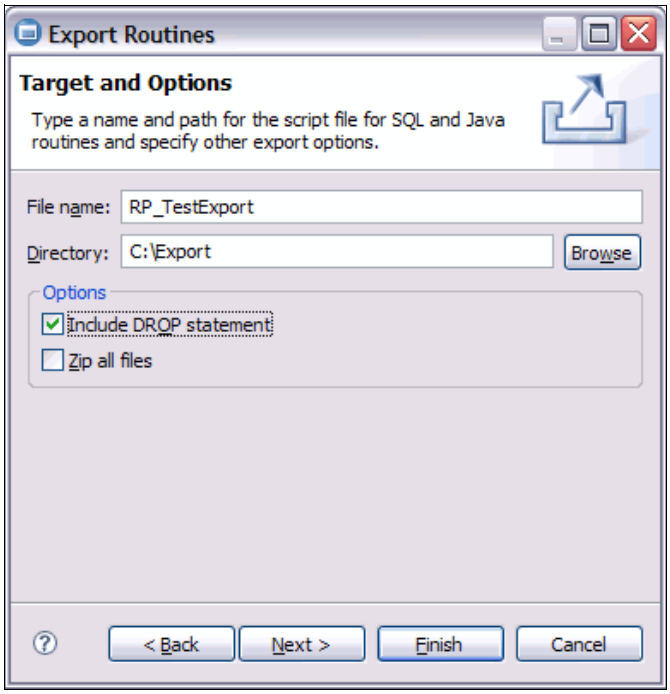

Figure 3-4 Export Wizard Target File name and location

6. Verify that the export is successful for each stored procedure in the Output view. In our example, we exported four stored procedures. The Output view in Figure 3-5 shows four entries, all of which are successful.

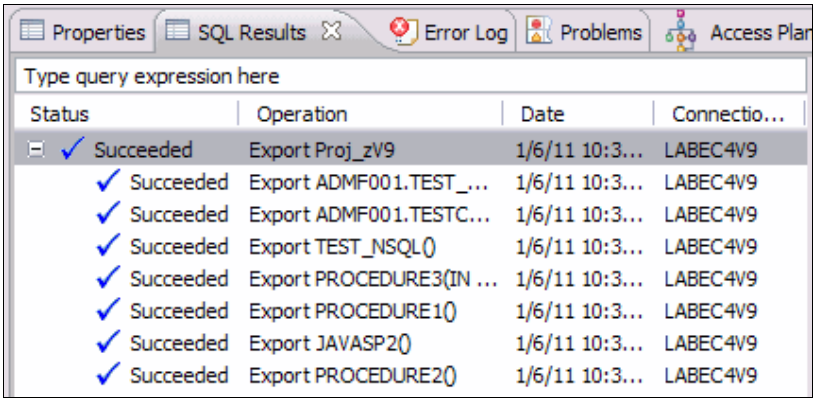

Figure 3-5 Export wizard, Output View Status table

Deploying the exported stored procedures

Data Studio generates several files to facilitate deploying the exported stored procedures and places these files in the directory specified during export.

Before deploying the stored procedures, ensure that your client's JAVA_HOME environment variable is pointing to a JDK level that is compatible with the JDK level at the server. The generated `ant.bat` file uses the JAVA_HOME setting when launching ant.

Open a DB2 command window and go to the directory that you specified for your exported files.

Deploying the native SQL stored procedures

To do this:

1. Edit the `<target export file>.sql` file.
2. Uncomment the CONNECT TO statement.
3. Update the user ID and password to the login user ID and password that you use to connect to the target server.
4. Modify the SET CURRENT SCHEMA statement to set a schema for unqualified database objects.
5. Uncomment the CONNECT RESET statement.
6. Uncomment the at sign (@).
7. Save the file.
8. Open a DB2 command window.
9. Run the following command:

```
db2 -td@ -vf <target export file>.sql
```

Deploying an external SQL stored procedure

To do this:

1. Edit the `<target export file>_sql.properties` file.
2. Update the properties where required. For example, the target connection properties are defined in the following properties:

```
db.userid=admf001
db.password=c0deshop
db.name=STLEC1
db.hostname=labec4.vmec.svl.ibm.com
db.port=446
```

3. Save the `<target export file>_sql.properties` file.
4. Run `ant.bat` to execute the script. Type `ant -buildfile <target export file>_sql.xml`.

Example 3-1 shows the output of the above **`ant`** command for an external SQL stored procedure.

Example 3-1 Output of ant deploy of stored procedure

```
Buildfile: ITSO_Project_sql.xml

init:

builddeploySps:
 [createsp] Debug options:
 [createsp]     file:/C:/Export/.options loaded
 [createsp] Could not connect to the target database.
 [createsp] [ibm][db2][jcc][t4][2013][11249] Connection authorization failure
occurred.  Reason: User ID or Password invalid.

BUILD SUCCESSFUL
Total time: 3 seconds
```

Deploying a Java stored procedure

To do this:

1. Edit the `<target export file>_java.properties` file.
2. Update the properties where required. The target connection properties are defined in the following properties:

   ```
   db.userid=admf001
   db.password=c0deshop
   db.name=STLEC1
   db.hostname=labec4.vmec.svl.ibm.com
   db.port=446
   ```

3. Save the `<target export file>_java.properties` file.
4. Run `ant.bat` to execute the script. Type `ant -buildfile <target export file>_java.xml`.

Excerpts from the output of the ant deploy of Java stored procedures are shown in Example 3-2.

Example 3-2 Output of ant deploy of SQLJTEST and JDBCTEST

```
Buildfile: ITSO_Project_java.xml

init:

builddeploySps:
 [createsp] Debug options:
 [createsp]     file:/C:/Export/.options loaded
 [createsp] DEVL7083.SQLJTEST - Deploy started.
[createsp] DEVL7083.SQLJTEST - Created temporary working directory
C:\Export\bld1196747605594.
 [createsp] DEVL7083.SQLJTEST - Translating the SQLJ source file
C:\Export\bld1196747605594\marichu\SQLJTEST.sqlj using
 [createsp]     SQLJ translator class: sqlj.tools.Sqlj
 [createsp]     SQLJ translator location: C:\SQLLIB\java\sqlj.zip;C:\Program...
 [createsp] DEVL7083.SQLJTEST - SQLJ translation completed.
 [createsp] C:\IBM_JDK15\bin\javac -classpath ".;C:\SQLLIB\java\sqlj.zip;C:\Program...
 [createsp] DEVL7083.SQLJTEST - Javac completed.
 [createsp] DEVL7083.SQLJTEST - Class file updated.
 [createsp] C:\IBM_JDK15\bin\javaw -cp ".\;C:\Program... C:\Export\bld1196747605594"
com.ibm.db2.jcc.sqlj.Customizer -url jdbc:db2://wtsc63.itso.ibm.com:12347/DB9A -collection
DSNJDBC -qualifier PAOLOR4 -user PAOLOR4 -password xxxxxxxx -bindoptions "QUALIFIER
PAOLOR4" -rootPkgName S928288 marichu\SQLJTEST_SJProfile0.ser

 [createsp] [jcc][sqlj] Begin Customization
 [createsp] [jcc][sqlj] Set qualifier for online checking to SCHEMA: PAOLOR4
 [createsp] [jcc][sqlj] Loading profile: marichu\SQLJTEST_SJProfile0
 [createsp] [jcc][sqlj] Customization complete for profile marichu\SQLJTEST_SJProfile0.ser
 [createsp] [jcc][sqlj] Begin Bind
 [createsp] [jcc][sqlj] Loading profile: marichu\SQLJTEST_SJProfile0
 [createsp] [jcc][sqlj] User bind options: QUALIFIER PAOLOR4
 [createsp] [jcc][sqlj] Driver defaults(user may override): BLOCKING ALL VALIDATE BIND
 [createsp] [jcc][sqlj] Fixed driver options: DATETIME ISO DYNAMICRULES BIND
 [createsp] [jcc][sqlj] Binding package S9282881 at isolation level UR
 [createsp] [jcc][sqlj] Binding package S9282882 at isolation level CS
 [createsp] [jcc][sqlj] Binding package S9282883 at isolation level RS
 [createsp] [jcc][sqlj] Binding package S9282884 at isolation level RR
 [createsp] [jcc][sqlj] Bind complete for marichu\SQLJTEST_SJProfile0
```

```
 [createsp] DEVL7083.SQLJTEST - SQLJ profile customization completed.
 [createsp] C:\IBM_JDK15\bin\jar uf spjar.jar  marichu\SPContext.class
marichu\SQLJTEST.class marichu\SQLJTEST_Cursor1.class marichu\SQLJTEST_SJProfile0.ser
marichu\SQLJTEST_SJProfileKeys.class
 [createsp] DEVL7083.SQLJTEST - Jar file created.
 [createsp] DELETE FROM SYSIBM.SYSJAVAOPTS WHERE JARSCHEMA = 'PAOLOR4' AND JAR_ID =
'SQLJTEST'
 [createsp] Call SQLJ.DB2_REPLACE_JAR (<<C:\Export\bld1196747605594\spjar.jar>>,
'PAOLOR4.SQLJTEST')
 [createsp] DEVL7083.SQLJTEST - SQLJ.DB2_REPLACE_JAR using Jar name PAOLOR4.SQLJTEST
completed.
 [createsp] Call ALTER_JAVA_PATH ('PAOLOR4.SQLJTEST', '')
 [createsp] DEVL7083.SQLJTEST - SQLJ.ALTER_JAVA_PATH using Jar name PAOLOR4.SQLJTEST
completed.
 [createsp] DEVL7083.SQLJTEST - Supporting Jars installed successfully.
 [createsp] Call SQLJ.DB2_UPDATEJARINFO ('PAOLOR4.SQLJTEST', 'marichu.SQLJTEST',
<<C:\Export\bld1196747605594\marichu\SQLJTEST.sqlj>>, 'S928288', 'DSNJDBC',
'ACTION(REPLACE)')
 [createsp] DEVL7083.SQLJTEST - Source saved to the server.
 [createsp] DEVL7083.SQLJTEST - Removed temporary working directory
C:\Export\bld1196747605594.
 [createsp] DSNT540I  DB9AWLMJ WAS REFRESHED BY PAOLOR4  USING AUTHORITY FROM SQL ID
PAOLOR4 : 0
 [createsp] DEVL7083.SQLJTEST - Deploy successful.
 [createsp]
 [createsp] ====================================

 [createsp] DEVL7083.JDBCTEST - Deploy for debug started.
 [createsp] DEVL7083.JDBCTEST - Created temporary working directory
C:\Export\bld1196747618983.
 [createsp] C:\IBM_JDK15\bin\javac -classpath "..... -g -source 1.4 -target 1.4
marichu\JDBCTEST.java
 [createsp] DEVL7083.JDBCTEST - Javac completed.
 [createsp] C:\IBM_JDK15\bin\jar uf spjar.jar  marichu\JDBCTEST.class
 [createsp] DEVL7083.JDBCTEST - Jar file created.
 [createsp] DELETE FROM SYSIBM.SYSJAVAOPTS WHERE JARSCHEMA = 'PAOLOR5' AND JAR_ID =
'JDBCTEST'
 [createsp] Call SQLJ.DB2_REPLACE_JAR (<<C:\Export\bld1196747618983\spjar.jar>>,
'PAOLOR5.JDBCTEST')
 [createsp] DEVL7083.JDBCTEST - SQLJ.DB2_REPLACE_JAR using Jar name PAOLOR5.JDBCTEST
completed.
 [createsp] Call ALTER_JAVA_PATH ('PAOLOR5.JDBCTEST', '')
 [createsp] DEVL7083.JDBCTEST - SQLJ.ALTER_JAVA_PATH using Jar name PAOLOR5.JDBCTEST
completed.
 [createsp] DEVL7083.JDBCTEST - Supporting Jars installed successfully.
 [createsp] Call SQLJ.DB2_UPDATEJARINFO ('PAOLOR5.JDBCTEST', 'marichu.JDBCTEST',
<<C:\Export\bld1196747618983\marichu\JDBCTEST.java>>, '', 'DSNJDBC', 'ACTION(REPLACE)')
 [createsp] DEVL7083.JDBCTEST - Source saved to the server.
 [createsp] DEVL7083.JDBCTEST - Removed temporary working directory
C:\Export\bld1196747618983.
 [createsp] DSNT540I  DB9AWLMJ WAS REFRESHED BY PAOLOR4  USING AUTHORITY FROM SQL ID
PAOLOR4 : 0
 [createsp] DEVL7083.JDBCTEST - Deploy for debug successful.
 [createsp]
 [createsp] ====================================


BUILD SUCCESSFUL
Total time: 24 seconds
```

3.1.6 Deploying SQL or Java stored procedures without recompiling

Some customers want to migrate compiled code and not rebuild the stored procedure on the target server. Data Studio's Deploy wizard can be used to deploy using binaries—SQL or Java stored procedures between source and target servers of the same platform.

External SQL stored procedures

Deploying external SQL stored procedures without rebuilding requires taking the following steps from the tooling:

1. Right-click the stored procedure and then click **Deploy**.
2. In the Deploy Wizard, change the target connection to **Use Different Database**. Select your target database from the pull-down list, or create a new connection to this database (see 2.1.2, "Creating a connection profile" on page 51).
3. If the stored procedure is unqualified, select the schema name to be used for this stored procedure.
4. Set your duplicate and error handling options.
5. Click the **Deploy using binaries, if available in the database** check box.
6. The target load library is enabled. Specify a partitioned data set (PDS) file to receive the load module in the target database.
7. Click **Next** to change the deploy and routine options, as discussed in 2.5, "Deploying a stored procedure" on page 71.
8. Click **Finish** to complete the deploy.

Figure 3-6 shows the completed Deploy wizard.

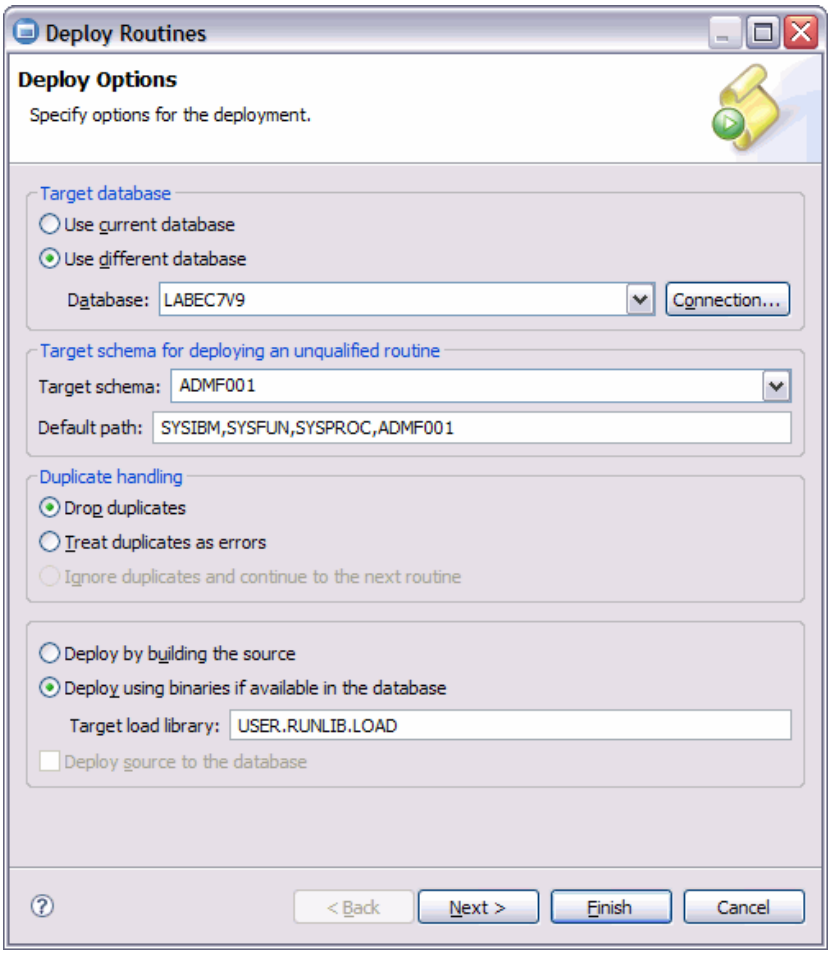

Figure 3-6 Deploy using binaries

For an external SQL stored procedure, the DBRM, package, and load module are copied from the source database to the target database, and the CREATE PROCEDURE DDL is executed to catalog the stored procedure in the target database. The SQL body is not copied to SYSROUTINES_SRC.

Native stored procedures

To deploy native stored procedures using binaries:

1. Right-click the stored procedure and then select **Deploy**.
   a. In the Deploy Wizard → Target connection, click **Use Different Database**.
   b. Select your target database from the pull-down list, or create a new connection to this database (see 2.1.2, “Creating a connection profile” on page 51).
   c. If the stored procedure is unqualified, select a schema name to be used for this stored procedure.
   d. Set your duplicate and error handling options.
   e. Click **Deploy using binaries, if available in the database**.
   f. Click **Next** to change the deploy and routine options, as discussed in 2.5, “Deploying a stored procedure” on page 71.
   g. Click **Finish** to complete the deploy.

Java stored procedures

Data Studio supports deploying using binaries only if the Java stored procedure was initially built on the client and the connection used the IBM Universal driver. Java stored procedures built using the utility DSNTJSPP are not eligible to be deployed using binaries.

The steps to deploy Java stored procedures using binaries are the same as those followed for native SQL stored procedures. There is no need to specify a target load library, as there is no load module involved. The stored procedure jar files are obtained from the source database and then installed in the target database. The java source code can be optionally copied if the **Deploy source to the database** check box is selected.

Note: Data Studio V2.2.1 supports deploy using binaries from DB2 9 to DB2 10 for z/OS but *not* vice versa. The DB2-supplied stored procedures used to perform this task were renamed in DB2 10. See APAR IC74104 for more details.

3.1.7 Managing privileges

When deploying an existing stored procedure to another database, you might want to set up the same authorizations in the target server. To do this:

1. Open the Database Administration perspective and look for the source database connection in the Administration Explorer view.
2. Expand the **Application Objects** folder. Click the **Stored Procedures** folder. The Object Editor lists all the stored procedures available in the server. You can filter this list as discussed in “The Object Editor” on page 36.
3. Right-click the stored procedure that you just deployed to the target database and select **Manage Privileges**.

4. Select the users who you want to give execute privileges to by clicking the corresponding check box (Figure 3-7).
5. Click **Preview DDL** to see the GRANT statements.
6. Click **Open with SQL Editor** to persist the statements into a script in the project. This is suggested, as you are able to run this script against multiple servers.
7. Click **Run DDL** to execute the GRANT statements against the currently connected server.

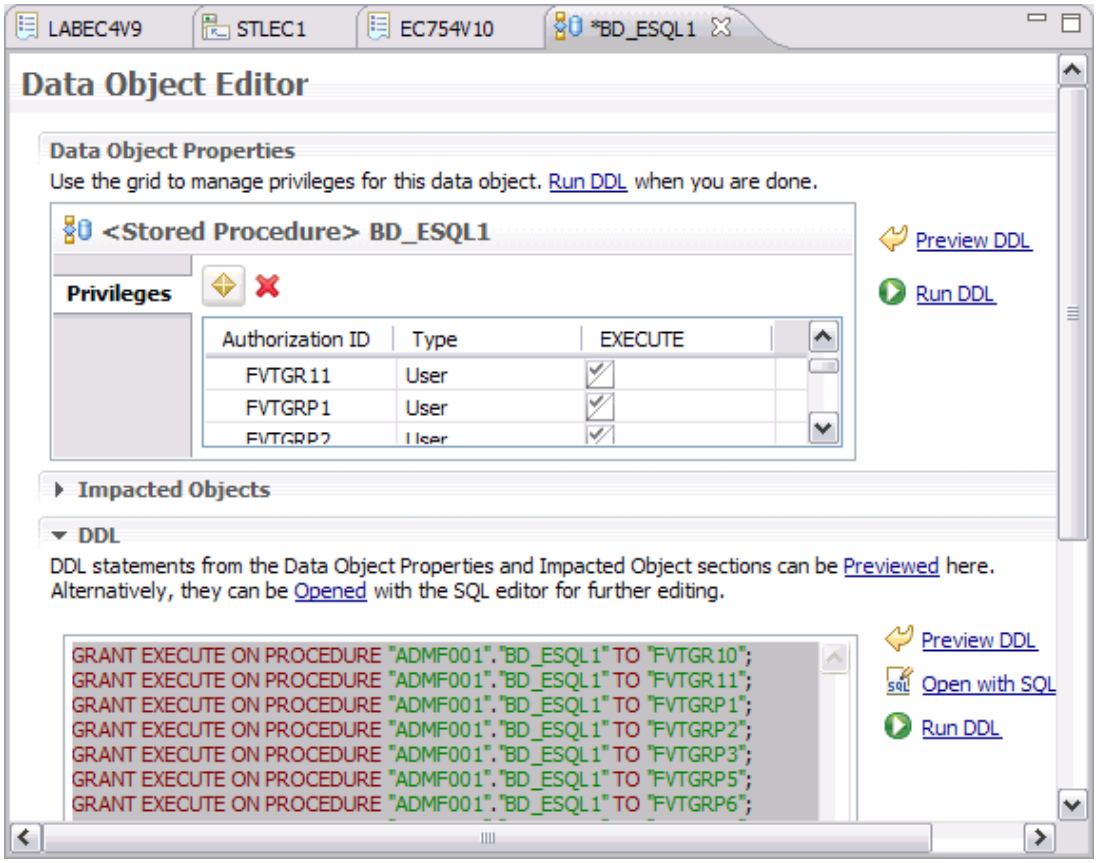

Figure 3-7 Manage privileges

3.1.8 Creating package variations

Data Studio supports creating packages with different bind options for SQL and SQLJ Java stored procedures on DB2 for z/OS[1]. The packages are created in different collection IDs. Use the New Package Variation wizard (Figure 3-8) to create a package variation of an existing stored procedure package.

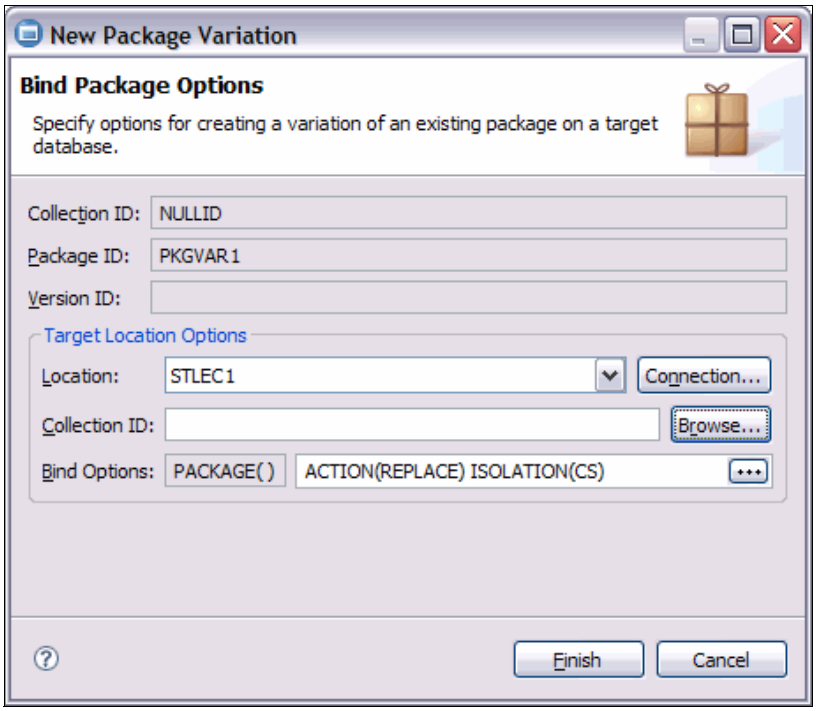

Figure 3-8 New Package Variation wizard

Creating a package variation

To do this:

1. Initially deploy the external SQL or SQLJ stored procedure on a DB2 for z/OS server using the NO COLLID option.
2. Bind the COLLID that you want the new package to reside in using com.ibm.db2.jcc.DB2Binder. Otherwise, you get a -805 at execution time.
3. From the Data Source Explorer, expand the connection to the newly deployed stored procedure.
4. Right-click the stored procedure and select **Packages** → **New**. The New Package Variation wizard is launched.
5. Select a target location. The default is the current database connection.
6. Select a collection ID. Click **Browse** to view a list of available collection IDs.
7. Specify the bind options for this package.
8. Click **Finish** to deploy the new package to the server. The new package is added to the `Packages` folder of the stored procedure.

[1] This applies to Java stored procedures using SQLJ and external SQL stored procedures, not to native SQL procedures. Native SQL procedures packages are DB2-controlled.

Executing a stored procedure using a specific package

To do this:

1. Right-click the stored procedure and then select **Run Settings** → **Options** tab.
2. Type the collection ID associated with the package version that you want to execute in the Collection ID field. Click **Browse** to view a list of the collection IDs of the packages associated with this stored procedure. Click **OK**.
3. Right-click the stored procedure and then click **Run** to execute the stored procedure.

For more information about package variation, see the article “Create package variations for z/OS DB2 stored procedures” on developerWorks at:

`http://www.ibm.com/developerworks/db2/library/techarticle/dm-0608parmeshwar/`

Note: If the user did not specify a package ID, the tooling generates a package ID with a value of SQL and a randomly generated number. After deploying the stored procedure, you can browse the deployed stored procedure’s properties in the Data Source Explorer using the property browser. The generated package ID is in the Options tab of the stored procedure’s Properties view. You can also browse the package variation properties in the same manner.

3.1.9 Multiple jar support for Java stored procedures

This feature, available starting with DB2 9 for z/OS, allows users to reference classes in supporting `jar` files. A *supporting jar* is a `jar` file that is referenced by another `jar` file at run time. Java stored procedures can reference classes that are either in the CLASSPATH of the associated Java stored procedure WLM procedure or in the `jar` in which the stored procedure resides.

Data Studio associates an installed `jar` file with a Java path and allows the user to specify this Java path. To include supporting jars into a Java stored procedure, you must first import the jars into the `Jars` folder of the project.

Importing a jar file

To do this:

1. In the Data Project Explorer, open the project. Right-click it and then click **Jars** → **Import**.
2. In the Import wizard, specify the JAR ID. Click **Browse**.
3. A file browser is launched. Point to the location of the `jar` that you want to import and click **OK**.
4. Check the **Deploy** check box.
5. Optional: Set the current schema for the deployed `jar`. The default is CURRENT SQLID.

6. Figure 3-9 shows the completed page. Click **Finish**.

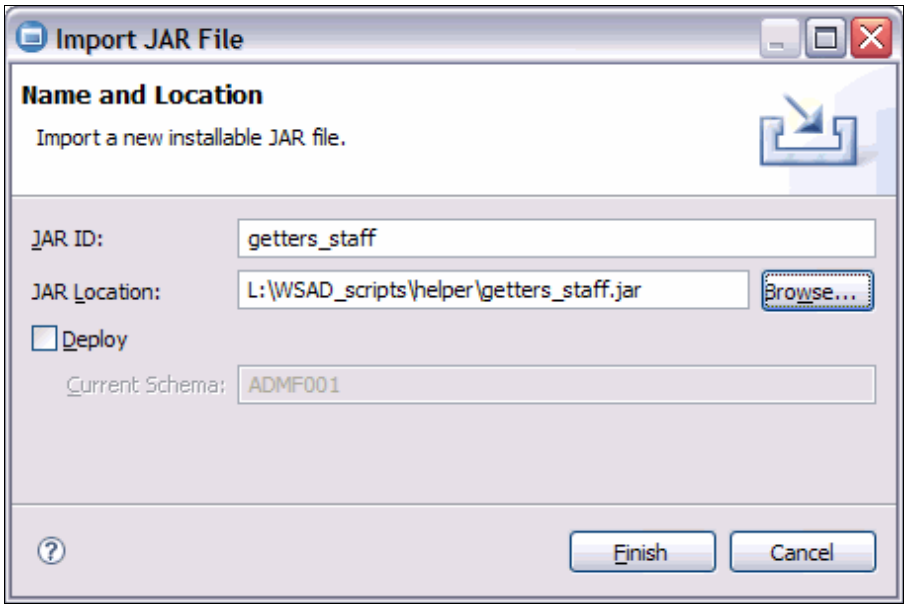

Figure 3-9 Import a jar file

Creating a Java stored procedure with a supporting jar

To do this:

1. Create a Java Stored procedure using the steps in 2.2, “Creating a new stored procedure” on page 60.
2. Select **Java** from the Language pull-down list of the New Stored Procedure wizard.
3. If the imported `jar` is in a different Java package, click the **Java** tab of the Preview section and add the necessary *import* statements in the Java source.
4. Click **Finish**.
5. Deploy the stored procedure.
6. On the Java Path tab of the Routine Options page of the Deploy wizard, click **Add**.
7. In the Add Installable Jar dialog, select the `jar` that you previously imported.
8. Click **Browse** next to Class Reference pattern.

9. Select the default class and click **OK**. Figure 3-10 shows the completed Java Path page.
10. Click **Finish**.

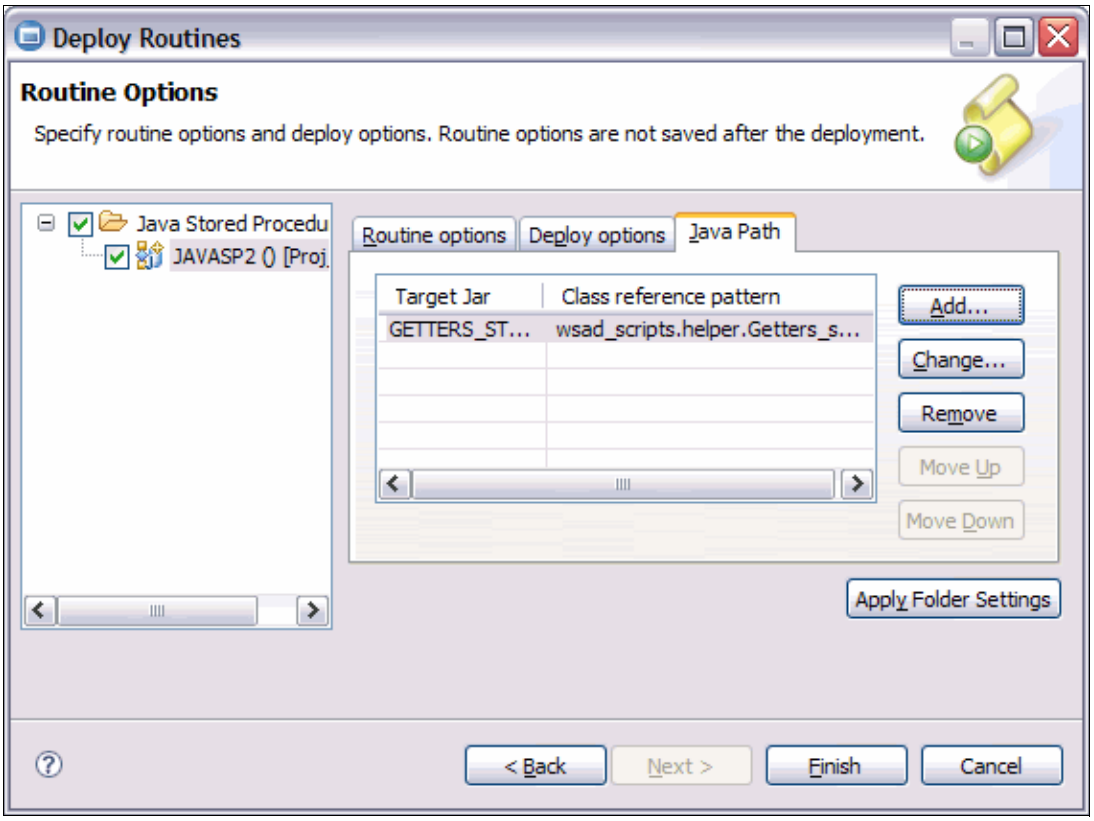

Figure 3-10 Add supporting jar for Java stored procedure

Data Studio creates a dependency on the added `jar` file and calls the ALTER_JAVA_PATH DB2-supplied stored procedure during the deploy. The supporting `jar` is also included in the `jar` file that is *installed* in the server.

You can also add previously deployed jars by dragging and dropping them from the Data Source Explorer into the `Jars` folder of the Data Development Project.

From the Data Source Explorer, you can also

- View or browse all jars installed on the server.
- View the catalog data and Java path in the Property Browser for each jar.
- View jars that are dependent on a particular jar.
- Drop supporting jars.

From the Data Project Explorer, you can:

- Deploy a `jar` to the server.
- Replace the `jar` if already installed in the server.
- Edit the supporting `jar` characteristics, such as the Java path during deploy.
- Deploy a Java stored procedure and all its supporting jars into the same `jar` file.

3.1.10 Creating a web service from a stored procedure

Data Studio offers a feature to add a stored procedure to an existing web services that exposes database operations (SQL SELECT and DML statements, XQuery expressions, or calls to stored procedures) to client applications.

For details, see *IBM Data Studio V2.1: Getting Started with Web Services on DB2 for z/OS*, REDP-4510.

To add a stored procedure to a web service:

1. Create a new web service. In the Data Explorer, select a project, then right-click the Web Services folder and select **New Web Service** (Figure 3-11).

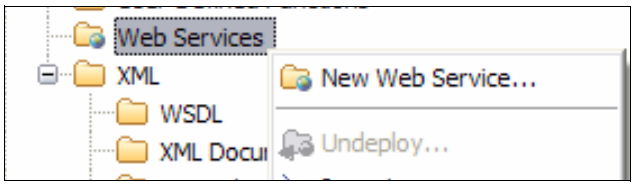

Figure 3-11 Create a new web service

2. The New Web Service dialog is displayed (Figure 3-12). Enter a name. You can specify your own namespace or use the default URI for this web service. Click **Finish**.

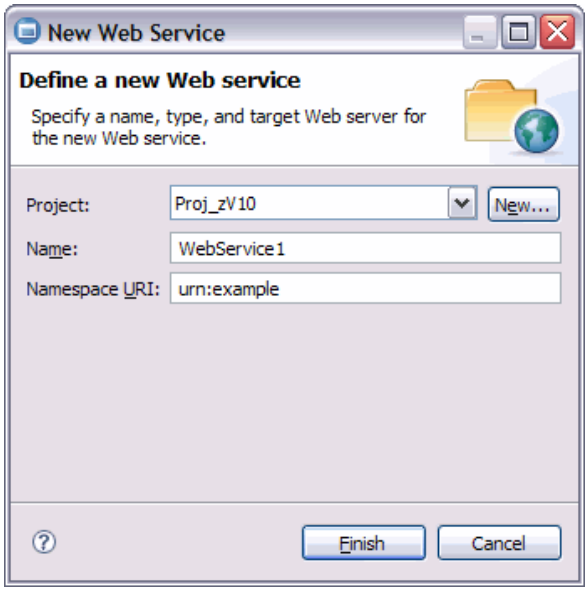

Figure 3-12 Define a new Web service

3. To add a stored procedure to this web service, Right-click the stored procedure and then select **Add to Web Service** (Figure 3-13).

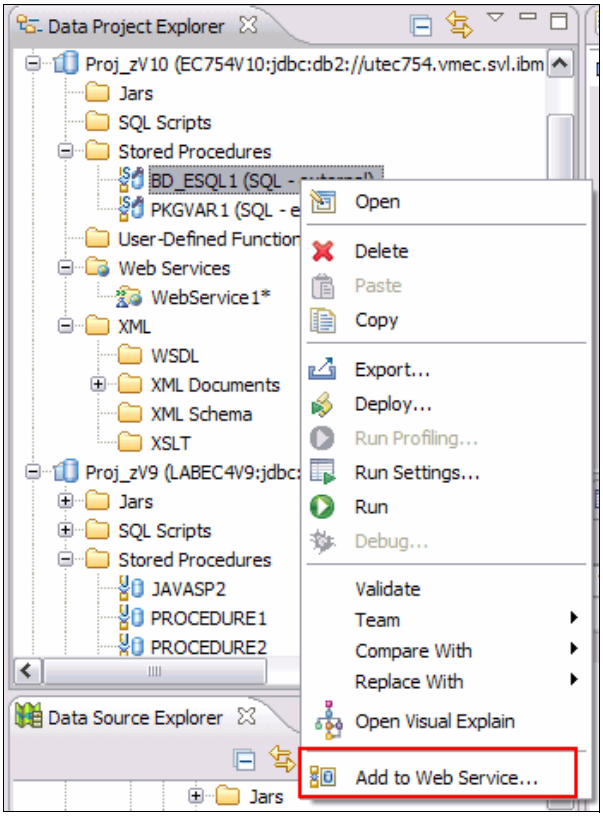

Figure 3-13 Add a stored procedure call to a web service

4. In the Add Operation to Web Services page, select the web service that you previously created in the left panel. Click **>**. The web service is moved to the right panel (Figure 3-14). Click **Next**.

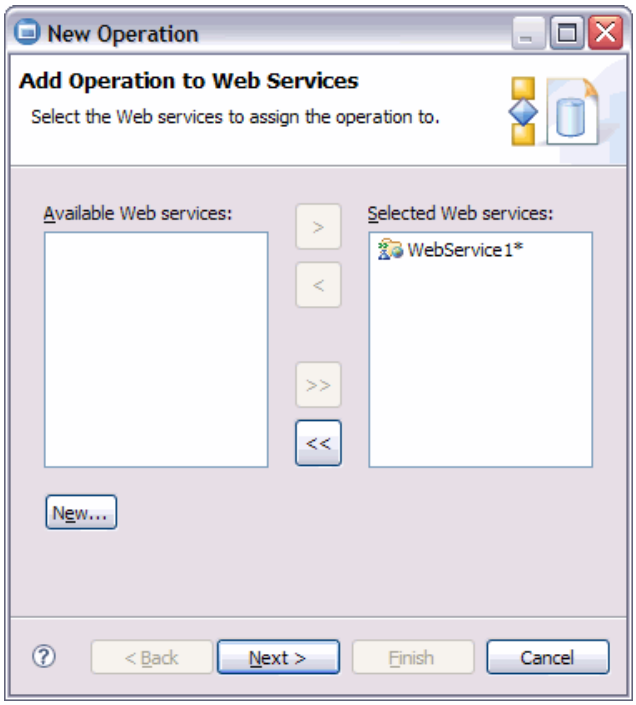

Figure 3-14 Select the web service to add this stored procedure to

5. In the Name and Operator page, the name and operation are prefilled with the stored procedure's name and a generated CALL to the stored procedure. Click **Next**.

   Data Studio can generate detailed or generic XML schemas for stored procedures that accept non-varying input values and return result sets that are always the same. To make the XML schema as detailed as possible, Data Studio needs to know the structure of these unchanging result sets. This information is obtained by running the stored procedure and capturing the input and result set information. The XML schema is generated from this run.

6. In the Generate XML Schema for Stored Procedure page (Figure 3-15), click **Generate**.

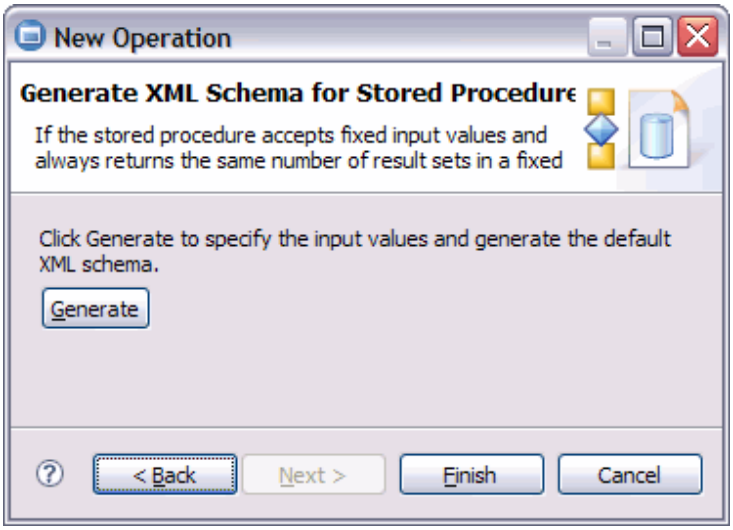

Figure 3-15 Generate XML Schema for stored procedure

7. The call to the stored procedure is executed, and you see the stored procedure result set in the Output view. Click **Finish**.
8. You can now deploy the web service to an application server like WebSphere Application Server or Tomcat. To deploy the web service, right-click the web service and select **Build and Deploy** from the context menu.

9. The Deploy Web Service dialog is launched (Figure 3-16). Click **Finish** to complete the deploy of the web service.

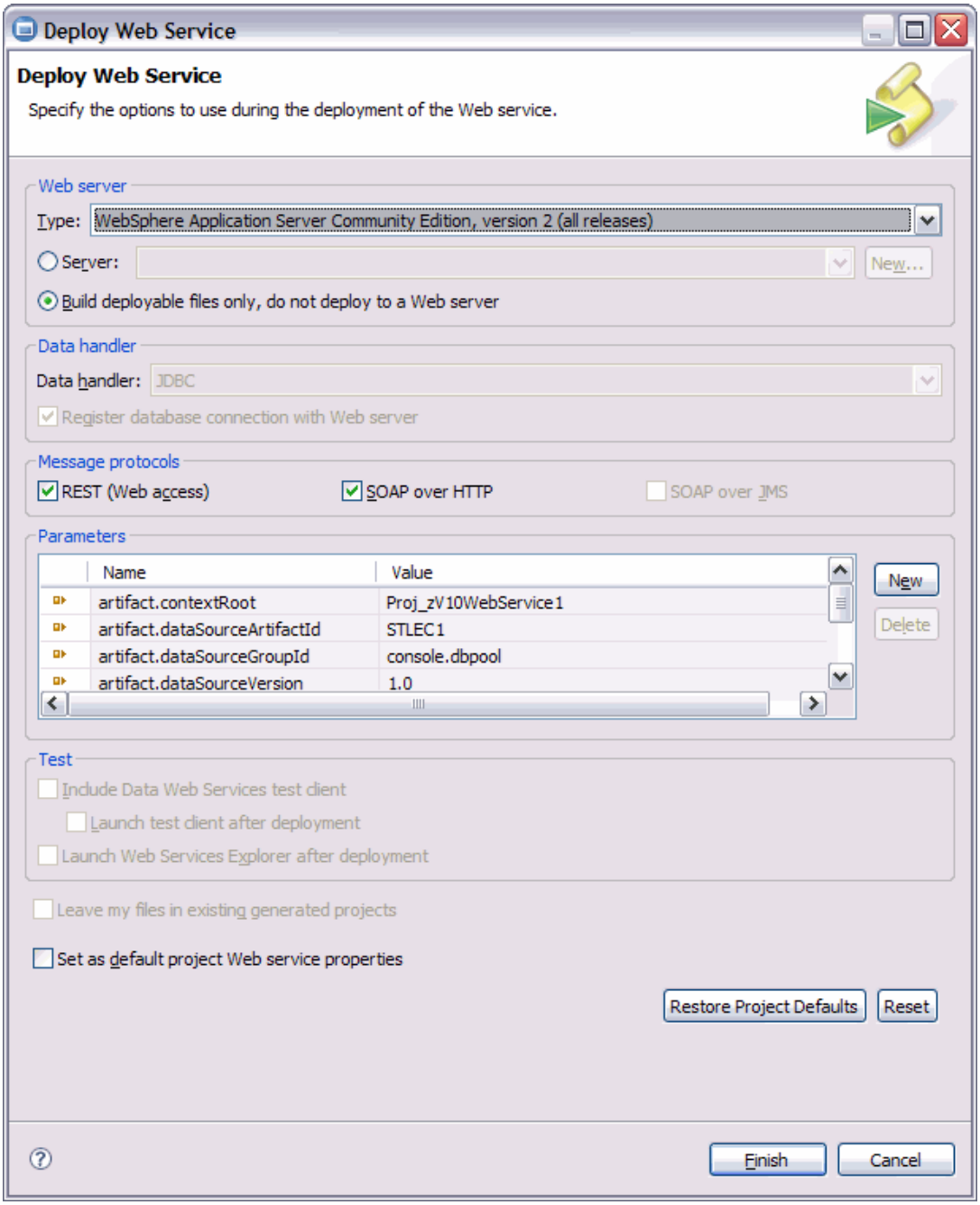

Figure 3-16 Specify options for deploying the web service

Data Studio generates the web service runtime files, such as the WebSphere Definition Language (WSDL) file. The project folder is refreshed and the WSDL is added to the XML folder (Figure 3-17).

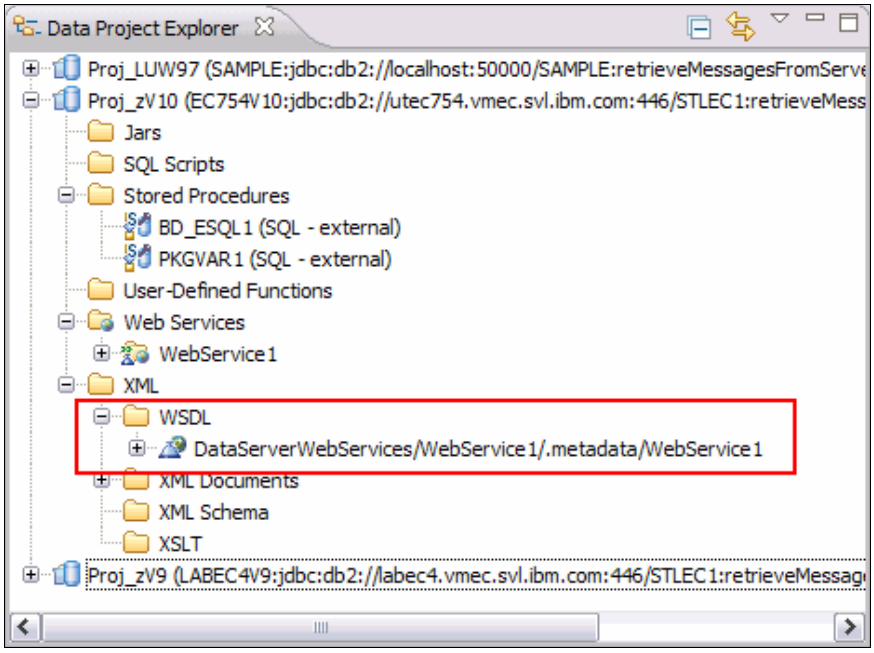

Figure 3-17 Generated WSDL file

3.2 Additional features in Optim Development Studio

Optim Development Studio provides you with additional capabilities to bring your stored procedure development into the enterprise level. These are:

- Configuration repository
- Enhanced connection error handling
- Server profiles
- Deployment groups
- pureQuery support

3.2.1 Configuration repository

As we saw in 2.1.2, “Creating a connection profile” on page 51, you can create and manage database connections to DB2 for z/OS in Data Studio using the New Connection profile wizard. After these connections have been created, you can share these connections by exporting the connection information to the file system and importing them into another (user’s) workspace.

In Optim Development Studio, you can organized and share these connections using a configuration repository. A configuration repository is a set of database tables that contain the information and properties of the shared connections.

The developerWorks article “Using common connections with Optim solutions” gives an excellent discussion and demonstration of how to create and manage database connections using configuration repositories. It is available from:

http://www.ibm.com/developerworks/data/library/techarticle/dm-0812devlin/index.html

3.2.2 Enhanced connection error handling

When you create a connection to DB2 for z/OS, this connection is created as a thread in DB2 for z/OS. If the client is not actively using the connection (as in when the user is simply editing or has left their workstation idle), if the idle thread timeout interval, IDTHTOIN, is reached, DB2 for z/OS severs this connection. Data Studio is not aware that the connection thread has been cancelled until a task requiring an active connection is done, such as retrieving an object's information, or deploying an object.

With Optim Development Studio 2.2.1, the connection is monitored and polled periodically. You can set the polling interval in **Window** → **Preferences** → **Data Management** → **Connectivity** → **Connection Management** (Figure 3-18).

.

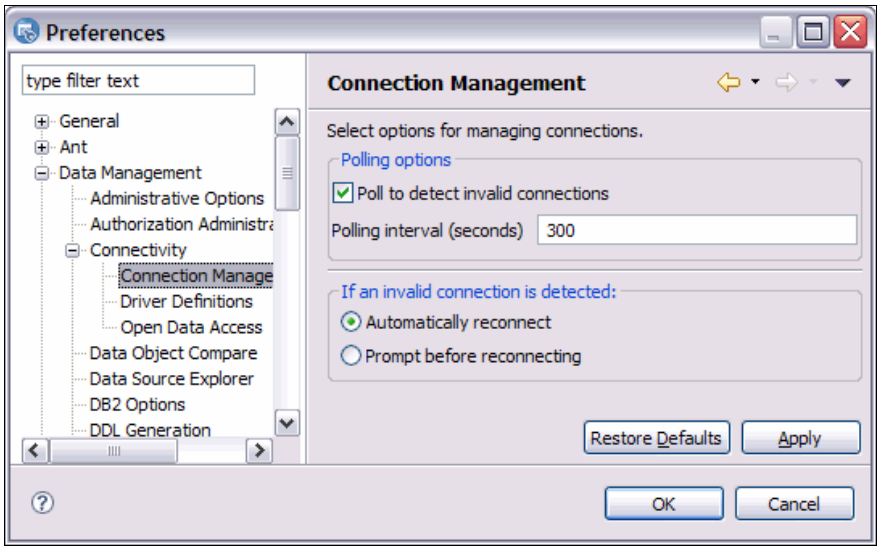

Figure 3-18 Preferences - Connection polling option

If the connection is lost, Optim Development Studio attempts to reconnect to the server. If Optim Development Studio cannot reconnect, it displays a message that the connection to this server has been lost. A dialog is launched to request an action from the user (Figure 3-19). Click **OK** to attempt to reconnect. Click **Cancel** to abort attempting to reconnect.

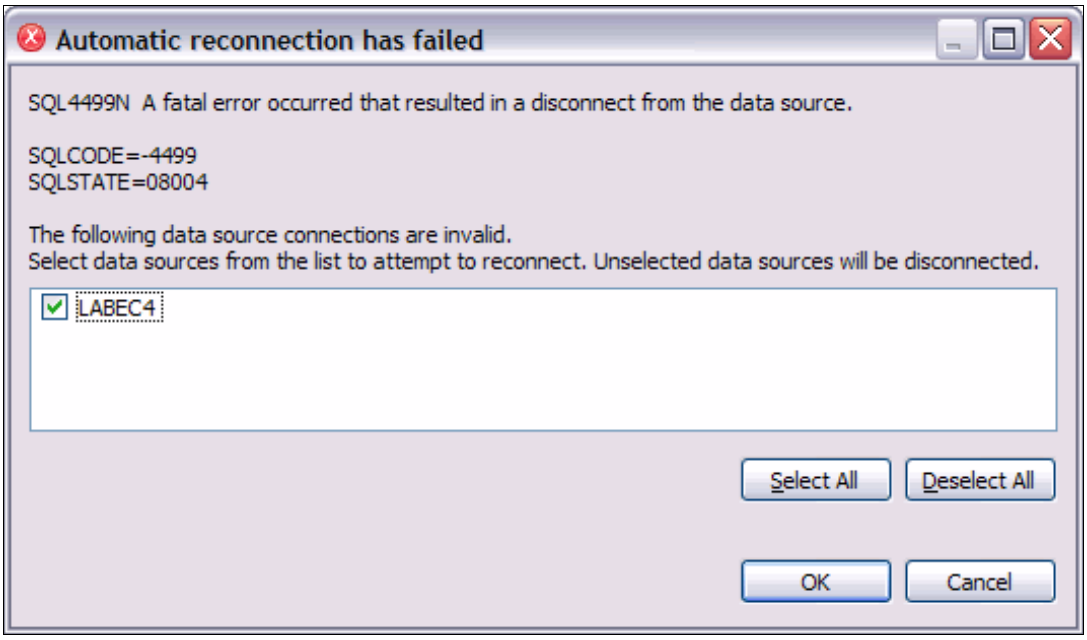

Figure 3-19 Autoreconnect failure

If your connection fails due to any other reason other than a lost connection, Optim Development Studio presents a dialog that explains the problem, and also gives you links to support information for debugging the problem (Figure 3-20).

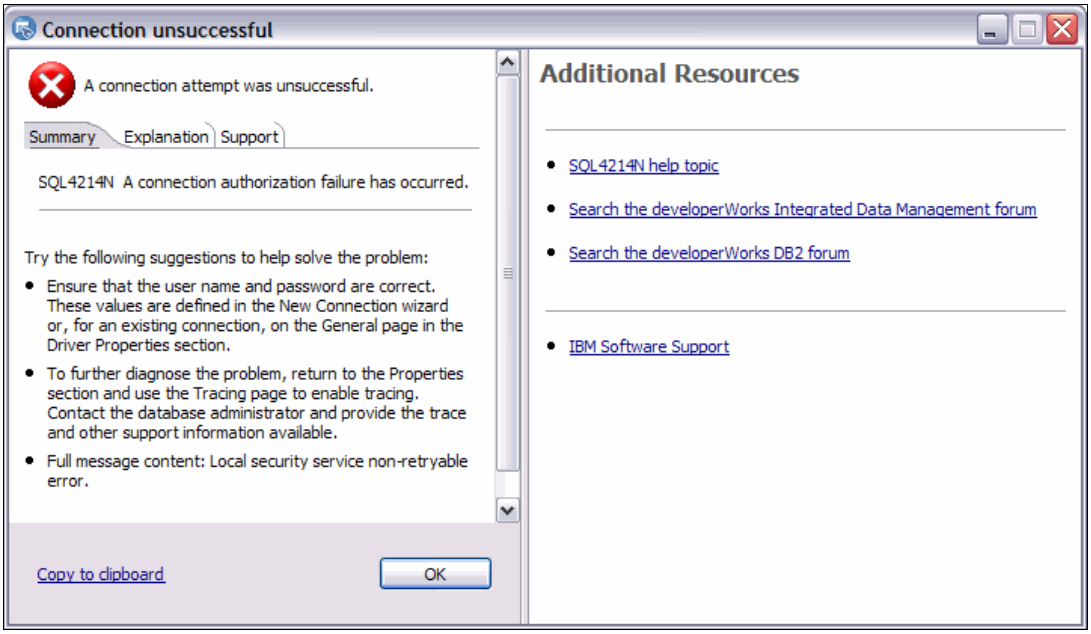

Figure 3-20 Standardized[2] connection error message

[2] Optim Development Studio continues to standardize error messages for all components in future releases.

3.2.3 Server profiles

The connection profiles that you create in Data Studio refer to connection profiles in the workspace containing the connection information necessary to connect to the server. The same connection can be used by various projects with different deployment needs.

Optim Development Studio 2.2.1 allows you to create server profiles, which contain not only the connection information but also a customized set of database settings, bind settings, and routine settings.

For each JDBC connection created in the Data Source Explorer, Optim Development Studio generates a server profile and lists this profile in the Server Profile Manager view (Figure 3-21).

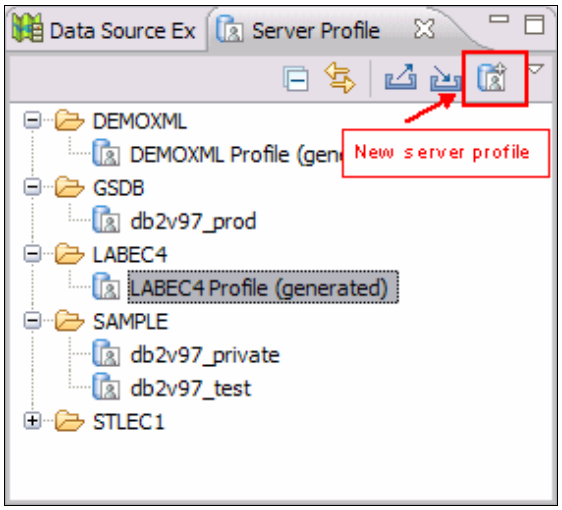

Figure 3-21 Server Profile Manager view

In addition, users can create their own non-generated server profiles with the New server profile toolbar button (Figure 3-22). Users can create multiple server profiles against the same connection, using different settings for different deployment needs.

Right-click a server profile icon and click **Open** or double-click a server profile to manage the settings for this profile.

As shown in Figure 3-22, you can set the:

- Special registers to be used for the database connection
- Default bind options and pureQuery static bind options
- Routine options for each type of routine

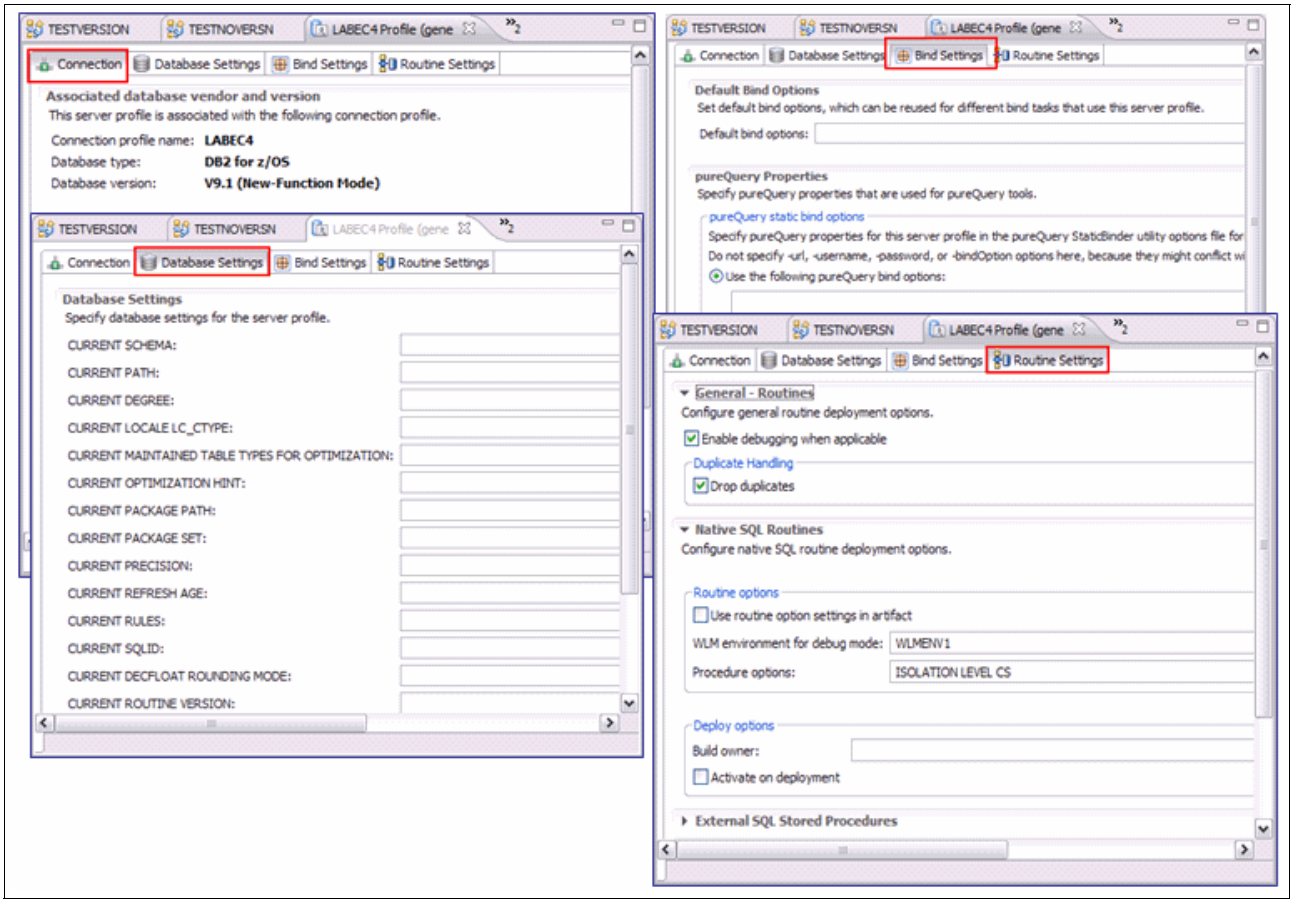

Figure 3-22 Multi-tab Server Profile property editor

The developerWorks article “IBM Optim Development Studio: Test deployment simplified” provides an excellent discussion on the use of server profiles in an enterprise environment. It is available from:

http://www.ibm.com/developerworks/data/library/techarticle/dm-1010testdeployment/index.html

3.2.4 Deployment groups

In Data Studio, when deploying one or more stored procedures from a project, there is a tight relationship between the target database server and the deployment options. If the user wanted to deploy some stored procedures with one set of options and another set of stored procedures with another set of options, he would have to either create two projects targeting

the same server or change the options each time that he redeploys. The deploy options are not persisted.

Optim Development Studio 2.2.1 introduces the concept of *deployment groups*. A deployment group is an ordered list of supported deployable artifacts from projects in your workspace. These artifacts include SQL Scripts, stored procedures, and user-defined functions. Deployment groups are associated with one or more server profile objects.

Deployment groups allow you to deploy artifacts in a repeatable and auditable manner. This is especially advantageous when the deployment order is important because of artifact dependencies and when working with other developers who might want to deploy the same set of artifacts in their environment.

The Deployment Manager view (Figure 3-23) lists the deployment groups, along with their associated artifacts and server profiles. The `Deployment Results` folder stores a history of the deployment.

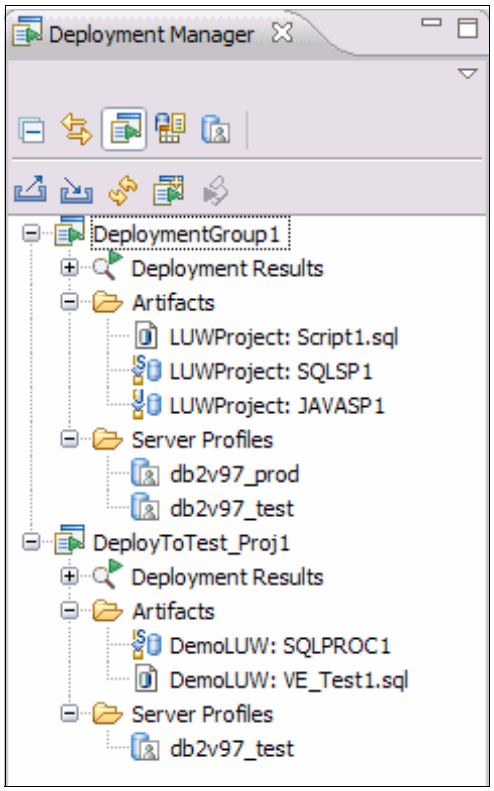

Figure 3-23 Deployment groups

As discussed in the previous section, server profiles are used to specify a target database connection, along with specific database, bind, and routine settings for deployment.

Deployment groups also keep track of the deployment history and status including timestamp information, status for a particular artifact in the group, and error/success results.

The developerWorks article “IBM Optim Development Studio: Test deployment simplified” provides an excellent discussion on the use of deployment groups in an enterprise environment, along with the discussion on server profiles. It is available from:

`http://www.ibm.com/developerworks/data/library/techarticle/dm-1010testdeployment/index.html`

3.2.5 pureQuery support

Included in Optim Development Studio V2.2.1 is support for pureQuery, which provides a high-performance data access platform that makes it easier to develop, optimize, secure, and manage data access. Several articles and tutorials about Optim Development Studio and pureQuery are available in developerWorks and in IBM support sites.

With Optim Development Studio V2.2.1, you can:

- Generate Java beans to represent table columns.
- Develop SQL applications using annotated methods and inline methods. For annotated methods, you can define your own methods in custom interfaces. With inline methods, SQL statements are passed as parameters in method invocations, similar to JDBC and SQLJ.
- Examine and locate a SQL statement or references to a relational object in the Java source.
- Run SQL against databases, in-memory collections, and iterator objects by using a single API.
- Map SQL data to Java types with little developer intervention, or supply customized code for complex mappings.
- Develop applications that follow the Data Access Object (DAO) pattern by creating data access object using annotated methods.
- Capture dynamic SQL in a JDBC application, package these statements, and later execute them statically at run time.
- And so on.

Several developerWorks articles and tutorials have been written about pureQuery, its benefits, and how to use Optim Development Studio for developing Java applications that use pureQuery. Among them are:

- “Increase productivity in Java database development with new IBM pureQuery tools, Part 1: Overview of pureQuery tools”

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0709surange/index.html

- “Increase productivity in Java database development with new IBM pureQuery tools, Part 2: Detect and fix SQL problems inside Java program”

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0709surange2/index.html

- “Increase productivity in Java database development with new IBM pureQuery tools, Part 3: pureQuery rapid application development”

  http://www.ibm.com/developerworks/data/tutorials/dm0711surange/

- “Increase productivity in Java database development with new IBM pureQuery tools, Part 4: Tour Data Studio and pureQuery for Informix databases”

  http://www.ibm.com/developerworks/data/tutorials/dm0802surange/index.html

- “What's new and cool in Optim Development Studio, Part 2: Exploring Optim Development Studio and pureQuery Runtime Version 2.2 Fix Pack 3”

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1006optimdeveloper2/index.html

4

Debugging stored procedures with Data Studio

The Unified Debugger available in Data Studio V2.2.1 and Optim Development Studio V2.2.1 gives you the ability to debug native SQL, external SQL, and Java stored procedures using an easy-to-use graphical interface.

In this chapter we discuss the product functions and how to set up and use the Unified Debugger in the following sections:

- The Unified Debugger
- Setting up the Session Manager

4.1 The Unified Debugger

The Unified Debugger is a single debugger that can be used to debug SQL and Java stored procedures on DB2 for z/OS, DB2 on iSeries, and DB2 on Linux, UNIX, and Windows. It is included in the IBM Data Studio and Optim Development Studio products.

Users investing in DB2 application development have a need for formal debugging facilities. This need is even intensified when working on DB2 stored procedure development because the code runs in isolation at a DB2 server.

For classic languages, such as COBOL and C, the compiler products and their associated runtime facilities provide debug capabilities. Conversely, we have interpreted languages such as SQL and Java, where debugging sometimes has caused problems. The Unified Debugger focuses on these newer languages.

With the Unified Debugger, you can observe the execution of SQL procedure code or Java code, set breakpoints for lines, and view or modify variable values. The Unified Debugger supports external SQL, native SQL, and Java stored procedures.

The Unified Debugger can debug nested SQL or Java stored procedures sharing the same client application call stack. This means that users debugging a Java routine can step into and debug a called SQL procedure.

There are three basic elements to the Unified Debugger:

- The Unified Debugger server library: APIs that provide the interface for the DB2 servers and the supported routine objects.
- The Session Manager and the Unified Debugger router stored procedures that implement the client interface to the DB2 servers. This layer is independent of any particular platform or DB2 server type. The Unified Debugger in Data Studio v2.2.1 allows the user to bypass the use of a Session Manager by using the built-in Session Manager.
- A debug client that provides the debugger UI to the user: The debug client is only available on LUW clients.

4.1.1 Processing overview of the Unified Debugger

The IBM Data Studio or Optim Development Studio can be used to debug both native and external SQL stored procedures and Java stored procedures against DB2 10 for z/OS.

The chart in Figure 4-1 describes the processing flow for debugging stored procedures using the Data Studio Unified Debugger.

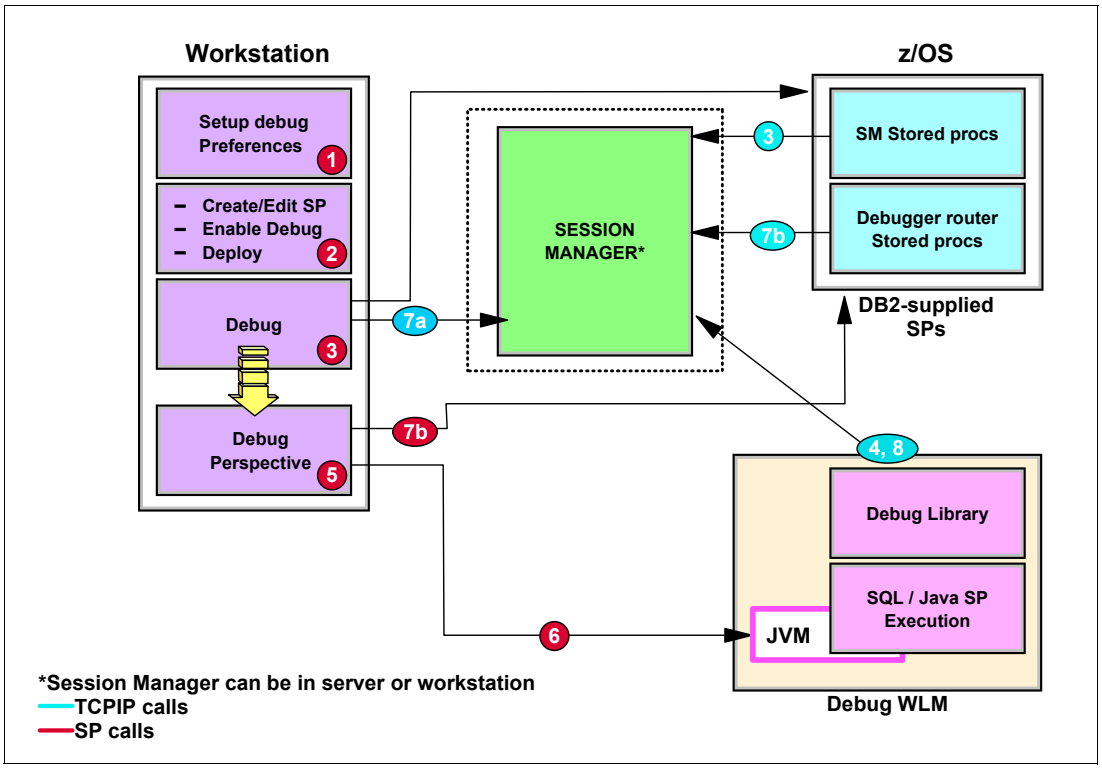

Figure 4-1 Processing overview - Unified Debugger with DB2 9 for z/OS

A quick overview of the process shown in Figure 4-1 follows:

1. Decide whether to use the built-in Session Manager or an external Session Manager (SM). If you want to use an external Session Manager, decide whether you want DS to use the SM that is connected to a server or use a SM that is currently active. You can start the SM either on the workstation, such as the client's workstation, or the z/OS server. Inform DS of this decision in **Window** → **Preferences** → **Run/Debug** → **DB2 Stored Procedure Debugger**. Select **Run the session manager on each connected server** or **Use already running session manager**. In the latter selection, the user must start the SM and capture the IP address or port of the SM.
2. Create, edit, and deploy with debug enabled an SQL or the Java stored procedure.
3. Debug rather than run the stored procedure. Do this by right-clicking the stored procedure and then selecting **Debug**.
4. The DB2 server hosting the stored procedure engages the SM via the debug server, establishing the debug session.
5. Data Studio (DS) switches the UI to the Debug Perspective, signalling that debugging is underway.
6. For Java stored procedures, DS drives the debug session by directly communicating with the JVM at the server.

7. For SQL procedure debugging, the SM is the central coordinating agent.
   a. DS drives the debug session with direct TCP/IP communication to the SM.
   b. DS drives the debug session via indirect communication to the SM. DS calls the debug router procedures (red 7b), which in turn communicates with the SM (blue 7b) via TCP/IP.
8. SQL procedures keep the debug server informed of the program status. The debug server coordinates control from DS via TCP/IP communication with the SM.

4.1.2 Setting up the Unified Debugger components

There are a couple of steps that you need to go through to be able to use the Unified Debugger functionality.

Installing the IBM Data Studio or Optim Development Studio Client

To install the IBM Data Studio or Optim Development Studio, execute the `LAUNCHPAD.EXE` file that comes with the product (see 1.4.2, “Client setup” on page 7). Unlike certain other products, there is hardly anything that you need to decide during your installation. See 1.2, “Understanding the Data Studio packaging” on page 2, for details on obtaining a copy of the no-charge IBM Data Studio or the charged product, Optim Development Studio.

Running install job DSNTIJRT

The DB2 10 for z/OS installation job DSNTIJRT creates the DB2 server objects required when using the Unified Debugger. Before DB2 10, the required DB2 server objects for the Unified Debugger are created using the post-installation job, DSNTIJSD.

The Unified Debugger uses the following DB2-supplied stored procedures:

- DB2DEBUG.DEBUGGERLEVEL
- DB2DEBUG.CREATE_SESSION
- DB2DEBUG.DESTROY_SESSION
- DB2DEBUG.QUERY_SESSION
- DB2DEBUG.LIST_SESSION
- DB2DEBUG.PUT_COMMAND
- DB2DEBUG.GET_REPORT
- SYSPROC.DBG_INITIALIZECLIENT
- SYSPROC.DBG_TERMINATECLIENT
- SYSPROC.DBG_SENDCLIENTREQUESTS
- SYSPROC.DBG_SENDCLIENTCOMMANDS
- SYSPROC.DBG_RECVCLIENTREPORTS
- SYSPROC.DBG_ENDSESSIONMANAGER
- SYSPROC.DBG_PINGSESSIONMANAGER
- SYSPROC.DBG_LOOKUPSESSIONMANAGER
- SYSPROC.DBG_RUNSESSIONMANAGER

DSNTIJRT also configures the WLM environment for the above stored procedures.

Note: Stored procedure DBG_RUNSESSIONMANAGER is only available for DB2 9 and 10 on z/OS and must run as an authorized program in an authorized environment.

Optional setup for the stored procedure WLM environment

A WLM procedure must be defined for executing the stored procedure. For SQL stored procedures, this WLM procedure optionally includes a DD card for collecting information during the debug session.

- For SQL stored procedures, a //PSMDEBUG statement can be added in the WLM procedure. The //PSMDEBUG statement defines a physical sequential data set with RECFM=VBA, LRECL=4096. This data set should only be included in the WLM procedure when requested by IBM Level 2, as the //PSMDEBUG statement presence causes records to be written to it for SQL Debugger problems that will impact performance.
- For Java stored procedures, a //JSPDEBUG statement can be added in the WLM procedure. The data set definition and usage of the file referenced in the //JSPDEBUG statement is the same as that used in the //PSMDEBUG statement.

4.2 Setting up the Session Manager

You can have the Session Manager run on any platform that you prefer. The SM is the component that handles the debug session between the debug client and the debug server. We recommend using the built-in Session Manager of Data Studio. There is no setup required for using the built-in Session Manager. In this section we discuss:

- Session Manager on the z/OS server
- Session Manager on a client
- Creating SQL stored procedures for debugging
- Debugging SQL stored procedures
- Using the Unified Debugger

4.2.1 Session Manager on the z/OS server

This section documents how to set up the SM on the z/OS server.

Defining the started task to RACF

The RACF definitions shown in Example 4-1 must be added to your z/OS security system.

Example 4-1 Define DB2UDSMD to RACF

```
//UDBG1    JOB ('RACF'),CLASS=A,MSGCLASS=A,MSGLEVEL=1,
//         USER********, PASSWORD*******
//*-------------------------------------------------------------------
//* Define the Unified Debugger Session Manager Started Task to RACF.
//* A security manager ID must be used to perform the definitions.
//*
//* The STARTED task is defined by a RACF profile named DB2UDSMD.**
//* USRT005 will be the ID associated with this Started Task.
//* Since the task will run a java program from OMVS, also assign an
//* OMVS segment definition to the user (UID, home dir, etc.)
//* Finally, activate the STARTED task definition in current memory.
//*-------------------------------------------------------------------
//RACFDEF  EXEC TSOBATCH
//SYSTSIN DD *
  ALTUSER USRT005 OMVS( UID(5) HOME('/u/usrt005') PROGRAM('/bin/sh') )
  RDEFINE STARTED DB2UDSMD.** STDATA(USER(USRT005))
  SETROPTS RACLIST(STARTED) REFRESH
END
```

The STARTED task is defined by a RACF profile named DB2UDSMD.**.

Important: It is mandatory to use DB2UDSMD as the started task name. The Session Manager is not tied to a specific DB2 subsystem, nor is it tied to any DB2 subsystem.

USRT005 was designated to be the ID associated with this started task. Because the task will run a Java program from OMVS, also assign an OMVS segment definition to the user (that is, UID, home dir, and so on). Finally, activate the STARTED task definition in current memory.

Creating the environment settings

The job shown in Example 4-2 is used to create a file in the HFS to hold the environment settings used by the Session Manager's started task.

Example 4-2 Job to create a file in HFS to hold the environment settings

```
//UDBG2    JOB 'USER=$$USER','<USERNAME:JOBNAME>',CLASS=A,
//         MSGCLASS=A,MSGLEVEL=(1,1), REGION=4096K,
//         USER=USRT005,PASSWORD=*******
//*--------------------------------------------------------------------
//* Create a file in the HFS to hold the Environment settings used when
//* the Unified Debugger Session Manager runs as a Started Task on z/OS
//*
//* USRT005 is the ID associated with the Started Task.
//* Place the file in that users home directory.
//* Name the file DB2UDSMDenvironment
//*--------------------------------------------------------------------
//*--------------------------------------------------------------------
//* Create a file in the HFS from inline data using COPY
//*--------------------------------------------------------------------
//OCOPY    EXEC PGM=IKJEFT01,DYNAMNBR=30
//SYSTSPRT DD SYSOUT=*
//HFSOUT   DD  PATH='/u/usrt005/DB2UDSMDenvironment',
//             PATHOPTS=(OWRONLY,OCREAT,OAPPEND,OTRUNC),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
//INLINE   DD *
#--------------------------------------------------------------------
# Environment settings for running the Unified Debugger Session Manager
# * _BPX_BATCH_SPAWN=NO
# * _BPX_SHAREAS=YES
#     Arrange for the JVM to run in the same address space. This avoids
#     launching 2 additional address spaces for the Started Task.
# * ENV=
#     Reference this file. Insulates from PATH and CLASSPATH changes
#     present in etc/profile.
# * PATH=
#     The location of the desired JAVA release, and system binaries.
# * CLASSPATH=
#     The location of the UDBG Session Manager jar file
# * JAVA_COMPILER=NONE
#     Disable the JIT. The Started Task runs the Session Manager only
#     one time, so disabling this saves space that will not be used.
#--------------------------------------------------------------------
_BPX_BATCH_SPAWN=NO
_BPX_SHAREAS=YES
ENV=/u/usrt005/DB2UDSMDenvironment
PATH=/usr/lpp/java150/J5.0/bin:/bin
CLASSPATH=/usr/lpp/db2/db2910_base/classes/db2dbgm.jar
JAVA_COMPILER=NONE
```

```
//SYSTSIN DD *
OCOPY INDD(INLINE) OUTDD(HFSOUT) TEXT
//
```

Create a file in the HFS to hold the environment settings used when the Unified Debugger Session Manager runs as a started task on z/OS.

An ID must be designated to be associated with the started task. Suppose that USRT005 is that ID. Place a file in that user's home directory to serve as an execution environment profile. The file must point to the location of the Session Manager `jar` file, `db2dbgm.jar`. Name the file something distinctive, such as `DB2UDSMDprofile`.

Note: Ensure that the CLASSPATH points to where the `db2dbgm.jar` file is installed. Otherwise, the started task, DB2UDSMD, does not come up.

In both Example 4-1 on page 119 and Example 4-2 on page 120 we used USRT005 as the place holder for the ID associated with started task DB2UDSMD, which you must change to your specific situation.

The use of a named HFS application profile is suggested for simple setup and segregation of duties. At a minimum, it needs to define the CLASSPATH to the Session Manager Java program. Other settings to tune the Java execution environment can be included. Note that BPXBATCH reads the `STDENV` file, so no shell script symbol substitution can be utilized here. Symbol substitution processing is only available to the user profile (.profile) for the started task user ID and scripts executed from the shell command line.

The Session Manager is independent of DB2, so it can run anywhere in the network. But the server platform (that is, the operating system that the stored procedure that you want to debug runs) is often a better default choice than running at the client workstation. The session Manager JAR file is now distributed on all server platforms, so it does not have to be obtained, downloaded, sent, pulled, pushed, or transported by you.

Creating a started task JCL

Create the started task JCL for DB2UDSMD and place it in the system proclib (Example 4-3). This is used to launch the Unified Debugger Session Manager on z/OS. USRT005 is the ID associated with this started task, as defined in the RACF STARTED class profile DB2UDSMD.**.

Example 4-3 Sample started task JCL for the Session Manager on z/OS

```
//UDBG3    JOB 'USER=$$USER','<USERNAME:JOBNAME>',CLASS=A,
//         MSGCLASS=A,MSGLEVEL=(1,1), REGION=4096K,
//         USER=********,PASSWORD=*******
//*--------------------------------------------------------------------
//* Create the Started Task JCL for DB2UDSMD. A START command will then
//* be able to launch the Unified Debugger Session Manager on z/OS.
//* USRT005 is the ID associated with the Started Task, as defined in
//* the RACF STARTED class profile DB2UDSMD.**
//*--------------------------------------------------------------------
//*--------------------------------------------------------------------
//* Use IEBUPDTE to write a JCL member into SYS1.PROCLIB
//*--------------------------------------------------------------------
//WRITEJCL EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

```
//SYSUT2   DD DISP=SHR,DSN=SYS1.PROCLIB
//SYSIN    DD DATA
./  ADD NAME=DB2UDSMD
//DB2UDSMD PROC PORT=4553,TIMEOUT=60
//*
//*    DB2 Unified Debugger Session Manager DAEMON FOR OPENEDITION
//*
//*    This JCL assumes no .profile exists for the user.
//*
//* Environment settings (PATH, CLASSPATH) come from STDENV file.
//*
//DB2UDSMD EXEC PGM=BPXBATCH,DYNAMNBR=128,REGION=0M,TIME=1440,
//             PARM='SH date;java com.ibm.db2.psmd.mgr.Daemon -timeout
//             &TIMEOUT -port &PORT -log /dev/null;date'
//STDOUT   DD  PATH='/tmp/DB2UDSMD.stdout',
//             PATHOPTS=(OWRONLY,OCREAT,OAPPEND,OTRUNC),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
//STDERR   DD  PATH='/tmp/DB2UDSMD.stdout',
//             PATHOPTS=(OWRONLY,OCREAT,OAPPEND,OTRUNC),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIROTH)
//STDENV   DD  PATH='/u/usrt005/DB2UDSMDenvironment',
//             PATHOPTS=ORDONLY
//             PATHOPTS=ORDONLY
./  ENDUP
/*
```

Note: BPXBATCH does not derive PATH from STDENV. For the Session Manager, PATH only needs to point to the Java run time. One approach is to specify the PATH directly on the command line (Example 4-2 on page 120). Another method requires the use of a shell script profile (.profile) for the started task user, which we have not included in this documentation. Note that the job step options in the JCL shown, which include the OMVS shell command, were carefully arranged to efficiently utilize the limited space available. The space is limited to 100 characters in total. As shown in Example 4-3 on page 121, there are still about 18 characters left to adjust the path specification for Java.

Considerations when coding the DB2UDSMD started task

Consider the following information:

- The PARM field

  When running the above task, you may get the following error:

  ```
  IEF642I EXCESSIVE PARAMETER LENGTH IN THE PARM FIELD
  ```

  This is because the PARM filed is limited to 100 characters only. To fix this, you can do either of the following options:

  – If the PARM field does not contain comma-delimited parameters and the PARM field contains less than 100 characters, enter PARM field data through position 71 and then continue exactly in position 16 of the next line and enclose the entire PARM field with apostrophes (Example 4-4).

  Example 4-4 Coding long PARM field into next line

  ```
  12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
  //STEP3___EXEC PGM=IEFBR14,PARM='LONG PARAMETER FIELD WITHOUT COMMA DEL
  //_____________IMITERS - CONTINUED IN COLUMN 16 OF THE NEXT LINE'
  ```

– If the PARM field does not contain comma-delimited parameters, and the PARM field is greater than 100 characters, use an STDPARM DD card. This workaround is available with APAR OA11699. See:

  http://www.ibm.com/support/docview.wss?uid=isg1OA11699

  The STDPARM file can accept 65.356 bytes.

  So, for example, create the file ADMF001.DB2UDSMD.PARMOMVS, which contains the following data:

```
SH date;/ZOS17TC/usr/lpp/java/j1.4_64/bin/java com.ibm.db2/psmd.mgr.Daemon -timeout
&TIMEOUT -port &PORT;date
```

  Code the DB2UDSMD procedure as shown in Example 4-5.

Example 4-5 DB2UDSMD procedure with STDPARM

```
//DB2UDSMD  EXEC PGM=BPXBATCH,DYNAMNBR=128,REGION=0M,TIME=1440
//STDPARM  DD DISP=SHR,DSN=ADMF001.DB2UDSMD.PARMOMVS
```

- STDERR and STDOUT files

  The HFS files pointed to by STDERR and STDOUT need to have the property access authorizations. Otherwise, the DB2UDSMD task will not start, and the user might see messages similar to those in Example 4-6 in the console.

Example 4-6 Authorization error

```
IEF403I DB2UDSMD - STARTED - TIME=15.52.36
ICH408I USER(DB2UDSMD) GROUP($STCGRP ) NAME(DB2UDSMD
  /app/db2/tmp/DB2UDSMD.stdout
  CL(DIRSRCH ) FID(01C1F7D7D9F0F4009621000000000003)
  INSUFFICIENT AUTHORITY TO STAT
  ACCESS INTENT(--X)  ACCESS ALLOWED(OTHER      ---)

IEF403I DB2UDSMD - STARTED - TIME=16.05.08
ICH408I USER(DB2UDSMD) GROUP($STCGRP ) NAME(DB2UDSMD
  /app/db2/tmp CL(DIRSRCH ) FID(01C1F7D7D9F0F4009621000000000
  INSUFFICIENT AUTHORITY TO CHDIR
  ACCESS INTENT(--X)  ACCESS ALLOWED(OTHER      ---)
  EFFECTIVE UID(0000000005)  EFFECTIVE GID(0000000100)

 IEF403I DB2UDSMD - STARTED - TIME=16.29.18
 ICH408I USER(DB2UDSMD) GROUP($STCGRP ) NAME(DB2UDSMD
   /app/db2/tmp/DB2UDSMD.stdout
   CL(DIRACC  ) FID(00000001000000010000000000000000)
   INSUFFICIENT AUTHORITY TO OPEN
   ACCESS INTENT(-W-)  ACCESS ALLOWED(OTHER      R-X)
```

To fix this, change the permission bits for the /app/db2 directory from 700 to 755. DB2UDSMD does not have the authority to change the directory at the /app/db2 directory. Every directory in the path has to have the permissions in order to change directory.

To change to /app/db2/tmp, the user ID has to have directory permission to the:

– /app directory
– /app/db2 directory
– /app/db2/tmp directory

Starting the Session Manager started task on z/OS

After you have executed all the JCL, you can now test whether the Session Manager is working:

1. From the z/OS console, issue a **START DB2UDSMD,TIMEOUT=1 operator** command to start your Session Manager.
2. Wait a few minutes. The Session Manager will time out after 1 minute of inactivity.
3. Check the contents of the HFS file in STDOUT, /tmp/DB2UDSMD.stdout. Figure 4-2 shows an example of a good output from the Session Manager.

```
Mon Mar 2 12:48:32 PST 2009
args[0]: -timeout
args[1]: 1
args[2]: -port
args[3]: 4553
args[4]: -log
args[5]: /dev/null
Code Level: 070418
Debug Session Manager started on IP: 9.30.88.135 - port: 4553
idleTimeOut: 1
Mon Mar 2 12:49:37 PST 2009
```

Figure 4-2 Example of a successful output from the Session Manager

Granting DEBUGSESSION privilege

Grant the DEBUGSESSION privilege to the user that runs the debug client. The DEBUGSESSION privilege is a new system authorization. Refer to the new catalog column DEBUGSESSIONAUTH in table SYSIBM.SYSUSERAUTH to obtain information about who has already been granted this privilege.

Tip: For more information about the Unified Debugger and DB2 for z/OS, read the following articles in developerWorks:

- Debugging stored procedures in DB2 z/OS with Optim Development Studio, Part 1: Use the Unified Debugger in a sample scenario

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0811zhang/index.html

- Debugging stored procedures on DB2 z/OS with Data Studio Developer, Part 2: Configure the stored procedure debug session manager on z/OS

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0903debugdatastudio/

Preparing your stored procedures for debugging

After you have successfully set up the environment for debugging your SQL stored procedures, you must now decide for every single stored procedure whether you want to debug it.

For a native SQL procedure, define the procedure with the ALLOW DEBUG MODE option and the WLM ENVIRONMENT FOR DEBUG MODE option.

For an external SQL procedure, use DSNTPSMP or the IBM Data Studio to build the SQL procedure with the BUILD_DEBUG option.

For a Java procedure, define the procedure with the ALLOW DEBUG MODE option, select an appropriate WLM environment for Java debugging, and compile the Java code with the -G option.

4.2.2 Session Manager on a client

To set up the Session Manager on a workstation where the IBM Data Studio is installed:

1. Open a command window and go to the directory where the IBM Data Studio is installed, for example, `C:\Program Files\IBM\SDP70\dwb\bin>`. From this directory, run `db2dbgm.bat`. Note the IP address and port of the Session Manager (Figure 4-3).

```
C:\Program Files\IBM\SDP70\dwb\bin>db2dbgm.bat
args[0]: -port
args[1]: 4554
args[2]: -timeout
args[3]: 50
Code Level: 070418
Debug Session Manager started on IP: 9.30.28.113 - port: 4554
idleTimeOut: 50
```

Figure 4-3 Debug Session Manager startup

2. Launch the IBM Data Studio. Click **Window** → **Preferences** → **Run / Debug** → **DB2 Stored Procedure Debugger**. On this page, click **Use already running session manager**. Fill in the host IP address and port number of the session manager (Figure 4-4 on page 126).
3. Click **Apply** → **OK**.

Setting up the Session Manager on the server

In 4.1, “The Unified Debugger” on page 116, we gave the steps for setting up the Session Manager on the z/OS server. If your client has a firewall, then it might not be feasible for the server to initiate communication with the Session Manager in the client. You might then want to use the Session Manager in the z/OS server.

To use the Session Manager, in the IBM Data Studio preferences shown in Figure 4-4, click **Run the session manager on each connected server**.

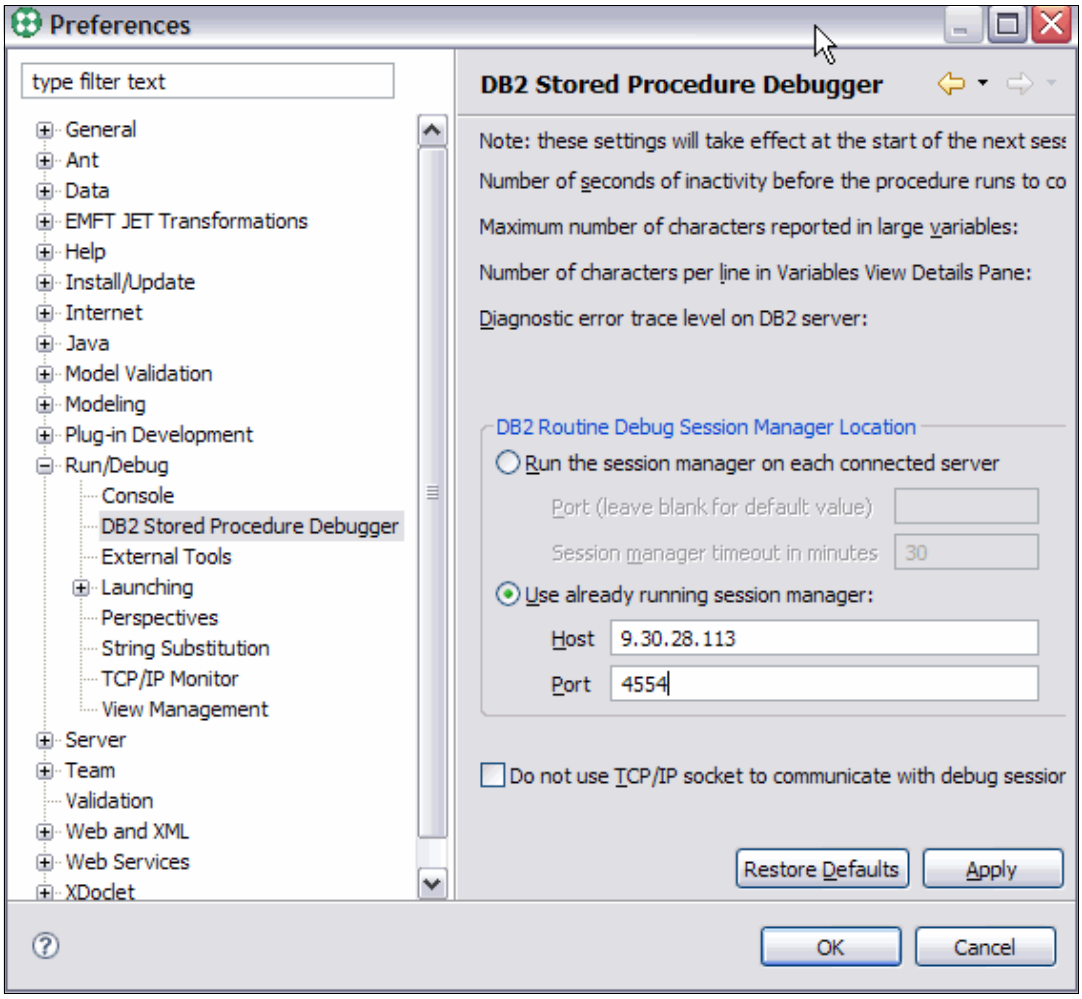

Figure 4-4 Preferences for using the client Session Manager

4.2.3 Creating SQL stored procedures for debugging

IBM Data Studio can be used to create SQL stored procedures, though it is not required. However, it must be used to build the stored procedure for debug. First, we describe the general steps to create an SQL stored procedure for debugging, followed by our EMPDTLSS case study:

1. Start IBM Data Studio from **Start** → **All Programs** → **IBM Software Development Platform** → **IBM Data Studio** → **IBM Data Studio**.
2. Create a new connection or reconnect to an existing DB2 9 for z/OS server.
3. Create a new project or open an existing Data Development Project.
4. Optional: Update the Unified Debugger time-out value.

   The default time-out value is 300 seconds. This means that while in debug mode, after 300 seconds of inactivity, the Debugger terminates and any locks held by the stored procedure being debugged are released. The SQL stored procedure will run to completion.

This value can be changed from the Preferences pages (Figure 4-4 on page 126). On the Preference page, this is the number of seconds of inactivity before the procedure runs to completion. We use the default value for our case study. Click **OK** to accept the default value.

5. Create a new stored procedure. During the creation of the SQL stored procedure using the New Stored Procedure wizard, check the **Enable debugging** check box on the Deploy Options page. See 2.2.1, “Creating a new stored procedure using templates” on page 61, for detailed steps on how to create a new SQL stored procedure.

The IBM Data Studio Data Output window includes a message indicating whether the stored procedure performed a build for debug. For an external SQL stored procedure, the build utility reports a BUILD_DEBUG function and whether it was successful (Example 4-7).

Example 4-7 BUILD_DEBUG function was completed successfully

```
Build utility function requested: BUILD_DEBUG
DSNT540I  DB9AWLMR WAS REFRESHED BY PAOLOR5  USING AUTHORITY FROM SQL ID PAOLOR5 :
0
PAOLOR5.EXTSQL_1101 - Deploy for debug successful.
```

For a native stored procedure, the output displays the beginning message `Deploy for debug started` and ends with the message `Deploy for debug successful`.

4.2.4 Debugging SQL stored procedures

Once the SQL stored procedure is successfully built for debug mode, we can debug the stored procedure. In 4.2.2, “Session Manager on a client” on page 125, we discussed how to launch the Session Manager. It is now ready to handle the communication between IBM Data Studio and the Unified Debugger code on the server. You can start debugging from three launch points:

- From Data Project Explorer, select the SQL stored procedure and right-click **Debug**.
- With the SQL stored procedure opened in the Routine Editor (which can be done by selecting the SQL stored procedure in Data Project Explorer and righting-click **Open**), click the **Source** tab. Right-click whitespace → **Debug As** → **Debug**.
- From the **Routine Editor** → **Configuration** tab, click **Debug** (Figure 4-5).

Figure 4-5 Starting the Debugger from the Routine Editor

When the stored procedure is launched in debug mode, the user is prompted to switch into the Debug Perspective. Click **Yes**. In the Debug Perspective, you can set breakpoints in the prefix area to the left of a valid statement, monitor and change the values of variables, and interactively debug.

Note: When connected to a DB2 9 server, IBM Data Studio assumes that if the imported SQL stored procedure contains a WLM environment, then the stored procedure is an EXTERNAL type, rather than native. You need to manually edit the CREATE PROCEDURE DDL before importing, and add the FENCED keyword to correctly identify this as an External SQL procedure.

- Summary: The above settings are summarized. Click **Finish** to build the stored procedure for debug.

4.2.5 Using the Unified Debugger

In this section we go through the features and tasks for using the Unified Debugger. See the following website for an IBM developerWorks article that talks about the Unified Debugger and other problem determination tips:

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0706scanlon/

When a debug session has been established, IBM Data Studio initiates a switch from the Data Perspective to the Debug Perspective.

The Debug Perspective

The Unified Debugger launches the Eclipse Debug Perspective (Figure 4-6) when a stored procedure is being debugged. This perspective is the same graphical interface used when debugging an SQL stored procedure, a Java stored procedure, a Java application, or any other resource in Eclipse.

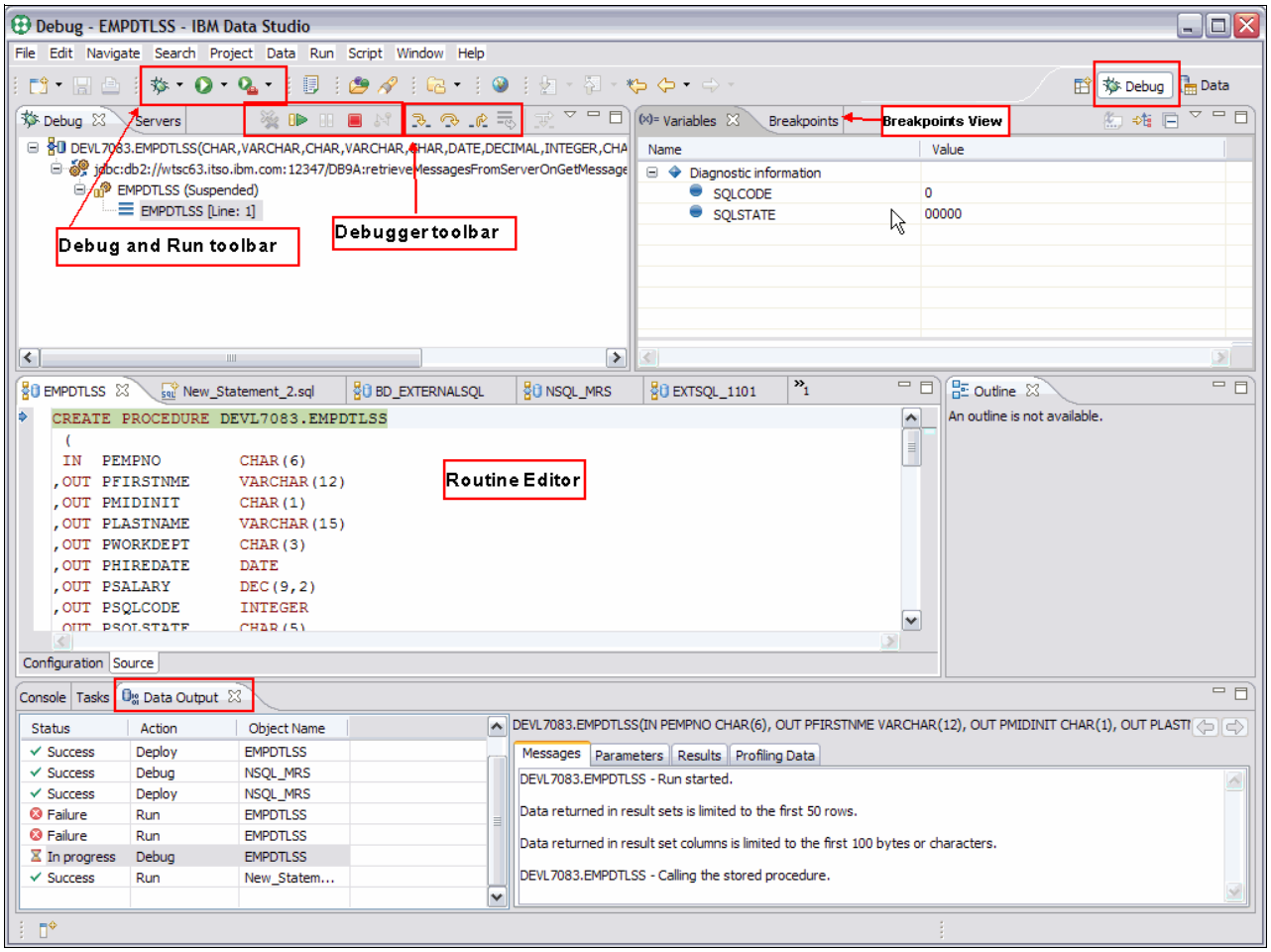

Figure 4-6 The Debug Perspective

The Debug Perspective is made up of the following related views and tool bars:

- Routine Editor view: Shows the SQL code.
- Breakpoints view: Shows the list of break points currently set.
- Variables view: Shows the list of defined variables.
- Outline view: Shows the variables and methods of the stored procedure under execution.
- Data Output: Shows the status history, execution messages, parameters, and result sets, if any.

- Execution toolbar: Provides icons to debug or simply execute a stored procedure. Also keeps a list of the most recently executed stored procedures.
- Debugger toolbar: Provides icons for the various debug execution step commands:
  - Step into
  - Step over
  - Step return
  - Resume
  - Pause
  - Terminate

These views are connected in the sense that the break points and variables views show the debug data for the stored procedure currently shown in the Routine Editor view. Switching to a different procedure in the Routine Editor causes the break points and variables views to display the debug data for the newly selected stored procedure code. The Debug Perspective also includes a specialized set of toolbars for debugging.

- Routine Editor view

  The routine editor displays the stored procedures being debugged. Each tab in the Editor view displays an open resource. You can set breakpoints in this view, either during debug or before initiating the debug in the Data Perspective.

- Execution toolbar

  The Execution toolbar includes three actions (Figure 4-7):

  - Debug: Executes a stored procedure in Debug mode. The pull-down list next to the icon shows all previously debugged stored procedures. The default is to debug the last executed stored procedure.
  - Run: Executes a stored procedure. The pull-down list next to the icon shows all previously executed stored procedures. The default is to execute the last executed stored procedure.
  - External tools: Executes an Ant script or another tool. IBM Data Studio does not use this action.

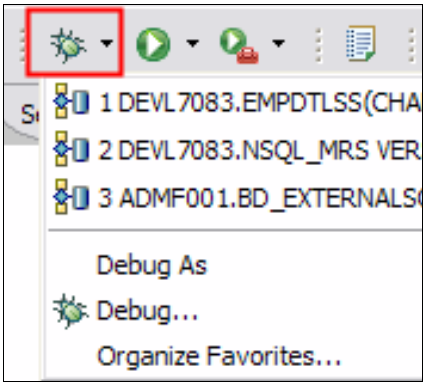

Figure 4-7 Debug and Run toolbar

- Debugger toolbar

  The Debugger toolbar includes the debug step commands illustrated in Table 4-1.

Table 4-1 Execution toolbar

Icon	Command description
	Step into the next line or block of SQL code. If the current statement is a stored procedure call, then the next line is the first line of the called stored procedure.
	Step over to the next line of execution. If the current line is a call to a nested stored procedure or the next line is an indented block of code, then the nested procedure or block of code will be executed as one statement unless a break point was encountered.
	Step return causes execution to resume at the next line in the parent stored procedure of the current nested stored procedure unless a break point is encountered. If the current stored procedure is the only stored procedure in the call stack, then execution will run to completion or the next break point encountered.
	Resume causes the stored procedure being debugged to run and stop or break at the next breakpoint.
	Pause causes the execution of the stored procedure to be suspended. Click **Resume** or **Terminate** to continue or terminate execution.
	Terminate causes the execution of the stored procedure to stop. This does not cause the stored procedure to run to completion. This simulates an abort.

- Variables view

  The Debug Perspective's Variables view displays the current values of the defined variables in the stored procedures. When debugging a Java stored procedure, the Variables view also lists inherited variables.

- Outline view

  The Outline view in the Debug Perspective is the same as in the Data Perspective. It shows on a higher level where in the stored procedure code execution is stopped. When debugging a Java stored procedure, this view shows the method where the stored procedure is stopped.

- Breakpoints view

  The Debug Perspective's breakpoints view and its associated toolbar allow you to manage the breakpoints that you have set. When you set a breakpoint in the Routine Editor, an entry is added in this view, with the resource name and line number. A check box next to this entry indicates that the breakpoint is active**.** To disable this breakpoint, but not remove it, uncheck the entry (Figure 4-8).

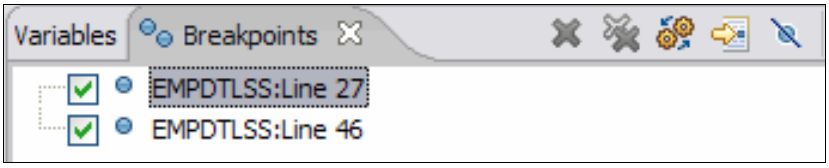

Figure 4-8 Breakpoints view

The Breakpoints view also has a specialized toolbar for managing the breakpoints (Table 4-2).

Table 4-2 Breakpoints view toolbar

Icon	Command description
	Remove a breakpoint. Click an entry or several entries, and click this icon to remove the breakpoint.
	Remove all breakpoints.
	Skip all breakpoints. This causes the execution to complete without stopping.
	Show all breakpoints.

Valid Unified Debugger breakpoint statements

The Unified Debugger highlights certain statements during a debug session. The highlighted statements are the only statements that you can step into or put breakpoints on. Certain SQL statements change variables, while other statements do nothing. Table 4-3 summarizes this.

Table 4-3 Valid SQL Debugger breakpoint and change variable statements

Category of statements	Statements
Statements that change variables	CALL FETCH <..> INTO GET DIAGNOSTICS SELECT <..> INTO SET

Category of statements	Statements

Statements that do not accept breakpoints and do not impact the Unified Debugger processing	DECLARE cursor WITH RETURN FOR <sql statement>
	DECLARE CONDITION (CONDITION) FOR SQLSTATE (VALUE) "..."
	DECLARE CONTINUE HANDLER
	DECLARE CURSOR
	DECLARE EXIT HANDLER
	DECLARE UNDO HANDLER (unless they are entered)
	DO
	ELSE
	END CASE
	END IF
	END FOR
	END REPEAT
	END WHILE
	LOOP
	REPEAT (as a keyword alone)
	THEN
	labels, e.g. P1: :

Statements that accept breakpoints (highlighted statements)	All SQL statements
	The following SQLPL statements:
	BEGIN
	BEGIN NOT ATOMIC
	BEGIN ATOMIC
	CLOSE CURSOR
	DECLARE var without default
	DECLARE RESULT_SET_LOCATOR [VARYING]
	DECLARE SQLSTATE
	DECLARE SQLCODE (unless there is a default)
	DO (inside a FOR)
	END
	FOR .. END
	FOR *select statement...*
	GOTO(LABEL)
	IF (EXPRESSION)
	ITERATE
	LEAVE
	OPEN CURSOR
	RESIGNAL
	SIGNAL
	RETURN(value)
	SELECT <..> INTO
	SET (EXPRESSION)
	UNTIL (EXPRESSION)
	WHEN (VALUE)
	WHILE (EXPRESSION)

Resources

The following 2-part articles found in developerWorks provide updated information about the Unified Debugger. These articles are updated by the Data Studio development team when needed.

- “Debugging stored procedures in DB2 z/OS with Optim Development Studio, Part 1: Use the Unified Debugger in a sample scenario”

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0811zhang/index.html?S_TACT=105AGX54&S_CMP=B1113&ca=dnw-945,

- “Debugging stored procedures on DB2 z/OS with Data Studio Developer, Part 2: Configure the stored procedure debug session manager on z/OS”

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0903debugdatastudio/?S_TACT=105AGX11&S_CMP=LP

A

Reference material

In this appendix we provide additional material referenced from the preceding chapters. This appendix contains the following:

- Data Studio and Optim Development Studio V2.2.1 support features
- Supported functions of the three Data Studio products
- Actions on database objects from the Administration Explorer

Data Studio and Optim Development Studio V2.2.1 support features

Table A-1 lists the supported data servers for Data Studio and Optim Development Studio.

Table A-1 Supported servers for Data Studio and Optim Development Studio

Database	Version	Comments
DB2 for Linux, UNIX, and Windows [a]	9.1	
	9.5	
	9.7	
	9.8	Version 2.2.1 toleration mode.
DB2 for iSeries	V5R4	
DB2 for z/OS[a]	8.1	Version 2.2.0.3 and later.
	9.1	Version 2.2.0.3 supports compatibility mode and new function mode.
	10.1	Version 2.2.1.
IBM Informix data server	10.0 [b]	
	11.1 [b]	
	11.5[b]	
	11.7	Version 2.2.1.
Oracle[a]	10g	
	10g R2	
	11g	

a. Supported for SQL management repository.
b. Supported for Optim pureQuery Runtime. Not supported for routine development and debugging.

Supported functions of the three Data Studio products

Table A-2 summarizes the suite of supported functions of the three Data Studio products introduced in 1.1, “Introduction” on page 2.

Table A-2 Data Studio and Optim Development Studio V2.2.1 support features

Product features	Data Studio RCP	Data Studio IDE	Optim Develop. Studio
Installation options			
Shell-sharing with other Optim and Rational products		X	X
Object and Data Management			
Create, alter, drop, and manage security for DB2 or IDS server objects.	X	X	X
Create, alter, drop, and manage security for Oracle objects.			X
Common DB support including DDL gen, analyze impact, compare and sync data, and property browser.	X	X	X
Generate DB2 commands and utilities for DB2 for Linux, UNIX, and Windows (LUW).	X	X	X
Edit, sample, load, and extract data.	X	X	X
Data distribution viewer.	X	X	X
Data transforms framework.			X
Optim model support: Generate an Optim OEF model and save that model in an OEF file.			X
ILOG® JView v8.7 enabled.			X
Database support			
Database overview diagrams.	X	X	X
Informix DB support (v10, v11.1, v11.5).		X	X
Physical database modeling - data design project.			X
Connection repository: Organize, store, and share database connection properties.			X
Database Administration perspective.	X	X	X
Database Administration Task assistants included for logging, configuration parameters, automatic maintenance, and so on.	X	X	X
pureScale™ support in database administration task assistants for DB2 on LUW.		X	X
Visually explore value distributions and relationships between the columns of a table.	X	X	X
Common logging. Launch integrated data tooling help.	X	X	X
DB2 database package utilities.			X
Integrated Query Editor (IQE) integrated with the Data Design project.			X

Product features	Data Studio RCP	Data Studio IDE	Optim Develop. Studio
Application development			
Integrated Query Editor (SQL and XQuery) with query formatting.	X	X	X
SQL Builder.	X	X	X
Advanced query formatting.			X
Extract and persist SQL results as a text file or an XML file.	X	X	X
Visual Explain for DB2 databases.	X	X	X
Visual Explain for Oracle database.			X
Single query tuning and statistics advisor.	X	X	X
SQL query environment capture for serviceability (z/OS only).	X	X	X
SQL routine editor and debugger.	X	X	X
Java routine editor and debugger.		X	X
PL/SQL routine editor against DB2 for LUW.		X	X
PL/SQL routine editor against Oracle.			X
SQLJ development.		X	X
XML editor, schema editor, and annotated XSD mapping editor.		X	X
Advanced XML editors and document generation.			X
Data web services development and deployment.		X	X
Deploy web services on DataPower® appliances and JMS.			X
Develop pureQuery applications in a Java environment and execute pureQuery code. Include tooling for Static SQL Binding for Java. Correlate SQL to Java source code. Impact analysis for Java and SQL Execution statistics.			X

Actions on database objects from the Administration Explorer

In 1.5.3, “Administration Explorer view” on page 35, we provide an overview of the capabilities and features of the Administration Explorer.

Table A-3 lists context menu actions on database objects from the Administration Explorer. These actions are displayed by right-clicking a specific database object in the object editor.

Table A-3 Context menu actions on database objects from the Administration Explorer

Database object type	Analyze impact	ALTER	DROP	COPY	Gener-ate DDL	Manage privil-eges	Add to over-view diagram	Unload load edit
Aliases	X	X	X	X	X		X	
Packages[a]	X		X	X		X		
Stored procedures	X		X	X	X	X		
User-defined functions	X		X	X	X	X		
User-defined types	X	X	X	X	X			
Auxiliary tables	X	X	X	X	X		X	
Constraints	X	X	X	X				
Databases	X	X	X	X	X	X		
Indexes	X	X	X	X	X			
MQTs	X	X	X	X	X	X	X	X[b]
Schemas	X	X	X	X	X	X	X	
Sequences	X	X	X	X	X	X		
Storage groups	X	X	X	X		X		
Synonyms	X	X	X	X	X	X	X	X[b]
Tables	X	X	X	X	X	X	X	X
Table spaces	X	X	X	X		X		
Triggers	X	X	X	X	X	X		
Users and roles	X	X	X	X	X	X		
Views	X	X	X	X	X	X	X	

a. The Object List Editor does not list packages yet for DB2 10 for z/OS. APAR 173902 documents this.
b. Load action not available

Abbreviations and acronyms

AIB	Application Interface Block
AIX®	Advanced Interactive eXecutive from IBM
APAR	Authorized program analysis report
ASCII	American National Standard Code for Information Interchange
BLOB	Binary large objects
CCA	Client configuration assistant
CCSID	Coded character set identifier
CLI	Call level interface
CLP	Command line processor
CPU	Central processing unit
DBAT	Database access thread
DBD	Database descriptor
DBID	Database identifier
DBMS	Data base management system
DBRM	Database request module
DC	Development Center
DCL	Data control language
DD	Distributed Debugger
DDCS	Distributed database connection services
DDF	Distributed data facility
DDL	Data definition language
DLL	Dynamic load library manipulation language
DML	Data manipulation language
DNS	Domain name server
DRDA®	Distributed relational database architecture
DSN	Data set name
DT	Debug Tool
DTT	Declared temporary tables
EBCDIC	Extended binary coded decimal interchange code
EDM	Environment descriptor management
FTP	File Transfer Program
GB	Gigabyte (1,073,741,824 bytes)
GBP	Group buffer pool
GRS	Global resource serialization
GUI	Graphical user interface
HPJ	High performance Java
I/O	Input/output
IBM	International Business Machines Corporation
IDE	Interactive development environment
IFCID	Instrumentation facility component identifier
IFI	Instrumentation facility interface
IPLA	IBM Program Licence Agreement
IRLM	Internal resource lock manager
ISPF	Interactive system productivity facility
ISV	Independent software vendor
IT	Information Technology
ITSO	International Technical Support Organization
IVP	Installation verification process
JCL	Job control language
JDBC	Java Database Connectivity
JDK	Java Development Kit
JDSD	Job Data Set Display
JFS	Journaled file systems
JNDI	Java Naming and Directory Interface
JRE	Java runtime environment
JVM	Java Virtual Machine
KB	Kilobyte (1,024 bytes)
LEL	Language Environment
LOB	Large object
LPA	Link pack area
LPAR	Logical partition
LPL	Logical page list
LRECL	Logical record length
LRSN	Log record sequence number
LUW	Logical unit of work
LVM	Logical volume manager
MB	Megabyte (1,048,576 bytes)
MFI	Main frame interface
MQT	Materialized query table
NPI	Non-partitioning index
ODB	Object descriptor in DBD

ODBA	Open Data Base Access
ODBC	Open Data Base Connectivity
OS/390®	Operating System/390®
PAV	Parallel access volume
PDS	Partitioned data set
PIB	Parallel index build
PSID	Pageset identifier
PSP	Preventive service planning
PTF	Program temporary fix
PUNC	Possibly uncommitted
QA	Quality Assurance
QMF™	Query Management Facility™
RACF	Resource Access Control Facility
RBA	Relative byte address
RECFM	Record format
RI	Referential integrity
RID	Record identifier
RR	Repeatable read
RRS	Resource Recovery Services
RRSAF	Resource Recovery Services attach facility
RS	Read stability
SC	Service class
SDK	Software developers kit
SDSF	System Display and Search Facility
SMIT	System Management Interface Tool
SPAS	Stored procedure address space
SPB	Stored Procedure Builder
SQL	Structured query language
SQL	Structured query language
SQLJ	Structured Query Language (SQL) that is embedded in the Java programming language
SU	Service unit
UCS	Unicode Conversion Services
UDF	User-defined function
UOW	Unit of work
USS	UNIX system services
WLM	Work load manager
WSAD	WebSphere Studio Application Developer
WSADIE	WebSphere Studio Application Developer Integration Edition
WSDL	WebSphere Definition Language
WSED	WebSphere Studio Enterprise Developer

Index

E

F

G

H

I

S

T

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *DB2 10 for z/OS Technical Overview*, SG24-7892
- *Extremely pureXML in DB2 10 for z/OS*, SG24-7915
- *DB2 9 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7604
- *DB2 9 for z/OS: Distributed Functions*, SG24-6952
- *IBM Data Studio V2.1: Getting Started with Web Services on DB2 for z/OS*, REDP-4510

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at the following website:

ibm.com/redbooks

Other publications

Additional information can also be found in the following publications:

- *DB2 10 for z/OS Application Programming and SQL Guide,* SC19-2969
- *DB2 10 for z/OS Application Programming Guide and Reference for Java,* SC19-2970
- *DB2 10 for z/OS Codes,* GC19-2971
- *DB2 10 for z/OS Command Reference,* SC19-2972
- *DB2 10 for z/OS Data Sharing: Planning and Administration,* SC19-2973
- *DB2 10 for z/OS Installation and Migration Guide,* GC19-2974
- *DB2 10 for z/OS Internationalization Guide,* SC19-2975
- *DB2 10 for z/OS Introduction to DB2 for z/OS,* SC19-2976
- *IRLM Messages and Codes for IMS and DB2 for z/OS,* GC19-2666
- *DB2 10 for z/OS Managing Performance,* SC19-2978
- *DB2 10 for z/OS Messages,* GC19-2979
- *DB2 10 for z/OS ODBC Guide and Reference,* SC19-2980
- *DB2 10 for z/OS pureXML Guide,* SC19-2981
- *DB2 10 for z/OS RACF Access Control Module Guide,* SC19-2982
- *DB2 10 for z/OS SQL Reference,* SC19-2983

- *DB2 10 for z/OS Utility Guide and Reference,* SC19-2984
- *DB2 10 for z/OS What's New?,* GC19-2985
- *DB2 10 for z/OS Diagnosis Guide and Reference,* LY37-3220

Online resources

These websites are also relevant as further information sources:

- Data Studio features and benefits

  http://www.ibm.com/software/data/optim/data-studio/features.html

- Data Studio support portal

  http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/IBM_Data_Studio

- *Support for Unicode: Using Conversion Services*, SC33-7050

  http://www.ibm.com/servers/s390/os390/bkserv/latest/v2r10unicode.html

- Data Studio stand-alone and IDE versions available from the IBM Support website

  http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/IBM_Data_Studio

- Data Studio also available as a downloadable feature from the DB2 for z/OS website

  http://www.ibm.com/software/data/db2/zos/downloads/

- Technote: “Information about which IBM Software products can be installed together so together so that they share a common environment”

  http://www.ibm.com/support/docview.wss?rs=2042&uid=swg21279139

- Section “Installing the IBM DB2 Driver for JDBC and SQLJ” on the DB2 for z/OS Information Center website

  http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc.inst/db2z_installjccintro.htm

- JDBC tracing options with Data Studio

  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0706scanlon/

- “Creating scripts more efficiently in the SQL and XQuery editor”

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1011sqlguidetour/index.html?cmp=dw&cpb=dwinf&ct=dwnew&cr=dwnen&ccy=zz&csr=111110

- “Create package variations for z/OS DB2 stored procedures” on developerWorks

  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0608parmeshwar/

- “Using common connections with Optim solutions”

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0812devlin/index.html

- “IBM Optim Development Studio: Test deployment simplified”

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1010testdeployment/index.html

- “IBM Optim Development Studio: Test deployment simplified”

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1010testdeployment/index.html

- Increase productivity in Java database development with new IBM pureQuery tools, Part 1: Overview of pureQuery tools

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0709surange/index.html

- Increase productivity in Java database development with new IBM pureQuery tools, Part 2: Detect and fix SQL problems inside Java program

  http://www.ibm.com/developerworks/data/library/techarticle/dm-0709surange2/index.html

- Increase productivity in Java database development with new IBM pureQuery tools, Part 3: pureQuery rapid application development

  http://www.ibm.com/developerworks/data/tutorials/dm0711surange/

- Increase productivity in Java database development with new IBM pureQuery tools, Part 4: Tour Data Studio and pureQuery for Informix databases

  http://www.ibm.com/developerworks/data/tutorials/dm0802surange/index.html

- What's new and cool in Optim Development Studio, Part 2: Exploring Optim Development Studio and pureQuery Runtime Version 2.2 Fix Pack 3

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1006optimdeveloper2/index.html

- Unified Debugger

  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0706scanlon/

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Data Studio and DB2 for z/OS Stored Procedures

Redpaper™

Understand IBM Data Studio V2.2.1 packaging

Use Data Studio with DB2 10 for z/OS stored procedures

Take advantage of the Unified Debugger

Stored procedures can provide major benefits in the areas of application performance, code re-use, security, and integrity. DB2 has offered ever-improving support for developing and operating stored procedures.

This IBM Redpaper publication is devoted to tools that can be used for accelerating the development and debugging process, in particular to the stored procedure support provided by the latest and fastest evolving IBM product: Data Studio.

We discuss topics related to handling stored procedures across different platforms. We concentrate on how to use tools for deployment of stored procedures on z/OS, but most considerations apply to the other members of the DB2 family.

This paper is a major update of Part 6, “Cool tools for an easier life,” of the IBM Redbooks publication *DB2 9 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7604.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

REDP-4717-00