Getting Started With

IBM Data Studio for DB2

for Linux, UNIX and Windows
Ideal for database developers and administrators

by:

Dimple Bhatia

Vinod Chirayath

Adam Faeth

Praveen Ghantasala

Hassi Norlén

Hardik Patel

Daniel Zilio

DB2 ON CAMPUS BOOK SERIES

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 grant 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.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

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.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

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

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

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

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

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

Table of contents

Preface

Keeping your skills current in today's world is becoming increasingly challenging. There are too many new technologies being developed, and little time to learn them all. The DB2 on Campus Book Series has been developed to minimize the time and effort required to learn many of these new technologies.

Who should read this book?

This book is intended for anyone who needs to learn the basics of database administration and development using Data Studio, the Eclipse-based tool provided at no charge. It replaces previous generation tools, such as Developer workbench and DB2 Control Center. The DB2 Control Center and other DB2 tools are deprecated in DB2 9.7, so it is important to become familiar with Data Studio and related products. The Version 3.1 release of IBM Data Studio incorporates the advanced features previously available only in Optim Database Administrator and Optim Development Studio, making it much more powerful for database development and administration.

> **Note:**
>
> This book assumes you have a basic knowledge of DB2. For more information about DB2, see *Getting Started with DB2 Express-C* or the DB2 Information Center here:
>
> http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp

A note about the second edition

Big changes have happened with IBM Data Studio since the first edition of this book was written. Capabilities that were previously only available in priced versions of the database tools have been consolidated into IBM Data Studio at no charge. This includes enhanced database management capabilities, such as advanced change management, as well as advanced development capabilities such as pureQuery. (Since pureQuery is the subject of another *Getting Started* book, we did not cover that in this edition.) In addition, the web console, previously known as the Data Studio Health Monitor has been extended to provide the ability to schedule and manage jobs as well as providing health information for your databases. For this reason, we felt it was important to include more about the web console in this edition.

Another addition to the book is a chapter on basic query tuning, which was added to the product in a previous release.

How is this book structured?

This book is structured as follows:

- *Chapter 1* includes an introduction to Data Studio and gets you up and running and familiar with the Data Studio Workbench (user interface).

- *Chapters 2* and *3* focus on database administration tasks*:*

  - *Chapter 2* gets you connected to the database teaches you how to create and change database objects as well as how to grant authority to others to see those objects.

  - *Chapter 3* goes into more advanced topics around maintaining the system and providing for recoverability.

- *Chapter 4* introduces the new Health Monitor in Data Studio which monitors the health of your DB2 databases, view alerts, applications, utilities, and storage.

- *Chapter 5* describes how to create a data development project, which is where artifacts you create for subsequent exercises are stored. It also describes how to use the SQL and XQuery editor (and optionally the Query Builder) to create scripts.

- *Chapter 6* introduces the new Job Manager which lets you create and schedule script-based jobs.

- *Chapter 7* discusses the set of basic query tuning capabilities included in Data Studio.

- *Chapters 8, 9, and 10* are focused on database development activities involving creating and debugging database routines and Data Web Services:

  - *Chapter 8* covers SQL stored procedure development and debugging.

  - *Chapter 9* is a short chapter on developing user-defined functions.

  - *Chapter 10* is Data Web Services Development (with advanced topics in *Appendix E)*

- *Chapter 11* provides you with more context around how Data Studio fits in with the greater data management capabilities from IBM, and how you can build on your Data Studio skills with use of these products for tasks such as data modeling and design, monitoring and optimizing database and query performance, managing test data, managing data privacy and much more.

Exercises are provided with most chapters. There are also review questions in each chapter to help you learn the material; answers to review questions are included in *Appendix A*.

A book for the community

This book was created by the community; a community consisting of university professors, students, and professionals (including IBM employees). The online version of this book is released to the community at no-charge. Numerous members of the community from around the world have participated in developing this book, which will also be translated to several languages by the community. If you would like to provide feedback, contribute new material, improve existing material, or help with translating this book to another language,

please send an email of your planned contribution to db2univ@ca.ibm.com with the subject "Data Studio book feedback."

Conventions

Many examples of commands, SQL statements, and code are included throughout the book. Specific keywords are written in uppercase bold. For example: A **NULL** value represents an unknown state. Commands are shown in lowercase bold. For example: The **dir** command lists all files and subdirectories on Windows®. SQL statements are shown in upper case bold. For example: Use the **SELECT** statement to retrieve information from a table.

Object names used in our examples are shown in bold italics. For example: The *flights* table has five columns.

Italics are also used for variable names in the syntax of a command or statement. If the variable name has more than one word, it is joined with an underscore. For example: **CREATE TABLE *table_name***

What's next?

We recommend that you review the following books in this book series for more details about related topics:

- *Getting started with Eclipse*

- *Getting started with DB2 Express-C*

- *Getting started with pureQuery*

- *Getting started with InfoSphere® Data Architect*

- *Getting started with WAS CE*

The following figure shows all the different ebooks in the DB2 on Campus book series available for free at ibm.com/db2/books

Textbooks **

Programming Fundamentals

Database Fundamentals

Software Engineering & Databases

** Good for university / college courses

Career books ***

Ten steps to career success

A career in databases

*** No software taught

Getting Started series

DB2 Express-C

DB2 App Development

Cloud Computing

IBM Data Studio

Eclipse

Java

.NET

C/C++

Web 2.0

Open source development

SOA

Mobile App. development

Info Sphere Data Architect *

Data Warehousing

WAS CE

pureQuery*

Adobe Flex

Ruby on Rails

PHP

Perl

Python

Mac

* Some software is only available as a trial

The DB2 on Campus book series

About the authors

Dimple Bhatia is an Advisory Software Engineer at the IBM Silicon Valley Lab in San Jose, California. She is currently the lead on the Data Studio web console component of Data Studio. She is also working on the common Web tooling infrastructure for all Web-based tools in the InfoSphere Optim portfolio. Before joining the InfoSphere Optim tools team, she worked as the lead on the Federation Catalog, and migration in WebSphere Federation Server. She holds a Master's degree in Computer Engineering from Syracuse University, New York.

Vinod Chirayath currently works on Web Services Engine development for the WebSphere Application Server at the Austin Research Labs. Prior to this role, he worked on the development team that built the administration components for Data Studio. Vinod was also the technical enablement focal for Data Studio speaking at various technical conferences and conducting customer POCs for Fortune 500 companies. He has co-authored several articles/publications and has a patent in the database tools area.

Adam Faeth is member of the User Experience team for Data Studio and Optim products. He holds a master's degree in Human Computer Interaction from Iowa State University.

Praveen Ghantasala is an Advisory Software Engineer with IBM's Lenexa, Kansas Lab. He is a developer in the administration component of IBM Data Studio product group. Praveen is also the current technical enablement focal for IBM Data Studio. Prior to this role, he worked as a lead developer in IBM DB2 .NET provider, IBM DB2 CLI driver, IBM Informix engine (SQL) development, IBM Informix Client Software Development Kit including ODBC, Embedded SQL for C, and OLEDB provider. Praveen holds a Master's degree in Computer Science from India's Andhra University. He also co-authored several articles.

Hassi Norlen is an information developer with IBM's Costa Mesa lab. He has worked extensively with database management and monitoring software such as InfoSphere Optim Performance Manager, IBM Data Studio, and IBM Data Studio web console. His subject of expertise is up and running and installation documentation, as well as user interface development using the progressive disclosure methodology. He started his IBM career ten years ago in the Enterprise Content Management (ECM) field, working on IBM FileNet P8 Application Engine/Workplace XT and IBM Enterprise Records. He holds a Master's degree in physics and a degree in journalism from Sweden's Uppsala University, and worked as a science and technology journalist and a science teacher before joining IBM.

Hardik Patel is a Staff Software Engineer at the IBM lab in Lenexa, Kansas. He is the owner of SQL and XQuery Editor component of Data Studio/InfoSphere Optim products. He is also responsible for SQL Builder, SQL Results view, JDBC Runner and Database Catalog Filtering. Hardik has worked on the Optim Development Studio and the IBM Migration Toolkit (MTK) teams. He holds a Master's degree in computer science from California State University, Fresno. Hardik also does Continuing Engineering for the core components of Data Studio, where he analyzes and provides solutions for customer

problems seen in products that use these core data tools components, including Rational. Hardik has co-authored several articles and tutorials for developerWorks.

Daniel Zilio is a senior developer is a senior developer in the IBM InfoSphere Optim Query Workload Tuner group in the IBM Silicon Valley Lab. He joined IBM in the IBM DB2 Optimizer team and has worked on the IBM DB2 Autonomic Computing team. As a member of the IBM DB2 team, he has worked on database design decision algorithms, query access planning, optimizer cost modeling, query access plan visualization (the explain facility), database simulation, self-tuning memory management, XML design selection, and automatic statistics collection. He was also a member of the team that designed and developed the initial DB2 LUW index advisor, and he later led the team that designed and developed its predecessor: the Design Advisor, which included materialized view, multi-node partitioning, and multidimensional clustering selection. While on the Query Workload team, Daniel designed and created (for DB2 for z/OS and Linux, UNIX, and Windows) a data mart advisor, a workload statistical views advisor (extending the workload statistics advisor), and the facility to capture/gather/view actual and estimated cardinalities for query plans. He also assisted in the development of the workload index advisor, workload statistics advisor, access plan comparison, and what-if index analysis. Before joining IBM, Daniel obtained his PhD from the University of Toronto in the area of physical database design selection, which included creating automatic partition and index selection algorithms.

Contributors

The following people edited, reviewed, provided content, and contributed significantly to this book.

Contributor	Company/ University	Position/ Occupation	Contribution
Dr. Vladimir Bacvanski	SciSpike	Founder	Review.
Onur Basturk	Anadolu University, Computer Research and Application Center	Faculty Member	Review.
Quddus Chong	IBM Silicon Valley Laboratory	Information Developer	Technical edit.
Raul Chong	IBM, Toronto Laboratory	Senior DB2 Program Manager and Evangelist	Review and project management.
Metin Deniz	Anadolu University, Computer Research and Application Center	Software Developer	Review.
Ireneo "Richie" Escarez	IBM Silicon Valley Laboratory	Information Developer	Technical edit and contributions to Chapter 1.
Arlan Finestead	IBM Lenexa Laboratory	Software Engineer	Technical review.
Holly Hayes	IBM Silicon Valley Laboratory	Product Manager, InfoSphere Optim Data Lifecycle Management solutions.	Review and contributions to Chapter 11.
Leon Katsnelson	IBM Toronto Laboratory	Program Director, IM Cloud Computing Center of Competence and Evangelism	Review and project management.
Mark Kitanga	IBM Silicon Valley Laboratory and New Mexico State University	Information Development Intern	Technical edit.

Contributor	Company/ University	Position/ Occupation	Contribution
Anson Kokkat	IBM Toronto Laboratory	Product Manager, DB2 Advanced Recovery Solutions	Review and contributions to Chapter 11.
Cliff Leung	IBM Silicon Valley Laboratory	Development Manager and Architect, InfoSphere Optim Query Tuner products.	Reviewed and provided input on query tuning chapter.
Ivan Lopes, Jr.	IBM Silicon Valley Laboratory	Quality Assurance Engineer	Technical review.
Kenta Nishioka	IBM Silicon Valley Laboratory and University of Washington	Information Development Intern	Review.
Vincent Petrillo	IBM Lenexa Laboratory	Product Manager and Development Manager, IBM Data Studio	Management support.
Kathryn Zeidenstein	IBM Silicon Valley Laboratory	Manager, Data Studio and InfoSphere Warehouse Information Development	Technical editing and project management.

Acknowledgements

The authors owe a significant debt to the authors of the previous edition of this book, which provided the foundation upon which we were able to build for this new edition:

- Debra Eaton
- Vitor Rodrigues
- Manoj K. Sardana
- Michael Schenker
- Kathryn Zeidenstein
- Raul Chong

We greatly thank the following individuals for their assistance in developing materials referenced in this book:

Paolo Bruni and the rest of the Redbook team who wrote materials used in the introduction to the Data Web Services chapter.

Tina Chen, IBM Silicon Valley Laboratory, for her stored procedure Proof of Technology, which served as a basis for the chapter on developing SQL stored procedures.

Holly Hayes, IBM Silicon Valley Laboratory, for her developerWorks article entitled *Integrated Data Management: Managing the data lifecycle*, which was used extensively in Chapter 11.

Robert Heath, IBM Silicon Valley Laboratory, for his technote on using query tuning in Data Studio, which was used as the basis for the material in Chapter 7.

Michael Rubin for designing the cover of this book.

Susan Visser for assistance with publishing this book.

Erin Wilson, IBM Silicon Valley Laboratory, for her instructions on setting up the GSDB sample database, and for the description and diagram used in *Appendix C.*

Ireneo (Richie) Escarez, IBM Silicon Valley Laboratory, for revision editing and contributions to the *Installing Data Studio* section of Chapter 1.

1

Chapter 1 – Overview and installation

IBM Data Studio is a member of the IBM® InfoSphere® Optim™ family of products, which provides an integrated, modular environment to manage enterprise application data and optimize data-driven applications, across heterogeneous environments, from requirements to retirement. This capability is more generally referred to as **Data Lifecycle Management.**

Data Studio consists of a client, which is available in two flavors, and an optional web-based server console. More details about the packaging are described below, in Section 1.1.1.

The Data Studio client is built on the open source Eclipse platform, and is available on both Windows® and Linux® platforms. You can use the Data Studio client at no charge to help manage and develop applications for any edition of DB2® for Linux®, UNIX®, Windows®, DB2 for i, DB2 for z/OS®, or Informix®. It also includes object management and development support for Oracle and Sybase, but this book focuses on DB2 support.

Note:

A common question we get is what capabilities in IBM Data Studio are supported for which data server. This handy document provides a matrix of supported features by database server and release across the *administration* client, the *full* client and the web console.

http://www-01.ibm.com/support/docview.wss?uid=swg27022147

IBM Data Studio replaces other tools that you may have used with DB2 databases in the past. It is a great tool for working with DB2 databases and we hope that you grab a cup of coffee or your favorite beverage, download IBM Data Studio and DB2 Express-C and put this book to good use.

In this chapter you will:

- Learn about Data Studio capabilities, packaging, and community
- Make sure your environment is ready to install the Data Studio product
- Install the Data Studio full client and navigate the Data Studio Eclipse **workbench** (the user interface)
- Install the Data Studio web console

1.1 Data Studio: The big picture

As shown in *Figure 1.1*, Data Studio provides database administration and database development capabilities for DB2. It is the primary tool for production database administration for DB2 for Linux, UNIX, and Windows environments, but also supports object management and routine development for DB2 for z/OS and DB2 for i. As of Version 3.1 Data Studio incorporates advanced administration and development tools from Optim Database Administrator and Optim Development Studio, which are not being developed any further, and is the replacement for DB2 Control Center, which is not developed any more and will be removed from a future version of DB2.
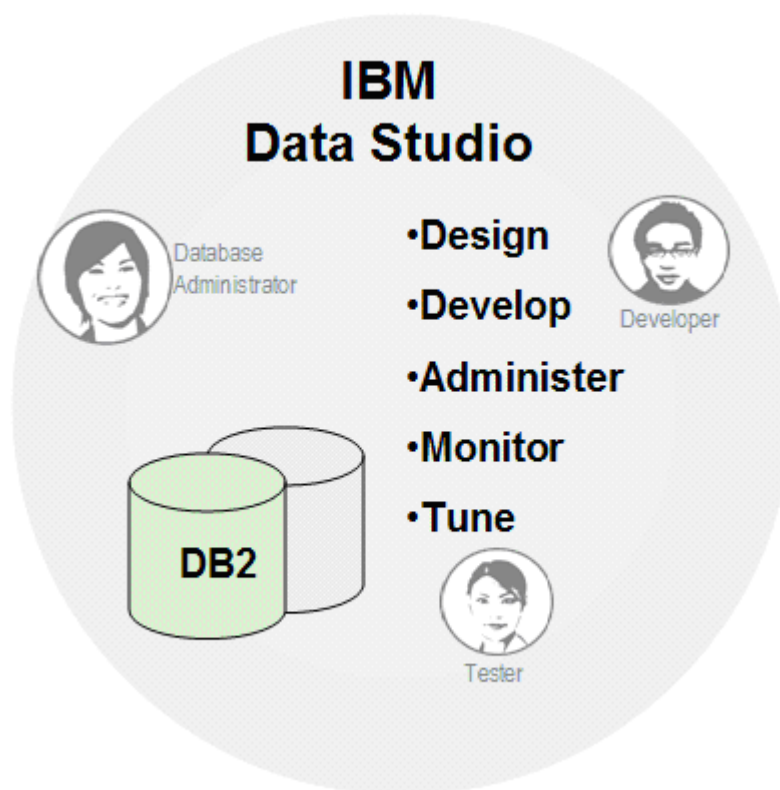
Figure 1.1 – Data Studio provides tools support for DB2 administrators and developers

For data development, it enables you to:

- Use wizards and editors to create, test, debug, and deploy routines, such as stored procedures and user-defined functions

- Use the SQL builder and the SQL and XQuery editor to create, edit, validate, schedule, and run SQL and XQuery queries

- Format queries, view access plans, and get statistics advice to analyze and improve query performance.

- Create, test, debug and deploy SQL or Java procedures (also including PL/SQL procedures for DB2 in compatibility mode and connecting to DB2 using the ANTs Software product IBM® DB2 SQL Skin for applications compatible with Sybase ASE (SSacSA).

- Java procedure support is available only in the full client, described in the next section.

- Create web services that expose database operations (SQL SELECT and data manipulation language (DML) statements, XQuery expressions, or calls to stored procedures) to client applications. Available only in the full client, described in the next section.

- Use wizards and editors to develop XML applications. Available only in the full client.

- Develop JDBC, SQLJ, and pureQuery applications in a Java project. pureQuery provides a way to accelerate Java development as well as provide insights into the Java and database relationship. For more information about pureQuery, see the ebook *Getting Started with pureQuery*. Java development is available only in the full client.

- Bind and rebind packages

- Manage routine and SQL deployments across multiple target development and test databases.

- View and force active connections

- View and manage jobs including job schedules, success or failure notification or actions, and job history

For data and database object management, Data Studio provides the following key features. Typically these tasks are done on test databases that you are using to test your applications. You can:

- Connect to DB2 data sources, filter, sort, and browse data objects and their properties

- Import and export database connections

- Monitor and view database health conditions (not available for DB2 for i)

- Use data diagrams to visualize and print the relationships among data objects

- Use editors and wizards to create, alter, or drop data objects.

- Modify privileges for data objects and authorization IDs

- Analyze the impact of your changes

- Copy tables

- View and edit table data

- These additional features are available with DB2 for Linux, UNIX, and Windows databases

- Manage database instances (including support for DPF and DB2 pureScale topologies) e.g. start and stop, quiesce, configure parameters, define high availability support, etc.

- Back up and recover databases

- Reverse engineer databases into physical models

- Compare and synchronize changes between models, databases, and the data definition language (DDL) used to define objects in the database.

- Manage change plans to coordinate complex or related changes across objects, including destructive changes requiring data and privilege preservation

- Manage table data including collecting statistics, reorganizing, importing, and exporting

- Configure automatic maintenance and logging for DB2 for Linux, UNIX, and Windows

- Create, validate, schedule, and run command scripts

Data Studio gives you the tools you need to become immediately productive on a DB2 data server while you build and enhance your skills into more advanced database development and management tasks. You can read more about additional capabilities provided using data lifecycle management solutions from IBM in *Chapter 11*.

1.1.1 Data Studio packaging

Data Studio is comprised of three installables: the full client, the administration client, and the web console.

- The *full client* includes all administrative capabilities as well as an integrated Eclipse development environment for Java, XML, pureQuery, and Web services. This is the client used in this book because it provides the complete client function as well as the ability to shell-share with other Eclipse-based tools.

- The *administration client* is a lighter weight subset of the full client designed specifically for administrators to get up and running quickly and easily. You can do all the exercises in this book with the administration client except for Data Web Services. Java development, pureQuery development, data web services development, and some other features are not included in the administration client. View a list of the differences in features between the full client and the administration client at http://www.ibm.com/support/docview.wss?uid=swg27022148. Information about installing the administration client is in *Appendix C.*

- The *web console* provides health monitoring, job management, and connection management. It uses a browser interface, but you can access commonly used

tasks such as viewing database status, listing connection, viewing job history, and so on from the Eclipse-based clients.

Note:

For more information about how these components work together and how you can use them in a team environment, see this topic in the Data Studio Information Center:

http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/index.jsp?topic=%2Fcom.ibm.datatools.ds.release.doc%2Ftopics%2Fgetstarted.html

1.1.2 Career path

Getting skilled with Data Studio can help you prepare for a path as a DB2 DBA or developer. Data Studio works with all members of the DB2 family – whether on Linux, UNIX, Windows, i5/OS, or z/OS – so the skills you learn are transferrable across those varied platforms.

At this point, there are no specific professional certifications for Data Studio; however, Data Studio is used in DB2 certification courses such as the one to become an IBM Certified Solution Developer - SQL Procedure Developer (Exam 735).

1.1.3 Popular community Web sites and discussion forum

There is a vibrant community around DB2 data servers, which includes discussions and information about Data Studio, including ChannelDB2.com for videos and social networking, db2university.com for free online courses, and PlanetDB2.com as a blog aggregator. You can read more about these communities in the ebook *Getting Started with DB2 Express-C.*

There is also a developerWorks discussion forum on the Data Studio product that many people in the community and in the software labs monitor and respond to at www.ibm.com/developerworks/forums/forum.jspa?forumID=1086

1.1.4 Related free software

Data Studio is often used with DB2 Express-C and WAS CE. Both are software products from IBM that you can use at no charge.

1.1.4.1 DB2 Express-C

DB2 Express-C is the free version of the DB2 database server. You can use it for development, test, deployment in production, and also embedded in your applications. It is built using the same code base as fee-based DB2 editions; this means that applications developed to work on DB2 Express-C will work with no modification on other DB2 editions. DB2 Express-C includes the Oracle compatibility feature which allows Oracle professionals to easily work with PL/SQL, and other Oracle features in DB2. This book uses DB2 Express-C for all exercises. For more information visit www.ibm.com/db2/express or review the ebook *Getting started with DB2 Express-C.*

1.1.4.2 WebSphere® Application Server Community Edition

Data Studio (full client) lets you build and deploy web services from database objects or queries. The examples used later in this book assume you are using IBM WebSphere Application Server Community Edition (WAS CE) version 2.1 as the application server for deployment of those Web services. WAS CE is a lightweight Java™ EE 5 application server available free of charge. Built on Apache Geronimo technology, it harnesses the latest innovations from the open-source community to deliver an integrated, readily accessible and flexible foundation for developing and deploying Java applications. Optional technical support for WAS CE is available through annual subscription. For more information, visit www.ibm.com/software/webservers/appserv/community/ or review the ebook *Getting started with WAS CE*

1.2 Getting ready to install Data Studio

This section explains the software prerequisites for Data Studio and provides links to downloads for other software that you may find useful when going through this book:

1. Ensure that your computer is using any of the following operating systems:

Linux

- Red Hat Enterprise Linux 5.0 AS/ES x86-32 or x86-64 running in 32 bit mode

- Red Hat Enterprise Linux Desktop 6.0 AS/ES x86-32 or x86-64 running in 32 bit mode

- Red Hat Enterprise Linux Server 6.0 AS/ES x86-32 or x86-64 running in 32 bit mode

- SUSE Linux Enterprise Server 10 x86-32 or x86-64 running in 32 bit mode

- SUSE Linux Enterprise Desktop 10 x86-32 or x86-64 running in 32 bit mode

- SUSE Linux Enterprise Server 11 x86-32 or x86-64 running in 32 bit mode

- SUSE Linux Enterprise Desktop 11 x86-32 or x86-64 running in 32 bit mode

    Note: Other distributions of Linux, such as Ubuntu, may also be used, but are not officially supported. Use at your own risk.

Windows

- Microsoft Windows XP Professional (SP2) x86-32 or x86-64 running in 32 bit mode

- Microsoft Windows Vista (Business, Enterprise, Ultimate) x86-32 or x86-64 running in 32 bit mode

- Microsoft Windows 7 x86-32 or x86-64 running in 32 bit mode

2. Review the installation prerequisites in the installation roadmap in the IBM Data Studio Information Center:

http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/index.jsp?topic=/com.ibm.
datatools.base.install.doc/topics/c_roadmap_over_product.html

It is also a good idea to check the IBM technotes for any late-breaking changes
to installation prerequisites:
http://www.ibm.com/support/docview.wss?uid=swg27021949

3. Ensure you have proper authority. For a launchpad installation, which is what
    is shown in this chapter, you must be an admin user, which means that you
    can write to the default common installation location.

    ▪   On Linux operating systems, this is the "root" or any user who is using
        "sudo" to start Installation Manager.

    ▪   On a Microsoft Windows XP operating system, a user with write
        administrative privileges is any user who is a member of the
        "Administrators" group.

    ▪   On a Microsoft Windows Vista and Windows 7 operating systems, this
        is the user who is using "Run As Administrator".

Ensure that your user ID does not contain double-byte characters.

Note:

To perform a non-administrative installation, you cannot use the launchpad. You
must instead switch to the InstallerImage_<*platform*> folder in the disk1
directory, and run `userinst.exe` (for Windows), or `userinst` (for Linux).

---

4. If you don't already have a DB2 data server installed, you can download and
    install DB2 Express-C Version 9.7

    We will use the free version of DB2, DB2 Express-C, for this book, although any
    supported version of DB2 you already have is fine as well. To download the latest
    version of DB2 Express-C, visit www.ibm.com/db2/express and choose the
    appropriate file to download for the operating system you are using. Ideally, you
    should install DB2 Express-C before you install Data Studio. Refer to the free
    ebook *Getting Started with DB2 Express-C* for more details.

5. Optionally, if you are planning on doing any Data Web Services exercises, you can
    download and install WebSphere Application Server Community Edition (WAS CE)
    Version 2.1.
    https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&sour
    ce=wsced_archive&S_PKG=dl.

6. Optionally, download the "Sample Outdoor Company" (GSDB) sample database.

Although you can use the SAMPLE database included with DB2 for many of the exercises in this book, we use another database, called GSDB that enables us to illustrate more capabilities. This database represents the sales and customer information for the fictional Sample Outdoor Company.

You can download the sample database from http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/topic/com.ibm.sampledata.go.doc/topics/download.html

Figure 1.2 shows the link you click on to get the sample database used in this book. It's fairly large (about 43 MB), so it might take some time to download depending on your download speed.

Download the GSDB sample data

The GSDB sample database is available for you to use in your own projects and for learning about IBM® products. The sample database contains a rich set of sample data that follows the fictional Sample Outdoor company and its sales and operations.

Download
GSDB Sample database (v2r2)

Related articles, tutorials, and resources

developerWorks® articles

  • IBM developerWorks query for articles that use the GSDB sample data

InfoSphere™ Data Warehouse for DB2® Linux, UNIX, and Windows tutorials

  • SQL Warehousing tutorial
  • Cubing Services tutorial
  • Mining tutorial

Figure 1.2 – Link to GSDB database from the IBM Data Studio Information Center

We will cover how to set up the database later in the next chapter where you will also learn how to create a connection to the database.

7.   Download the IBM Data Studio product.

To download Data Studio, find the link to the package you want on the Data Studio download page on developerWorks (*Figure 1.3*): http://www.ibm.com/developerworks/downloads/im/data/

Component /Operating system	Version	Size	Method	Download
Administration client Red Hat Linux, SUSE Linux, Windows	V3.1	359MB - 362MB	HTTP | Download Director	▼ Download
Full client Red Hat Linux, SUSE Linux, Windows	V3.1 *	1211MB - 1212MB	HTTP | Download Director	▼ Download
Full client Windows, Linux	V3.1 *	180MB	Installation Manager ** (Recommended)	▼ Download
Web console AIX, Red Hat Linux, SUSE Linux, Windows	V3.1	175MB - 202MB	HTTP | Download Director	▼ Download

Figure 1.3 – Links to Data Studio downloads on developerWorks

The exercises in this book assume you are using the full client, but you can download the administration client if you prefer and then follow the instructions in *Appendix C* to install. If you want to work with the web console, you can go ahead and download that as well.

Note:

The Installation Manager method shown in *Figure 1.3* actually downloads a very small executable file. Once that file is invoked, if you already have Installation Manager on your machine, it'll reuse that instance of Installation Manager to install Data Studio from a remote repository. If you don't already have Installation Manager on your system, it will then install both Installation Manager and Data Studio, also from remote repositories.

A direct link to the registration page for the full client is here:
http://www.ibm.com/services/forms/preLogin.do?lang=en_US&source=swg-idside

A direct link to the registration page for the administration client is here:
https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=swg-idssa

---

Note:

If you do not have an IBM ID already, you will need to create one. You may need to wait for some time (perhaps even as long as a day) before being allowed to download the code.

After you get through registration, you can choose the Linux or Windows package. We will walk through the installation process in the next section.

1.3 Installing the Data Studio full client

The Data Studio full client can be installed using the Launchpad GUI, which launches IBM Installation Manager, or silently, which means you create a response file of your chosen installation options, and then run that response file. Silent install is mainly useful for larger installations in which installation must be pushed out to many machines.

IBM Installation Manager is a program for installing, updating, and modifying packages. It helps you manage the IBM applications, or packages, that it installs on your computer. Installation Manager does more than install packages: It helps you keep track of what you have installed, determine what is available for you to install, and organize installation directories. For more in Installation Manager, see this topic in the Data Studio Information Center:
http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/topic/com.ibm.datatools.base.install.doc/topics/c_plan_imover.html

This chapter focuses on the Launchpad installation. It assumes you do not have IBM Installation Manager installed. This means that installing Data Studio starts by installing IBM Installation Manager. If you choose to install additional products that also use that release of Installation Manager, you do not need to install Installation Manager again.

Figure 1.4 shows the installation process described in this chapter.
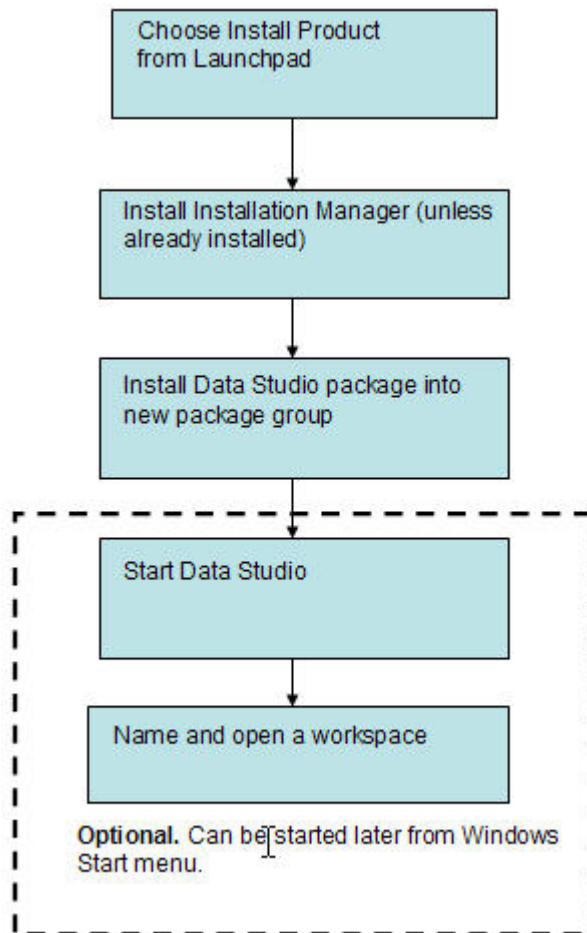
Figure 1.4 – A basic installation flow

Follow these steps to install the Data Studio full client:

1. After you unzip the download package, start the launchpad as follows:

▪ Windows: Execute the `setup.exe` file located in the ibm_data_studio_full_client_v31_windows directory as shown in *Figure 1.5*.
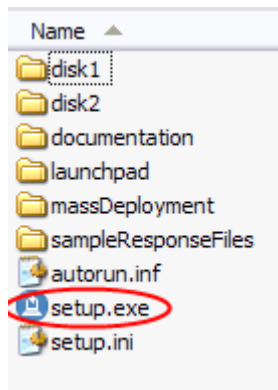
Figure 1.5 – Click setup.exe from unzipped Data Studio package

- ▪ Linux: Execute the `setup` command from the root path where you unzipped the image.

2. The Welcome screen comes up. In the left pane, select *Install Product* as shown in *Figure 1.6.*
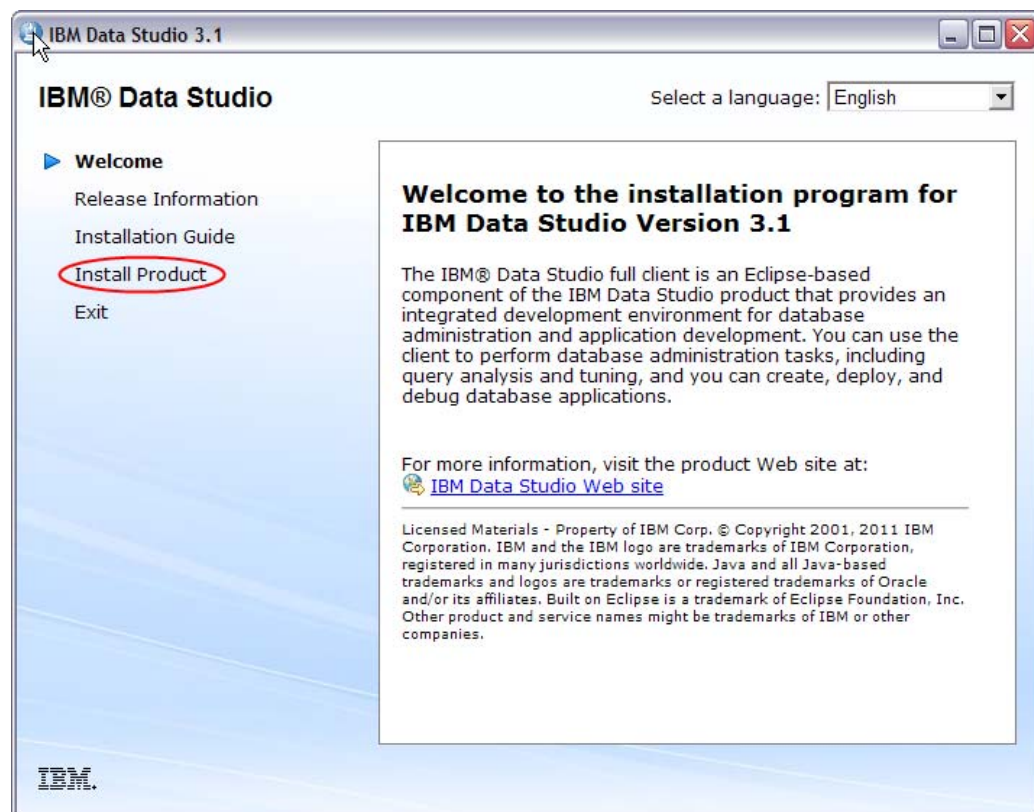
Figure 1.6 – Click Install Product to launch Installation Manager

3. You are given the option for administrative and non-administrative installations. Select *Administrative Installation* to continue.
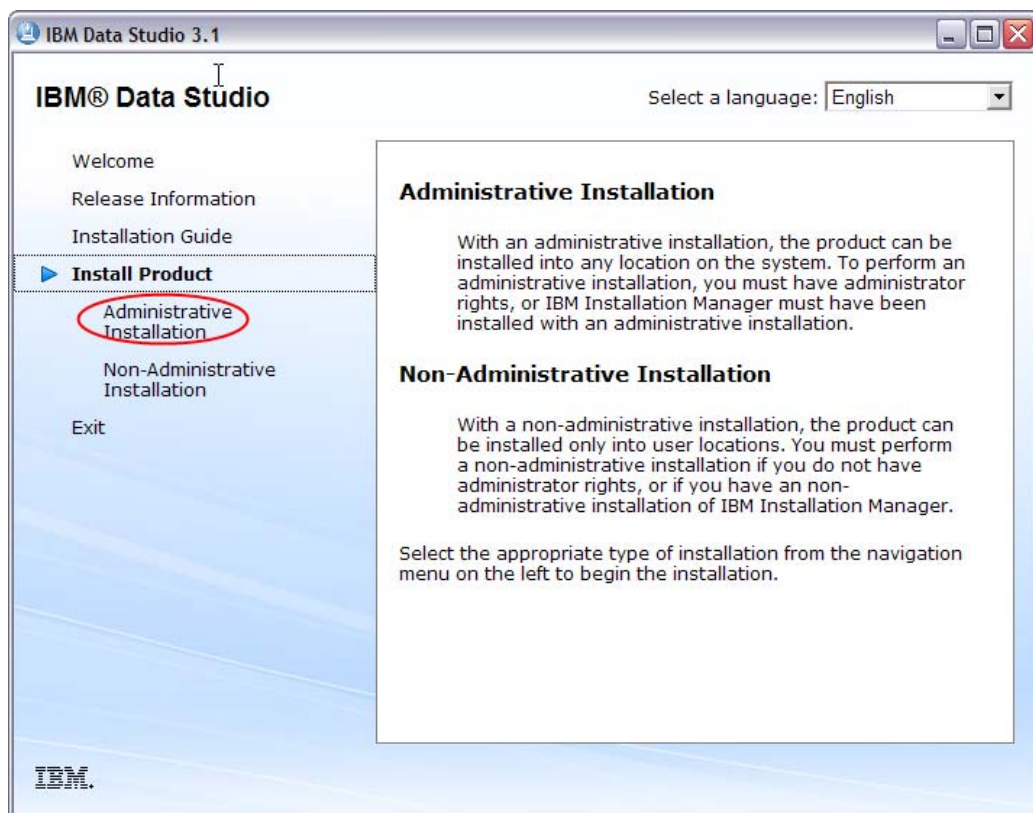
This launches Installation Manager. You will then see a screen that lets you choose which packages to install.

4. Assuming you already have Installation Manager on your machine, you will select the default settings to install Data Studio as shown in *Figure 1.7*. Then click *Next*.
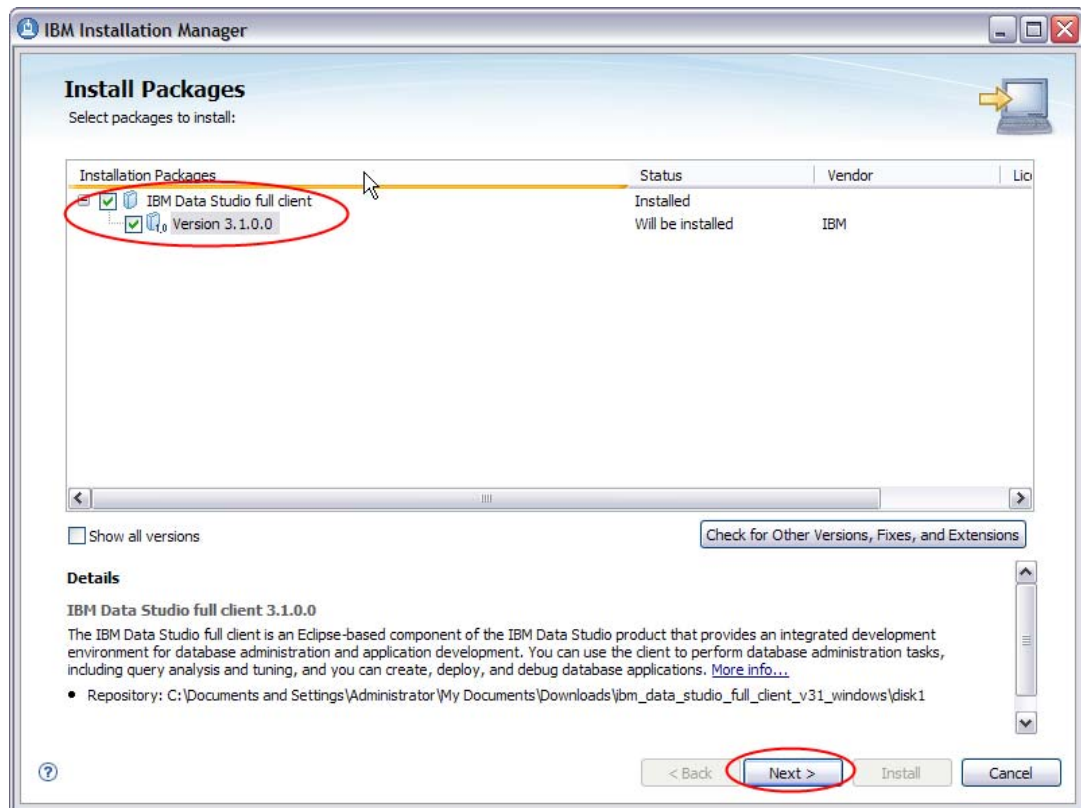
Figure 1.7 – Install Data Studio packages

5. After accepting the license, click *Next*. Depending on what is installed on our computer, you may then be presented with a screen that lets you specify the location directory for shared resources *(Figure 1.8)* You can keep the defaults; however, you'll want to keep in mind that you should choose a drive with more space than you think you need just for Data Studio in case you decide to shell-share with other Eclipse-based products in the future.
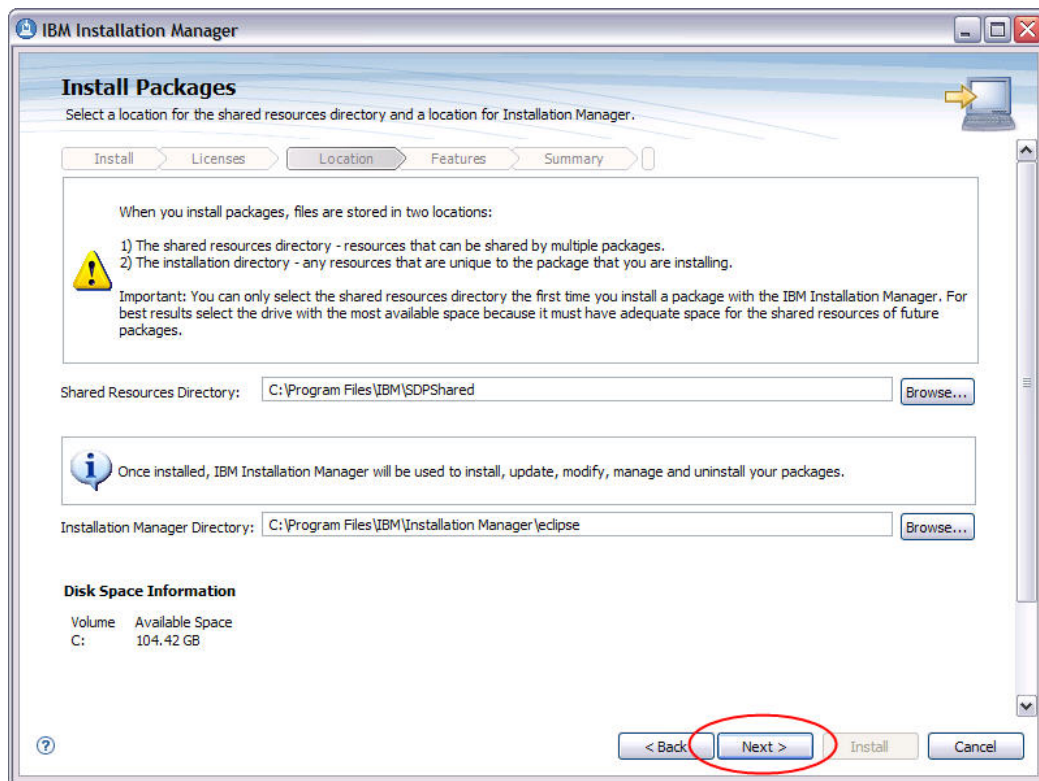
Figure 1.8 – Select location for shared resources

6. You will then see a screen that lets you choose whether to create a new package group or extend an existing one. Because we are installing on a machine that does not include any existing package groups, select the radio button to *Create a new package group*, as shown in *Figure 1.9*.
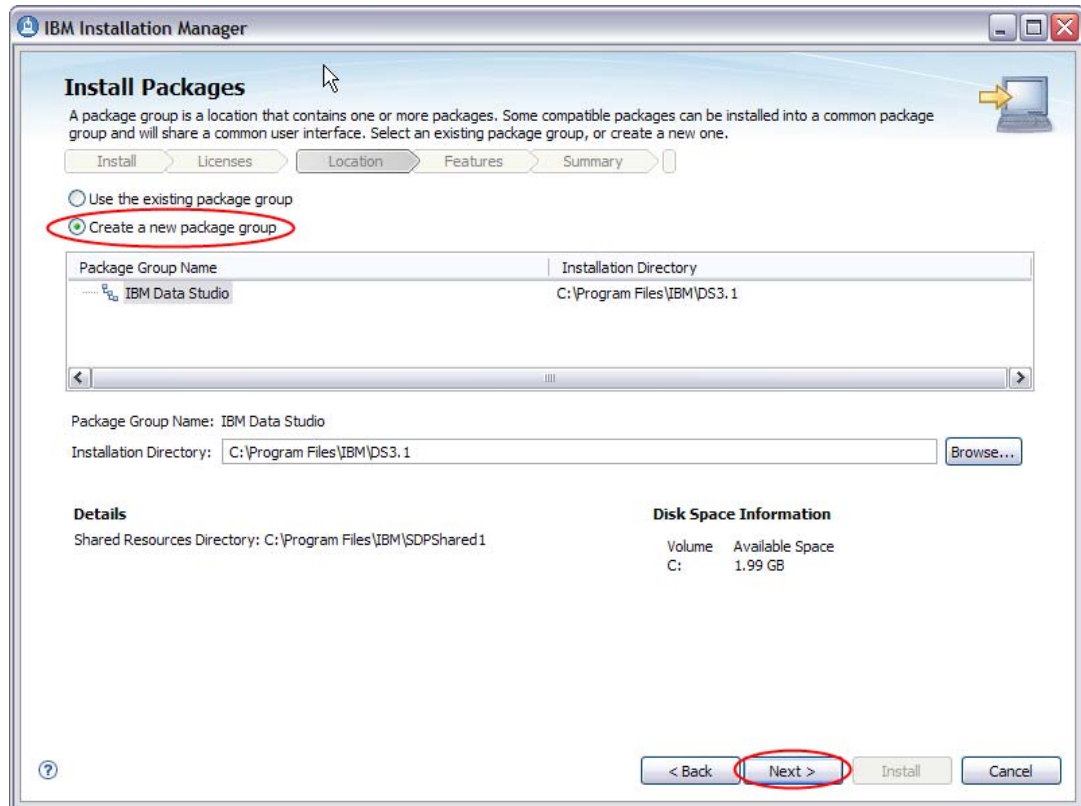
Figure 1.9 – Create a new package group for Data Studio

7. In the next screen, take the default option to install the Eclipse that is included with the Data Studio installation.

8. The next screen lets you choose any additional translations you may wish to install. Select all appropriate translations and then click *Next*.

9. The next screen shows the lists of features to be installed; take the defaults and then click *Next*.

10. The next screen lets you configure how your help system accesses the help content. The default setting is to access your help content from the web. You can change these configuration settings anytime after the product installation.
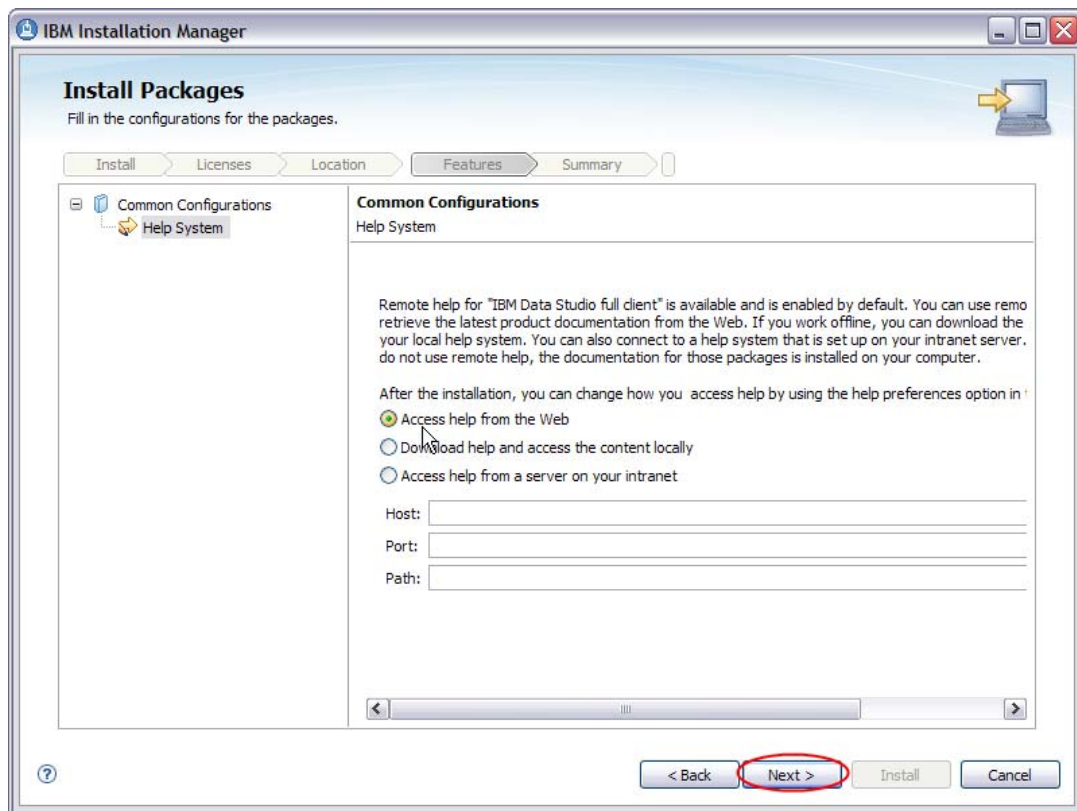
Figure 1.10 – Configuring the help system

11. Finally, you are presented with a summary screen from which you can click the *Install* button as show in *Figure 1.11*.
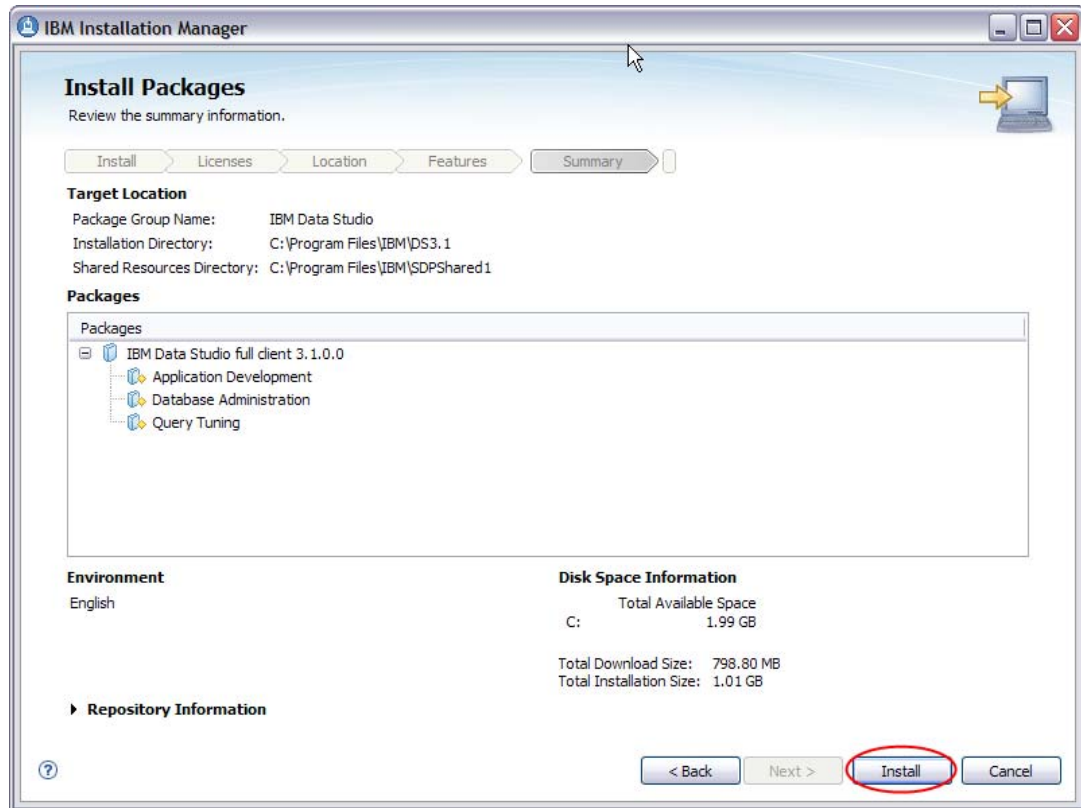
Figure 1.11 – Review summary information and then click Install

Installation Manager begins the installation. There may be a pause in the progress bar at some point; be sure to wait and not interrupt the processing. When the product successfully installs, you see the screen shown in *Figure 1.12*.
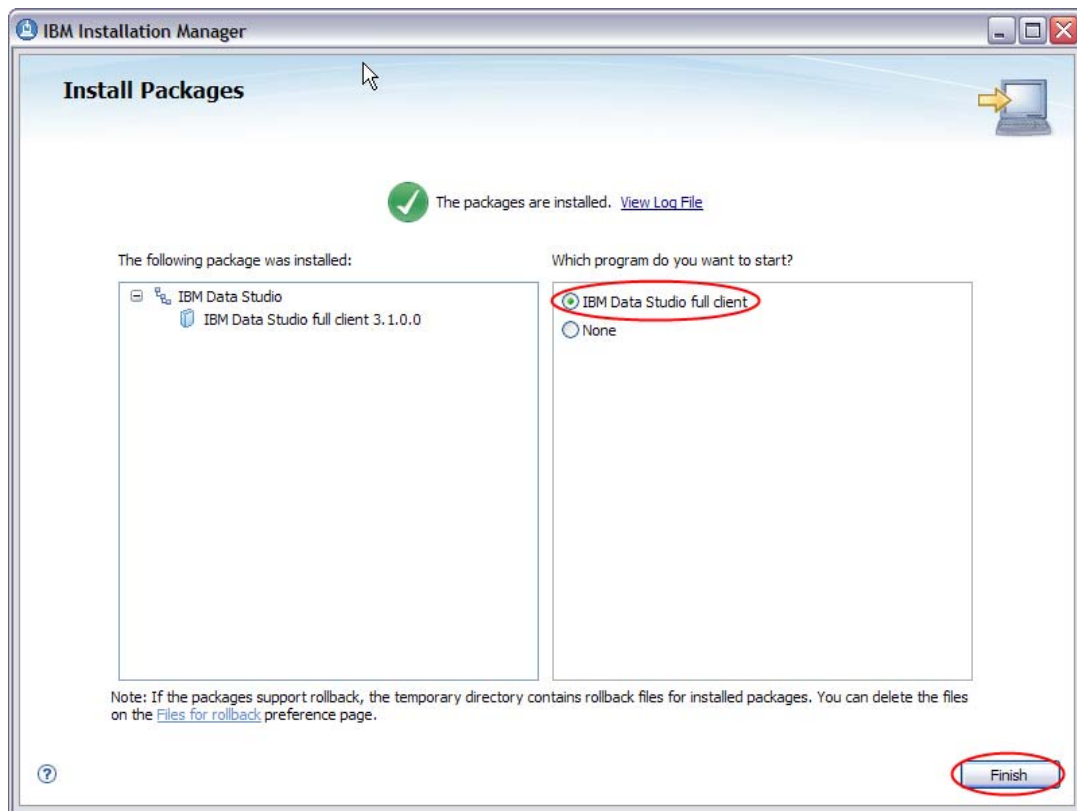
Figure 1.12 – Congratulations! A successful install.

12. From the success screen shown in *Figure 1.12*, click on *Finish* to bring up Data Studio.

13. You will be asked to select a Workspace name. Enter `GettingStarted` as the name of your workspace as shown in *Figure 1.13*.

Note:

A *workspace* is a location for saving all your work, customizations, and preferences. Your work and other changes in one workspace are not visible if you open a different workspace. The workspace concept comes from Eclipse.
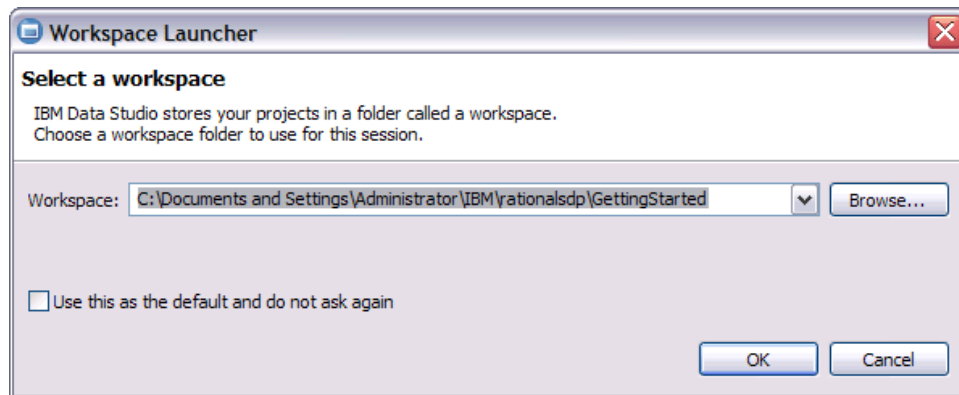
Figure 1.13 – Enter a workspace name

The default perspective appears and displays the *Task Launcher* as shown below in *Figure 1.14.*
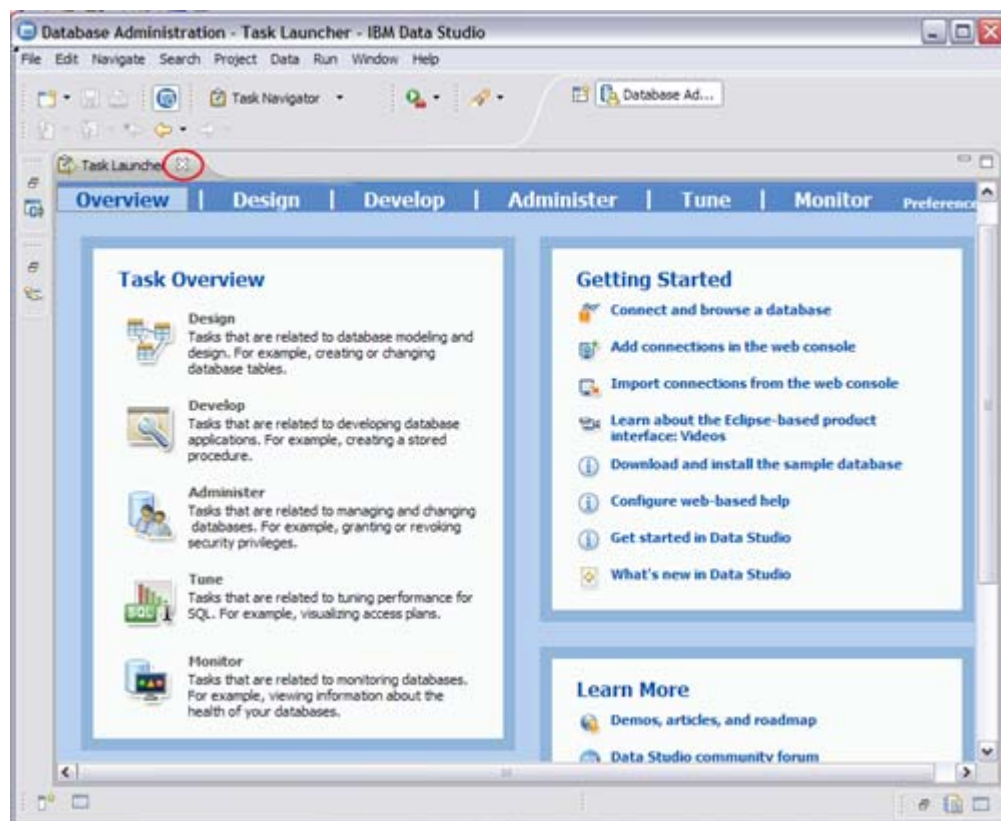
Figure 1.14 – The Task Launcher in Data Studio

13. The Task Launcher highlights the key tasks that are available in each phase of the data management lifecycle. You can use it to launch the initial context for each of the tasks. Click any of the tabs in the Task Launcher to view tasks specific to a single phase of the data management lifecycle, then click any of those tasks to get

started. You will also see links to online resources in the *Learn More* section. Feel free to explore some of these materials, leave the Task Launcher view open, or go ahead and click on the *X* as shown in *Figure 1.14* to close it.

As you'll learn more about in the next section, a perspective is basically a configuration of views and actions that are associated with particular tasks. A view shows your **resources**, which are associated with editors. The default perspective for Data Studio is the **Database Administration** perspective as shown in *Figure 1.15*. You can see the names of the various views there including Administration *Explorer* and *Data Project Explorer*. We'll explore the views and the various perspectives a bit more in section 1.4.

Note: If by some chance you already had a workspace named GettingStarted, it would appear with the default views under which you had previously saved it.
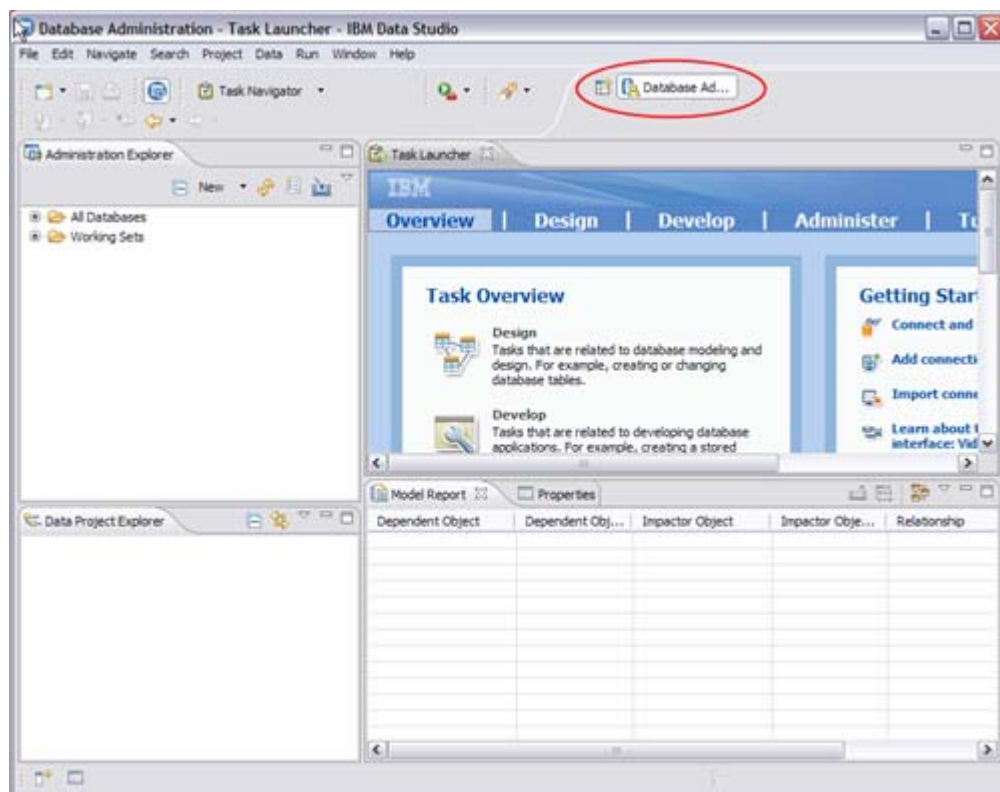
Figure 1.15 – The default Database Administration perspective in Data Studio

1.4 Touring the Data Studio Client workbench

The term **workbench** refers to the desktop development environment. This concept is from Eclipse. If you are familiar with Eclipse, you may skip this section. The workbench aims to

achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management, and navigation of workspace resources.

Each Workbench window contains one or more perspectives. **Perspectives** contain views and editors and control what appears in certain menus and tool bars based on a certain task or role. So you will see different views and tasks from the Debug perspective (for Java debugging) than you will for the Database Administration perspective.

Let's look at the Java perspective for fun.

One way to open a different perspective is to click on the icon shown below in *Figure 1.29* and select Other… than select *Java*. An alternate way to open a perspective is to click on *Window -> Open Perspective*.
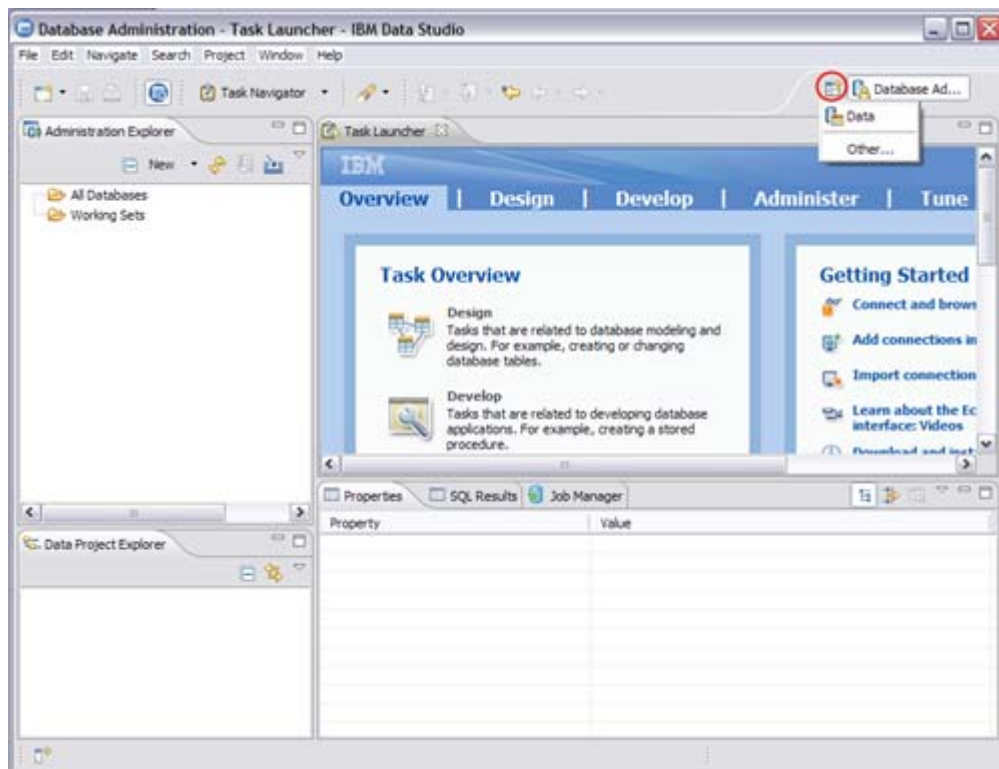
Figure 1.29 – Opening up a different perspective

As you can see by comparing *Figure 1.29* with *Figure 1.30 (*below*)*, the Java perspective has a different task focus (Java development) than the Database Administration perspective. The outline in this case, for example, would work with Java source code in the editor. The explorer shows Java packages as opposed to database objects.
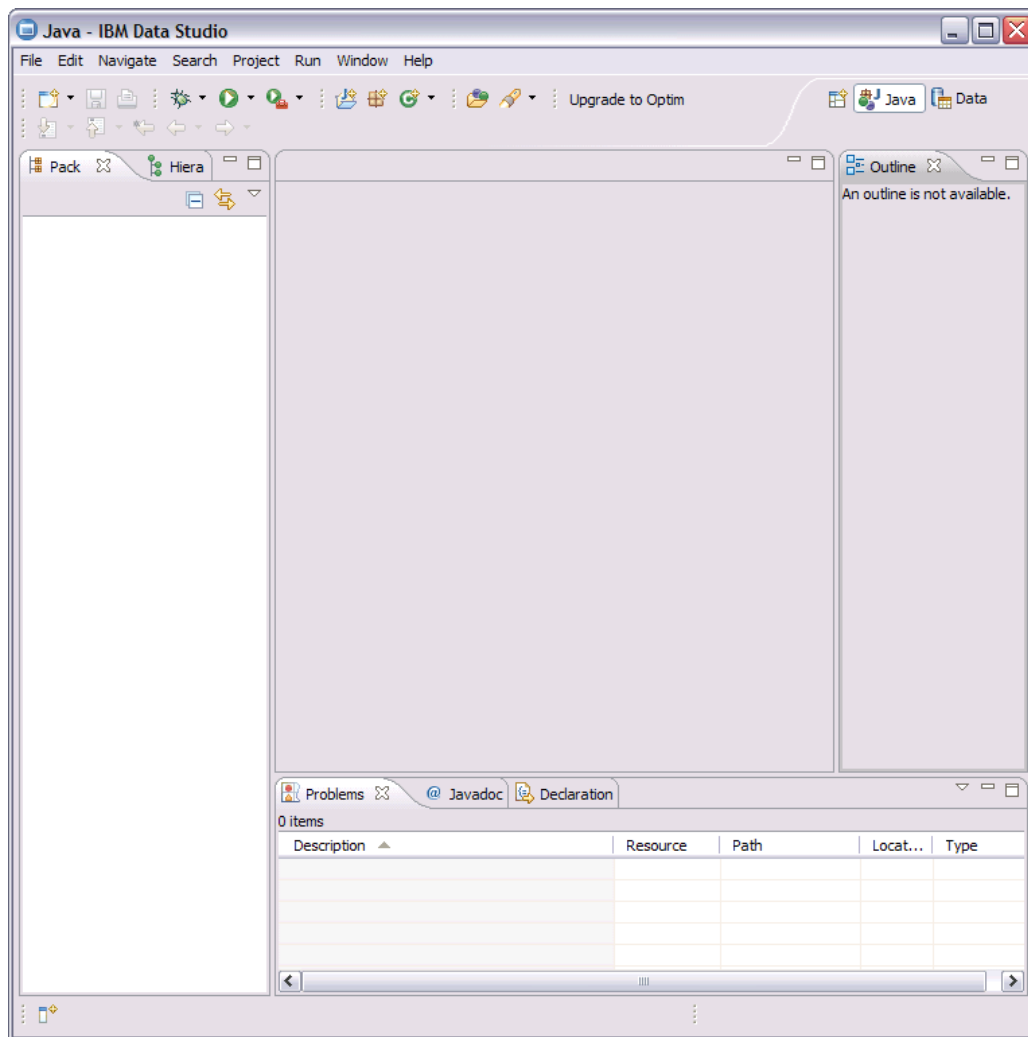
Figure 1.30 – The Java perspective

Click on the *Data* perspective to switch back again so we can describe more fully the capabilities of the Data perspective.

Note:

For more information about perspectives and views, see the ebook *Getting Started with Eclipse.*

1.4.1 Touring the Database Administration perspective and its views

Because most of the work you'll do in this book is in the Database Administration perspective, let's go ahead and change perspectives by clicking on the icon shown in *Figure 1.29* and selecting *Database Administration*, which once again brings up the perspective shown in *Figure 1.31*.

As we described earlier, *views* are the windows you see on workbench such as Administration Explorer and Properties. A view is typically used to navigate a hierarchy of information, open an editor, or display properties for the active editor. The changes that you make to the views (their sizes and positions), and the resources that you create in the views are saved in your workspace, as we mentioned previously.
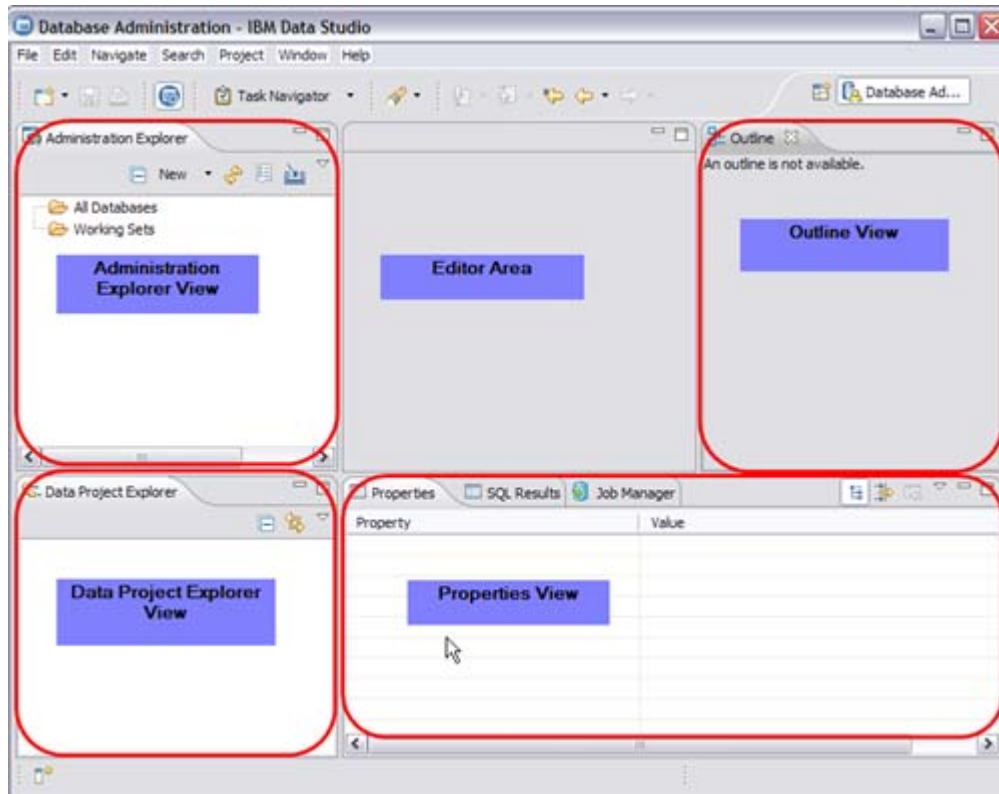
Figure 1.31 – Database Administration perspective views

The views shown in *Figure 1.31*, working counterclockwise from the top left, are described in *Table 1.1* below.

View	Description
Administration Explorer	This view allows you to administer a database. It automatically displays detected databases, but you can add new database connections.
Editor area	Typically used to view and manipulate data. For example, the Object List Editor lets you view and manipulate the objects within a database.
Outline	Displays an outline of a structured file that is currently open in the editor area and lists structural elements. So if you were editing an XML file, you would see the elements of the XML file in an outline format.
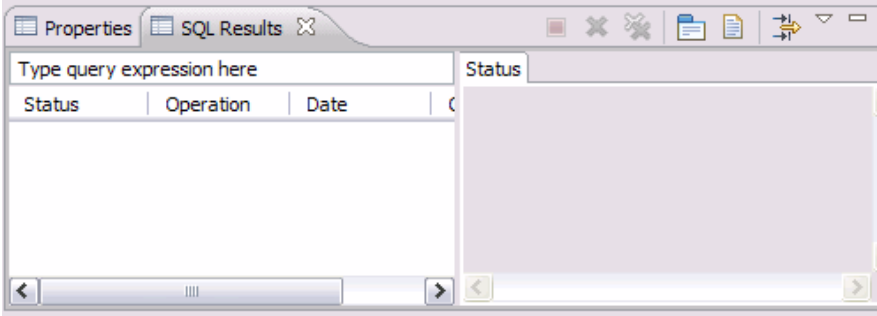
View	Description
Properties	This view shows the properties of the object currently selected in the workspace. For some objects, you can use this view to edit properties, such as making changes to database objects selected in the Administration Explorer. From this view you can also see the *SQL Results* tab, which brings up that view, described below.
SQL Results	Shows results after you execute SQL or XQuery statements.
Data Project Explorer	This view is used by a database developer. It shows Data Development projects (which you will use for SQL and XQuery scripts, stored procedures, functions and Data Web services) and Data Design projects.

Table 1.1 – Views in the default Data perspective

1.4.2 Manipulating views

The basic view controls are shown in *Figure 1.32*.
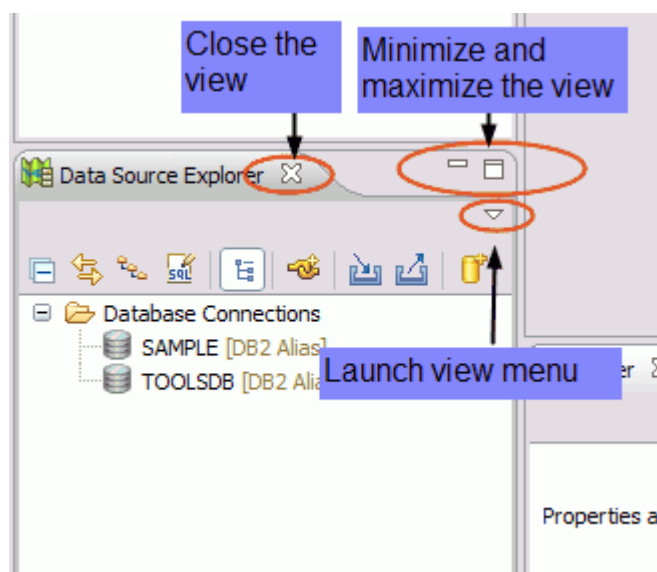
Figure 1.32– View controls

To close a view, click on the *X* to the right of the view name as shown in *Figure 1.32*. There's no need to panic if you close a view accidentally. Simply go *Window -> Show View* and select the view you want to re-open. (See *Figure 1.33* for an example.) If you don't see the view you want, click *Other…*
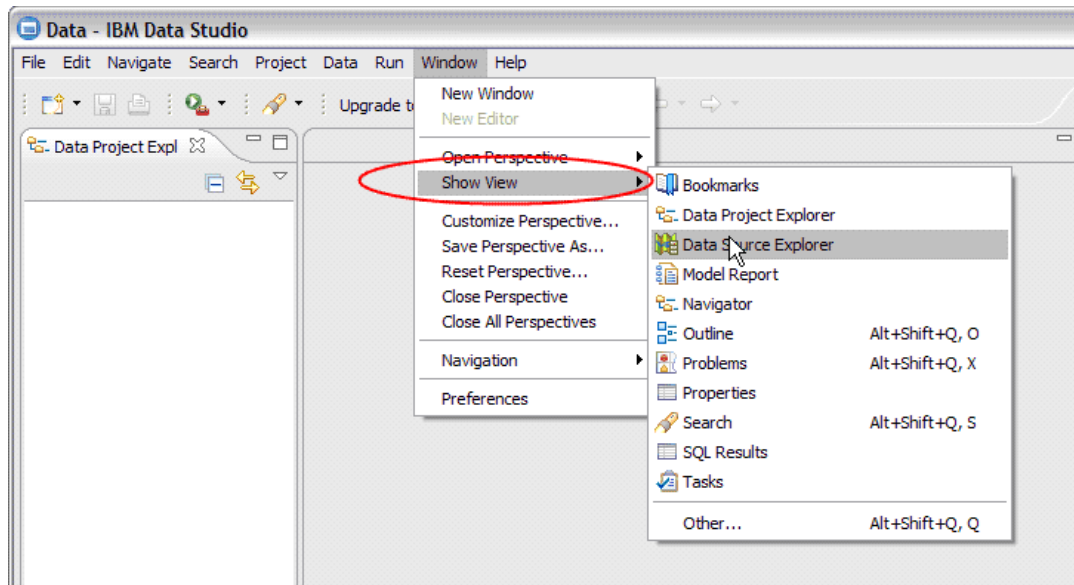
Figure 1.33– Making a closed view reappear

1.4.3 Resetting the default views for a perspective

We encourage you to play around with the views and perspectives in the Workbench. For people not familiar with Eclipse, it can seem a bit strange to have views appearing and disappearing. If you get to the point where you just want it back to the way it was before you started playing, you can reset the perspective from the Window menu as shown in *Figure 1.34.*
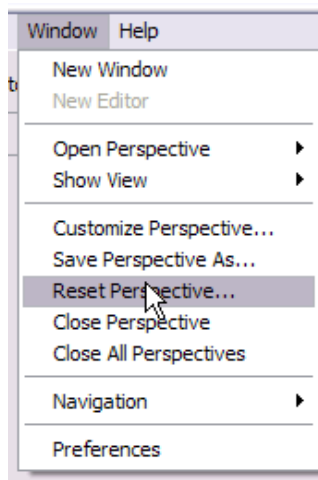
Figure 1.34 -- Reset the views to the defaults for the currently open perspective

> **Note**:
>
> The *Reset Perspective* menu option shown in *Figure 1.34* only resets the current perspective. If you want to change a different perspective, you can go to *Windows -> Preferences -> General -> Perspectives*, choose a perspective and click the *Reset* button. The next time you open the Perspective, it will be restored to the default layout.

1.5 Getting ready to install Data Studio web console

The web console part of Data Studio consists of a server and a client that you use in a web browser. The server must be running to provide its intended functionality such as health monitoring and alerts, or scheduling of jobs. You will learn more about these capabilities in *Chapter 4* and *Chapter 6*, respectively. If you want to use these capabilities you must install and configure the Data Studio web console, and this section helps you prepare for that.

For the purposes of this book you will install Data Studio web console on your local computer. In a production environment you would install the Data Studio web console on a dedicated computer.

1.5.1 Installation overview and first steps

In the subsequent sections, you will learn how to install the web console. This chapter describes the installation for single user access to the Data Studio web console using the default administrative user ID that you create during the installation. To learn more advanced topics such as integrating the Data Studio web console with Data Studio full client and running the Data Studio web console in multi-user mode, see *Appendix B.*
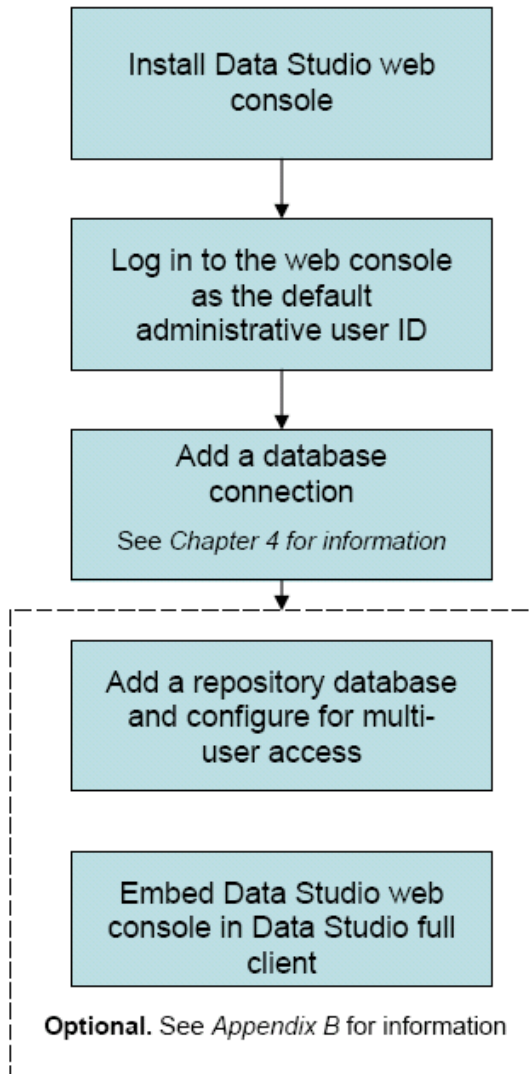
Figure 1.35 – Up and running tasks

1.5.2 Before you install

Before you install Data Studio web console:

1. Review the installation requirements at:
   http://www.ibm.com/support/docview.wss?uid=swg27021949

2. Review the installation prerequisites in the installation roadmap in the Data Studio information center at:
   http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/topic/com.ibm.datatools.db.web.health.install.doc/topics/dshm_roadmap.html

3. Download the IBM Data Studio web console from here:
   http://www.ibm.com/developerworks/downloads/im/data/

1.6 Installing the Data Studio web console

The server-side component of the Data Studio web console can be installed either using the graphic user interface (GUI), through a command-line console, or through the silent install option. The silent install option lets you edit a sample response file with your chosen installation options, and then run that response file. Silent installs are mainly useful for larger installations in which installations must be deployed to many computers.

Follow these steps to install the Data Studio web console:

1. After you unzip the download package, start the installation program as follows:

   - Windows: Run the `DSWC.v3.1.install-on-windows-x86_64.exe` file located in the Data Studio web console installation media.

   - Linux and UNIX: Run the `DSWC.v3.1.install-on-<platform>.bin` command from the root path where you unzipped the image.

2. The splash screen comes up. Click *OK* to continue as shown in *Figure 1.36.*

Figure 1.36 – IBM Data Studio web console installation program splash screen

3.   The Welcome screen comes up. Close all applications and click *Next*.
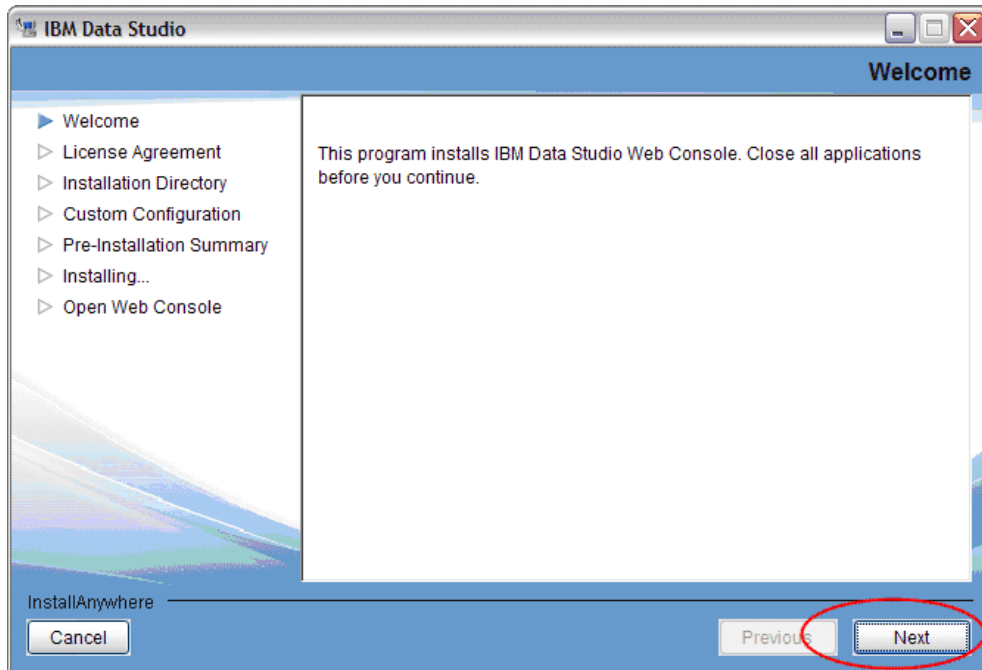
Figure 1.37 – From the Welcome, click Next

4.   You are given the option to accept the license agreement. Select *I accept the terms in the license agreement* to continue.
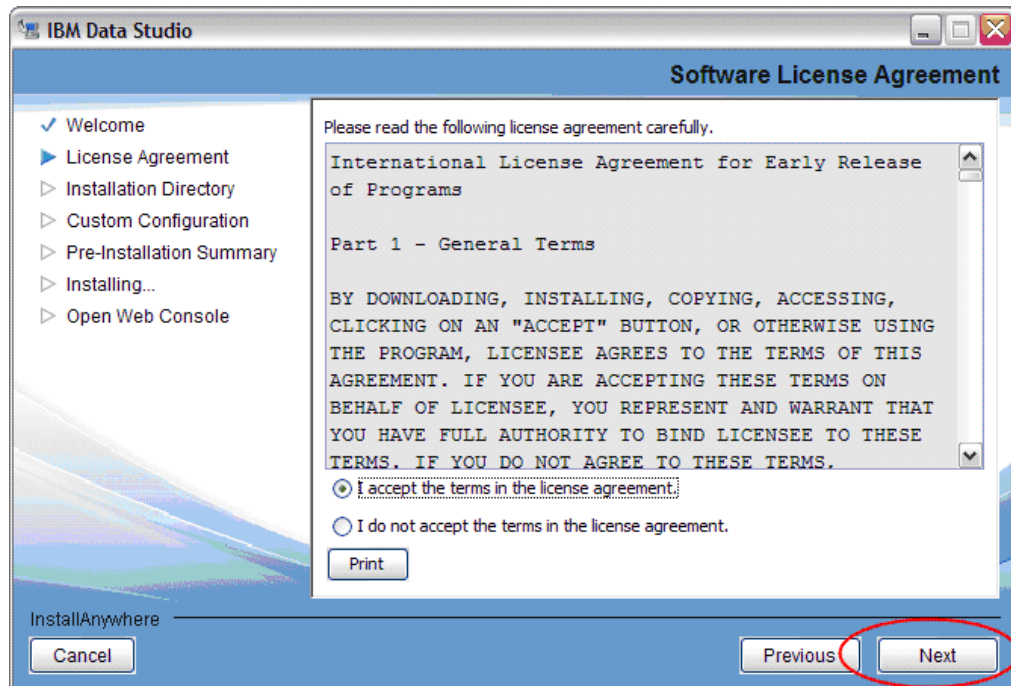
Figure 1.38 - Accept the license agreement

5. Select *Install a new product* to install the Data Studio web console as shown in *Figure 1.39* Choose the default installation directory, or browse to a directory of your choice and then click *Next*.
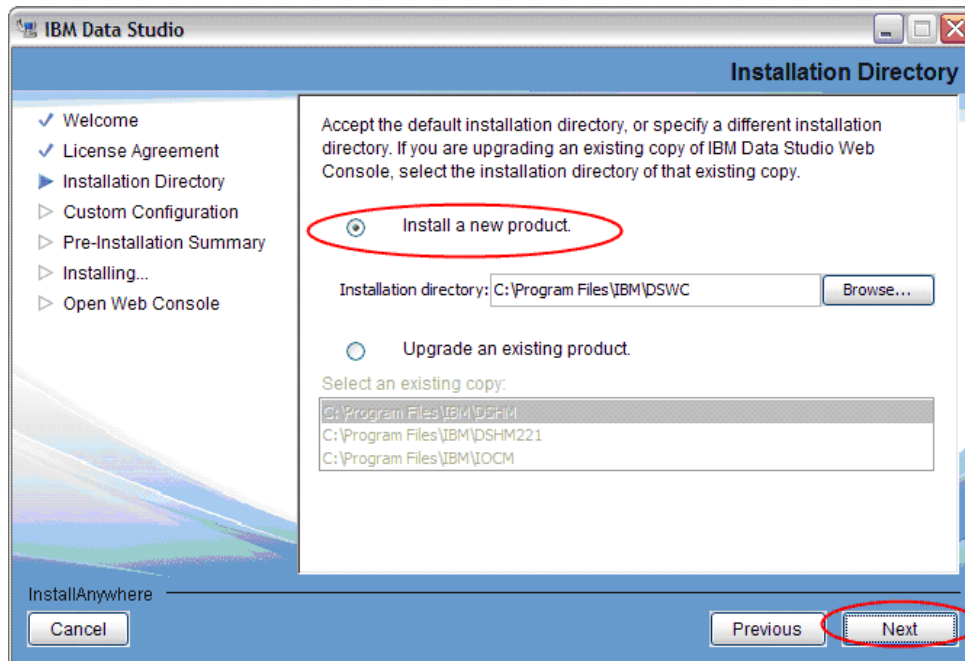
Figure 1.39 – Install a new copy of Data Studio web console

> **Note:**
>
> The Data Studio web console installation program can also update an installed version of IBM Data Studio Health Monitor (the predecessor product to the Data Studio web console), if you have that program installed. If you choose to update an existing Data Studio Health Monitor installation, all the existing product settings except the port numbers and the default administrative user ID are transferred over from the previous installation. You must specify new product port numbers, and a new default administrative user ID for the Data Studio web console.

6. Specify a user ID and password for the ***default administrative user*** as shown in *Figure 1.40,* then click *Next*.

> **Note:**
>
> You can use the default administrative user ID to log in to the web console in single-user mode, and to perform web console administrative tasks such as adding database connections and configuring email alerts. The web console default administrative user ID is

separate from the other administrative user IDs that are used with the product, such as the database user IDs that are required when you add database connections. If you want to allow additional users to log in with their own user IDs, you must configure the Data Studio web console for multi-user access. For more information, see *Appendix B*.
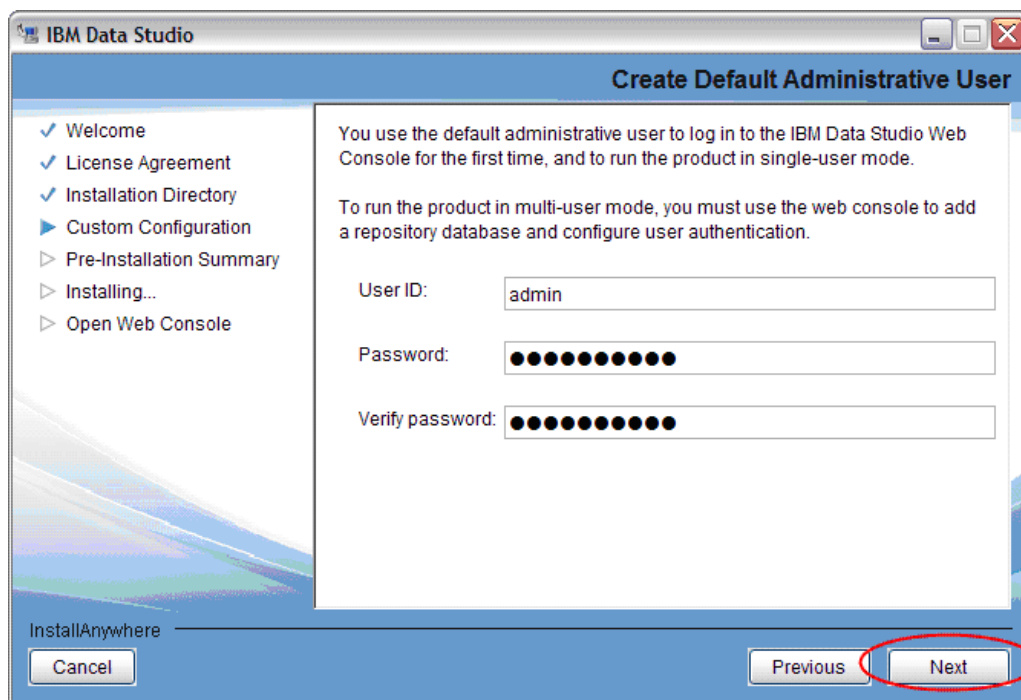
Figure 1.40 – Specify user ID and password for the default administrative user

7.   Specify the port numbers that will be used to connect to the Data Studio web console. You must enable at least one port to be able to log in to the web console. In addition to the web console URL ports, you must also specify a control port that is used locally by the application as shown in *Figure 1.41*. Make sure that the three ports you selected are not used by any other products that you have installed on your computer.
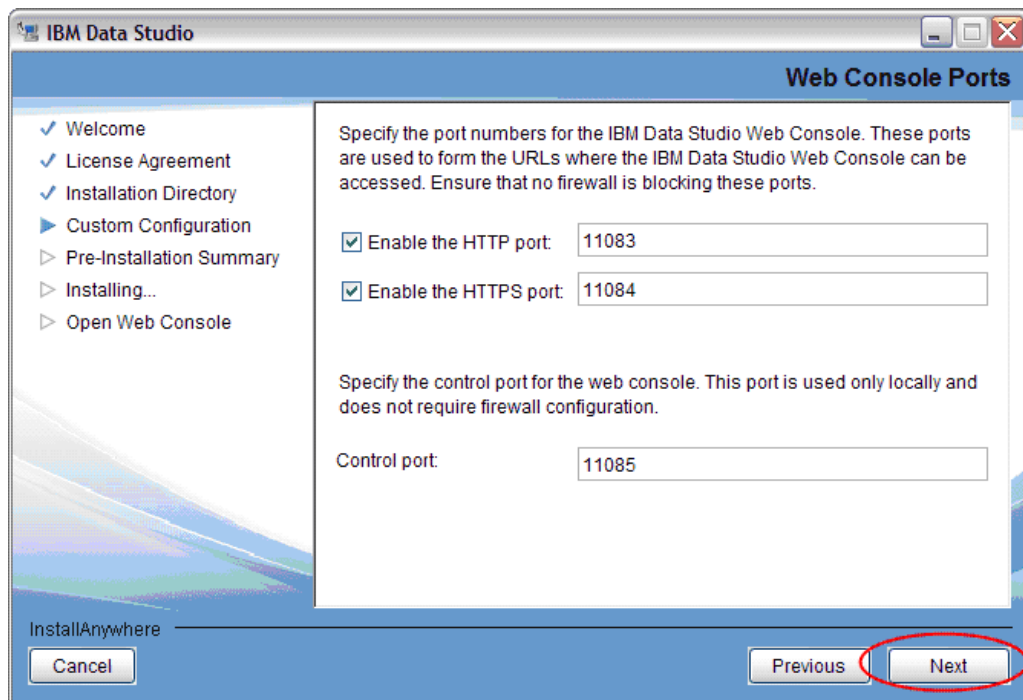
Figure 1.41 – Specify the Data Studio web console ports

8. Verify that the installation information is correct as shown in *Figure 1.42*. If you need to make changes to anything you have entered, click *Previous* to step back through the installation program and correct the entry. Then click *Install* to install Data Studio web console on your computer.
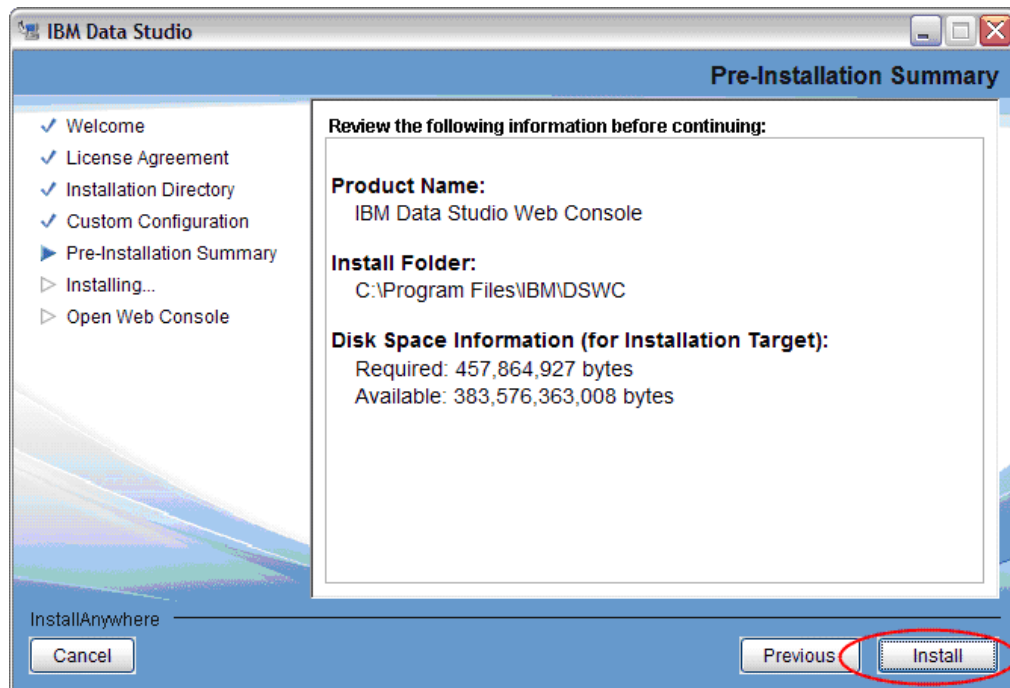
Figure 1.42 – Verify the installation information

9. After the installation program completes you can choose to open the web console to log in to the product locally, as shown in *Figure 1.43*.   Then click *Done* to close the installation program.
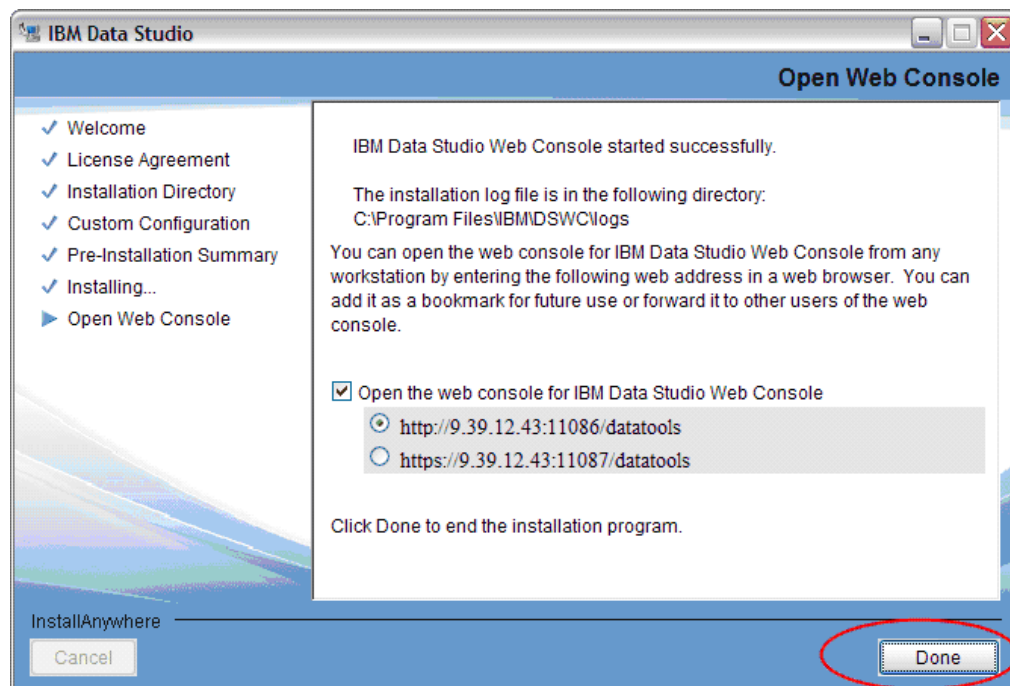
Figure 1.43 – Optionally open the web console

10. Log in to the Data Studio web console with the default administrative user ID and password that you specified during the installation.

1.6.1 Accessing the web console

You can access the web console in the following ways:

- Manually entering the web console URL in a browser
- The Windows Start menu on the Data Studio web console server
- From within the Data Studio full or administration client

To access the web console from this computer or from a computer other than the one you installed the product on, enter the following URL in a web browser:
`http://<IP_address>:<port_number>/datatools/`

> **Note:**
> The following are the default URLs for the Data Studio web console:
> `http://localhost:11083/datatools/`
> `https://localhost:11084/datatools/`

Important: Before opening the web console from your web browser, make sure that you have the Adobe Flash Player plug-in installed on the browser.

On Windows you can also open the web console on the computer on which you installed the web console from the Start menu:
Click *Start > All Programs > IBM Data Studio > IBM Data Studio Web Console V3.1 > Web Console* or *Web Console (Secure)*

If you use the Data Studio full client or administration client, you can also open the Data Studio web console embedded within the workbench. You can use the health pages of the embedded web console to view alerts, applications, utilities, storage, and related information and use the job manager pages to create and manage script-based jobs across your connected databases. You can also schedule scripts as jobs directly from the Data Studio client SQL script editor.

For more information on how to embed the Data Studio web console in Data Studio client, see *Appendix B*.

1.7 Exploring the web console's Task Launcher

When you are logged in to Data Studio web console, you will see the Task Launcher page which lists a number of key tasks and getting-started tasks as shown in *Figure 1.44*.

> **Note:**
> For a complete list of the available web console tasks, click the *Open* menu.
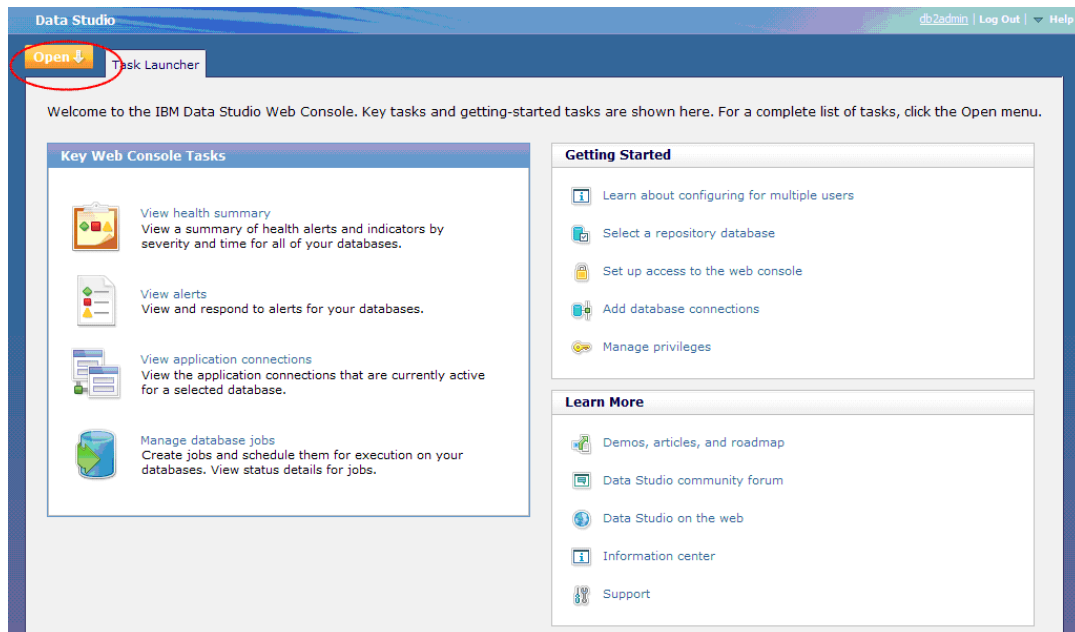
Figure 1.44 - The web console opens on the Task Launcher page

The Task Launcher shows you the most common Data Studio web console tasks, such as:

- Add database connections – Before you can do most tasks in the web console, you must have a connection to the database.

- View health summary - View a summary of health alerts and indicators by severity and time for all of your databases.

- View alerts - View and respond to alerts for your databases.

- Manage database jobs - Create jobs and schedule them for execution on your databases. View status details for jobs.

Important: More information about adding database connections is described in *Chapter 4, Section 4.3.* Depending on your environment, you might have one or more databases running. Once your database connections have been added, you can use the Data Studio web console to begin monitoring the health of these databases (*Chapter 4).* You can also create and schedule scripted jobs on these databases using the job manager (*Chapter 6)*

Note:

This book is written with the assumption that you will use the default administrative user as the only user that will log in to the web console. To add additional users for the web console you must select a repository database, set up access to the repository database, and then grant log in privileges to the web console to the users of the repository database. For more information about configuring Data Studio web console for multiple users, see the *Getting Started* part of the Task Launcher and also *Appendix B*.

1.8 Exercises

In this set of exercises, you will install Data Studio, get comfortable using the Workbench/Eclipse controls, and install the Sample Outdoor Company database.

1. Install Data Studio following the instructions in this chapter.

2. Spend some time getting comfortable with the Data Studio Workbench. For example:

    - Change to the Data perspective.

    - Close the Outline view.

    - Minimize and maximize some of the view windows.

    - Find the menus for each of the views.

    - Reset the Data perspective to its default setting.

3. Optionally, set up the Sample Outdoor Company database using the instructions you can find here:
http://publib.boulder.ibm.com/infocenter/idm/docv3/topic/com.ibm.sampledata.go.doc/topics/config_interactive.html
See *Appendix D* for more information about the Sample Outdoor Company database.

> We'll show you how to create a connection to *GSDB* in the next chapter.

4. Explore the product documentation. For Data Studio, the online information topics are included in the IBM Data Studio Information Center at
http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/index.jsp and shown in *Figure 1.45*. Read the product overview and take the relevant tutorials.
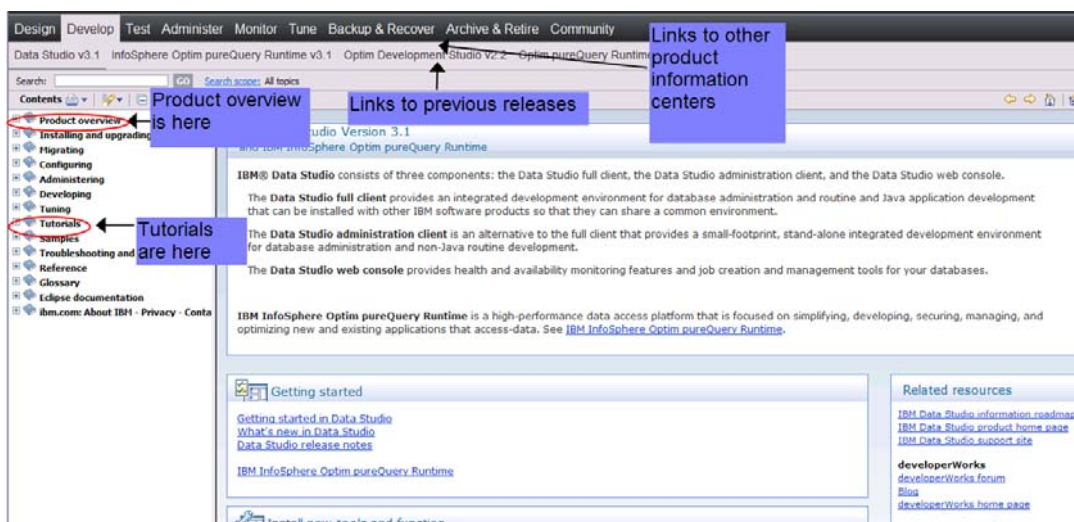
Figure 1.45 – IBM Data Studio Information Center Welcome screen

> **Note:**
>
> As shown in Figure 1.45, the Data Studio information center also includes information about IBM InfoSphere Optim pureQuery Runtime, because there is a development-only license of pureQuery Runtime included for use on the same computer as Data Studio.
>
> In addition, the information center includes links to previous releases of the products (in pale gray bar) and to other products in the InfoSphere Optim Data Lifecycle Tools portfolio (under the task-oriented tabs in the black bar).

In the next set of exercises you will install Data Studio web console, add a database connection to the Sample Outdoors Company database, and use the task launcher to get comfortable using the web console interface.

1. Install Data Studio web console using the instructions in this chapter.

2. If you haven't already done so, set up the Sample Outdoor Company database using the instructions you can find here:

   http://publib.boulder.ibm.com/infocenter/idm/v2r2/topic/com.ibm.sampledata.go.doc/topics/config_interactive.html

3. Explore the Data Studio web console overview documentation. For Data Studio web console, the online information topics are included in the Data Studio information center at
   http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/index.jsp

1.9 Summary

IBM Data Studio provides tools support for database administration and data development tasks for any member of the DB2 family, making it much easier to learn skills for a particular database system and to transfer those skills to other database systems and platforms.

Data Studio is provided at no charge for download and full IBM support is provided for anyone who has a current license of a DB2 data server or Informix server. There is also an active discussion forum at
www.ibm.com/developerworks/forums/forum.jspa?forumID=1086 that can provide informal support.

The Data Studio client is built on the open source Eclipse platform and, if you are using the full client, it can "shell share" (be installed into the same Eclipse instance) with other products that are on the same release of Eclipse, including other InfoSphere® Optim, and Rational® products. This creates a rich and seamless environment in which you can tailor the capabilities of your workbench to the roles you perform on the job. You will learn more about some of these other products and capabilities in *Chapter 11*.

This chapter also covered the details of installing the Data Studio full client. Installation instructions for the administration client are described in *Appendix C*.

We also reviewed how to navigate the Eclipse workbench for Data Studio, including how to open up different perspectives and how to manipulate views in a perspective.

1.10 Review questions

1. What open source platform is the Data Studio client built on?

2. Which IBM products does Data Studio support?

3. What are "perspectives" in an Eclipse-based product such as Data Studio?

4. What is the default perspective after you install the Data Studio client?

5. True or false: Data Studio can be used at no charge with supported databases.

6. Which of the following development capabilities is *not* included in Data Studio?

    A. Development of SQL and Java stored procedures

    B. Development of SQL and Java user-defined functions

    C. .NET development

    D. SQL and XQuery scripting

    E. Web Services development

7. Which of the following database administrative capabilities is provided in Data Studio?

    A. Browse data objects and view their properties

    B. Recover databases

    C. Create, alter, and drop database objects

    D. Authorize users to access database objects

    E. All of the above

8. Which of the following correctly reflects the installable components for the Data Studio product?

    A. Binary and source

    B. Full client, administration client, and web console

    C. C++ and Java

    D. Free and chargeable

    E. None of the above

9. What is the name of the Eclipse view used to browse of the projects that hold SQL scripts, data web services artifacts, and stored procedures?

    A.   Thin Client

    B.   Data Source Explorer

    C.   Data Project Explorer

    D.   Outline

    E.   None of the above

10. In which Eclipse view do results of SQL operations appear?

    A.   Data Source Explorer

    B.   Properties

    C.   Data Project Explorer

    D.   Editor

    E.   None of the above

11. What is the default user ID of the default administrative user that is created for the Data Studio web console when you install the software?

12. True or False: The Data Studio web console can be viewed by itself within a browser or embedded within the Data Studio full or administration client.

13. Which of the following capabilities is not supported from the Data Studio web console:

    A.   Configure and view health alerts for supported databases

    B.   Deploy Data Web Services

    C.   Schedule jobs to run automatically, such as SQL scripts or utilities.

    D.   Configure database connections.

14. What is the default page that opens the first time you log into Data Studio web console?

15. What additional configuration steps are required to start using Data Studio web console after you have added your first database connection?

    A.   Configure alert thresholds for all alert types

    B.   Add a repository database

    C.   Add web console users and configure their privileges

    D.   Configure all Data Studio web console services

    E.   No additional steps are required

2

Chapter 2 – Managing your database environment

Whether you are a developer or DBA, everyone working with or connecting to a database needs to understand the basics of managing their database environment. This chapter discusses how to manage your DB2 database environment using Data Studio.

In this chapter you will learn:

- How to stop and start a DB2 instance
- How to create and connect to a database and navigate through the database.
- How to create tables, views and indexes and deploy them using a change plan
- How to manage users and grant them access to database objects
- How to generate entity-relationship diagrams
- How to work with existing tables to edit data and generate DDL

Note:

This book does not explain basic DB2 concepts, but shows you how to work with them. If you are not familiar with DB2 Express-C, review the *Getting Started with DB2 Express-C* book, which is part of this DB2 on Campus series.

2.1 Managing your database environment: The big picture

As mentioned in *Chapter 1*, Data Studio is the successor of other tools, such as the DB2 Control Center. Control Center was officially deprecated in DB2 9.7, which means it is still supported in DB2 9.7 but will no longer be enhanced and will be removed from a subsequent release of DB2. Data Studio includes support for many DBA tasks, which are shown in *Figure 2.1.*

- Manage databases and instances
- Manage database objects
- Manage security privileges
- Manage space
- Manage and plan for availability
- Manage configurations

Database Administrator

Figures 2.1 – DBAs have a wide range of responsibilities

Figure 2.1 shows the basic tasks that any DBA needs to perform. There are other responsibilities such as complying with data privacy requirements that are beyond the scope of Data Studio but are covered in other IBM solutions. You can read more about this in *Chapter 11*.

This chapter briefly explains some basic things DBAs need to know, such as managing instances and connections, and then goes into managing objects, views and indexes and granting privileges. In the next chapter, we will describe tasks required to support availability and maintenance, such as managing table spaces, updating statistics, importing and exporting data, managing user privileges, and managing buffer pools.

2.1.1 Database Administration perspective

The ***Database Administration perspective***, as the name suggests, focuses on database administration tasks. You may notice that this view is similar in many ways to the Data perspective, and you can do the same tasks in the Data perspective; however, the Database Administration perspective is tailored to suit the needs of DBAs and is laid out to provide a more straightforward user interface for those tasks.

You can switch to the Database Administration perspective by going to *Window -> Open Perspective -> Other* and selecting *Database Administration*. *Figure 2.2* below shows the views in the Database Administration perspective. For more details regarding the perspectives and their views refer to *Chapter 1*.
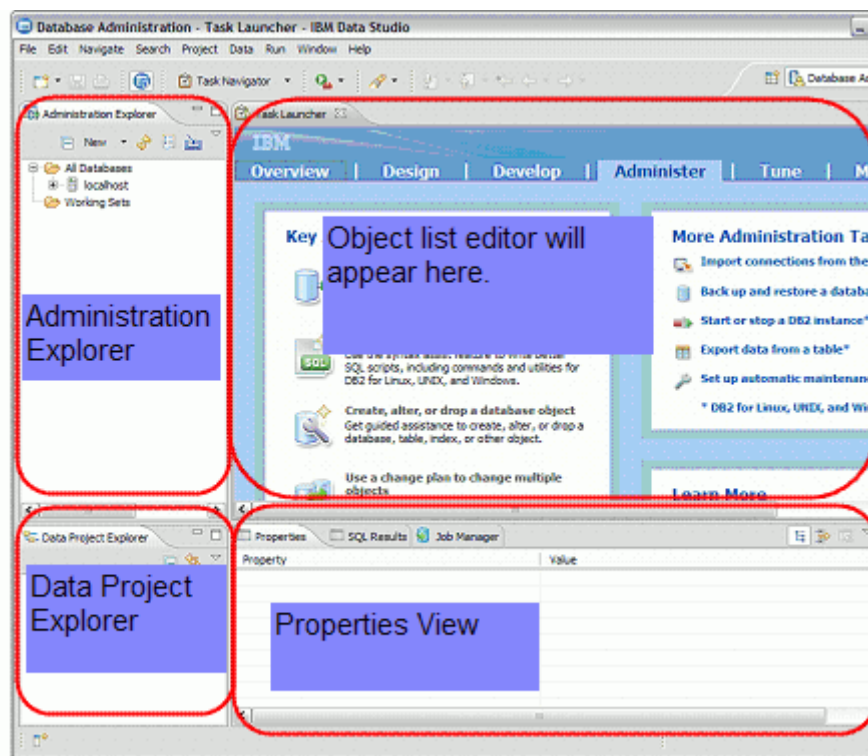
Figure 2.2 – Database Administration perspective

The ***Administration Explorer*** displays an overview of your databases. When Data Studio starts, it reads the local DB2 client catalog and then automatically creates connections to databases on the local machine. You can also create your own connections as explained in the following sections.

When you expand the All Databases folder, the Administration Explorer will display the machines that your DB2 servers run on. Under each machine, the databases are organized into ***Instances***, which will be explained in the following section. Below the instance nodes, you will see connection objects to each database.

The ***Object List Editor*** allows you to sort, and filter lists of database objects such as tables, views, and indexes. When you select a folder of objects in the Administration Explorer, the Object List Editor will display a list of the objects of that type in the database.

The ***Properties View*** displays the attributes of the current selection in Data Studio. When you select a database object in the Object List Editor, the properties view displays the attributes of that object.

The ***Data Project Explorer*** shows the projects created to keep track of your work in Data Studio. Some projects will be created automatically to store your changes to databases. You may also create new projects to organize your own scripts, stored procedures and packages.

2.2 Working with your DB2 databases

In this section you will learn how to create a new database, or work with an existing database. You will also learn how to connect to a database, create connection profiles, and explore database objects.

2.2.1 Creating a new database

To create a new database using Data Studio:

1. In the Administration Explorer, click on the down arrow next to the *New* dropdown to expand the options. Choose the *New Database* item as illustrated in *Figure 2.3.*
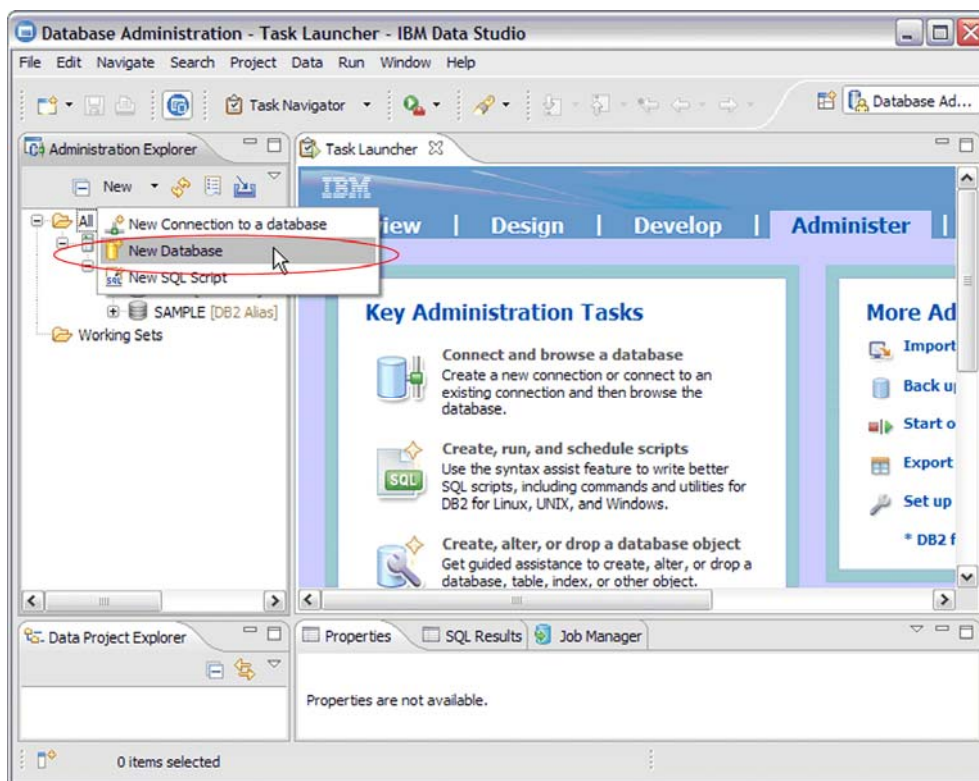
Figure 2.3 – Creating a new database

2. In the *New Database: Specify Instance Details* dialog, fill in the required details for the instance where the database will reside. The instance detail fields are explained in *Table 2.1* below and illustrated in *Figure 2.4*.
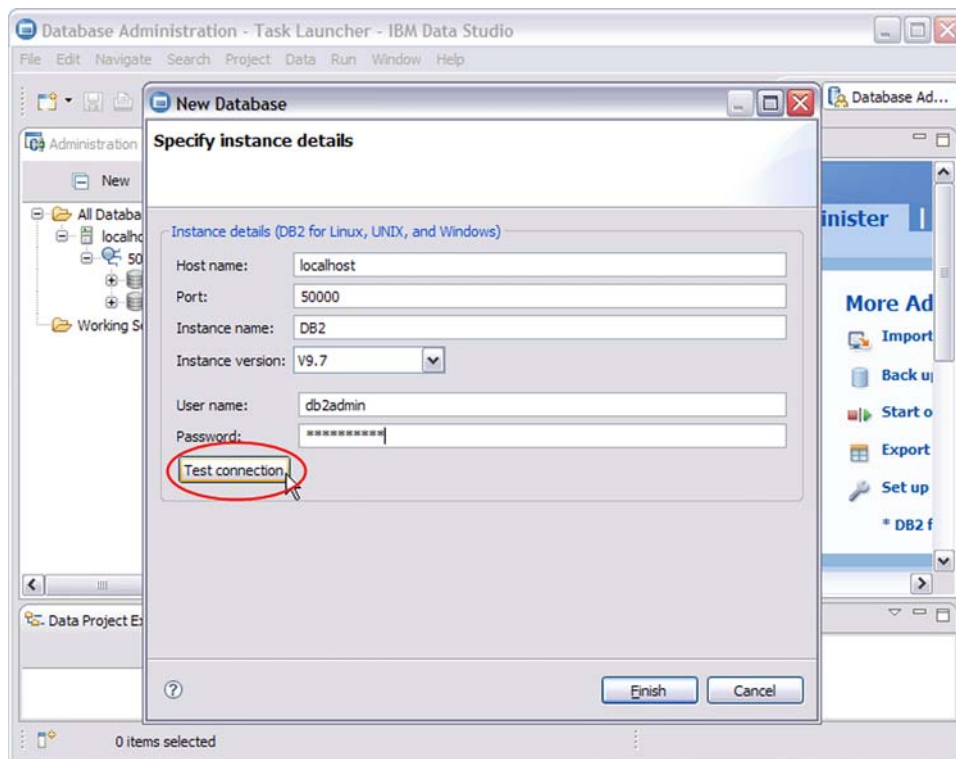
Figure 2.4 – Instance details for a new database

The instance detail fields are explained in *Table 2.1* below.

Field	Description
Host name	The IP address or Host name of the system where the DB2 server is installed. You can use `localhost` for a database on the local machine.
Port	The port number where the instance is listening. By default, DB2 instances use port 50000.
Instance name	The name of the instance where the database will reside. The default instance is **DB2**.
Instance version	The version of DB2 installed for this instance
User name	The name of the user to create the database
Password	The password of the specified user

Table 2.1 – Instance detail fields for a new database

3. After filling in the required details, verify that you can attach to the instance by clicking on *Test Instance* button to make sure that the details are correct (see *Figure 2.4*). If you can successfully attach, you will see an *Instance Connection Succeeded* message in the *New Database - Specify Instance Details* dialog.

4. Click *Finish*. This will open the *New Database* assistant in the editor panel as shown in *Figure 2.5* below.
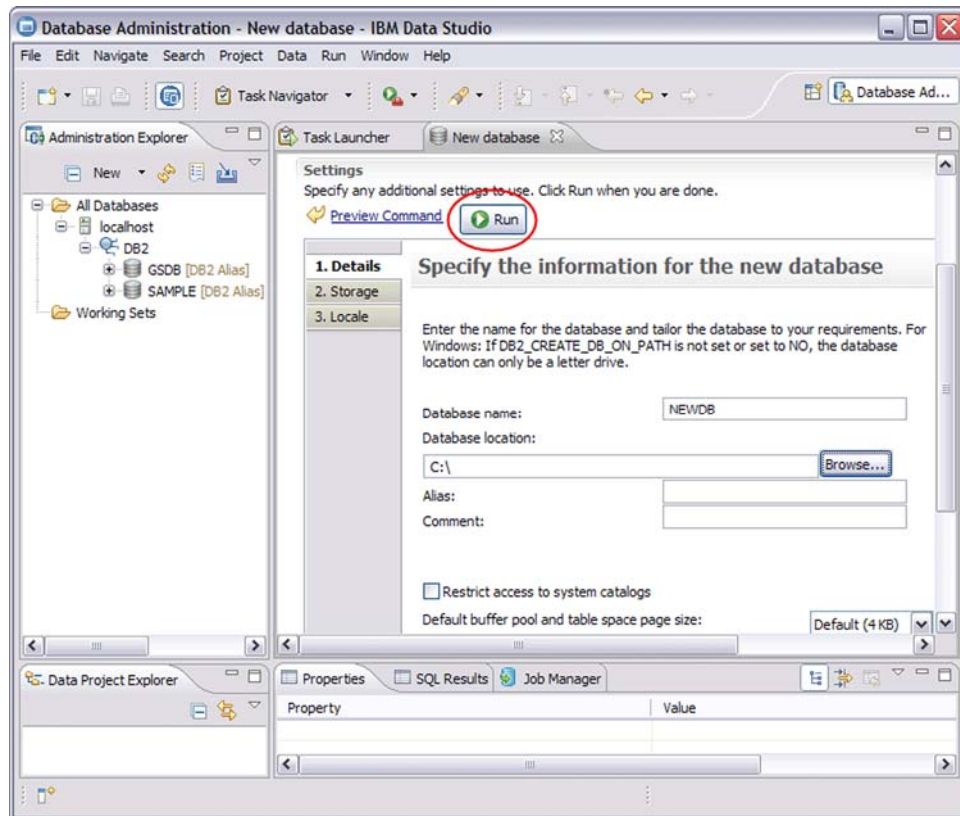
Figure 2.5 – Database creation wizard

In *Figure 2.5*, we used the name *NEWDB* for the database and the *C:* drive for the database location. We used the default values for the rest of the options. We will talk more about them in next chapter. You can see the command that will get executed by clicking on the *Preview Command* link (you may need to scroll down the editor window to see the command).

5. Click on the *Run* button (circled in *Figure 2.5*). This may take a minute or so to complete. On successful execution of the command, you will be able to see *NEWDB* database in the *Administration Explorer*. This is shown in *Figure 2.6*.
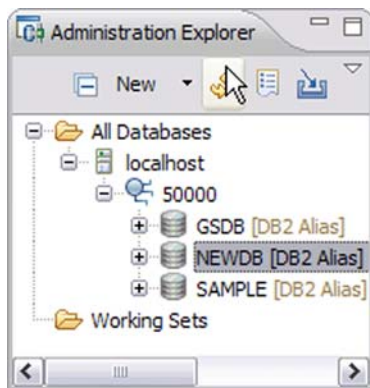
Figure 2.6 – Administration Explorer with new database

2.2.2 Connect to a database in the Administration Explorer

To connect to a database from Data Studio, such as the *NEWDB* created above, ensure it is visible in the *Administration Explorer*. If it is visible, select it, right-click on it and choose *Connect*. The window shown in *Figure 2.7* will open.
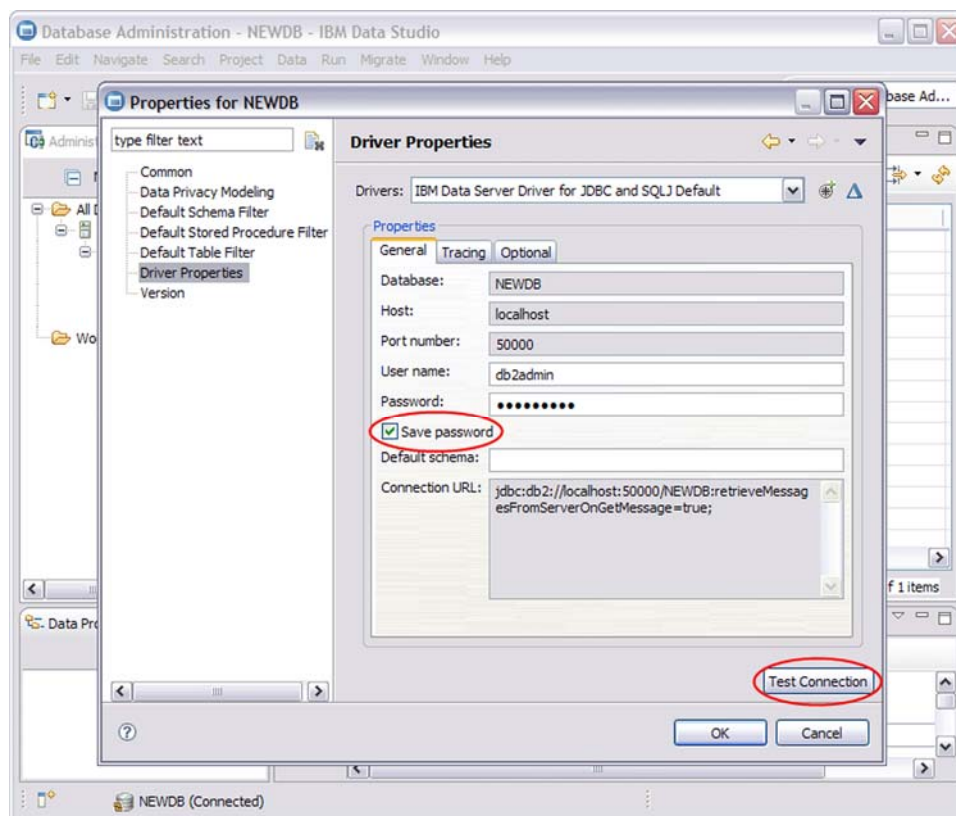
Figure 2.7 – Connecting to a database just created from Data Studio

The database name and the URL will be filled in by default. Enter the user ID and password and click *OK*. You can select *Save password* box to save the password for the future connections.

> **Note**:
>
> If for any reason the window shown in *Figure 2.7* above does not come up automatically, or you get an error message, select the database, right-click on it, and choose *Properties*. Review all the properties and make sure that all the values are correct for your database. If anything is wrong, fix it and try the connection again.

2.2.3 Adding an existing database to the Administration Explorer

By default, the Administration Explorer displays a list of databases on the same machine with Data Studio. If you want to connect to a database on another machine, you can add that database to the Administration Explorer. To do this, click on the *New* dropdown in the Administration Explorer toolbar. Click on the *New Connection to a database* action. In the New Connection window, select *DB2 for Linux, UNIX and Windows*.
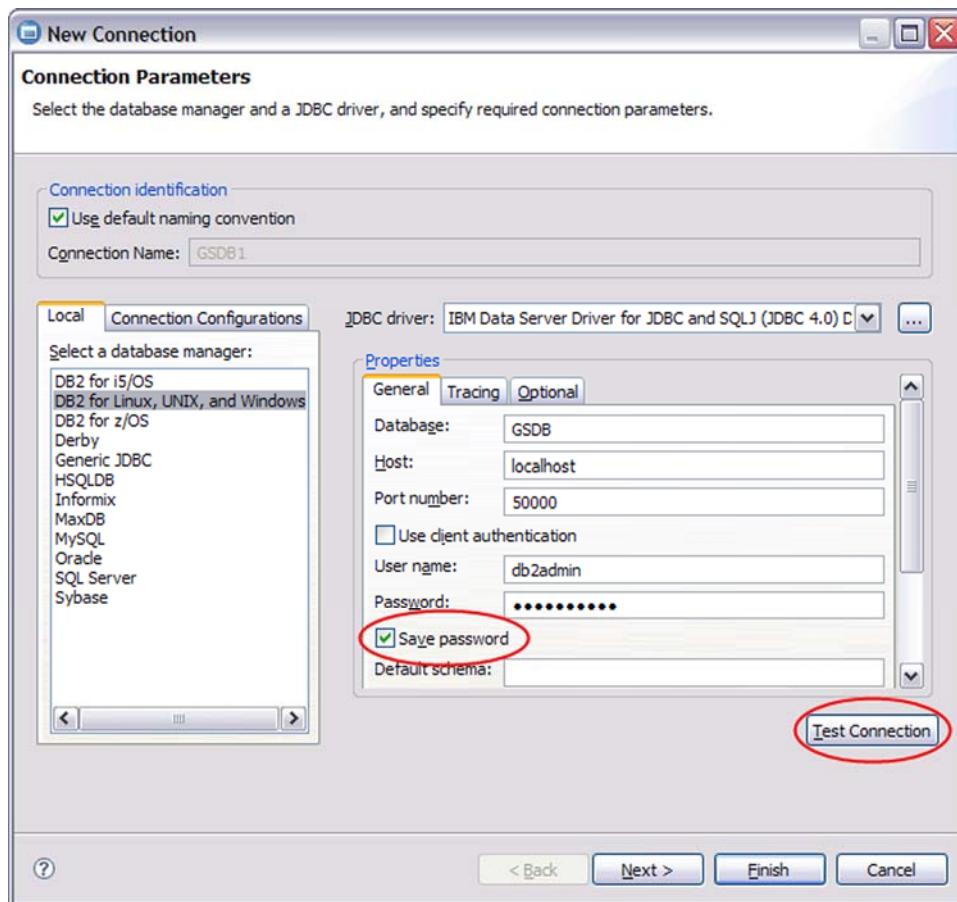
Figure 2.8 – Connection to an existing database

As shown in *Figure 2.8*, fill in the database name, host, port number, user name and password for the database connection. The details in Figure 2.8 would create a connection to the GSDB database, but if you could also connect to a database on another machine if you have one available. Click on the *Test Connection* button on the bottom left side of the panel to test the connection. If the connection is successful, click *Finish*.

2.2.4 Reusing connections with connection profiles

When you created a connection to an existing database in the previous section, you created a ***connection profile***. Connection profiles store the details about how to connect to a database in a file so that other members of your group can share and re-use the same connections. Connection profiles also allow you to save the password and standardize the JDBC driver for each connection.

> **Note:**
>
> For more details on exporting and importing connection profiles, see this topic in the Data Studio information center:

http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/topic/com.ibm.datatools.connection.ui.doc/topics/cdbconn_impexp.html

For a more advanced solution that gives DBAs an efficient way to share connections with others on the team, see the developerWorks article entitled *Managing database connections using the Data Studio web console*, which you can find here:

http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/topic/com.ibm.datatools.connection.ui.doc/topics/cdbconn_impexp.html

2.2.5 Organizing databases with instances

A *DB2 instance* is a logical database manager environment where you catalog databases and set configuration parameters. When you install DB2 on Windows, the installer creates a default instance called `DB2`. On Linux and UNIX, you choose the name of the initial instance during installation. Data Studio has tools to help you manage instances, however, to create or drop instances, you will need to use another tool such as the DB2 Command Window on Windows, or from a Linux/UNIX shell*.*

2.2.6 Stopping and starting instances

To view the instance associated with your current database, in the Administration Explorer, expand the tree under the machine node that your database server runs on as shown in *Figure 2.9*.

Note:

For instances that are automatically created by Data Studio, you might see the port number instead of the actual instance name. This is automatically updated once you connect to any database under that instance*.*

To start or stop the instance, select the instance and right click on it. You will see the options to start/stop it as shown in *Figure 2.9.* You can also perform other operations at the instance level, such as instance configuration, quiescing an instance, and so on. You can explore these options on your own.
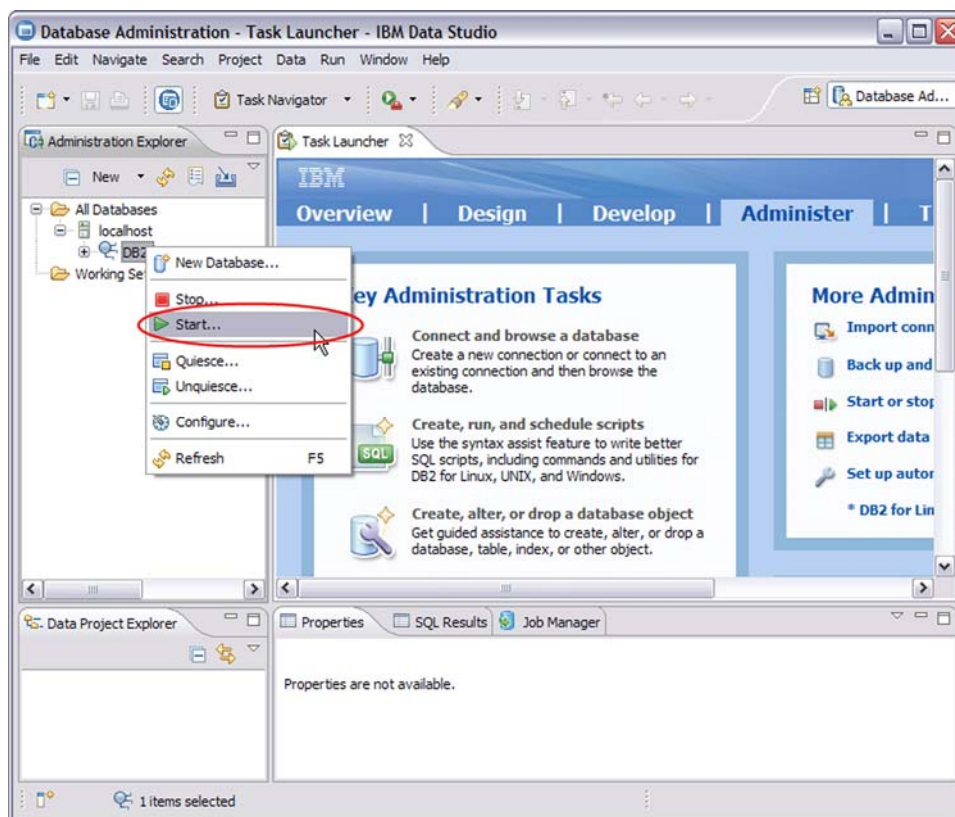
Figure 2.9 – Performing an action such as stopping or starting an instance

2.3 Navigating the database

So far, we used the Administration Explorer to navigate between databases. By expanding a database in the Administration Explorer, we can select a folder of objects, such as schemas, and explore a list of those objects in the Object List Editor. The Object List Editor can narrow down the list of objects using several kinds of filtering.

2.3.1 Filtering the Object List Editor (OLE)

Since a database might have hundreds or thousands of tables, schemas, and other objects, it can be useful to focus the Object List Editor on the objects you are interested in. If you know the name of the object, you can filter the OLE to show objects that have that name.

First, connect to the GSDB database. as described in *Section 2.2.2*. Then expand the list of folders under the GSDB database and select the Schemas folder. A *schema* organizes database objects such as tables and views. When you choose the schemas folder in the Administration Explorer, the Object List Editor will display a list of all of the schemas in the GSDB database.

To filter the list of objects in the Object List Editor, type the first few letters of the object name in the search box shown in *Figure 2.10*. For example, type in "GOS" to filter out the system schemas and display only the schemas that are part of the GSDB data model.
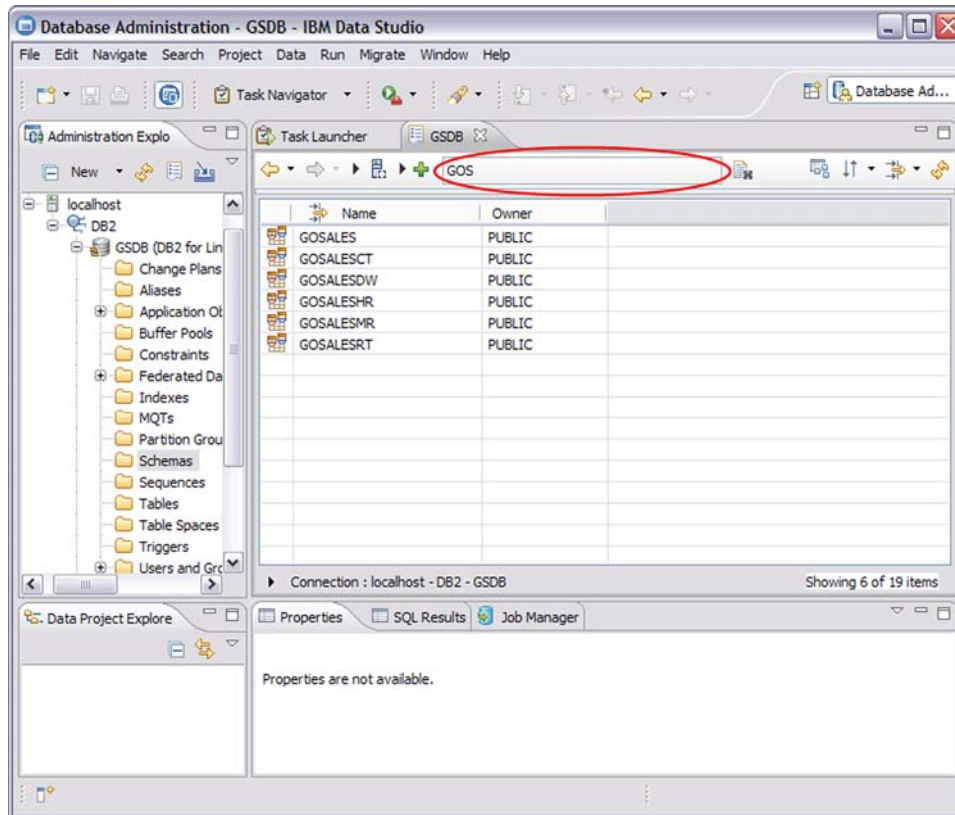
Figure 2.10 – Filtering the Object List Editor

In the search box shown in Figure 2.10, the % character is a wildcard that matches any number of characters in the name. For example, GOSALES% would match the *GOSALES* and *GOSALESCT* schemas. The underscore (_) character will match any one character, so if you type GOSALES_T, the list will display only the GOSALESCT schema.

2.3.2 Exploring objects with the Show menu

Data Studio also provides a way to navigate to other objects that are related to an object displayed in the Object List Editor. When you right-click on an object, the *Show* submenu displays options for navigating to related objects.

To navigate to a list of tables organized underneath the *GOSALES* schema, right click on the *GOSALES* schema in the Object List Editor. In the *Show* submenu, select the *Tables* menu item shown in *Figure 2.11*. The Object List Editor will display the list of tables under the *GOSALES* schema. You can return to the list of schemas by clicking on the back button in the top left of the Object List Editor's toolbar.
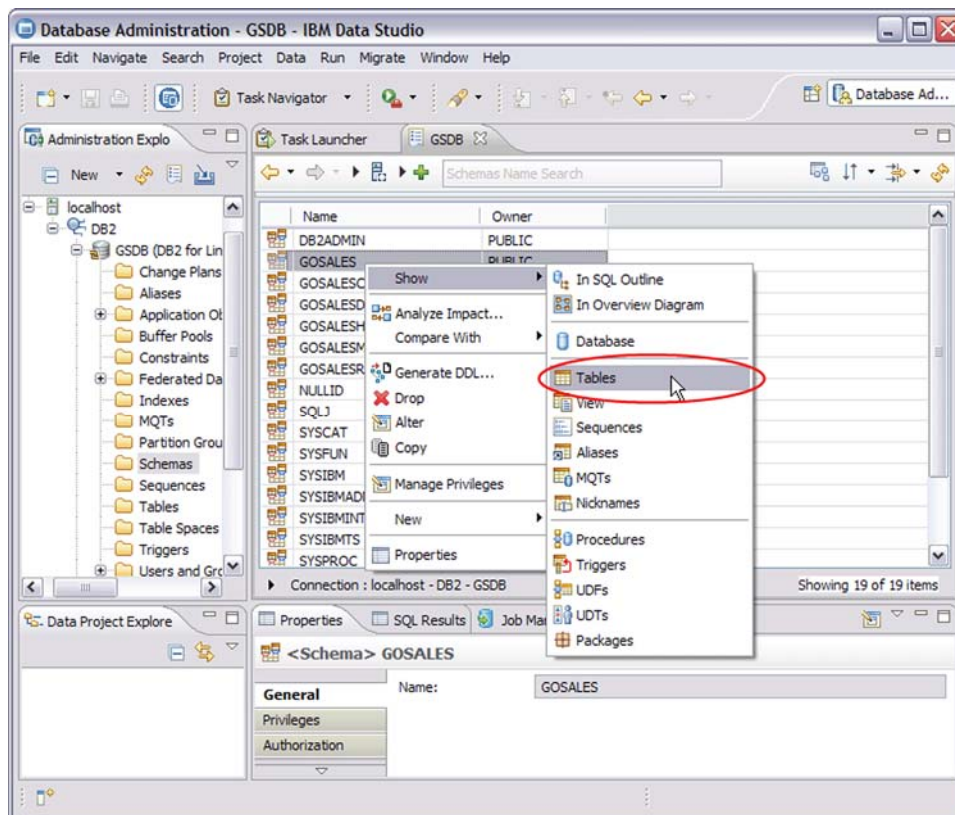
Figure 2.11 – Navigating with the show menu

2.4 Creating database objects

Once you have a database in place and are able to connect to it successfully, you can create database objects such as tables, views, and indexes. This section will show you how to create these objects, and how to create a schema to collect these new objects into a group.

In prior versions of Data Studio, you needed to deploy each change to the database individually. Data Studio 3.1 has a new feature that allows you to create *Change Plans* in order to group multiple related changes together and deploy them in a single step. This section will show you how to create a new schema, a new table, a new index, and a new view in the same change plan. You will then deploy the change plan to create these objects in the database at the same time in *Section 2.4.5*.

2.4.1 Creating schemas

A DB2 schema allows you to organize database objects together and prevent name conflicts when two objects may share the same name. While some of the schemas are already created by the DB2 installation to store system catalog tables, you can create your own schema to group together objects that you create.

To create a schema:

1.   In the Administration Explorer, expand the tree under your database (you may
     need to make sure that you are connected to the database first). Right click on the
     Schema folder and select *Create Schema* as shown in *Figure 2.12.*
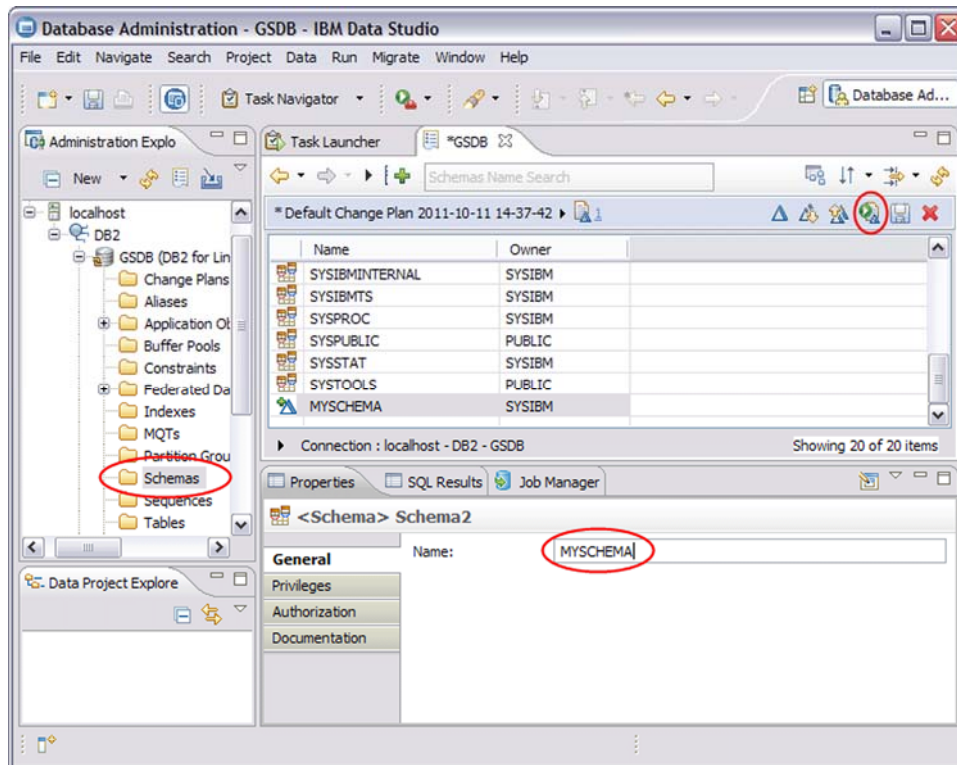
Figure 2.12 – Creating a new schema

2. In the Properties view, fill in the name for the new schema (we used **MYSCHEMA**) and leave the default options in the other fields.

In the Object List Editor, you will see a blue delta ( ) next to the new schema and a blue bar across the top of the Object List Editor, as shown in *Figure 2.12*. This blue planned changes bar appears when you have made changes to the database that have not yet been deployed to the database.

To deploy **MYSCHEMA** to the database:

1. Click on the *Review and Deploy* button ( ) in the blue planned changes bar (circled in *Figure 2.12*).
2. In the Deployment dialog, ensure the *Run* option is selected at the bottom of the dialog as shown in *Figure 2.13*, and then choose *Finish*.
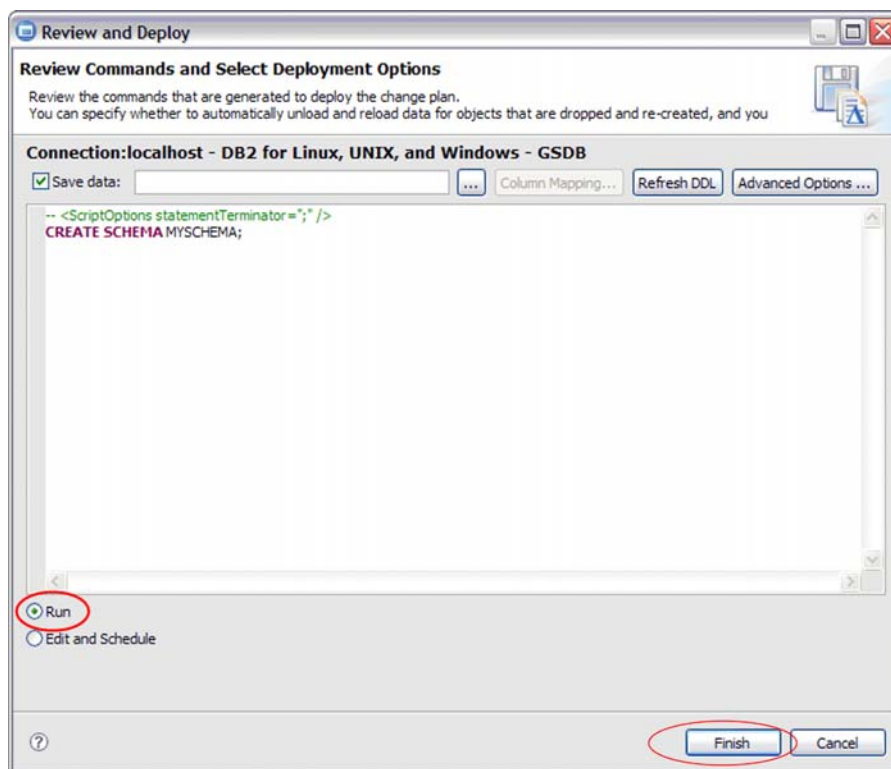
Figure 2.13 – Deploying MYSCHEMA to the database

In the following sections, you will use the same steps to deploy multiple database objects at the same time.

2.4.2 Creating tables

To create a table:

1. Right click on the *Tables* folder and choose *Create Table*. This opens a dialog window where you must select the schema that you want the new table created in. Select **MYSCHEMA** and click *OK*. The Properties view displays the attributes of the new table, as shown in *Figure 2.14*.
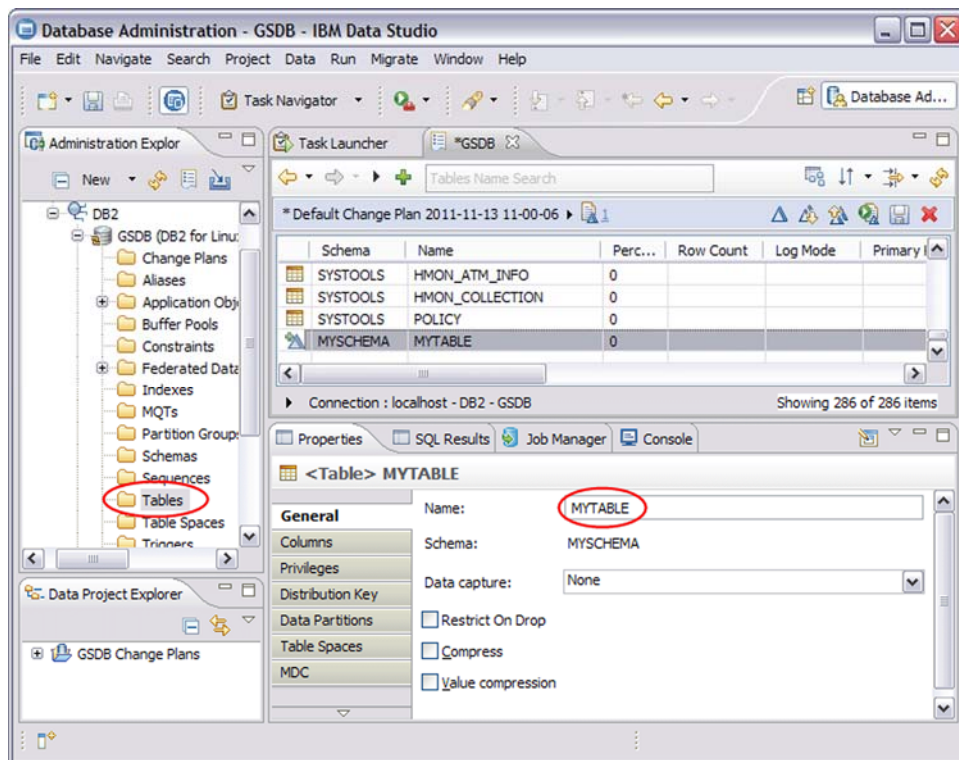
Figure 2.14 – Creating a new table

2. Enter the name of the table in the *General* tab. We used **MYTABLE** in this example.

3. Click on the *Columns* tab to define the columns for this table. Click on the *New* button (♦) to create a new column. This is illustrated in *Figure 2.15*.
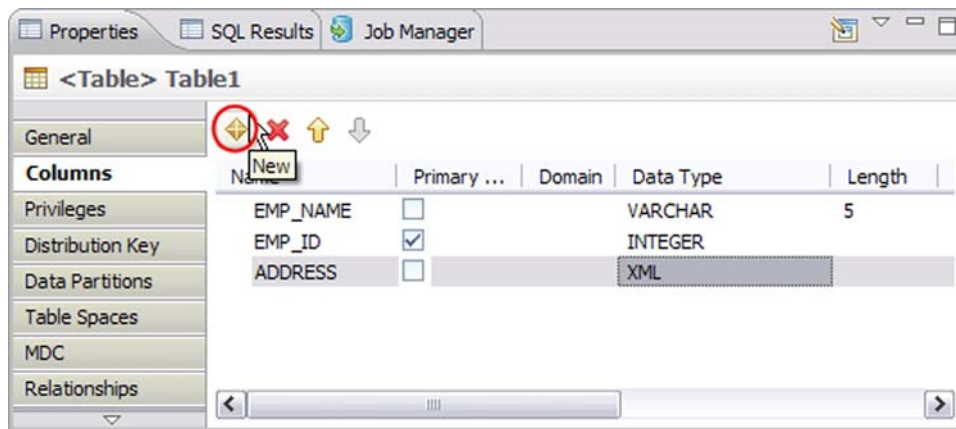
Figure 2.15 – Adding columns to a new table in the Properties View

4. Fill in the details for the column (you may need to resize the object editor window to see all the fields). In the example shown in *Figure 2.15*, we have added three columns:

EMP_NAME, which is of type VARCHAR and length 5.

EMP_ID, which is of type INTEGER and a primary key.

ADDRESS, which is of type XML.

Table 2.2 below describes each of the fields.

Field	Description
Name	Name of the column.
Primary Key	Click this box if you want this column to be the primary key for the table.
Data Type	Data type of the column. Click in the field to activate it for editing and then use the pulldown to see all the data types supported in the drop down menu.
Length	The length of the column. For some of the data types it is fixed and in those cases you cannot edit this.
Scale	Specify the scale for the column type wherever applicable. Again, if it's not applicable to this data type, you won't be able to edit this.
Not Null	Click the box if the column value cannot be null. This check box will automatically be checked for primary key columns, because primary keys are not allowed to be null.
Generated	Click this box if you want the DB2 system to automatically generate the value of this column based on a default value or expression that you provide.

Field	Description
Default Value/Generated Expression	If the *Generated* box is checked, you need to specify a default value or an expression that the DB2 system will evaluate to generate the value of this column whenever a value is not specified in the **INSERT** statement. For example a total salary column can be the sum of basic salary (column name *basicSalary*) and allowances (column name *allowances*). You can specify for *salaryColumn* as Generated expression of *basicSalary + allowances*

Table 2.2 – Column details

In the planned changes bar of the Object List Editor, you will see the count of changed objects update to include the new table. In *Section 2.4.5*, you will deploy the new table to the database with other objects in a change plan.

2.4.3 Creating indexes

To create an index on a column of the table:

1. Right-click on the Indexes folder in the Administration Explorer and choose *Create Index*. This opens a dialog window where you must select the table that you want to create the index for. Under the **MYSCHEMA** node, select the **MYTABLE** table and choose *OK*. The Object Editor will open as shown in *Figure 2.17*.
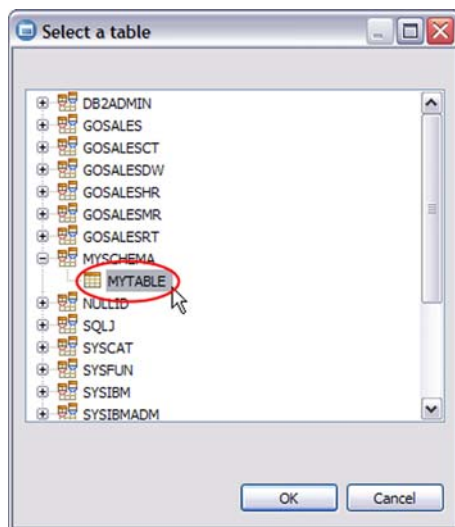
Figure 2.16 – Defining a new index on column(s) of a table

2. On the *General* tab, enter a name for the index (or take the default).

3. On the *Columns* tab, select the columns that will make up the index. To select the
column, click on the ellipses button (…). This is show in *Figure 2.17.*
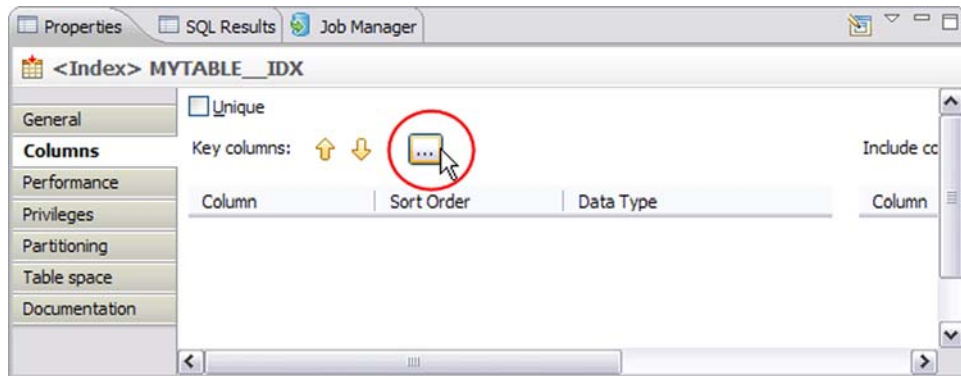
Figure 2.17 – Choosing columns for an index

4. After clicking the ellipses button, you will see a new window pop up which will allow
you to select the columns for the index as shown in *Figure 2.18.* Select the
EMP_NAME column and choose the *OK* button.
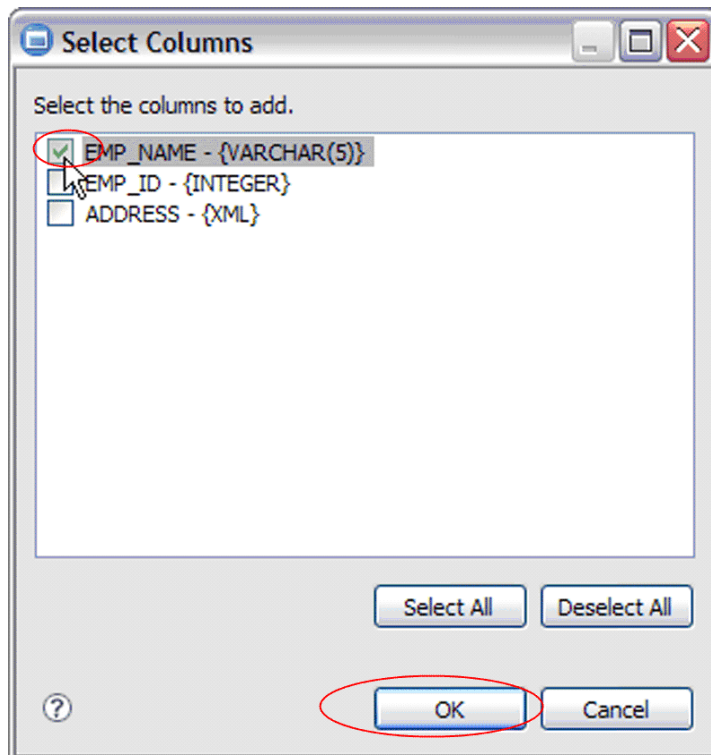
Figure 2.18 – Selecting the columns of the index

In the planned changes bar of the Object List Editor, you will see the count of changed objects update to include the new index. In *Section 2.4.5*, you will deploy the new index to the database with other objects in a change plan.

2.4.4 Creating views

Views provide another way of representing the data that exists in the tables of a database. Views don't contain the data, but they allow you to write **SELECT** and **JOIN** statements to represent data stored in tables without exposing the details of how the data is actually structured. You can use a view to select an entire table, join together multiple tables, or omit specific columns as a way to restrict access to the data.

To create a view over columns in a single table or in multiple tables:

1. Right-click on the *Views* folder in the Administration Explorer and click *Create View*. This opens a new dialog window where you must select the schema that you want the new view created in. Choose **MYSCHEMA** and click *OK*. The Properties View will display the attributes of the new View, as shown in *Figure 2.19*.
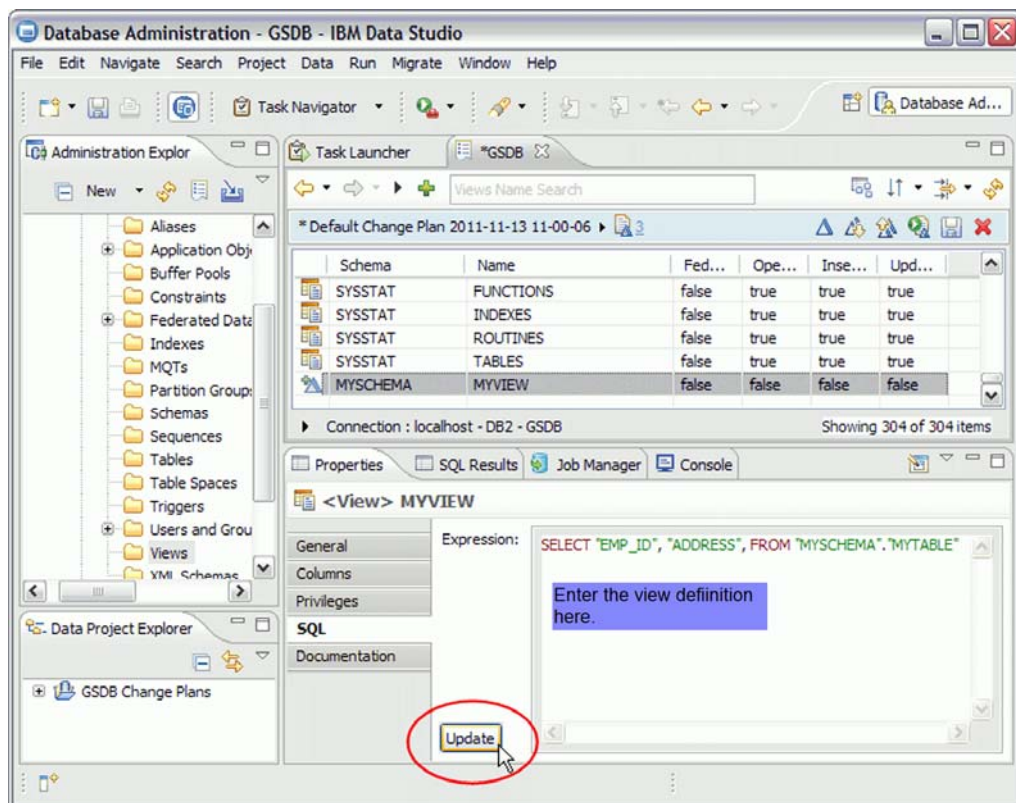
2.19 – Defining a new view

2. In the *General* tab, fill in the name of the view, **MYVIEW**.

3. In the *SQL* tab define the columns for this view using an SQL query. Enter the following query in the *Expression* text box to select the **EMP_ID** and **ADDRESS** columns from **MYTABLE**:

    ```
    SELECT "EMP_ID", "ADDRESS" FROM "MYSCHEMA"."MYTABLE"
    ```

4. Click on *Update* to update your view definition.

In the planned changes bar of the Object List Editor, you will see the count of changed objects update to include the new view. In the next section, you will deploy the new view to the database with other objects in a change plan.

2.4.5 Deploying multiple changes with a change plan

In the previous sections, you created a set of changes to a database in a change plan. A change plan groups multiple related changes together and allows you to deploy them in a single step. This section will show you how to deploy the changes to the database.

To deploy the change plan to the database:

1. In the Object List Editor, click on the number of changes in the Planned Changes bar to display the objects in the change plan. Then, choose the *Review and Deploy* button on the planned changes bar, shown in *Figure 2.20*. Data Studio displays the Review and Deploy window, with a preview of the commands that will be used to deploy the changes to the database.
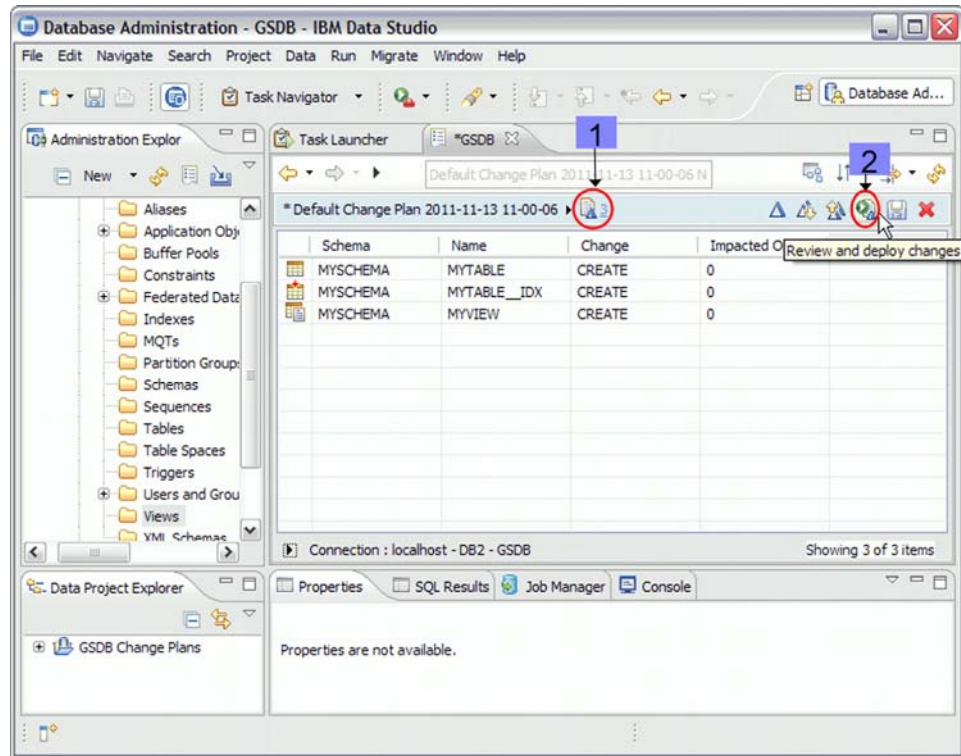
Figure 2.20 – Review and Deploy the changes from the Object List Editor

The Review and Deploy window will appear as shown in *Figure 2.21*. The options in this window determine how the commands will deploy the changes to the database.
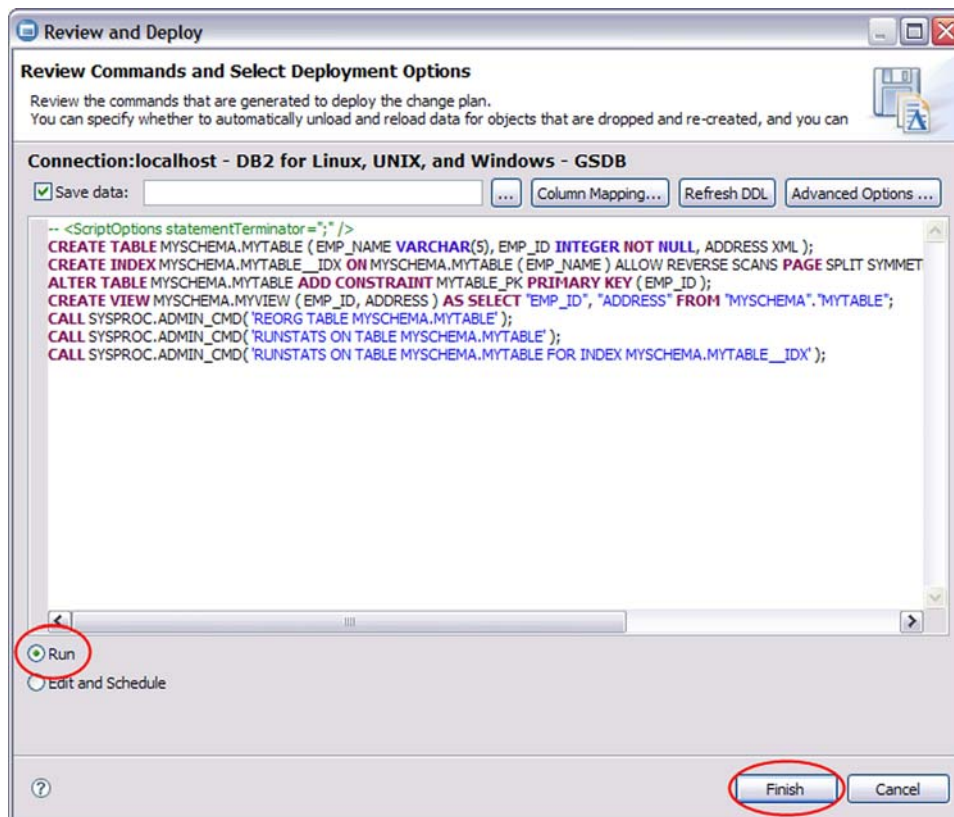
Figure 2.21 – Running the commands from the Review and Deploy window

The *Save data* path defines the location where Data Studio will store any backup files needed to preserve the contents of the database tables. With the save data option enabled, Data Studio will create backup files for dropped tables and changes which require dropping and recreating a table.

To control how Data Studio maps the columns of an existing table to the columns of the changed table, choose the *Column Mapping* button. If you add and remove columns from a table, it is a good idea to review how Data Studio plans to map the columns.

The *Advanced Options* window determines which supporting commands Data Studio will generate with the commands to change the objects. For example, if you do not want Data Studio to generate RUNSTATS commands for changed tables, you can disable the *Generate RUNSTATS commands* checkbox in the *Advanced Options*.

At the bottom of the Review and Deploy window, the *Run* option has Data Studio run the deployment commands when you choose the *Finish* button. To edit the commands in the SQL and XQuery Editor, or to schedule the changes to occur later, select the *Edit and Schedule* option before choosing the *Finish* button.

2. Make sure the *Run* option is selected at the bottom of the Review and Deploy window as shown in *Figure 2.21*. Click *Finish.*

In the SQL Results window, Data Studio will display the progress of the deployment commands, as shown in *Figure 2.22*. When you deploy the commands to the database, the planned changes bar no longer appears in the Object List Editor because your local objects match the objects in the database. You can review past change plans by selecting the *Change Plans* folder in the Administration Explorer.
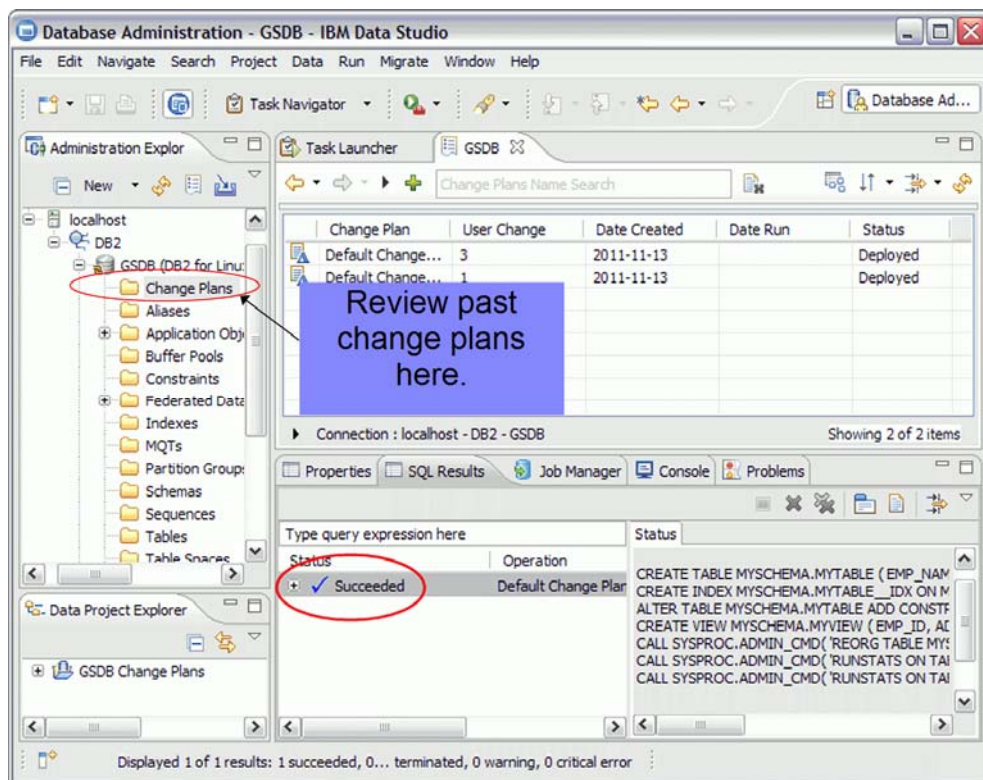
Figure 2.22 – SQL Results shows the progress of the commands

2.4.6 Altering tables

The Object List Editor provides an object editor that can be used to create new or alter existing objects, including tables. In order to make changes to an existing table, right click on the table in the list of tables and choose *Alter*. The Properties View opens the selected table as shown in *Figure 2.23*.
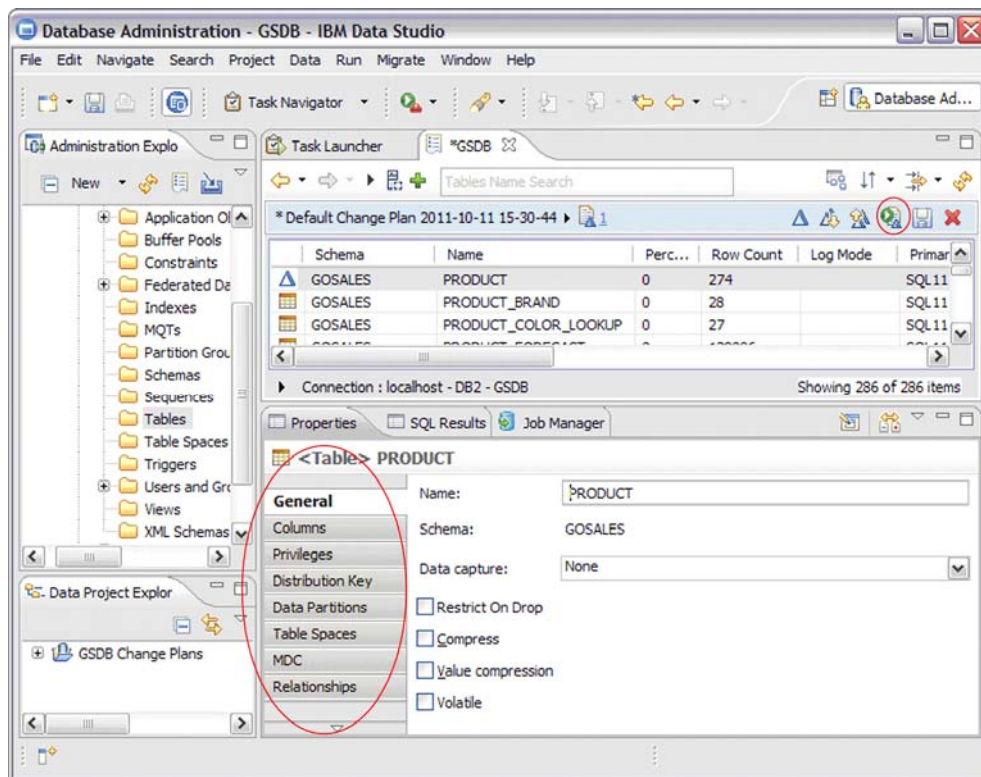
Figure 2.23 – Using the properties view to alter a table

The editor lets you alter several properties of a database table, including its name, compression, privileges, distribution key, data partitions and dimensions, table spaces and table columns. It is also possible to view the table's statistics and relationships using the editor, as well as the list of objects possibly impacted by changes to the table.

Once you have made the changes you want to apply to the table, you can deploy the changes by clicking the Review and Deploy button in the Object List Editor, as described in *Section 2.4.5.*

2.5 Managing database security

Securing the data for unauthorized access is one of the important tasks for the database administrator. You can secure data by granting only the specific privileges that a user needs. This topic will cover adding users and managing privileges on those users.

2.5.1 Creating users

Data Studio can help you to create and manage roles in the same Object List Editor that you used in the previous section to manage database objects. In the following steps, we will create a role to allow developers to create and alter database objects in a single schema.

> **Note:**
>
> Though the Data Studio menus uses the phrase **Create User**, this may incorrectly give you the impression that you can create and store users in a DB2 database by default; this is not the case. User creation is left by default to an external facility such as the operating system, LDAP, active directory, and so on. If you would like to create and store users in the DB2 system you can create your own plug-in to accomplish such mechanism as describe in the article *Develop a security plug-in for DB2 database authentication* which can be found at:
> http://www.ibm.com/developerworks/db2/library/techarticle/dm-0802kligerman/index.html). In this section we use the phrase **Add User** instead.

To add a new user:

1. In the Administration Explorer, expand the *Users and Groups* folder. Right-click on the *Users* folder and choose the *Create User* context menu item.

2. The attributes of the new role will appear in the *Properties* view. In the General Tab, enter the name of the new user, `MYUSER,` in the name field.

3. In the *Privileges* Tab, click on the >> dropdown, and select the *Schema* item as shown in *Figure 2.24* below.
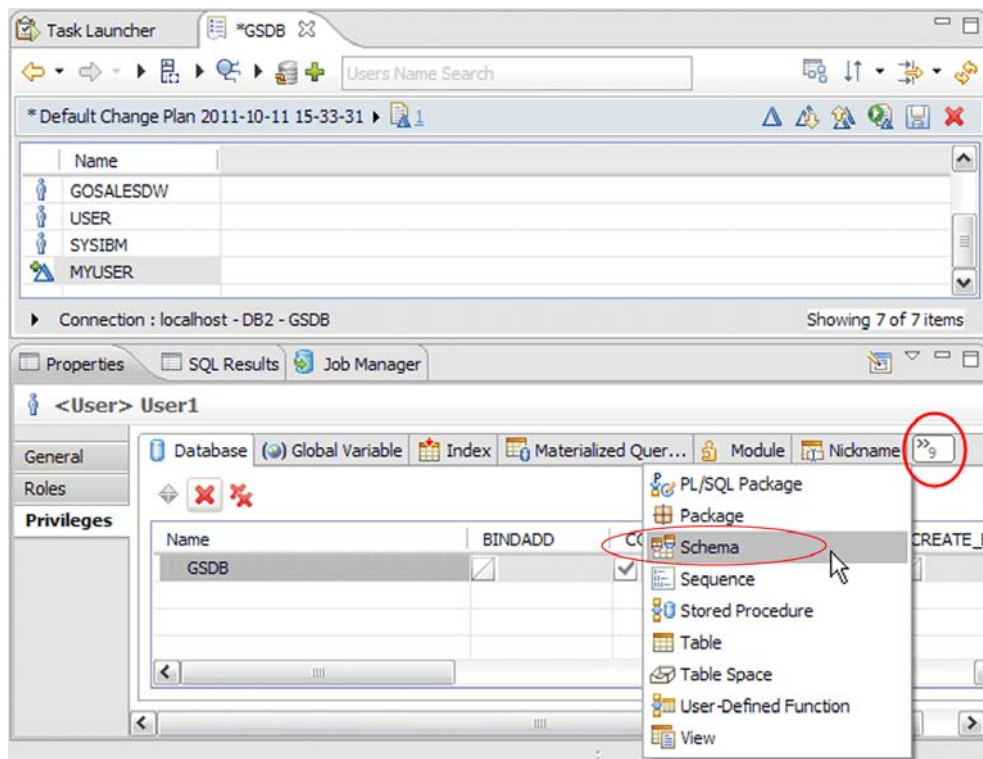
Figure 2.24 – Navigating to schema privileges for the new user

4.   Choose the *Grant New Privilege* button (◈). In the Select a Schema window, choose the *GOSALES* schema, and choose the *OK* button.

5.   Click on the checkboxes in the grid to grant *ALTERIN*, *CREATEIN*, and *DROPIN* authority to *MYUSER*, as shown in *Figure 2.25*.
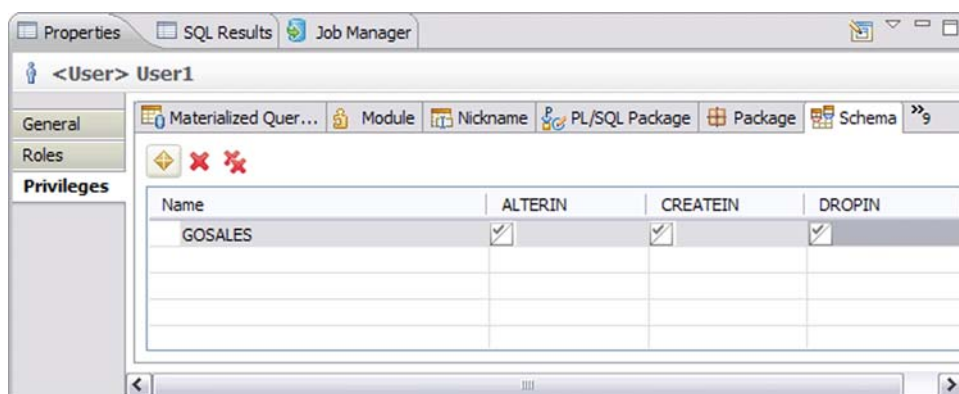
Figure 2.25 – Granting privileges to a user

6.   In the Object List Editor, click on the *Review and Deploy* button and run the DDL to deploy the role to the database, following the same steps as *Section 2.4.5*.

2.5.2 Assigning privileges

Whenever you create an object, you can give the privileges associated with that object to the different users available. *Figure 2.26* shows the privileges tab that appears for table creation in Data Studio. You can use the same *Privileges* tab to change an existing object by right-clicking on the object and selecting the *Manage Privileges* action.
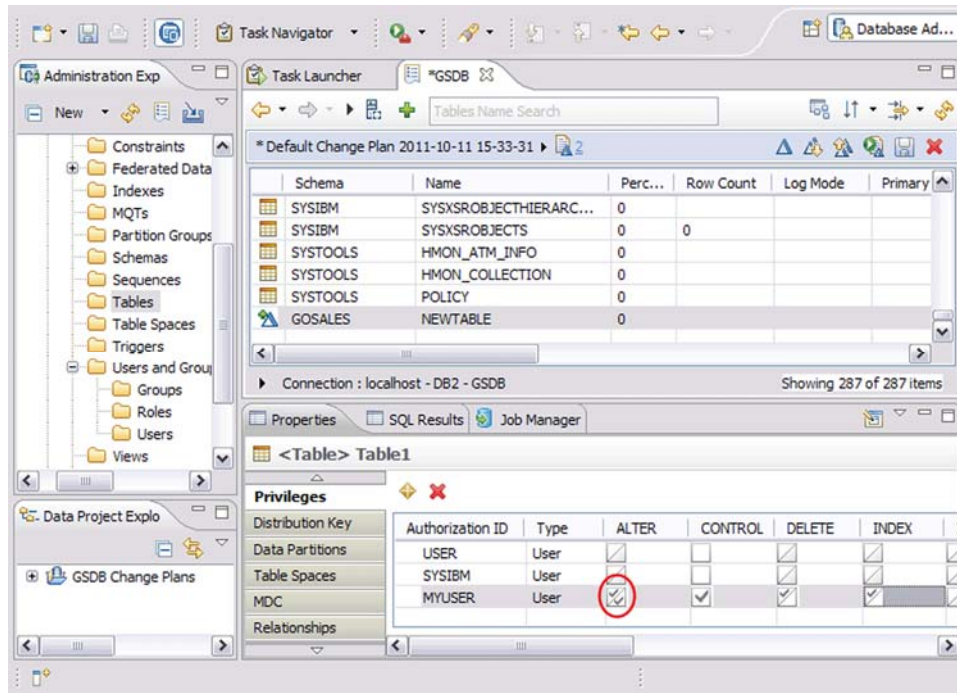
Figure 2.26 – Privileges option while creating a table

If you don't see the user already in the privileges tab, click on *Grant New Privilege*. A new row will appear which will allow you to select the grantee, and then grant privileges to the user on that object.

You grant the user the appropriate privilege by selecting the check boxes in the row of the grid. If you click on a checkbox twice, you can also give the user the authority to grant the privilege to others (*WITH GRANT OPTION*). This authority on the **MYUSER** user is circled in *Figure 2.26*.

For most of the objects that you create in Data Studio, you will find a *Privileges* tab wherever applicable, and you can use the above method to give appropriate privileges to the different users. You can also find the *Privileges* tab on the properties of a user, where it will allow you to grant privileges on multiple objects for that specific user.

Note:

For more information about privileges, see this topic in the DB2 Information Center:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.sec.doc/doc/c0005548.html

2.6 View relationships between objects

When you are learning the structure of an existing database or planning changes to a database, it is useful to find out how the database objects are related to each other. While Data Studio will help you manage dependent objects when you alter or drop an object, it also lets you visualize relationships among objects.

2.6.1 Analyze impact

Before making changes to a database object, it is wise to verify that no dependent objects will become invalid because of your changes. Data Studio can detect dependencies in database objects, so that you can see a summary of the objects affected by the changes. You can find this by right-clicking on a table and choosing the *Analyze Impact* action.

The I*mpact Analysis* window will open, and you can select *Dependent Objects* to see all the objects that are dependent on that object. Click the *OK* button, and the *Model Report* view will open and list the dependent objects as shown in *Figure 2.27*. The Impact Analysis Diagram editor will also display a graph of the dependent objects.
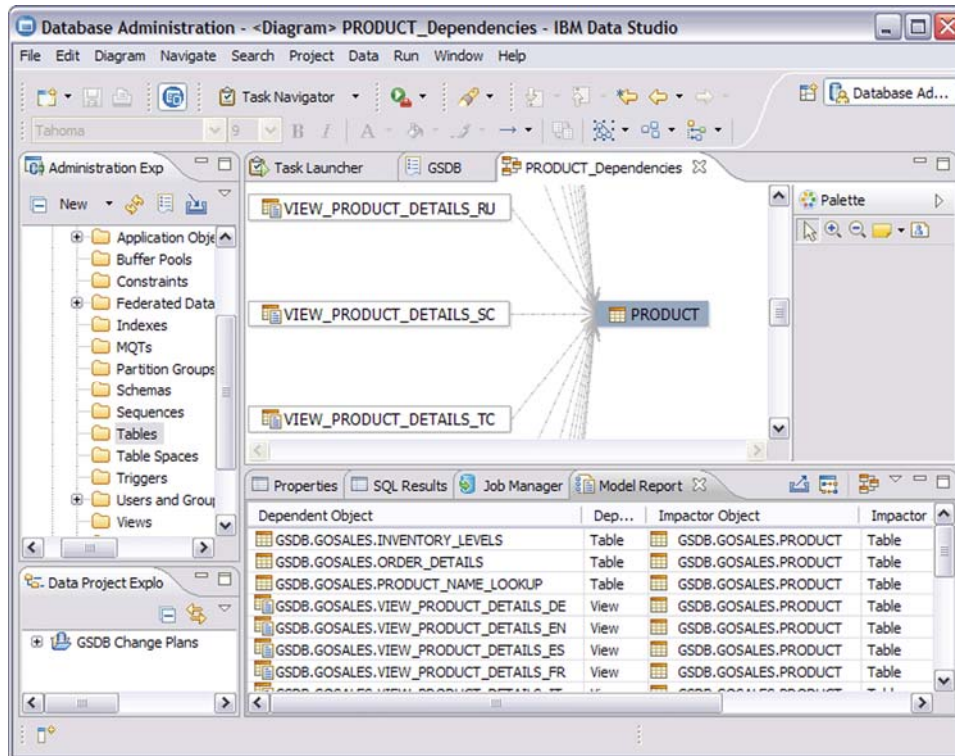
Figure 2.27 – Impacted objects for table PRODUCT

The impact analysis shows that there are several objects impacted by changes in the table **PRODUCT**, including foreign key objects, tables and views. When altering the table **PRODUCT**, Data Studio will help you make sure that the changes would not invalidate the dependent objects, by offering to drop or recreate those objects. However, when a dependent object is composed of SQL, such as a View, you must verify that changes will not invalidate the SQL of the dependent object.

You can experiment with the other options such as contained and recursive objects in the Impact Analysis window to learn more about the relationships between objects. To choose another object, close the Impact Analysis Diagram editor or return to the Object List Editor tab.

2.6.2 Generating an Entity-Relationship diagram

Entity-Relationship (ER) diagrams are a conceptual way to represent data and are commonly used in database modeling. ER diagrams are useful for documenting and analyzing relationships among several entities. For database modeling, it becomes a handy tool to understand the relationships among different tables.

To generate an overview ER diagram in Data Studio, right-click on a database table and choose the *Show* menu, then choose the *In Overview Diagram* item, as shown in *Figure 2.28*.
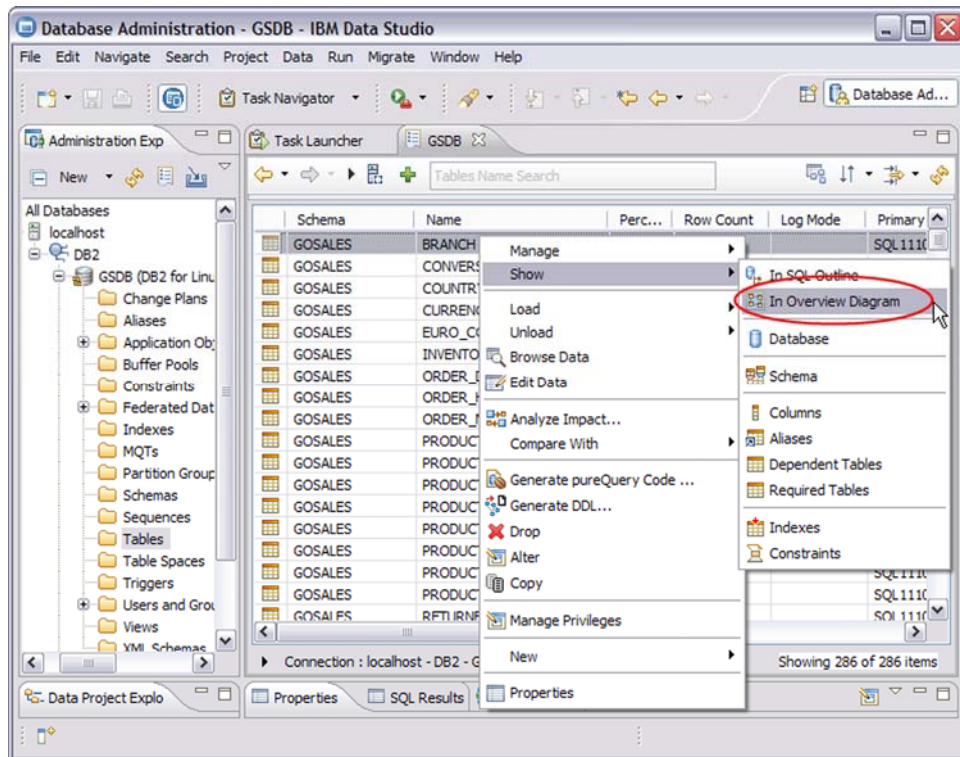
Figure 2.28 – Generating overview ER diagram

The *Overview Diagram Selection* window lets you select which tables you want to include in the overview diagram. Select the tables **PRODUCT, PRODUCT_BRAND, PRODUCT_COLOR_LOOKUP, PRODUCT_LINE, PRODUCT_NAME_LOOKUP, PRODUCT_SIZE_LOOKUP, and PRODUCT_TYPE**, as shown in *Figure 2.29*.
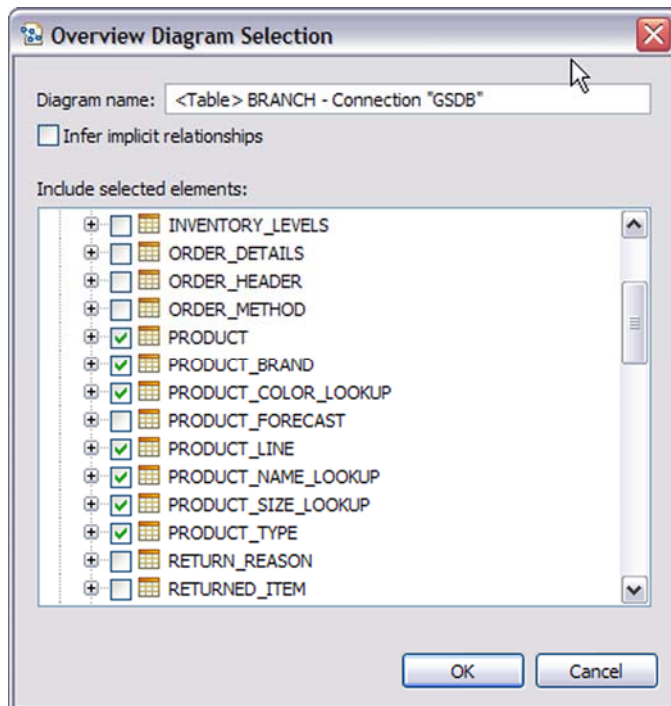
Figure 2.29 – Selecting tables to include in overview diagram

Once you have selected the tables, click *OK* and the overview ER diagram will be generated, as shown in *Figure 2.30.*
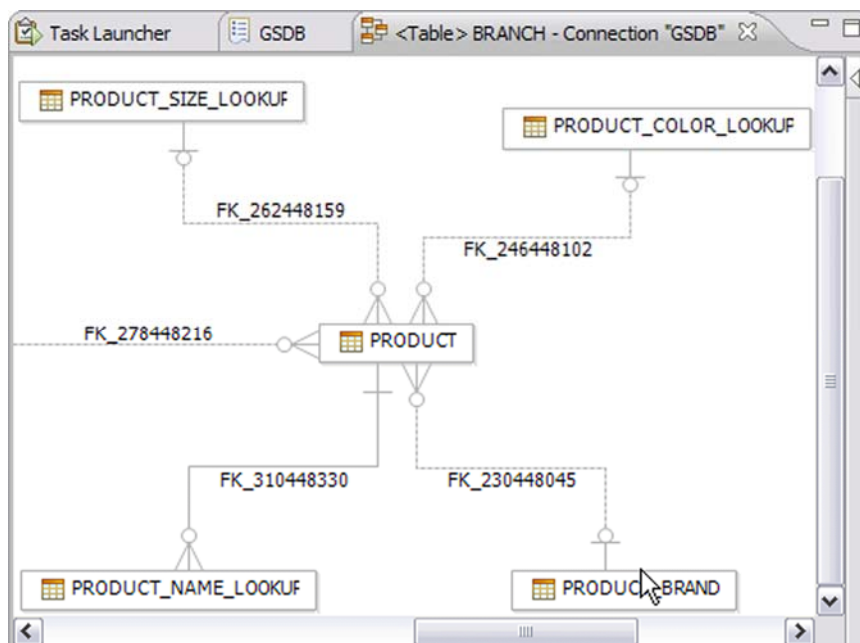
Figure 2.30 – Entity-Relationship diagram for selected tables

Using ER diagrams during development can be crucial to understand the database design and increase your productivity.

Note:

The generation of ER diagrams is to help you visualize an existing database structure. To create logical models using UML or to create physical models that can be used for deployment, you need to extend your environment with a data modeling product such as InfoSphere™ Data Architect. Refer to the ebook *Getting started with InfoSphere Data Architect* for more details at
https://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Getting+started+with+InfoSphere+Data+Architect

---

2.7 Working with existing tables

Data Studio also has tools that assist in maintenance of the existing data, which is stored in tables. In this section we will show how to edit the data contained in a table and look at the DDL to recreate the table. The Run Statistics action will be covered in more detail in *Chapter 3. Figure 2.31 –* shows the actions that can be executed on database tables from the Object List.
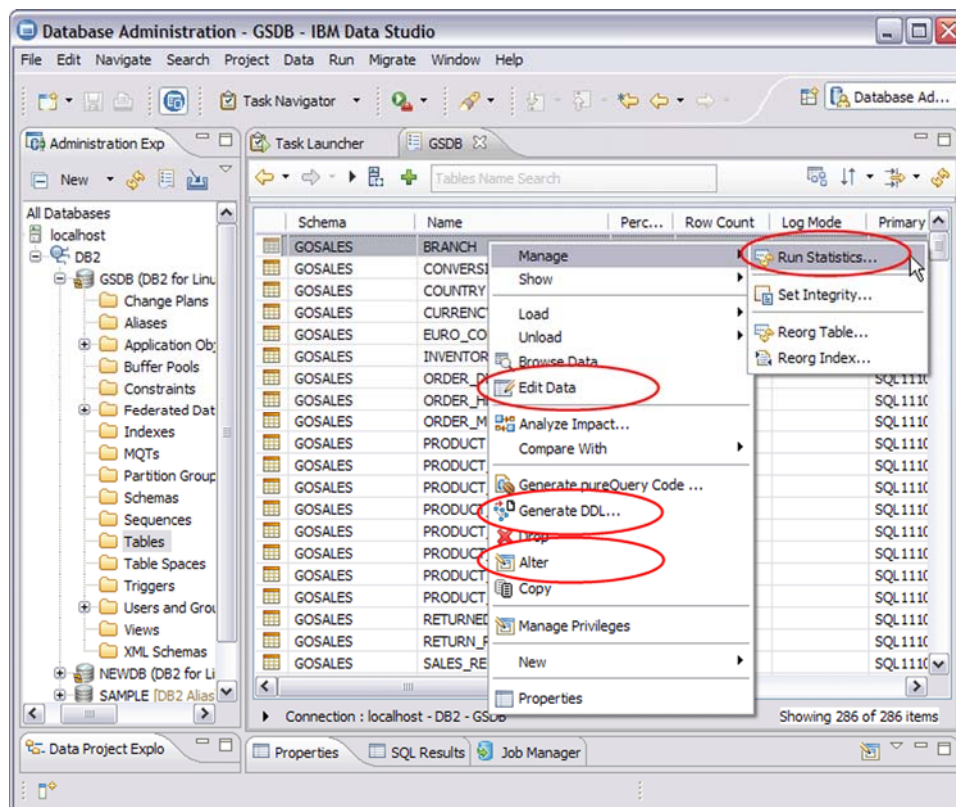
Figure 2.31 – Available actions for database tables

2.7.1 Editing table data

When developing database applications, you will frequently need to update table data so that you can force a complete exposure of your application's code path and induce error conditions to test the application's error handling. With Data Studio, you can edit the table data by right clicking on the table and selecting the *Edit Data* action. The table data editor will open to display the existing data in the table, as shown in *Figure 2.32.*

Figure 2.32 – Editing table data

You can edit the table's contents by selecting a cell and changing its value. Once you have changed a value, cell will be highlighted, and an asterisk in the editor tab will identify it as having unsaved changes. You can commit changes to database by saving the editor changes, either using the shortcut Ctrl+S or by selecting *File -> Save*.

2.7.2 Generate DDL

When there is the need to duplicate a database table, the simplest way is to generate a DDL script that can be executed on the target database. Data Studio provides a *Generate DDL* option available in the right click menu for several types of database objects, including tables.

The *Generate DDL* wizard lets you select several options to be included in the generated DDL, including drop statements, fully qualified and delimited names, dependent object, and so forth. After choosing the options, the generated DDL is displayed and you can choose whether to run this DDL against a database server or simply save it into a local project for later use. For example, *Figure 2.33 –* shows the DDL generated for the table **PRODUCT**.
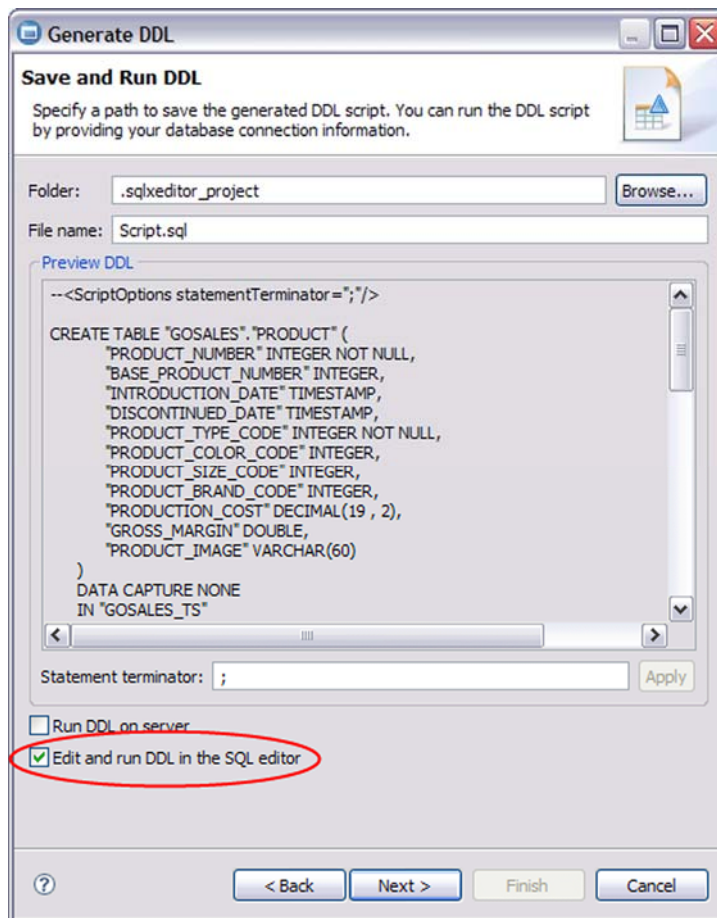
Figure 2.33 – Generating DDL for table PRODUCT

If you want to use the Generate DDL feature to recreate a database object in another schema or database, uncheck the Run DDL on Server option, and check the Edit and run DDL in the SQL Editor option. After you choose Finish from the wizard, you can edit the file to change the schema and name of the object. After changing the name, you can run the script from the SQL Editor or save the DDL to a file.

2.8 Exercises

In this chapter you learned how to start and stop instances, create and connect to databases, create tables, views, and indexes, and how to grant access to users. Here are some exercises to practice what you learned. You can use any of the connection you created in this chapter whenever the name of the database is not mentioned explicitly in the exercise.

Exercise 1: You created a GSDB database in the previous chapter and learned to connect to it using Data Studio in this chapter. Browse the database tree to find out the various schemas in the database and the various objects associated with those schemas.

Exercise 2: Try creating a table with various data types and insert some values into it. Try creating an index of single or multiple columns on this table.

Exercise 3: Try creating a table with a primary key including single or multiple columns. Does an index automatically get created? What columns does it contain?

Exercise 4: Try adding a user and see how many privileges you can give. Browse through all the possible privileges.

Exercise 5: Create a table where the value for a specific column will always be generated by the DB2 based on an expression defined by you.

2.9 Summary

In this chapter you have learned how to create and connect to a database and how to navigate the databases. You also learned how to use change plans to create objects in the database. You also learned how to manage different database objects, and view the relationships between objects. You learned how to add users and grant privileges using Data Studio as well as how to create an overview diagram that shows the relationships among the objects.

2.10 Review questions

1. How can you generate an Entity-Relationship diagram in Data Studio?

2. Related database objects are grouped together in a _____.

3. Why are connection profiles useful?

4. When creating a new user in Data Studio, which tab in the object editor enables you to specify which objects that person has access to?

5. When connecting to a database, which of these is a mandatory parameter?

    A. Port number

    B. Hostname/IP Address

    C. User name/password

    D. All of the above

    E. None of the above

6. In Data Studio, where would you alter the attributes of a table?

    A. SQL Editor

    B.   Properties View

    C.   Routine Editor

    D.   Database Table Editor

    E.   All of the above

7.   While creating a table, when is an index automatically created?

    A.   When you define the primary key

    B.   When you define a NOT NULL constraint for the column

    C.   When the column value is defined as auto-generated

    D.   No index gets created automatically

    E.   All of the above

8.   You can create a view using:

    A.   A full `SELECT` statement on a table

    B.   A `SELECT` statement with specific columns from a table

    C.   A `JOIN` statement for multiple tables

    D.   All of the above

    E.   None of the above

3

Chapter 3 – Maintaining the database

In the previous chapter, you learned how to connect to a database and create various objects. During the course of this chapter, you will learn more about data placement, data movement, and backup and recovery, all of which are critical DBA activities.

In this chapter, you will learn:

- How to manage storage and memory

- How to move data within the database

- How to make a backup of a database and restore from it

3.1 Database maintenance: The big picture

Data Studio provides most of the maintenance functionalities that DBAs need to perform their day-to-day tasks, as shown in *Figure 3.1.*
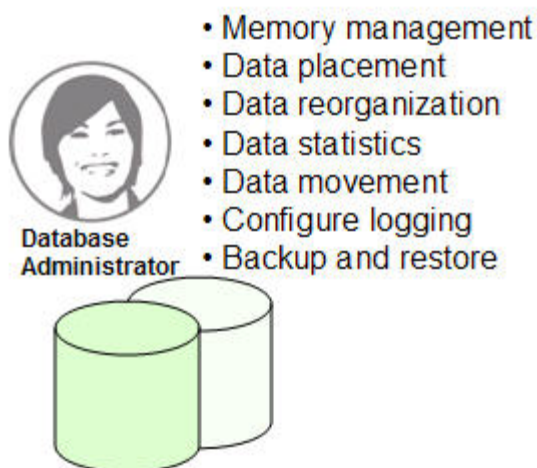
Figures 3.1 – DBAs are responsible for storage and availability

In the previous chapter, we covered basic DBA tasks related to creating and managing database objects such as tables, and indexes. In this chapter, you will learn operational tasks that are critical to keeping the database up and running efficiently and to help prevent

and recover from failures. These tasks become more and more critical as an application moves to a production environment, when database performance and availability become critical success factors for an application. This chapter outlines how some of these tasks can be performed using Data Studio.

> **Note:**
>
> For any advanced capabilities that are not included in Data Studio, refer to related products in *Chapter 11*.

3.2 Managing storage and memory for better performance

A DB2 data server can use the file system or raw devices to store data. The data storage in a DB2 data server is defined using *table spaces*. While you can create tables using a default table space, many DBAs need more control over how data is placed in storage and how to manage the characteristics of that storage and will want to explicitly place tables into specific table spaces, depending on their performance and access requirements.

While executing a query, the DB2 system fetches the required data into main memory for processing. The memory areas used to fetch the data from storage are called *buffer pools*. Again, because the performance requirements of tables and applications can differ, you may wish to have more control over memory usage by different tables.

This section will define these storage and memory areas and explain how you can create and work with them.

3.2.1 Creating and managing table spaces

A *table space* is a logical database object that maps the logical objects like tables, indexes, and so on, to the physical storage memory. It consists of *containers,* which could be an operating system file, directory, or a raw device. In this section, we will concentrate on files and containers. Raw devices, although supported, are not widely used in a typical database due to advances in disk and file system performance.

A DB2 data server can have multiple types of table spaces depending on how the memory is managed and how the containers are defined:

- A *system-managed table space (SMS)* is managed by the operating system and can have directories as its containers.

- A *database-managed table space (DMS)* is managed by the database manager and can have files and raw devices as its containers.

- An *automatic storage table space* is the alternative of SMS and DMS in which DB2 itself manages the containers. You just need to specify the path where the containers should be created and the maximum size that the DB2 server can use for these containers.

A table space can also be categorized based on the type of data it stores–regular, large, or temporary.

3.2.1.1 Creating a table space

To create a table space using Data Studio:

7. Select the *Table Spaces* folder under the database tree, right-click it, and then select the option to create a regular table space, as shown in *Figure 3.2* below.
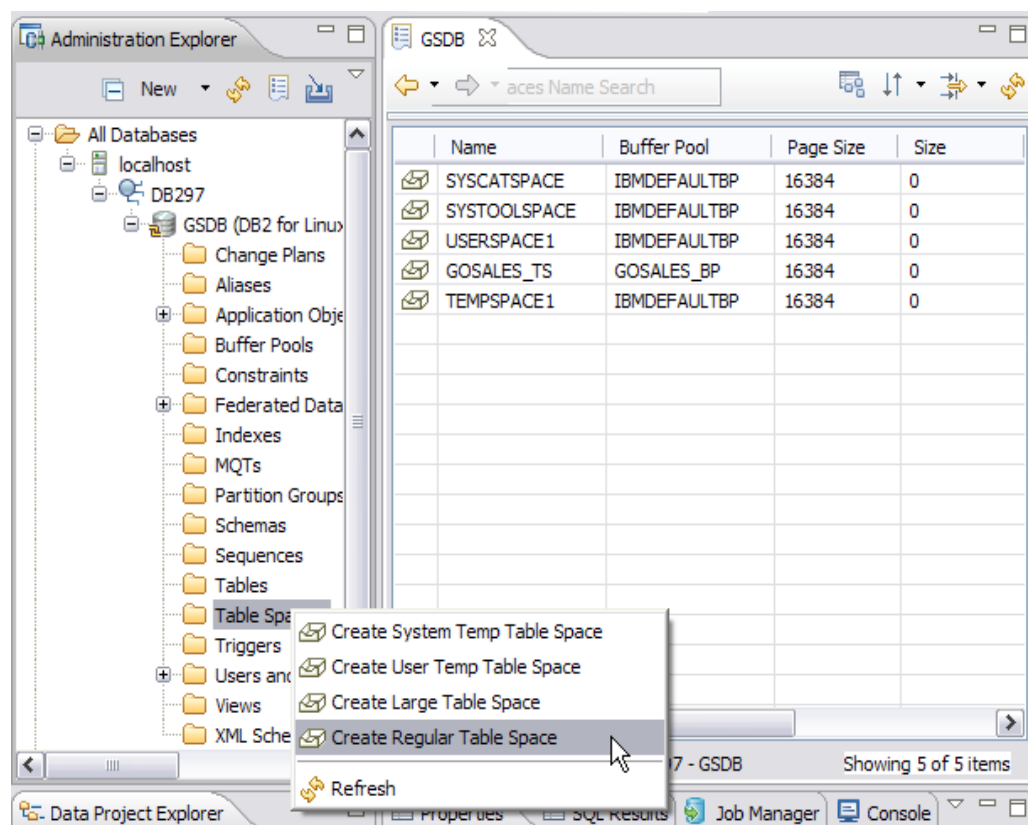
Figure 3.2 – Creating a new table space

2. A new table space with a default name appears in the Object List Editor, as shown in *Figure 3.3*. Click on the new table space name and observe the default properties below in the Properties view. You can provide basic information in the General tab. In the *Management* field, you can select the type of the table space (SMS, DMS, or automatic storage). For this example, select *System Managed* table space, as shown in *Figure 3.3*.
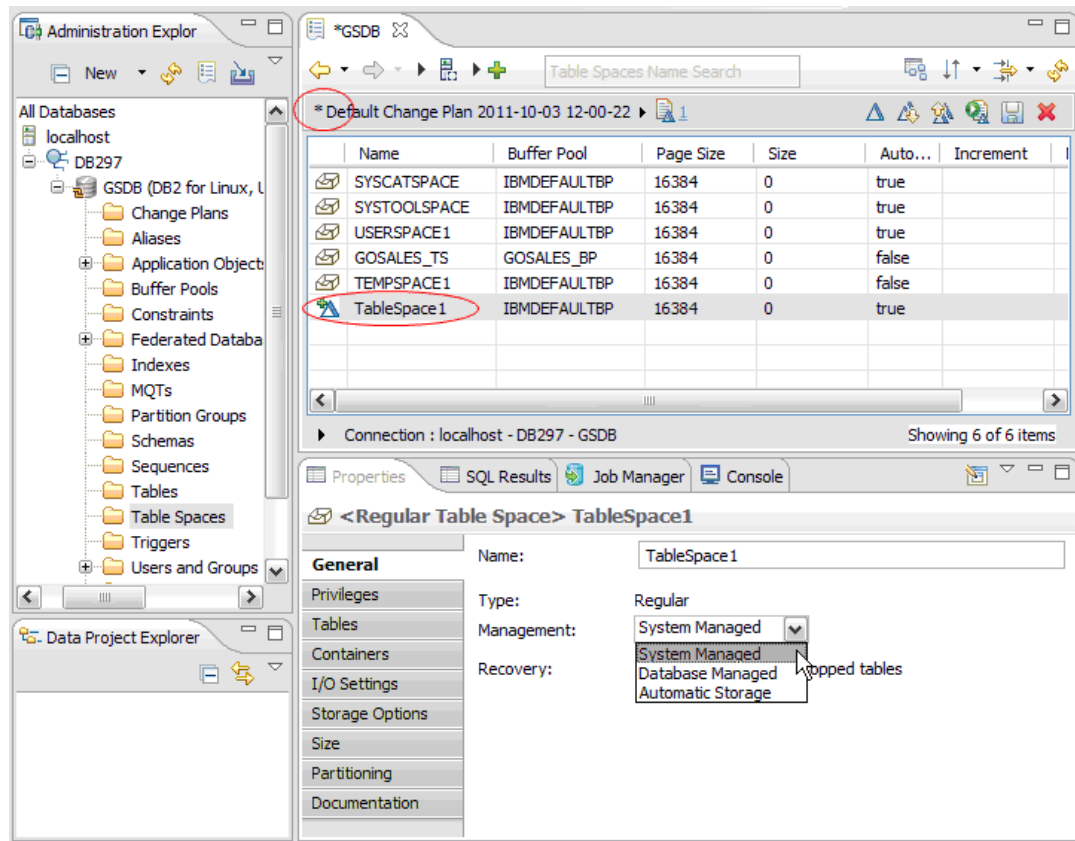
Figure 3.3 – **Defining basic table space properties**

3. In the *Containers* tab, click on the *New* icon. In the *Type* field, the drop down will present you with options for the *System Managed* table space, as shown in *Figure 3.4.*
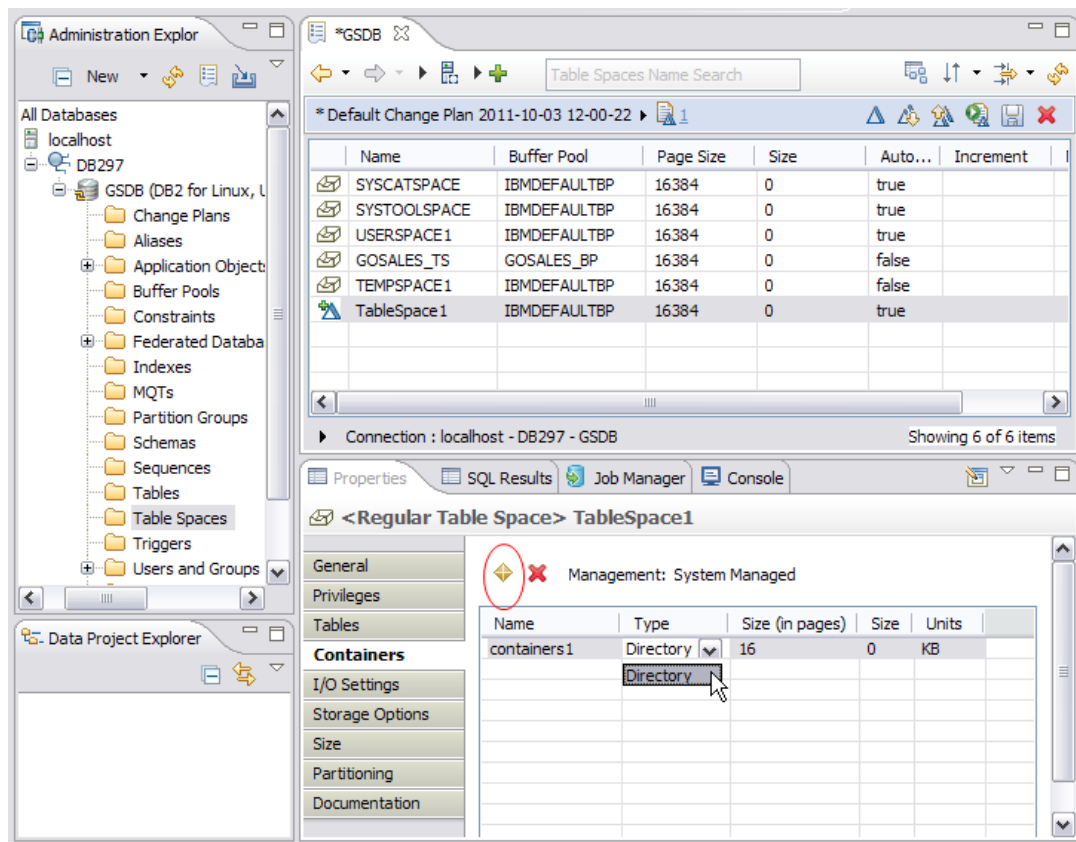
Figure 3.4 – **Defining the containers**

In our example, since we have specified the type as SYSTEM_MANAGED, the only available option is *DIRECTORY*.

If you have specified the type as DATABASE_MANAGED, you will be able to define the container as DEVICE or FILE. However, if you have specified the type as AUTOMATIC_STORAGE, there is no need to define the containers. In this case, you need to define the *Initial size*, *Increase size,* and *the Maximum size* under the *Size* tab. Initial size will be allocated at the time of creation and will be increased by increase size whenever more storage memory is required until the time maximum size limit is reached*.*

4. Optionally, you can move the tables stored in other table spaces to this new table space, by selecting the table names in the *Tables* tab as shown in *Figure 3.5* below. For this example, do not move any tables.
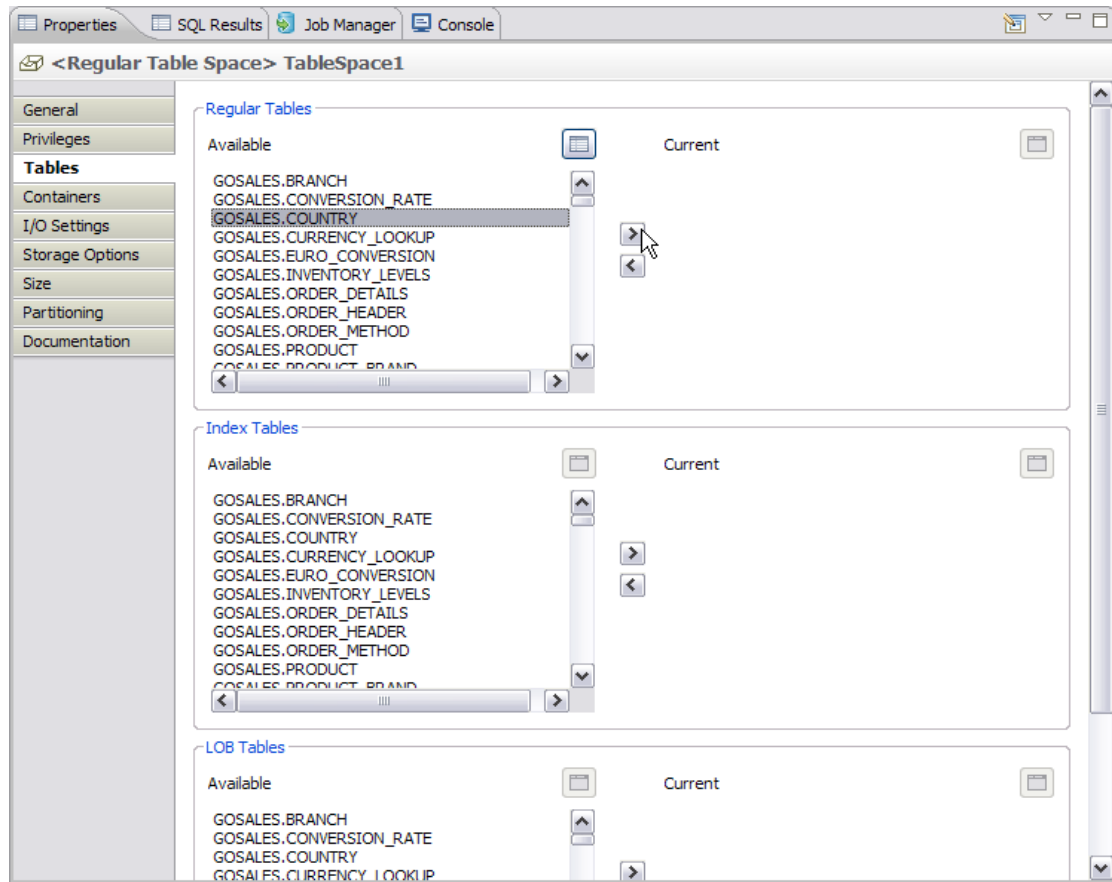
Figure 3.5 – Moving tables to the table space

5.   Now click on the Change Plan link on top of the Object List Editor for an overview
     of the changes, circled in *Figure 3.6*. The number 1 in the change plan link
     indicates that there is one changed object in the current change plan.
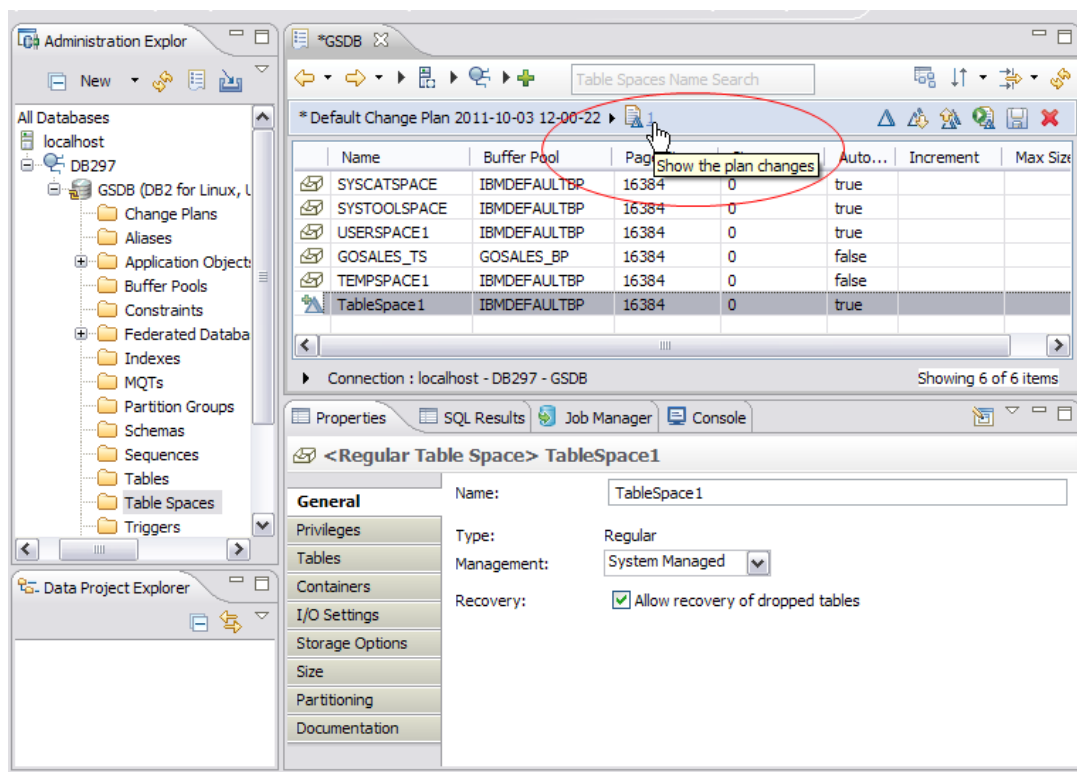
Figure 3.6 – Opening the change plan from Object List Editor

6. From the change plan overview, click on the icon *Review and deploy changes* to review the changes in detail, and deploy them to the database, as shown in *Figure 3.7*.
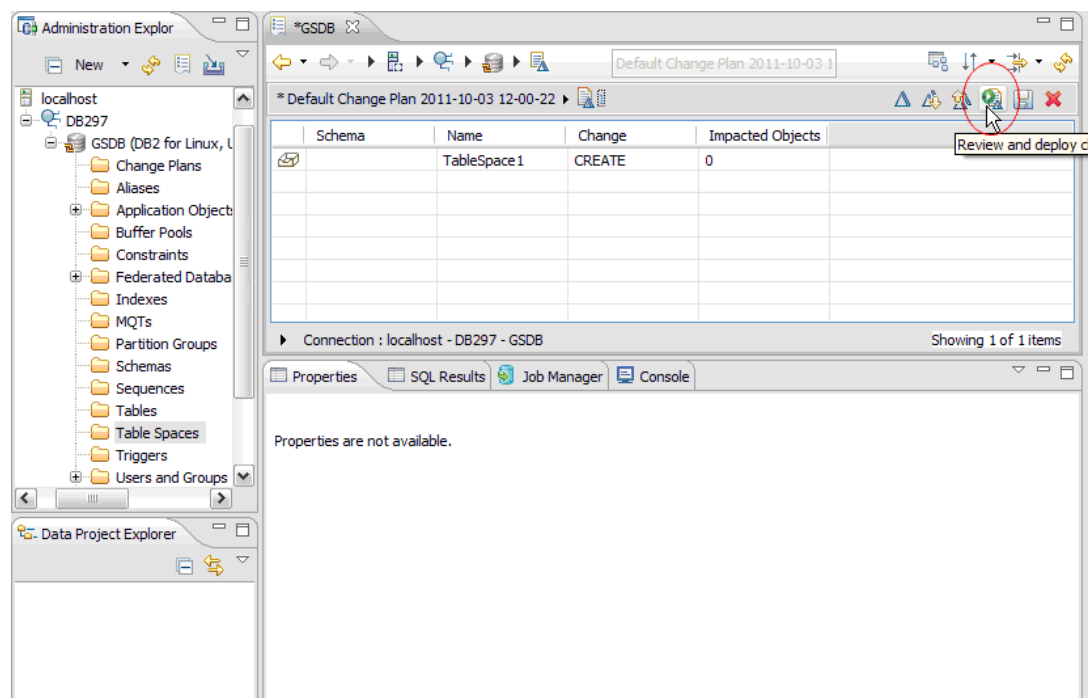
Figure 3.7 – Change plan – Review and Deploy changes option

7. From the *Review and Deploy* dialog, leave the default *Run* selection as is, and click on *Finish* to deploy the commands. This is shown in *Figure 3.8*.
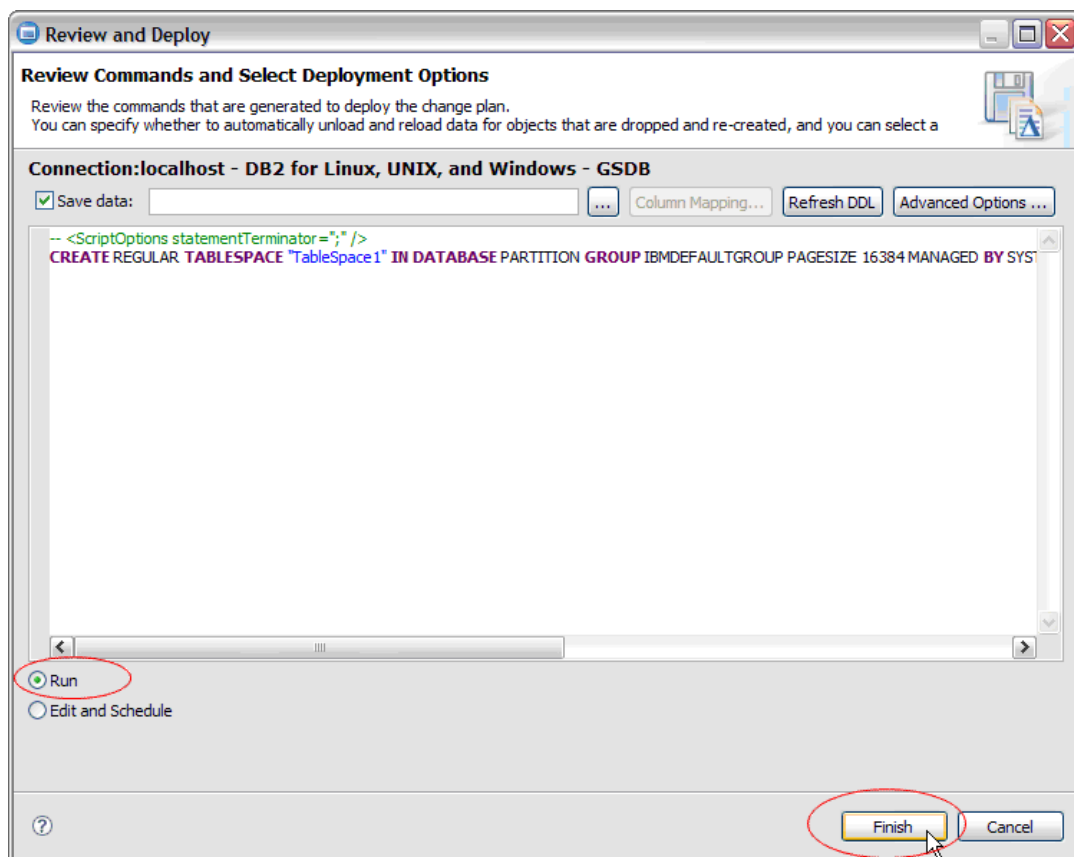
Figure 3.8 – Review and deploy dialog

8. The deployment succeeds and creates a new regular SMS table space in the database. *Figure 3.9* shows the *SQL Results* view that displays the result of every command execution. The *Status* tab on the right displays the command syntax, any command output or error, and the execution time of the command.
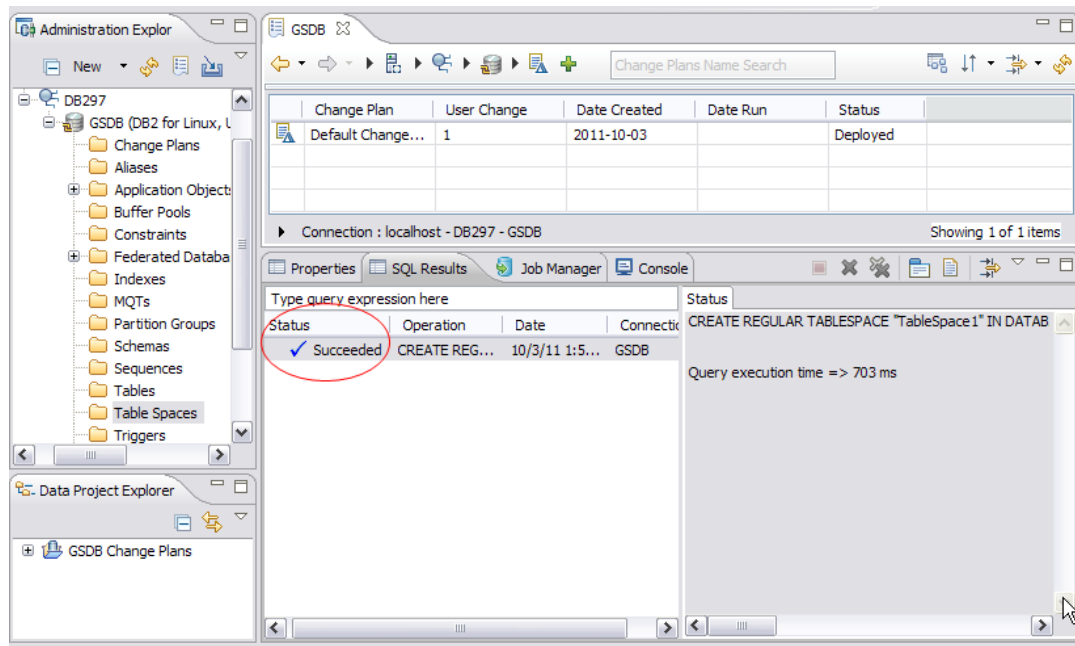
Figure 3.9 – SQL Results view displays the results of deployment

3.2.1.2 Use the new table space

Now you have a new table space that you can associate with tables, indexes, etc. that you create. This association tells the DB2 data server to use this table space for physical storage of those objects. To associate any existing objects with this new table space, you can perform an alter operation on those objects.

3.2.1.3 Drop a table space

You can drop a table space by selecting the table space from the Object List Editor. However, exercise caution while doing this operation as it deletes any associated tables and data as well. To drop a table space, select the *Table Spaces* folder in the *Administration Explorer*, right click on the table space in Object List Editor on the right, and select *Drop*. This is shown in F*igure 3.10.* To deploy the drop on the database, follow the deployment steps from *Section 2.4.5* of *Chapter 2*.
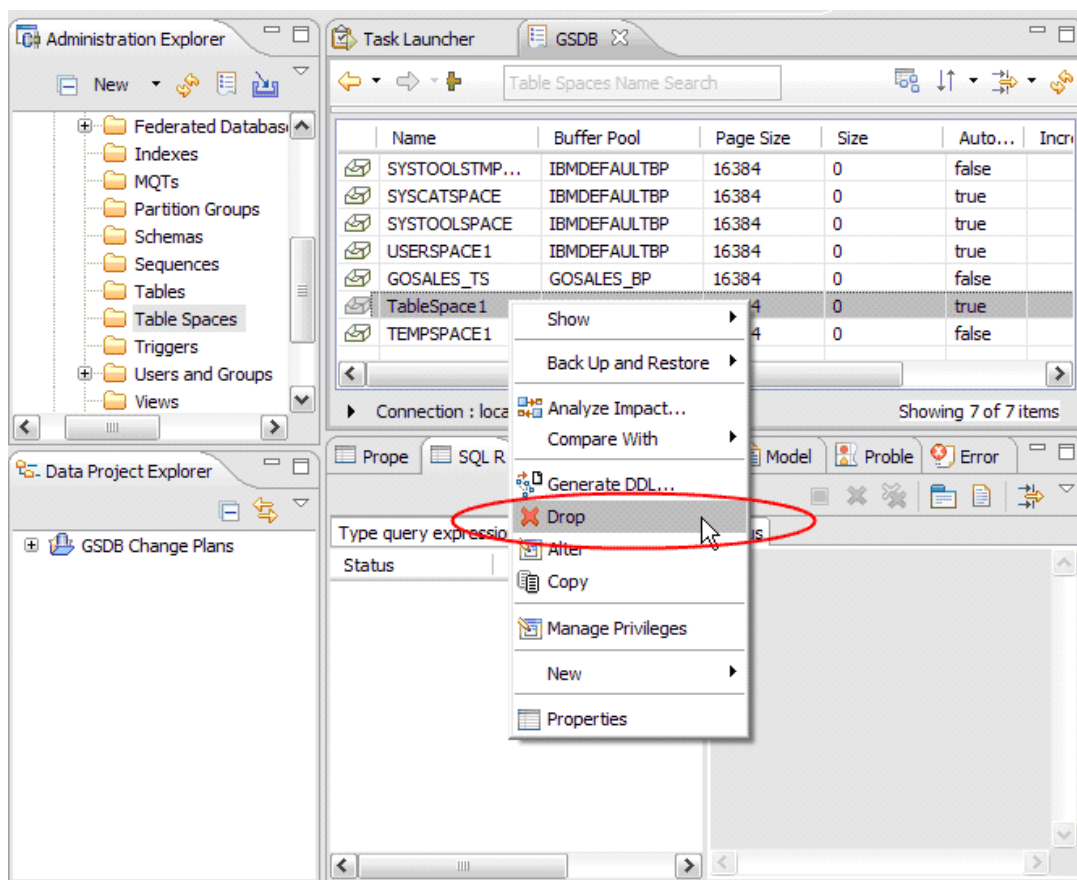
Figure 3.10 - Dropping a table space

If you have associated any objects with this table space, Data Studio will ask if you want to drop these impacted objects at the same time. The drop may fail unless you select the option to drop the impacted objects. Alternatively, you can alter the corresponding objects first to disassociate them from this table space (you can associate it with the default or some other table space), and then drop the table space.

3.2.2 Creating and managing buffer pools

A **buffer pool** is database memory cache used to store data and indexes for faster access. This memory cache is also where the changes made by an application to a database object are performed, before persisting it to the database. For any query execution, data is fetched from the table spaces to this memory for processing, before giving back the result to the application. To fetch data from a table space, a buffer pool with the same page size must exist.

Normally, once the data is fetched into the buffer pool, it remains in the buffer pool until the buffer pool gets full, in which case the old data is wiped out to make space for the new. Performance can be greatly improved if the data required by any query exists in the buffer pool instead of having to be retrieved from the data on disk.

By default, a buffer pool named **IBMDEFAULTBP** gets created when you create the database. This default buffer pool is used for all objects unless you explicitly assign them to another buffer pool.

3.2.2.1 Creating a buffer pool

To create a buffer pool using Data Studio:

1. Select the *Buffer Pools* folder, right click on it and select *Create Buffer Pool.* This is shown in *Figure 3.11* below.
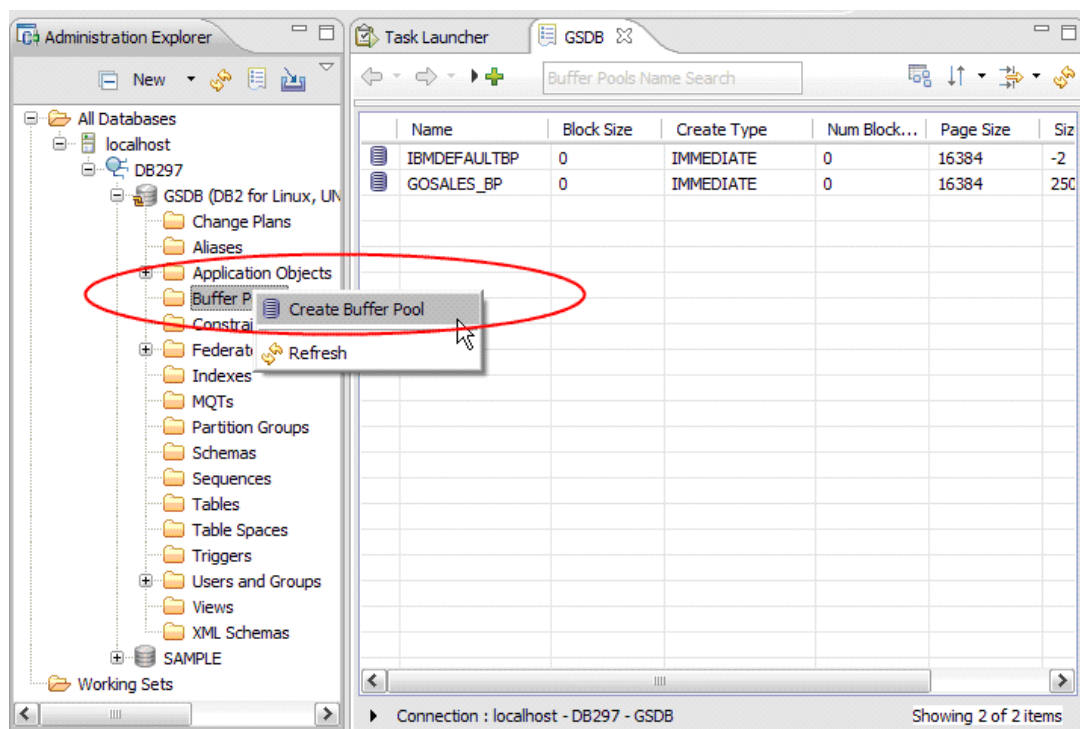
Figure 3.11 – Creating a new buffer pool

2. As shown in *Figure 3.12* below, use the *General* tab to provide basic information like the buffer pool name, total size, page size, etc. If you want to create this buffer pool immediately after the execution of the DDL, the *Create type* field should be set to *IMMEDIATE;* otherwise you can defer it for the next database start by selecting the *DEFERRED* option.
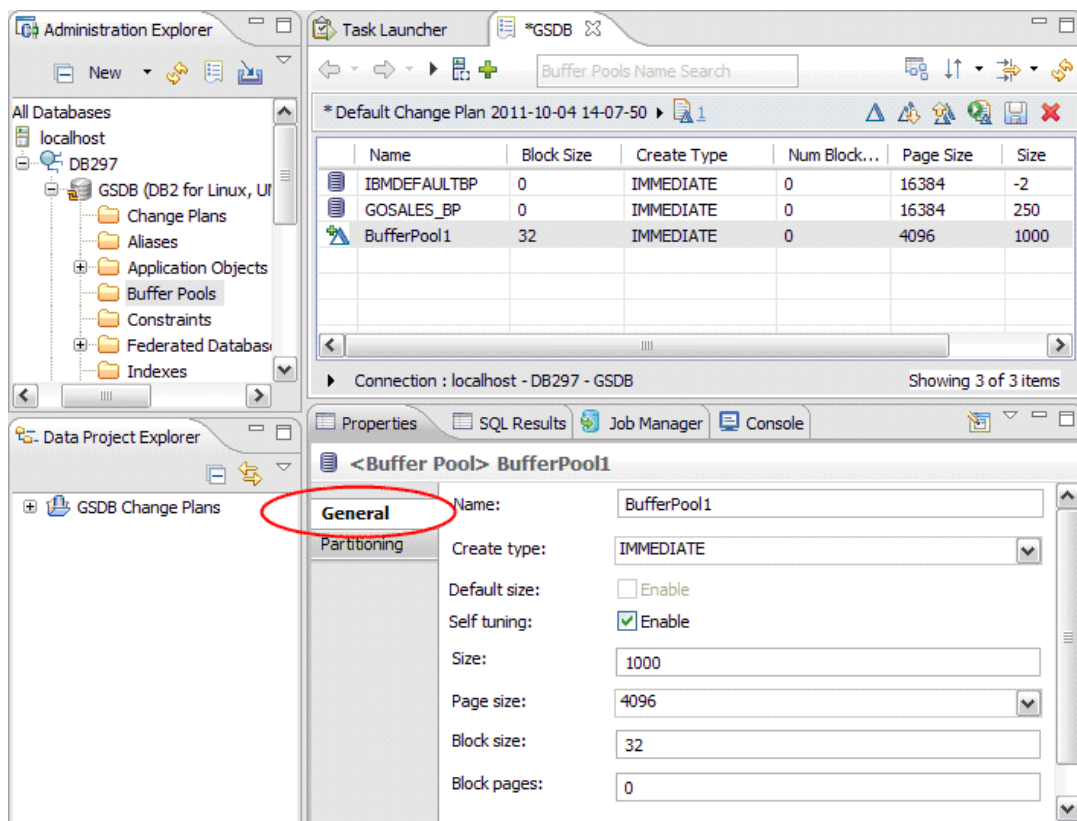
Figure 3.12 – Defining the buffer pool properties

3. Create the buffer pool by following the deployment steps from *Section 2.4.5* of *Chapter 2*.

3.2.2.2 Use the new buffer pool

Now you have a new buffer pool that you can associate with table spaces that you create. Both the table space and buffer pool must be of the same page size. This association tells the DB2 data server to use this buffer pool to fetch the data from this table space. For existing table spaces, you can alter them to associate them with this new buffer pool.

3.2.2.3 Drop a buffer pool

You can drop a buffer pool by selecting the buffer pool from the Object List Editor. The list of buffer pools can be viewed by clicking on the *Buffer Pools* folder in the Administration Explorer. To drop a buffer pool, right click on it in Object List Editor, and select *Drop*. This is shown in F*igure 3.13*. To deploy the drop on the database, follow the deployment steps from *Section 2.4.5* of *Chapter 2*.
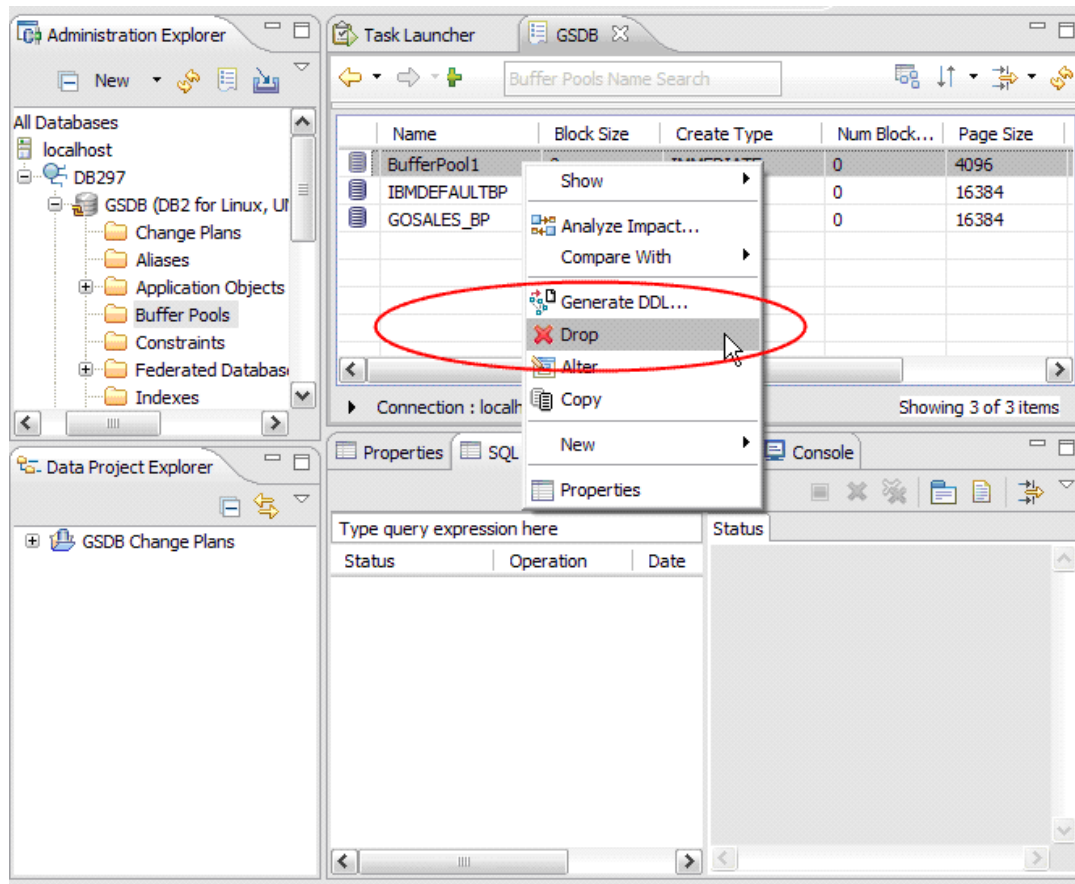
Figure 3.13 - Dropping a buffer pool

If you have associated any table space with this buffer pool, Data Studio will prompt you to drop the impacted table space at the same time. The drop will fail unless you select the option to drop the impacted table space. Alternatively, you can alter the corresponding table space first to disassociate it from this buffer pool (you can associate it with the default or some other buffer pool) and then try dropping this buffer pool again.

3.2.3 Reorganizing data

Normally, data is written to memory in a sequential manner. However, frequent operations on the database objects can fragment the data, which may mean that data is stored non-sequentially and can increase the size of the table as the data spans multiple data pages. Fragmentation of the data may result in multiple I/O operations to fetch the same data that without fragmentation would have taken only a single I/O operation. Reorganization of the tables and indexes will defragment the data, and hence improve the I/O cost and in some instances reduces memory usage.

To reorganize data using Data Studio:

1. Click on the *Tables* folder in the Administration Explorer and select a table from the Object List.

2. Right click on the table and select *Reorg Table option* from *Manage* sub menu. To reorganize the indexes of the table, you can select *Reorg Index*. *Figure 3.14* shows the *Reorg* options for the table *ORDER_DETAILS* in *GOSALES* schema*.*
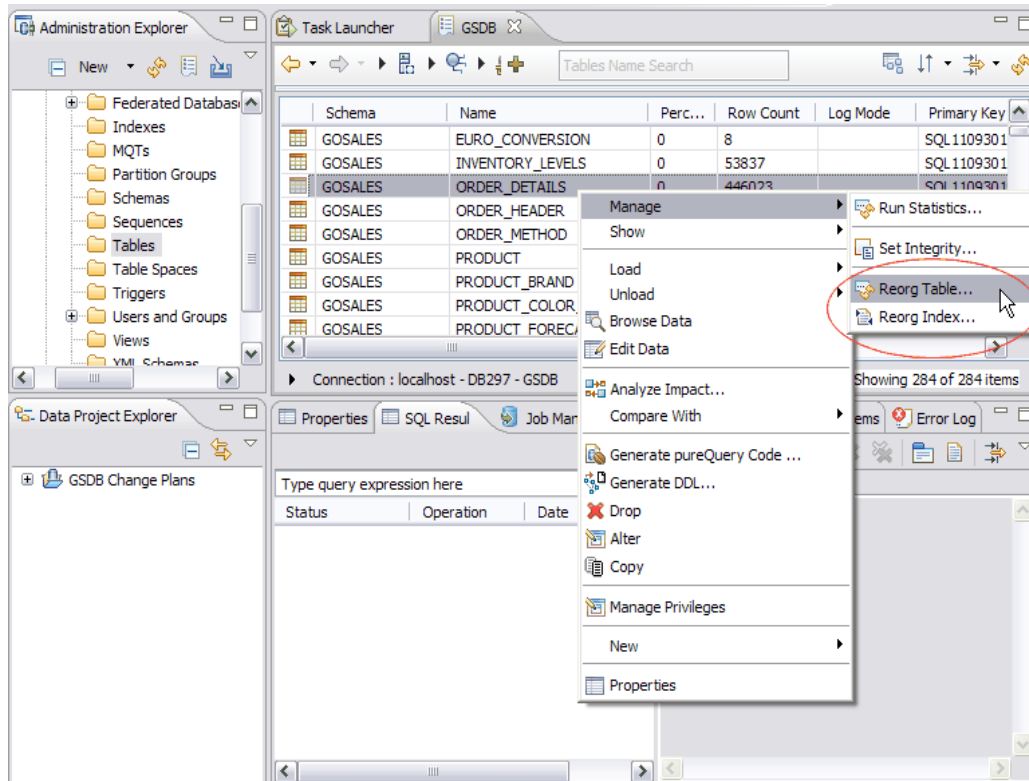
Figure 3.14 – Reorganization options for tables

3. A new editor will appear which lets you configure the Reorg operation. This is shown in *Figure 3.15.*
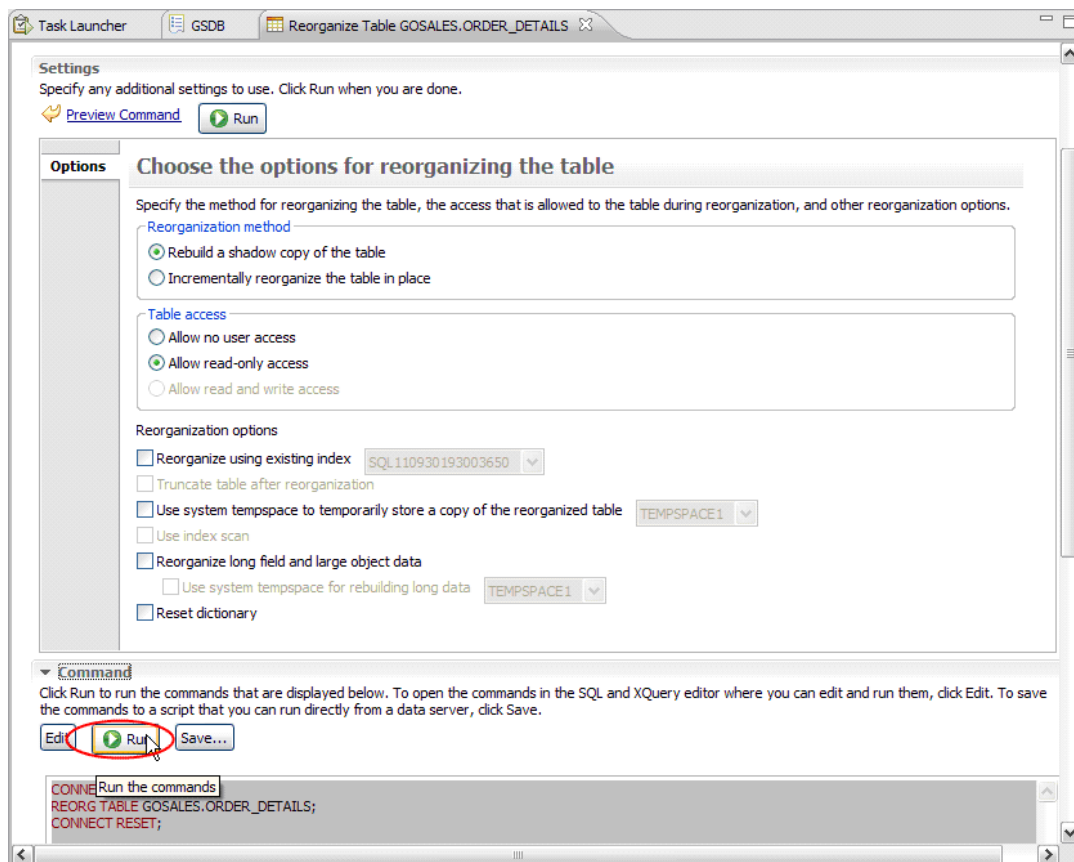
Figure 3.15 – Options for reorganizing a table

Here are the details for these options.

You can reorganize a table in two ways.

- In-place reorganization (called *Incrementally reorganize the table in place* in the *Options* tab shown in *Figure 3.15*), allows reorganization to occur while the table/index is fully accessible. If you select this option, you can set the table access control to allow read, or read and write access.

- Offline reorganization (called *Rebuild a shadow copy of the table* in the *Options* tab) means that reorganization occurs in offline mode. You can specify whether to allow read access or not during offline reorganization.

While offline reorganization is fast and allows perfect clustering of the data, online reorganization lets the table remain online to applications. If the need to write to the table is critical to the application during this period, then an online reorganization is preferred. During online reorganization, you have more control over the process and can pause and restart it; however online reorganization takes more time to complete.

You can reorganize the table using an existing index. This will allow faster access to the data while reorganizing the table.

Offline reorganization can use the temporary table space for storing the copy of the reorganized table. You can choose to use this by checking the option *Use system tempspace to temporarily store a copy of the reorganized table.* You can also choose to reorganize long fields and large object data as well as the option to reset the dictionary if data compression is enabled.

4. After choosing appropriate options, you can run the command by clicking *Run*, as shown in *Figure 3.15* above.

5. Close the Editor before moving to the next task.

3.2.4 Gathering statistics

Whenever a query is executed by an application, the DB2 optimizer compiles the query and creates an **access plan,** which describes the sequence of steps to execute the query and fetch the data it returns. Access plans give estimations on the cost and time for executing a query. The sequence of steps created as part of the access plan depends on a number of factors, such as:

- Size of the database table, indexes and views

- Distribution of data in the specific columns of the tables

- Average length and the cardinality of the column values

- The number of null and the highest and lowest values of the columns

As update, insert, and delete transactions happen on a database, the data grows or shrinks and often changes its distribution. This means the statistics that the optimizer currently knows about are outdated and no longer reflect reality. If the information stored in the catalogs is not up to date, the steps created as part of the access plan may not be accurate and can generate a less than optimal access plan, which may negatively affect performance.

You should gather statistics regularly so that DB2 optimizer generates optimized and efficient access plans. Statistics can be collected on tables, indexes and views. You can use Data Studio to help you determine how and when to run statistics. See *Chapter 7* for more details.

Note:

Even though it is possible to automate the update of table statistics, in a production environment it is recommended that DBAs manually update the statistics for the most critical tables in order to provide continuous enhanced performance for workloads using those tables.

To gather the statistics using Data Studio:

1. Click on the *Tables* folder in the Administration Explorer and select a table from the Object List.

2. Right click on the table and select *Run Statistics* option from *Manage* sub menu. *Figure 3.16* shows the *Run Statistics* option for the table *BRANCH* in *GOSALES* schema.
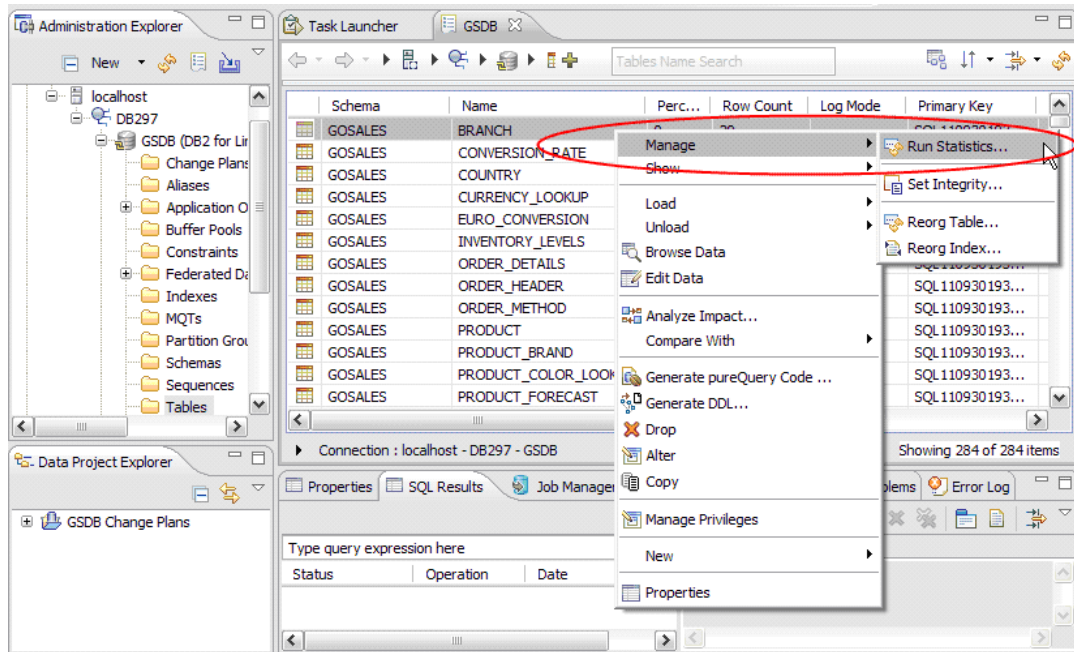
Figure 3.16 – Updating Statistics on a table

3. A new editor will appear which lets you configure the *Run Statistics* operation. This is shown in *Figure 3.17.* The runstats utility of the DB2 data server provides the option to register and use a statistics profile, which specifies the type of statistics that are to be collected for a particular table; for example, table statistics, index statistics, or distribution statistics. This feature simplifies statistics collection by enabling you to store runstats options for convenient future use. The *Profile* tab in *Figure 3.17* lets you specify the profile settings. To update the statics immediately, select the option to *Update statistics now*.
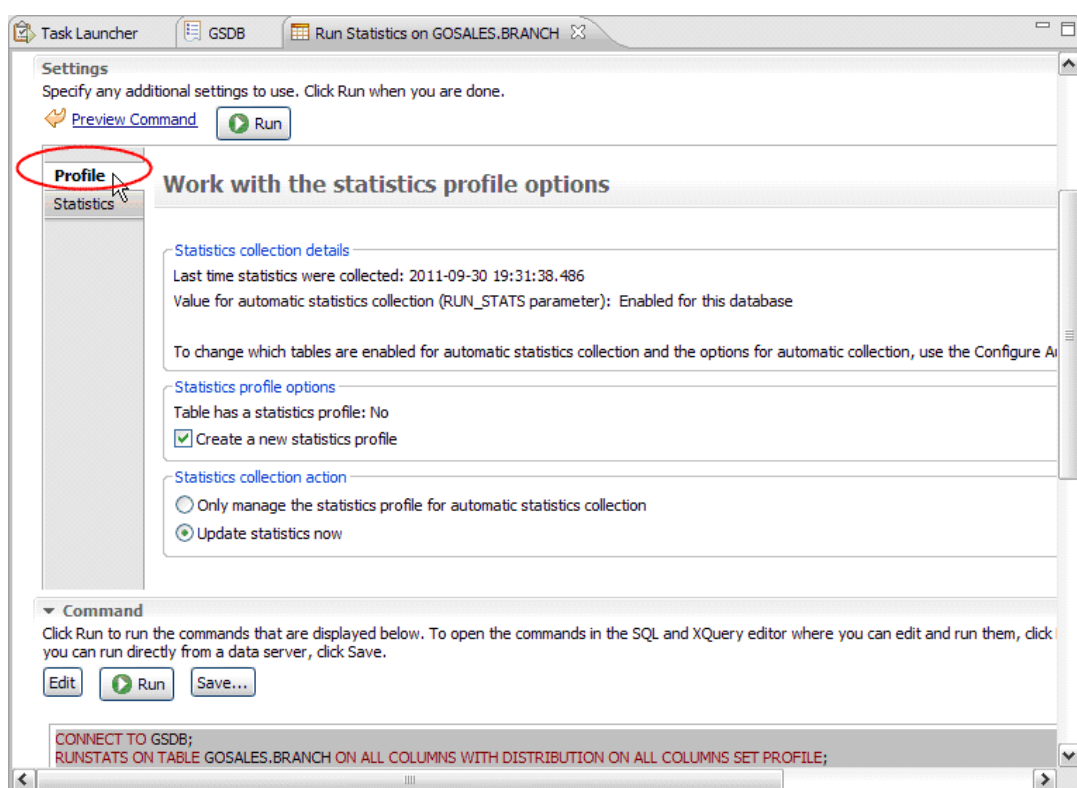
Figure 3.17 – Profile options for updating statistics on a table

4. Click on *Statistics* tab to specify the type of statistics you want to collect. You can gather statistics on all columns, with an option to collect data distribution statistics. You also have an option to collect basic statistics on indexes, including optional extended statistics with data sampling that is useful for large indexes. As shown in *Figure 3.18*, you can leave the default settings, which are the recommended options.
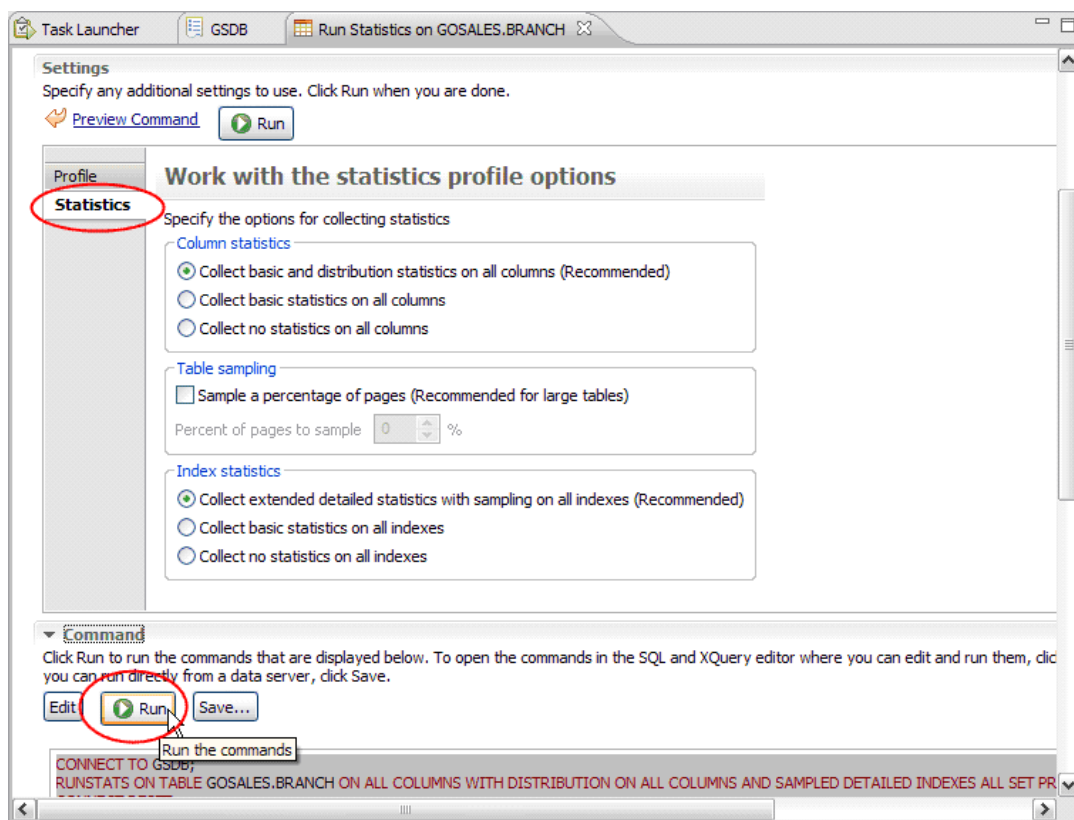
Figure 3.18 – Options for updating statistics on a table

6. After selecting appropriate options, you can run the command by clicking *Run*, as shown in *Figure 3.18* above.

7. Close the Editor before moving to the next task.

3.3 Moving data

With Data Studio, you can move your data from the database tables to the file system (**Export**) and bring back the data from file system to the tables (**Import** or **Load**). This operation lets you transfer the data from one table in one database to another table in the same or different database. This can also be useful when you want to import a large volume of data into a table that includes large objects. This section will outline how to export data into file system and import the data from file system into a table.

> **Note**: The location of the data files that are used for exporting or importing data is expected to be on the computer where your DB2 database server exists for the options *Unload->With Export Utility*, *Unload->With Optim High Performance Unload*, *Load->With Import Utility*, and *Load->With Load Utility*. However, in case of *Unload->With SQL*, and *Load->With SQL* options, the location of the data file is expected to be on the same

computer as the Data Studio client.

3.3.1 Exporting data

To export the data from a table to the file system using Data Studio:

1. Click on the *Tables* folder in the Administration Explorer. Browse through the tables in the Object List Editor. Select the table you would like to export, right click on it, and select *Unload -> With Export Utility*, as shown in *Figure 3.19*.
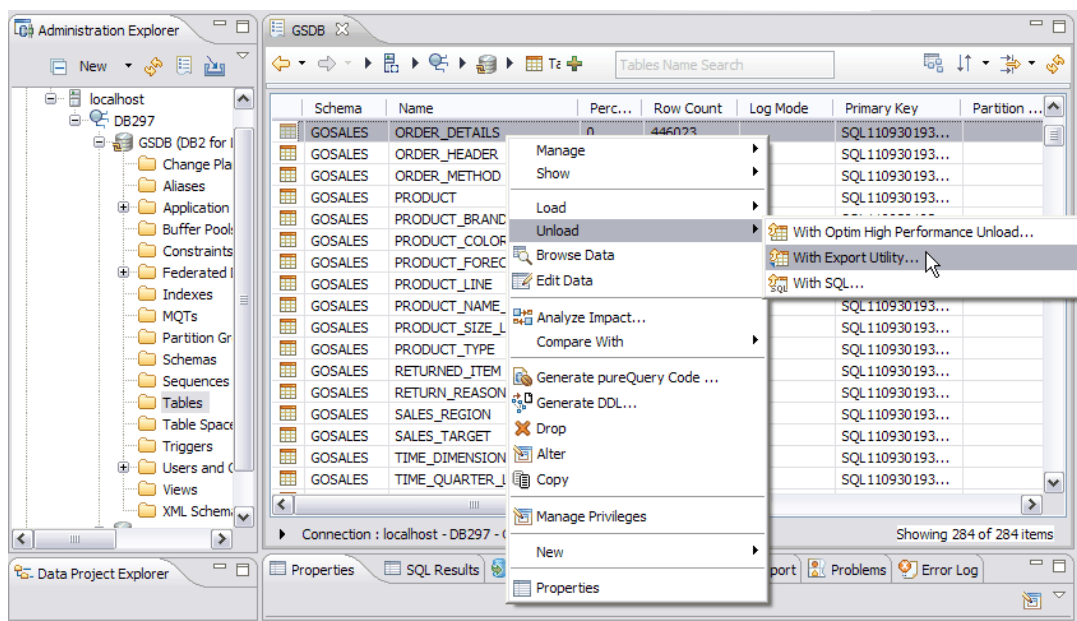
Figure 3.19 – Exporting data

2. A new editor will open which will let you select the file name and the format for the exported data in *Target* tab. As shown in *Figure 3.20*, choose the delimited format, and specify a path and file name for output.
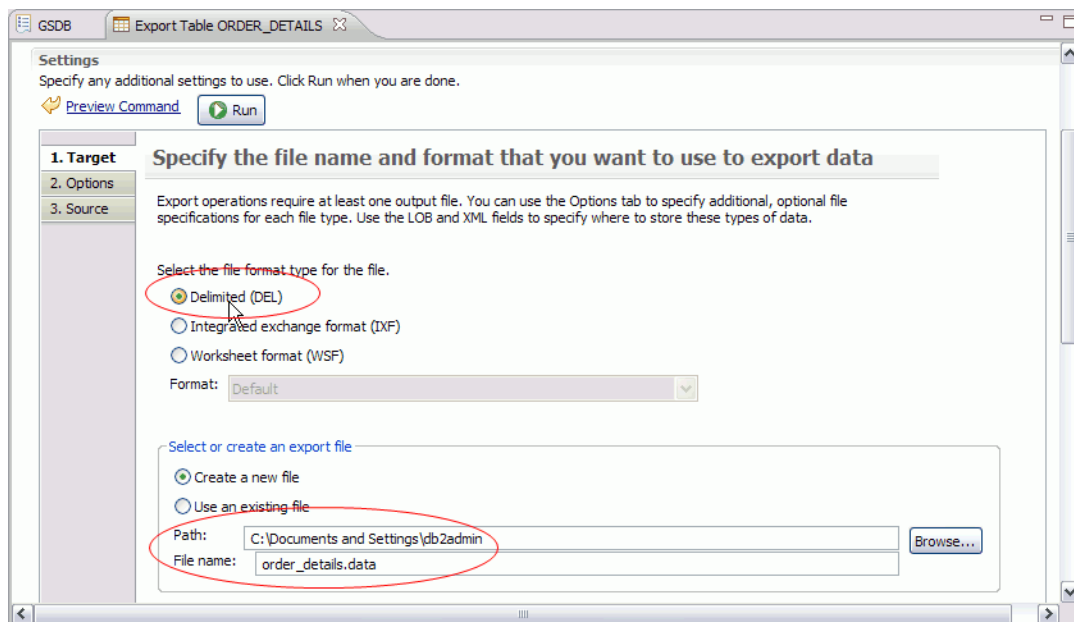
Figure 3.20 - Specifying the format and the file name for the exported data

The three file formats supported are Delimited, Integrated Exchange Format, and Worksheet Format.

- ***Delimited (DEL) format*** is generally used when the data needs to be exchanged between different database managers and file managers. It stores the data in a delimited text file where row, column and character strings are delimited by delimiters.

- ***Integrated Exchange Format (IXF)*** is a rich format, which stores the data in a binary format. It can also store the structural information about the table (DDL) and hence can be used to create the table during an export.

- ***Worksheet Format (WSF)*** is used when the data needs to be exchanged with products like Lotus® 1-2-3® and Symphony™.

3. You can also specify whether to export large object (LOB) data into a separate file or files. Similarly, XML data can also be exported to a separate file or files.

4. Under the *Source* tab, you can specify an SQL statement to select the data to export. As shown in *Figure 3.21,* a full **SELECT** will automatically be created by default; however you can edit the generated SQL statement to choose any specific columns.
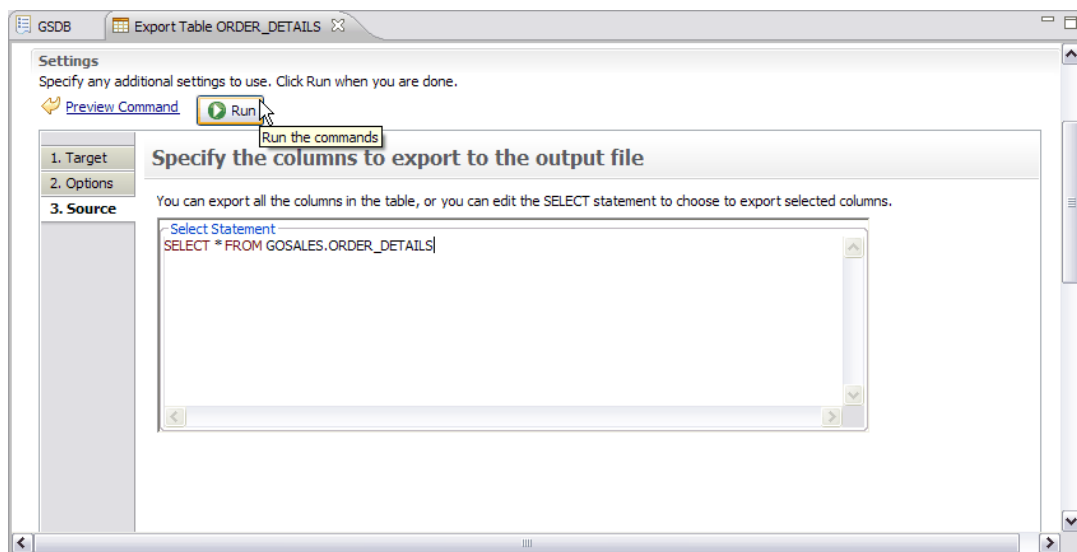
Figure 3.21- Source SQL for exporting data

5. After selecting appropriate options, you can run the command by clicking *Run*, as shown in *Figure 3.21* above. This will export the data to the file system.

6. Close the Editor before moving to the next task.

3.3.2 Importing data

You can load/import data into a table. To do that you should have the data in your file system in one of the supported formats (DEL, IXF, or WSF) as described in the previous section.

Note: If you are importing a large data, you may need to increase the database log size as described in Section *3.4*, or specify automatic commit in the *Advanced Options* of the Import editor that is shown in *Figure 3.24*.

To import the data into a table using Data Studio:

1. Click on the *Tables* folder in the Administration Explorer. Browse through the tables in the Object List Editor. Select the table you would like to load, right click on it, and select *Load -> With Import Utility,* as shown in *Figure 3.*22.
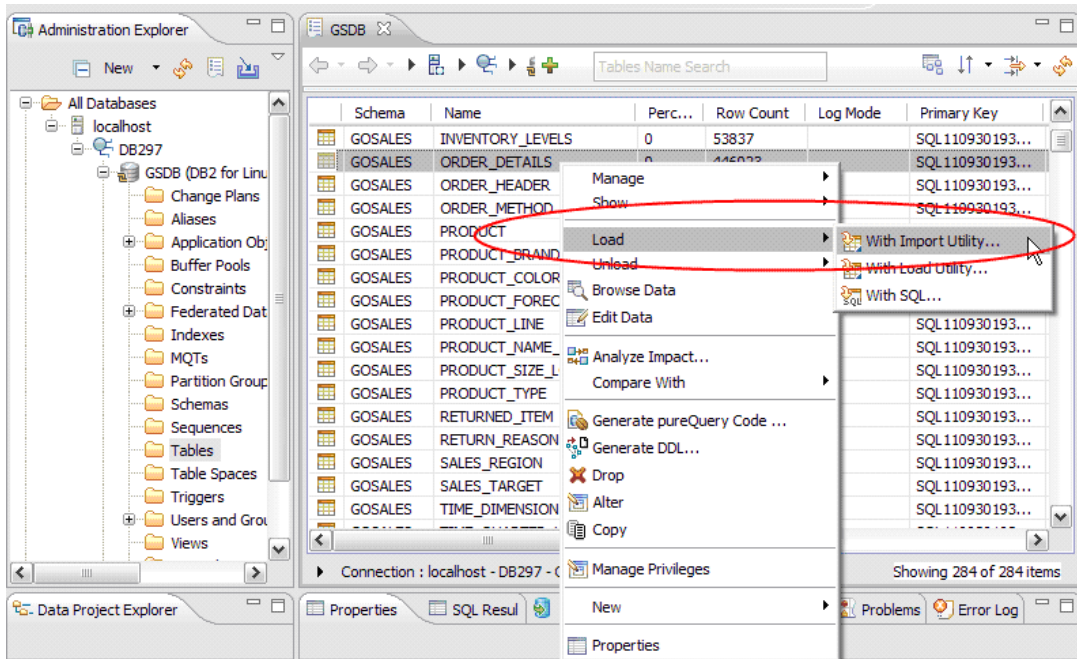
Figure 3.22 - Importing data

2. A new editor window will appear which will let you specify the name and the format of the data file to import, as show in *Figure 3.23*.
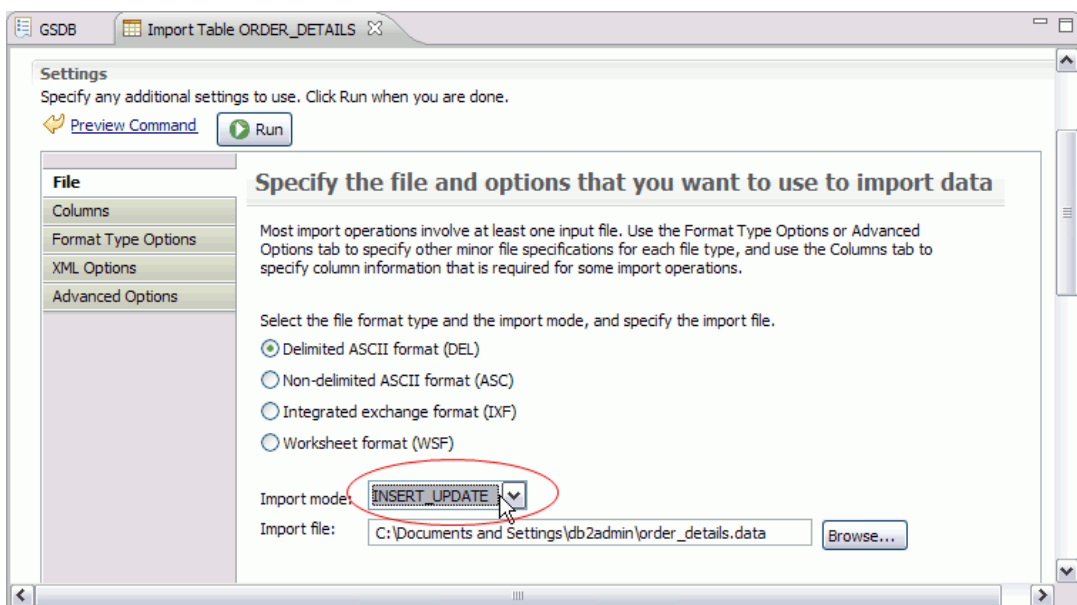
Figure 3.23 – Selecting the data files format and the location

3. You can choose between the Import modes *INSERT*, *INSERT_UPDATE* or *REPLACE*.

- INSERT means the imported data will be appended to the existing data in the table.

- The REPLACE option will overwrite the existing data.

- INSERT_UPDATE updates the row if a particular row already exists; otherwise it inserts the new row.

4. You can also specify different options like commit frequency, table access during import, the maximum number of rows to be inserted, warning threshold, etc., in the *Advanced Options* Tab. In this example, select *Automatically Commit* as shown in *Figure 3.24*:
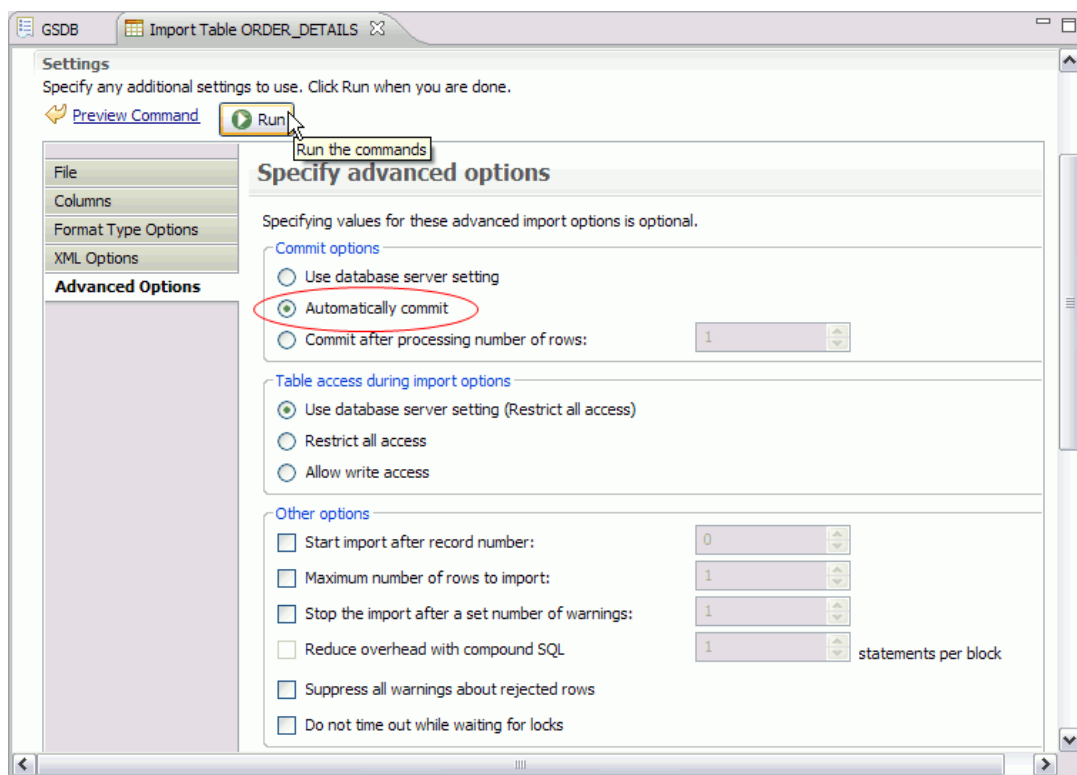
Figure 3.24 – Selecting advanced options for Import

For details on these options see the DB2 documentation for **IMPORT** command here http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.cmd.doc/doc/r0008304.html

5. If you have large objects and XML data to be imported, you can specify from where that data can be retrieved, in the *Columns* tab options.

6. Once you are done providing all the necessary options, you can click on the *Run* button, to import the data into the table, as shown in *Figure 3.24* above.

7. Close the Editor before moving to the next task.

Note:

You can also use the Load utility to load the data into the table. Load achieves much the same result as Import, but can be faster for large quantities of data. Load is a multi-step process whereas Imports can do most processing in one step. For more information about the Load utility, refer to the documentation from DB2 Information Center at this location: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.ad min.dm.doc/doc/c0004587.html. Once you understand the Load process, you can try it using Data Studio.

3.4 Planning for recovery: Configuring DB2 logging

The DB2 logging facility logs any SQL statement updates done on the database. These logs help in recovering a database state in case of any system failure. DB2 logging can be categorized as follows:

Circular Logging — the changes are logged only for those transactions that are not committed. With this kind of logging, you can recover the data from the latest backup only. This is the default when you create a database.

Archive logging — all the changes including the committed ones are logged here. With this kind of logging, a roll-forward can be done after restoring from a backup. The roll-forward process applies all the transaction that occurred since the last backup. In addition, archive logging is a must for online backup operations.

Note:

For more information about database logging and the logging options, see this DB2 Information Center topic:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0006073.html

You can change the logging type by right clicking the database and selecting S*et Up and Configure -> Configure Database Logging*, as shown in *Figure 3.25* below.
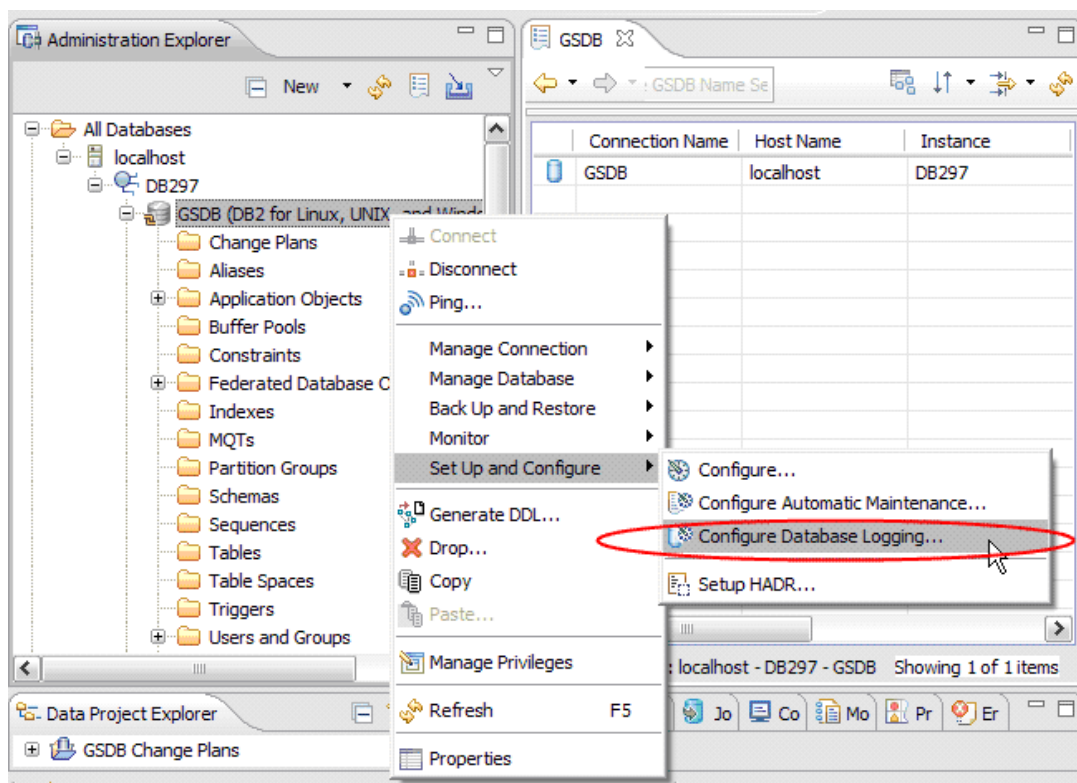
Figure 3.25 – Configuring Database logging

A new editor will appear which will let you select the kind of logging you want. If you select archive logging, you need to specify the location of the archive logs (in *Logging Type* tab), and location of the new backup (in *Backup Image* tab). A new backup of the database is required so that in case of failure, a roll-forward is possible after restoring the database. You will find more information about restore and roll-forward in the next section. *Figure 3.26* below shows the options for configuring logging.
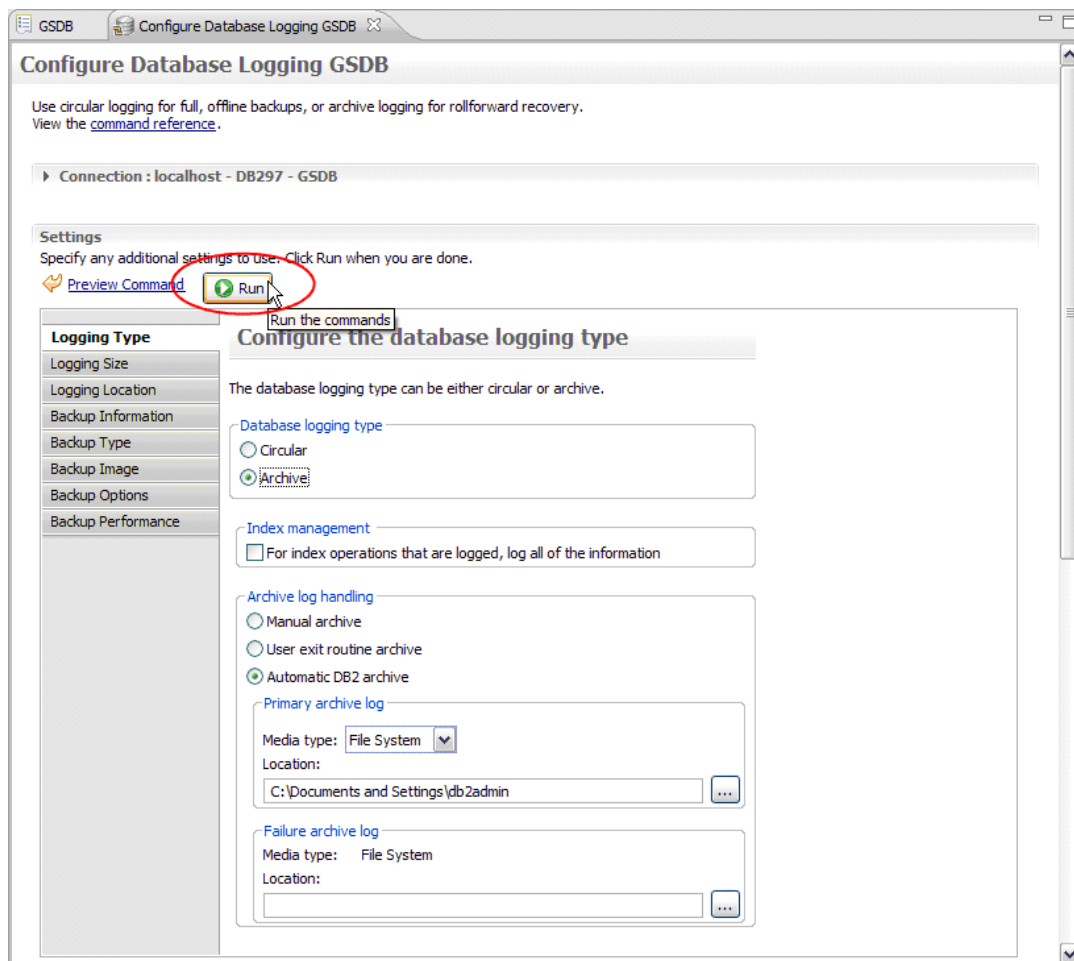
Figure 3.26 – Configuring Archive logging

After providing the logging type and backup image details, you can click on the *Run* button as shown in *Figure 3.26* to configure the required logging. As always, close the editor when you are done with the task.

3.5 Backing up and recovering databases

Backups allow the database to be recovered in case of crash or database failure. Backups also allow you to move complete databases from one location to another. You can take the backup of a database and recover it in the same or a different compatible system; however, not all system combinations are supported. For more details on system compatibility, refer to the documentation in DB2 information center:
http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0005960.html.

3.5.1 Backup

You can create a backup of your database using Data Studio. The database can be restored at a later point in time using this backup.

Note:

For more information about backup, see this topic in the DB2 Information Center:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0006150.html

To create a backup of a database using Data Studio:

1. Select the database you want to back up in Administration Explorer, right click on it and select *Back Up and Restore* -> *Backup*, as shown *Figure 3.27*.
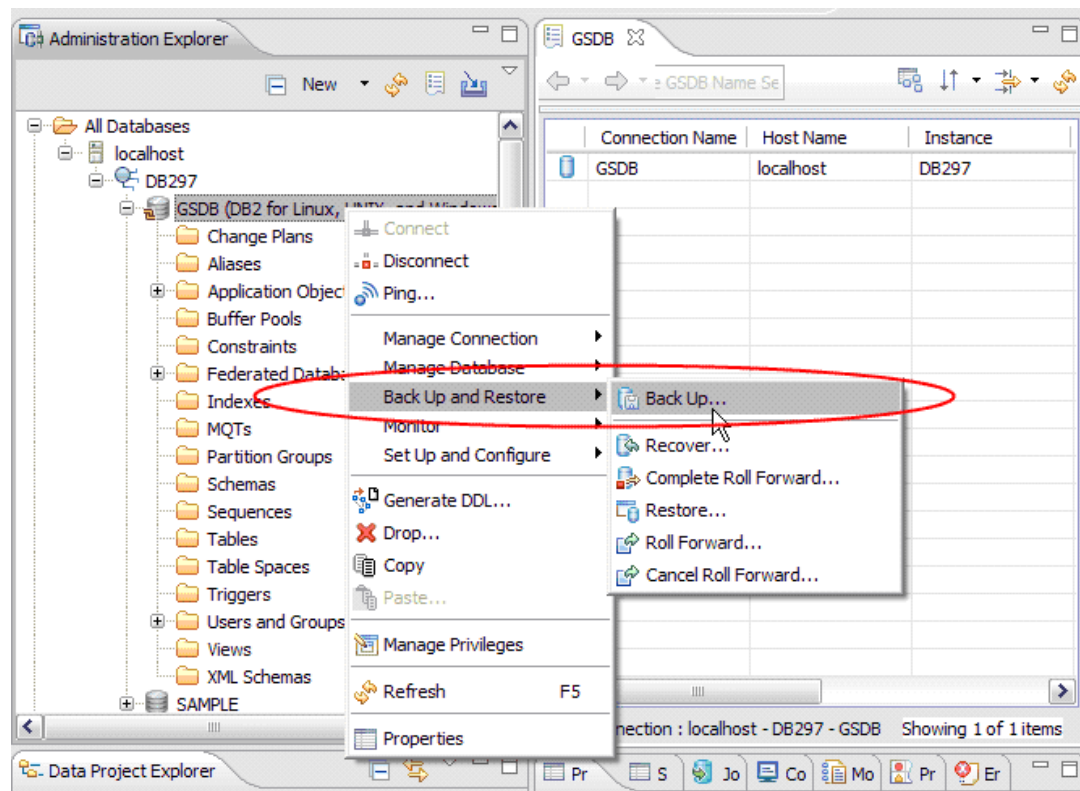
Figure 3.27 – Back up database

2. A new editor will open. Under the *Backup Image* tab, you can provide the media type where you want to take the backup and the location of the backup image. This is shown in *Figure 3.28.*
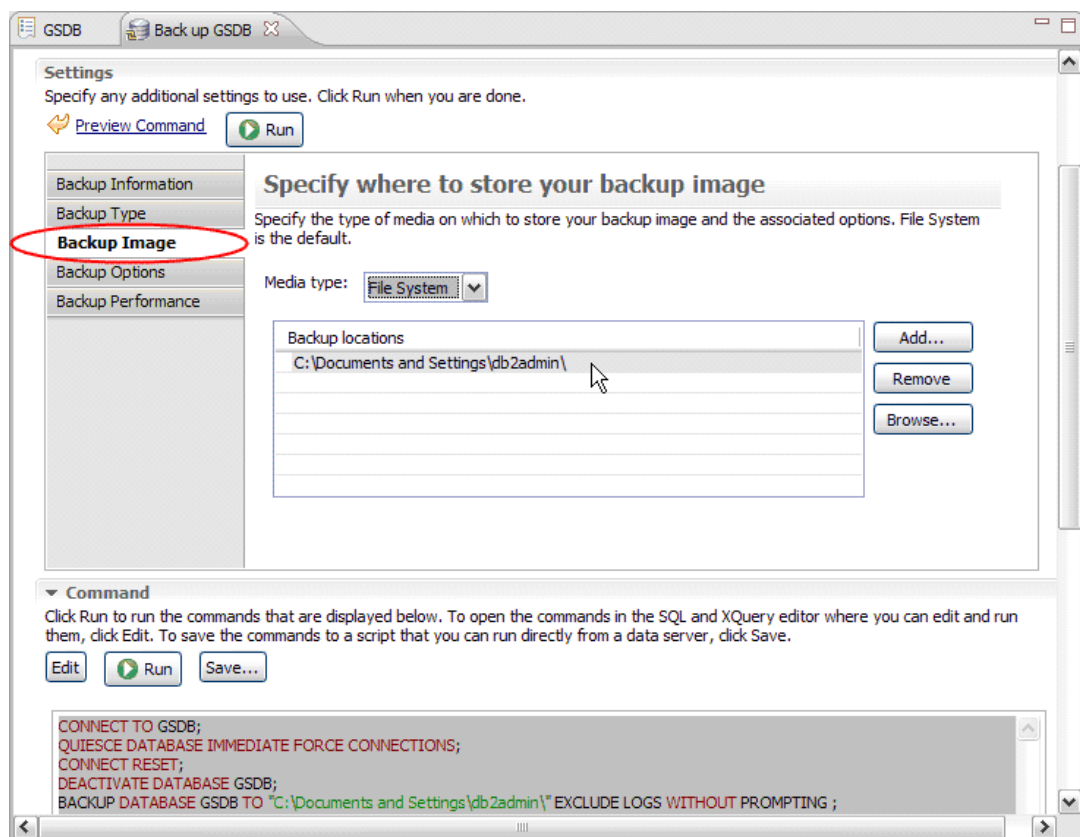
Figure 3.28 – Taking a backup on a file system

3.   Click on *Backup Type* tab to specify whether you want to backup the entire
     database, or backup selective table spaces. Select *Back up the entire database*
     as shown in *Figure 3.29*.
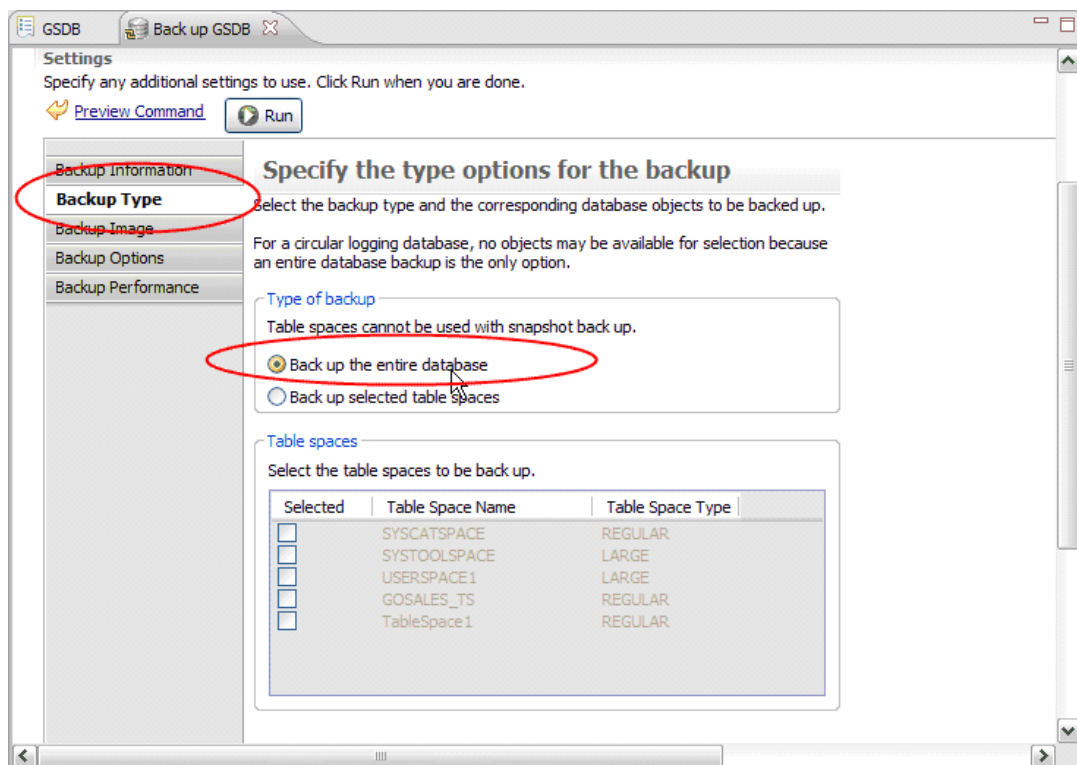
Figure 3.29 – Taking a backup of entire database

4. Under the *Backup Options* tab, you can specify more options for backup. These are explained below:

Backup Type - Lets you create **full**, **incremental** and **delta** backups.

- A full backup contains a complete backup of the entire database.

- An incremental backup contains any data that is new or has changed since the last full backup.

- A delta backup contains any data that is new or has changed since the last backup of any type: full, incremental or delta.

Availability - You can specify if you require the database to be online during the backup process. Online backup is possible only when archive logging is being used. *Figure 3.30* shows these options.
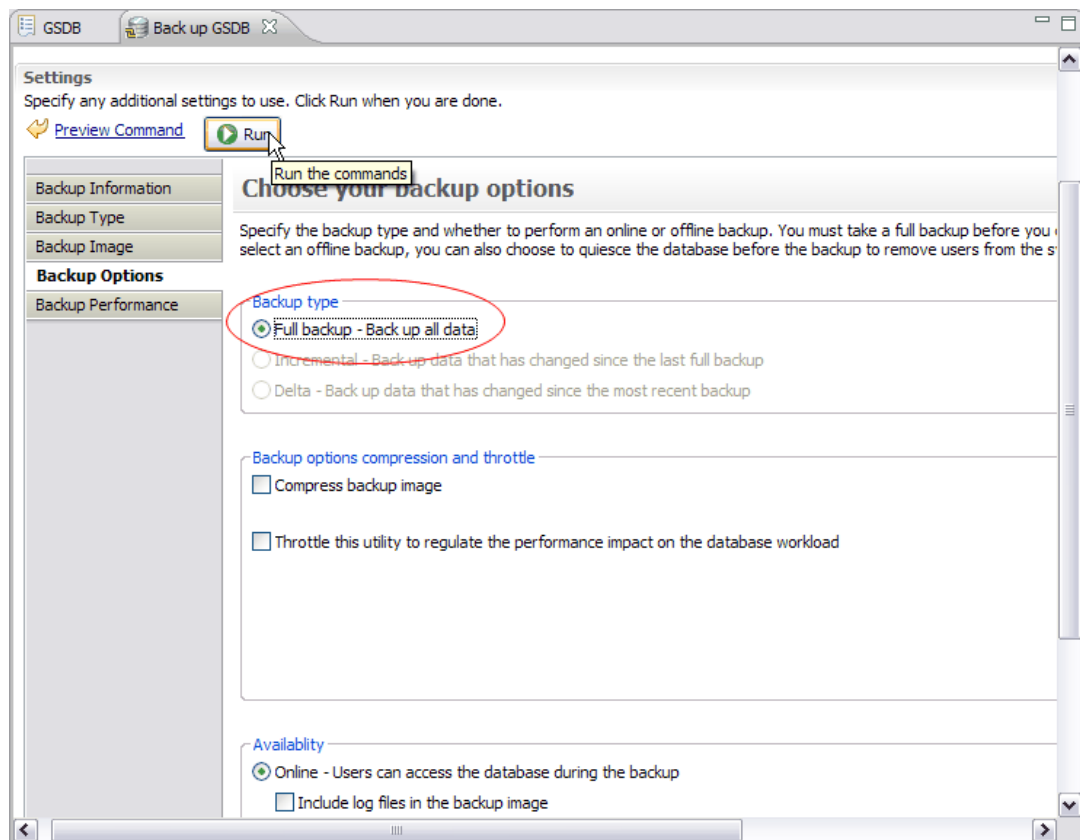
Figure 3.30 – Backup options

5.   Finally, you can select *Run* button to take the backup of the database.

6.   Close the editor before moving to the next task.

3.5.2 Restore

If something happens to your database, you can restore it from the backup.

Note:
For more information about restore, see this topic in the DB2 Information Center: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0006237.html

To restore a database from a backup image using Data Studio:

1.   Right click on the database to restore, select *Back Up and Restore -> Restore* as shown in *Figure 3.31* below.
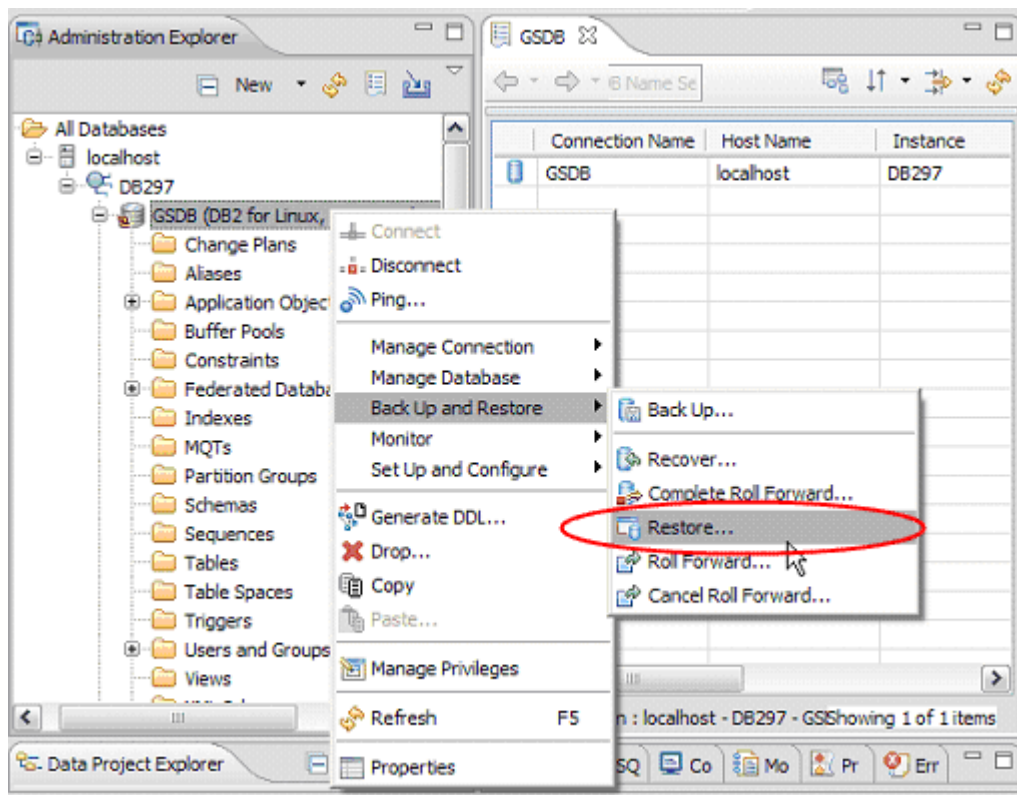
Figure 3.31 – Restoring a database

2. A new editor will appear as shown in the *Figure 3.32*. In the *Restore Type* tab, you can select to restore to an existing database, or create a new database.
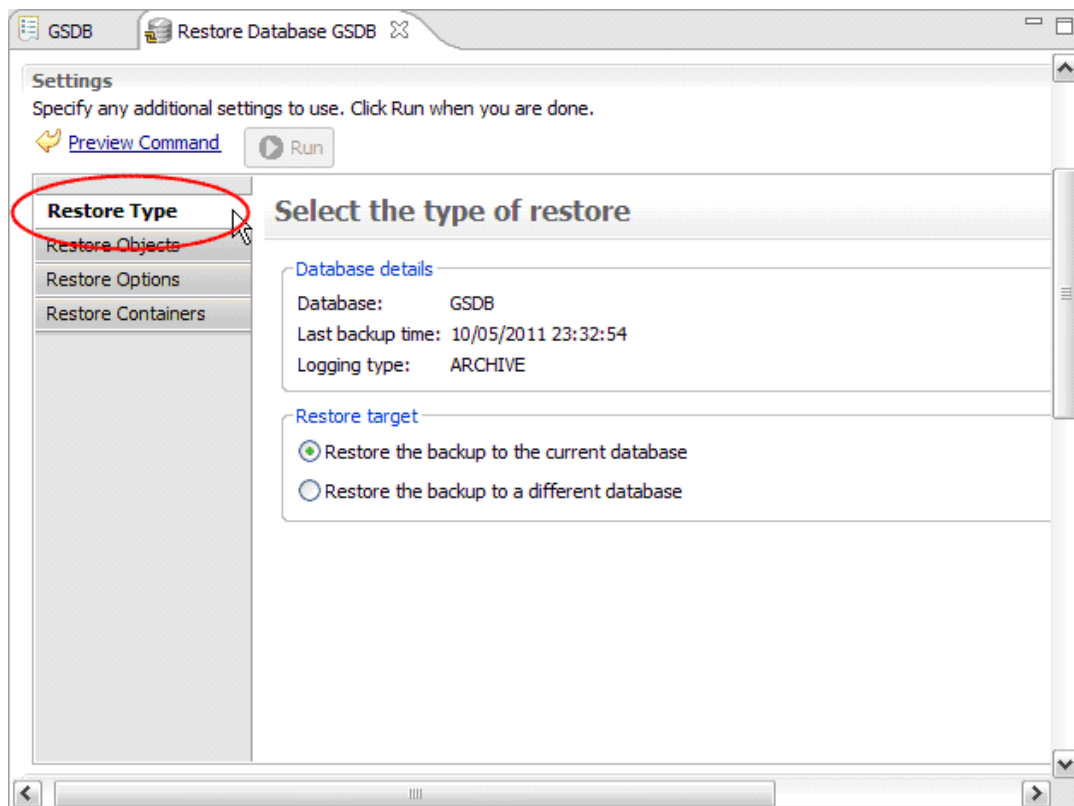
Figure 3.32 – Selecting the restore type

3. Click on the *Restore Objects* tab. Here, you can select to restore only the history file, entire database, or specific table spaces. A history file contains the information regarding the backups taken in the past. This file helps the recover command to find the appropriate backups to be used for recovering the database. For this example, select *Restore the entire database option* as shown in *Figure 3.33*.

> **Note**:
>
> **RESTORE** and **RECOVER** are two different commands. **RECOVER,** as we will see in a later section, provides a simplified command to perform a **RESTORE** followed by a **ROLLFORWARD** command.

4. For the backup image selection from which to restore, you can specify if you would like to manually enter the backup image location or want to select from the list DB2 has maintained in a history file. If the system where the backup is taken is the same as that to which you are restoring, and you have not moved the backup files manually, you will be able to see the backup images in the list. However if you have moved the image manually to the other system to restore it, you can select the location manually.

5. *Figure 3.33* shows the selection of restoring entire database, and the list of the backup images detected by DB2. You can select one of them to restore.
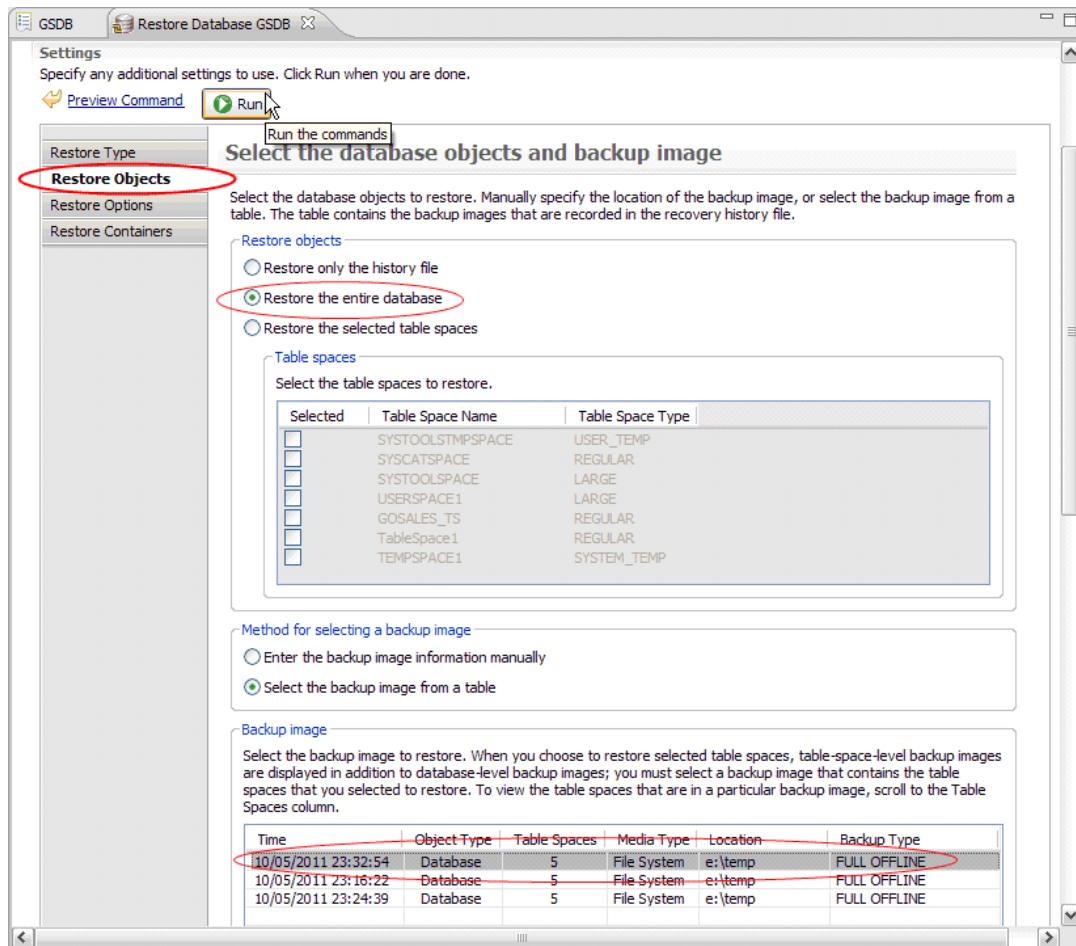
Figure 3.33 – Selecting the objects to restore

6. Under the *Restore Containers* tab, you can specify new container for the table spaces in the database to be restored. This option is useful when the backup image is getting restored on a different system where the same container paths do not exist.

7. Under the *Restore Options* tab, you can select if you want to replace the history file. You can also select if you want to just restore a backup image or if you also want to restore the transactions that would have happened between the time the backup image was taken and a restore operation is performed. This operation is called ***roll-forward***. For this example, leave all the defaults as shown in *Figure 3.34*, and just select the option to remove all connections to the database before starting the restore operation; so that the restore operation does not fail.
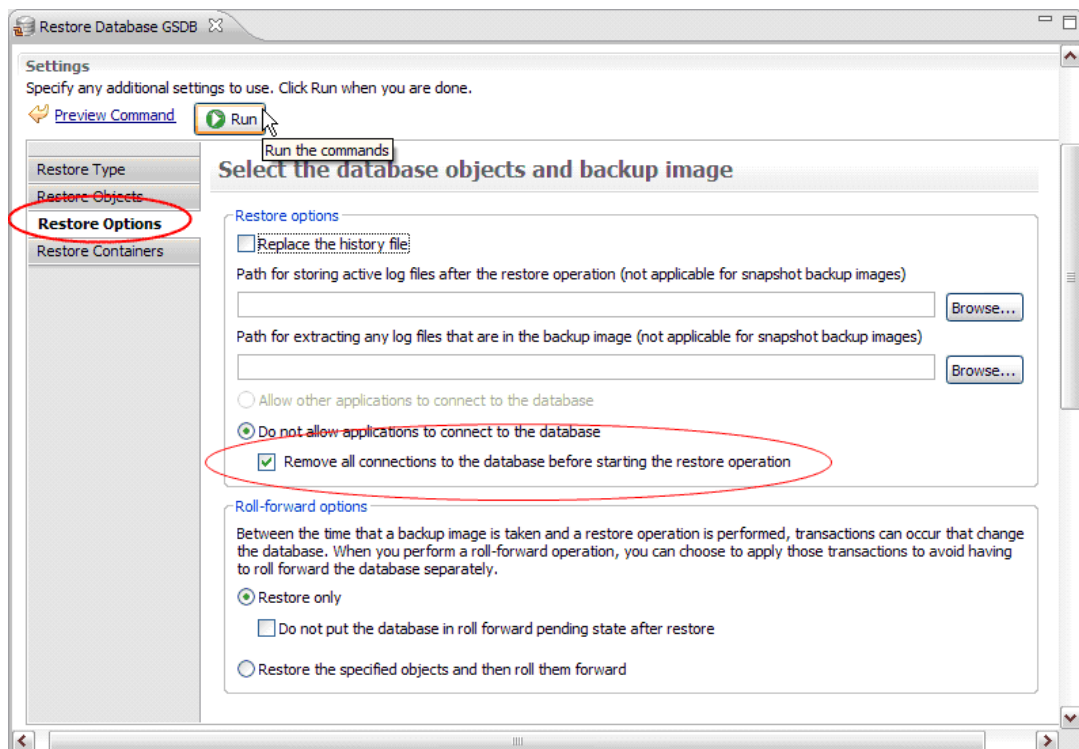
Figure 3.34 – Selecting additional options for restore

8. Select *Run* to restore the database.

9. Close the editor before moving to the next task.

3.5.3 Rollforward

A rollforward operation applies transactions on the restored database, which are recorded in the database logs. This way you can make a database reach to a specific point after restoring it from a backup image.

Note:
For more information about the rollforward operation, see this topic in the DB2 Information Center: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0006256.html

To roll-forward a database to a specific point using Data Studio:

1. Right click on the database, and select *Back Up and Restore -> Roll Forward,* as shown in *Figure 3.35.*
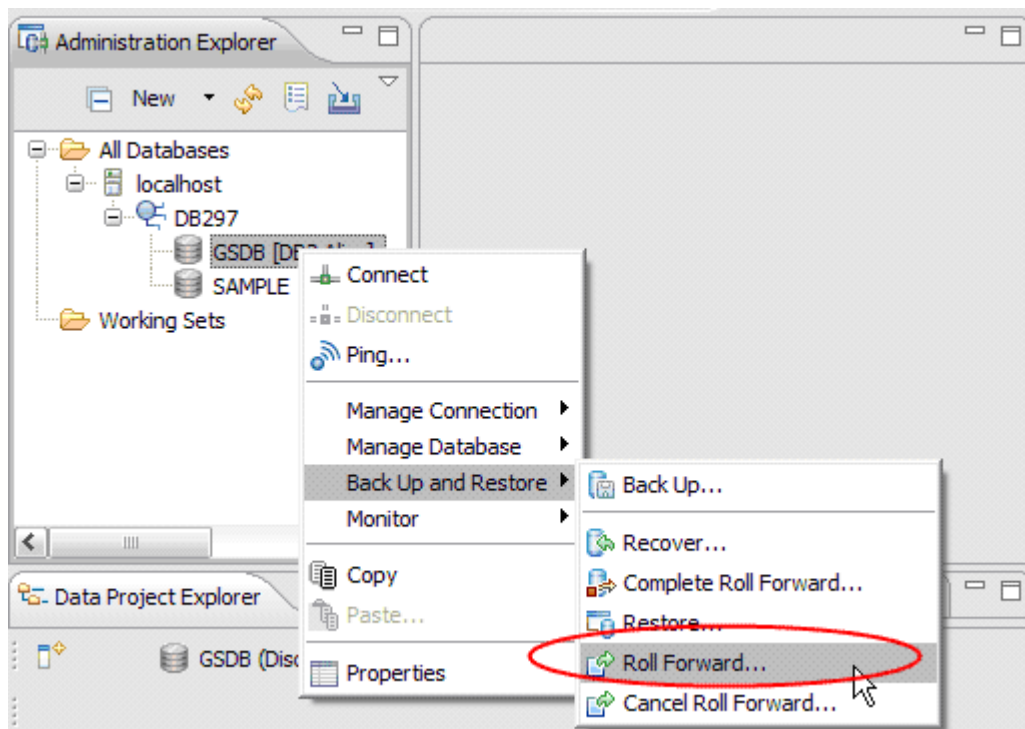
Figure 3.35 – Roll Forward a database

2. A new editor will open. From the *Roll-forward Type* tab, you can select if you want to apply all the logs or up to a specific point in time. This is shown in *Figure 3.36*.
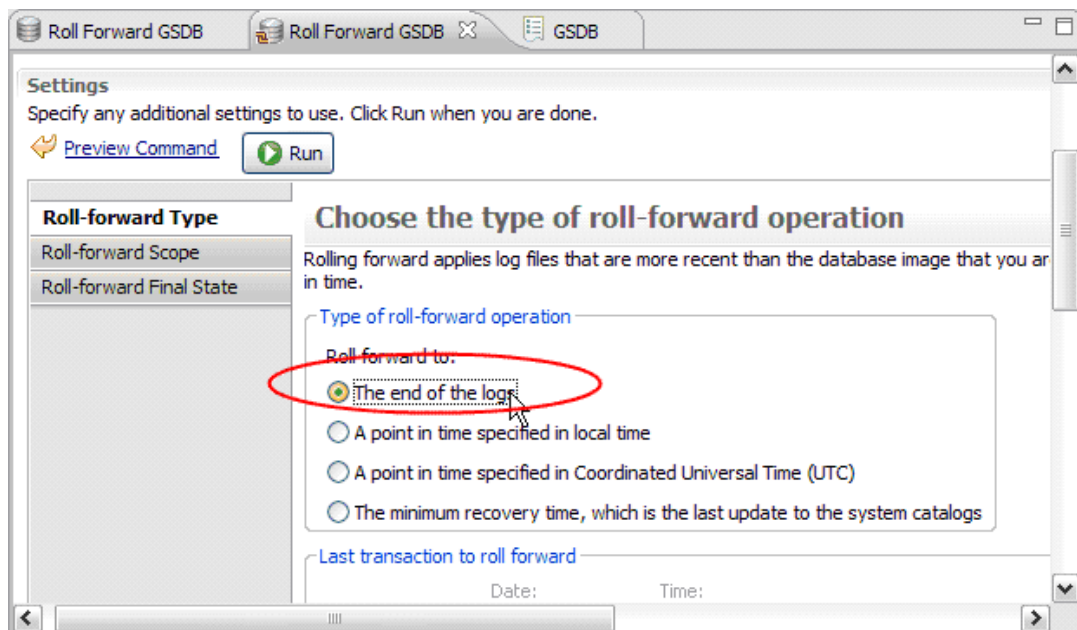
Figure 3.36 – Choose the type of roll-forward operation

3. Under the *Roll-forward Scope* tab, you can select if you want to roll-forward the complete database or just a particular table space. However, if you restored an entire database before, you will not see the option to roll-forward selective table spaces.

4. Under the *Roll-forward Final State* tab, you can select if you want to complete the roll-forward operation or want the database to remain in roll-forward pending state. If you decide to leave the database in the roll-forward pending state, you can complete the roll-forward at a later point in time by doing right click on the database and selecting *Back Up and Restore -> Complete Roll Forward*. For this example, select the option to complete the roll-forward operation, as shown in *Figure 3.37*.

5. Select *Run* to roll-forward the database and complete the roll-forward operation.

6. Close the editor before moving to the next task.
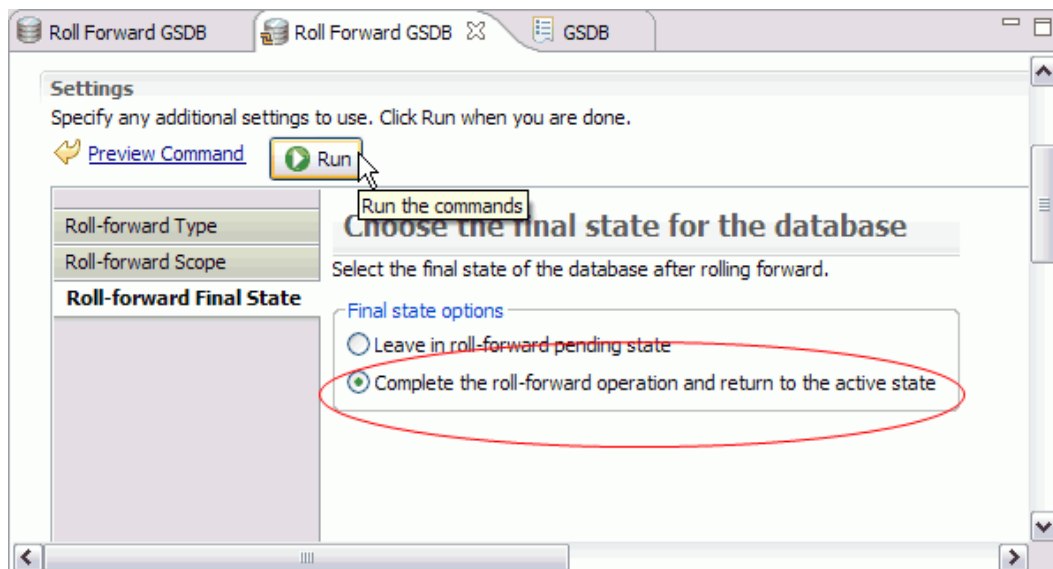
Figure 3.37 – Complete the roll-forward operation and make the database active

3.6 Other maintenance tasks

In addition to what we discussed so far, Data Studio provides additional options to deal with more database maintenance operations. Here is brief description of few of them.

- Recovering a database – Recover is a combination of restore followed by roll-forward. It allows you to specify a point in time or end of logs, and restores the database with appropriate backup image (depending on the specified time and the history file), and then does a roll-forward on that database to the specified time. This can be performed using Data Studio by launching the appropriate utility by right clicking on the database, and selecting *Back Up and Restore -> Recover*.

- Completing a roll-forward recovery – Completes a roll-forward recovery by stopping the roll forward of log records, rolling back incomplete transactions, and turning off the roll forward pending state. Users regain access to the database or table spaces that are rolled forward. As explained in section *3.5.3*, Data Studio provides a single step to initiate the roll-forward operation as well as complete it. However, it also provides another option to just complete the roll-forward explicitly if required. This can be performed by launching the appropriate utility by right clicking on the database, and selecting *Back Up and Restore -> Complete Roll Forward*.

- Cancelling a roll-forward recovery – A roll-forward operation that is in progress but cannot be completed might need to be cancelled. This can be performed using Data Studio by launching the appropriate utility by right clicking on the database and selecting *Back Up and Restore -> Cancel Roll Forward*.

- Configuring automatic maintenance – The database manager provides automatic maintenance capabilities for performing database backups, keeping statistics current, and reorganizing tables and indexes as necessary. This can be configured using Data Studio by launching the appropriate editor by right clicking on the database, and selecting the option *Set Up and Configure -> Configure Automatic Maintenance*.

- Configure database or instance parameters – The database manager provides many parameters at the database level as well as database manager level that can be configured for specific behavior and tuning of a database or database manager. This can be achieved using Data Studio by launching the appropriate utility by right clicking on the database, and selecting *Set Up and Configure -> Configure*. This option is also available on the instance node in Administration Explorer.

3.7 Exercises

To practice what you have learned in this chapter, try out the following exercises.

1. Create a table space *Tb1*. Now create a table *T1* and make sure it's stored in table space *Tb1*. Now create an index on table *T1* and store in the default table space (*USERSPACE1*).

2. Create a table space *Tb2* and check what the privileges available are. Give all the possible privileges to another user.

3. Take an online backup of the *GSDB* database. Do some updates on some of the tables. Now restore the database back and roll-forward it until the end of the logs.

3.8 Summary

In this chapter, you were introduced to some DB2 concepts such as table spaces, buffer pools, and logging. Using Data Studio, you learned how to create these objects, perform **REORG**, **RUNSTATS**, **BACKUP**, **RESTORE**, **RECOVER** and **ROLLFORWARD** operations. For

more detail about these DB2 concepts, refer to the ebook *Getting started with DB2 Express-C*.

3.9 Review questions

1. What are the three types of table spaces in DB2 data servers?

2. What are the file formats supported for exporting data?

3. Which file format allows you to create the table while exporting?

4. Name the two different logging methods supported in DB2 data servers.

5. What two operations are combined to compose the Recover operation?

6. A system-managed table space can have containers of the following type:

    A. Directory

    B. File

    C. Raw Device

    D. All of the above

    E. None of the above

7. A buffer pool is used mainly

    A. To store the intermediate results of queries

    B. To fetch data from disk to volatile memory/RAM

    C. To store updates to the data

    D. To process the data before returning it to the application

    E. All of the above

8. An online table reorganization is preferred when

    A. Access to database tables is critical while reorg is occurring

    B. Data needs to be clustered perfectly

    C. More storage space is required

    D. Index rebuild is required

    E. All of the above

9. When is statistics collection recommended?

    A. When the data is fragmented

    B. After many selects on the data

    C. After many updates on the data

    D. All of the above

E.   None of the above

10. An incremental backup contains:

A.   All the data

B.   Only the modified data after the last full backup

C.   Only the modified data after the last full or incremental backup

D.   Only the modified data after any kind of backup

E.   None of the above

4

Chapter 4 – Monitoring the health of your databases

In this chapter you will learn:

- How to identify which databases to monitor

- The meanings of the various health monitor displays

- How to configure and collaborate with alerts, including how to send alerts via e-mail

4.1 Health Monitoring: The big picture

The Data Studio web console allows you to monitor the health and availability of your DB2 databases (for Linux, UNIX, and Windows and z/OS) databases, and view alerts, applications, utilities, storage, and related information. You can open the health monitoring pages from the web console in a web browser, or seamlessly launch it from the Data Studio client. For more information on installing the web console, see *Chapter 1*.

4.2 Getting started

Important: Before opening the web console from your web browser, make sure that you have the Adobe Flash Player plug-in installed on the browser.

To open the web console for health monitoring in a web browser:

1. Start the web console server. From Windows, you can start the server from the menu option under *Programs >IBM Data Studio >Data Studio Web Console >Start Web Console Server*, or directly from the Control Panel. If you are using the web console in a Linux or UNIX environment, use the `start.sh` script from the Data Studio web console installation `/bin` directory.

2. Connect to the Data Studio web console server from a web browser, using a URL with the following information:

```
http://<host>:<port>/datatools
```

where <host> is the IP address or hostname of the machine where the Data Studio web console is installed, and <port> is the web console port number that you specified during installation. If this is the first time you are logging in to the web console, you will need to log in as the administrator, using the password you provided during the installation.

Note:
The following are the default URLs for the Data Studio web console:
```
http://localhost:11083/datatools/
https://localhost:11084/datatools/
```

4.3 Identifying databases to monitor

Before you can see any monitoring activity, you need to indicate which databases to monitor, which requires that you *add* the database for monitoring. This is basically creating a connection to the database, similarly to what you have done previously in the Data Studio client. You will add the database using the Databases page of the web console, which you can get to as follows:

- From the Getting Started section of the Task Launcher, click on *Add database connections*

- From the *Open* menu, click on *Databases*, as shown in *Figure 4.1*
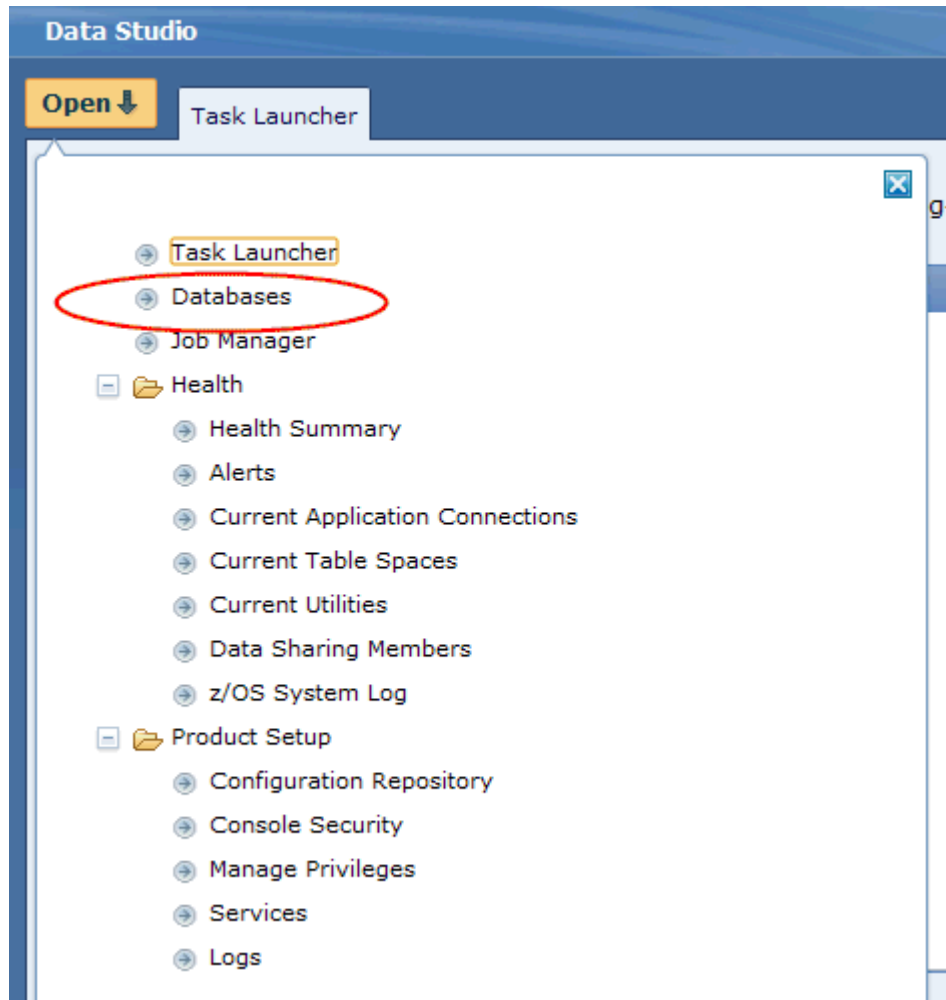
Figure 4.1 – Open menu on the Data Studio web console

From the Databases page, as shown in *Figure 4.2* below, you can add the databases that you wish to monitor by clicking the *Add* button
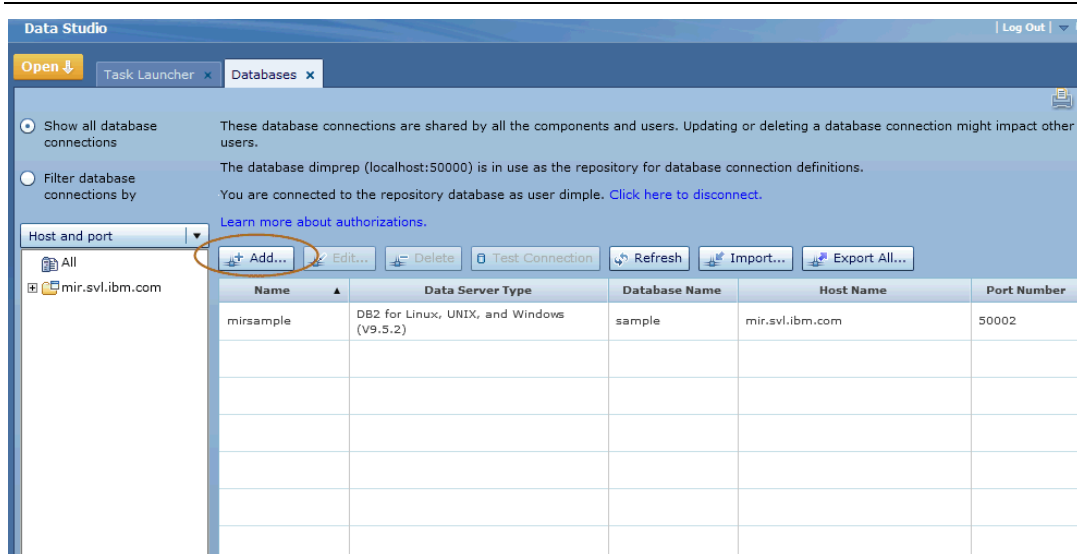
Figure 4.2 – Add databases to monitor

Add the information for the database you want to monitor, as shown in *Figure 4.3*, then click *Test Connection* to validate that everything is working. Then click *OK*.
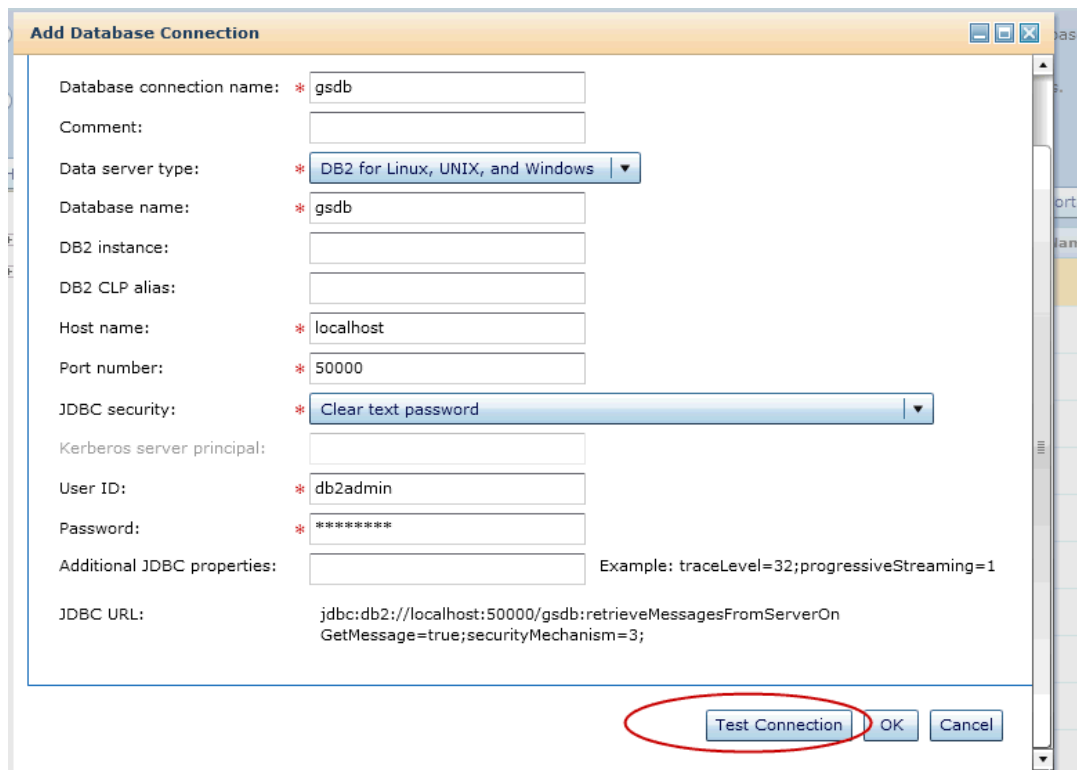
Figure 4.3 – Specifying database connection information

Note:

As you can imagine, for an environment with a large number of databases, it might get tedious to add each connection manually. By clicking on the *Import* button, you can also import a list of databases from comma-separated files. You can use the `DatabaseConnectionsImportCSV.txt` file in the samples folder in the installation directory as an example of importing from a file.

Once a database is added, health monitoring is automatically enabled, and you are ready to monitor your databases! To see the results, go to *Open -> Health -> Health Summary.*

Because the default monitoring interval is 10 minutes, it may take that length of time to see results on the Health Summary, which is described in the next section.

4.4 Overview of the Health Summary page

The web console monitors the health of the databases by periodically querying the database and displaying the health status, including alert conditions, on the Health Summary page.

You can configure the web console to send you an alert when it identifies a problem with a database that it is monitoring. The web console generates alerts based on the thresholds and settings that you specify. When you add a new database for monitoring, a default configuration of thresholds and settings is already specified for you. You can alter the default configurations from the Health alert configurations page as described in *Section 4.5.4*. You can view the generated alerts by clicking the *Open* menu, and then selecting either *Health > Health Summary* or *Health > Alerts*.

The Health Summary provides a grid of summary health information for each monitored database, as shown in *Figure 4.4*.
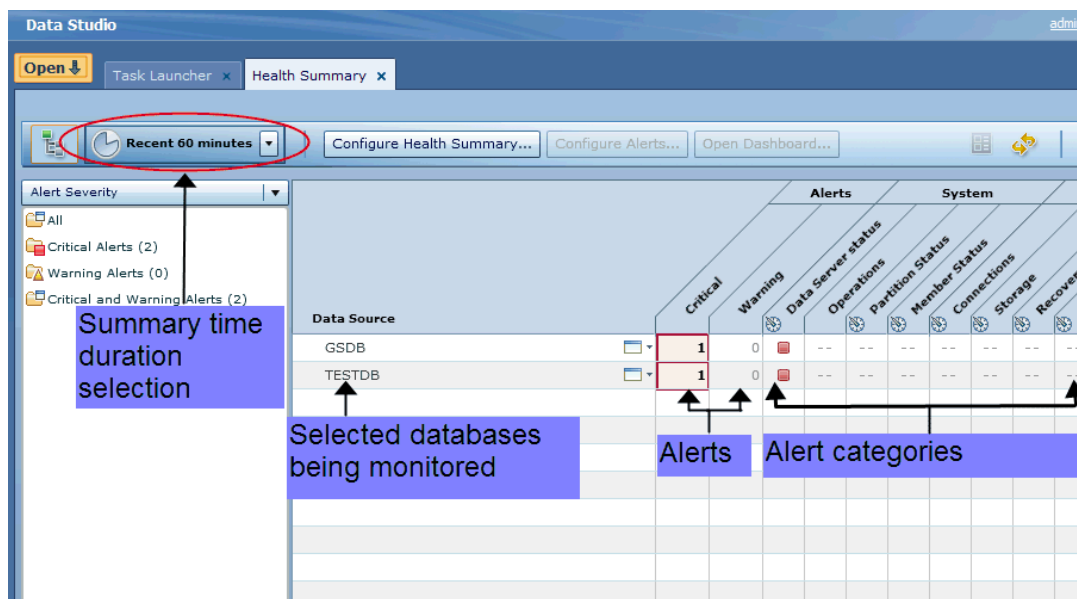
Figure 4.4 – The Health Summary

The alert icons in the grid cells identify a database status (Normal, Warning, or Critical) across categories such as data server, connections, storage, and recovery. A double dash -- in a cell indicates that no alerts were issued in the selected timeframe, as indicated by the time duration selection in the upper left hand corner of *Figure 4.4*. Each icon represents a summary of one or more individual alerts that were encountered for the selected duration of that specific database. For example, if you had two Storage alerts for a database, one Critical and one Warning, then the alert summary icon would identify this as Critical. When you click on the cell, you can drill down to the individual alerts themselves that detail the problems, including any appropriate actions you should take.

You can choose to view alerts for specific time periods by selecting the Summary time pulldown, currently set at 60 minutes in *Figure 4.4*... If you set your recent summary for 60 minutes, as shown in *Figure 4.5*, the *Recent* view will give you a summary of alerts that occurred during the last hour. The refresh summary is set for 5 minutes, which means the status will be checked every 5 minutes. You should set the refresh time to be more frequent than the total summary period. Note that it is possible that no alerts are raised during the most recent period.
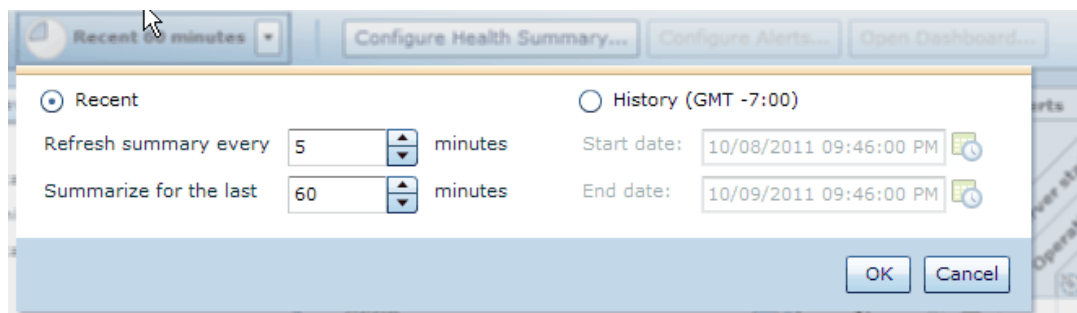
Figure 4.5 – Selecting the time period to display

4.5 Working with alerts

While the alert summary is good for quick detection of possible problems, you will likely want to drill in deeper to find out more information about the condition causing the alert. This section describes:

- Drilling down on an alert from the Health Summary and how to view alerts in a tabular format

- Commenting an alert and share that comment with others

- Configuring alerts, including how to send alerts via e-mail.

4.5.1 Seeing alert details from the Health Summary

You can view alerts generated by the Data Studio web console for the following categories and situations:

- **Data Server Status** - Creates an alert for different data server states including: Available, Unreachable, Quiesced, Quiesce Pending, and Rollforward.

- **Connections** - An alert is generated when the number of connections to the database exceeds the threshold. This alert is disabled by default.

- **Storage** - An alert is generated for the following situations:

    o   Table Space utilization exceeds the threshold

    o   Table Space container utilization exceeds the threshold. This alert is disabled by default.

    o   Table Space container is inaccessible

    o   Table Space is in Quiesced state

    o   Table Space is Offline

- **Recovery** - An alert is generated for the following situations:

    o   Table Space is in Restore Pending or Rollforward Pending state

    o   Table Space is in Backup Pending state

    o   Table Space is in Drop Pending state

    o   Primary HADR is disconnected

- **Partition Status** – An alert is generated when the status of a partition is OFFLINE

- **Status of DB2 pureScale members** – An alert is generated if any of the DB2 pureScale members is in ERROR, STOPPED, WAITING_FOR_FAILBACK, or RESTARTING state.

- Cluster Facility status of DB2 pureScale – An alert is generated if the DB2 pureScale cluster facility is in any of the states - ERROR, STOPPED, PEER, CATCHUP, RESTARTING.

- Cluster Host Status of DB2 pureScale – An alert is generated if the DB2 pureScale Cluster Host status is INACTIVE

For example, look at the Data Server Status Alert category. When the GSDB database is available, the Data Server status summary cell has a green diamond, as shown in *Figure 4.6*.
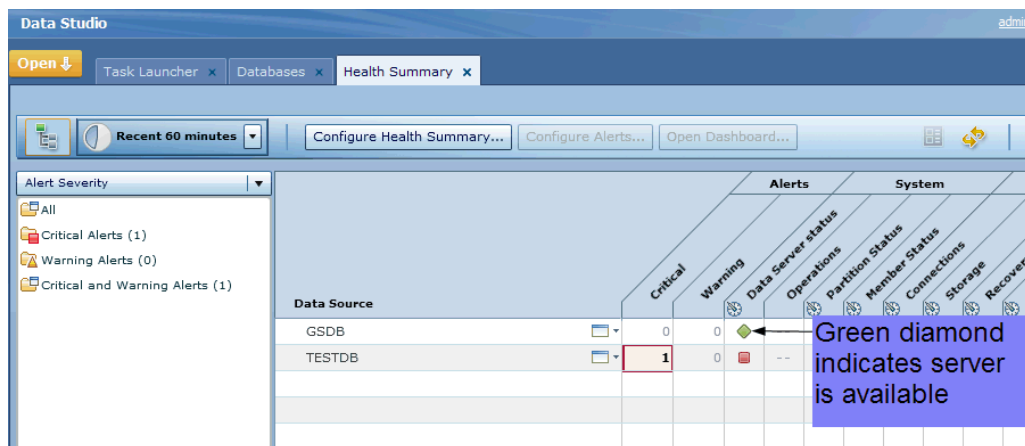
Figure 4.6 – The data server status indicator.

Now let's say the database GSDB has been quiesced. With the default configuration settings, a critical alert will be generated for the GSDB database as shown in *Figure 4.7*
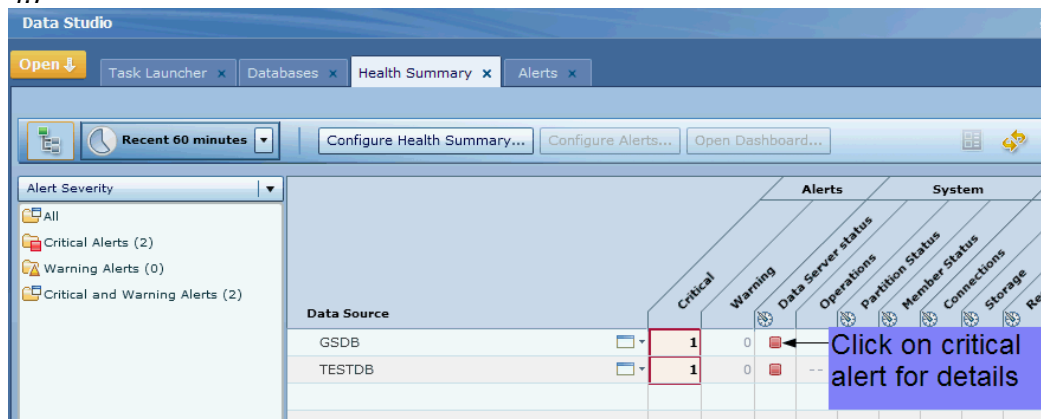
Figure 4.7 – Critical data server status.

You can view more information about the alert by clicking on the red icon under Data Server Status. This brings up the details shown in *Figure 4.8*.

Figure 4.8 – Data server status details.

Figure 4.8 also shows the list of individual alerts for the Data Server Status category. The Data Server Status is a special category that represents the status of the monitored database. It displays the status as green when the database is available and reachable. The *Start Time* and *End Time* columns show when the alerts originated, and if and when they ended. An alert without an end time indicates that the problem is still ongoing. In the example above, the alert is caused because the database is quiesced. When the database is unquiesced, the next monitoring cycle for this database will identify this situation, and the critical alert will be closed.

4.5.2 Displaying a tabular listing of alerts - the Alert List page

While the Health Summary page provides an aggregated view, the Alerts List page provides a detailed view of the individual alerts and lets you review either recent alerts or even go back in tie using a historical view.

There are two ways to open the Alert List page:

- From the *Open* menu , click on *Health > Alerts*

- From the Health Summary page, double click the alert icon, and from detailed alert status click the *Open Full List* button (circled in *Figure 4.8*). .

An example Alerts List page is shown in *Figure 4.9*. From this page, you can configure health alerts, configure to send alerts by e-mail, and add comments to alerts.
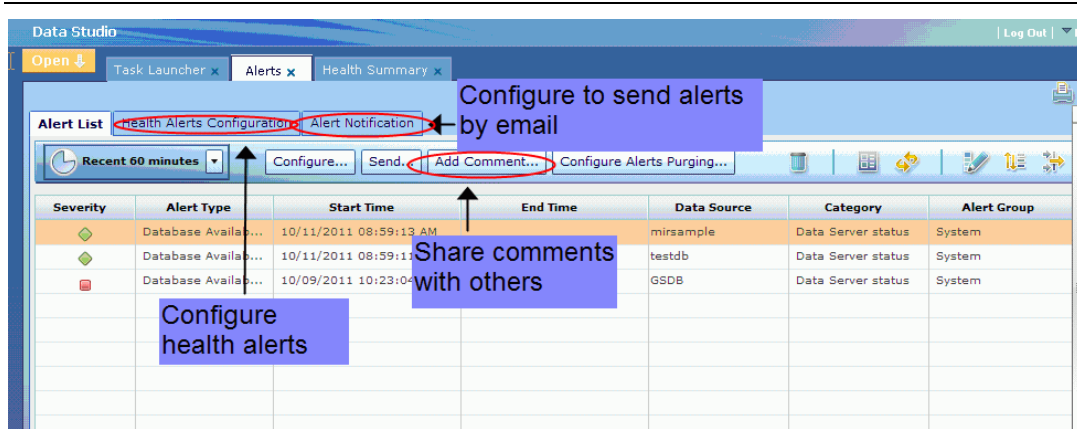
Figure 4.9 – The alert list

4.5.3 Sharing alerts with others

You can add comments for an alert and share them with other users to help you keep track of your own investigations or responses to a problem. To get to the comments field shown in Figure *4.10*, select an alert on the Alerts List page and then click the Add *Comment* button.
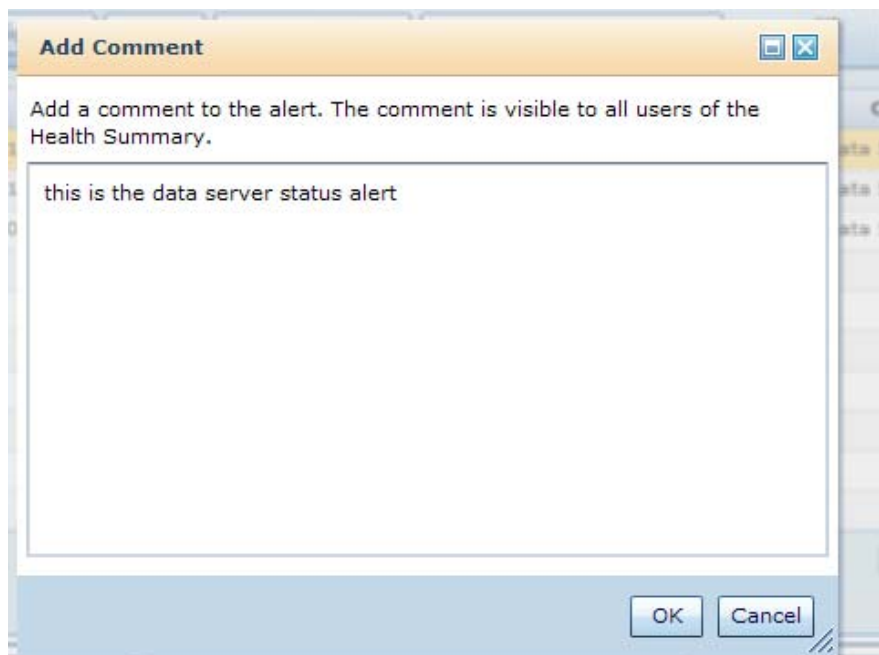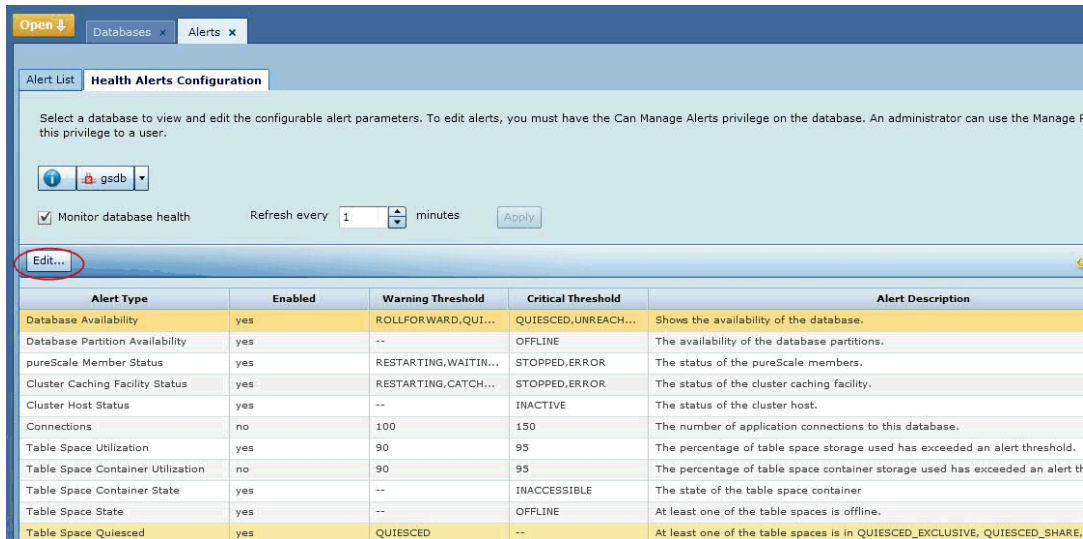
Figure 4.10 – Add a comment to an alert.

4.5.4 Configuring alerts

By default, health monitoring is enabled automatically for all databases that are added from the Databases page. The web console checks the state of each database every 10 minutes, but you can disable monitoring entirely, or change the frequency of monitoring

from the Health Alerts Configuration page, shown in *Figure 4.11*. The currently selected warning and critical threshold values are also listed once you select a database.

Figure 4.11 – Configuring health alerts.

Click *Edit* to modify the default threshold values, or to disable alerts for an alert type. Editing alert configuration settings is restricted to users who have the Can Manage Alerts privilege for that database, as described in the Database Privileges section in *Appendix B*. For example, look at the editing of the Database Availability alert type configuration as shown in *Figure 4.12*.
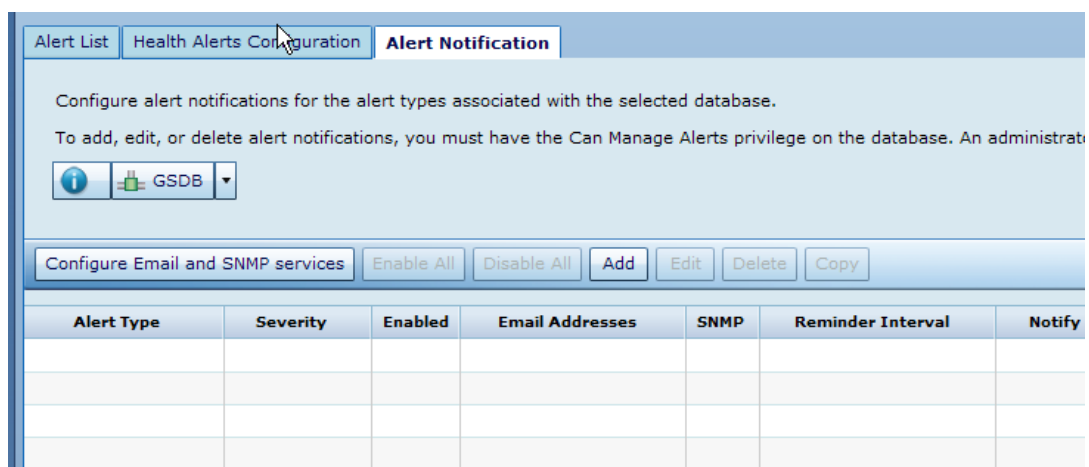
Figure 4.12 – Setting alert parameters.

4.5.5 Configuring alert notifications

You can set up to be notified for certain alerts types and a database on the *Alert Notification* page via email or SNMP traps. You can open the Alert Notification page by gong to *Open-> Health > Alerts* and clicking on the *Alert Notification* tab, as shown in *Figure 4.13*.

Figure 4.13 – Alert notifications

To add an alert notification:

1. Before you can add an alert notification, you need to configure the email/SNMP service. To configure the email or the SNMP service, select the Services page under the *Open* menu. Select the email or the SNMP service from the grid and click on *Configure*, as shown in *Figure 4.14.*

Services	Status	Users	Last Updated
Email service			07/20/2011 03:25:58 PM
AlertNotificationLogger		Default	09/23/2011 03:24:12 PM
HealthSnapshotLogger		Default	04/29/2011 02:26:54 PM
System logging service		Default	04/29/2011 02:26:33 PM
JobManagerLogger		Default	05/21/2011 11:27:15 PM
SNMP		Default	10/11/2011 02:51:07 PM
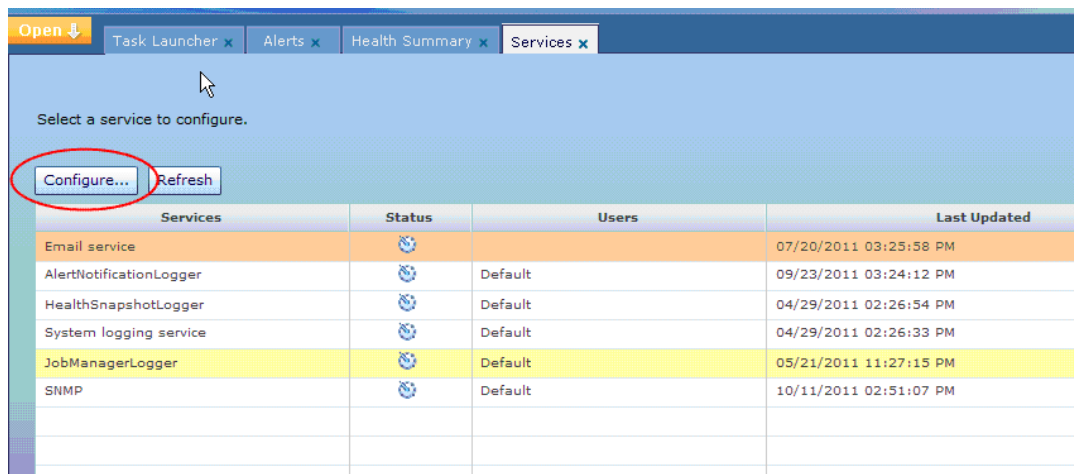
Figure 4.14 – Configure the email service

2. Select a database from the drop down, then click the *Add* button and choose the alert types, as shown in *Figure 4.15*.
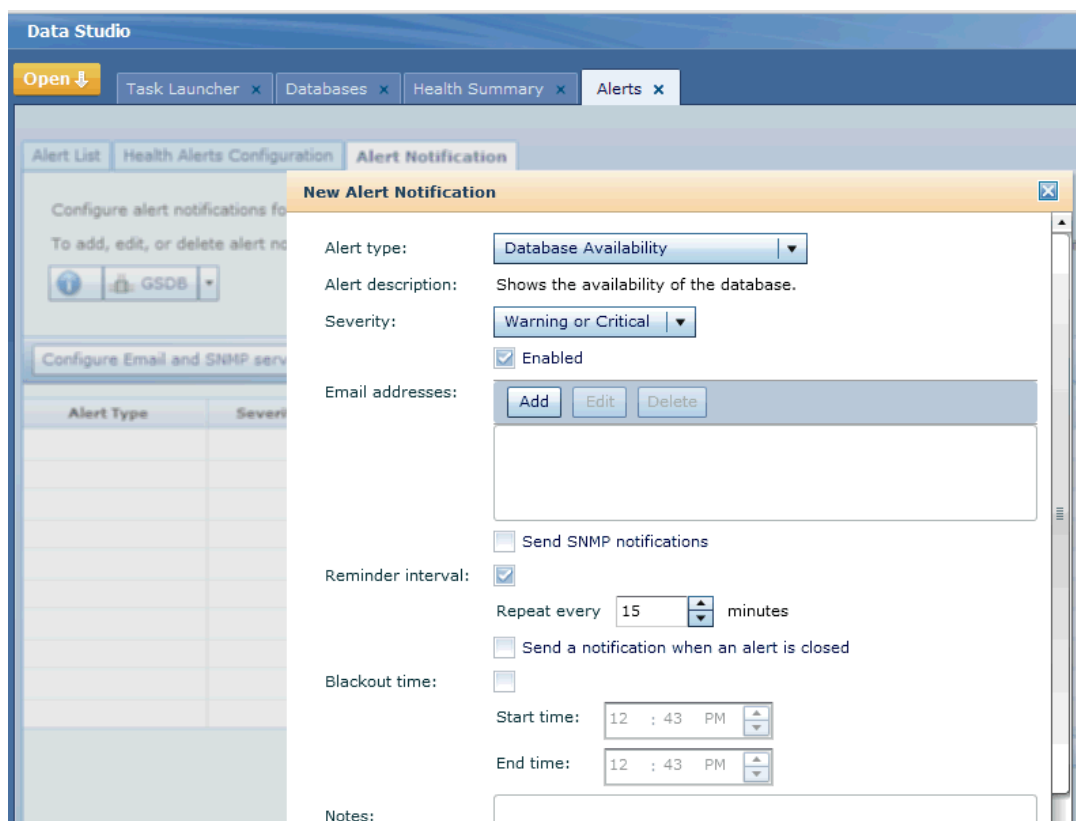
Figure 4.15 – Edit the alert parameters

4.6 Displaying current application connections

The Current Applications Connections page, as shown in *Figure 4.16*, lets you get a list of the applications that are currently connected to the selected database.

To get to this page go to *Open -> Health -> Current Application Connections*

From this page, you can view information about these connections, such as the Name, Idle time, and User authorization. If a connection has been idle for a period of time, you can choose to disconnect, or you can force an application connection off.

> **Note:**
>
> For more information about forcing applications, see
> http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0001951.html
>
> For more information about disconnect, see
> http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.sql.ref.doc/doc/r0000943.html

Note that in the Current Applications Connections page, you will need to connect to the selected database with your credentials to retrieve data and perform other actions, like

forcing applications. If you do not have the right privileges on the database, then the operation will fail. The prompt screen provides you with the option to save your credentials for subsequent logons to make this easier.
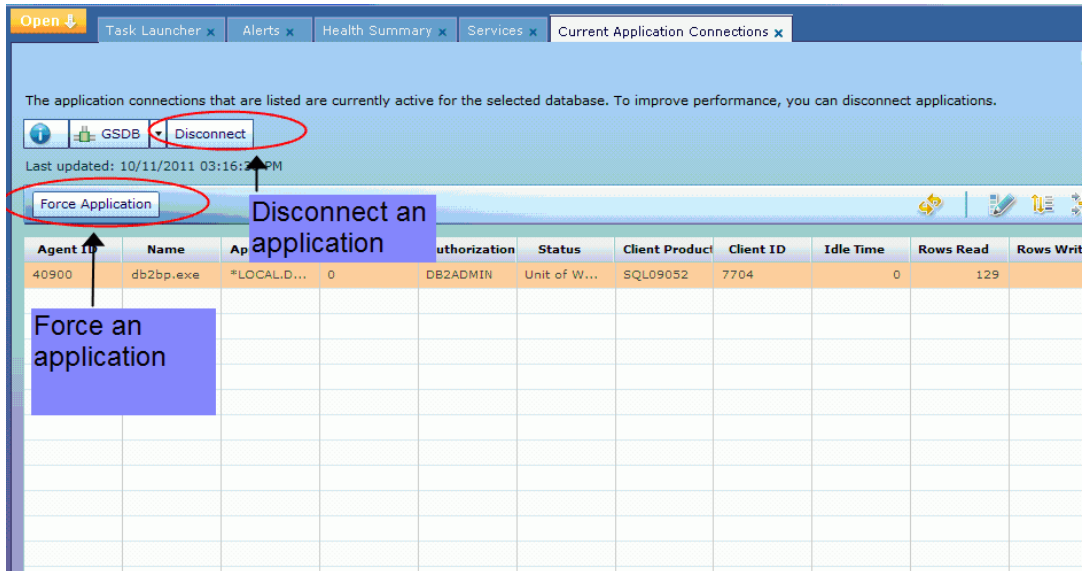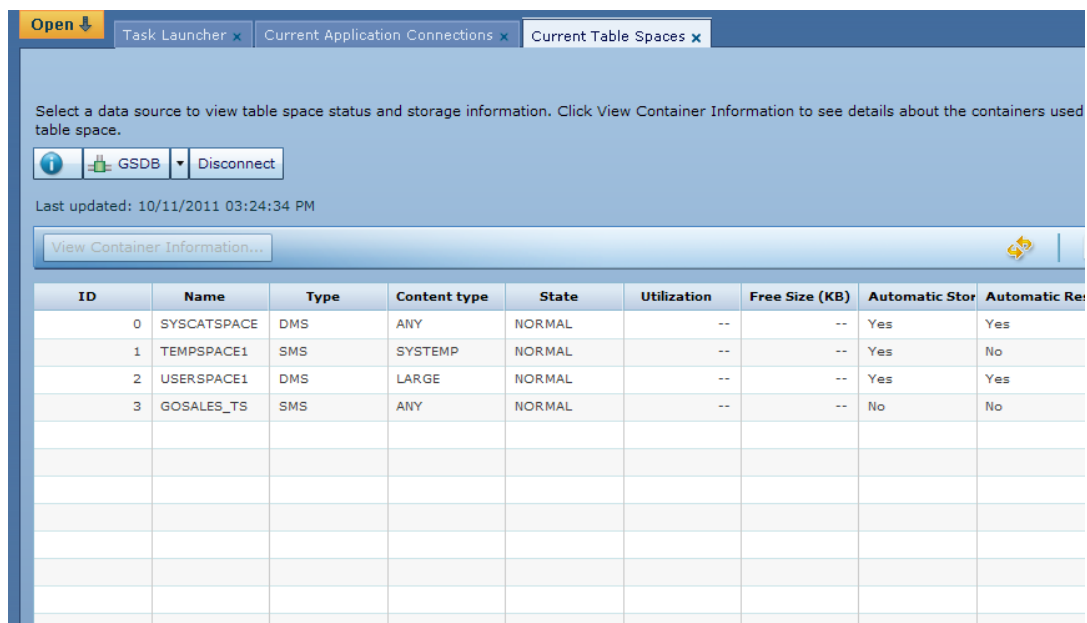
Figure 4.16 – Listing the current applications that are active for a database.

4.7 Getting information about current table spaces

The Current Table Spaces page, as shown in *Figure 4.17*, displays live information about all the table spaces in the selected database. You can also view the table space container information for a selected table space.

To get to this page go to *Open -> Health -> Current Table Spaces*

Note that in the Current Table Spaces and Current Utilities pages, you will need to connect to the selected database with credentials to retrieve data and perform other actions, like forcing applications. If the user ID you use to connect does not have the right privileges on the database, the operation will fail. The prompt screen provides you with the option to save your credentials for subsequent logons to make this easier.

Figure 4.17 – Current table spaces

4.8 Display current utilities

The Current Utilities page, as shown in *Figure 4.18*, displays the status of any utilities currently running in the selected database, such as RUNSTATS or BACKUP.

To get to this page go to *Open -> Health -> Current Utilities*

Figure 4.18 – Current utilities

4.9 Accessing Health Monitoring features from the Data Studio client

You can use the Data Studio client to access the health monitoring features in the Data Studio web console.

4.9.1 Configuring the Data Studio web console

Before you can open the web console from the Data Studio client, you must configure it from the preferences page.

To configure the Data Studio web console for the Data Studio client:

1. From the client, open the preferences page by clicking *Window > Preferences > Data Management > Data Studio Web Console*.

2. When you access a health monitoring feature, if you have not previously entered the URL, your login credentials, or selected the *Save password* checkbox, then you will be prompted to enter them to login into the server as shown in *Figure 4.19*.
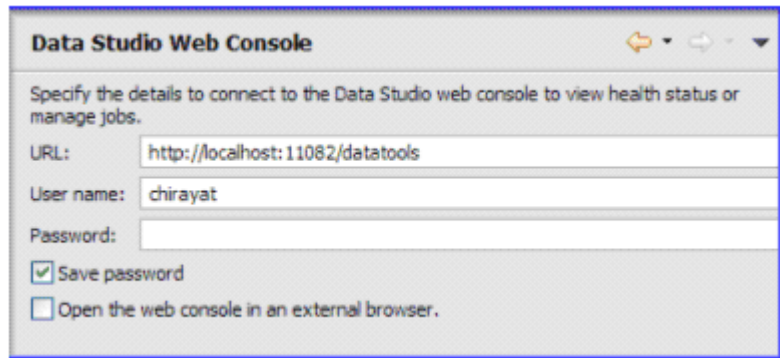
Figure 4.19 – Configuring access to the Data Studio web console from the Data Studio client

To identify the web console server, enter the URL. It is the same as the URL used with the web browser, and has this format:

```
http://<host>:<port>/datatools
```

where <host> is the IP address or hostname of the machine where the Data Studio web console is installed, and <port> is the web console port number that you specified during installation. You should also fill in the appropriate user name and password,

Optionally, you can select the *Save password* checkbox to save the password for subsequent logins to the web console server. If the *Save password* option is cleared, then whenever a Monitoring feature is launched, you must enter your credentials again.

4.9.2 Opening the Health Monitor from the client

You can open the various health monitoring tasks from either the Data Studio client Task Launcher, from a selected database in the Administration explorer, or from the Administration explorer view toolbar, as shown in *Figure 4.20*.

When launched from the Administration Explorer, the selected database serves as the content when the monitoring task is launched. For the Application Connections, Table Space Usage, and Utilities tasks, such a launch causes the same database to be pre-selected in the monitor page.

- From the Monitor tab of Task Launcher, if you select any one of the view health summary, view alerts list, view application connections or view table space storage usage monitoring tasks, the corresponding page will open up in the Data Studio web console Monitor tab.

- From the Administration Explorer, right-click on a database to find the Monitor menu group. Under the *Monitor* menu you find the following health monitoring options: Health summary, Alerts, Application Connections, Table spaces, and Utilities.

- From the *View* menu option in Administration Explorer, the same Monitor menu is available.
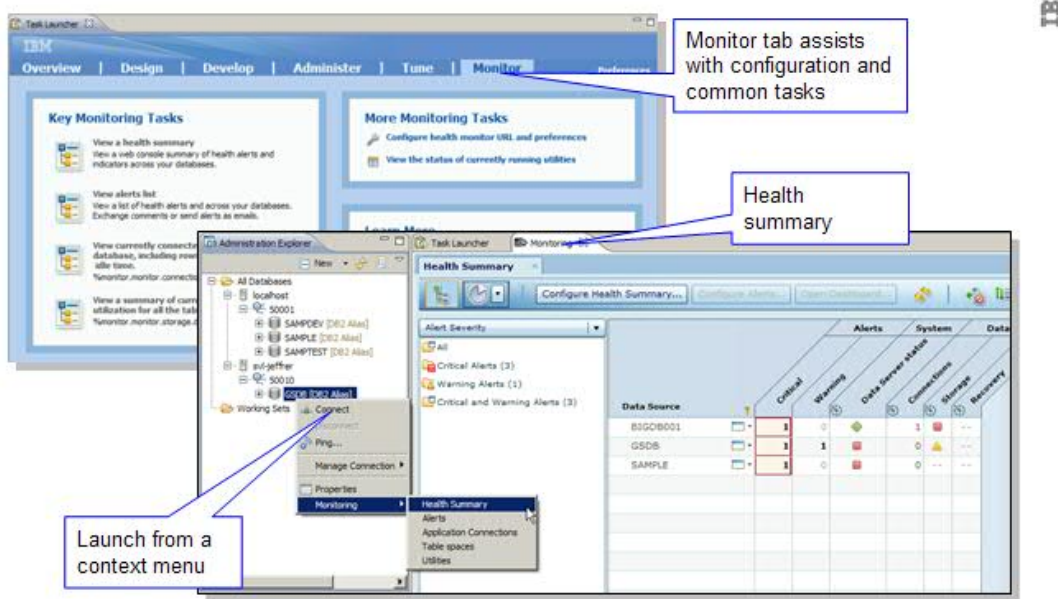
Figure 4.20 – The embedded Data Studio web console interfaces.

4.10 Exercises

In this section, you will create a variety of error scenarios so that you can get a better understanding of how this product works, and how you can take advantage of its features. For the purposes of this exercise, we recommend that you use the GSDB sample database as your test database. For more information on how to install the GSDB database, see *Chapter 1*.

To keep this exercise simple, you will create these error scenarios in a controlled manner, and lower thresholds to more quickly cause situations to be flagged as alerts. This section includes scenarios for the following alert types:

- Database Availability - This generates a data server status alert.

- Connections - This generates a connections alert.

- Table Space Quiesced - This generates a storage alert.

4.10.1 Adjust the monitoring frequency

First, navigate to the *Health Alerts Configuration* page in the Data Studio web console and select a database. You will see the default refresh value is 10 minutes, which means that the monitoring of this database will happen every 10 minutes. For this exercise, change this value to 1 minute, so you will not have to wait as long to see the alerts being generated.

4.10.2 Adjust the page refresh rates

Navigate to the *Health Summary* page, and find the option to change refresh intervals. Click the down arrow next to the *Recent xx* minutes label. A dialog appears that lets you change how often the Health Summary pages will auto-refresh itself to display recently found alerts. The default is set to 5 minutes. For this exercise, change it to 1 minute to minimize testing wait times.

4.10.3 Database availability

The Database Availability Alert checks the following status types for a database: Normal, Quiesce Pending, Quiesced, Rollforward, and Unreachable. By default, a warning alert is generated if the database is in Quiesce Pending or Rollforward state. A critical alert is generated if the database is in Quiesced or Unreachable state.

The following error scenario will put the database in Quiesced state, so that a critical alert will be generated.

1. Open a DB2 command window.

2. Connect to the database with the following command:

    db2 connect to GSDB user db2admin using db2admin

3. Quiesce the database with the following command:

    db2 quiesce db immediate

4. Navigate to the *Health Summary* page. Wait for one minute to see the *Data Server Status* change from Normal (green) status to Critical (red) status. The *Critical Count* should also now be raised by 1. Click the *Critical Status* icon, or the *Critical Count Number*, to get more information on the alert.

4.10.4 Updating the alert configuration

In this section, follow these steps to generate a warning for a database in Quiesced state, instead of the currently defined critical alert.

1. Navigate to the *Health Alerts Configuration* page.

2. Select the database that is in Quiesced state.

3. Highlight the *Database Availability* row and click *Edit*.

4. Ensure that the *Enabled* checkbox is selected.

5. In the *Critical* section, clear the *Quiesced* checkbox.

6. In the *Warning* section, select the *Quiesced* checkbox.

7. Click *OK*.

After the monitoring refresh occurs on the Health Summary page, you will see the Data Server Status change from Critical (Red) to Warning (Yellow). The Warning Count should now be raised by 1. Do not be alarmed if the Critical Count is still showing 1. The counts are a cumulative (summary) history of how many alerts have occurred within the past xx minutes. If you select the critical alert, you will also see that there is now an end time for it, meaning that the alert has closed. The start-end time duration is the time during which this problem was present.

Finally, follow these steps to un-quiesce the database:

1. Open a DB2 command window.

2. Connect to the database with the following command:

    `db2 connect to GSDB`

3. Unquiesce the database with the following command:

    `db2 unquiesce db`

After the monitoring refresh occurs on the Health Summary page, you should see the Data Server Status change from Warning (yellow) back to Normal (green). The previous warning alert should also end. After the situation has returned to normal, you can still determine that the database was quiesced because this information is reflected in the summary counts, as well as in the Alert List page for that time duration.

4.10.5 Connections

The Connections Alert warns you when there are too many connections to a database at the same time. By default, the Connections Alert is turned off, but preset to generate a warning alert if the number of connections detected is greater than or equal to 100, and a critical alert if greater than or equal to 150. Typically, you need to decide what number constitutes a critical level, and what constitutes a warning level. What may be a perfectly reasonable limit for one database may not be so for another database.

For this exercise, follow these steps to lower the thresholds for the alert, so you don't have to create over 100 connections to trigger it.

1. Navigate to the *Health Alerts Configuration* page.

2. Select the database that you want to use for this scenario.

3. Highlight the *Connections* row and click *Edit*.

4. Select the *Enabled* checkbox (if it is not already selected).

5. Lower the values for *Warning* threshold to **1**, and *Critical* threshold to **5**.

6. Click *OK*.

Connect to this monitored database from the DB2 command line, or from the Data Studio client, and leave the connections active. Once the monitoring refresh occurs on the Health Summary page, you should see the Connections status change from No Alerts to either Critical (red) or Warning (yellow), depending on how many connections are currently active on that database.

If you close the connections to below the threshold value, this should close the alerts. Note that the critical or warning summary icon would still be present to indicate that there had been problems in that selected duration, but that these alerts now have an end time. This is unlike the Status summary icon with the Data Server Status alert category.

Remember to reset this alert parameter (or disable the alert) to whatever is appropriate for your database when you finish with this exercise

4.11 Summary

In this chapter, you've gotten a detailed look at the health monitoring features in Data Studio web console, and learned how to alter configurations to better suit your monitoring requirements for each database. You also learned how to try out some of the alerting features of Data Studio web console, as well as how to invoke the monitoring features from within Data Studio client.

4.12 Review Questions

1. Can you modify the threshold for the alerts supported by Data Studio web console? If so, how do you modify them?

2. List the different alerts supported by Data Studio web console.

3. How can you share the alerts generated by the web console with colleagues?

4. How can you open the Data Studio web console from Data Studio client?

5

Chapter 5 – Creating SQL and XQuery scripts

In this chapter, we will describe some basic data development tasks using SQL and XQuery scripts in Data Studio.

The SQL and XQuery editor helps you create and run SQL scripts that contain SQL and XQuery statements. This chapter describes how to use some of the features of the editor to help you develop your SQL scripts more efficiently. The features in the editor are available for all the data servers that are supported in the workbench, except for any that are specifically noted as not supported. The editor includes the following capabilities:

- Syntax highlighting
- SQL formatting
- Content assist
- Statement parsing and validation with multiple version-specific database parsers
- Semantic validation

You can run scripts serially against multiple database connections and choose an execution environment, such as JDBC or the command line processor (CLP). The editor provides you with flexibility by letting you change special registers to modify the current schema and current path. In addition, you can export SQL scripts from and import SQL scripts to the editor.

Through the editor, you can also schedule scripts for execution using the Job Manager described in *Chapter 6* and access the query tuning capabilities of Data Studio, described in *Chapter 7*.

5.1 Creating SQL and XQuery scripts: The big picture

In a nutshell, SQL and XQuery scripts assist with data development, which involves the development, testing and deployment of database objects. You can store these scripts in different types of data projects within the *Data Project Explorer* view. The primary projects to store SQL and XQuery scripts in Data Studio are the *Data Development Project* and the *Data Design Project. Figure 5.1* shows both projects.
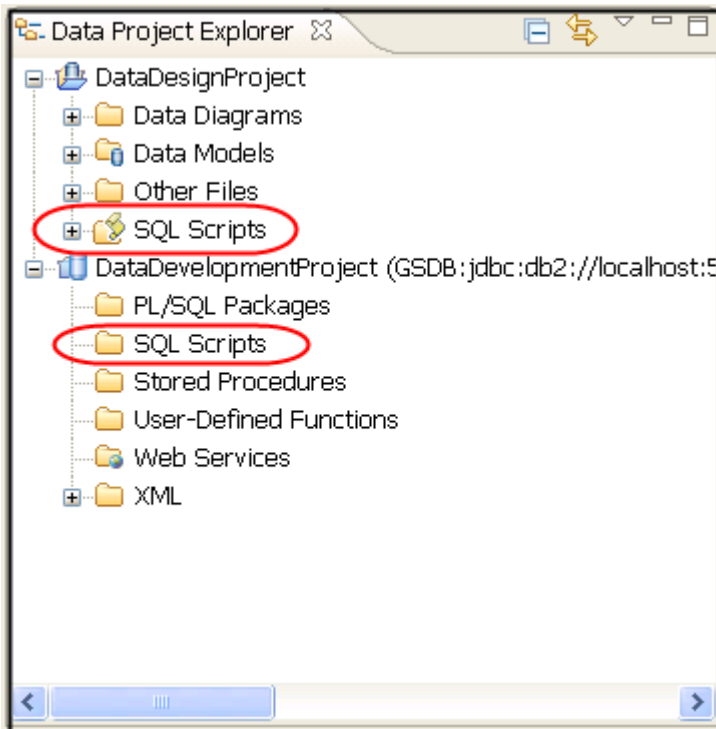
Figure 5.1 – The Data Design and Data Development projects can be used to store SQL scripts

5.1.1 Creating a Data Development project: SQL and XQuery scripts

Previously, we described the many perspectives available in Data Studio, including the Data perspective. In this chapter, we will use the Data perspective for our data development tasks. We will also focus on SQL scripts.

The first step is to create a Data Development project. You use Data Development projects to develop and test database artifacts such as PL/SQL packages, SQL scripts, stored procedures, user-defined functions, data web services, and XML documents. Related artifacts such as Web Services Description Language (WSDL) documents, XML schemas, XML style sheets, and XML mappings are all stored in the project.

To create a new Data Development project, make sure you are in the Data perspective, and then select *File -> New -> Data Development Project*, as shown in *Figure 5.2*.
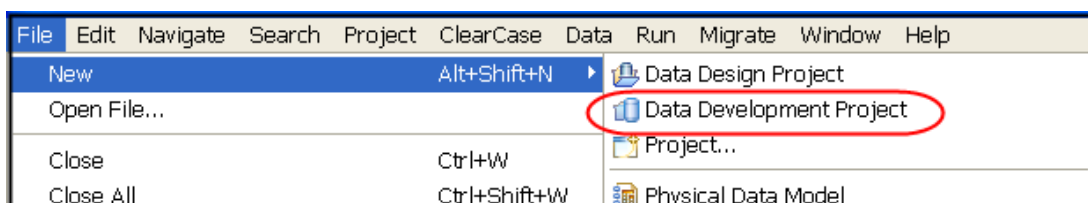
Figure 5.2 – Creating a new Data Development Project

The *New Data Development Project* wizard, as shown in *Figure 5.3,* will appear, guiding you through the steps necessary to create a Data Development project. In this first page of the wizard, insert the project's name. Call it **DataDevelopmentProject**, as this is the project name we'll use throughout this chapter.
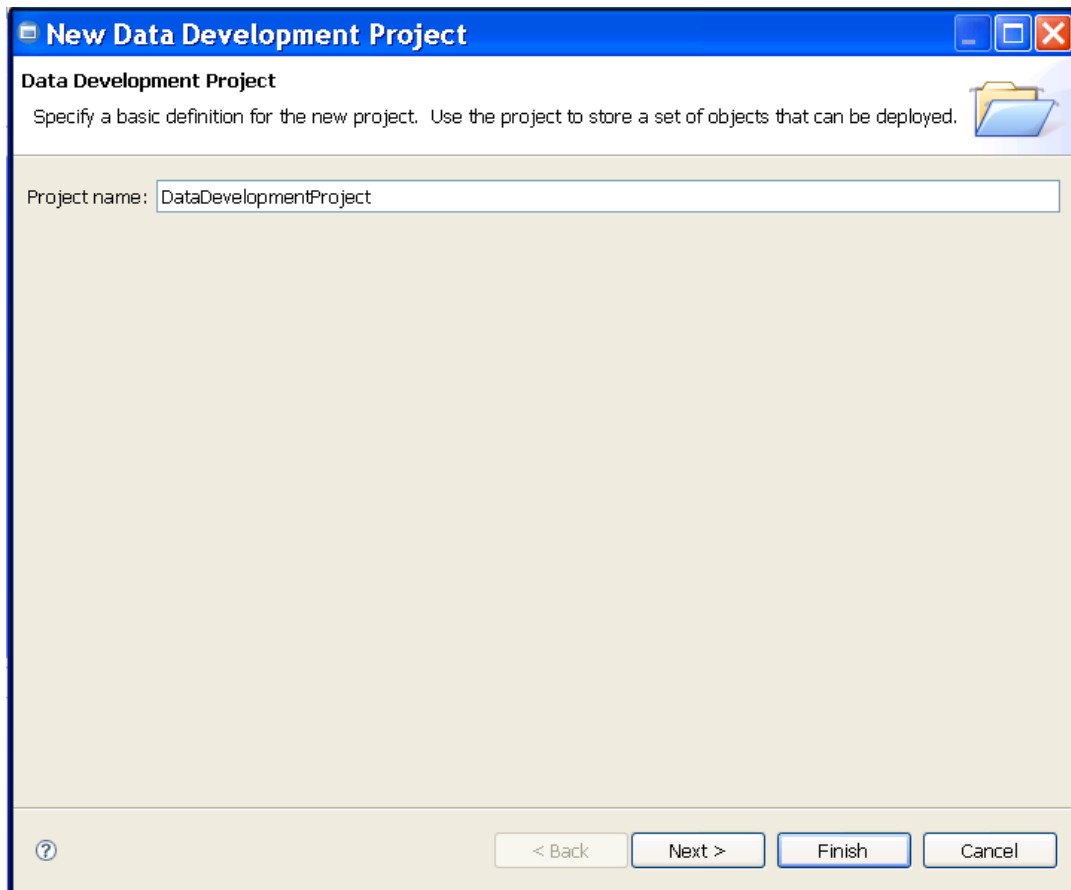
Figure 5.3 – Specifying the name for the new Data Development project

The next page on the wizard, as shown in *Figure 5.4*, lets you select the database connection that will be associated with the project. You can select an existing connection or create a new one by clicking the *New…* button. In our case, we will select the *GSDB* database connection.
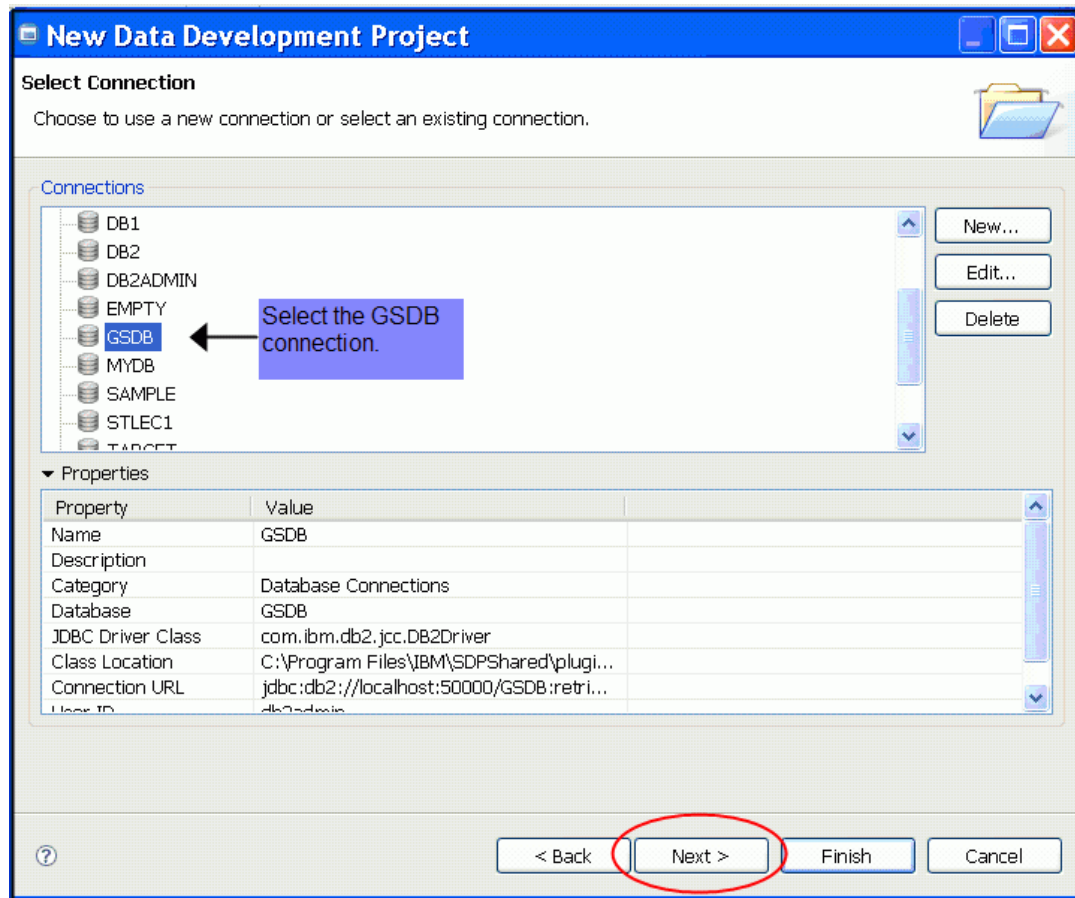
Figure 5.4 – Selecting a database connection

After a database connection is selected, you can either click *Next* or *Finish*. Clicking *Next* will allow you to specify some application settings like default schema and default path, as shown in *Figure 5.5*. If you decide to click *Finish* instead, default values will be used for these settings.

Figure 5.5 – Default Application Process Settings

Here are the descriptions of the fields shown in *Figure 5.5*:

- The *Default schema* setting is used to set the database CURRENT SCHEMA register when deploying and running database artifacts like SQL scripts, stored procedures, and user-defined functions. The CURRENT SCHEMA register is used to resolve unqualified database object references.

- The *Default path* setting is used to set the database CURRENT PATH **r**egister when deploying and running database artifacts. The CURRENT PATH register is used to resolve unqualified function names, procedure names, data type names, global variable names, and module object names in dynamically prepared SQL statements.

Note:

The application process settings available on this wizard depend on the database

> server you are connecting to. The example shown lists the settings available for
> DB2 servers.

The default values for schema and path are based on the database server and the
connection user ID. Since all of the tables we will be using from the *GSDB* database are in
the *DB2ADMIN* schema, you should change the application settings to include that schema
in the path and use it as the default schema, too.

You can do this by clicking in the drop down list for default schema and selecting
DB2ADMIN, as shown in *Figure 5.6*.
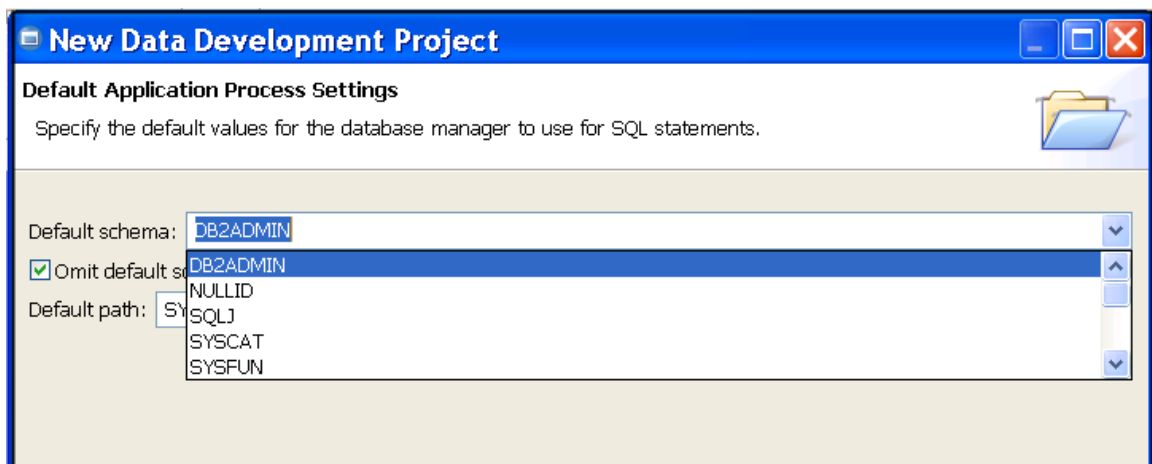
Figure 5.6 – Selecting a value for Default schema

One useful thing about Data Studio is that it provides content assist in several different
contexts. As you've just seen, it lists all the existing schemas in the database so that you
can just select one from a drop down list for the default schema. Content assist is only
available when you have an established connection to the database, either in live or offline
mode.

You also need to change the current path to account for the *DB2ADMIN* schema, as shown
in Figure *5.7* –
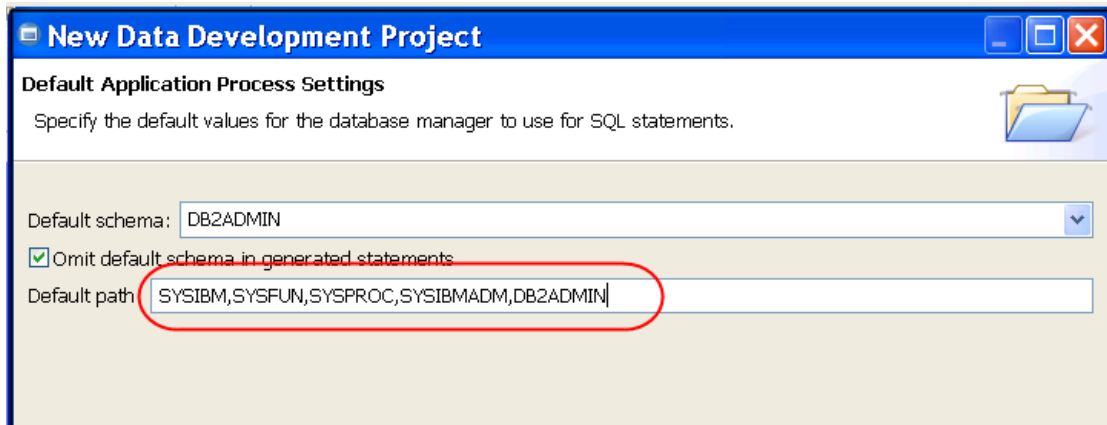
Figure 5.7 – Default path for current path value

Now that you have specified your project's application settings, click *Finish,* and the new project will be created in your workspace, showing up in the *Data Project Explorer* view, as shown in *Figure 5.8.*

Figure 5.8 – Data Development project

Figure 5.8 also shows that Data Development projects contain subfolders that can be used to create and store database artifacts that you develop. The subfolders of a project depend on the database server product and version used. For example, the *PL/SQL Packages* subfolder is displayed only for projects associated with a DB2 for Linux, UNIX and Windows server Version 9.7 or above.

5.1.2 Creating a Data Design project

Next we will create a Data Design project. You use Data Design projects to store data modeling objects like logical data models, physical data models, domain models, glossary models, and SQL scripts, including DDL scripts. Other file types such as doc files, text files, presentations, or spreadsheets are also stored in the project.

To create a new Data Design project, make sure you are in the Data perspective, and then select *File -> New -> Data Design Project*, as shown in *Figure 5.9.*
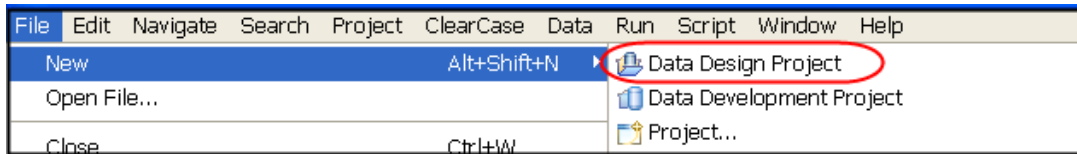
Figure 5.9 – Creating a new Data Design Project

The *New Data Design Project* wizard, as shown in *Figure 5.10*, will appear, guiding you through the steps necessary to create a Data Design project. In this first page of the wizard, insert the project's name. Call it **DataDesignProject**, as this is the project name we'll use throughout this chapter.
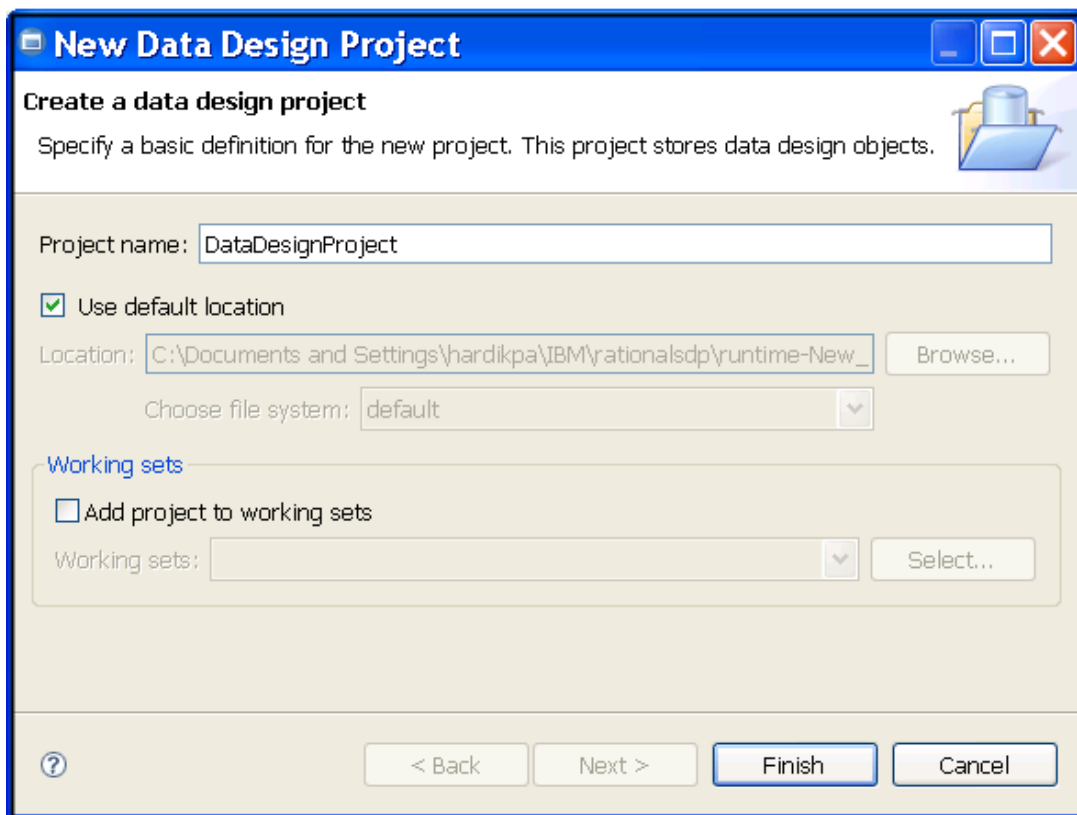
Figure 5.10 – Specifying the name for the new Data Design project

Since you do not need a database connection to create a Data Design project, no additional steps are needed. Clicking the *Finish* button will create the new project in the *Data Project Explorer* view, as shown in *Figure 5.11*.
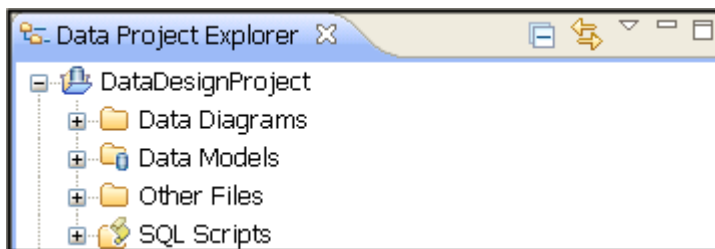
Figure 5.11 – Data Design project

Figure 5.11 also shows that Data Design projects contain subfolders that can be used to create Data Diagrams and Data Models. In this chapter we will focus on just the SQL Scripts folder.

5.1.3 Creating new SQL and XQuery scripts: Using Data Projects

Data Studio provides development of SQL scripts for all database servers it supports. In this section, you will learn how to create a new SQL script.

To create a new SQL script, right click on the *SQL Scripts* folder and select *New -> SQL or XQuery Script*, as shown in *Figure 5.12.*
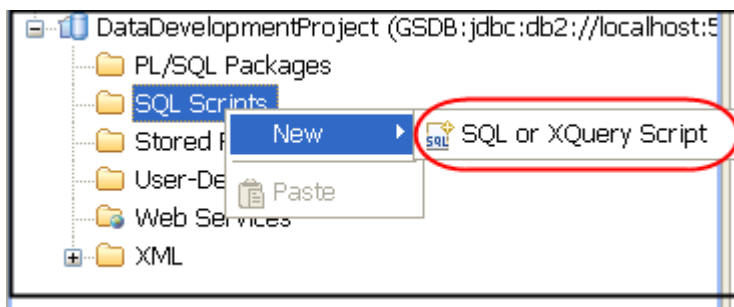
Figure 5.12 – Creating a new SQL or XQuery Script

Note:

Development of XQuery scripts is supported by Data Studio when connecting to a server with XQuery support, such as DB2.

Selecting the *SQL or XQuery Script* option will bring up the *New SQL or XQuery Script* wizard, as shown in Figure 5.13 –
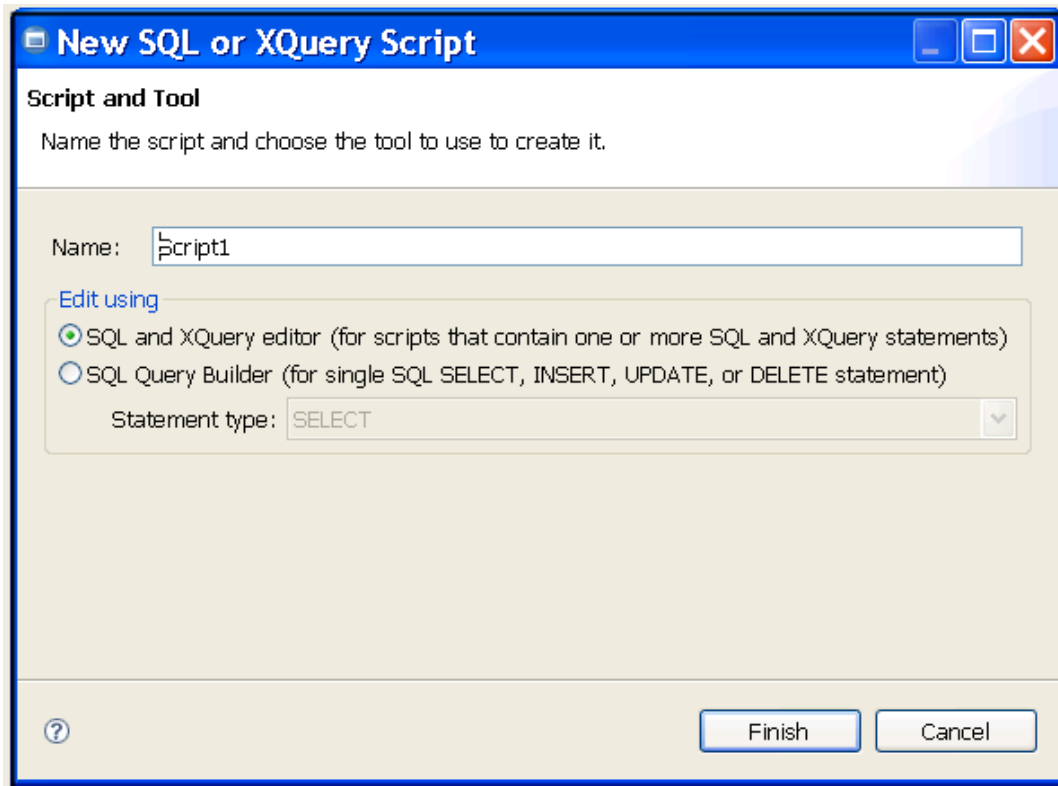
Figure 5.13 – Creating a new SQL script using the SQL or XQuery editor

You can create SQL or XQuery scripts in two different ways: by just opening an empty SQL and XQuery editor (first radio button option in *Figure 5.13 –*); or by using the SQL Query Builder. (The SQL Query Builder does not support XQuery.) The recommended way to develop SQL or XQuery scripts in Data Studio is by using the SQL and XQuery Editor. In this book, we describe both approaches; first with the editor and then achieving the same result using the SQL Query Builder.

To create an SQL script using the editor, select the *SQL and XQuery editor* option on the first page of the *New SQL or XQuery Script* wizard, as shown in *Figure 5.13.*

Clicking *Finish* will quickly bring you to the SQL and XQuery editor for the newly created `Script1.sql,` as shown in *Figure 5.14.*
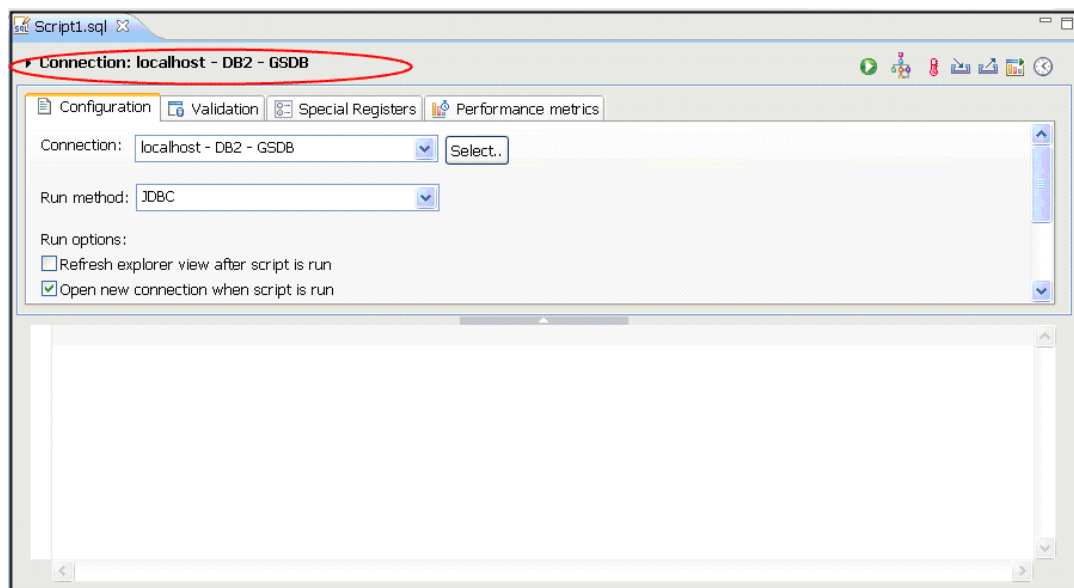
Figure 5.14 – SQL and XQuery editor for Script1.sql

Figure 5.14 shows the empty script in the editor. At the top of the editor is the editor toolbar, which displays the database connection information. In this case, *Connection: localhost-DB2-GSDB*. You can show or hide the detailed database connection information in the editor toolbar by clicking the control arrow next to the connection, as shown in *Figure 5.15*.
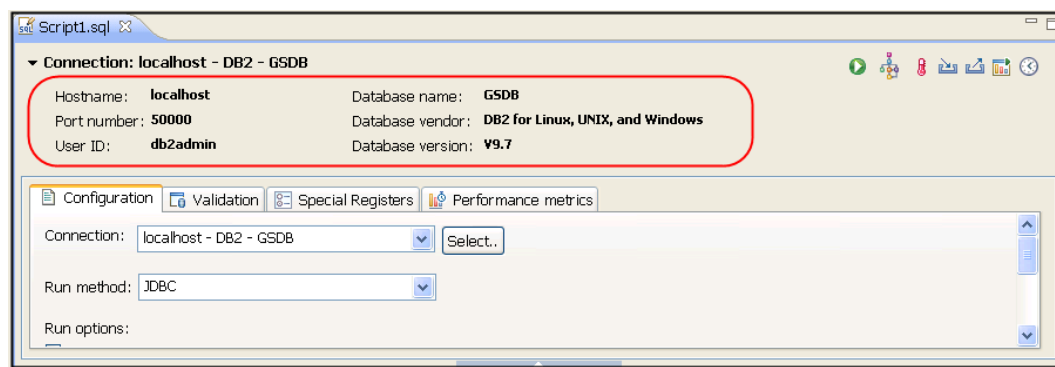
Figure 5.15 – Database connection information

Below the editor toolbar is the *Command* pane, which is a tabbed window that you can show or hide while working in the editor that controls the configuration, validation, special registers, and performance metrics for your scripts. We will discuss these pages in depth in the next sections of the chapter.

5.2 Changing the database connection

As we saw in the last section, the editor toolbar shows information about the current connection to the associated database. When you are developing your SQL script, the database connection determines the information that is available in content assist, and the vendor and version of the database determines the syntax validation of your SQL statements. The database connection also might affect the results when you run the SQL script. You can change the database that the script is connected to by clicking the *Select* button on the *Configuration* tab, as shown in *Figure 5.16*.
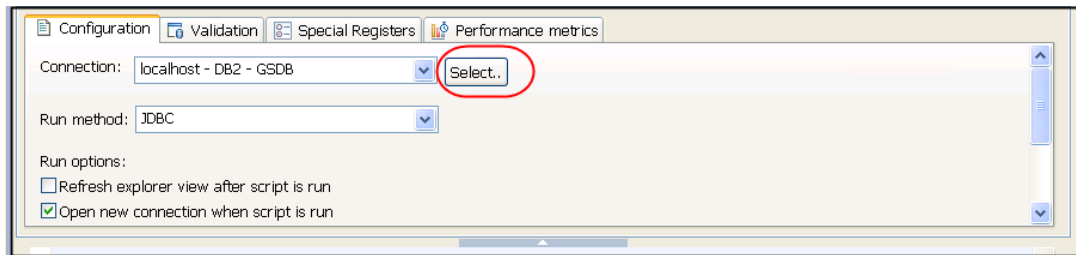
Figure 5.16 – Configuration tab

If you have connections to two or more databases in the *Data Source Explorer* view, then select a different connection profile in the *Select Connection Profile* wizard, as shown in *Figure 5.17*.
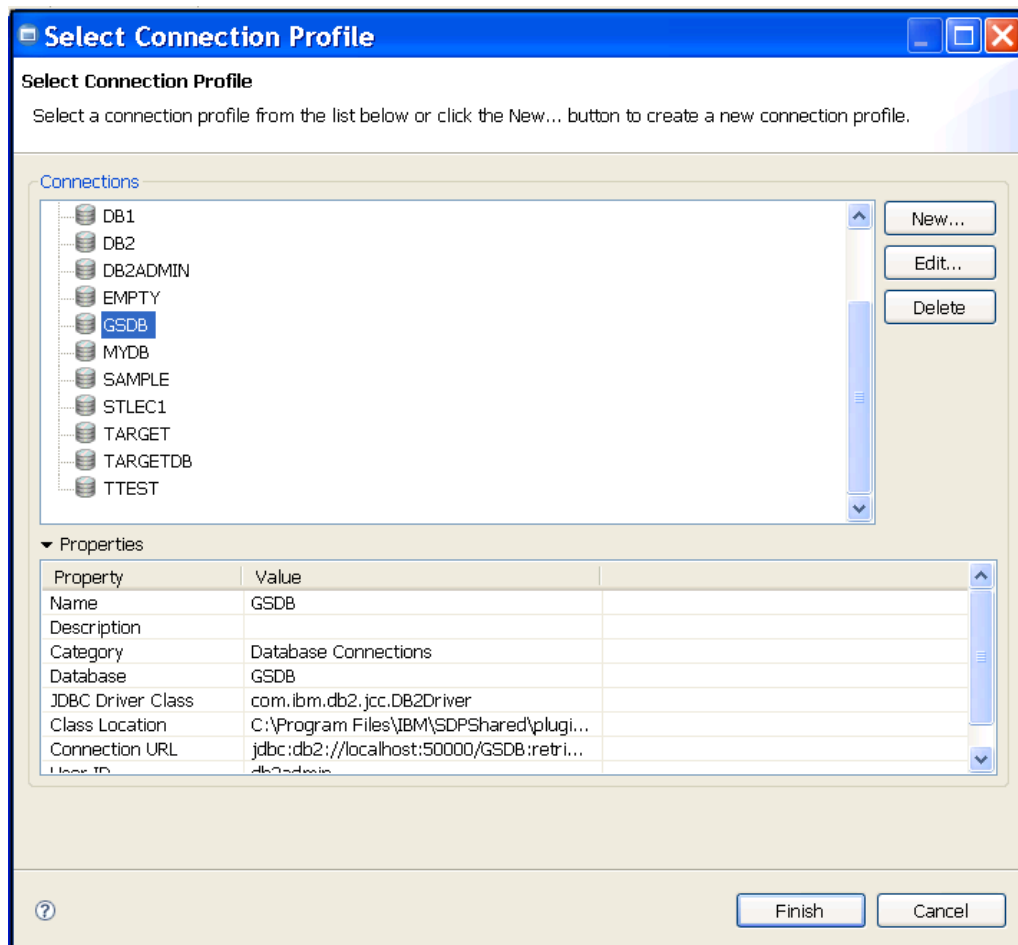
Figure 5.17 – Selecting a database connection

Alternatively, if you are connected to only one database, you can click *New* in the *Select Connection Profile* wizard, and then define a new connection in the *New Connection Profile* wizard.

You also can disconnect the script from a database. This is useful, for example, when you want to work offline. To disconnect the script, select *No Connection* in the drop-down list, as shown in *Figure 5.18*.
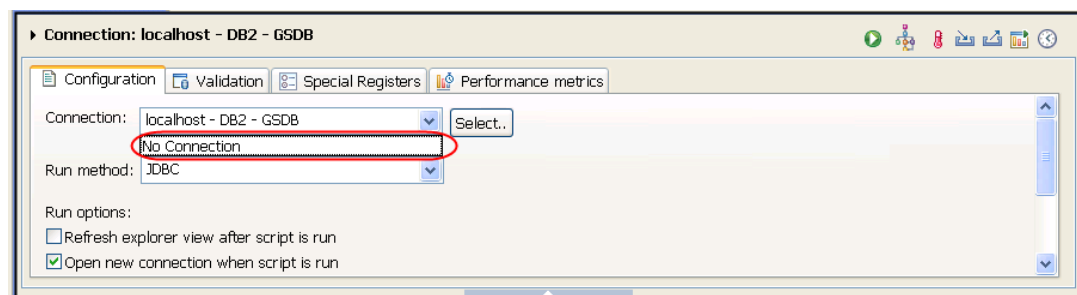
Figure 5.18 – Disconnecting the script from the database to work offline

After you select *No Connection*, the *Command* pane is hidden automatically, as shown in *Figure 5.19*, but you can restore the pane when you want to reconnect to the database. Do this by clicking the *No Connection* link on the editor toolbar. This will bring the *Command* pane to view where you can select the connection profile for the database that you want to connect to. .
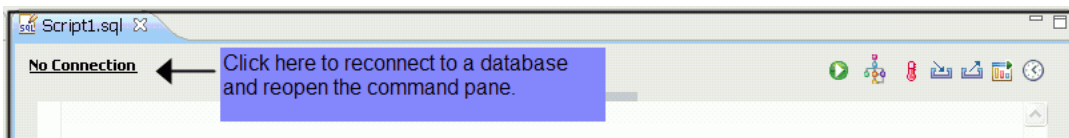
Figure 5.19 – When working offline, command pane is hidden

We will discuss the settings for *Run method* and *Run options* in *Section 5.5*.

5.3 Designing a script

The *Validation* page of the *Command* pane, as shown in *Figure 5.20*, contains controls that you can use while you are creating the statements in your SQL script. You can validate the syntax, the semantics, or both in the statements that you are creating. You also can change the statement terminator for all the statements in the script.
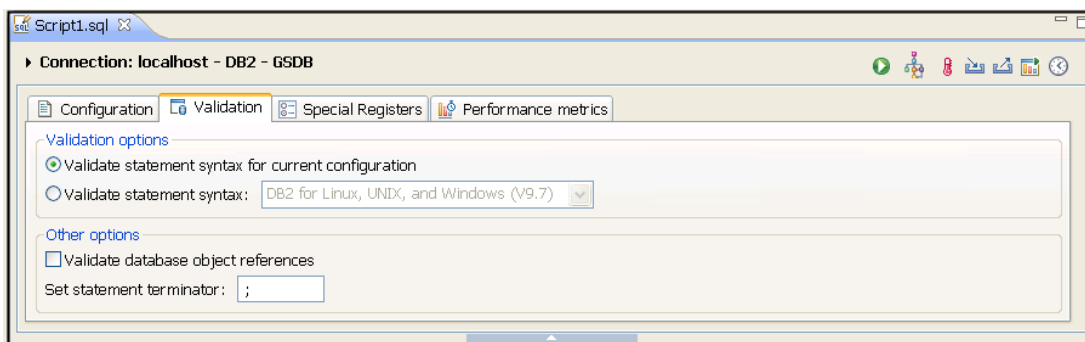
Figure 5.20 – Validation page

5.3.1 Validating the syntax in SQL and XQuery statements

As you enter statements in the editor, the syntax of the statements is validated. They are parsed to determine whether keywords and their location in the statements are valid. The selection in the *Validate options* section determines the parser that is used.

By default, the selected parser is based on the type of database that the script is connected to. As you can see in *Figure 5.20*, the *Validate statement syntax for current configuration* option is selected, which means the selected parser is based on the type of database that the script is connected to on the *Configuration* page.

For example, suppose you develop a script that creates a SALES table with an index in the DB2ADMIN schema of the *GSDB* database. In *Figure 5.21*, the script shows no syntax errors with the DB2 for Linux, UNIX and Windows (V9.7) parser.
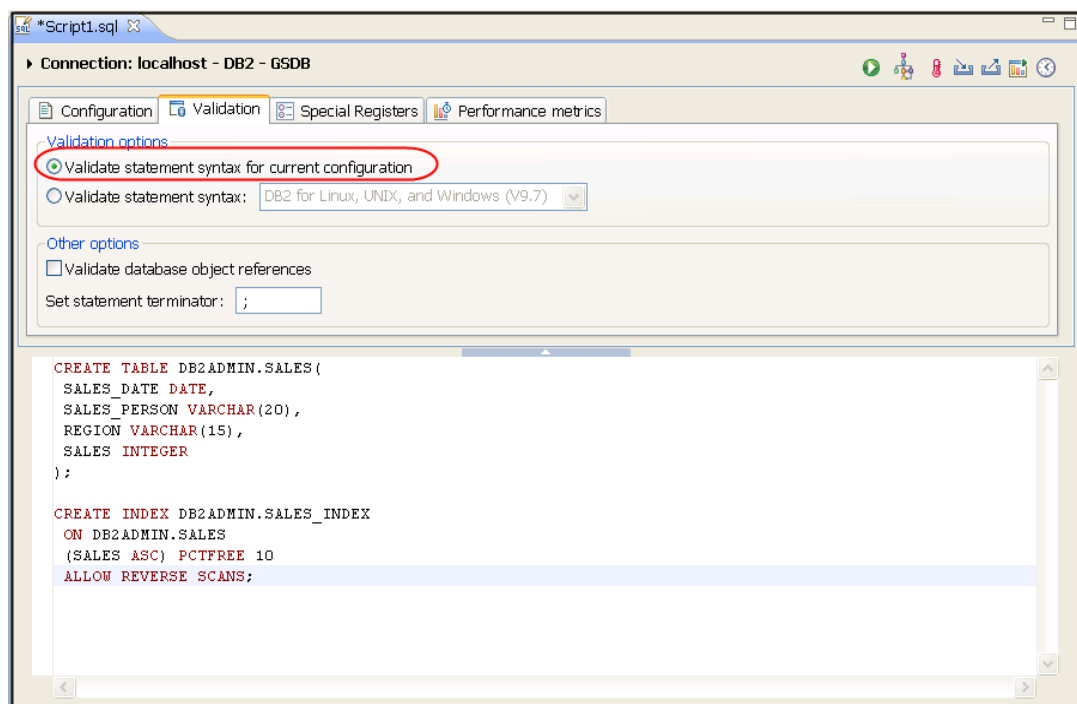
Figure 5.21 – Statements validated with current configuration associated with the script

However, if you want to eventually run this script on a different database server, you can choose to validate it against that server type without changing your current database connection. Simply select the radio button for *Validate statement syntax* and select a different parser from the drop down list. Currently, parsers for the following types of databases are available in the SQL and XQuery editor:

- DB2 for Linux, UNIX, and Windows (V9.7)

- DB2 for Linux, UNIX and Windows (V9.8)

- DB2 for z/OS (V10)

- DB2 for z/OS (V9)

- DB2 for i

- Informix

Note:

Version-specific syntax checking for DB2 for Linux, UNIX and Windows prior to V9.7 will use the V9.7 parser, and any version after V9.8 will use the V9.8 parser.

For example, suppose you want to use the script that creates the *SALES* table with its index in a database on a DB2 for z/OS V10 server. To validate the script for the target database, you can simply change the parser to DB2 for z/OS (V10**)**, which you can do while the script is still connected to the current database.

In this case, the `ALLOW REVERSE SCANS` clause in the `CREATE INDEX` statement is invalid with the DB2 for z/OS V10 parser. The editor flags the validation error with red markers in the left and right margins and underlines the invalid syntax with a red squiggly line. As shown in *Figure 5.22*, you can see an explanation of a syntax error in a pop-up window by moving your mouse pointer over an error marker in the margin.
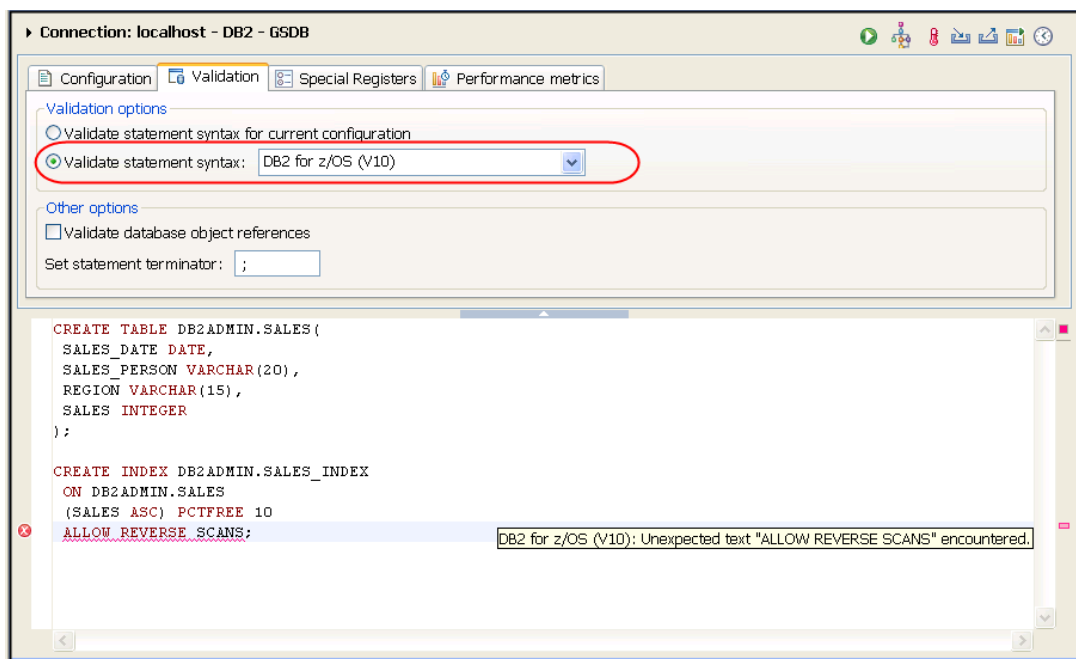
Figure 5.22 – Script statements validated with the DB2 for z/OS (V10) parser

If you prefer, you can stop syntax validation by selecting the *No validation* option from the drop down list of the *Validate statement syntax* option.

If you are working offline (that is, with *No Connection* selected on the *Configuration* page), you can still validate the syntax in the SQL and XQuery statements that you are writing. On the *Validation* page, select the parser for the appropriate database type from the drop down list for *Validate statement syntax* option, as shown in *Figure 5.23*. After you validate for one database type, you can proceed to validate statements with the parser for a different database type.
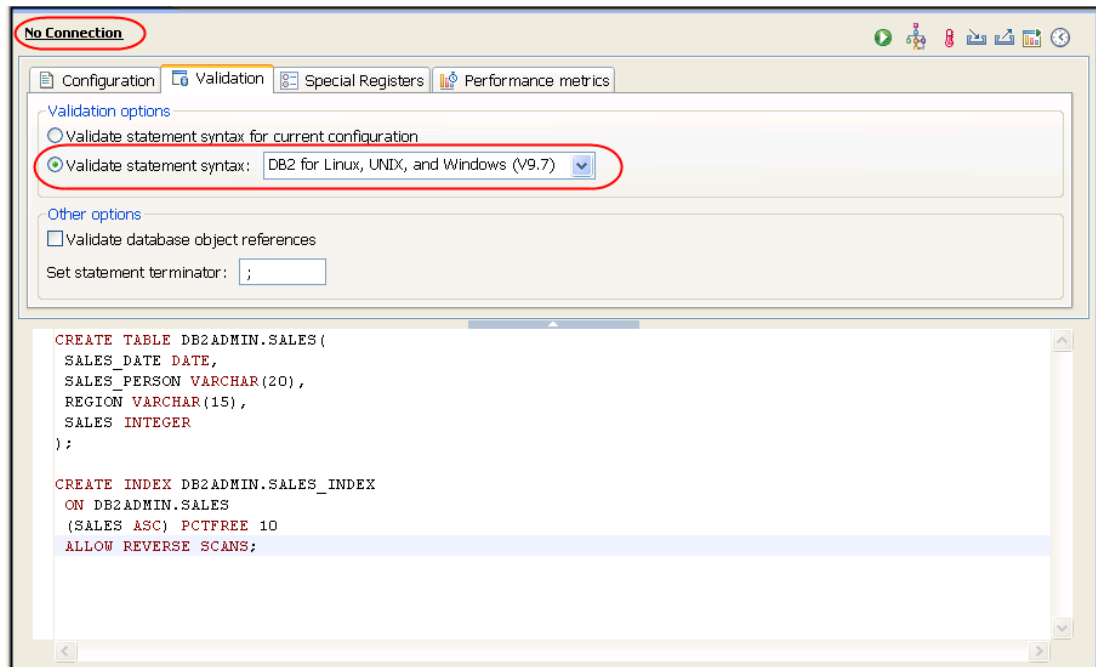
Figure 5.23 – Validating SQL statements offline with no database connection

5.3.2   Validating the semantics in SQL statements

You can also validate the references to tables and stored procedures in the database that the script is connected to. Database object references are validated only in SQL data manipulation language (DML) statements not data definition language (DDL). The state of the *Validate database object references* option determines whether semantic validation occurs as you type.

Semantic validation is associated only with the database that the script is currently connected to. The parser selected in the *Validation options* section has no effect on semantic validation. You can select the option at any time during script development, whether or not you select a parser for syntax validation.

Figure 5.24 shows a semantic error for a reference to the SAMPLE_SALES1 table, which does not exist in the DB2ADMIN schema of the GSDB database. The editor shows the same error indicators for semantic and syntax errors.
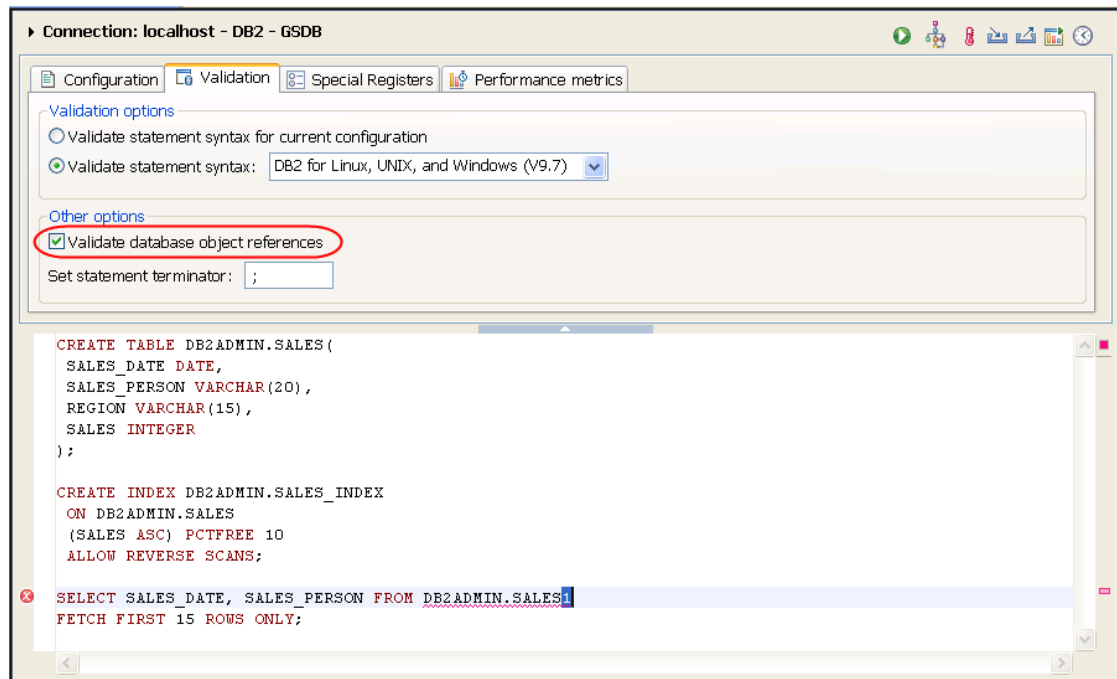
Figure 5.24 – Validate database object reference error in a DML statement

5.3.3   Changing the statement terminator

When you have multiple statements in an SQL script, each statement must be separated from the one that follows by a statement terminator. By default, the SQL and XQuery editor uses a semicolon (;) as the statement terminator. You can change the default statement terminator for all scripts that you create in the editor by specifying the new default statement terminator in (*Window -> Preferences*).

You can use the field on the *Validation* page to set the statement terminator for a specific script. The statement terminator that you set in an SQL script persists every time that you open the script in the SQL and XQuery editor.

In a given script, you can use only one statement terminator. That is, all the statements in an SQL script must use the same statement terminator. When you set the statement terminator in an SQL script that contains existing statements, the editor does not update the existing statement terminators automatically. Instead, you must manually update all existing statement terminators in the script.

Figure 5.25 shows an example of the syntax validation error that occurs after you set the statement terminator to an exclamation point (!), and do not update an existing statement terminator. You will get an unexpected token error if you run the script after you stopped syntax validation.
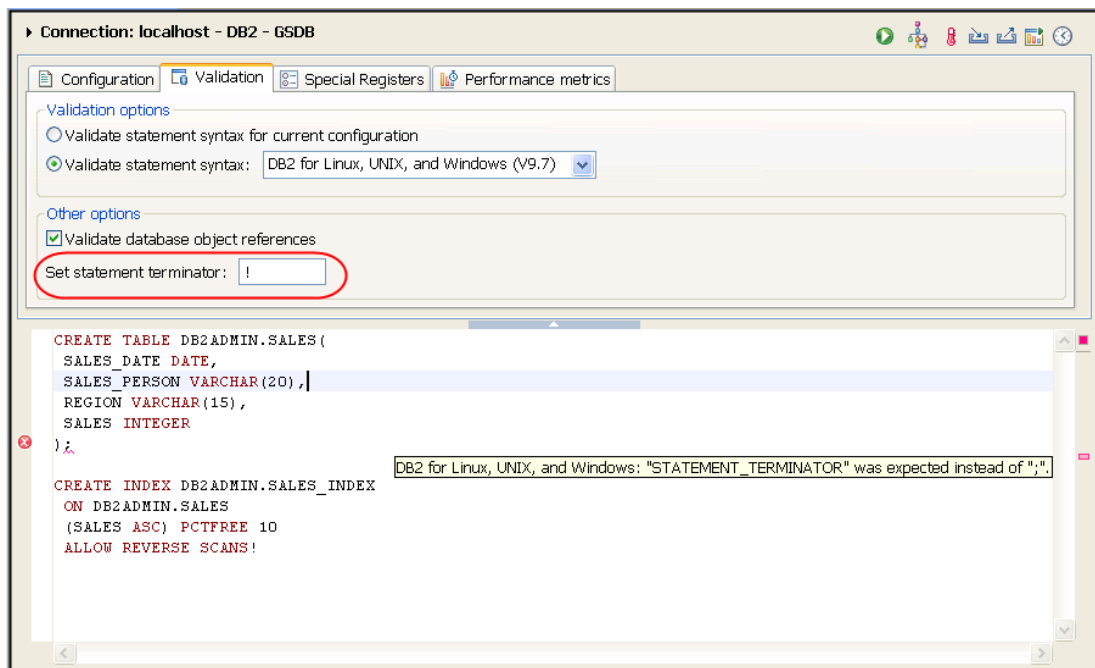
Figure 5.25 – Invalid statement terminator after change from default

5.3.4   Content assist in the SQL and XQuery editor

Like many other Data Studio features, the SQL and XQuery editor provides content assist to create SQL statements. Similar to the Java editor in Eclipse, content assist can be triggered by pressing the key combination *Ctrl+Space*.

To create your SQL statement with content assist, type the expression `select * from` and then press *Ctrl+Space*. This sequence of steps will display the content assist for selecting database tables. When referencing fully qualified table names, you can take advantage of content assist for multiple steps, as shown in *Figure 5.26*. *Label 1* shows content assist for the selecting a schema name, *Label 2* for the table name, and *Label 3* for the column name.
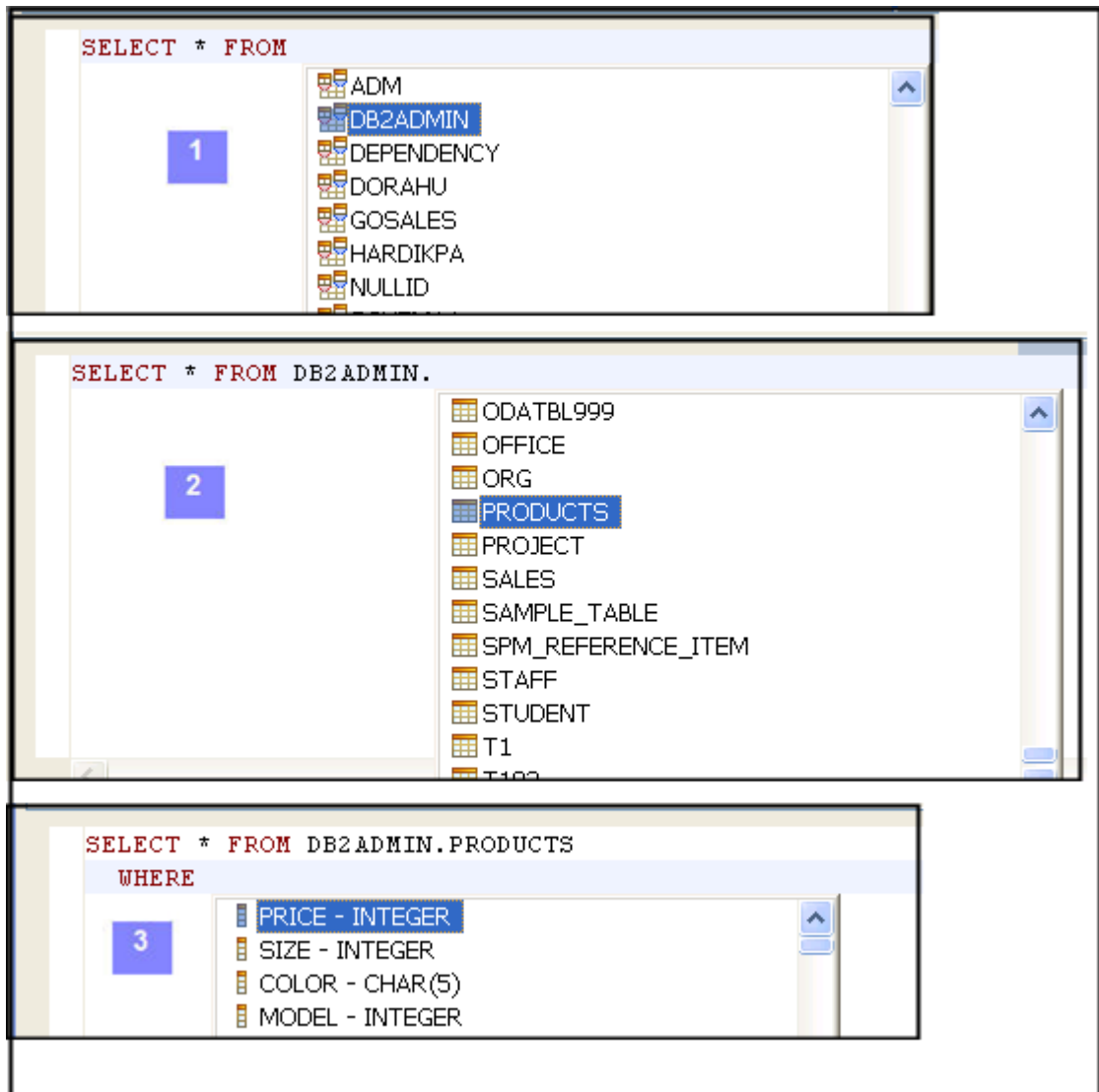
Figure 5.26 – Content assist in the SQL and XQuery editor

After you add the required table to the *FROM* clause of the SQL statement, the content assist can also help you find additional columns from that table. You can use this capability to help you complete the SQL statement. *Figure 5.27* shows the column `COLOR` being added to the SELECT clause of the SQL statement.
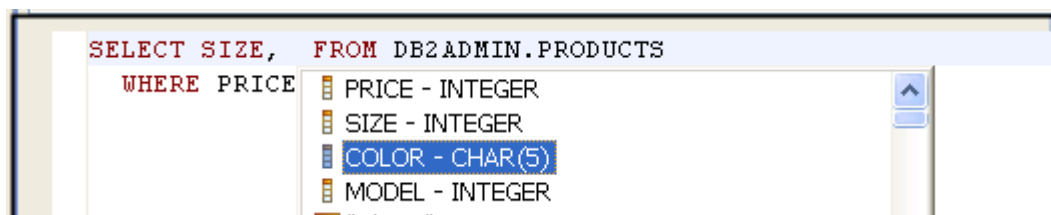
Figure 5.27 – Content assist to reference table columns

5.4   Special registers

With the *Special Registers* page, you can specify the current schema and path that you want to use to run the SQL or XQuery statements against.

The *Current schema* register is used when you deploy and run database objects with your SQL scripts. It resolves unqualified database object references. By default, the database name from the current connection profile on the *Configuration* page will be displayed if your SQL does not specify a schema. To change it, click on the *Select* button, then select a different schema from the *Select a schema* window, as shown in *Figure 5.28*.
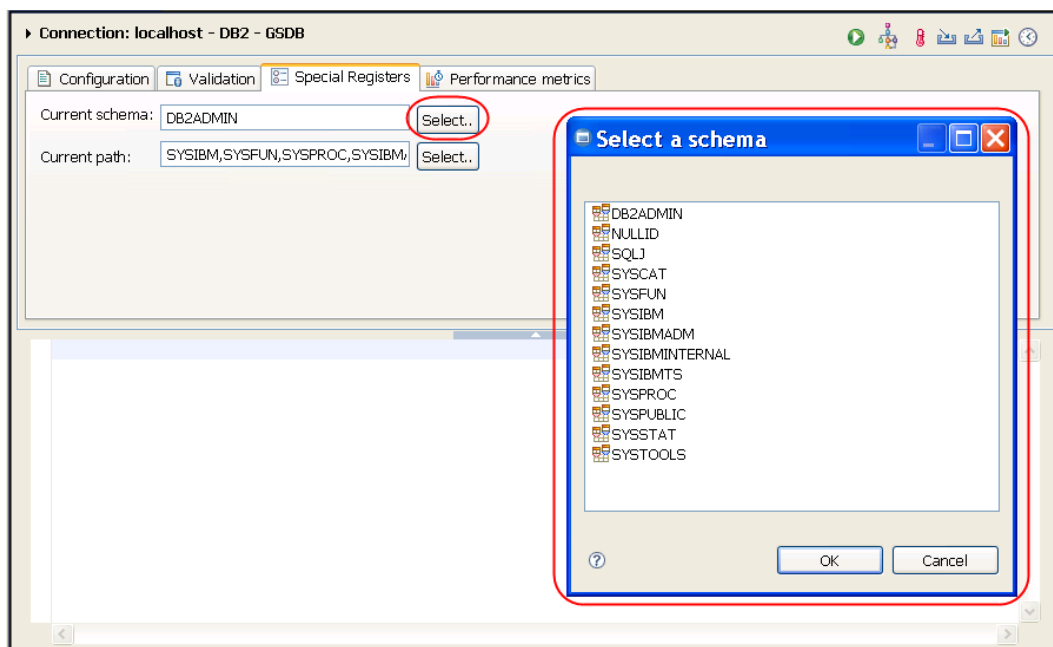
Figure 5.28 – Changing the current schema to different one

The *Current path* register is used when you deploy and run database objects with your SQL scripts. It resolves unqualified function names, procedure names, data type names, global variable names, and module object names in dynamically prepared SQL statements. You can add schemas to the *Current path* by clicking the *Select* button, as shown in *Figure 5.29*. Select one or more from the *Select schemas* window that opens.
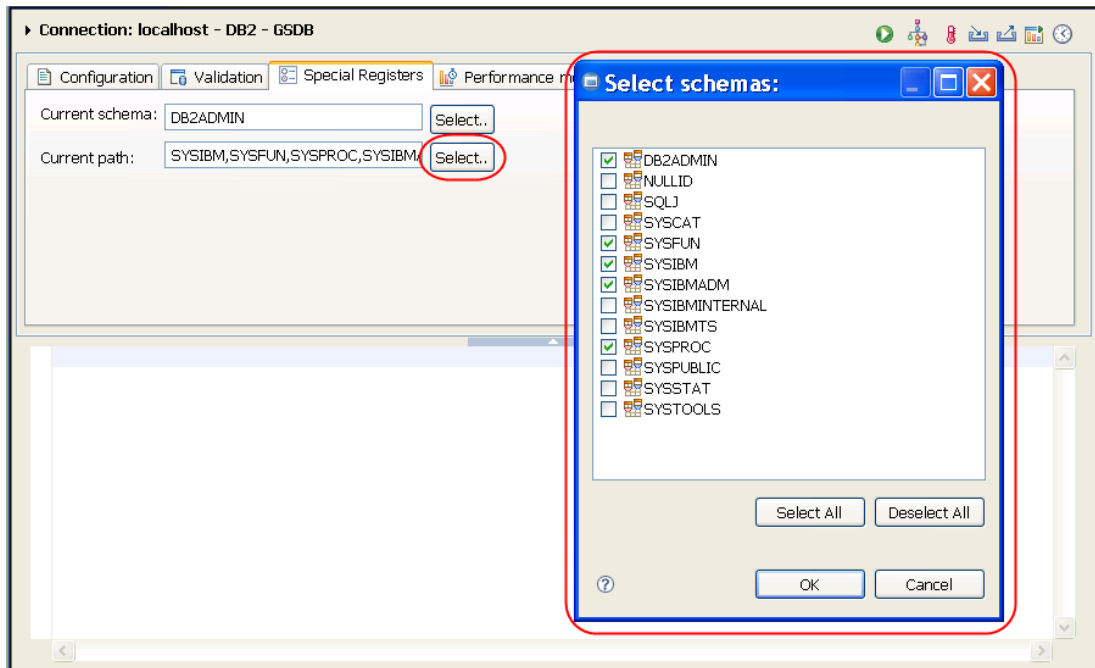
Figure 5.29 – The Select schemas window opens to change the Current path

5.5 Running the script

To determine if an SQL script you are developing returns the expected results, you need to run it. But first, you may want to modify how the SQL script will run. You can do this with the run preferences on the *Configuration* page, as shown in *Figure 5.30*.
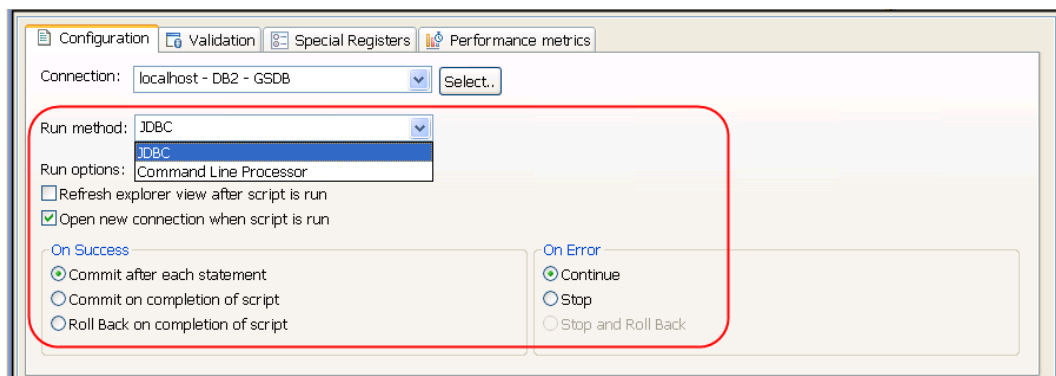
Figure 5.30 – Run method and Run options.

Run method

You can set the execution environment for the SQL and XQuery Editor with this preference. The available execution environments are JDBC and Command Line Processor (CLP).

Refresh explorer view after script is run

You can select this option to refresh the Data Source Explorer or Administrator Explorer view after you run the script.

Open new connection when script is run

You can select this option to create a new connection to the target database that is used to run your scripts.

By default, this option is selected. The *Run method* must be set to a JDBC environment if you wish to deselect it. If you do, the information from the *Connection* field is used and the *Commit on completion of script* and *Roll Back on completion of script* options are disabled.

On Success

These options specify how statements are handled when they are run successfully. The availability of each option depends on the run method you select. More information is available in the next section.

On Error

These options specify how statements are handled when an error occurs. The availability of each option depends on the run method you select. More information is available in the next section.

5.5.1 JDBC result preferences

Table 5.1 describes the behavior when statements are run successfully or encounter errors in a JDBC environment:

On Success	On Error	Result
Commit after each statement	Continue	If a statement is successful, it is committed to the specified database. If an error occurs, the next statement will run.
Commit after each statement	Stop	If a statement is successful, it is committed to the specified database. If an error occurs, the script will stop running.
Commit on completion of script	Continue	If all of the statements in the script are successful, all statements are committed to the specified database. If an error occurs, the next statement will run, and all successful statements are committed to the specified database.
Commit on completion of script	Stop and Commit	If all of the statements in the script are successful, all statements are committed

On Success	On Error	Result
		to the specified database. If an error occurs, the script will stop running, and any statements that were successfully run are committed to the specified database.
Commit on completion of script	Stop and Roll Back	If all of the statements in the script are successful, all statements are committed to the specified database. If an error occurs, the script will stop running, and all successful statements are rolled back.
Roll Back on completion of script	Continue	If all of the statements in the script are successful, all statements will be rolled back. If an error occurs, the next statement in the script will run, and any successful statements are rolled back.
Roll Back on completion of script	Stop and Roll Back	If all of the statements in the script are successful, all statements will be rolled back. If an error occurs, the script will stop running, and all successful statements are rolled back.

Table 5.1 – Choosing success and error behavior in JDBC

5.5.2 CLP result preferences

Table 5.2 describes the behavior when statements are run successfully or encounter errors in the CLP.

On Success	On Error	Result
Commit after each statement	Continue	If a statement is successful, it is committed to the specified database. If an error occurs, the next statement will run.
Commit after each statement	Stop	If a statement is successful, it is committed to the specified database. If an error occurs, the script will stop running.

On Success	On Error	Result
User managed commit	Continue	If a COMMIT statement is included in the script, the statement is committed at that point. If an error occurs, the next statement will run.
User managed commit	Stop and Commit	If a COMMIT statement is included in the script, the statement is committed at that point. If an error occurs, the script will stop running, and any statements that have run are committed to the specified database.

Table 5.2 – Choosing success and error behavior in the CLP environment

5.5.3 SQL Results view

Now that you have explored the Run preferences, you are ready to run your SQL script.

Click the Run SQL button (   ) on the editor toolbar. The script is run against the database that the script is currently connected to. The progress and results of running the script are then displayed in the *SQL Results* view.

Figure 5.31 shows the execution result of SQL statements with JDBC run method and with JDBC preferences in the *SQL Results* view.
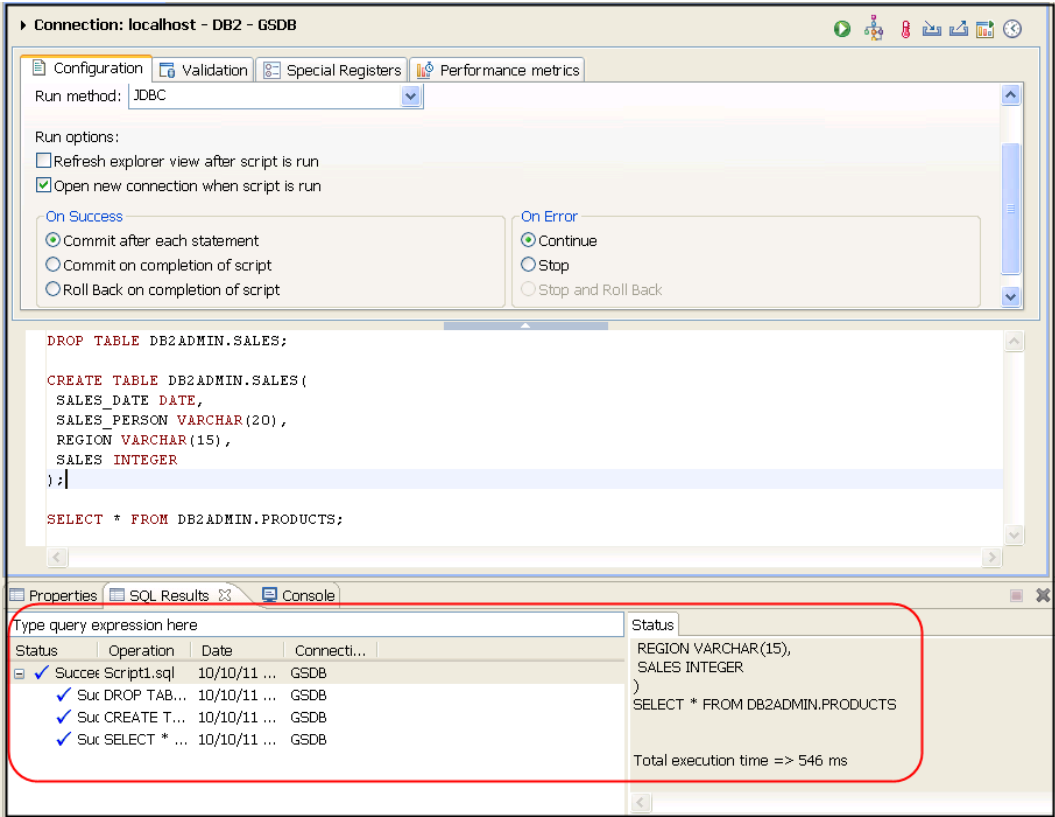
Figure 5.31 – JDBC run method and with JDBC preferences

Figure 5.32 shows the execution result of SQL statements with Command Line Processor (CLP) run method and with CLP preferences in the *SQL Results* view.
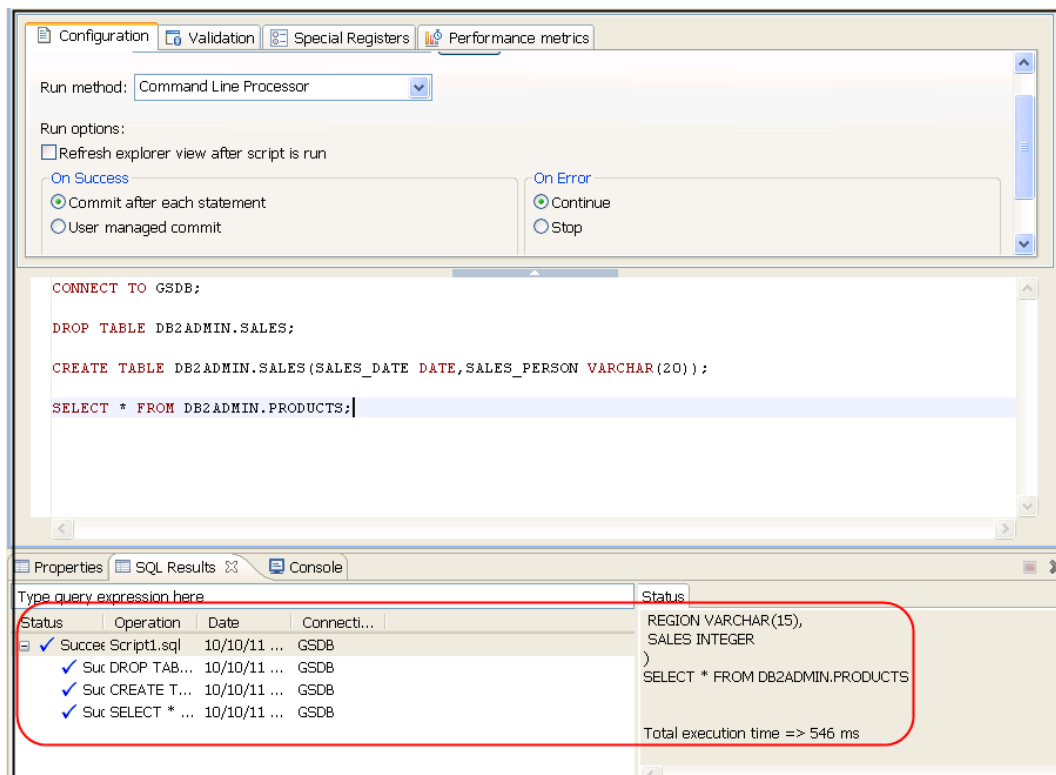
Figure 5.32 – Command Line Processor (CLP) run method and with CLP preferences

5.6 Creating SQL statements with the SQL Builder

In this section, you will learn how to develop a query using the SQL Query Builder instead of the SQL and XQuery editor.

From your Data Development project in the **Data Project Explorer**, right click and select *New ->SQL or XQuery Script* and choose the radio button for *SQL Query Builder*, as shown in *Figure 5.35.*
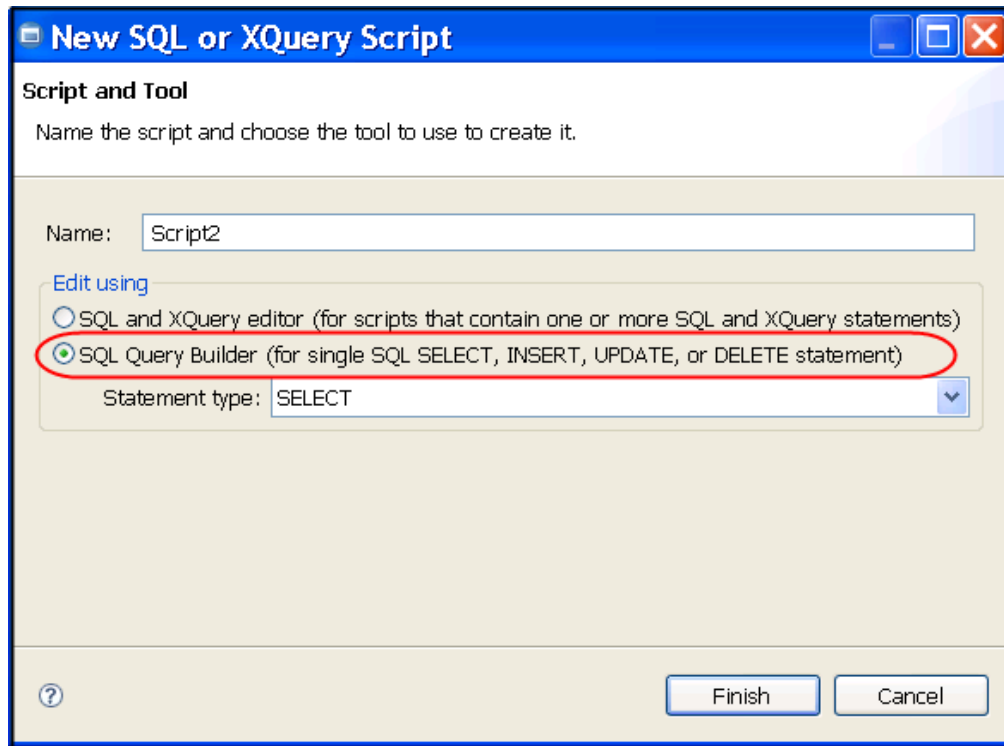
Figure 5.35 – Choose to build an SQL script using Query Builder

When using the SQL Query Builder, you can select from several SQL statement types to be created: **SELECT**, **INSERT**, **UPDATE**, **DELETE**, **FULLSELECT**, and **WITH**, as shown in *Figure 5.36.*
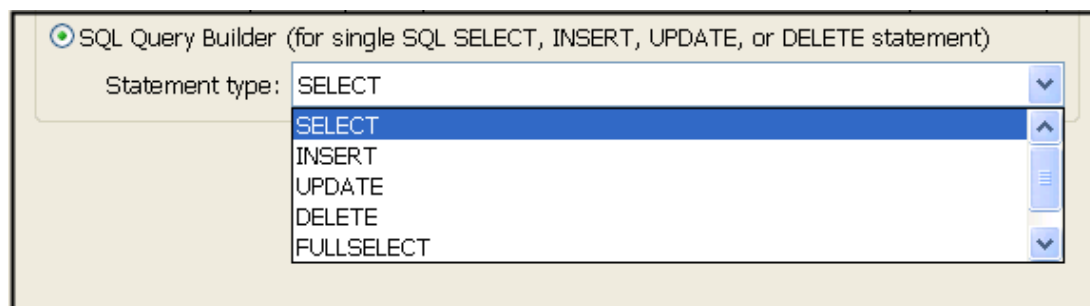
Figure 5.36 – Selecting a statement type

After you have selected the statement type, click *Finish*. In this example, choose the **SELECT** statement type.

After you click *Finish*, you will notice the new file `Script2.sql` in the *SQL Scripts* folder of your project. The SQL Builder will also automatically open so that you can construct your statements, as shown in *Figure 5.37*.
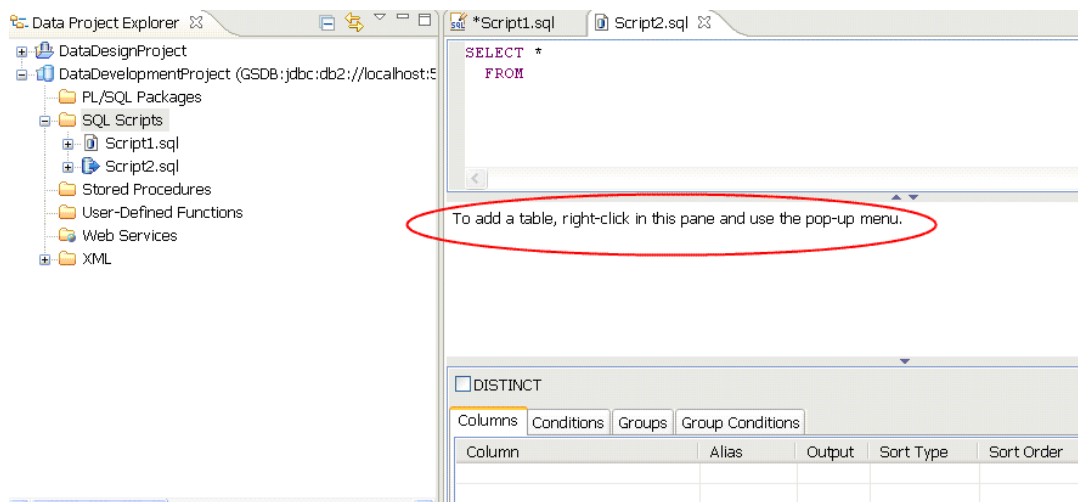
Figure 5.37 – Script2.sql opened in the SQL Builder

The SQL Builder provides an easy-to-use interface to create SQL statements. You can specify which tables will be included in the statement and, from those tables, select the columns to be returned or used for filtering.

Start by following the instructions in the editor to add a table:

1. Right click in the middle pane and use the pop-up menu option *Add table…* to select a table from the database. Choose **PRODUCTS**, which then adds this table automatically to your script.

2. Then select the table columns you want to include in your SQL **SELECT** statement. You can choose them by selecting them directly from the pane that appears when you selected the table. Select the columns **PRICE** and **COLOR,** as shown in *Figure 5.38,* below.

3. In the *Conditions* tab, add the value filter by selecting the column **SIZE**, the operator =, and typing in the value '5' for the value, as shown in *Figure 5.38.* When you move your mouse from this input table, this **WHERE** clause is added to the script.
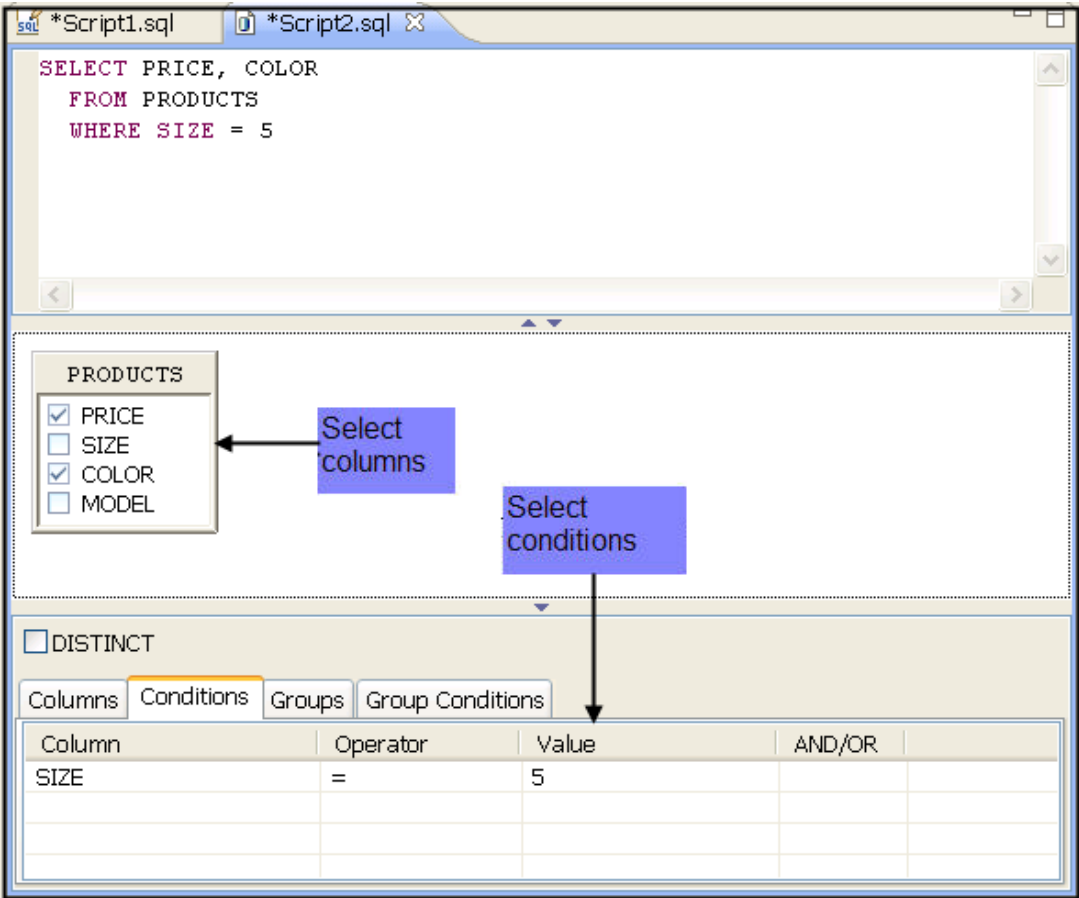
Figure 5.38 – Using the SQL Builder to create a SQL statement

 The SQL Builder is useful when you need to create queries that use joins, full selects, and subselects because it lets you add several tables, select multiple columns from different tables, and specify conditions, grouping and sort order.

Here are a few examples that show how SQL Builder can help you create more complex queries:

Example 1: *Figure 5.39* shows a join query created by using **ROUTINEDEP** and **ROUTINES** tables from the **SYSCAT** schema. You can see how the interface lets you create the join query by specifying the columns, conditions, groups and group conditions.

Figure 5.39 – Using the SQL Builder to create a JOIN query statement

Example 2: *Figure 5.40* shows a full select statement (`UNION ALL`) of tables *SYSCAT.ROUTINEDEP* and *SYSCAT.ROUTINES* that also includes an `ORDER BY` clause.
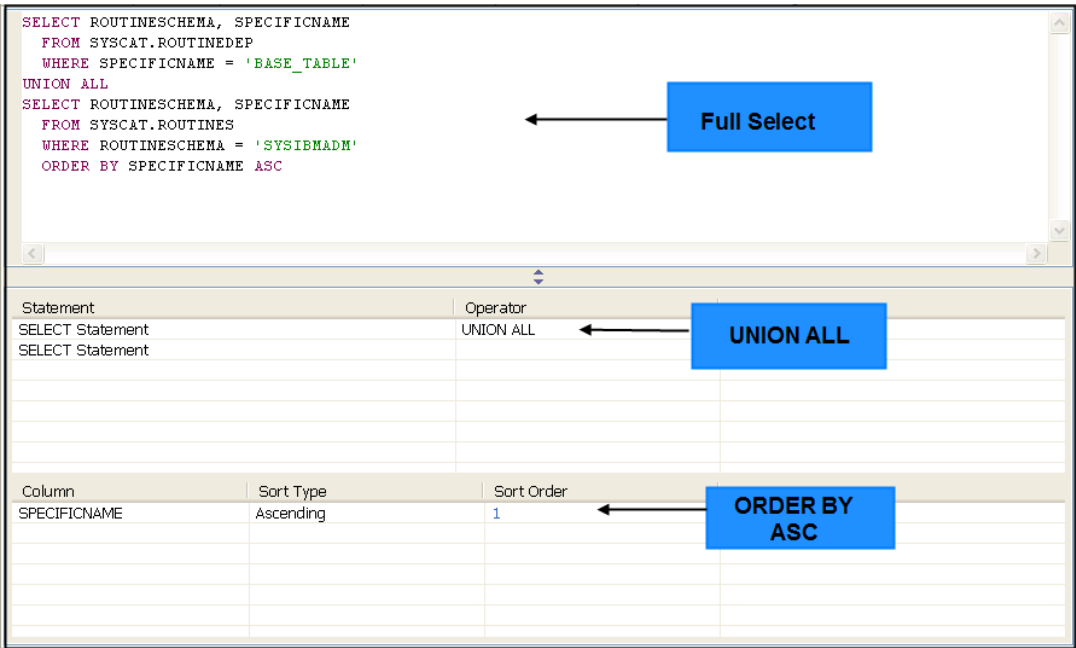
Figure 5.40 – Using the SQL Builder to create a full select statement

Example 3: *Figure 5.41* shows an `INSERT` statement created with a subselect.
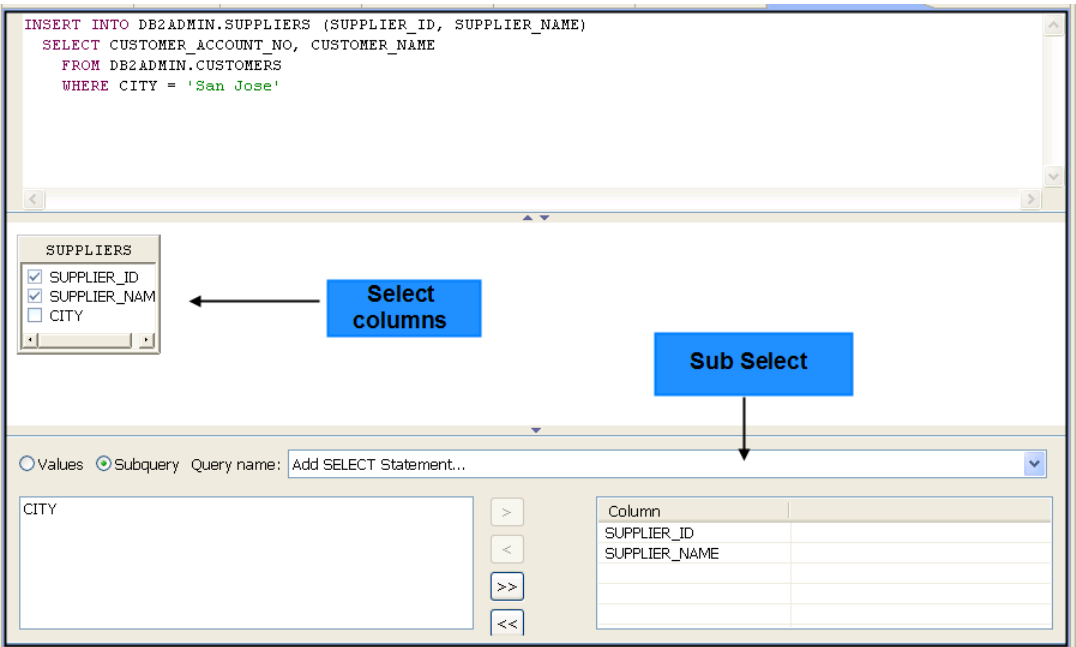
Figure 5.41 – Using the SQL Builder to create an INSERT with subselect statement

5.7 Summary

In this chapter we described some basic tasks that can help you develop SQL scripts more efficiently, with features such as syntax highlighting, SQL formatting, content assist, semantic validation, statement parsing, and validation. Being able to execute SQL scripts with multiple database vendors and navigate the SQL results from a single view can help simplify creating SQL script development.

5.8 Review questions

1. Which are the two available types of data projects for creating SQL scripts in the Data Project Explorer?

2. List and describe the DB2 application process settings that you can configure when creating a Data Development project.

3. The Data Project Explorer is part of which Data Studio perspective?

   A. Database Development perspective

   B. Data perspective

   C. Database Debug perspective

   D. Resource

   E. None of the above

4. Which tab in the Command pane of SQL and XQuery Editor helps you to change the database connection?

   A. Configuration

   B. Validation

   C. Special Registers

   D. None of the above

5. What are the available database vendors and versions for syntax validation under the "Validate statement syntax" option?

6. Is content assist available in the SQL and XQuery Editor?

7. What special registers are available in the SQL and XQuery Editor?

   a. Current schema

   b. Current path

   c. Current schema and Current path

   d. None of the above

8. What are the available run methods?

9. Is the rollback preference option is available for the CLP (Command Line Processor) run method?

10. In which Data Studio view can you see the results of executing your SQL statements?

   a. Data Source Explorer

   b. Project Explorer

   c. SQL Outline

   d. SQL Results

   e. None of the above

11. What other capability is supported by the SQL and XQuery editor?

   a. Visual Explain

   b. InfoSphere Optim Query Tuner

   c. Performance Metrics

   d. Job Manager

   e. All of the above

12. Which tool or tools can be used to create SQL scripts?

   a. SQL Editor

   b. SQL and XQuery Editor, SQL Query Builder

   c. Database Object Editor, SQL Query Builder

   d. Routine Editor, SQL and XQuery Editor

   e. None of the above

13. Which of the following set of commands represent all the commands that are available for creating SQL statements with the SQL Query Builder?

   a. SELECT, INSERT, UPDATE, DELETE

   b. SELECT, INSERT, UPDATE, JOIN, FULLSELECT, WITH

   c. SELECT, INSERT, UPDATE, DELETE, FULLSELECT, WITH

   d. SELECT, INSERT, UPDATE, DELETE, JOIN, FULLSELECT,   WITH

   e. None of the above.

14. What are the major features that differentiate the components of the SQL Query Builder and the SQL and XQuery editor?

6

Chapter 6 – Managing jobs

Job management is a feature of the Data Studio web console component that is intended to replace the DB2 Control Center Task Center and other scheduling tools that you might have used with DB2 databases in the past. The job manager provides you with the tools needed to create and schedule script-based jobs on your DB2 for Linux, UNIX, and Windows and DB2 for z/OS databases from a web console interface.

In this chapter you will:

- Learn about the job scheduling capabilities of the Data Studio web console
- Create a SQL script-based job
- Run the job manually
- Schedule the job to run on the sample database
- Set up monitoring of the job and check the job history

6.1 Job management: The big picture

As mentioned in *Chapter 1*, Data Studio web console contains both health monitoring and job management features. This chapter gives an overview of the basic capabilities of the job manager feature of the web console.

For the purpose of this chapter, you will be using the Data Studio web console default administrative user (default user ID: admin) for all job creation and scheduling. You created this user when you installed the Data Studio web console. The default administrative user has administrative access to the web console.

The default administrative user privileges only apply to the web console interface. The default administrative user does not have any privileges on the databases that you want to schedule jobs on. When you schedule or run a job on a database, you must provide a user ID that exists on the database on which you want to run the job. That user ID must have permissions to execute the commands contained in the script part of the job on the database.

Note:
Optionally, you can set up the Data Studio web console to allow users other than the default administrative user to log in to the web console. To do this, you must add a

repository database and configure the Data Studio web console to allow users of that database to log in to the web console. For more information on how to use the Data Studio web console with multiple users, see *Appendix B.*

6.2 The Data Studio web console

To manage jobs with Data Studio you must install the Data Studio web console. If you have not installed the web console, follow the instructions in *Chapter 1*. After logging in to the web console, you can open the job manager by selecting *Open > Job Manager*, as seen in *Figure 6.1*.
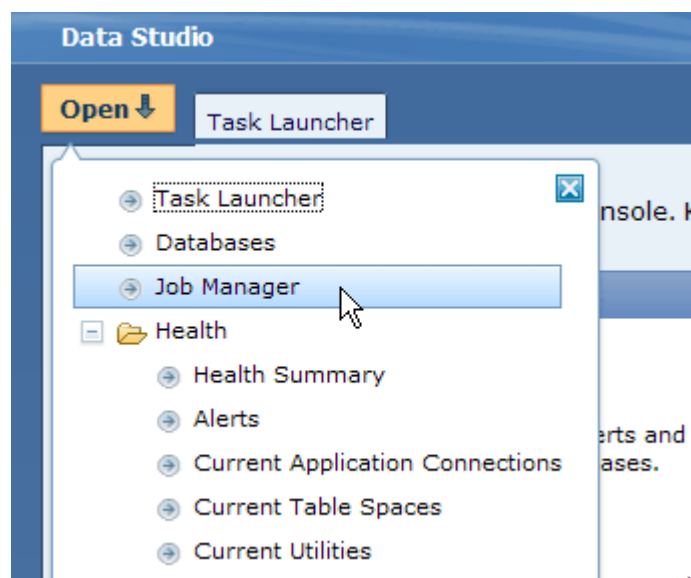
Figure 6.1 – Opening the job manager from the web console.

Note:
You can open the job manager embedded in the Data Studio client to extend the function of the client with health monitoring and job management. For information about how to embed the Data Studio web console in the Data Studio client, see *Appendix B.1 Integrating Data Studio web console with Data Studio full client*.

6.3 Jobs and job components

A job is a container that holds the components that are required to run a script on one or more databases. When you create a job, you can either schedule it to run on one or more databases, or you can leave the schedule component of the job empty. A job can also be run manually from the Job List tab of the job manager. You can also configure multiple jobs to run sequentially by adding a chain of jobs to the job that you are creating.

The job does not contain any information about which databases that the job will run on. The database information is included in one or more schedules that you can create for each job.

6.3.1 The components of a job

A job consists of several components, all of which can be defined when you create a job. The only job components that are required to create a valid job are the job name, the job ID, and the script. These are described in *Table 6.1*.

Job component	Description
Job ID and name (required)	The job ID is the unique identifier of the job and is generated automatically by job manager. The job name that you specify can be used as a descriptive way to easily identify the job in the job list or when scheduling jobs.
Script (required)	The script is the executable part of a job and defines the actions that are done on the database when the job is run. You can enter one script per job. **Important:** The job manager does not provide a script editor and does not verify that the scripts that you enter are valid. Run the script on a database or use other methods to verify that the script is correct and that it produces the expected results before you schedule the job in the job manager.
Schedule	A schedule defines when a job will be run, whether the job is repeating, and whether the schedule is limited in number of runs or in time. The schedule also defines one or more databases on which to run the job. A job can have any number of schedules attached to it, but each schedule only applies to one job. A schedule must be active to run the job. Each schedule has a unique ID that is based on the job ID with an integer appended at the end of the ID. **Example:** The first schedule that is attached to job ID 1234 will have schedule ID 1234-1. The second schedule that is attached to the same job will have schedule ID 1234-2, and so on. A schedule is not required; you can also run jobs manually from the job manager. When you schedule a job on a single database, you can define the user ID that will run the job. If you schedule a job to run on more than one database, the job is run on each database by the user ID that is stored in the database connection for that database.
Notification	Notifications are one of two methods that you can use to determine whether your jobs ran successfully. Use notifications to alert you by email on the successful or failed run of a job on your databases. The other method of determining the outcome of your jobs is to look in the job history tab of the job manager, where you can see the status of all jobs on all databases that are configured for your system.
Chains	You can add a chain of subsequent jobs that run depending on the outcome of the primary job. A job chain can consist of three consecutive jobs: the

	primary job, a job that runs if the primary job is successful or a job that runs if it is unsuccessful, and finally an ending job that runs at the end of the chain regardless of the outcome of the preceding jobs. **Important:** When a job is run as part of a chain, any schedules and chains that are associated with that job are ignored.

Table 6.1 – Components of a job

6.3.2 Job types

The job manager supports three default job types. The job type indicates the way that the job manager connects to the databases to run the scripts.

Job type	Connection method
SQL-only script	The job manager connects to the database and runs the SQL commands that are included in the job script directly on the database.
DB2 CLP script	The job manager logs in to the database server by using SSH as the user ID defined in the database connection, and then it runs command line processor (CLP) commands directly on the DB2 console of the server.
Executable/Shell script	The job manager logs in to the database server by using SSH as the user ID that is defined in the database connection, and then runs shell commands directly on the server.

Note:

To run DB2 CLP script jobs or Executable/Shell script jobs on a database the user ID that is used to run the job must have permission to log in to the database server by using SSH.

6.4 Create and schedule a job

Use the Job List tab in the web console to create, schedule, and manage jobs for your databases or to run existing jobs directly against a database without scheduling. The first time you open the job manager, no jobs have been created, and the job manager informs you what needs to be done to create one.

1 Chapter 6 – Managing jobs 203
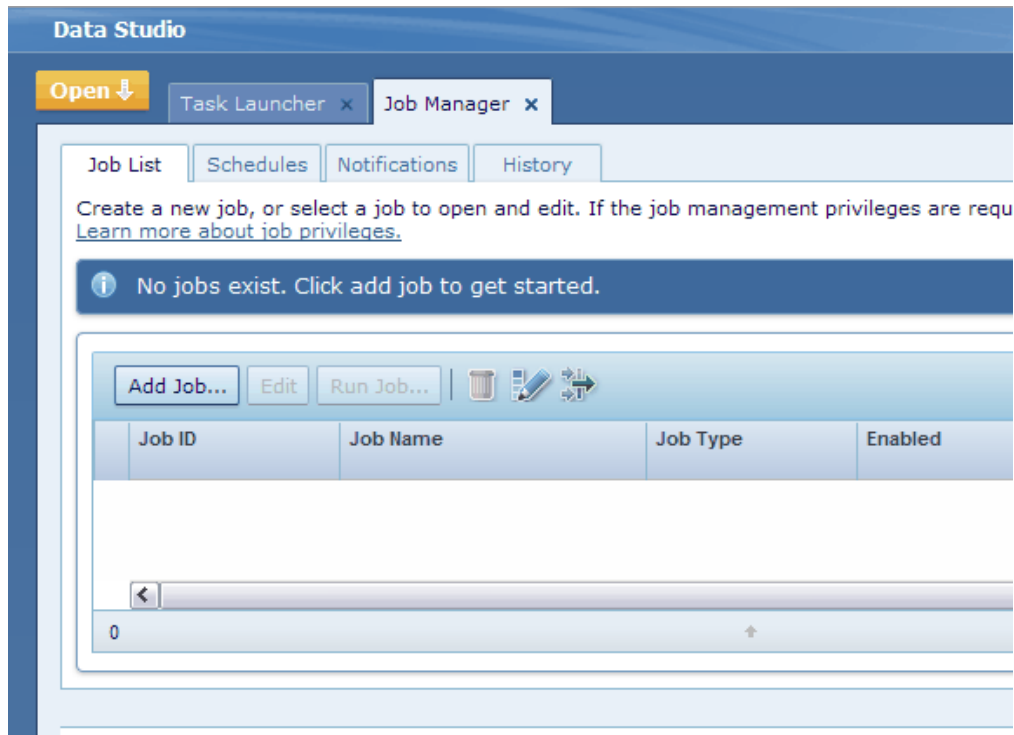
Figure 6.2 – The job list page with no jobs created

When you create a job or open an existing job, the job details open in the job editor. If you have more than one job open for editing, each job opens in its own tab. Within each tab you can use the *Job Components* menu to drill down into the components of each job.

If you have configured the Data Studio full client for Data Studio web console, you can also schedule a script to run as a job directly from the SQL script editor. See *6.7 Scheduling jobs from the Data Studio client* for more information.

6.4.1 Creating jobs

To create a job using job manager:

1. From the Job List, click *Add Job* to open the Add Job wizard where you can enter the basic properties of the new job.
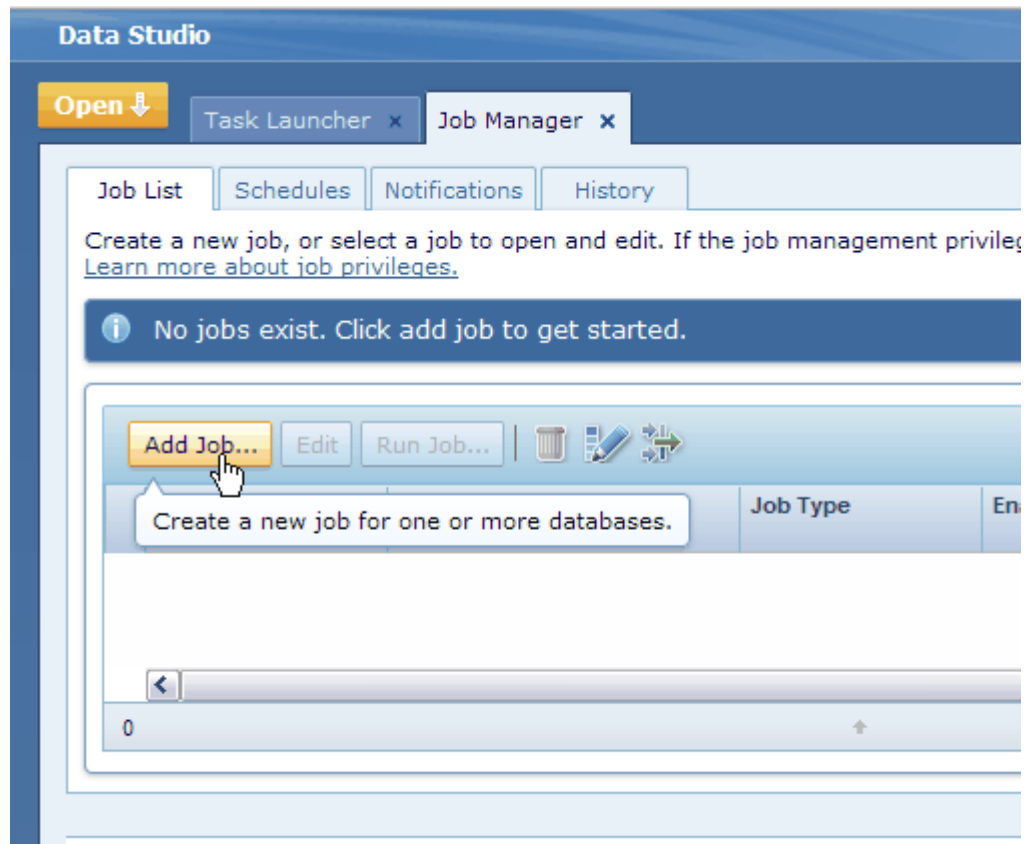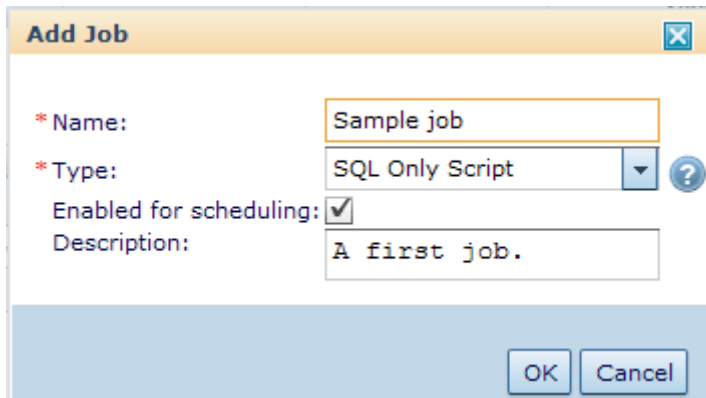
Figure 6.3 - Click Add Job to add a new job

The basic job properties are:

- *Name* – A descriptive name for the job.

- *Type* – The type of job you want to create. The job type decides how job manager connects to the database to run the script.

- *Enabled for scheduling* – Select this box to enable the job for scheduling. If the box is not selected you cannot schedule the job, but you can still run it manually from the job list.

- *Description* – A short description of the job.

Figure 6.4 – Enter the basic properties for a job

Once you have entered the basic properties, an entry is created for the job in the Job List and you can configure the remaining job components.

2. Click a component in the *Job Components* menu to configure the component.
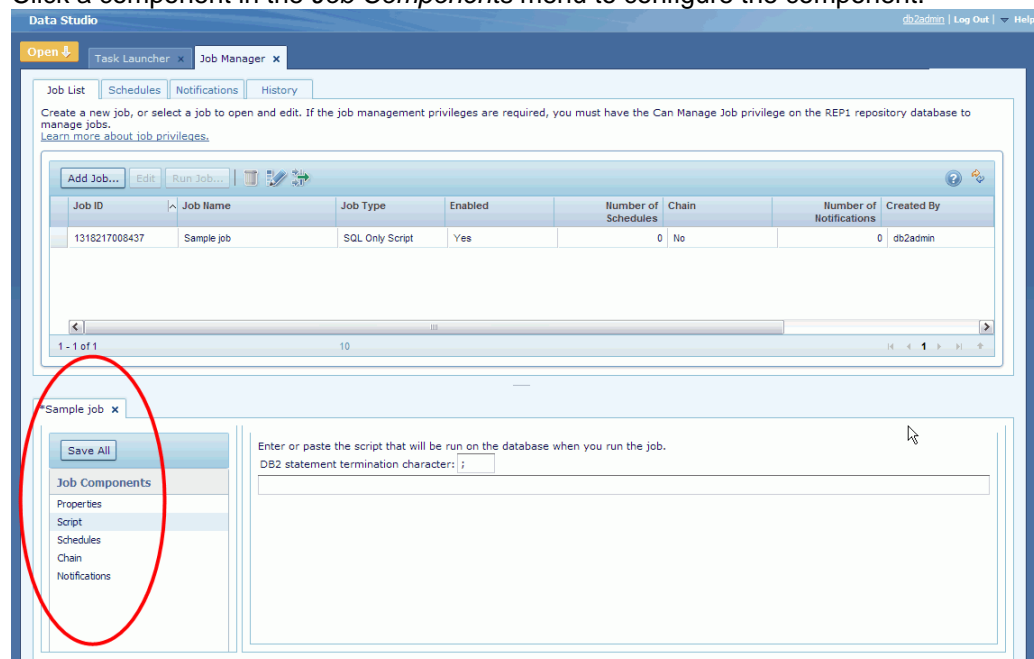
Figure 6.5 – The job editor and the job components menu

3. Click *Script* to add a script to the job. The script is the executable part of a job and defines the actions that are done on the database when the job is run. A job must contain a script. Important: The job manager does not provide a script editor and does not verify that the scripts that you enter are valid. Run the script on a database or use other methods to verify that the script is correct and that it produces the expected results before you schedule the job in the job manager.

For this example, we will use the following sample test script which will create a new table and then immediately drop the same table, leaving the database untouched:

```
create table employee(c1 int, c2 int);
drop table employee;
```
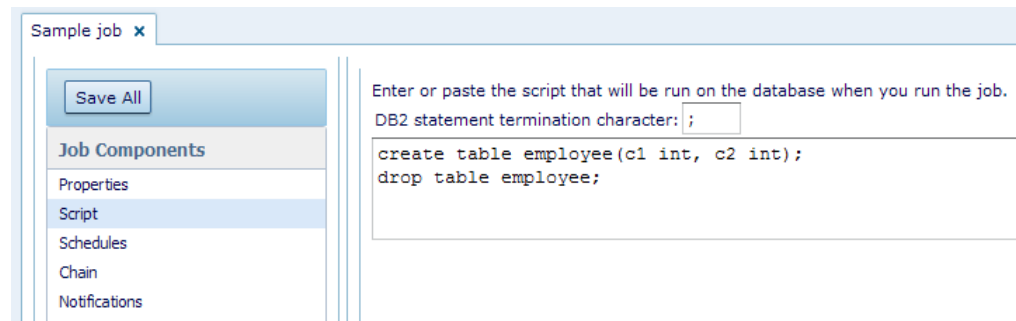

Figure 6.6 – The script component

4. (Optional) Click *Schedules* and then click *Add Schedule* to add one or more schedules to the job.
   A schedule defines when a job will be run, whether the job is repeating, and whether the schedule is limited in number of runs or in time. The schedule also defines one or more databases on which to run the job. A job can have any number of schedules attached to it, but each schedule only applies to one job.

5. Complete the schedule details and databases sections and then click *Apply Changes* or *Save All* to add the schedule to the job:

   - Specify the schedule details by selecting a start date and start time for the job. If you want the job to repeat, select the Repeats checkbox, and set the repetition parameters for the job. A schedule must be active to run the job.

Figure 6.7 – Specify a schedule.

- Specify the databases on which you want to run the job.
  Important: To be able to select a database, the database must first be added as a database connection in the web console. For information on how to add database connections, see *Chapter 4.*

Figure 6.8. – Select a database.

When you schedule a job on a single database, you can define the user ID that will run the job. If you schedule a job to run on more than one database, the job is run on each database by the user ID that is stored in the database connection for that database.

Important:

If the user ID that is used to run the job does not have the required permissions to perform the commands that are defined by the script for the database, the job fails with a permissions error. If you are scheduling the job for a single database, update the schedule with a user that has the required permissions. If you are scheduling the job to run on more than one database, use the Databases page to update the database connection with a user ID that has the required permissions.

6. (Optional) Click *Chain* to add a chain of additional jobs that will run conditionally when the main job has completed. In a chain, the main job is followed by a secondary job that is dependent on the outcome of the main job, and then followed by a finishing job that performs cleanup operations, such as RUNSTATS and BACKUP. You can add a chain of subsequent jobs that run depending on the outcome of the primary job.

Figure 6.9. – Select additional jobs that will be chained to the current job.

7. (Optional) Click *Notifications* and then *Add Notification* to configure email notifications to be sent to one or more users depending on the success or failure of the job. For more information about how to set up notifications, see [below]

8. Click *Save All* to save the job and its schedule to the web console. You can now select the job in the Job List tab to run it or edit the job components if needed.

6.4.2 Scheduling an existing job

You can add new schedules, or modify existing schedules for a job from the Schedules tab in the web console. Use the Schedules tab to create and manage schedules for the jobs you have created for your databases. The schedule also defines one or more databases on which to run the job. A job can have any number of schedules attached to it, but each schedule only applies to one job.

To add a schedule to an existing job:

1. In the Job Manager page, select the *Schedules* tab and click *Add Schedule*.
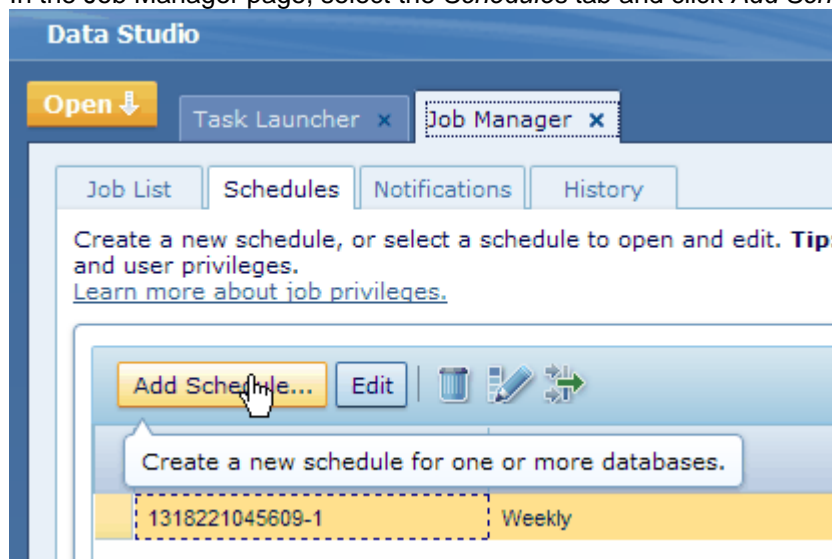
Figure 6.10 – Adding a schedule to an existing job.

2. In the Add Schedule wizard, select a job that you want to schedule and click *OK*. The job opens with the Schedules component selected.
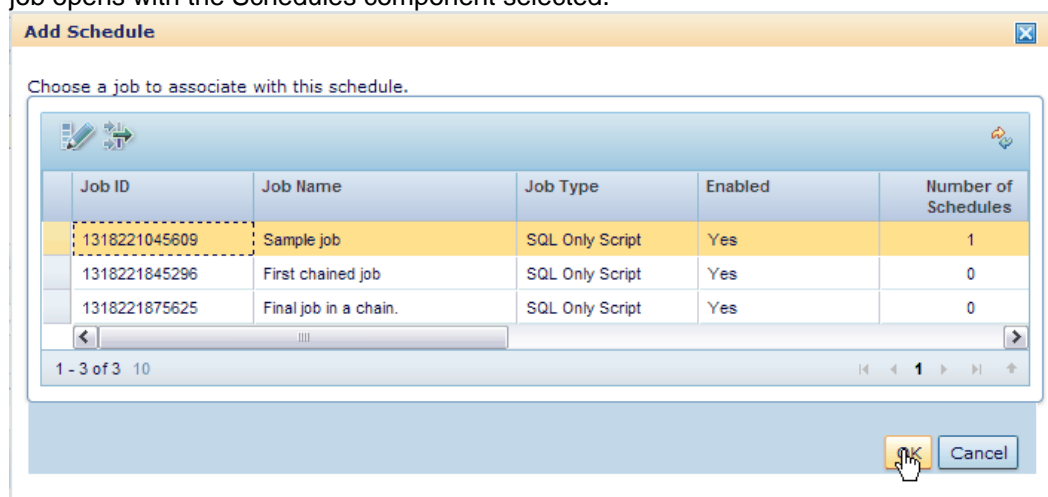
Figure 6.11 – Selecting the job.

3. Click *Add Schedule* to add one or more schedules to the job. A schedule defines
    when a job will be run, whether the job is repeating, and whether the schedule is
    limited in number of runs or in time.

    Complete the schedule details and databases sections:

    -   Specify the schedule details by selecting a start date and start time for the job.
        If you want the job to repeat, select the Repeats box, and set the repetition
        parameters for the job. A schedule must be active to run the job.

    -   Specify the databases on which you want to run the job.
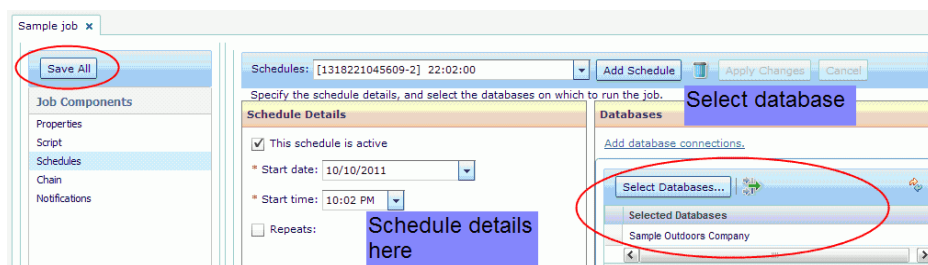
Figure 6.12 – Adding a schedule to a job.

4. Click *Save All* to save the new schedule to the web console. You can now select the
    schedule in the Schedules tab to edit it if needed.

6.5 Running a job without scheduling

You do not have to add a schedule a job to run it. You can run your jobs directly from the
Job List tab to verify that they work correctly before you schedule them.

To run a job directly:

1. From the Job List, select the job that you want to run and click *Run Job*.
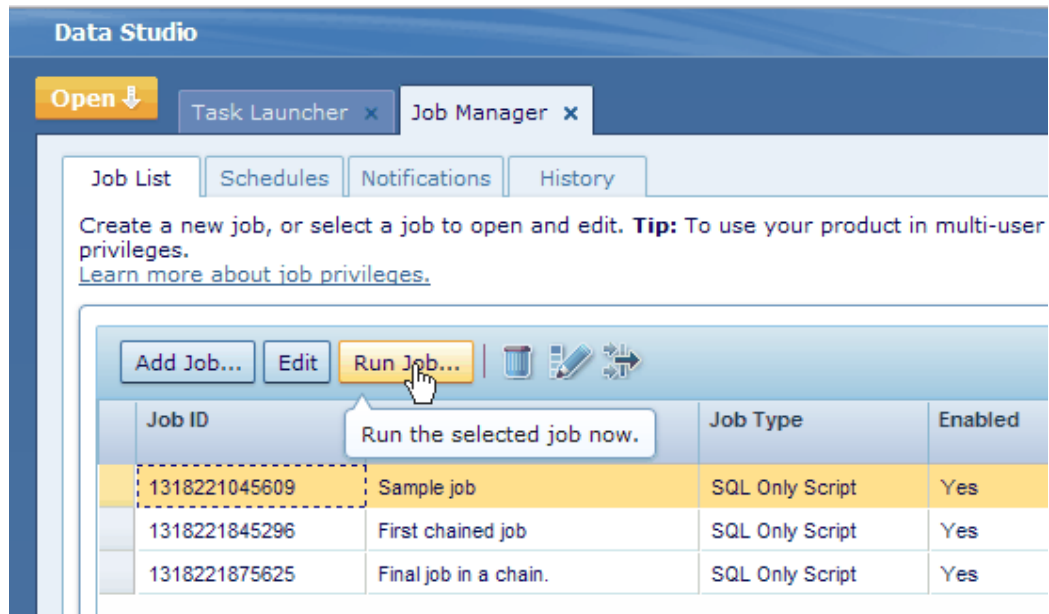
Figure 6.13 – Selecting to run a job directly.

2. Select one or more databases on which to run the job. When you select to run a job on a single database, you can define the user ID that will run the job or use the user ID that is stored in the database connection for that database. If you schedule a job to run on more than one database, the job is run on each database by the user ID that is stored in the database connection for that database.

3. Click *OK* to run the job on the selected databases. Open the History tab to see the job status details and the log file for the job.

6.6 Monitoring jobs - Notifications and job history

The jobs managed by the job manager run on the databases that you select when you attach a schedule or when you run the job directly. You can monitor the status of your jobs without having to log in to the web console by setting up email notifications for your jobs. You can also view the historical record for each job from the web console by browsing the complete job history overview or by accessing the job log files.

6.6.1 Setting up email notifications

The job manager notifications help you monitor the execution results for your jobs across multiple databases and schedules without requiring access to the web console.

Each job can have any number of notifications configured, and each notification can be set up with different conditions, a different set of users to notify, and different collections of databases to monitor.

To set up email notifications:

1. From the Job List, select the job that you want to add email notifications for and click *Edit.*

2. In the job that opens, from the Job Components menu, select *Notifications.*

3. In the Email Recipients field, enter an email address, or enter two or more email addresses separated by commas.

4. Select one or more databases. Notifications will be sent when the job runs on the database. Select the criteria for which a notification will be sent. The criteria can be that the job fails, that the job succeeds, or that the job fails or succeeds.

5. Click *Save All* to save the notification for the job.

Notifications are added specifically for a job. Each job can have one or more schedules attached to it, where each schedule has its own collection of databases that the job will run on.

Figure 6.14 – Configure notifications for the job.

Important:
To send notifications you must first configure the web console with the details about your outbound SMTP mail server so that information can be sent to e-mail addresses. From the web console, select *Open > Product setup > Services.* In the Services tab, select *Email service* and click *Configure* to configure the email service. To configure this service, you need an SMTP host name and port number for your email server. If the SMTP server uses authentication, you will also need the user authentication details.

6.6.2 Viewing the history of a job

The history tab of the main job manager page shows a basic status overview of the jobs that ran over the last few days. The job history is displayed for jobs that ran according to a schedule in addition to jobs that you ran manually. The job history is accessible at two levels of detail: the overview displayed in the History tab, and detailed job logs.
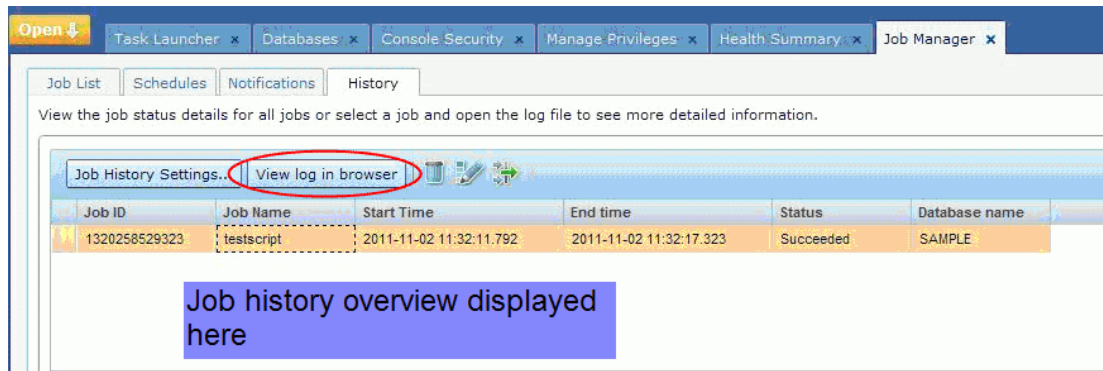
Figure 6.15 – Browsing your job history

To view more detailed information about a job, you can open the individual log for each job by selecting the job in the job history grid and clicking *View log in browser*. The log contains the output of the job script and lists any exceptions or other messages related to the job. If a job failed for some reason, the job log can help you to troubleshoot the problem.

Figure 6.16 – Viewing more details for your jobs.

By default, the job manager keeps the history of a job for three days. You can configure how long the job history records are kept in the job history settings. You can also set the type of job results that you want to keep. By default, both successful and failed job records are kept. To change the job history settings for the Data Studio web console, from the job history tab, click *Job History Settings*.
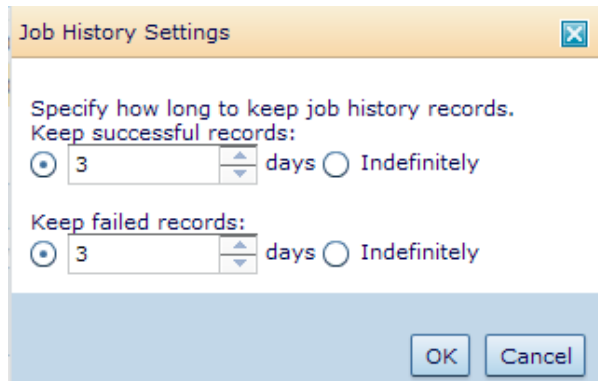
Figure 6.17 – Configure the job history settings.

6.7 Scheduling jobs from the Data Studio client

You can run and schedule jobs from the Data Studio client using the SQL script editor. When you have created your script you can select to schedule the script to run as a job in job manager. Selecting this will open the job manager either embedded in the workbench or in Data Studio web console in a standalone web browser window.

When you schedule a job from the SQL script editor, the job manager creates a new unique job with the script prefilled. You must then select one or more databases to run the job against. You can also schedule the job to run at a specific time, or save the job without a schedule if you want to run the job manually.

> **Note:**
> To schedule jobs from the Data Studio full client, you must first open the Data Studio web console embedded in the client. For more information, see *Appendix B.1 Integrating Data Studio web console with Data Studio full client.*
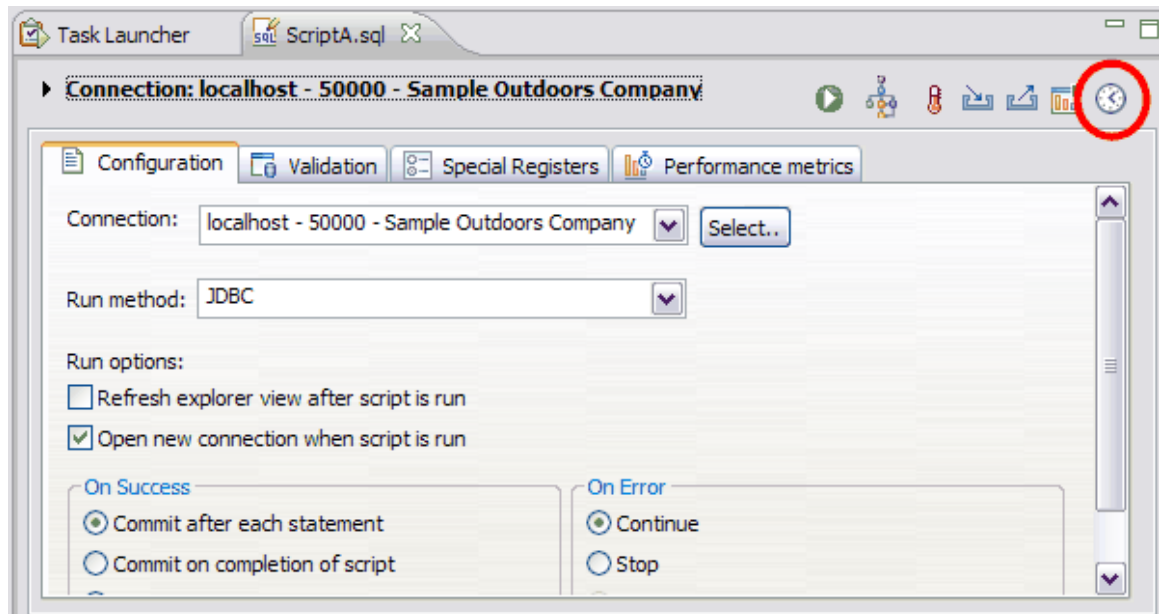
Figure 6.18 – Schedule a script from the Data Studio client SQL editor.

6.8 Exercises

In this set of exercises, you will create a job and run it on the Sample Outdoors Company database, then schedule the job to run at a later time. You will also verify that the job ran successfully by looking at the job history for the job.

1. Use the Job Manager page to create a new job on the Sample Outdoors Company database using the sample script in this chapter. Do not add a schedule to the job when you create it. You will add a schedule later.

2. Run the job manually from the Job List tab.

3. Use the Job History tab to verify that the job ran successfully.

4. From the Schedules tab, schedule the job to run in five minutes.

5. Use the Job History tab to verify that the job ran successfully.

6.10 Summary

In this chapter you have learned about the components of a Data Studio web console job, and about the various types of jobs that can be scheduled. You have also learned how to create and schedule jobs on your databases, and how to run a job directly without scheduling. At the end you have learned how to view the status of your jobs by email notification and by viewing the history of your jobs. And finally you have learned how to schedule a job from the Data Studio full client.

6.11 Review questions

1. True or false: A job contains all the information that is needed to run a script on a database server, including the script, the database information, and the date and time when the job will run.

2. If you want to configure the job manager to send notifications on successful job executions, what must you configure first in the Data Studio web console?

3. Which jobs can you run directly from the job list?

   A. Only jobs that have no schedules attached.

   B. Only jobs that have schedules attached.

   C. All jobs.

4. What job types are supported by the job manager?

5. If you schedule a job to run on more than one database, you can:

   A. Specify the user ID and password that will run the job for all databases.

   B. Specify the user ID and password that will run the job for one of the databases only.

   C. Not specify the user ID and password that will run the job for the databases. The user ID and password is specified in the database connection for each database.

7

Chapter 7 – Tuning queries

In this chapter, you'll learn more about some of tools and solutions from IBM that can help you address the bigger challenges of tuning your queries for DB2 to provide improved performance.

In this chapter you will learn how to:

- Configure DB2 to enable query tuning

- Use the SQL and XQuery Editor to generate query execution plans

- Capture SQL statements from various sources (such as a file or other products that have SQL statements)

- Invoke query tuning and how to analyze the results, run reports, and save the analysis

7.1 Query Tuning: The big picture

When you first learn to write SQL statements, you are probably most concerned about ensuring that you are formulating the query correctly to ensure that you're getting the right information back. However, if the query you write unexpectedly causes a slowdown on the application, any end users you are supporting may be very unhappy.

It is good practice for any SQL developer to gain a basic understanding of how the SQL they write is translated and executed by DB2 and to have some idea of whether the SQL they write is efficient. For example, if the SQL you write causes DB2 to have to scan the entire table to return each qualifying result, that could be a major performance problem that could be solved by including an index. Or maybe DB2 is making its choices for how to find the data based on inaccurate information stored in the DB2 catalog (that is, catalog statistics about the size of tables, data distribution, etc). Simply updating the statistics to be current might solve a problem.

The process of analyzing a statement's execution and changing the statement or environment to improve its performance is referred to as *query tuning*.

> **Note:**
>
> Understanding how DB2 chooses an access path and other concepts related to query tuning are beyond the scope of this book. It's a good idea to read up on some concepts. Here are some sources:
>
> DB2 Information Center:
> http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.perf.doc/doc/c0054924.html
>
> DB2 Best Practices article:
> http://www.ibm.com/developerworks/data/bestpractices/querytuning/
>
> Blog about writing SQL:
> https://www.ibm.com/developerworks/mydeveloperworks/blogs/SQLTips4DB2LUW/?lang=en

Data Studio can help make query tuning easier for you by providing the following capabilities that let you:

- Visualize the execution plan used by DB2 to run the statement

- Organize the syntax of the statement. For long and complex SQL, this tool can help you understand the statement better.

- Obtain feedback as to whether the existing statistics collected on your database are adequate. If not, the tool recommends statistics that should be collected.

Note that there are more extensive query tuning tools and advisors available in the chargeable product, InfoSphere Optim Query Workload Tuner 3.1. You can install this product in a "shell-sharing" environment with IBM Data Studio to extend the query tuning features described in this chapter.

7.2 Configuring DB2 to enable query tuning

To use query tuning, you must create a database connection as given in *Chapter 2* and that connection must be configured to enable query tuning. This section describes how to use the configuration wizard to configure query tuning.

Before setting up a connection, you need to change the IBM Data Studio display to use the *IBM Query Tuning* perspective.

Right click on the Windows menu and select *Open Perspective > Other*, then choose the I*BM Query Tuning* Perspective, as shown in *Figure 7.1*
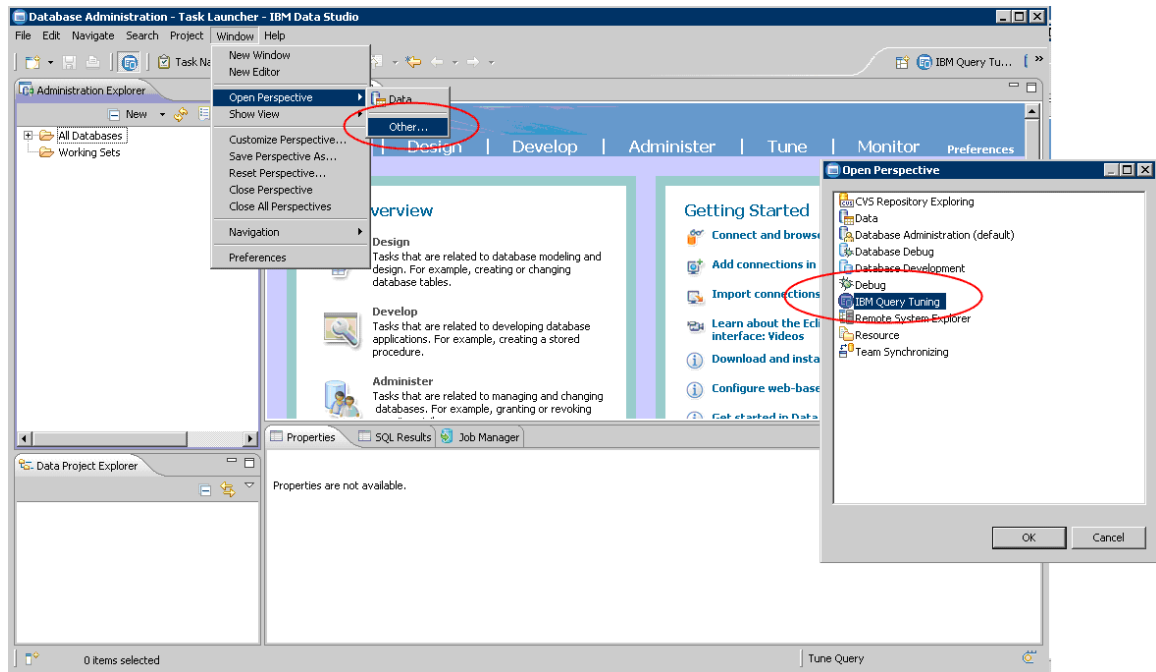
Figure 7.1 – Opening the IBM Query Tuning perspective

The IBM Query Tuning perspective is shown in *Figure 7.2.* Use the Data Source Explorer to connect to local or remote databases.
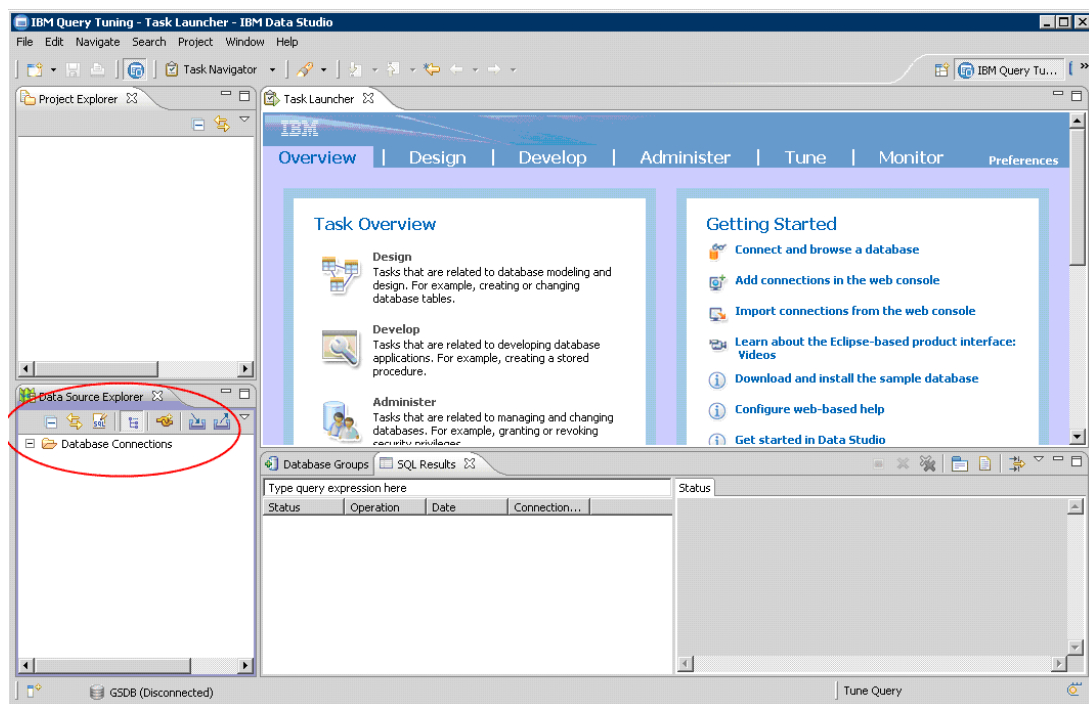
Figure 7.2 –IBM Query Tuning perspective

To create a new database connection, follow the steps in *Chapter 2*.

After you create the connection, configure the database connection by right clicking on the database name in the Data Source Explorer view and selecting *Analyze and Tune > Configure for Tuning.> Guided Configuration* as shown in *Figure 7.3.* When you choose Guided Configuration, DB2 will be configured automatically. This configuration includes the creation of the DB2 explain tables, if they don't already exist. For more information about explain tables see IBM DB2 9.7 Information Center topic "Explain facility" described here: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.perf.doc/doc/c0005134.html

You can choose to instead use the *Advanced Configuration and Privilege Management* as in *Figure 7.3*.
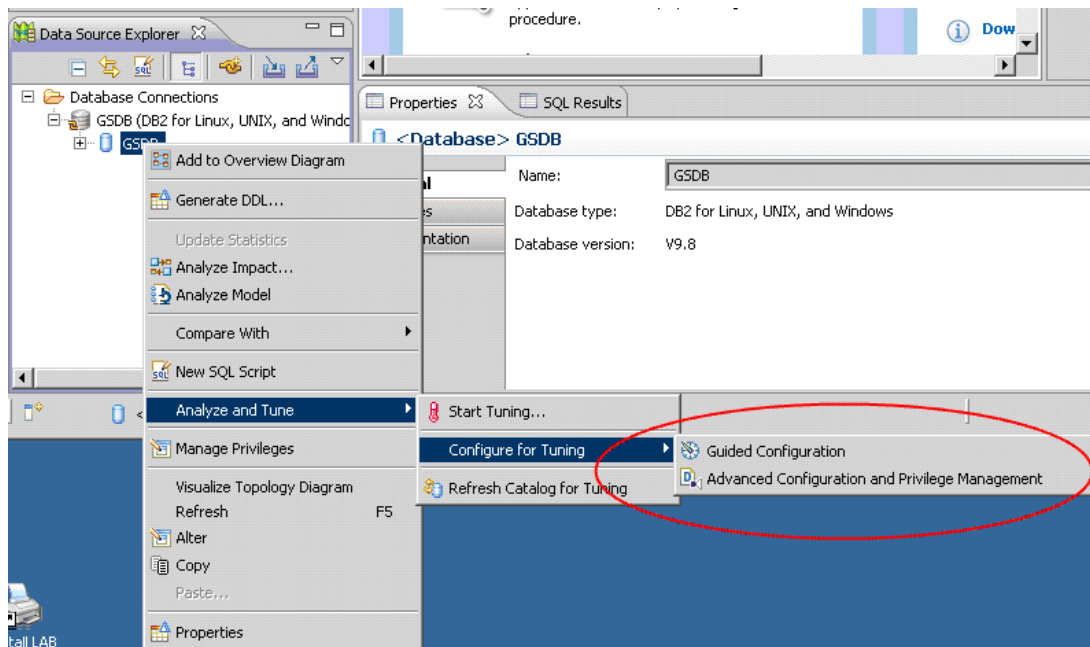
Figure 7.3 –Configuring tuning for a database connection.

On selecting the *Advanced Configuration* option, a window will open (*Figure 7.4*) If the EXPLAIN tables are not created on DB2 you can create them from the Advanced Configuration window by selecting the *CREATE* button.
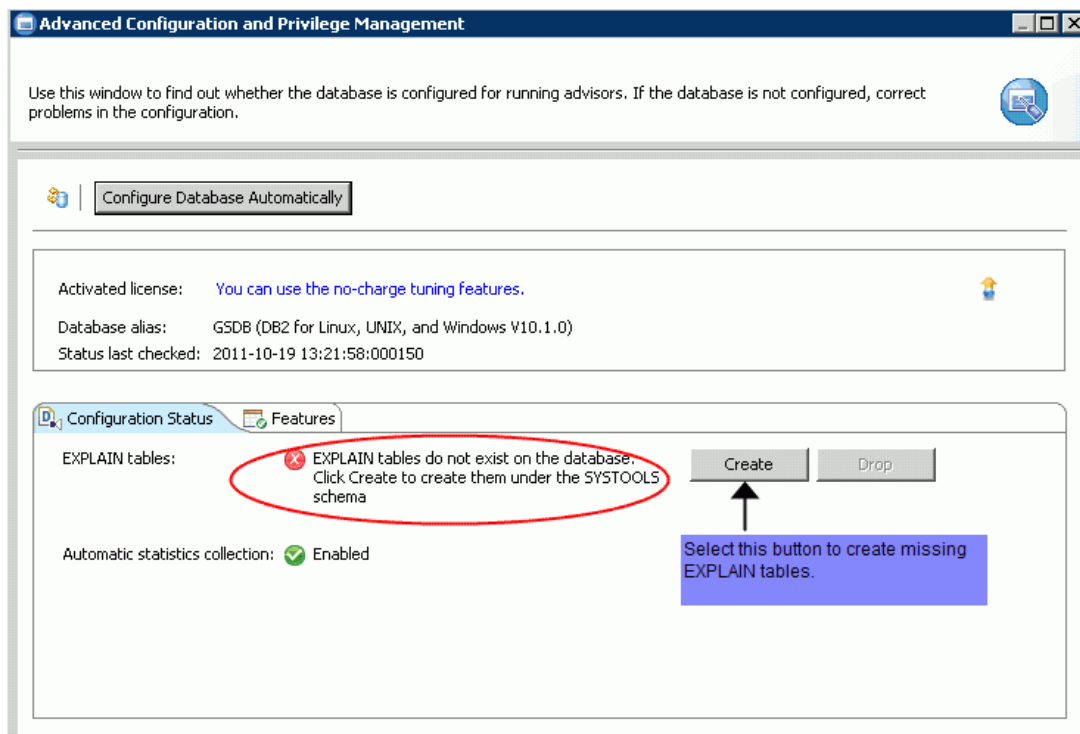
Figure 7.4 –Advanced configuration view

Once the explain tables are created properly, the EXPLAIN tables option will be shown with a green check mark as shown in *Figure 7.5*.
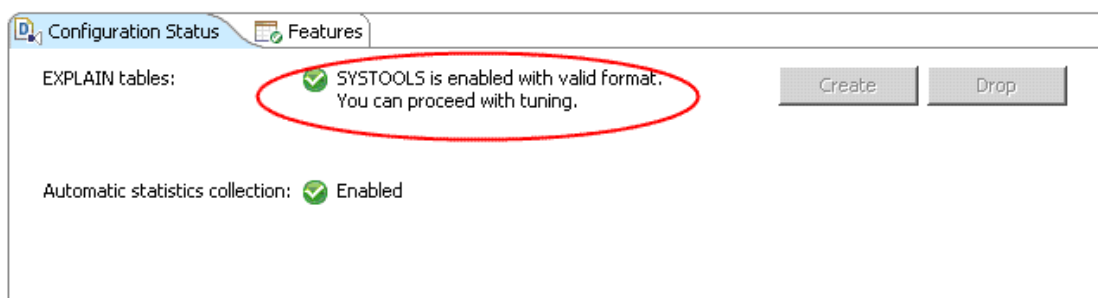
Figure 7.5 –Advanced configuration view of enabled EXPLAIN tables

The features that are now available can be seen by selecting the Features tab in the Advanced Configuration window. The available features will appear as in *Figure 7.6*. The available features include:

- The Statistics Advisor, that will analyze the usefulness of existing statistics collected and possibly recommend new statistics to collect,

- Query Formatting, which displays the formatted query's syntax and semantics,

- The Access Plan Graph, which displays the query execution plan used for a query by the database subsystem,

- Summary Reports to tie all this information together in a format that can be exported to other DBAs is needed.
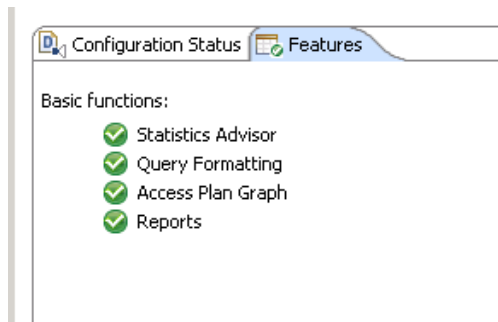
Figure 7.6 –Advanced configuration view of the available query tuning features

7.3 Start tuning

There are two ways to start tuning:

- Right click the database connection name in the Data Source Explorer view. In the menu, as shown in *Figure 7.7*, select *Analyze* and *Tune > Start Tuning*.
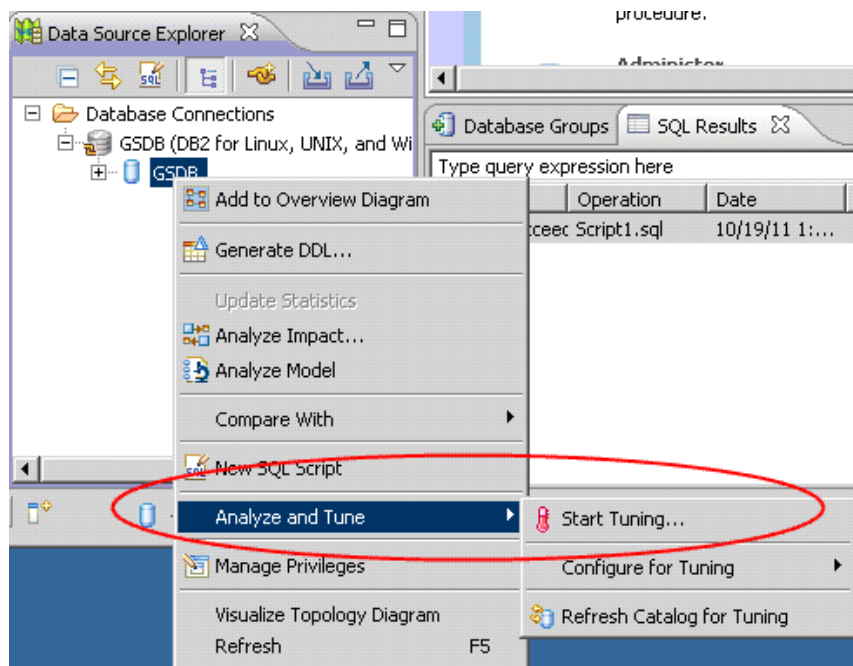
Figure 7.7 –Invoking the query tuning features

- Another way to invoke query tuning is from the SQL and XQuery Editor. To open the editor, select *New SQL Script.*

As described in *Chapter 5*, the SQL and XQuery Editor lets you enter and execute SQL statements. You can also obtain a Visual Explain, and invoke query tuning.

SQL statements can be added to the editor window as shown in *Figure 7.8*.
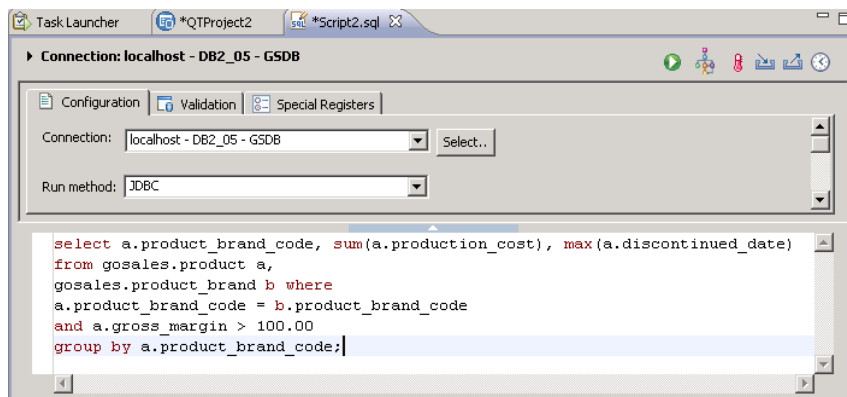
Figure 7.8 –SQL and XQuery Editor

You can invoke query tuning by highlighting a statement in the editor and right clicking in the editor, and selecting *Start Tuning* as shown in *Figure 7.9*.
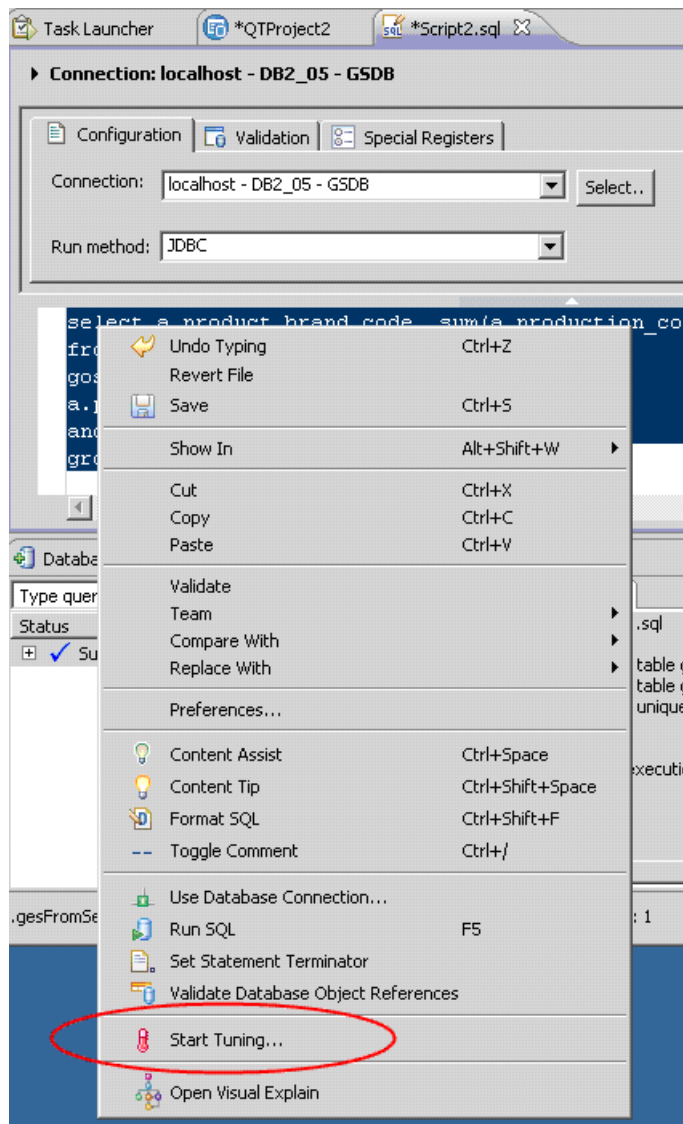
Figure 7.9 – Invoking the query tuning features from the SQL and XQuery Editor

7.4 Tuning an SQL statement

This section leads you through the typical workflow of tuning an SQL statement, from selecting the statement to be tuned all the way through analysis.

7.4.1 Selecting statements to tune (Capture view)

When you select *Start Tuning* the display will open the Query Tuner Workflow Assistant to the Capture view as shown in *Figure 7.10*, which lets you obtain a statement from various sources such as:

- The query editor ("Input Text")

- A file

- An Optim Performance Manager Repository

- The package cache

- A bound package

- The statements from an SQL procedure.

In this case, we are tuning the statement entered in the statement editor area for the "Input Text" option.

1. Enter a statement using the "Input Text" option and click on the *Invoke Advisors and Tools* button as in *Figure 7.10*.
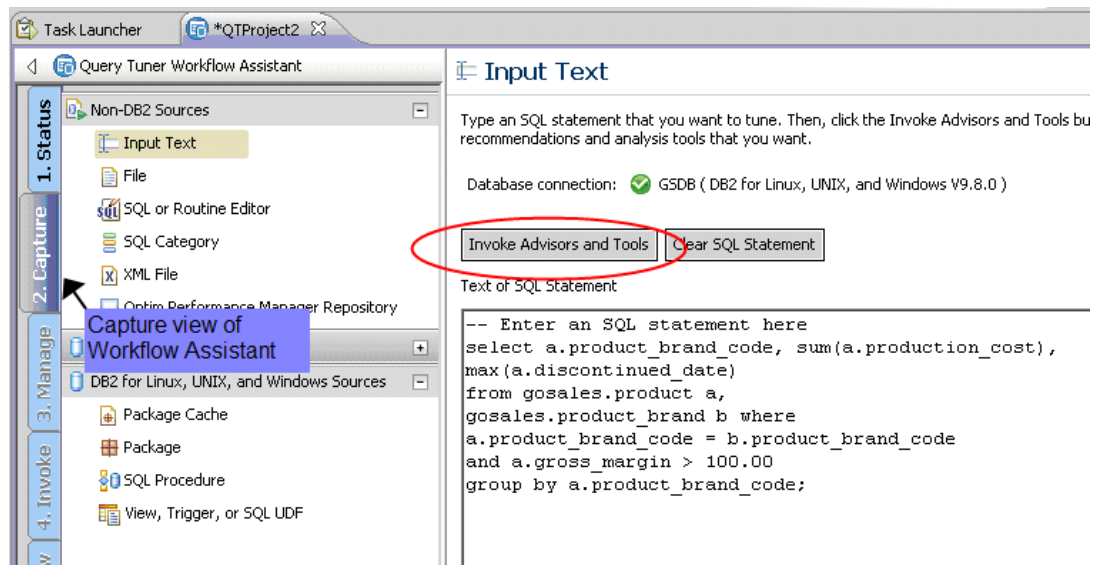
Figure 7.10 – Specifying the statement to be tuned in the Query Tuner Workflow Assistant (Capture view)

The display will change to the Invoke view.

7.4.2 Run query advisors and tools (Invoke view)

In the Invoke view, show in *Figure 7.11,* you will see the SQL statement that is to be tuned. You can edit the statement. You can also change the CURRENT SCHEMA used to explain the statement by changing the value in the Schema area.

1. To run the query tuning advisors and tools, select the *Select What to Run* button as shown in *Figure 7.11*.
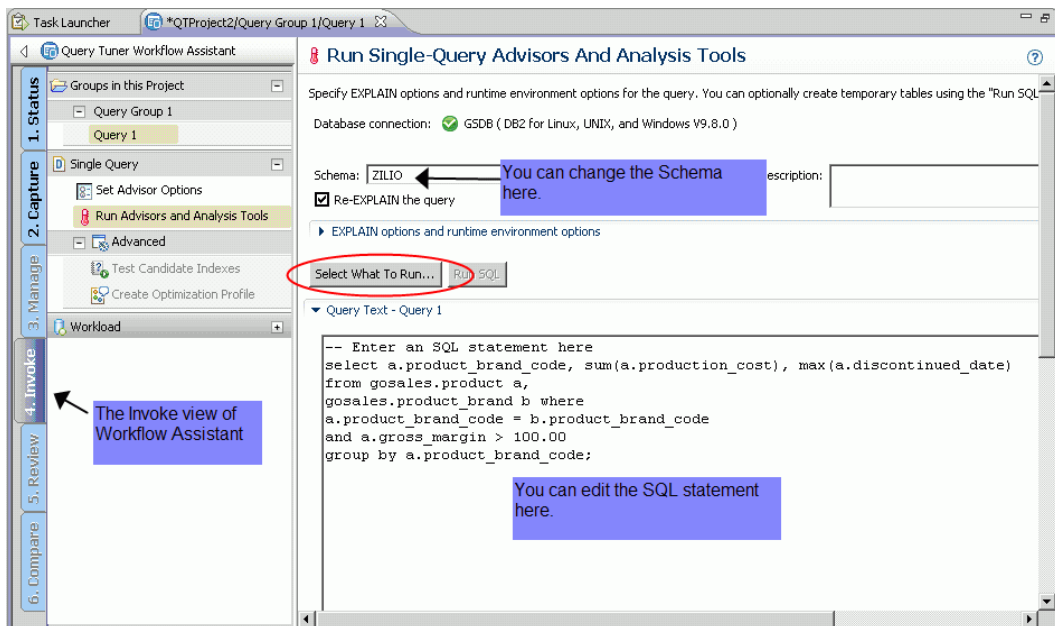
Figure 7.11 – Invoking the advisors and tools in the Query Tuner Workflow Assistant (Invoke view)

A new window will open to select query tuning advisors and tools that are to be executed as shown in *Figure 7.12*. Features that are not available to IBM Data Studio 3.1 are not selectable.

2. Select a checkbox of any feature you wish to execute. Once you have selected the features to run, select the *OK* button in the *Select Activities* window.
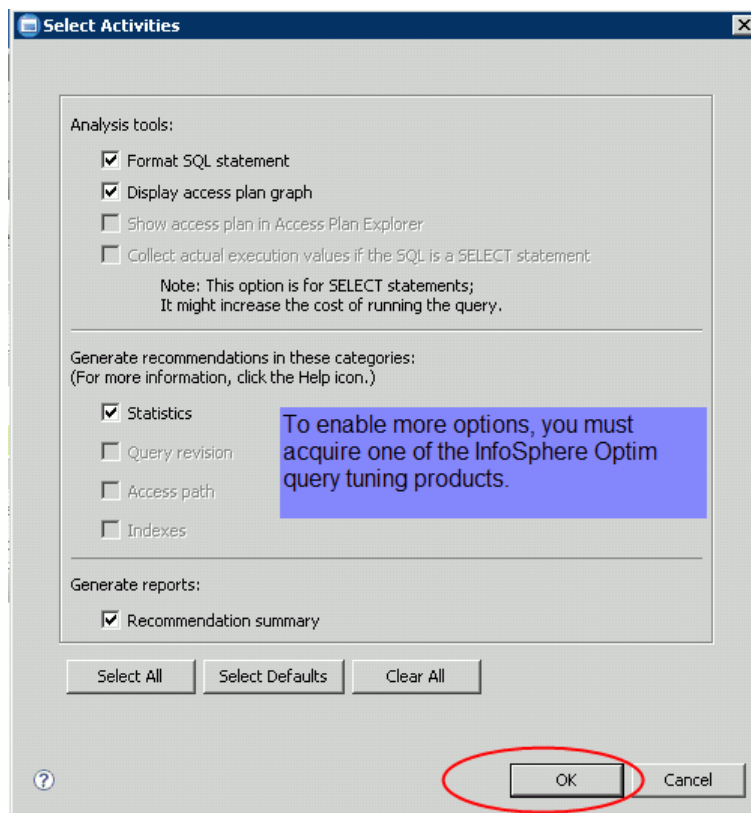
Figure 7.12 – Select Activities window to select query tuning features to execute

When the query tuning features that were selected above are completed, you will be placed into the *Review* view where the results and recommendations are provided.

7.4.3 Review the results and recommendations (Review view)

When the query tuning activities you selected are complete, you are placed into the Review view, as shown in *Figure 7.13*.
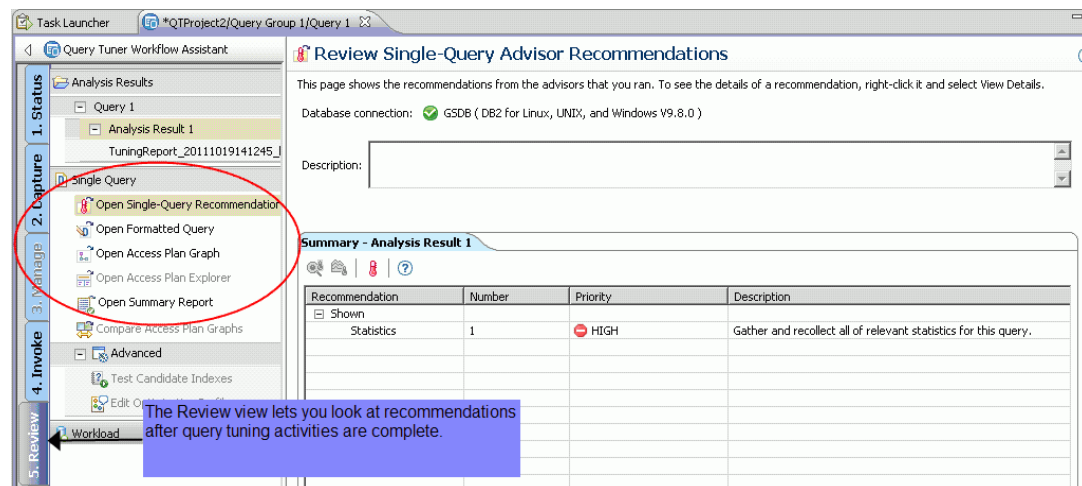
Figure 7.13 – Review view after query tuning advisors and tools are completed

On the left side, there is a list of options for *Single Query*. You can choose any of the ones that are not grayed out.

- *Open Single-Query Advisor Recommendations* displays the Statistics Advisor recommendations.

- *Open Formatted Query* option displays a formatted query, as shown in *Figure 7.14*. The formatting will place each column, table and predicate on separate lines with indentation so that you can analyze the syntax of the statement to determine if it makes sense. You can select a column, predicate or table in the formatted view and all column and predicate references as well as the table will be highlighted. The highlighting allows you to quickly identify how the table is being used and if it's being used properly in the statement.
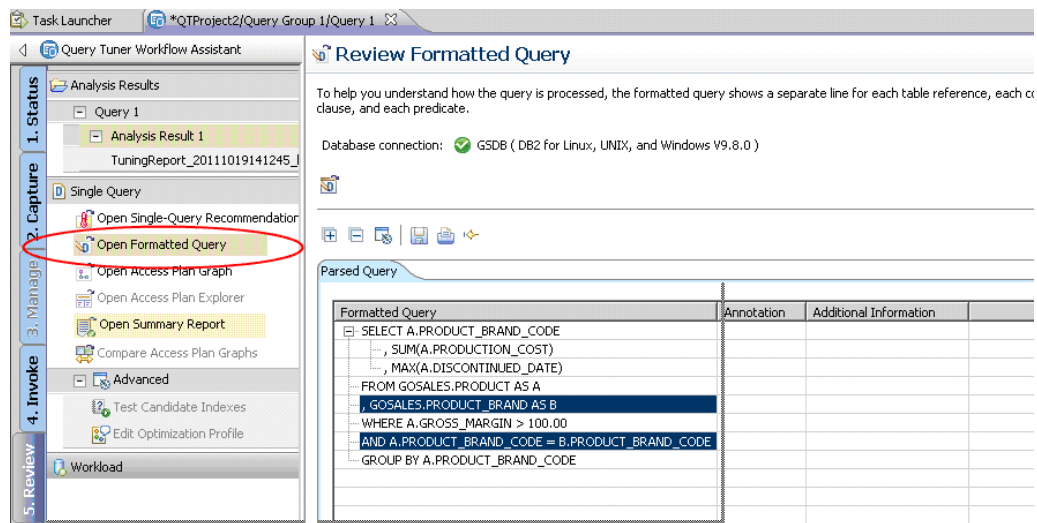
Figure 7.14 – Open Formatted Query view

▪ *Open Access Plan Graph* shows a visual representation of the access plan that DB2 uses to execute a statement, select as shown in *Figure 7.15*. The access plan graph is shown as a collection of connected operators.
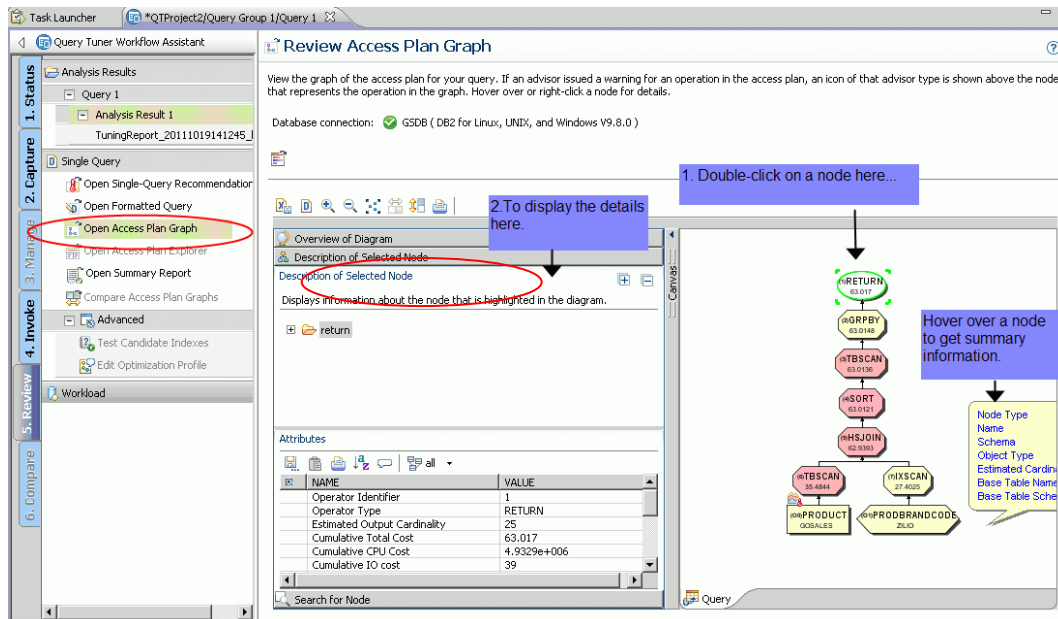
Figure 7.15 – Open Access Plan Graph view

In the graph area, you can hover over any operator and summary information will popup including the operator's estimated cardinality, and the total cost estimated by the DB2 optimizer.

For more operator details, you can click or double click on the operator to see that information under *Description of Selected Node.*

Note: The plan graph can also be generated using the *Open Visual Explain* option from the SQL and XQuery Editor as shown in *Figure 7.9*. More details on using Visual Explain option are provided in *Section 7.5*.

1. Open the Statistics Advisor summary by selecting the *Open Single-Query Advisor Recommendations* option as shown in *Figure 7.16*.
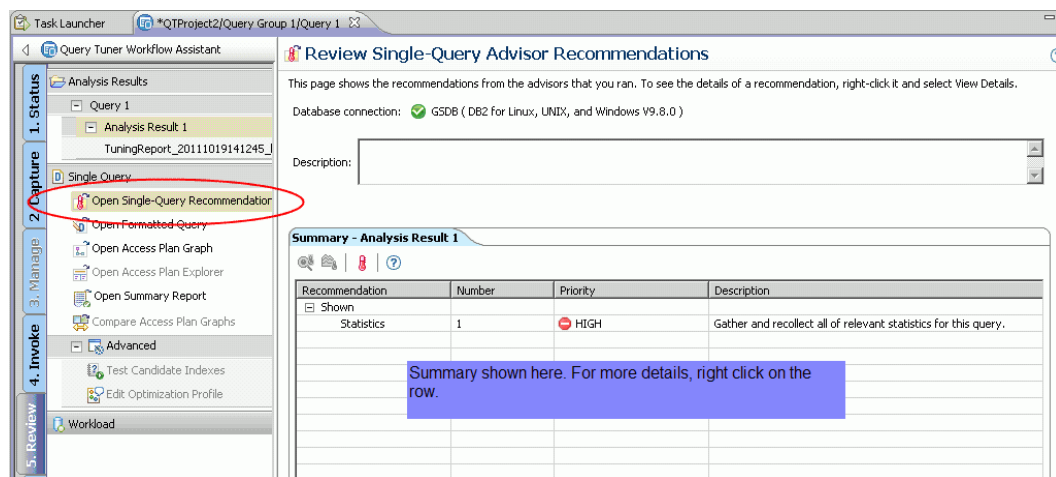
Figure 7.16 – Open Single-Query Advisor Recommendations view

2. To see details of the recommendation, you can double click on the recommendation in the summary tab, or you can right click on the Statistics row and select *View Details,* as shown in *Figure 7.17*.
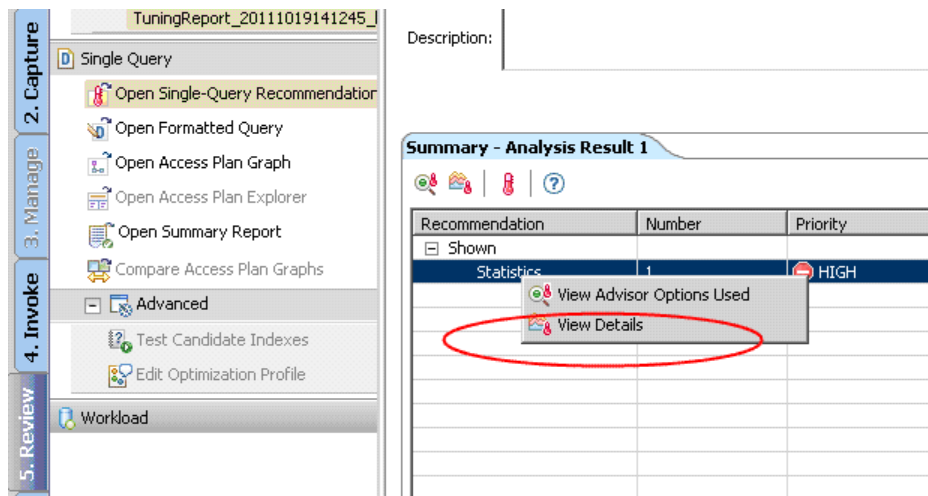
Figure 7.17 – Viewing Statistics Advisor options or recommendations

An example of the Statistics Advisor recommendation details are shown in *Figure 7.18.*
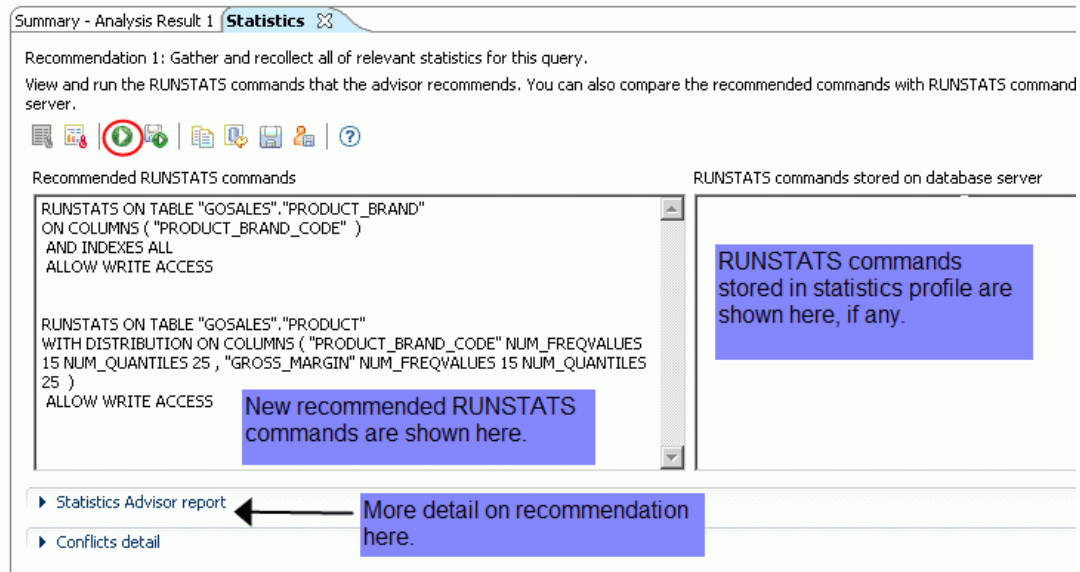
- The area labeled *RUNSTATS commands stored on data server* contains the previous RUNSTATS commands stored in the statistics profile; otherwise, this area will be blank.

- The area labeled *Recommended RUNSTATS commands* contains new recommended RUNSTATS commands. The advisor is able to recommend statistics it deems to be missing that the DB2 optimizer can make use of to improve the query access plan. These recommended statistics can include distribution statistics and column groups for base tables and materialized query tables.

- The *Statistics Advisor report* section contains more details on the recommendation.

Figure 7.18 – Statistics Advisor recommendations

When you are satisfied with the recommendation and want to proceed with executing on it, you can select the green "run" icon (⊙) as shown in *Figure 7.18* to execute the RUNSTATS commands.

7.4.4 Review the query tuner report

You can view a textual version of these advisors and tools in HTML format by selecting *Open Summary Report* as shown in *Figure 7.19*. The information in this report includes access plan information, such as the tables accessed, all columns of that table, statistics, indexes, and predicates. Also included are the RUNSTATS recommendations from the Statistics Advisor. You can pass this report to other users to share and compare results.
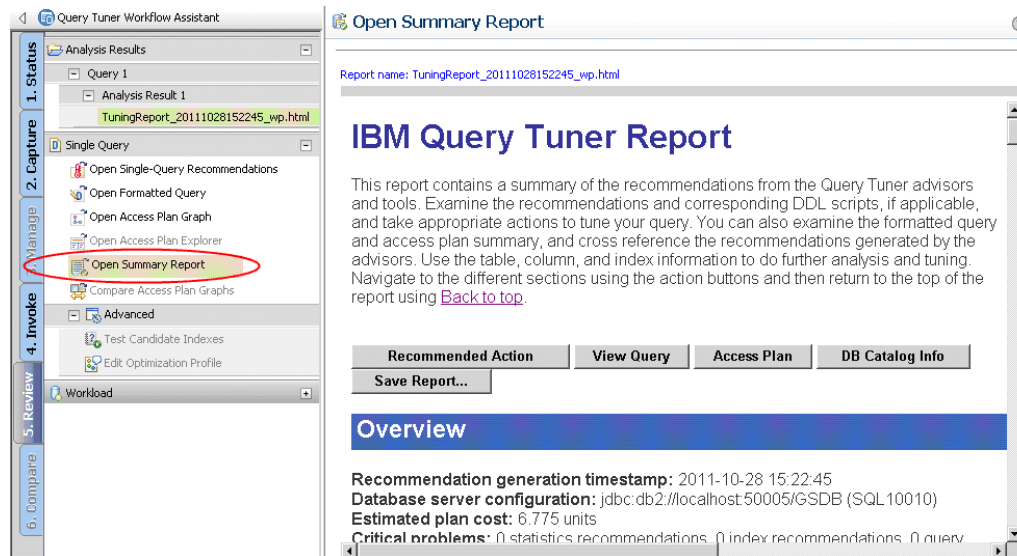
Figure 7.19 – IBM Query Tuner Report

7.4.5 Save the analysis results

You can exit a tuning session by selecting the X in the tab title for the tuning session you're in.

When you exit the query tuner workflow assistant, you will be prompted (as shown in *Figure 7.20)* to save the analysis result into a project in the *Project Explorer* area. The project you are saving is a Query Tuner project and will let you archive previous analysis results and compare them if needed.

Recommendation: Create a project for each different connection. The analysis result will be saved under a query group name in that project.

To save the result, select *Save and Exit.*
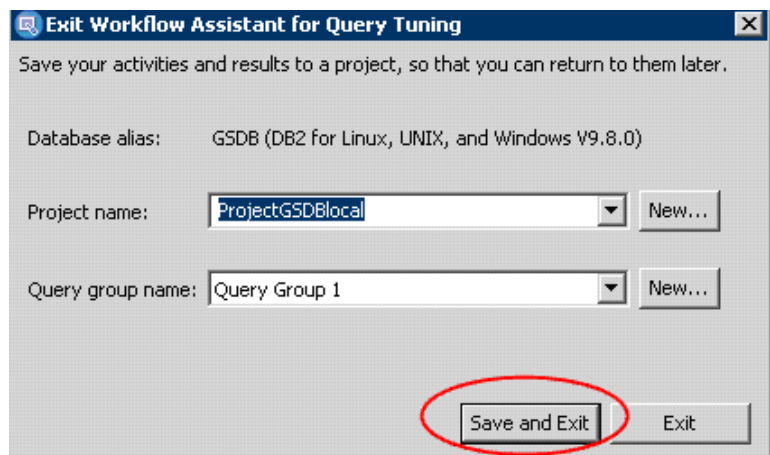
Figure 7.20 – Prompt to save the results from a tuning session

The analysis result is stored under the query group name as shown in *Figure 7.21*. You can store multiple results under this name for the same or different queries. When you want to view the analysis results again, simply double click the result you want to review. The query tuner workflow assistant will open with the analysis result. You can choose to rerun query tuning features by going to the *Invoke* tab, or you can view the analysis results under the *Review* tab. You can even re-capture a statement in the assistant from this previous result.
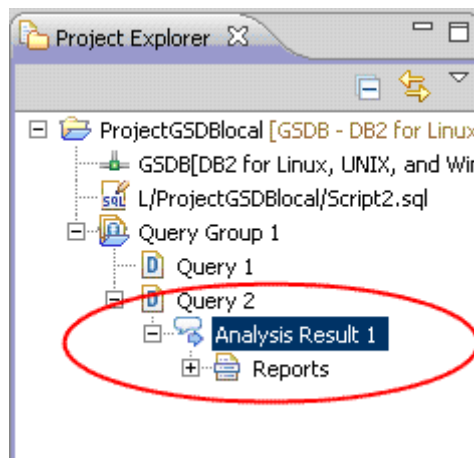
Figure 7.21 – Project Explorer area under the IBM Query Tuning Perspective

7.5 Invoking Visual Explain from the SQL Editor

If you are developing SQL statements and quickly want to see a visualization of the access plan, you can invoke Visual Explain directly from the SQL editor.

1. From the SQL Editor, right click on the query and select *Open Visual Explain*.

2. Enter the information required for collecting the data, as shown in *Figure 7.22.* You indicate the statement delimiter and whether to retain the explain information on the data server (that is, store the data in the explain tables).
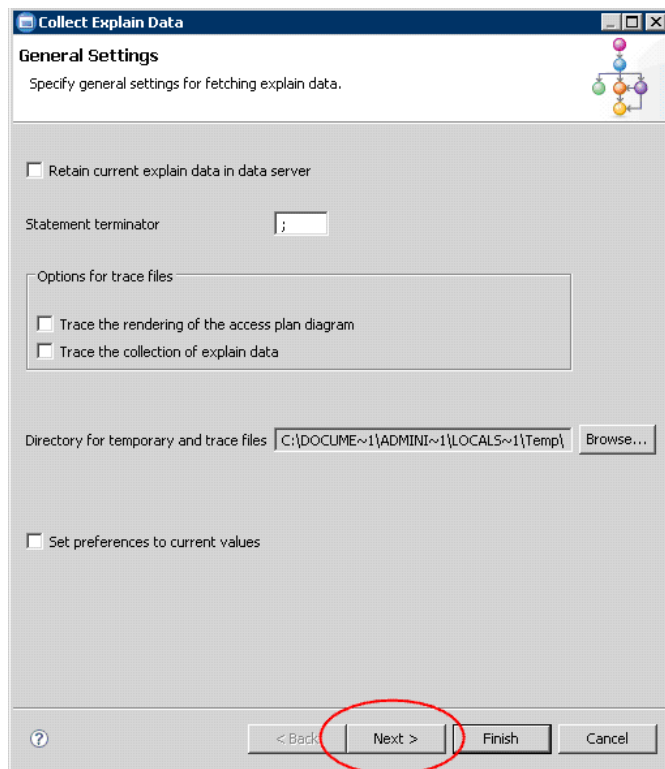
Figure 7.22 – Collect Explain Data window

3. Optionally, choose *Next* to change the default special registers used by the optimizer to generate the access plan as shown in *Figure 7.23*. These include the CURRENT SCHEMA and QUERY OPTIMIZATION LEVEL. If no value is added in a registers window, the defaults are used. Click *Finish* to obtain the Visual Explain.
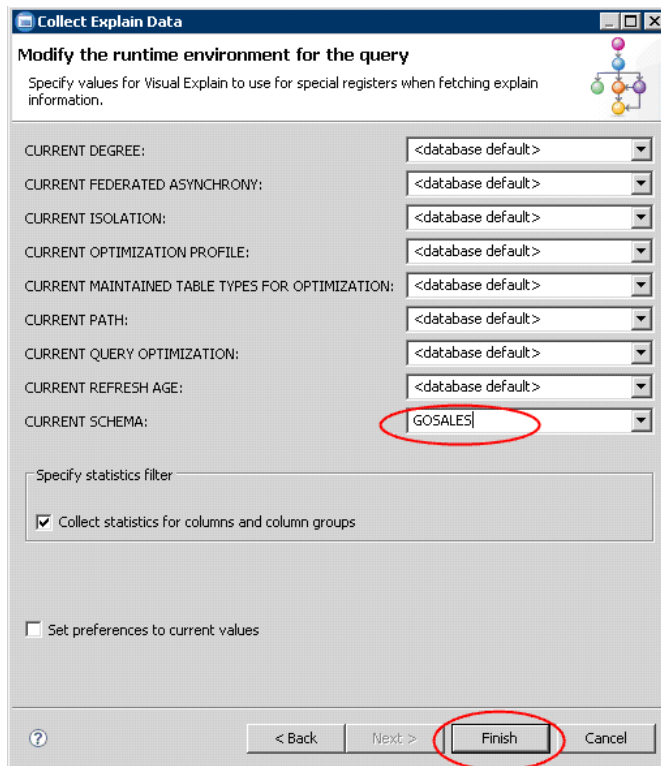
Figure 7.23 – Collect Explain Data window to set special registers

The Visual Explain has the same appearance as the Access Plan Graph as shown in
Figure 7.24.
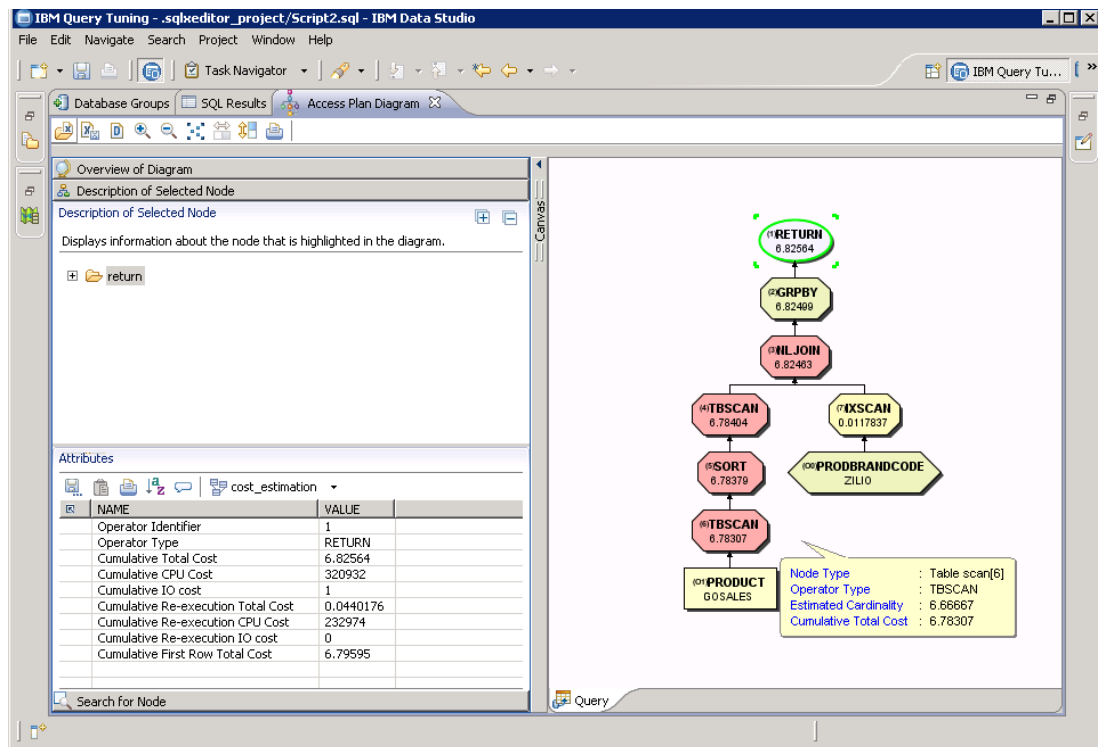
Figure 7.24 – Visual Explain display

7.6 Summary

In this chapter, you learned about the query tuning capabilities that are included in Data Studio and how to configure your DB2 system to enable query tuning. You learned how to choose SQL statements to tune, how to run the advisors and tools, and how to review the output and recommendations.

7.7 Review questions

1. After creating a database connection, what one-time activity do you need to do to enable the query tuning features?

2. What are the query tuner features available with IBM Data Studio 3.1?

3. Name the two places in which you can start the query tuner workflow assistant?

4. What are the two ways you can customize the query tuner features?

5. Where are query tuner analysis results stored?

6. Which query tuning feature will give you the query syntax output in a readable fashion:

    A. Statistics Advisor

    B. Visual Explain

    C. Access plan graph

    D. Query Formatter

7. Which query tuner feature provides the query execution plan:

    A. Statistics Advisor

    B. Visual Explain and Access plan graph

    C. Summary Report

    D. Query Formatter

8. Which button in the Invoke view of the Query Workflow Assistant will start the query tuner features:

    A. Select What To Run

    B. Open Single-Query Advisors and Tools Recommendations

    C. Invoke Single-Query Advisors and Tools

    C. Set Advisor Options

8

Chapter 8 – Developing SQL stored procedures

Stored procedures provide an efficient way to execute business logic by reducing the overhead of SQL statements and result sets that are passed back and forth through the network. Among the different languages that DB2 supports to write stored procedures, SQL is the language of preference because of its efficiency and simplicity. Moreover, SQL stored procedures are simpler to develop and manage.

Data Studio supports stored procedure development and debugging. In this chapter, you will learn:

- Why stored procedures are so popular and useful

- An overview of the steps to develop and debug a stored procedure

- How to create, test, and deploy a sample SQL stored procedure using Data Studio

- How to edit, and debug a sample SQL stored procedure using Data Studio

Note:

DB2 for Linux, UNIX and Windows supports stored procedures written in SQL (SQL PL), PL/SQL, Java, C/C++, Cobol, and CLR. However, from Data Studio you can only develop stored procedures using SQL, PL/SQL and Java. In this chapter, we focus on writing SQL procedures. More information can be found at:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.apdv.sqlpl.doc/doc/c0024289.html

8.1 Stored procedures: The big picture

Stored procedures can help improve application performance and reduce database access traffic. All database access must go across the network, which, in some cases, can result in poor performance. *Figure 8.1* illustrates the stored procedure data flow.
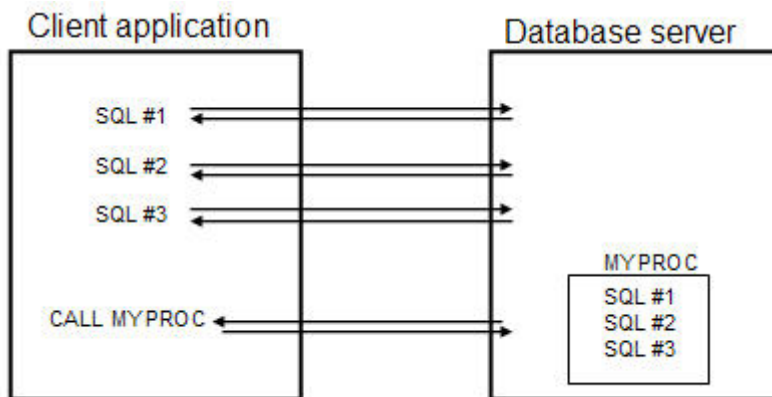
Figure 8.1 Stored procedure data flow

As shown in the above figure, an application must initiate a separate communication with the DB2 database server for each SQL statement. So, SQL #1, SQL #2, and SQL #3 require individual communication traffic.

To improve application performance, you can create stored procedures that run on your database server. A client application can then simply call the stored procedures (*MYPROC* in *Figure 8.1*) to obtain results of all the SQL statements that are contained in the stored procedures. Because a stored procedure runs the SQL statements on the server for you, the overall performance is improved. In addition, stored procedures can help to centralize business logic. If you make changes to a stored procedure, the changes are immediately available to all client applications.

Stored procedures are also very useful when deleting or updating large numbers of rows. You can specify a cursor in a stored procedure, and then loop through the result and delete or update rows. This reduces locking activity and is useful in an OLTP environment.

8.2 Steps to create a stored procedure

The recommended perspective for stored procedure development is the *IBM SQL and Routine Development* perspective. To switch to this perspective, select *Window -> Open Perspective -> Other -> IBM SQL and Routine Development* from the main menu.

As discussed in *Chapter 5*, you must create a data development project to store routines. Each data development project is associated with a single database connection. In this chapter, we'll be connecting to the *GSDB* sample database that has been used in previous chapters. See *Chapter 1* for information about downloading and creating the GSDB database, and *Chapter 2* for information about creating a connection to that database.

Data Studio provides helpful templates, editors and views to productively create, deploy, run, view, edit, and debug stored procedures. The overview of the steps to develop a stored procedure in Data Studio is shown in *Figure 8.2*.
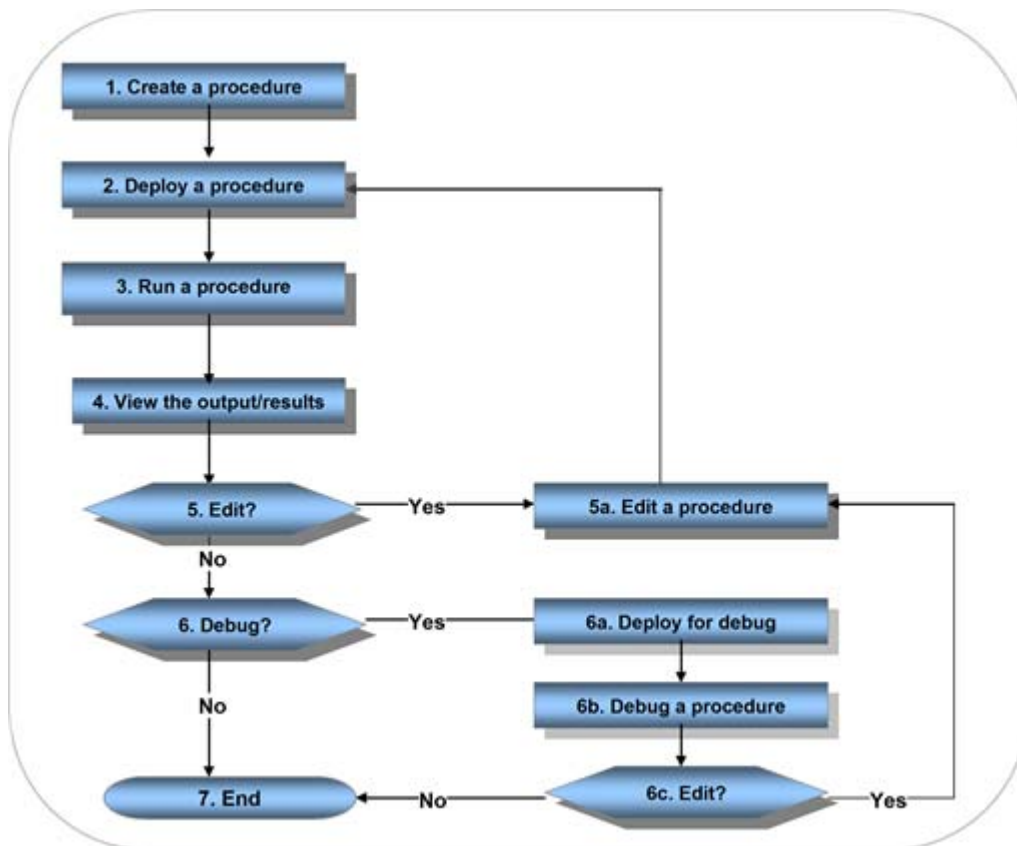
Figure 8.2- Steps to develop a stored procedure

1. The first step is to *create* the stored procedure. Data Studio supports a template based stored procedure creation, including the required input / output variables, and SQL statements. The stored procedure's source code is saved in your project workspace. The stored procedure appears in the *Data Project Explorer* view in the *Stored Procedures* folder under the project in which you created it.

2. Next, you *deploy* the stored procedure. When you deploy a stored procedure, Data Studio submits the `CREATE PROCEDURE` statement to the DB2 database server, which compiles it. If it is successfully deployed on the database server, the stored procedure can be found in the database when you drill down from the *Data Source Explorer*.

3. Next, *run* the stored procedure for testing purposes by providing any data for input variables.

4. *View* the output or results from your test run. When you run the stored procedure, you can determine whether the run is successful, and whether its result sets are what you expect. You can also test the logic of the routine and the accuracy of

output arguments and result sets. When you run a stored procedure from Data Studio, the results of the stored procedure are displayed in the *SQL Results* view.

5. At this point, you could optionally use the *Routine Editor* to make changes to the stored procedure depending on your business requirements. The routine editor is a tool to view and **edit** the source code. You need to redeploy the stored procedure whenever there are any changes.

6. Finally, the last step is to optionally **debug** the stored procedure, which requires that you actually deploy the stored procedure for debugging. In other words, there is an option on deployment that you must specify, to enable the integrated debugger. By stepping through your code while you are running in debug mode and viewing the results, you can discover problems with your stored procedure or better understand the functional behavior of your stored procedure in certain scenarios.

8.3 Developing a stored procedure: An example

The following example walks you through the steps to create, test, deploy, debug and edit a DB2 SQL stored procedure in Data Studio. In this example, we will create the stored procedure illustrated in *Listing 8.1.* This stored procedure will accept one input parameter and return one output parameter with the objective of testing a conditional **IF** statement.

```
CREATE PROCEDURE SP1 (IN p_in INT, OUT p_out INT)
--  DECLARE an input and output parameter
P1: BEGIN
     -- Code an IF statement
     IF p_in = 1 THEN
         SET p_out = 2;
     ELSEIF p_in = 2 THEN
         SET p_out = 3;
     ELSE
         SET p_out = 4;
     END IF;
END P1
```

Listing 8.1 – A sample SQL stored procedure

8.3.1 Create a data development project

In this example, we use the same workspace and database that was used in previous chapters.

1. Open *IBM SQL and Routine Development* perspective by selecting *Window -> Open Perspective -> Other -> IBM SQL and Routine Development* from the main menu.

2. From the *Data Source Explorer* view, connect to the **GSDB** database, and expand *Database Connections -> GSDB -> GSDB -> Schemas -> GOSALESCT* to view the schema that you will use in this chapter.

3. In the *Data Project Explorer* view**,** right-click the white space within the view and select *New -> Data Development Project* as shown in *Figure 8.3* below*.*
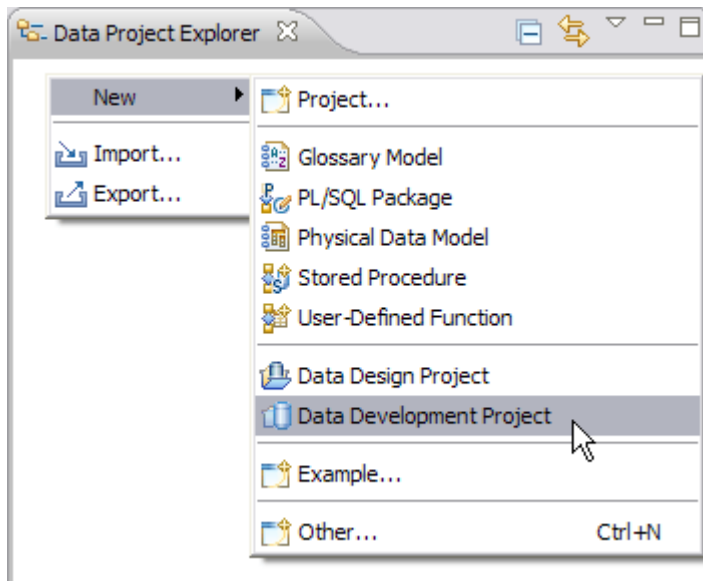
Figure 8.3 – Create a data development project

4. In the *Data Development Project* window, type *Stored Procedure Project* as *Project name* as shown in *Figure 8.4*, and select *Next.*
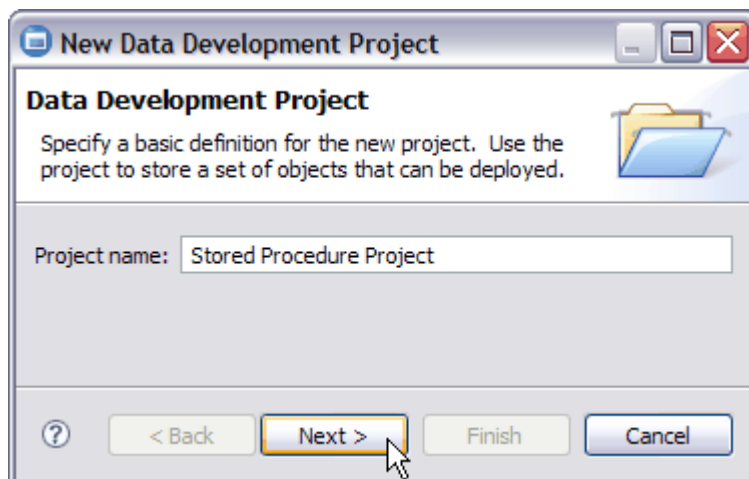
Figure 8.4 – Specify a project name

5. In the *Select Connection* window**,** select the *GSDB* connection as shown in *Figure 8.5*, and select *Next.*
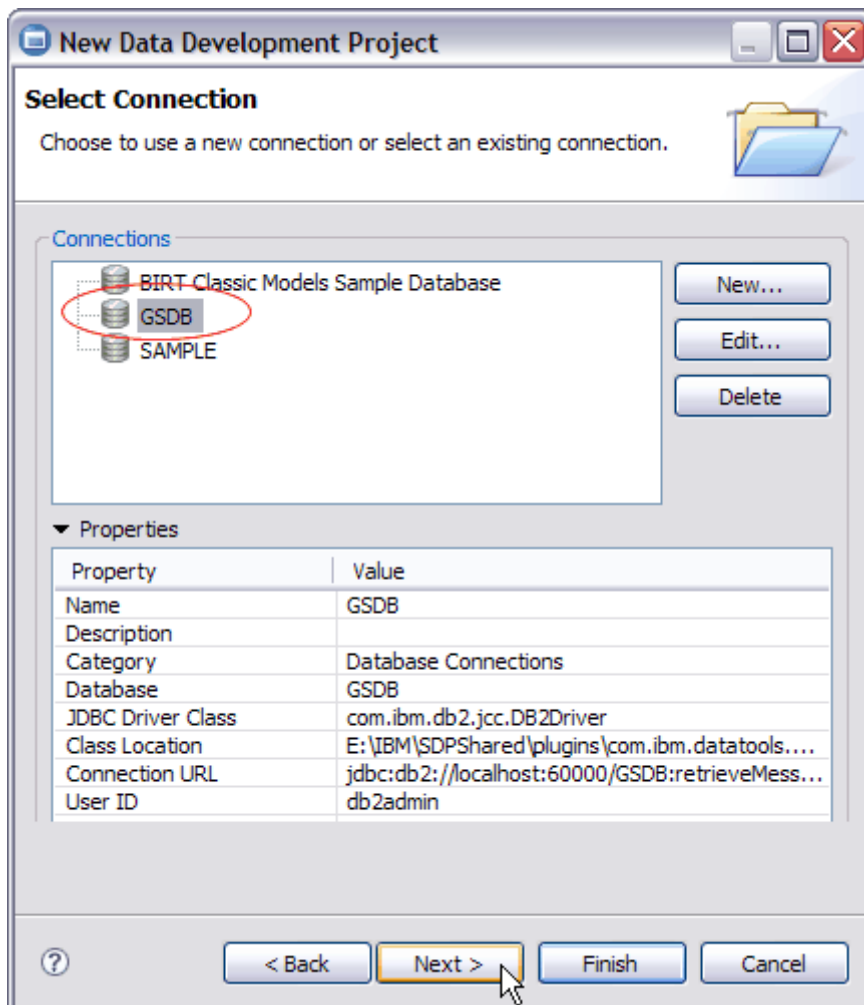
Figure 8.5 – Associate the data development project with GSDB connection

> 6. In the *Default Application Process Settings* window**,** select *GOSALESCT* as the default schema as shown in *Figure 8.6*, and select *Finish*.

Note:
If you don't change the schema here, the default will be a schema under the name you logged in as, such as DB2ADMIN.
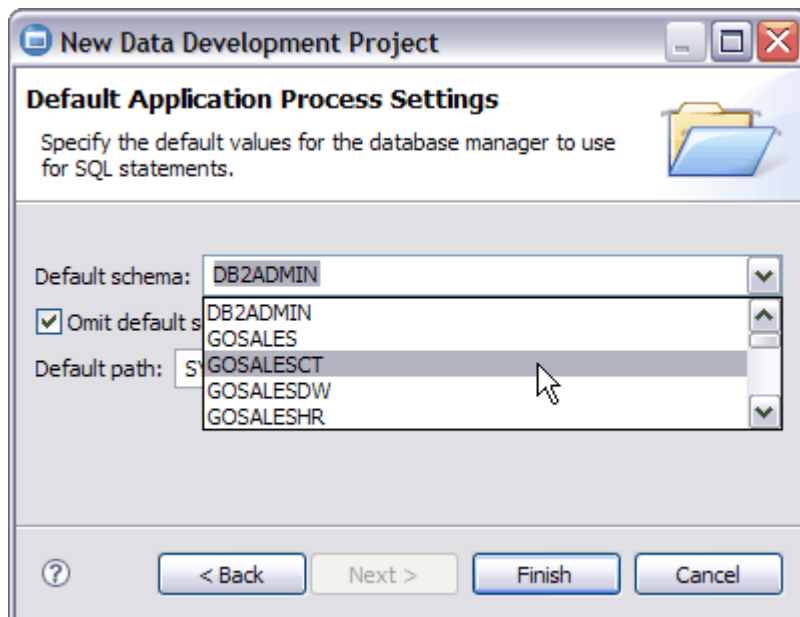
Figure 8.6 – Specify a default schema for the stored procedure

7. In the *Data Project Explorer* view, expand the hierarchy tree of *Stored Procedure Project*, to view its folders as shown in *Figure 8.7*.
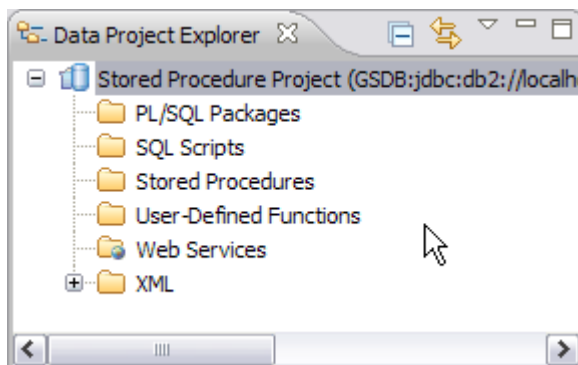
Figure 8.7 – View project resources

8.3.2 Create a stored procedure

Data Studio includes many routine templates that allow the user to create stored procedures easily. A routine template is a predefined text with some standard code, comments, best practices, etc. that makes it easy for developers to start creating new stored procedures.

To create a stored procedure using Data Studio:

1. In the *Data Project Explorer*, right-click the *Stored Procedures* folder under *Stored Procedure Project*, and select *New -> Stored Procedure* as shown in *Figure 8.8*.
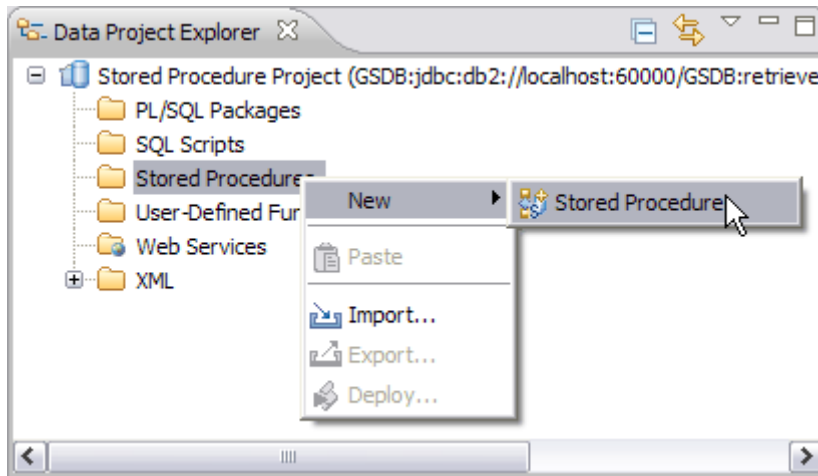
Figure 8.8 – Create a new stored procedure

2. Data Studio supports creating stored procedures in three languages on DB2 for Linux, UNIX, and Windows: Java, SQL and PL/SQL. In this example, you will create a **SQL** stored procedure. Type *STOREDPROCEDURE1* in the *Name* field, and select *SQL* as the *Language* as shown in *Figure 8.9*.

3. Under the '*Select a template*' section, a list of available templates based on the selected language and server type are displayed. Select the template with the name '*Deploy & Run: Return a result set*'. The *Preview* section shows the details of the template through *Template Details* tab, and the actual predefined code through the *DDL* tab. This is also shown in *Figure 8.9*.
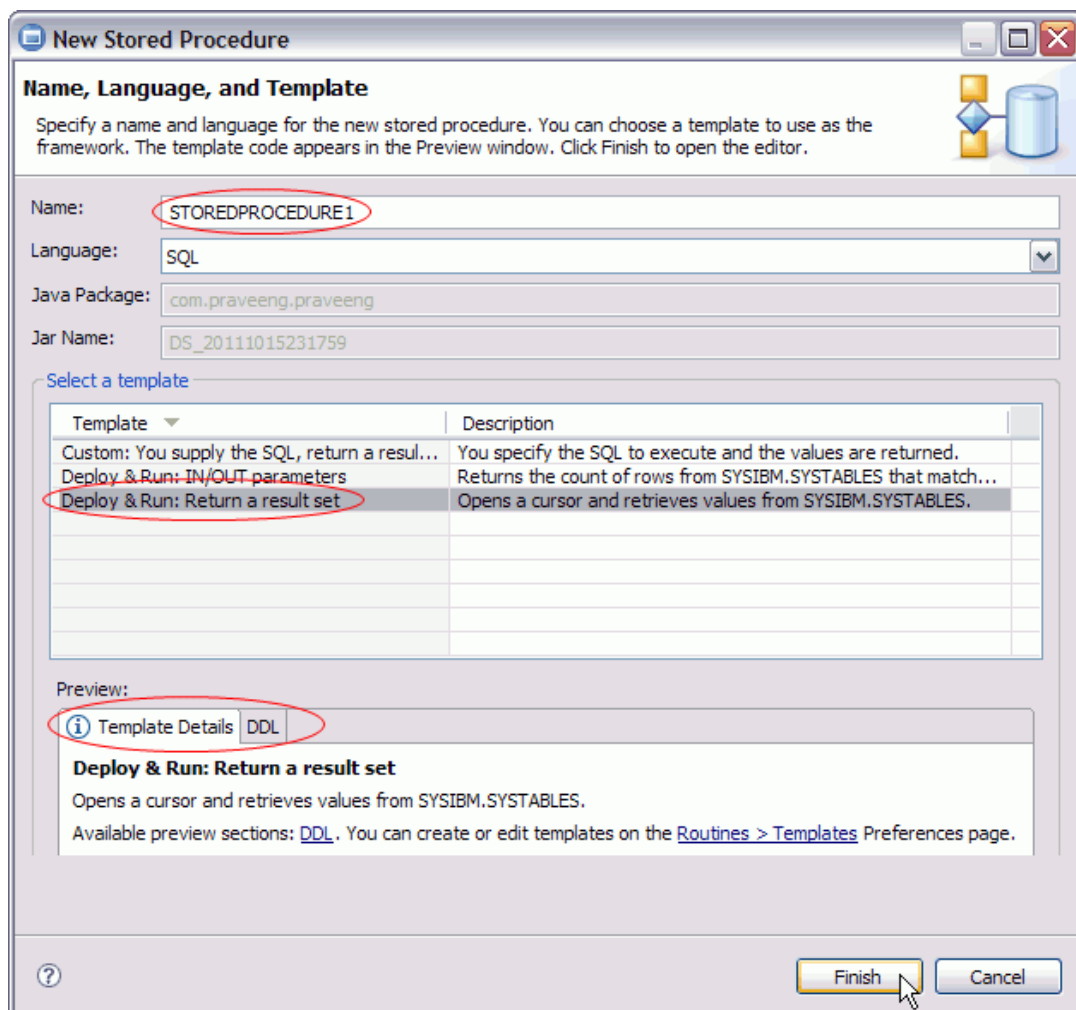
Figure 8.9 – Specify the procedure's name, language, and template

4. Click *Finish* to open the routine editor with the contents of the template pre populated for further editing. The new stored procedure *STOREDPROCEDURE1* is also added to the *Stored Procedures* folder in *Data Project Explorer*, as shown in *Figure 8.10*. For this example, do not make any additional changes in the routine editor.
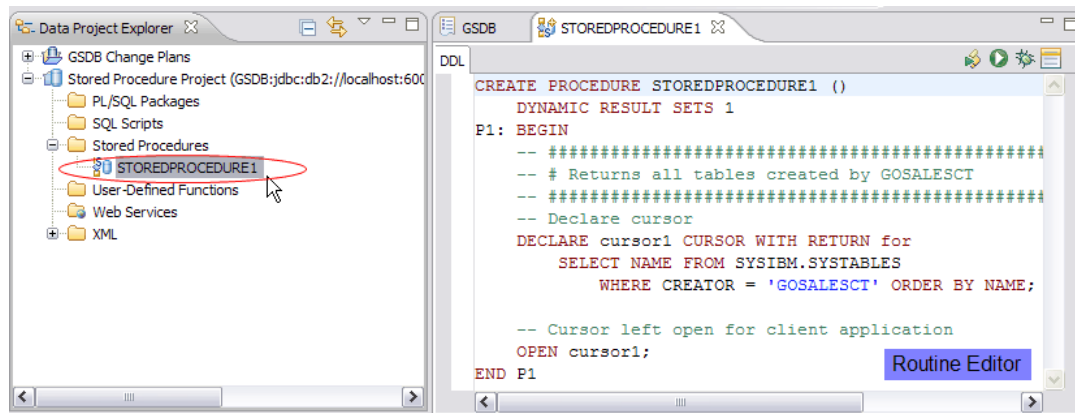
Figure 8.10 – View the stored procedures folder and the Routine Editor

---

Note:

For more information about using Data Studio for template based routine development, refer to the developerWorks article entitled '*IBM Optim Development Studio: Routine development simplified*' at: http://www.ibm.com/developerworks/data/library/techarticle/dm-1010devstudioroutines/index.html. This article uses an earlier version of Optim Development Studio product, but the information is the same for Data Studio.

8.3.3 Deploy the stored procedure

At this point in the stored procedure development process, the stored procedure source code exists as a file in your workspace. Before you can run your stored procedure against your database, you must deploy it to the database server. When deployed, the DB2 database server compiles the source code, and creates a new stored procedure object in the database with the compiled code. In case of any compile errors, the deployment will fail and an appropriate error will be sent back to the client.

To deploy the stored procedure using Data Studio:

1. Expand the *Stored Procedure Project* in the *Data Project Explorer* view, right-click *STOREDPROCEDURE1* from *Stored Procedures* folder, and select *Deploy* as shown in *Figure 8.11*.
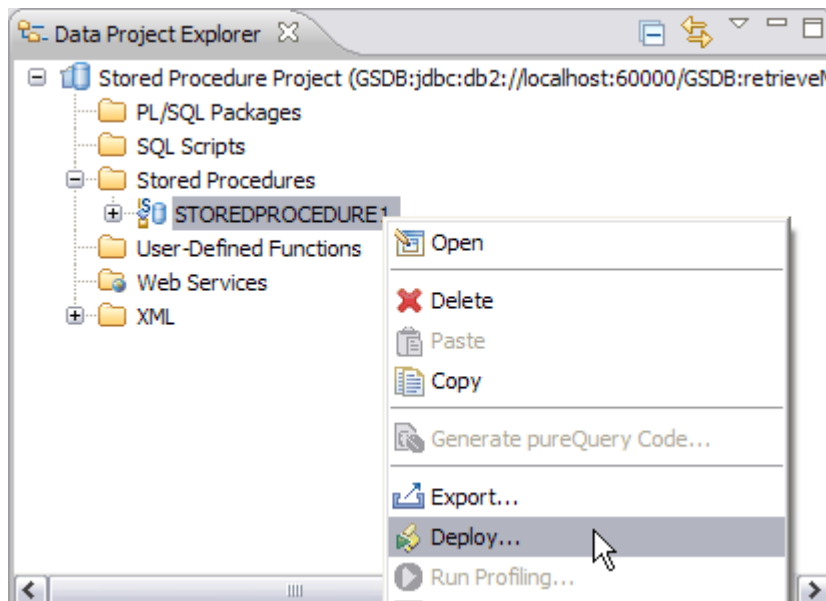
Figure 8.11 –Deploy the stored procedure

2. In the *Deploy Routines* window, make sure that the target schema is *GOSALESCT*.
    The *Target database* options allow the user to either deploy on the current
    database or on a different database. The *Duplicate handling* options can be used
    to specify if Data Studio should first drop any duplicates or not. Accept all defaults,
    and select *Finish* as illustrated in *Figure 8.12*.
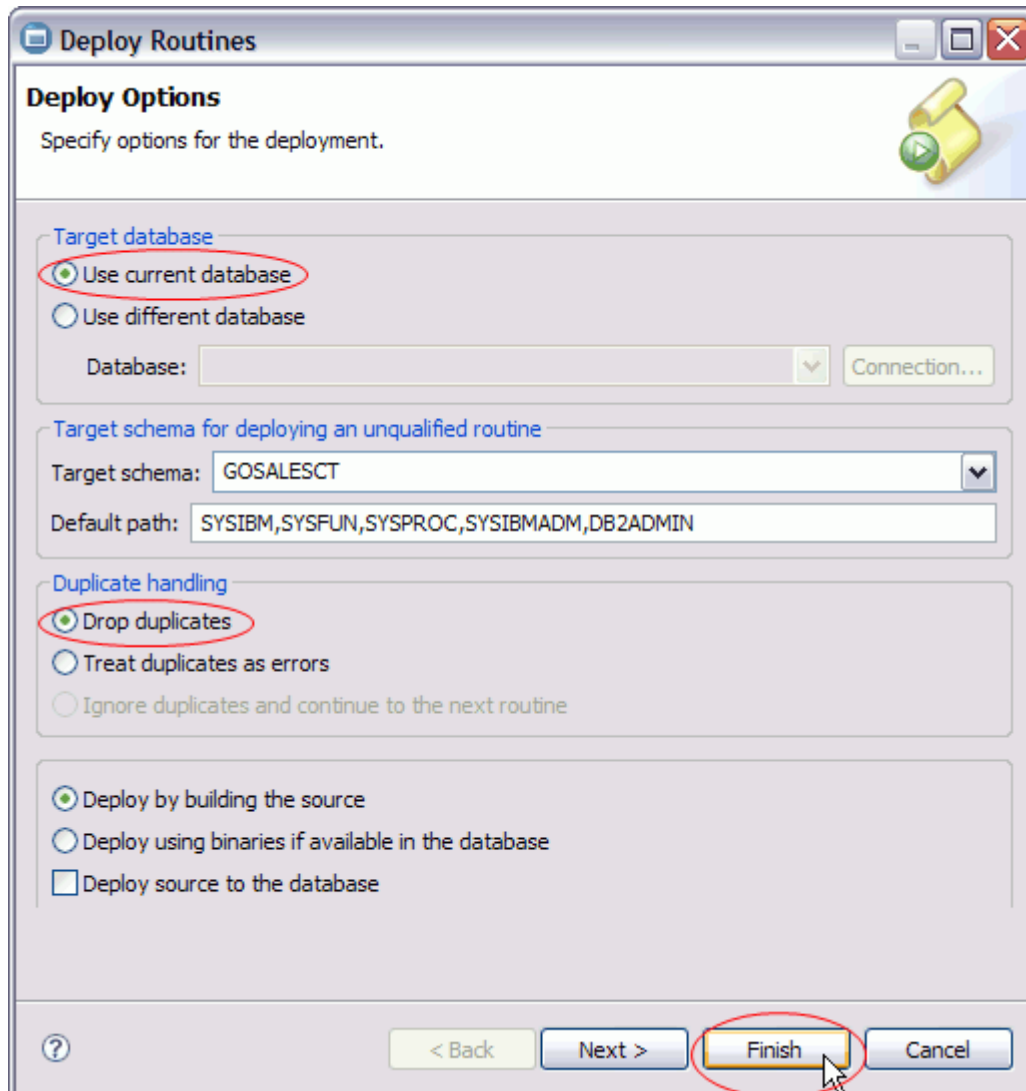
Figure 8.12 – Specify deployment options

3. Data Studio provides several views which provide quick access to informational output. Look at the entry for your *Deploy GOSALESCT.STOREDPROCEDURE1* operation in the *SQL Results* view. Wait until the operation completes, then verify that the deploy operation shows *Succeeded* as shown in *Figure 8.13.* The *Status* tab on the right shows more detailed output.
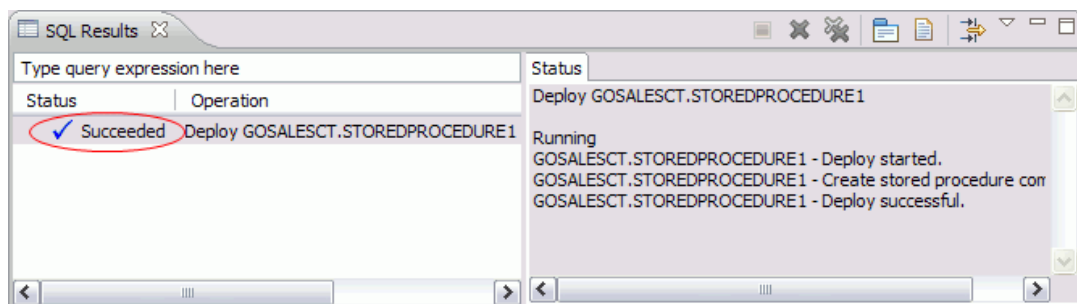
Figure 8.13 – View the deployment status

4. When you successfully deploy a stored procedure to the DB2 database server from the *Data Project Explorer* view, this new stored procedure object will be reflected in the *Stored Procedures* folder of the respective database in the *Data Source Explorer* as well. To verify this, expand *Database Connections -> GSDB -> Schemas -> GOSALESCT -> Stored Procedures* folder in *Data Source Explorer* and observe the entry for *STOREDPROCEDURE1*. This is shown in *Figure 8.14*. If the new object is not yet visible, right click *Stored Procedures* folder, and select *Refresh*.
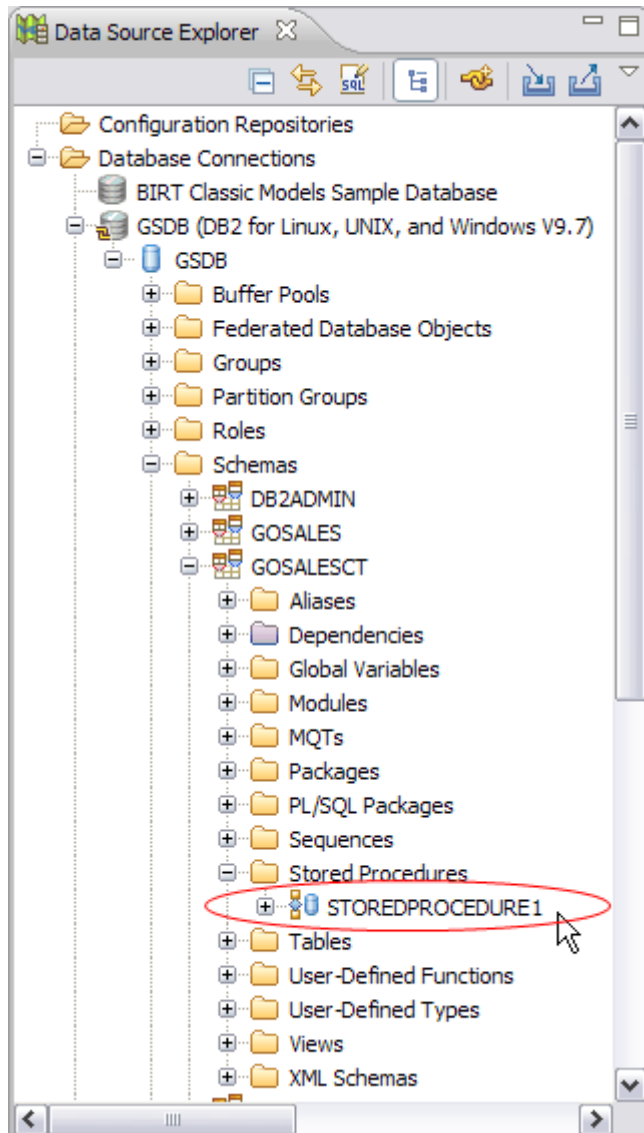
Figure 8.14 – The new stored procedure appears in the Data Source Explorer

8.3.4 Run the stored procedure

After you deploy your stored procedure to the database, you can run the stored procedure. From the *Data Project Explorer*, navigate to the *Stored Procedures* folder in *Stored Procedure Project*, right-click *STOREDPROCEDURE1*, and select *Run* as shown in *Figure 8.15*.
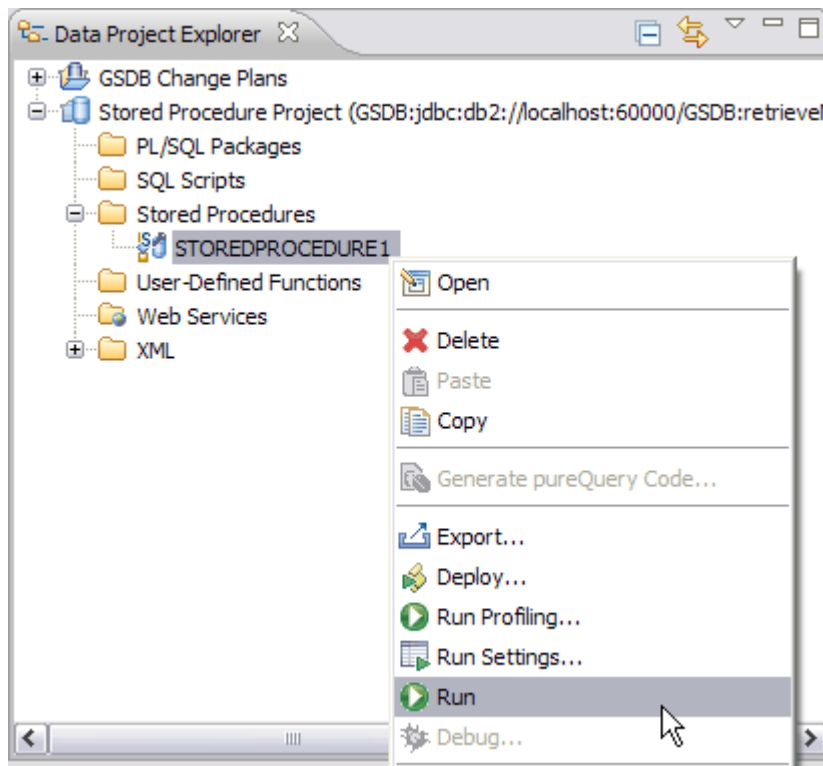
Figure 8.15 –Run the stored procedure

8.3.5 View the output

The *SQL Results* view shows the status of running your stored procedure. The *Status* column of this view indicates success or failure of the operation. The Operation column shows the command type or the actual command itself. The *Status* tab on the right shows detailed output. In this case, since the stored procedure returns a result set, its corresponding rows are shown in a separate tab *Result1*. This is illustrated in *Figure 8.16*.
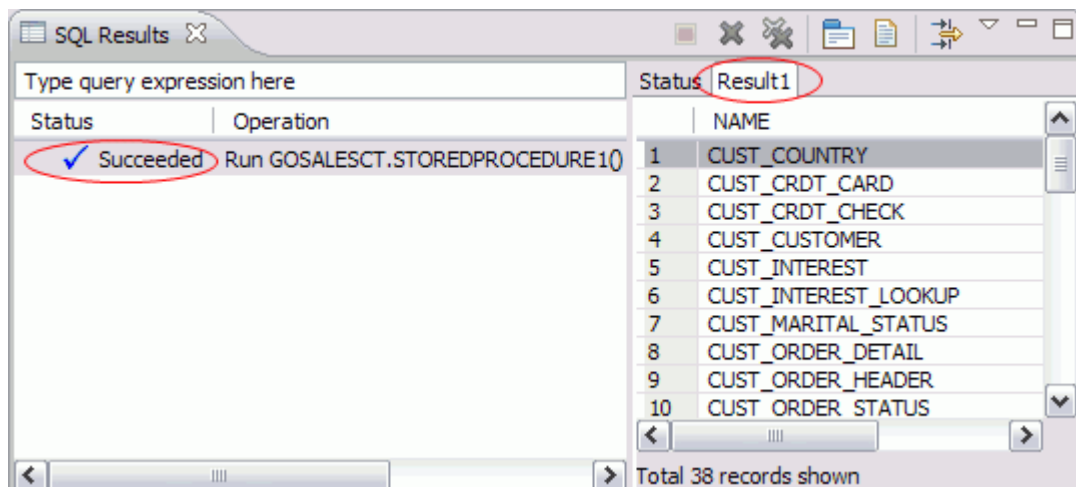
Figure 8.16 – View results of stored procedure execution

8.3.6 Edit the procedure

In Data Studio, a stored procedure can be viewed and edited in the *Routine Editor*. To
open a stored procedure in the *Routine Editor*, double-click the stored procedure name or
right-click and select *Open* as shown in *Figure 8.17*.
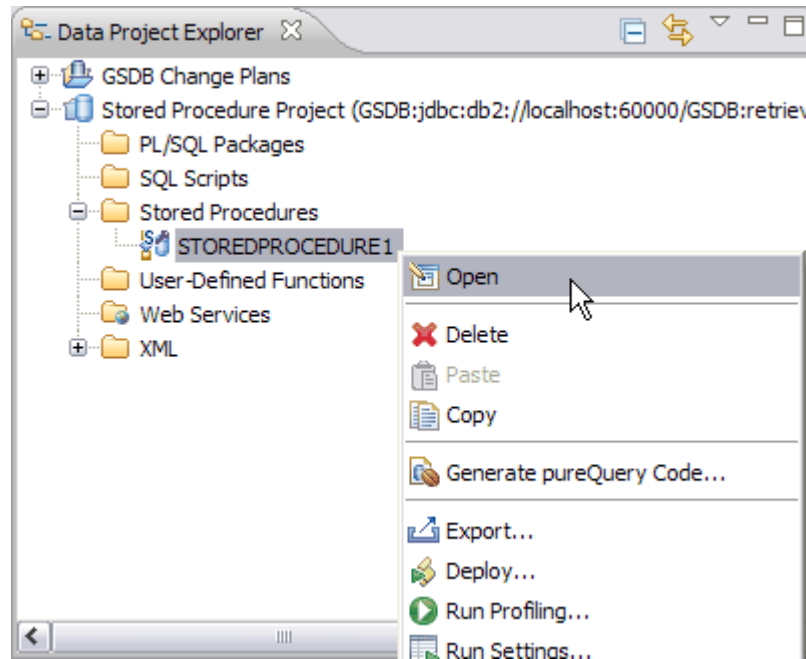
Figure 8.17 – Open an existing stored procedure in Routine Editor

The *Routine Editor* provides syntax checking, and context sensitive semantic help that is
similar to the one in *SQL Editor* that was discussed in *Chapter 5*. In addition, the *Routine
Editor* provides few useful options through a task bar on the top right corner of the editor,
for deploying, running, and debugging a stored procedure or to edit the routine template
preferences. This is shown in *Figure 8.18*.
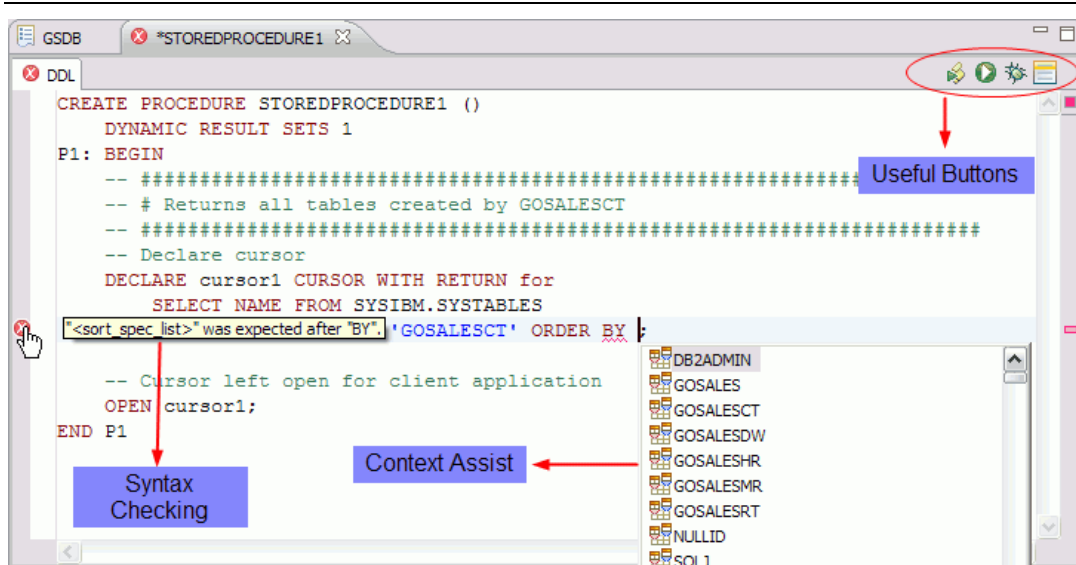
Figure 8.18 – Routine Editor's layout

To edit *STOREDPROCEDURE1* using Data Studio:

1. Open the stored procedure *STOREDPROCEDURE1* in *Routine Editor* as shown in *Figure 8.17*.

2. Modify the stored procedure as shown in *Figure 8.19* by following the code snippet in *Listing 8.1*. The updated stored procedure has an input and output parameter, and an *IF* statement.
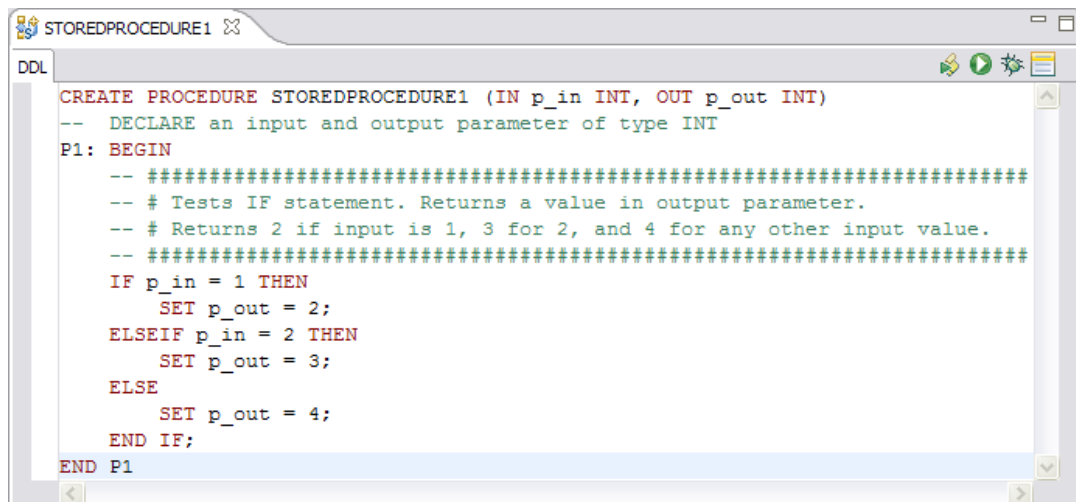
Figure 8.19 – Edit the stored procedure

3. In the *IBM SQL and Routine Development* perspective tool bar, click on the *Save (Ctrl+S)* button with this icon: 🖫 . Alternatively, select *File -> Save* option from the main menu.

8.3.7 Deploy the stored procedure for debugging

Data Studio includes an integrated stored procedure debugger which steps you through your stored procedure code to resolve problems. Before you can debug a stored procedure, you must deploy it for debugging. To deploy a stored procedure for debugging from Data Studio:

1. Follow the steps to deploy a stored procedure as described in *8.3.3* as shown in *Figure 8.11* to *Figure 8.12*. However, instead of selecting *Finish*, select *Next*.

2. In the *Routine Options* window, check the box to *Enable debugging* as shown in *Figure 8.20*, and select *Finish*.
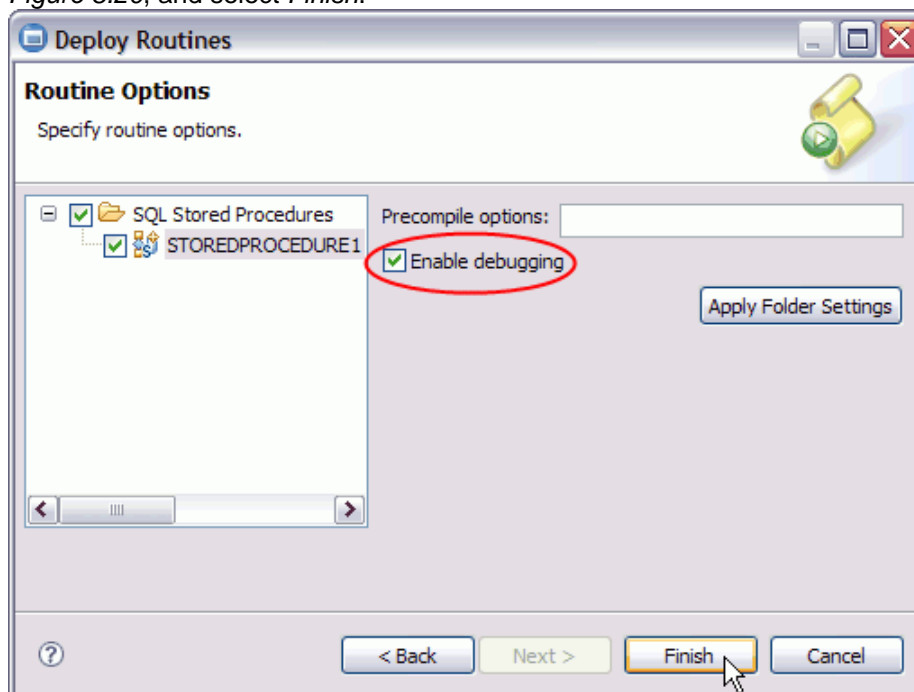
Figure 8.20 – Enable debugging

3. As before, you can use the *SQL Results* view to verify the deployment result.

8.3.8 Run the stored procedure in debug mode

To run the stored procedure in debug mode using Data Studio:

1. In the *Data Project Explorer*, navigate to the *Stored Procedures* folder, right-click *STOREDPROCEDURE1*, and select *Debug* as shown in *Figure 8.21*.

Note:

If you have not deployed the stored procedure for debugging as described in the previous section, the *Debug* option in the menu will be grayed out. Go back and deploy the stored procedure with the *Enable debugging* option checked.
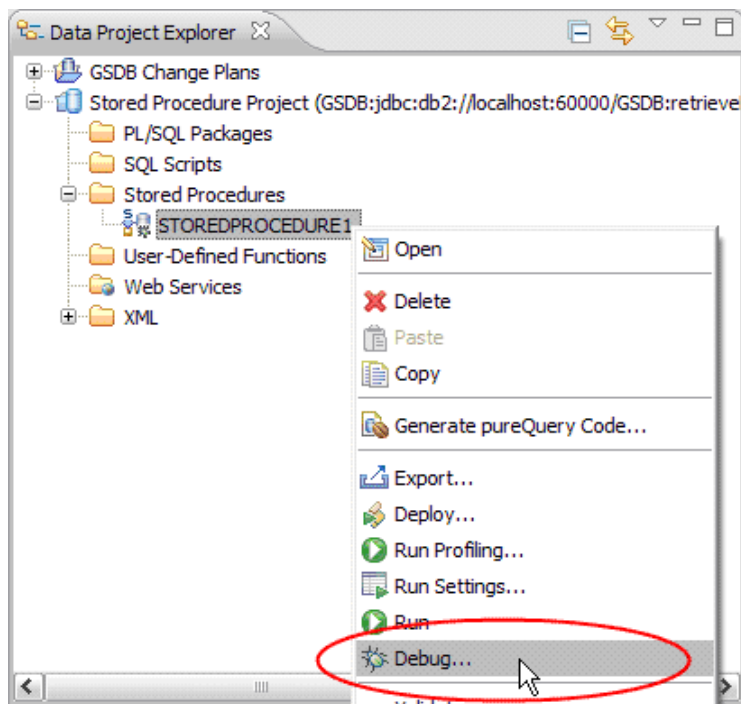
Figure 8.21 – Debug a stored procedure

2. A dialog window appears that lets you specify the initial value of the input parameter. Double-click the cell in *Value* column of *P_IN* parameter, and enter a numerical value *1* as shown in *Figure 8.22*.

3. The button *Set to Null* can be used if you want to set NULL as the input value. The buttons *Save Values* and *Load Values* let you save the input value(s) in XML format and load them for future executions of the stored procedure. The checkbox *Remember my values* lets you save these input values in memory in the current session. Accept all defaults and select *OK*.
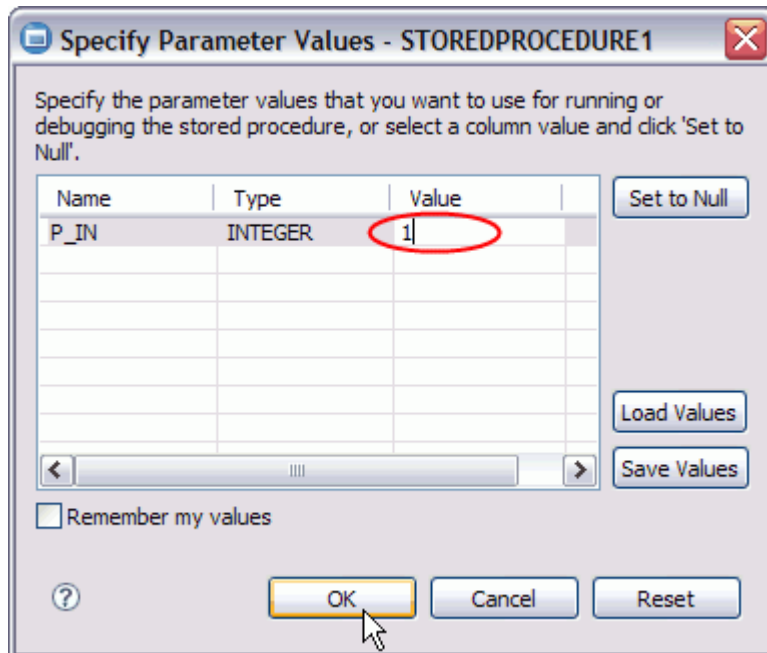
Figure 8.22 – Specify any input parameter values for the stored procedure

4. Eclipse has a standard *Debug Perspective* that is the default for debugging programs. A new window will appear asking you to confirm the perspective switch. In the *Confirm Perspective Switch* window, select *Yes* as shown in *Figure 8.23* to switch from the *IBM SQL and Routine Development* perspective to the *Debug* perspective.
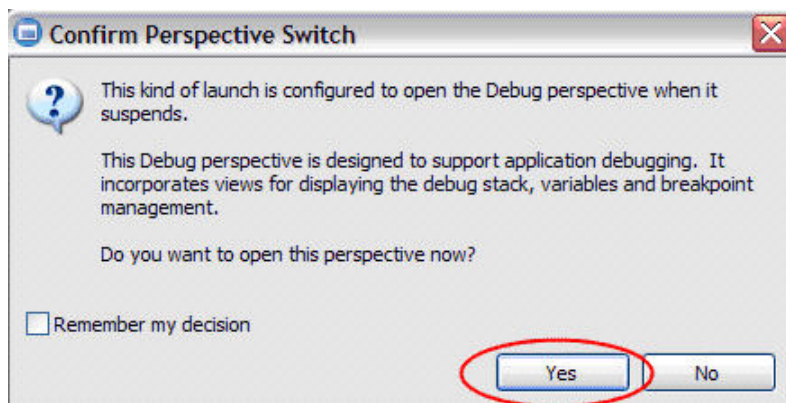
Figure 8.23 – Confirm perspective switch

5. A debug editor similar to the Routine Editor shows up with the debugger positioned on the first line of the stored procedure, which is the *CREATE PROCEDURE* statement. The current line where the debugger is positioned is always highlighted and a small arrow will be shown on the left margin. This is shown in *Figure 8.24*.

```
GOSALESCT.STOREDPROCEDURE1.2.dbg

CREATE PROCEDURE STOREDPROCEDURE1 (IN p_in INT, OUT p_out INT)
--   DECLARE an input and output parameter of type INT
P1: BEGIN
    -- #####################################################################
    -- # Tests IF statement. Returns a value in output parameter.
    -- # Returns 2 if input is 1, 3 for 2, and 4 for any other input value.
    -- #####################################################################
    IF p_in = 1 THEN
        SET p_out = 2;
    ELSEIF p_in = 2 THEN
        SET p_out = 3;
    ELSE
        SET p_out = 4;
    END IF;
END P1
```
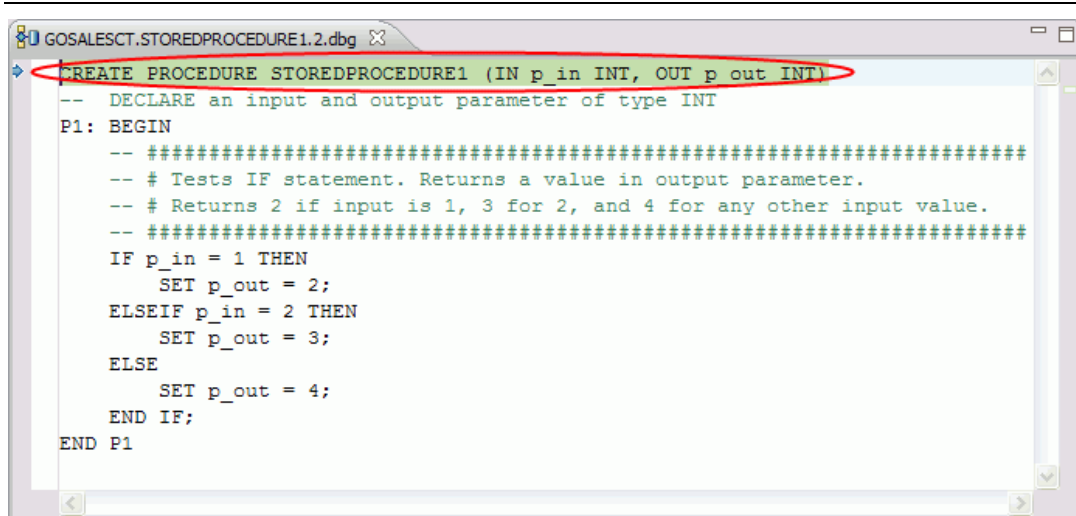
Figure 8.24 – Debugger positioned on the first line of the stored procedure

6. Set break points: In the Debug perspective there is a Debug task bar, as shown in *Figure 8.25.*

Figure 8.25 – Debug task bar

The arrow icons on the Debug task bar provide the *Step Into*, *Step Over*, and *Step Out* features while debugging a program (in this case, a stored procedure):

- The *Step Into* arrow ( ) positions you inside a condition, loop, or other similar feature.

- The *Step Over* arrow ( ) positions you after a condition, loop, or other similar feature.

- The *Step Return* arrow ( ) positions you outside of a condition, loop, or other similar feature.

While in the Debug perspective, in the debug editor for STOREDPROCEDURE1, double-click on the left vertical margin on the **IF**, **ELSEIF**, **ELSE,** and **END IF** statement code lines to set breakpoints as shown by the circles in the left margin in *Figure 8.26*.
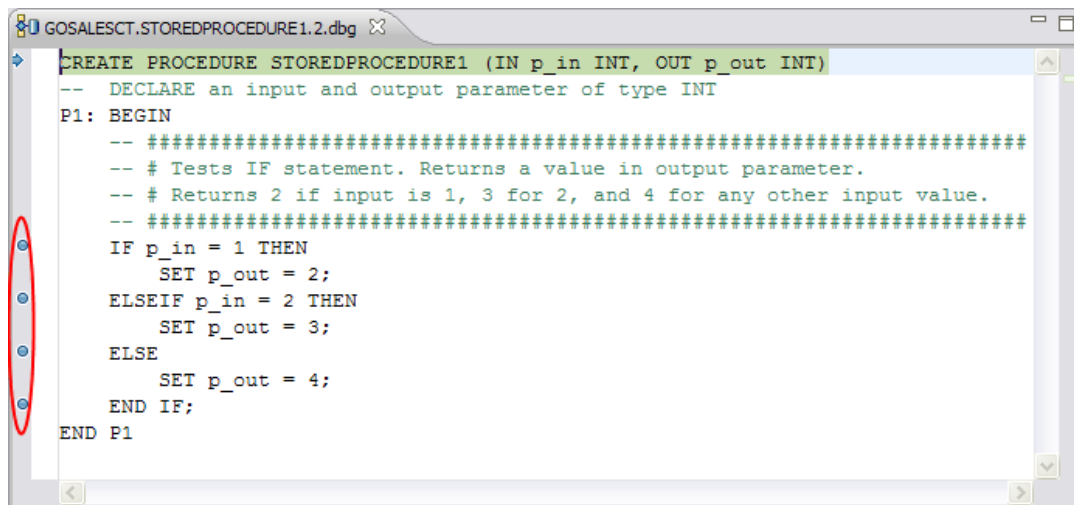
Figure 8.26 – Set breakpoints in left margin of the editor

5. Select *Step Over* ( ) to position the debugger on the first statement of the stored procedure body, which is the **IF** statement.

6. Change variable value: The *Variables* view in the Debug perspective lets you change the value of your input parameters, monitor the values of output parameters, observe and change the values of any local variables of the stored procedure, etc. For this example, even though we initiated the stored procedure execution with an input value 1, let's change it to 2 while in debug mode. To do this, in the *Variables* view for the parameter **p_in**, left-click the value 1 and enter value 2 as shown in *Figure 8.27*.
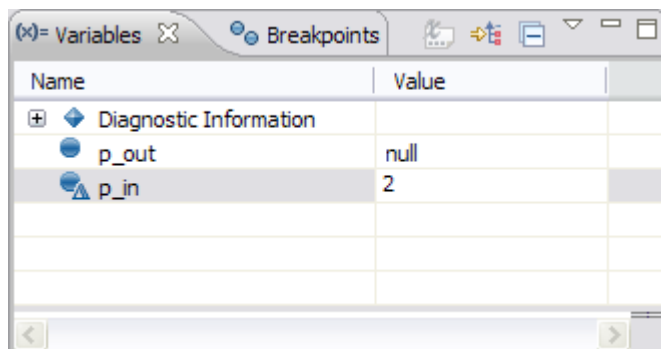
Figure 8.27–Change the value of input parameter in debug mode

7. Resume the debugger: From the Debug task bar, select the *Resume* button ( ). This will position the debugger on the **ELSEIF** statement in the stored procedure as shown in *Figure 8.28*. If there are breakpoints, the resume will always progress to the next breakpoint and stay there for user's next action. As mentioned before, in the debug editor, the highlighted line and the arrow in the left margin indicate the current line of code being debugged.

```
GOSALESCT.STOREDPROCEDURE1.2.dbg

CREATE PROCEDURE STOREDPROCEDURE1 (IN p_in INT, OUT p_out INT)
--   DECLARE an input and output parameter of type INT
P1: BEGIN
    -- ##############################################################################
    -- # Tests IF statement. Returns a value in output parameter.
    -- # Returns 2 if input is 1, 3 for 2, and 4 for any other input value.
    -- ##############################################################################
    IF p_in = 1 THEN
        SET p_out = 2;
    ELSEIF p_in = 2 THEN
        SET p_out = 3;
    ELSE
        SET p_out = 4;
    END IF;
END P1
```
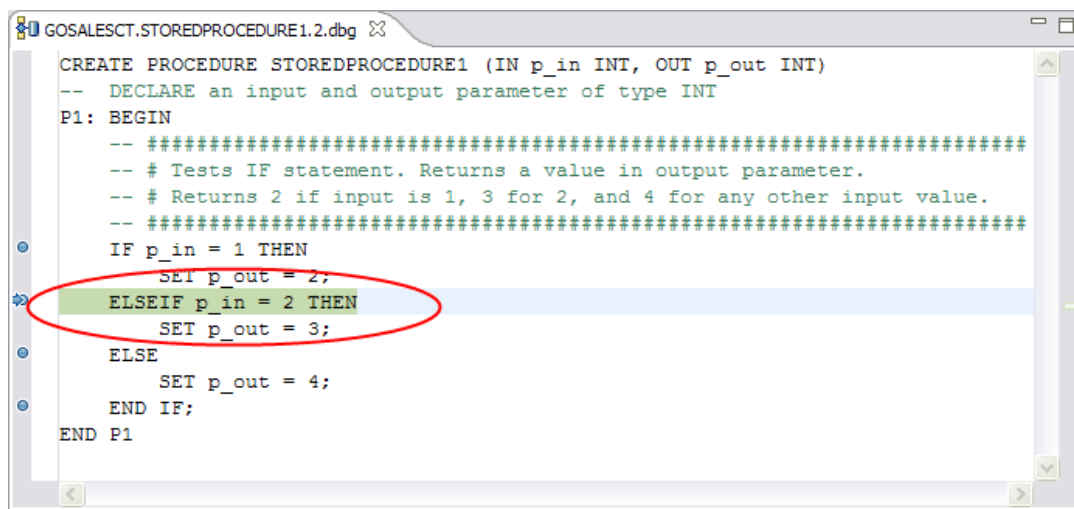
Figure 8.28 – Resume will position the debugger on the next break point

8. From the debug task bar, select Step Into icon (   ); this will step you into a condition or loop. In STOREDPROCEDURE1 debug editor view, the current line will be the **SET** statement in the **ELSEIF** condition as shown in *Figure 8.29*.


```
GOSALESCT.STOREDPROCEDURE1.2.dbg

CREATE PROCEDURE STOREDPROCEDURE1 (IN p_in INT, OUT p_out INT)
--   DECLARE an input and output parameter of type INT
P1: BEGIN
    -- ##############################################################################
    -- # Tests IF statement. Returns a value in output parameter.
    -- # Returns 2 if input is 1, 3 for 2, and 4 for any other input value.
    -- ##############################################################################
    IF p_in = 1 THEN
        SET p_out = 2;
    ELSEIF p_in = 2 THEN
        SET p_out = 3;
    ELSE
        SET p_out = 4;
    END IF;
END P1
```
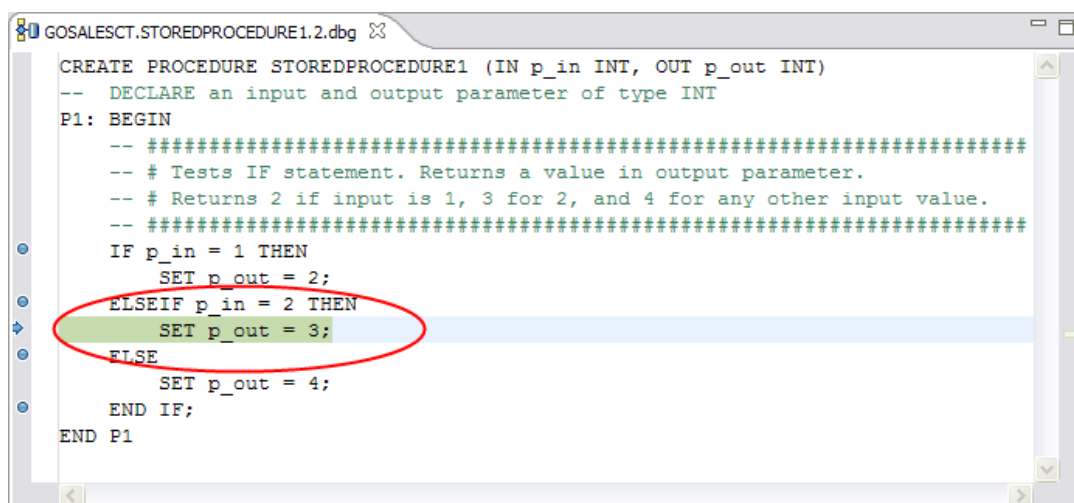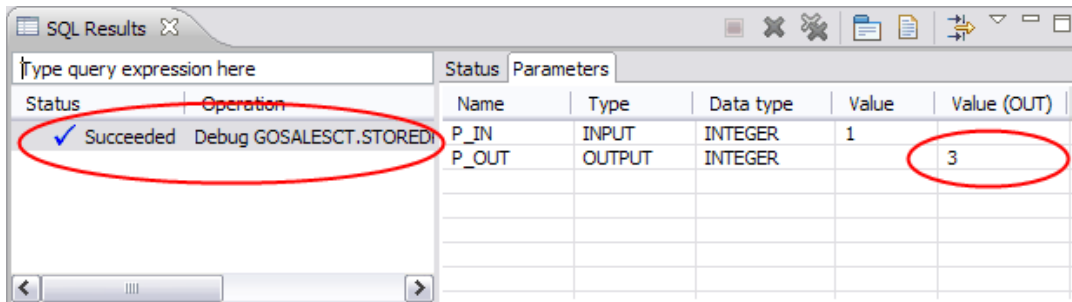
Figure 8.29 – Step into the logic

9. Resume the debugger: From the debug task bar, select *Resume* icon (   ) to finish running the stored procedure.

10. View the results: The *Debug* perspective provides the same *SQL Results* view as in the *IBM SQL and Routine Development* perspective so that you can see the status and results of running the stored procedure. The *Parameters* tab on the right in the *SQL Results* view will show your stored procedure's input and output parameters. This is shown in *Figure 8.30*. The value of **p_in** is 1, since that is

the value with which the stored procedure execution was triggered (even though you changed it to 2 while in the debug mode before executing the *IF* condition). The value of `p_out` is 3.

Figure 8.30 – View the debug results

8.4 Exercises

Now that you have gone through the process of creating, deploying, testing, debugging and running a stored procedure, it is time for you to test this yourself by creating the following procedure. Note the procedure has one intentional bug for you to discover. The output of the procedure should be 2, 3 or 4.

```
CREATE PROCEDURE SP1 (IN p_in INT, OUT p_out INT)
P1: BEGIN
     IF p_in = 1 THEN
          SET p_in = 2;
     ELSEIF p_in = 2 THEN
          SET p_out = 3;
     ELSE
          SET p_out 4;
     END IF;
END P1
```

8.5 Summary

In this chapter, you learned the value of using stored procedures to improve performance of SQL access by being able to process a set of SQL on the database server rather than sending each request over the wire separately. In addition, by encapsulating the database logic, those stored procedures can be called and used by multiple applications.

You also learned about the typical process for developing, deploying, and debugging stored procedures. Using a simple stored procedure, you learned that stored procedures are stored in the *Stored Procedures* folder of a *Data Development* project, and you also learned how to edit an existing stored procedure using the *Routine editor*.

Before a stored procedure can be run against the database, it must be *deployed*, which means the source code is compiled on a connected DB2 server. After being deployed to the target schema, the stored procedure will appear in the *Data Source Explorer* for that

database connection. To debug a stored procedure, you must first *deploy it for debug*, which will activate the debugger. A key fact to remember is that the integrated debugger is only activated when a stored procedure is specifically deployed for debug. Using the *Debug Perspective*, you learned how to set breakpoints and how to resume running a stored procedure after reaching a breakpoint. You also learned how to change the value of a variable using the *Variables* view of the Debug perspective.

8.6 Review questions

1. What is one likely reason that the Debug option would be inactivated (grayed out when you right click on the stored procedure in the Data Development project)?

2. If you don't specify a target schema, into which schema will new stored procedures be deployed?

3. Which view in IBM SQL and Routine Development perspective contains the status of the operation you performed?

4. Which Debug perspective view allows you to change a variable value?

5. Which Debug perspective view lists your breakpoints by line number?

6. Which procedure languages does Data Studio support when you connect to DB2 for Linux, UNIX, and Windows?

   A. Java, Cobol and PL/SQL

   B. Java, SQL and PL/SQL

   C. Java, C++ and SQL

   D. Cobol, SQL and PL/SQL

   E. None of the above

7. Deploying a stored procedure in Data Studio means which one of the following:

   A. Source code exists as a file on the DB2 server

   B. Source code exists as a file in your workspace

   C. Compiling the source code in your workspace

   D. Compiling the source code on the DB2 server

   E. All of the above

8. Which is the correct order to develop a stored procedure in Data Studio?

   A. Create a procedure, view the output or results, deploy a procedure, run a procedure, debug a procedure

   B. Create a procedure, debug a procedure, deploy a procedure, run a procedure, view the output or results,

    C. Create a procedure, deploy a procedure, run a procedure, view the output or results, debug a procedure

    D. Create a procedure, deploy a procedure, view the output or results, run a procedure, debug a procedure

    E. None of the above

9. What is the name of the view that shows the status of running your stored procedure in the IBM SQL and Routine Development perspective?

    A. SQL Results

    B. Data Source Explorer

    C. Data Project Explorer

    D. SQL Editor

    E. None of the above

10. Which of the following icons enables you to resume running the stored procedure after a breakpoint?

    A.

    B.

    C.

    D.

    E. None of the above

9

Chapter 9 – Developing user-defined functions

In this chapter, you will learn how to develop user-defined functions (UDFs) using Data Studio.

9.1 User-defined functions: The big picture

Like stored procedures, UDFs encapsulate reusable code. Because UDFs can be used in SQL statements, they let you extend the SQL language with your own logic. For example, you might want to create a function that encapsulates the logic to calculate a tax value in your country when a source value is given as input, or you can create a function that extracts information from an XML document and returns it in a tabular form that can be used to perform a JOIN with another table. UDFs provide a very flexible way to extend your application.

The supported languages for UDFs that are developed using Data Studio are: SQL, and PL/SQL. The DB2 server supports other languages for UDFs, such as Java and C++, but those are not supported in Data Studio.

Note:

Although Data Studio lets you create PL/SQL functions for DB2 projects, PL/SQL support is not available in DB2 Express-C, which is the edition of DB2 used for the examples in this book. If you wish to develop PL/SQL functions, you'll need to upgrade to a different edition of DB2 that contains PL/SQL support.

UDFs developed in Data Studio can have one of the following return types:

- **Scalar:** UDFs that accept one or more scalar values as input parameters and return a scalar value as result. Examples of such functions include the built-in *length*(), and *concat*() functions.

- **Table:** UDFs that accept individual scalar values as parameters and return a table to the SQL statement which references it. Table functions can be referenced in the *FROM* clause of a *SELECT* SQL statement.

Scalar functions are widely used in SQL statements to do processing of individual or aggregate values. UDFs that receive multiple values as input and return a scalar value are called **aggregate functions**.

Here is an example of using the built-in scalar function *concat*():

```
db2 => values CONCAT('Hello', ' World')
1
-----------
Hello World
1 record(s) selected.
```

You can use table functions in several different ways. You can use a table function to operate (with SQL language) on data that is not stored in a database table, or even to convert such data into a table. You can use them to read data from files, from the Web, or from Lotus Notes databases, and return a result table. The information resulting from a table function can be joined with information from other tables in the database, or from other table functions.

9.2 Creating a user-defined function

Data Studio supports template based user-defined function creation. There are many predefined routine templates that allow the user to create user-defined functions easily. A routine template is a predefined text with some standard code, comments, best practices, etc. that makes it easy for developers to start creating new user-defined functions.

To create a UDF in Data Studio, follow these steps:

1. Create a new data development project as you learned to do in *Chapter 8*. However, name the project '*UDF Project*'.

2. Right-click on the *User-Defined Functions* folder under *UDF Project*, and select *New -> User-Defined Function* as shown in *Figure 9.1*.
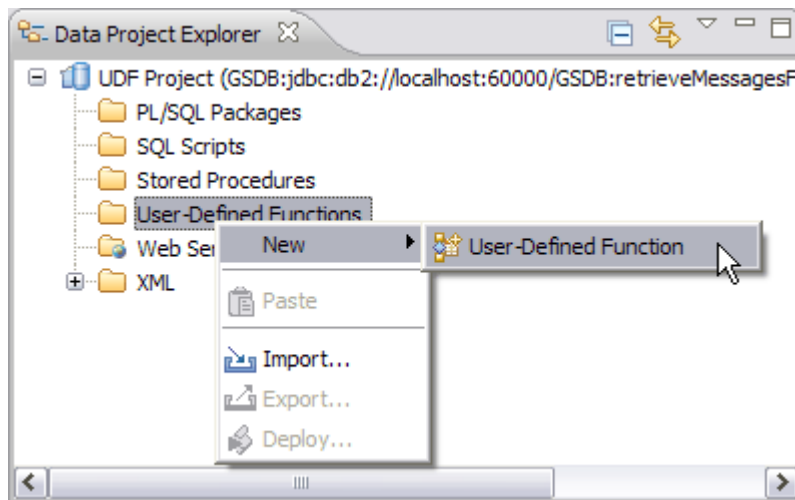
Figure 9.1 - Creating a new user-defined function

3. In this example, you will create a SQL user-defined function. Type *UDF1* in the *Name* field, and select *SQL* as the *Language* as shown in *Figure 9.2*.

4. Under the *Select a template* section, a list of available templates based on the selected language and server type are displayed. Select the template with the name *Deploy & Run: (Table) Return a result set*. The *Preview* section shows the details of the template through *Template Details* tab, and the actual predefined code through the *DDL* tab. This is also shown in *Figure 9.2*.
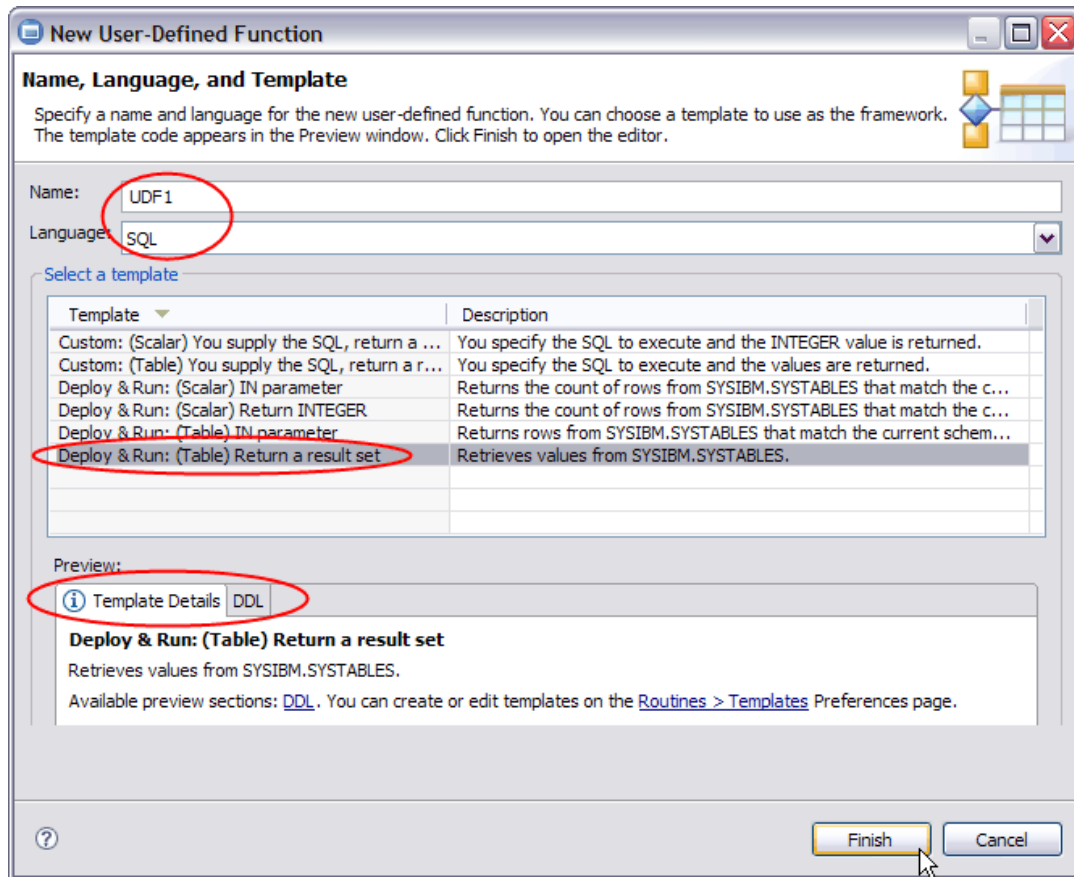
Figure 9.2 – Specify the UDF name, language, and template

5. Select *Finish* to open the *Routine Editor* with the contents of the template pre populated for further editing. The new user-defined function *UDF1* is also added to the *User-Defined Functions* folder of *UDF1 Project* in *Data Project Explorer* view, as shown in *Figure 9.3*. For this example, do not make any additional changes in the routine editor.
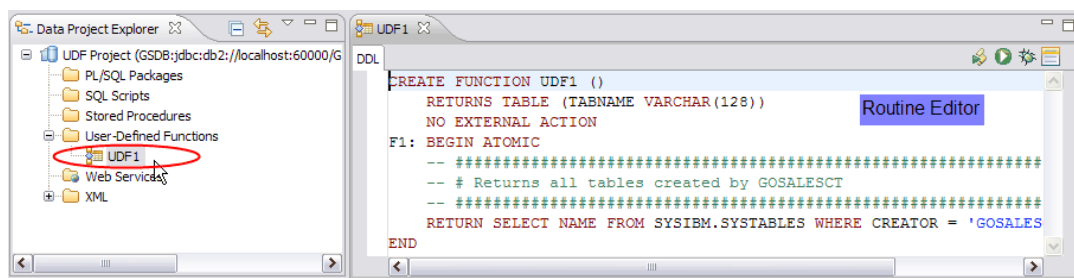
Figure 9.3 – View the User-Defined Functions folder and the Routine Editor

---

Note:

For more information about using Data Studio for template based routine development, refer to the developerWorks article entitled '*IBM Optim Development Studio: Routine development simplified*' at:
http://www.ibm.com/developerworks/data/library/techarticle/dm-1010devstudioroutines/index.html. This article uses an earlier version of Optim Development Studio product and stored procedures as an example, but the information is similar for Data Studio and UDFs.

9.3 Deploy the user-defined function

At this point in the UDF development process, the UDF source code exists as a file in your workspace. Before you can run your UDF against your database, you must deploy it to the database server. When deployed, the DB2 database server compiles the source code, and creates a new UDF object in the database with the compiled code. In case of any compile errors, the deployment will fail and an appropriate error will be sent back to the client.

To deploy the UDF using Data Studio:

1. Expand the *UDF Project* in the *Data Project Explorer* view, right-click *UDF1* from *User-Defined Functions* folder, and select *Deploy* as shown in *Figure 9.4*.
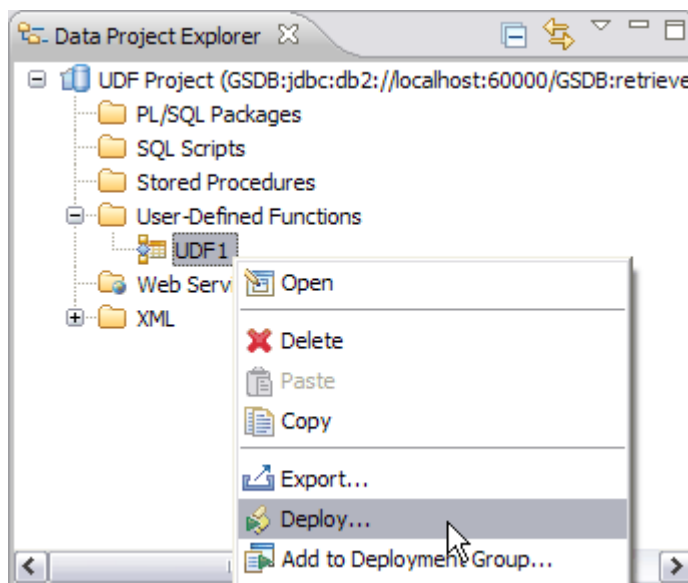
Figure 9.4 – Deploy the UDF

2. In the *Deploy Routines* window, make sure that the target schema is *GOSALESCT*. The *Target database* options allow the user to either deploy on the current database or on a different database. The *Duplicate handling* options can be used

to specify if Data Studio should first drop any duplicates or not. Accept all defaults, and select *Finish* as illustrated in *Figure 9.5*.
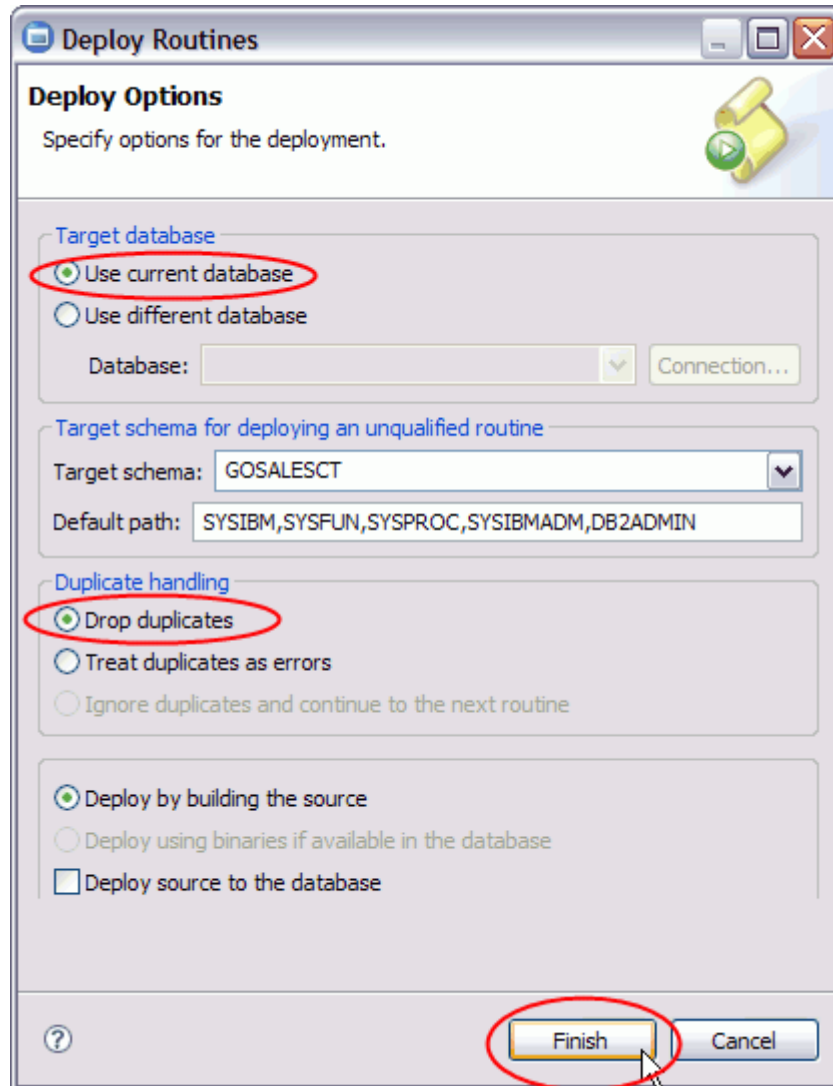
Figure 9.5 – Specify deployment options

3. Data Studio provides several views which provide quick access to informational output. Look at the entry for your *Deploy GOSALESCT.UDF1* operation in the *SQL Results* view. Wait until the operation completes, then verify that the deploy operation shows *Succeeded,* as shown in *Figure 9.6.* The *Status* tab on the right shows more detailed output.
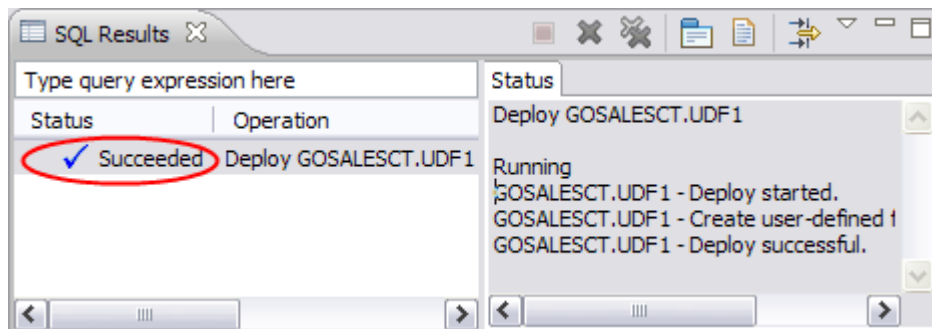
Figure 9.6 – View the deployment status

Note:

Data Studio supports debugging of non-inline scalar UDFs, or PL/SQL UDFs for DB2 for Linux, Unix, and Windows V9.7 and above. For more information on deploying for debugging, and debugging a routine, see *Chapter 8*.

---

4. When you successfully deploy a UDF to the DB2 database server from the *Data Project Explorer* view, this new UDF object will be reflected in the *User-Defined Functions* folder of the respective database in the *Data Source Explorer* as well. To verify this, expand *Database Connections* -> *GSDB* -> *Schemas* -> *GOSALESCT* -> *User-Defined Functions* folder in *Data Source Explorer* and observe the entry for *UDF1*. This is shown in *Figure 9.7*. If the new object is not yet visible, right click *User-Defined Functions* folder, and select *Refresh*.
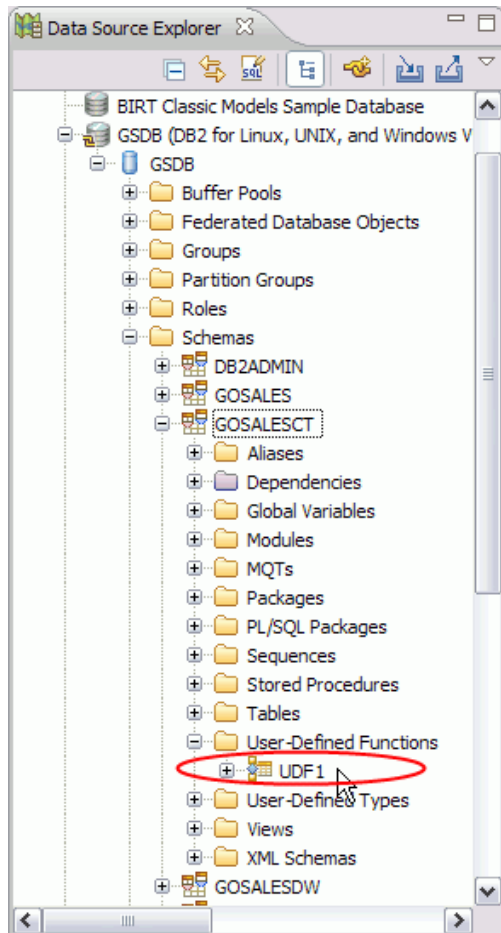
Figure 9.7 – The new UDF appears in the Data Source Explorer

9.4 Run the user-defined function

After you deploy your UDF to the database, you can run the UDF. From the *Data Project Explorer*, navigate to the *User-Defined functions* folder in *UDF Project*, right-click *UDF1*, and select *Run* as shown in *Figure 9.8*.
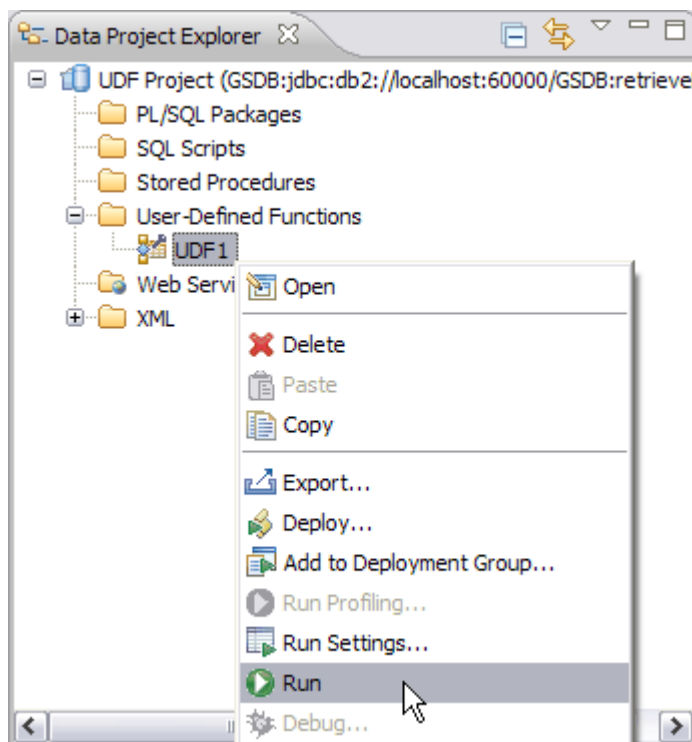
Figure 9.8 –Run the user-defined function

9.5 View the output

The *SQL Results* view shows the status of running your UDF. The *Status* column of this view indicates success or failure of the operation. The Operation column shows the command type or the actual command itself. The *Status* tab on the right shows detailed output. In this case, since the UDF returns a result set, its corresponding rows are shown in a separate tab *Result1*. This is illustrated in *Figure 9.9*.
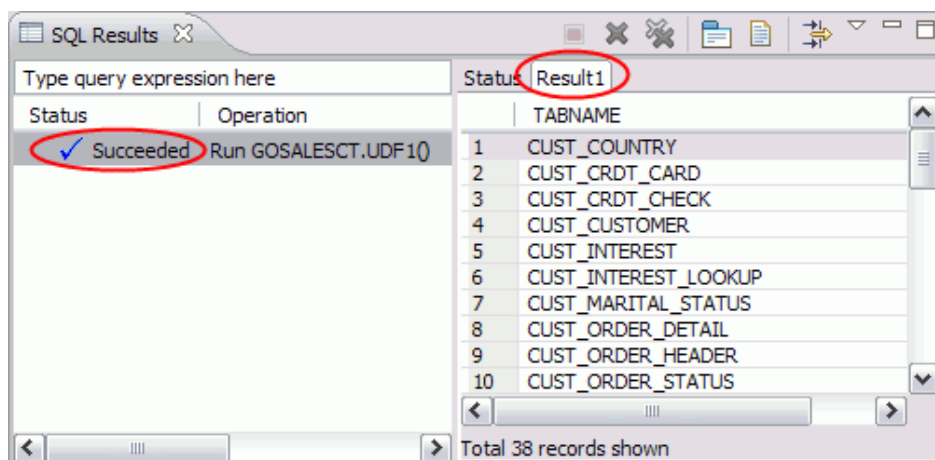
Figure 9.9 – View results of UDF execution

9.6 Edit the procedure

In Data Studio, a UDF can be viewed and edited in the *Routine Editor*. To open a UDF in the *Routine Editor*, double-click the UDF name or right-click and select *Open* as shown in *Figure 9.10*.
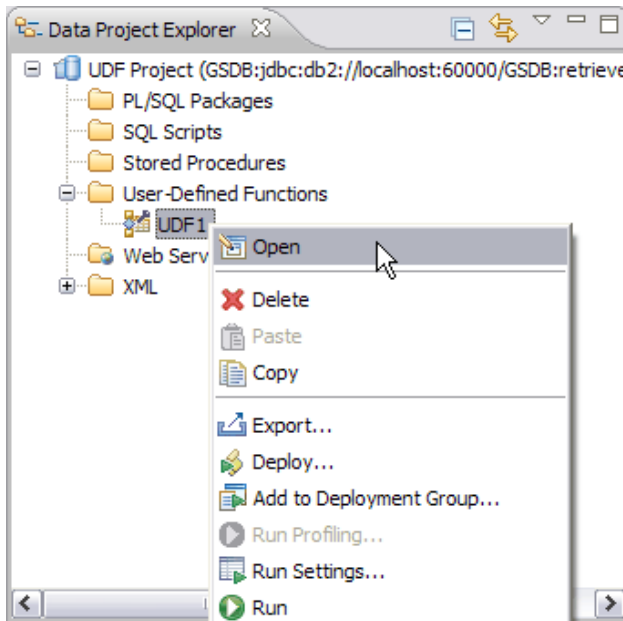
Figure 9.10 – Open an existing UDF in Routine Editor

The *Routine Editor* provides syntax checking, and context sensitive semantic help that is similar to the one in *SQL Editor* that was discussed in *Chapter 5*. In addition, the *Routine Editor* provides few useful options through a task bar on the top right corner of the editor, for deploying, and running a UDF or to edit the routine template preferences. This is shown in *Figure 9.11*.
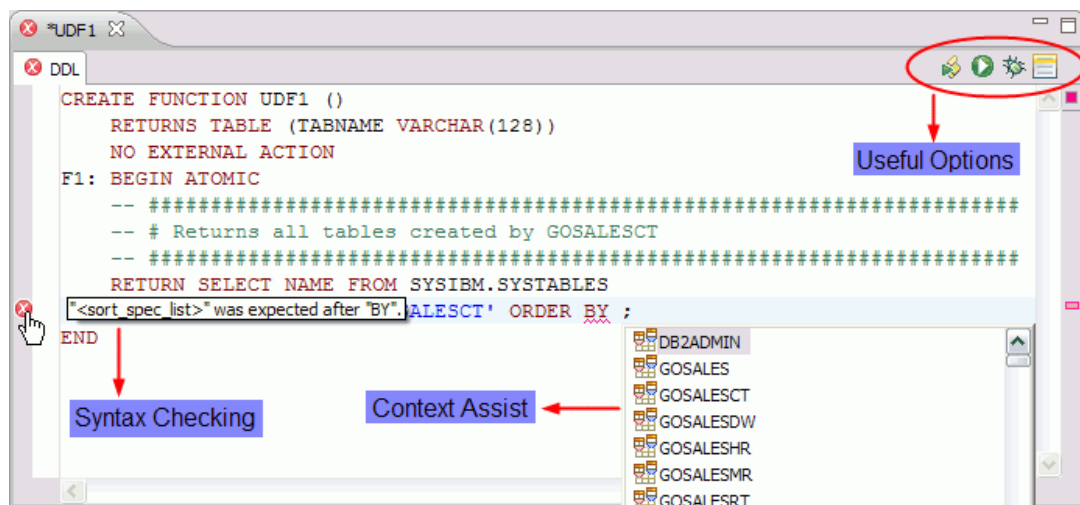
Figure 9.11 – Routine Editor's layout

To edit *UDF1* using Data Studio:

1. Open the user-defined function *UDF1* in *Routine Editor* as shown in *Figure 9.10*.

2. Modify the UDF as shown in *Figure 9.12*. The updated UDF is a scalar UDF that accepts one input parameter and returns the count of tables in *GOSALESCT* schema that match the input string pattern.
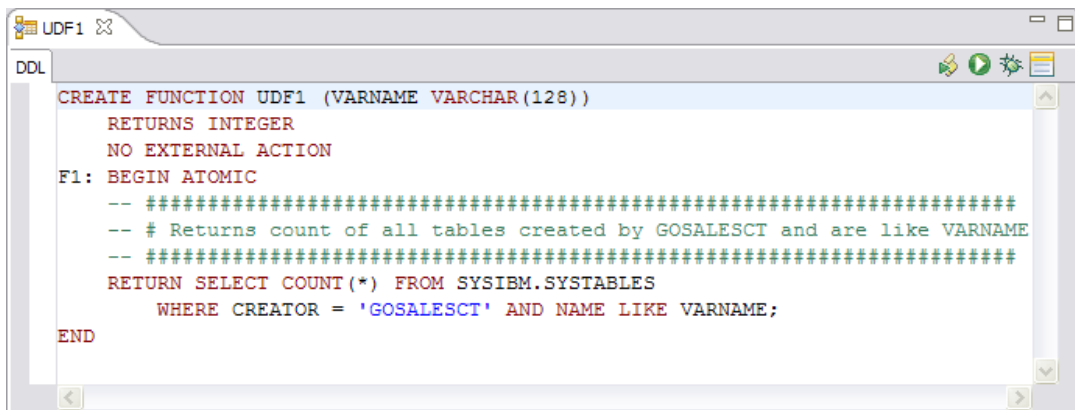
Figure 9.12 – Edit the user-defined function

3. In the *IBM SQL and Routine Development* perspective tool bar, click on the *Save (Ctrl+S)* button with this icon: 💾. Alternatively, select *File -> Save* option from the main menu.

4. At this point, the updated UDF is saved in your workspace. To reflect these changes in the database server, you need to deploy this again by following the steps in *section 9.3*.

5. To run the UDF again, follow the steps highlighted in *section 9.4*. However, since the updated UDF expects an input parameter, Data Studio will prompt you for entering a value for the input parameter as shown in *Figure 9.13*.

6. Enter the value shown in *Figure 9.13* and select *OK* to run the UDF.
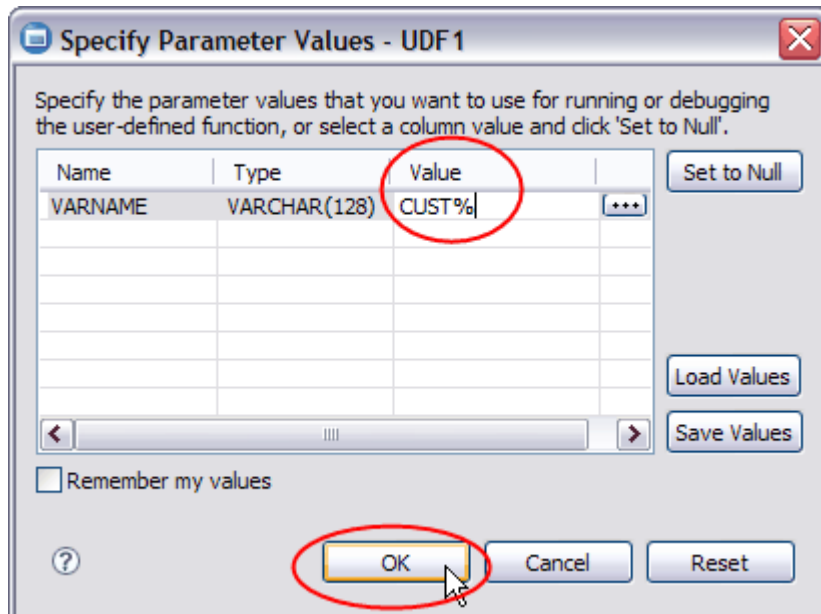
Figure 9.13 – Specify a value for the input parameter of the UDF

7. As explained in *section 9.5*, the status of running the UDF, and any returned value can be observed in *SQL Results* view. In this case, the returned value will be a scalar integer value which is the number of tables in *GOSALESCT* schema that start with the name *CUST*.

9.7 Summary

In this chapter, you learned the value of using user-defined functions to improve performance of SQL access by being able to process a set of SQL on the database server rather than sending each request over the wire separately. In addition, by encapsulating the database logic, those UDFs can be called and used by multiple applications. UDFs also provide a way to extend the SQL language with your own functions. You also learned the most important steps in the creation and maintenance of user-defined functions using Data Studio.

9.8 Exercise

As an exercise for this chapter, create a table UDF that returns the name and schema of all functions that have the qualifier equal to the value passed as a parameter to the function.

9.9 Review questions

1. Explain some of the advantages of using UDFs for database application development.

2. What is the difference between a scalar UDF and a table UDF?

3. Describe a scenario where you would use a UDF instead of a stored procedure.

4. Describe a scenario where you would use a UDF instead of plain SQL statements.

5. What is an aggregate UDF?

6. What languages are supported for user-defined function development in a Data Development project associated with a DB2 Linux, UNIX and Windows connection?

   A. SQL, PL/SQL

   B. SQL, OLE DB, PL/SQL

   C. SQL, Java, OLE DB, PL/SQL

   D. SQL, OLE DB

   E. None of the above

7. What result type or types are supported for SQL user-defined functions in Data Studio?

   A. scalar, list

   B. table, list

   C. scalar, table

   D. scalar, table, list

   E. None of the above

8. Which editor can be used to edit user-defined functions in Data Studio?

   A. SQL and XQuery Editor

   B. Data Object Editor

   C. Routine Editor

   D. Database Table Editor

   E. All of the above

9. What type of statement or statements can make up the body of a user-defined function?

   A. SQL statement

   B. SQL statement, SQL expression

   C. SQL expression

   D. SQL expression, regular expression

   E. All of the above

10. Where can you see the results of running a UDF?

   A. Console

   B. SQL editor

    C. SQL Results View

    D. Data Source Explorer

    E. None of the above

10

Chapter 10 – Developing Data Web Services

Data Web Services significantly eases the development, deployment, and management of Web services-based access to DB2 and Informix database servers. Data Web Service provides a tools and runtime framework that makes it easy to create Web services based on database operations, like SQL statements and stored procedure calls, using a simple drag and drop action. All Web service artifacts are generated by Data Studio. The generated Web services can be directly deployed to an application server and tested with the built-in Web Services Explorer.

In this chapter, after an overview of Data Web Services capabilities, you will learn a basic scenario for end-to-end development of a Data Web Service including:

How to configure WebSphere Application Server Community Edition (WAS CE) so you can deploy and test the Data Web Service you will create. You will need to have WAS CE installed before you can deploy the Data Web Service. The Data Web Services capability in Data Studio 3.1 currently works with WAS CE 2.1.x, which you can download from here: https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=wsced_archive&S_PKG=dl . You do not need to download anything other than the server itself.

- How to create a new Data Web Service in a Data Development project using existing SQL stored procedures and SQL scripts to provide the business logic.
- How to deploy the Web service to WAS CE
- How to use the Web Services Explorer to test the Data Web Service.

Appendix E contains information that can help you with different situations, such as options for consuming Web services using different clients, customizing the messages, and much more.

10.1 Data Web Services: The big picture

Web services, in general, are standards that allow applications to share information through services on the Web. There are a multitude of materials on the Web about Web services, and you can also refer to the ebook entitled *Getting Started with Web 2.0* for more information. In summary, however, Web services are designed to allow for communication between machines in a loosely coupled fashion. This can be accomplished by use of a Web Services Description Language (WSDL) XML document that provides the

description required by the invoker to call the service (where is the service, what binding to use, etc) and to understand the messages (in XML) returned by the service.

Data Web Services, in particular, refer to the ability to wrap Web services around the logic provide by the database. For example, you might already have a SQL script or stored procedure that provides business logic for returning the current price of a particular item in inventory from the database. Using Data Web Services, you are simply making it much easier for a Web application (or other client) to invoke that capability, perhaps even as simple as putting the HTTP request in a Web browser.

This approach to creating a Web service based on existing database operations/business logic is called "bottom up" as opposed to a to "top down" approach in which the Web services description is defined first and then logic is provided to map to that particular description.

Data Studio (and Optim Development Studio) supports the development and deployment of Data Web Services without you having to write a single line of code. *Figure 10.1* provides an overview of data Web services using Data Studio.
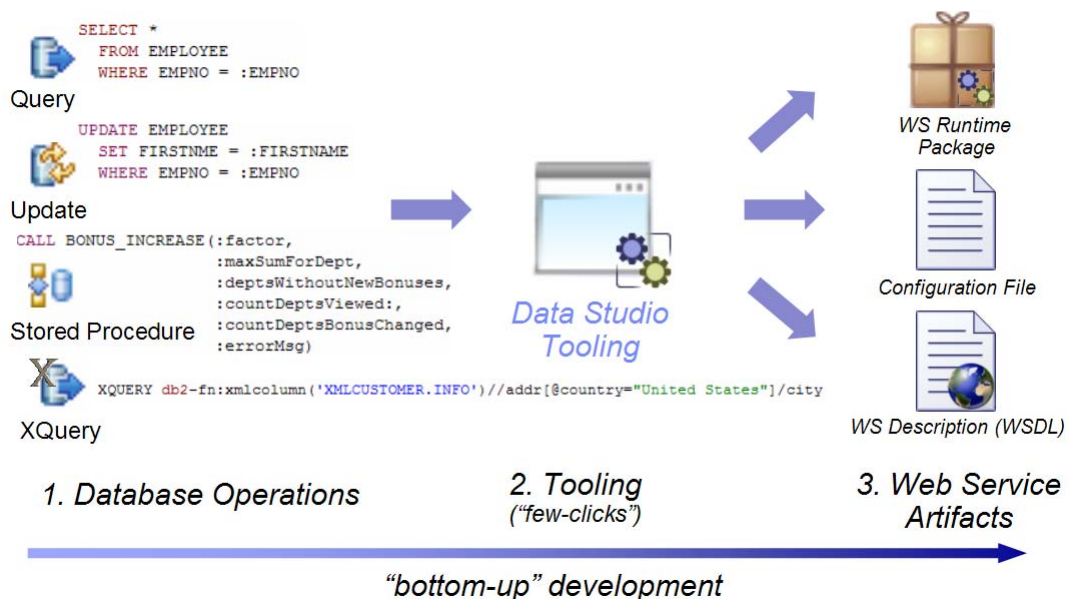
Figure 10.1 - Developing data Web services with Data Studio

On the left side of *Figure 10.1*, you can see different database operations. For example, there is a query to return all information about an employee when an employee number is provided. There is an update statement to update the first name of an employee based on an employee number; there is a stored procedure that does some bonus calculations, and there is an XQuery that is retrieving information from an XML document. Using Data Studio, these operations can be converted to data Web services without any coding on your part. A few clicks are all you need to have the Data Web Service created for you. On the right side of the figure, you can see that Data Studio automatically creates the artifacts

needed to deploy this Web service, including the WSDL document, and the JAVA EE runtime artifacts such as a configuration file and the runtime package.

10.1.1 Web services development cycle

Just like developing a JAVA EE application, the Data Web Service development cycle consists of the following steps, as shown in *Figure 10.2:*

1. Create the service
2. Deploy the service to a JAVA EE application server
3. Test the service.
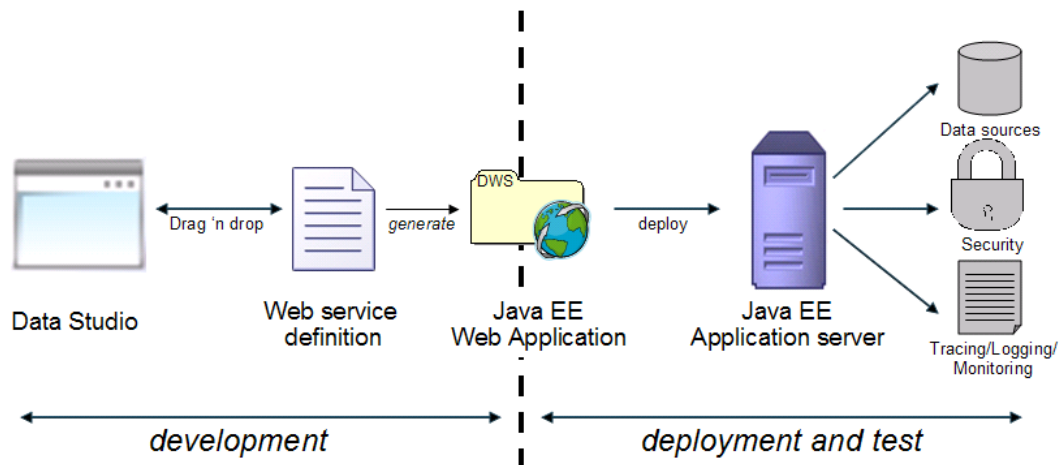
Figure 10.2 - Development and deployment of a data Web service

As shown in the figure, after you drag and drop an operation to create a Web service, Data Studio generates the corresponding Web service definitions that make a Data Web Service. The service runtime artifacts are packaged as a Java EE Web application. The Java EE application is ready to be deployed into a Java EE application server. You can apply additional settings for security, monitoring, logging, and more during the deployment phase.

10.1.2 Summary of Data Web Services capabilities in Data Studio

Here is a summary of Data Web services features provided by Data Studio:

- Using Data Web Services, you can take Data Manipulation Language (DML) statements, such as select, insert, update, delete, and XQuery, and stored procedures, and generate Web service operations by dragging and dropping those operations into a Web services.

- Data Web Services provide a full Web-service interface, which includes support for SOAP and HTTP(RPC)/REST-styled bindings.

- Web service artifacts like the WSDL and the runtime application are created automatically. There is no manual coding necessary.

- Data Web Services supports an integrated test environment that lets you deploy and test the generated services with a few clicks of the mouse.

- Data Web Services can apply server-side Extensible Style-sheet Language Transformation (XSLT) to generate different formats like HTML.

Note:

In the Web services world data is represented as XML. IBM Data Studio generates a default XML schema describing the input and output messages for each operation. You can manipulate the XML message format by assigning an XSL script, perhaps if your messages need to follow a particular industry standard format or if you want to generate an HTML document from the contents of the message. *Appendix E* shows you how to use XSL to transform the output of a Web service operation into an HTML document.

Data Web Services support these runtime environments: WAS CE version 2 all releases

- Apache Tomcat 5.5

- Apache Tomcat 6, all releases

- IBM WebSphere DataPower

- WAS 6, all releases

- WAS 7, all releases

10.2 Configure a WAS CE instance in Data Studio

Data Studio supports the direct deployment of a Web service to WebSphere Application Server Community Edition (WAS CE). The following steps show the setup required to hook up Data Studio with a WAS CE instance. This procedure assumes that you have already installed WAS CE on your system. See the ebook *Getting Started with WAS CE* or the [WebSphere Community Edition documentation](#) for more information about installing WAS CE.

1. Make sure you are in the Data perspective of Data Studio and then open the Server view by selecting *Window -> Show View -> Other…*

Expand *Server* and select *Servers* as shown in *Figure 10.3*.
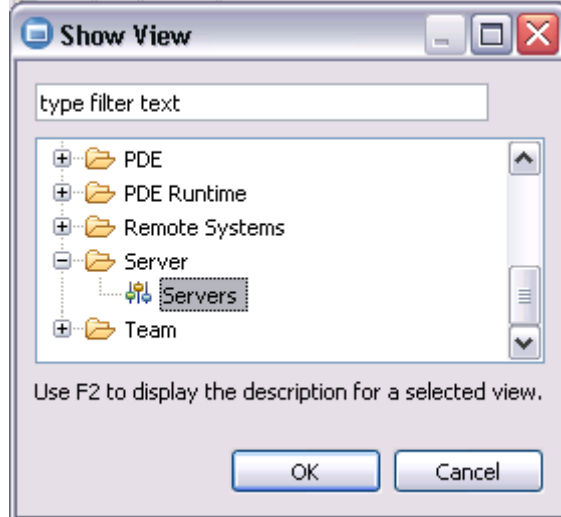
Figure 10.3 – Selecting the Server view in the Show View dialog

This opens a new tab called *Servers* in your workspace window.

2. Right-click inside the *Servers* tab and select *New -> Server* as shown *Figure 10.4.*
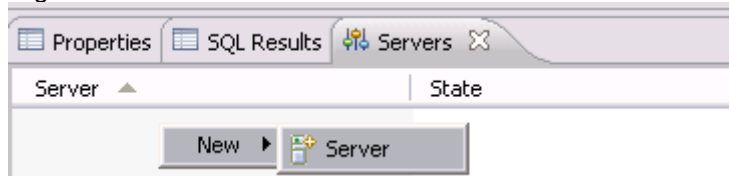
Figure 10.4 – Creating a new server

This will bring up the *New Server* dialog.

3. Accept all preset selections, as shown in *Figure 10.5*. The server's host name is set to localhost because WAS CE has been installed on your machine where Data

Studio is also installed. Select *Next*.
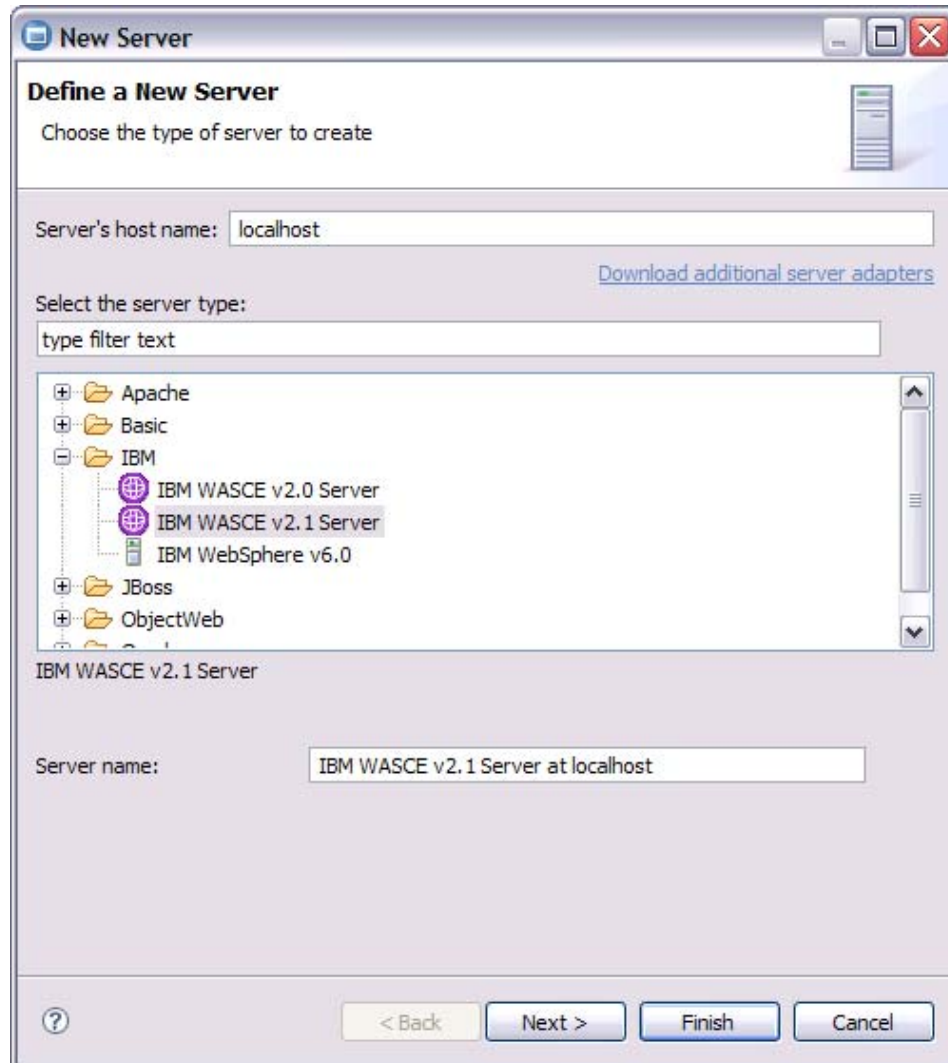
Figure 10.5 – The New Server dialog
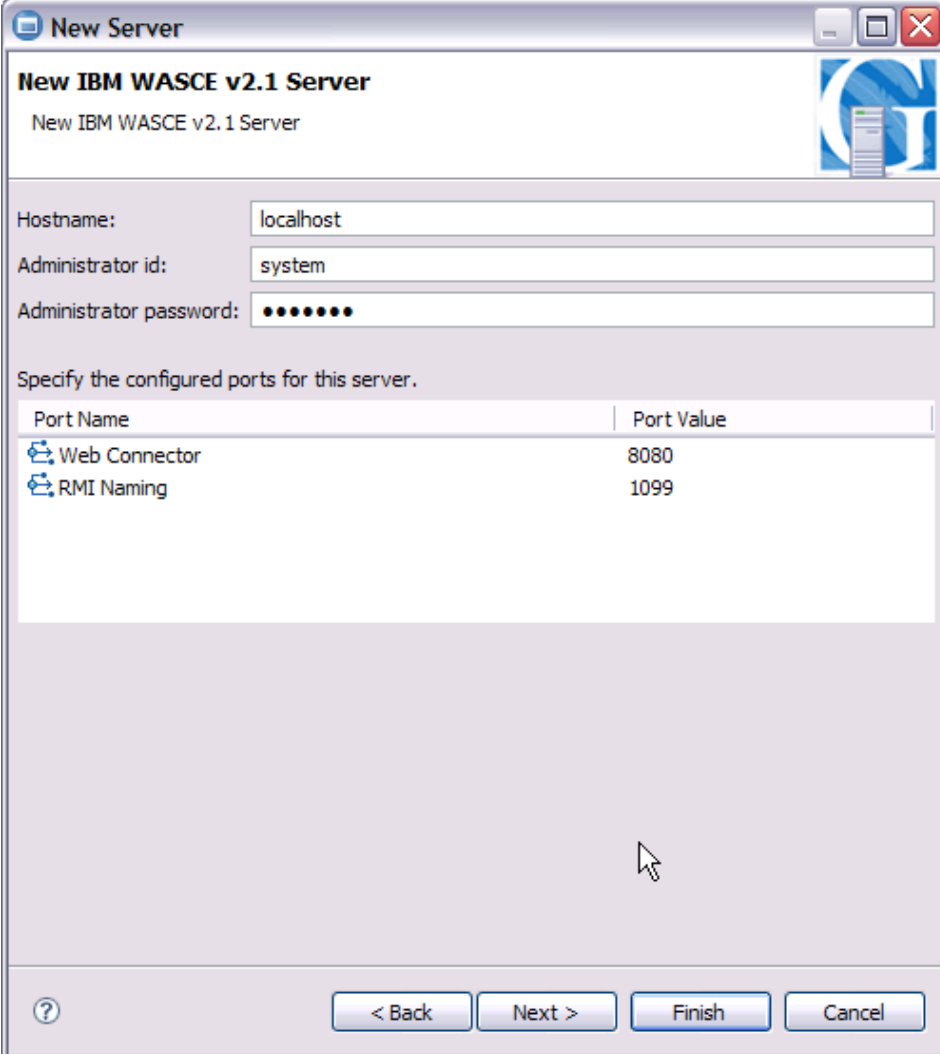
4. If you have not yet configured a WAS CE Runtime, you need to configure it in the next window – as shown in *Figure 10.6*. You are asked to provide a Java Runtime Environment (JRE™) and the absolute path to the WebSphere Application Server Community Edition installation. We select the default workbench JRE, which comes with Data Studio. You will receive a warning message because this version is a 1.6 JVM™ and WAS CE V2.1 is only certified for the 1.5 JVM, but you can ignore the warning since you will use WAS CE only for testing purposes and it

works fine with the 1.6 JVM.

Figure 10.6 – Configuring the Server runtime

The next window is already populated for you as shown in *Figure 10.7*.

Figure 10.7 – Configuring the server connectivity information

The Administrator ID and Administrator Password are the credentials of the WAS CE admin user. By default, the Administrator ID is "system" and the password is "manager". You might need the Administrator ID and Administrator Password at a later time when you try to log on to the Administration console from within or outside of Data Studio.

The Web Connector defines the TC/IP port for the HTTP protocol, which, by default, is **8080**.

The Remote Method Invocation (RMI) Naming defines the port that is used by Data Studio to perform administrative tasks at the application server. By default, this port

is **1099**. Both port values need to match according to the definitions in the WAS CE configuration.

5. Click *Finish*. You have successfully added the WebSphere Application Server Community Edition instance to your Data Studio, and the server definition also appears in the lower right corner of your Data Studio window as shown *Figure 10.8*.
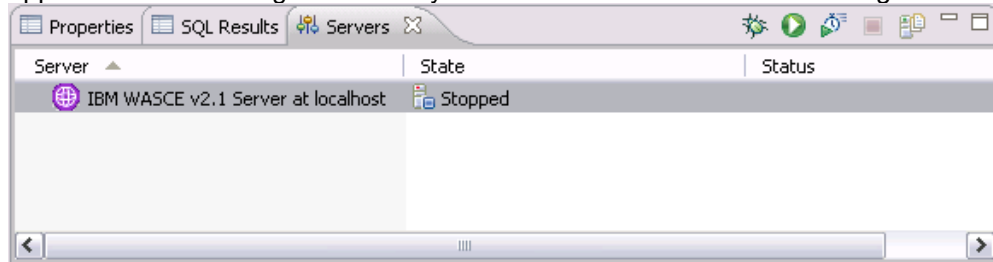
Figure 10.8 – The new server instance in the servers view

10.3 Create a Data Development project

After all the preparation is done you can start creating your first Web service. All you need is an active connection to your DB2 instance and a Data Development project based on that connection.

Using the instructions shown in *Chapter 2*, connect to the *GSDB* sample database and create a new Data Development project called **WebServices**. We will be using tables and stored procedures from the *GSDB* database to create a new Data Web Service.

Figure 10.9 shows the new Data Development project and the connection to the *GSDB* sample database.
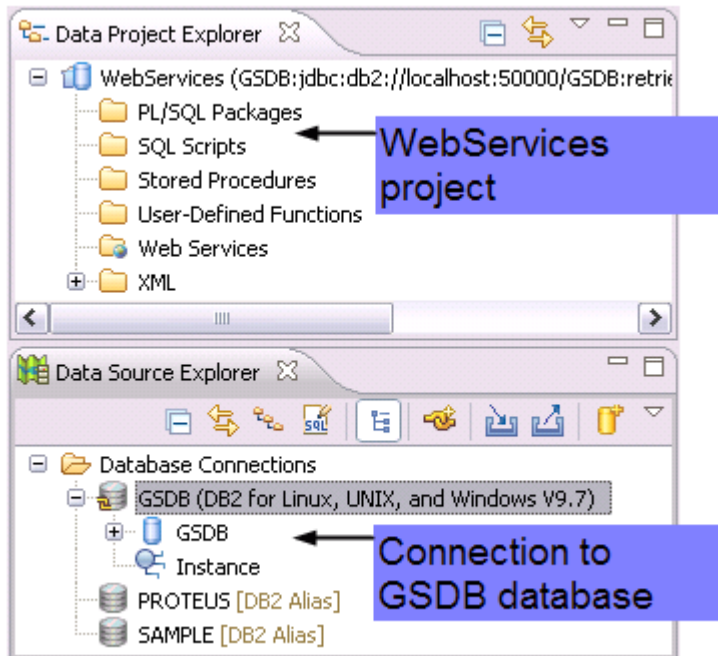
Figure 10.9 – The WebServices Data Development Project

10.4 Define SQL statements and stored procedures for Web service operations

Now it's time for you to decide what database data and logic should be exposed as Web service operations. Typically a Web service represents a set of operations with business logic which are grouped together because they are related – mainly from a business level perspective, but also for other reasons like security requirements, data structures, quality of service, and so on.

In the database world, stored procedures are the prime candidates to become Web service operations since they can contain a significant amount of business logic. However, an SQL statement can also be seen as a set of business logic – for example a SELECT statement that retrieves customer information.

The SQL statements and stored procedures used for this example are relatively simple.

10.4.1 Stored procedures used in the Web service

Although you usually create Web services using existing database operations, we need to create a couple of stored procedures here so we can show you how to use them in Data Web Services. If you want to follow along with the steps in the chapter, you will need to create the following procedures as well:

- GET_CUSTOMER_NAME, has input and output parameters
- PRODUCT_CATALOG returns a result set.

The logic is kept simple since we focus on how to add a stored procedure to a Web service rather than the stored procedure programming itself. We use SQL stored procedures here, but you can add procedures written in any language to a Web Service.

GET_CUSTOMER_NAME

This procedure returns customer information for a given customer ID. It is created under the *GOSALESCT* schema. It has only input and output parameters. Using the information you learned in *Chapter 8*, create the following procedure (you can cut and paste the text below into the SQL procedure editor). Be sure to deploy it into the *GOSALESCT* schema.

```
CREATE PROCEDURE GOSALESCT.GET_CUSTOMER_NAME(
    IN  CUSTOMERID    INTEGER,
    OUT FIRST_NAME    VARCHAR(128),
    OUT LAST_NAME     VARCHAR(128),
    OUT PHONE_NUMBER VARCHAR(128))
SPECIFIC GOSALESCT.GET_CUSTOMER_NAME
BEGIN
    SELECT CUST_FIRST_NAME INTO FIRST_NAME FROM GOSALESCT.CUST_CUSTOMER
        WHERE CUST_CODE = CUSTOMERID;
    SELECT CUST_LAST_NAME INTO LAST_NAME FROM GOSALESCT.CUST_CUSTOMER
        WHERE CUST_CODE = CUSTOMERID;
    SELECT CUST_PHONE_NUMBER INTO PHONE_NUMBER FROM GOSALESCT.CUST_CUSTOMER
        WHERE CUST_CODE = CUSTOMERID;
END
```
Listing 10.1 – GET_CUSTOMER_NAME procedure

PRODUCT_CATALOG

This procedure is defined under the *GOSALES* schema. It returns a result set containing all products from the catalog for a given product type. Using the information you learned in *Chapter 8*, create the following procedure (you can cut and paste the text below into the SQL procedure editor). Be sure to deploy it into the *GOSALES* schema.

```
CREATE PROCEDURE GOSALES.PRODUCT_CATALOG (IN PRODUCT_TYPE VARCHAR(50))
    DYNAMIC RESULT SETS 1
SPECIFIC GOSALES.PRODUCT_CATALOG
------------------------------------------------------------------------
-- SQL Stored Procedure
------------------------------------------------------------------------
P1: BEGIN
        -- Declare cursor
        DECLARE CURSOR1 CURSOR WITH RETURN FOR
                SELECT P.PRODUCT_NUMBER, Q.PRODUCT_NAME,
                        Q.PRODUCT_DESCRIPTION,
                        P.PRODUCTION_COST, P.PRODUCT_IMAGE
              FROM GOSALES.PRODUCT AS P,
                    GOSALES.PRODUCT_NAME_LOOKUP AS Q,
                    GOSALES.PRODUCT_TYPE AS R
            WHERE P.PRODUCT_NUMBER = Q.PRODUCT_NUMBER
              AND Q.PRODUCT_LANGUAGE = 'EN'
              AND R.PRODUCT_TYPE_CODE = P.PRODUCT_TYPE_CODE
```

```
            AND R.PRODUCT_TYPE_EN = PRODUCT_TYPE;
    -- Cursor left open for client application
    OPEN CURSOR1;
END P1
```

Listing 10.2 – PRODUCT_CATALOG procedure

10.4.2 SQL statements used in the Web service

We use two SQL statements for our Web service:

- GetBestSellingProductsByMonth

- RankEmployee.

GetBestSellingProductsByMonth

The SQL statement shown in *Listing 10.3* returns the top 50 products by shipping numbers for the given month. Using the information in *Chapter 5*, create a new SQL script with the name *GetBestSellingProductsByMonth* and copy the below statement into that script.

```
SELECT PN.PRODUCT_NAME, PB.PRODUCT_BRAND_EN, SUM(IL.QUANTITY_SHIPPED) AS
NUMBERS_SHIPPED, PN.PRODUCT_DESCRIPTION
   FROM GOSALES.INVENTORY_LEVELS AS IL, GOSALES.PRODUCT AS P,
       GOSALES.PRODUCT_NAME_LOOKUP AS PN, GOSALES.PRODUCT_BRAND AS PB
   WHERE IL.PRODUCT_NUMBER = PN.PRODUCT_NUMBER
     AND IL.PRODUCT_NUMBER = P.PRODUCT_NUMBER
     AND P.PRODUCT_BRAND_CODE = PB.PRODUCT_BRAND_CODE
     AND IL.INVENTORY_MONTH=:MONTH
     AND PN.PRODUCT_LANGUAGE = 'EN'
   GROUP BY PN.PRODUCT_NAME, IL.INVENTORY_MONTH,
           PB.PRODUCT_BRAND_EN, PN.PRODUCT_NAME, PN.PRODUCT_DESCRIPTION
   ORDER BY NUMBERS_SHIPPED DESC FETCH FIRST 50 ROWS ONLY
```
Listing 10.3 – SQL SELECT for the GetBestSellingProductsByMonth operation

---

Note:

You can define parameter markers in two ways – via the question mark notation (`a = ?`) or via a named marker using the colon (`a = :<name>`) notation. For Web services both notations work, but the named parameter markers are preferable since the names will be used for the input parameter names of the resulting Web service operation. If question mark notation is used the parameter names are just a sequence of p1, p2, …, pN. We use the named parameter marker notation in our statement for this reason.

RankEmployee

This statement inserts a new ranking record for a given employee number and an English ranking value term into the *RANKING_RESULTS* table of the *GOSALESHR* schema and returns the new row. Create a new SQL script named *RankEmployee,* and add the statement text as shown in *Listing 10.4*.

```
SELECT * FROM FINAL TABLE (
    INSERT INTO GOSALESHR.RANKING_RESULTS (
        RANKING_DATE, RANKING_YEAR, EMPLOYEE_CODE, RANKING_CODE)
        VALUES (CURRENT TIMESTAMP, YEAR(CURRENT TIMESTAMP),
                :EMPLOYEE_CODE,
                (SELECT RANKING_CODE FROM GOSALESHR.RANKING WHERE
UPPER(RANKING_DESCRIPTION_EN) = UPPER(LTRIM(RTRIM(CAST(:RANKING AS
VARCHAR(90))))))))))
```
Listing 10.4 – SQL INSERT for the RankEmpoyee operation

10.5 Create a new Web service in your Data Project Explorer

At this point you should have all the pieces together to start creating your Web service. The following steps show how to create the Web service.

1. If you're not there already, switch to the *Data* perspective. Right-click on the *Web Services* folder in your Data Development project and select *New Web Service…* as illustrated in *Figure 10.10.*
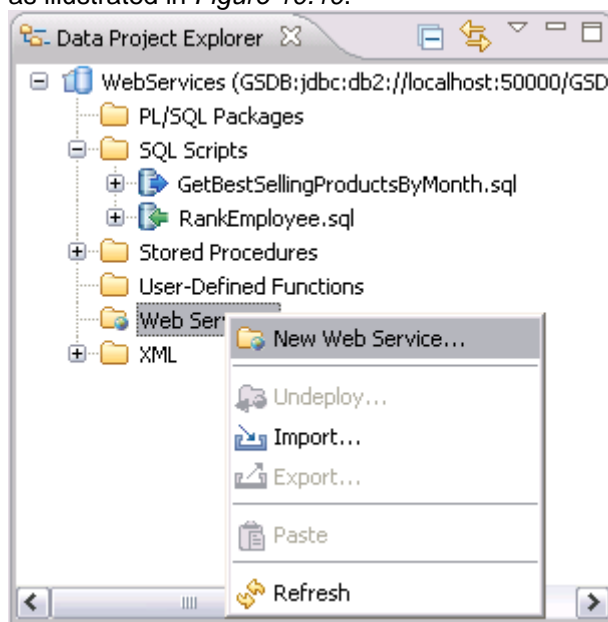
Figure 10.10 – Right click Web Services folder to create a new Web service

2. As shown in *Figure 10.11*, change the name of your service to *SimpleService* and use http://www.ibm.com/db2/onCampus as the Namespace URI. Note that a namespace URI is just a way to identify a collection of XML elements and attributes and does not need to point to an actual resource. Therefore it does not

need to be a URL.
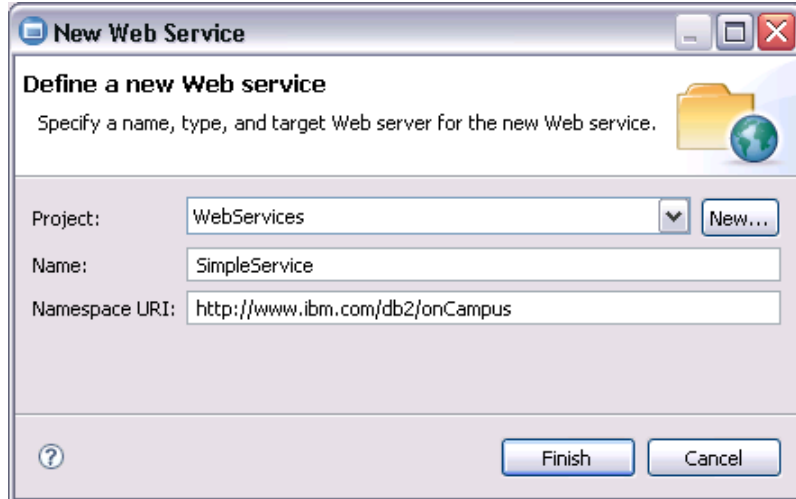
Figure 10.11 – Provide the basic Web service information

3.   Click on *Finish* to create the Web service. The Web Services folder now contains
     the new Web service as shown in *Figure 10.12.*
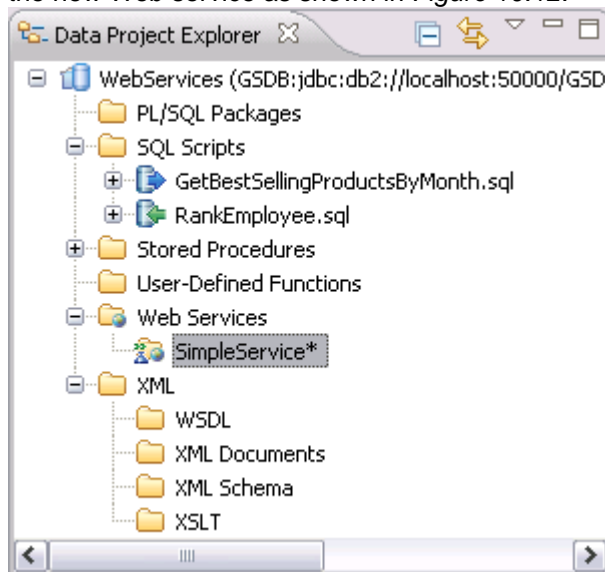
Figure 10.12 – The new Web service in the Data Development Project

The asterisk by the Web service name means that the Web service was not built and deployed since it was created or last changed.

10.6 Add SQL statements and stored procedures as Web Service operations

After creating the Web service you can add SQL statements and stored procedures as Web service operations, as follows:

1. Open the *GOSALES* and *GOSALESCT* schema in the Data Source Explorer. Select the *GET_CUSTOMER_NAME* procedure from the *GOSALESCT* schema and the *PRODUCT_CATALOG* procedure from the *GOSALES* schema and drag and drop them into your newly created *SimpleService* Web service, as shown in *Figure 6.13*.
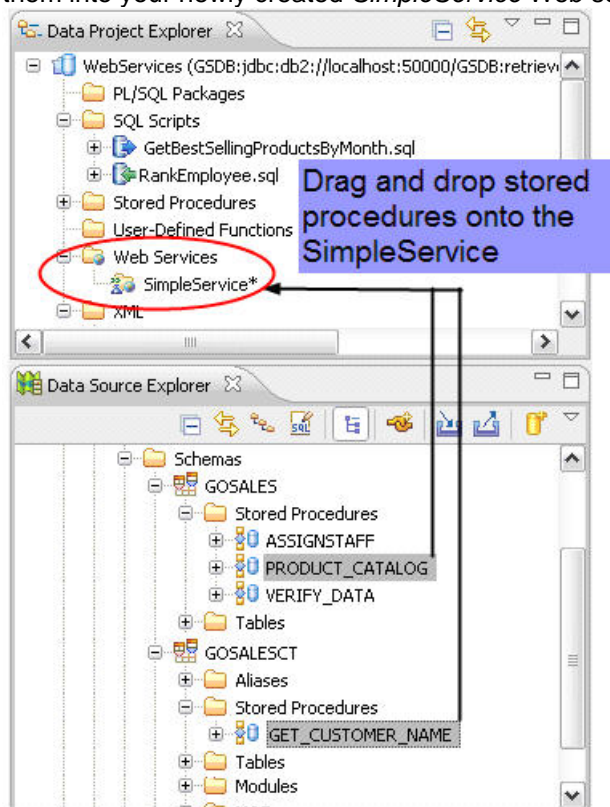
Figure 10.13 – **Drag and drop stored procedures into the Web service**

2. Select your SQL statements in the SQL Scripts folder and drag and drop them onto your *SimpleService* Web service as well, as shown in *Figure 10.14*.
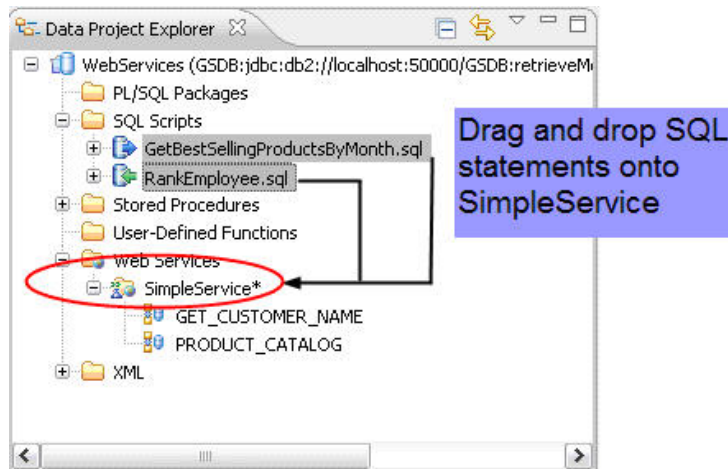
Figure 10.14 – Drag and Drop SQL statements into the Web service

Congratulations! You have finished your first Data Web Service. In your Data Explorer view, review the results. You should now see the two SQL scripts and the two SQL procedures under your Web service name, as shown in *Figure 10.15.*
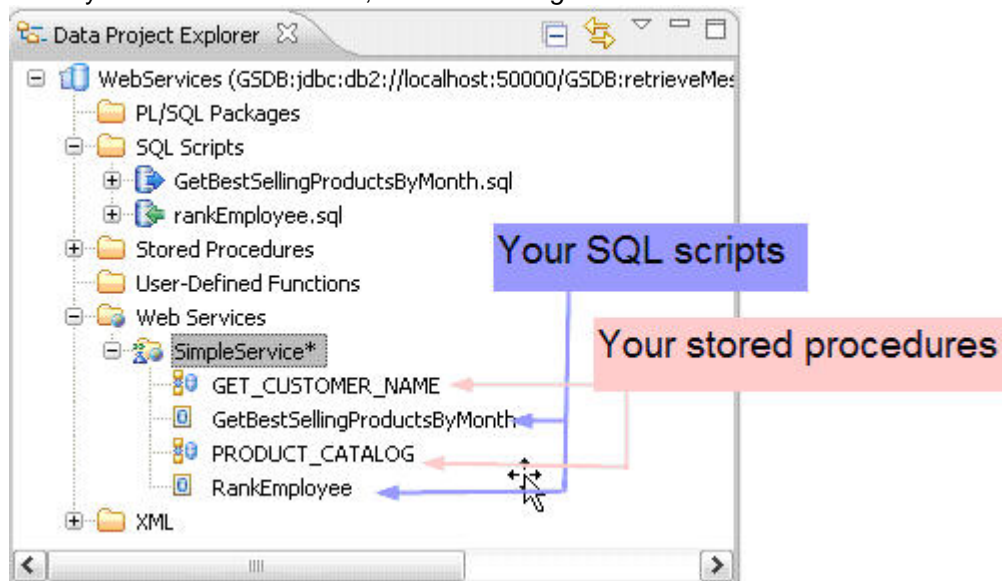
Figure 10.15 – The finished Web service

10.7 Deploy the Web Service

The *SimpleService* Web service can now be deployed on the prepared WAS CE instance, as follows:

1. Right-click the *SimpleService* Web service and select *Build and Deploy…* as shown in *Figure 10.16*. This brings up the *Deploy Web Service* dialog (*Figure*

10.17).
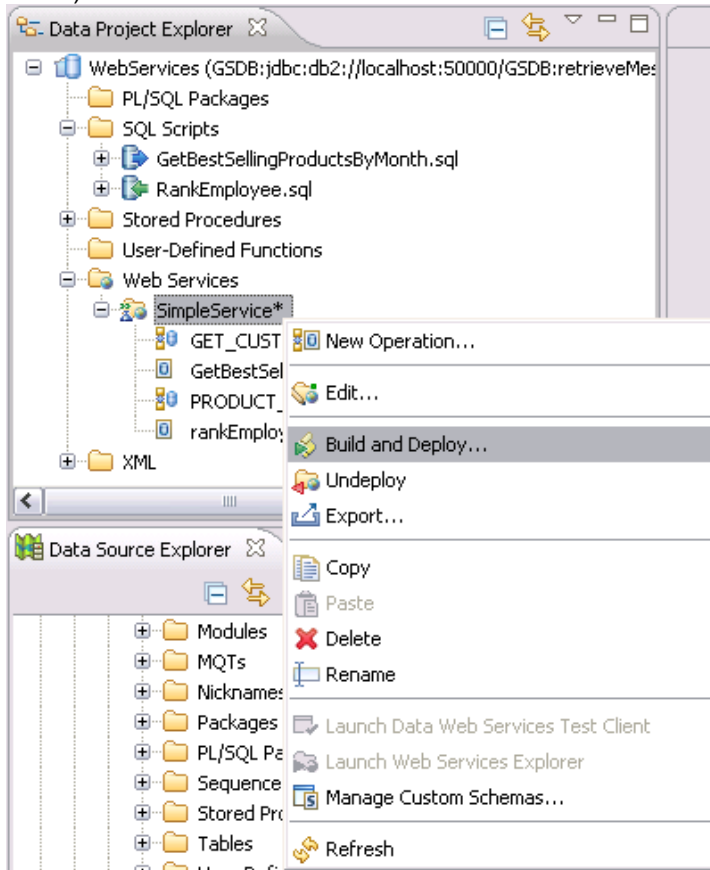
Figure 10.16 – The Build and Deploy option in the Web service context menu

2. As shown in *Figure 10.17*, select *WebSphere Application Server Community Edition version 2 (all releases)* as the *Web server Type* and check the *Server* radio button to indicate that you want to deploy the Web service directly to an application server. From the *Server* drop down box, select the WAS CE you configured previously.

3. Check the *Register database connection with Web server* check box. This selection triggers the automatic creation of a data source configuration for your database with your Web service and eliminates the need to perform this setup step manually.

4. Select *REST* and *SOAP* as the *Message Protocols.* You may notice that *JMS* is grayed out. You need Optim Development Studio to use the JMS (Java Message Service) binding.

5. Keep the settings in the *Parameters* section.

6. Check the *Launch Web Services Explorer after deployment* check box. This starts the Web services Explorer test environment after the deployment, which allows you to test your Web service.
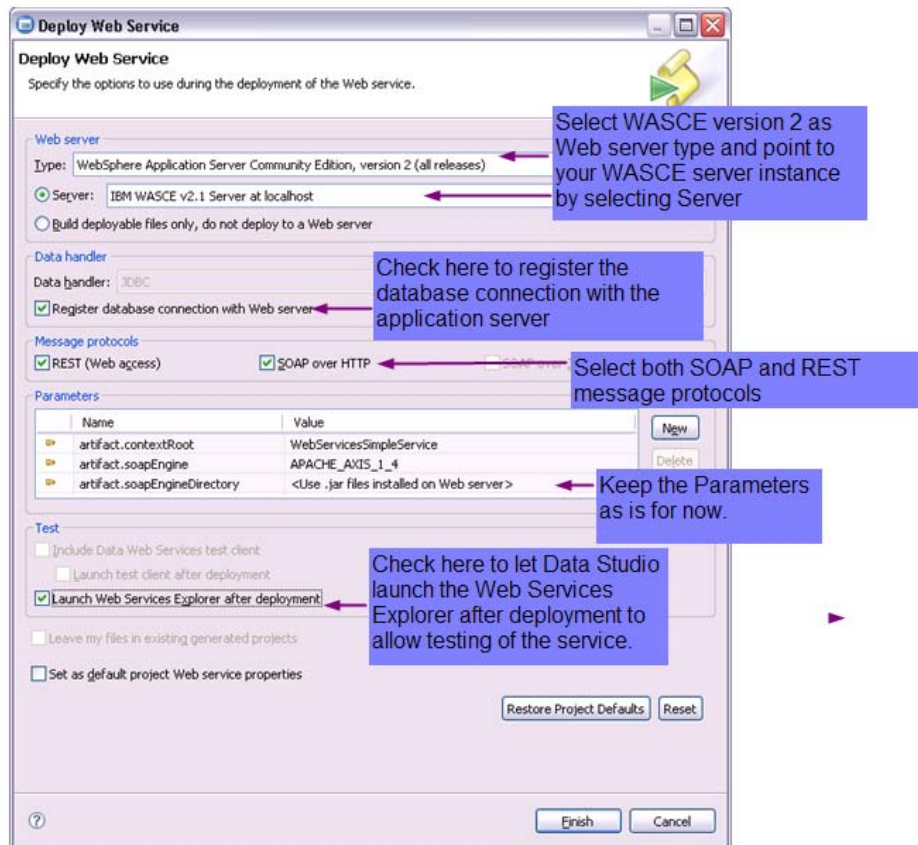
Figure 10.17 – The "Deploy Web Service" dialog

7. Click *Finish.*

While Data Studio deploys the Web service to the WAS CE server instance you will see the "Operation in progress..." message shown in *Figure 10.18*.
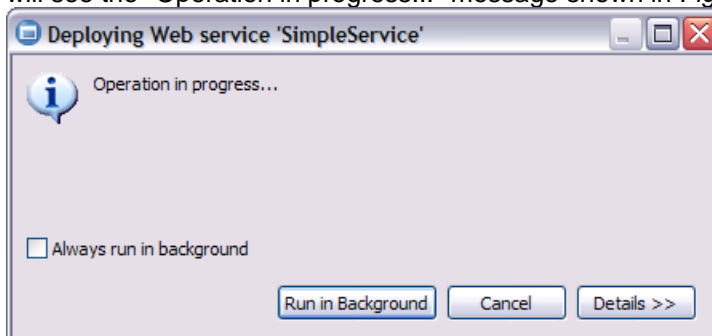
Figure 10.18 – The Web service is being deployed

Under the covers, Data Studio starts up the WAS CE instance (if it's not started already). If this is the first time you've deployed a Data Web Service, you may get asked if Data Studio should update the DB2 JDBC driver at the application server. You should confirm this message to be sure that the latest DB2 JDBC driver is used.

In the next step Data Studio generates the Web service runtime artifacts – like the WSDL file, a JAVA EE Web application project (WAR) and deploys all artifacts to the application server. For more information on what those artifacts are and how to locate them, see *Appendix E*.

In addition, because you checked the box to bring up the Web Services Explorer automatically, it will come up automatically for testing. We'll cover testing in the next section.

Note:

You can also just build the Web service runtime artifacts without automatically deploying them to the application server by selecting *Build deployable files only, do not deploy to a Web server.*

Data Studio generates the Web application project and *.war file for the Web service. You can now take the *.war file and use the application server administration tools to deploy the application manually.

10.7.1. The location of the generated WSDL

The *content* of a SOAP message is usually described in the WSDL (*Web Service Description Language*) document. WSDL is based on XML as well. Data Studio generates a WSDL for each Data Web Service. In fact, the Web Services Explorer requires the WSDL file to be able to communicate with the Web service.

To locate the WSDL file for your *SimpleService* Web service, expand the folder *XML -> WSDL* in your Data Development Project as shown in *Figure 10.19*.
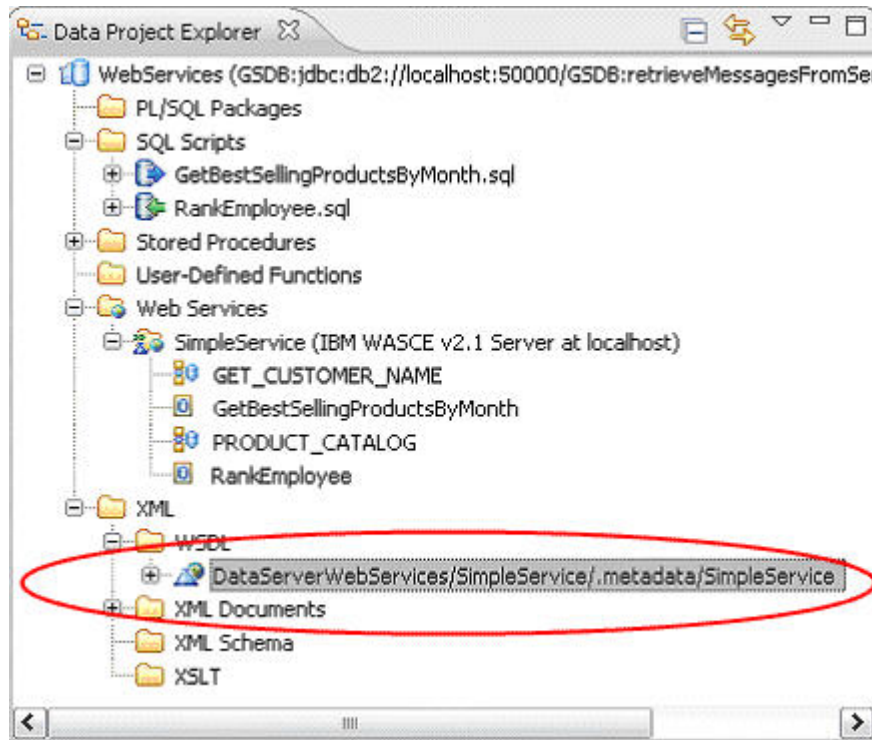
Figure 10.19 – Locating the WSDL file for the SimpleService

You will find a *SimpleService.wsdl* file that represents the WSDL file for your service.

You can also retrieve the WSDL document using a URL after the Web service was deployed on an application server. The URL is:

```
http(s)://<server>:<port>//<contextRoot>/wsdl
```

In the case of the **SimpleService,** the URL would look like this:

```
http://server:8080/WebServicesBookSimpleService/wsdl
```

Explaining the structure of a WSDL document in detail is beyond the scope of this book. You should know that the WSDL contains all the information a Web service client needs to invoke an operation of your Web service. This includes the operation names, XML schemas for input and output messages and, service endpoint definitions.

Note:

Data Studio also includes a WSDL editor. You open the editor by double-clicking on the WSDL document.

10.8 Test the Web Service with the Web Services Explorer

There are many ways to test your new Web service; one easy way is to use the built-in Web Services Explorer of Data Studio. The Web Services Explorer is a dynamic Web services client that uses the WSDL document of the service to initialize itself.

Note:

The Web Services Explorer can test for invocations over SOAP over HTTP. For other bindings, such as JSON or simple HTTP clients without SOAP, you will need to do a bit more work as explained in *Appendix E*. The other option is to use Optim Development Studio, which contains an HTML-based test client that supports all the Data Web Services bindings.

From the previous deployment step, the Web Services Explorer should already be started. In case it is not, you can start it as follows:

1. Go to the *Data Project Explorer v*iew, open your project, and explore your Web service.

2. Click your Web service name and click *Launch Web Services Explorer* to start the Web Services Explorer in Data Studio, as shown in Figure 10.20.
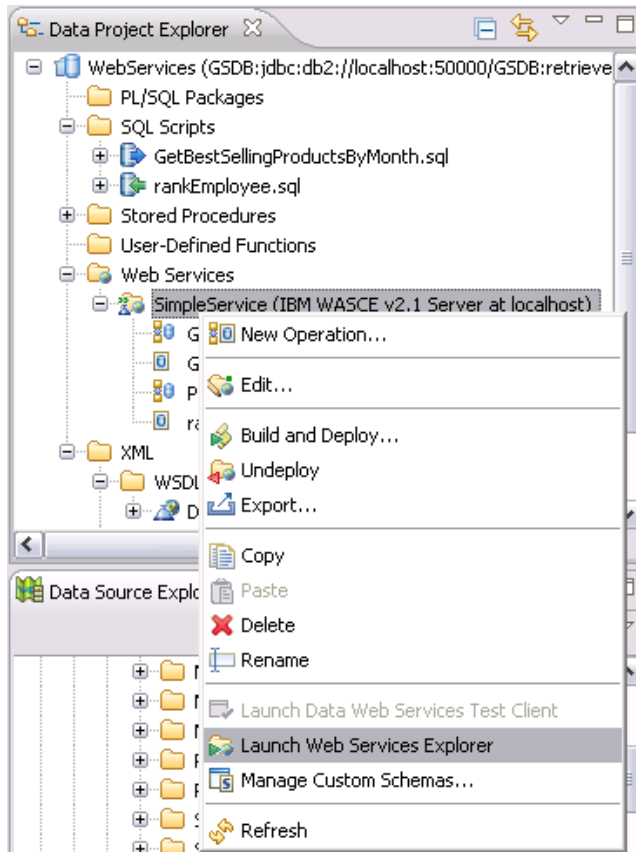
Figure 10.20 – The Launch Web Services Explorer option in the Web service context menu

Figure 10.21 shows a more detailed view of the Web Services Explorer. On the left side, there is a detailed list of all of the components that form your Web service. When expanding the *SimpleService* node you see three Web service bindings listed: *SimpleServiceHTTPGET*, *SimpleServiceHTTPPOST*, *and SimpleServiceSOAP***.** The different bindings will be discussed later in this chapter. Under each binding you find the available operations that can be invoked for the binding. In our case, there are two SQL scripts and two stored procedures. The endpoint to which the arrow points is the location of the service endpoint for the selected binding – in this case, it's the SOAP binding.
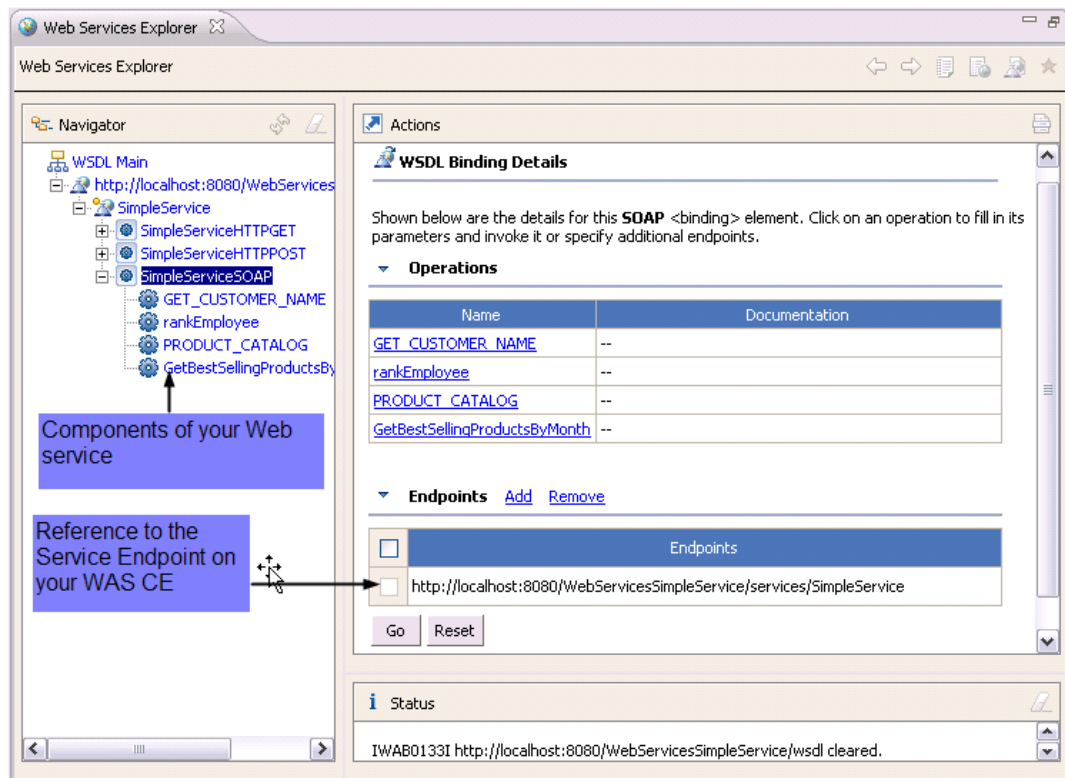
Figure 10.21 – The Web Services Explorer

10.8.1 Testing the GetBestSellingProductsByMonth operation

Now it's time to actually test your Web service operations and inspect the results. You can start with the *GetBestSellingProductsByMonth* operation and the SOAP binding.

1. As shown in *Figure 10.22,* expand the *SimpleServiceSOAP* node in the *Web Services Explorer Navigator* pane, and select the *GetBestSellingProductByMonth* operation.

2. The right-hand frame changes and presents an input field for the month. Remember that this operation is based on an SQL **SELECT** statement with a named parameter marker called **MONTH**. Provide any valid numeric month value – in this case we used 4 (for April).
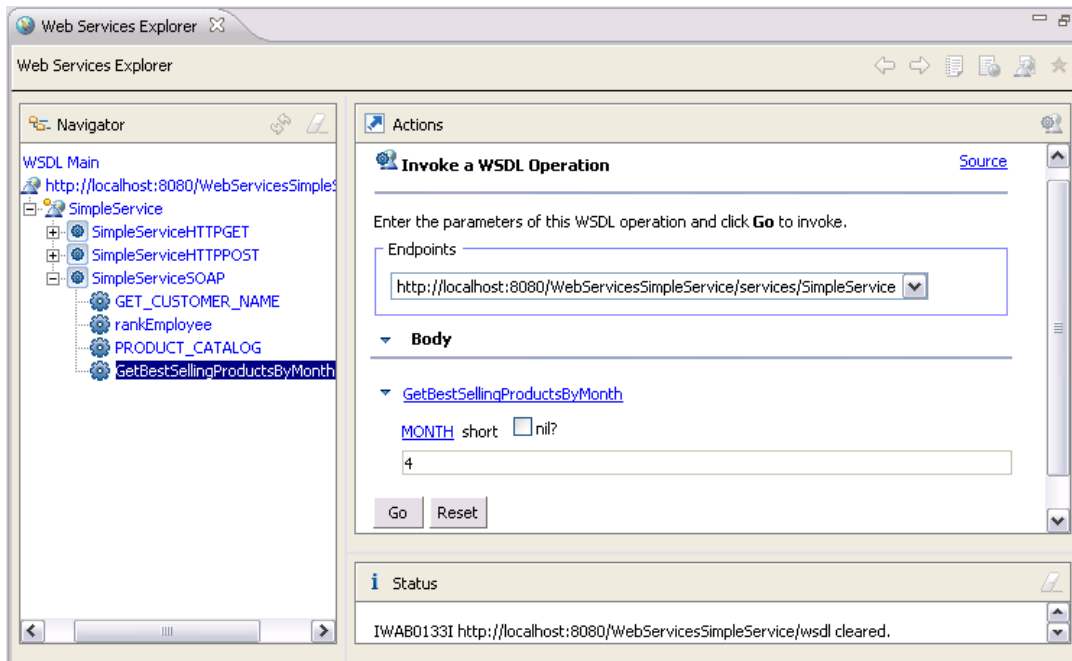
Figure 10.22 – Select the GetBestSellingProductsByMonth operation

3.   Select *Go* to issue the Web service request.

The Web Services Explorer generates the appropriate SOAP request message and sends it to your Web service on WAS CE. The Web service invokes the SQL SELECT statement and returns the result set formatted as XML in a SOAP response message back to the Web Services Explorer. The Web Services Explorer parses the SOAP response message and presents the result in the lower right *Status* window as shown in *Figure 10.23*. (You may need to expand the view and use the scroll bar to see the results.) This is known as the **Form** view because it displays the request message parameters in an HTML-like form.
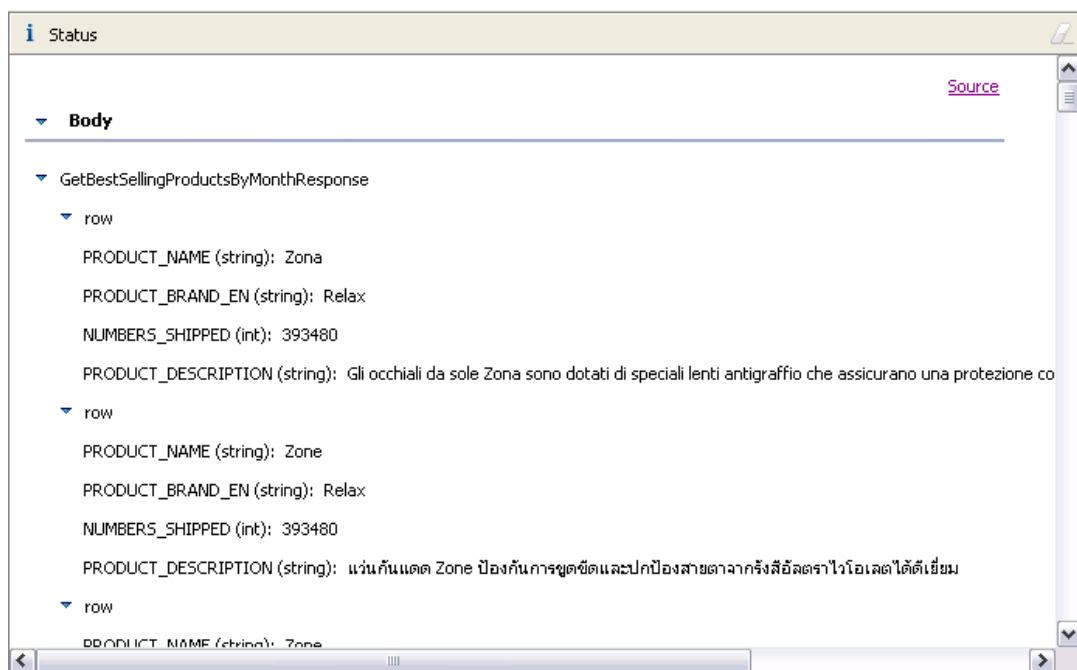
Figure 10.23 – The Web service response in the Form view

4. You can examine the raw SOAP request and response messages by clicking *Source* in the upper right corner of the *Status* window. The source appears as shown in Figure 10.24 in what is known as the **Source** view.
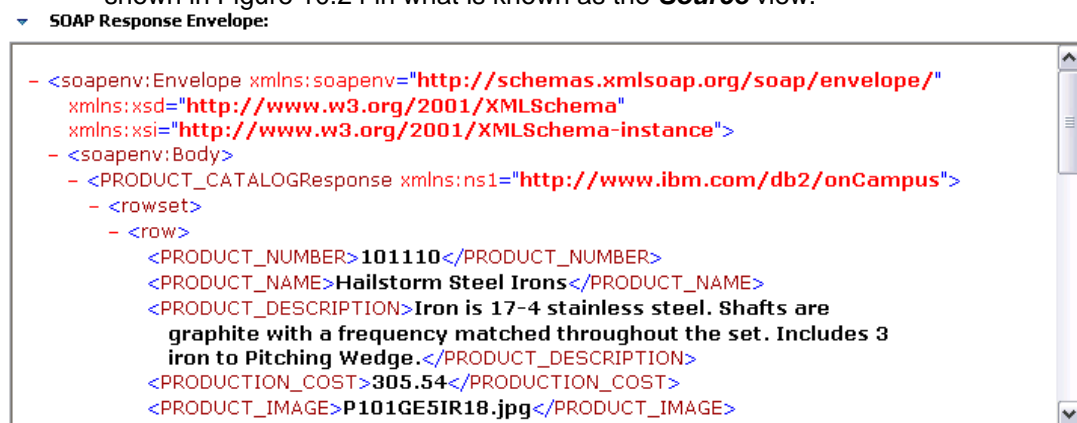
Figure 10.24 - The source view of the SOAP request and response messages

10.8.2 Testing the *PRODUCT_CATALOG* operation

Let's try another operation.

1. This time select the *PRODUCT_CATALOG* operation from the *Web Services Explorer Navigator* under the *SimpleServiceSOAP* node.

2. This operation is based on a stored procedure that returns a product catalog excerpt by a given PRODUCT_TYPE. Enter `Irons` for the PRODUCT_TYPE input parameter and click *Go* to issue the request.

3. You may notice (*Figure 10.25*) that the form-based response looks a bit strange. Not all columns for a product catalog item are displayed.
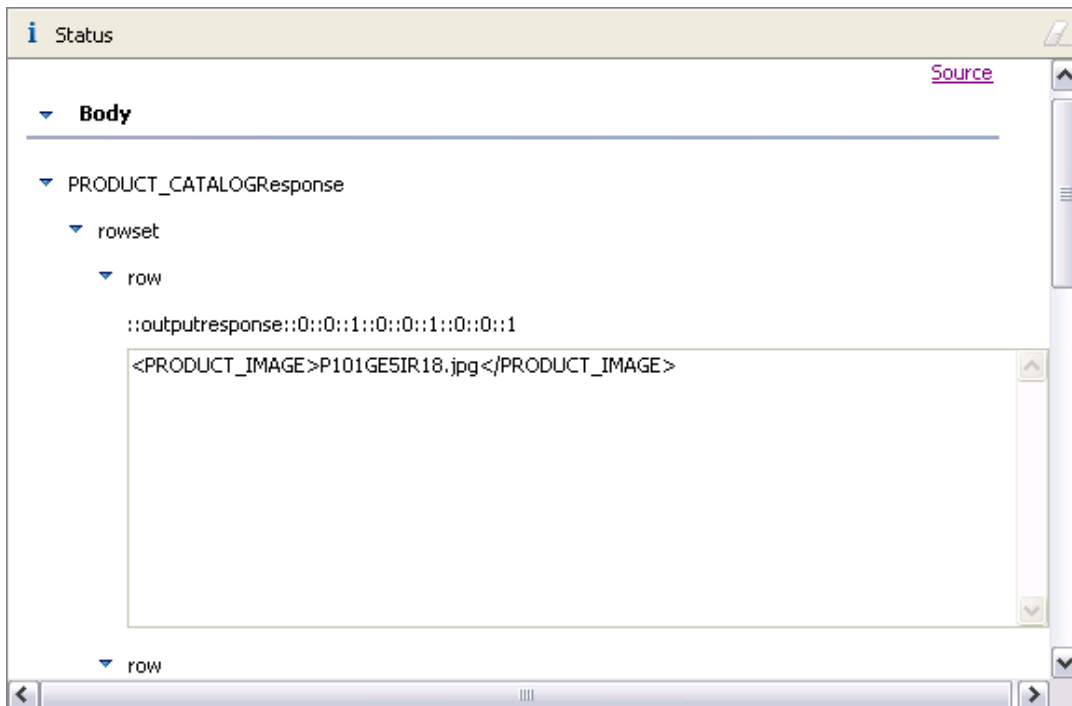
Figure 10.25 – A stored procedure response with result set in the Form view

4. But when switching to the SOAP message source view (*Figure 10.26*) you can see that all the data is present.
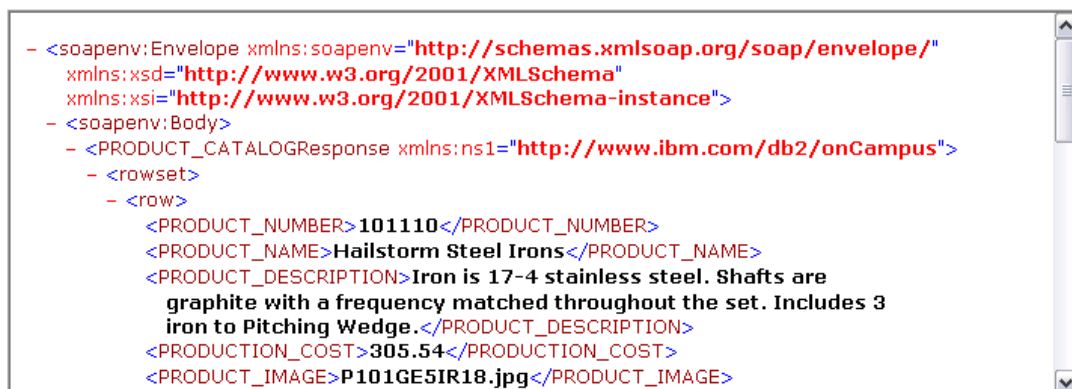
Figure 10.26 – A stored procedure response with result set in the Source view

The reason that you don't see all the columns in the *Form* view is because of the fact that the DB2 catalog does not contain metadata for stored procedure result sets. Therefore Data Web Services can only apply a very generic result set schema, which may not contain

enough information for Web service clients to handle the data. In *Appendix E*, we show how you can work around this limitation.

You can now try to test the other Web service operations with the Web Services Explorer.

Figure 10.27 shows you the result of the **GetBestSellingProductsByMonth** operation when using HTTP POST, which just displays the results as an XML document.
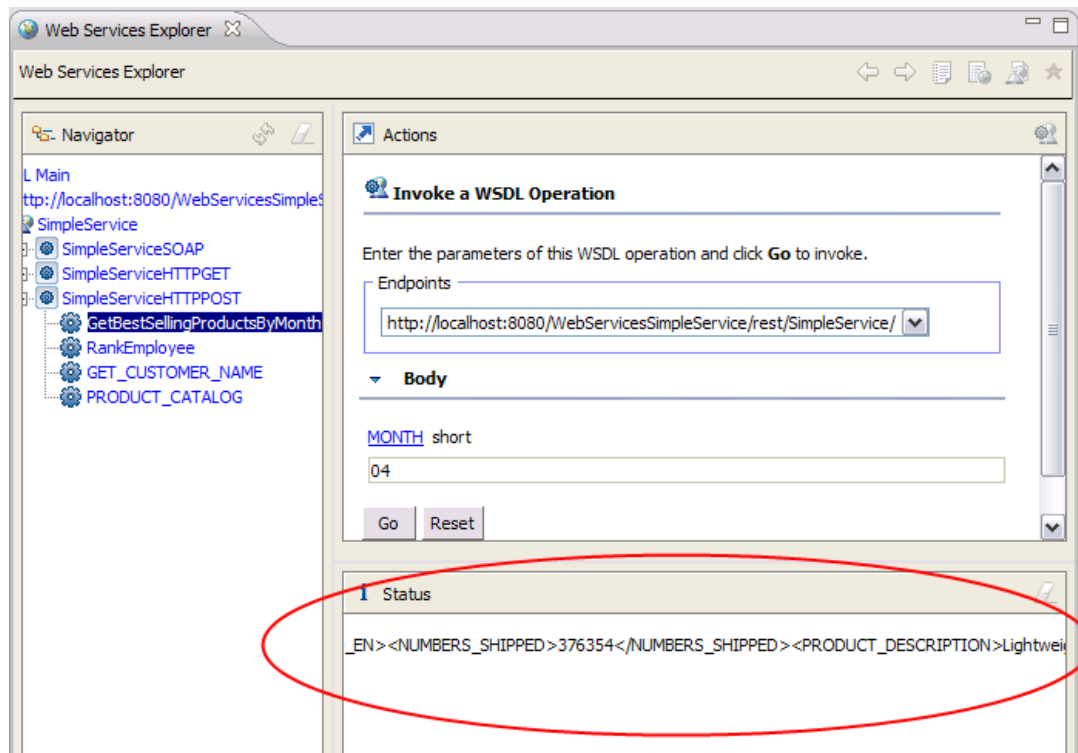
Figure 10.27– HTTP POST response is shown as an XML document

10.9 Exercises

1. Test the **RankEmployee** operation. All English ranking descriptions can be found in the **RANKING_DESCRIPTION_EN** column of the **GOSALESHR.RANKING** table. You can use any of the rankings as an input value for the **RANKING** parameter while testing. Select an **EMPLOYEE_CODE** from the **GOSALESHR.EMPLOYEE** table. Verify that your new ranking has been added by looking in the **GOSALESHR.RANKING_RESULTS** table.

2. Create a new Web service operation which updates the ranking for a given **EMPLOYEE_CODE** and a given **YEAR** to a given **RANKING_DESCRIPTION.**

3. Invoke the **GET_CUSTOMER_NAME** operation using a Web browser via the HTTP GET binding. **Hint:** You can execute the HTTP GET binding from the Web Services Explorer and copy-and-paste the URL into a Web browser.

4. Change the SQL statement which represents the
   GetBestSellingProductsByMonth operation to allow a user to provide the
   month name instead of the month number. **Hint:** You can use the following
   expression to find the month name for the
   GOSALES.INVENTORY_LEVELS.INVENTORY_MONTH column:

   ```
   MONTHNAME('2009-' || TRIM(CHAR(INVENTORY_MONTH)) || '-01-
   00.00.00')
   ```

5. Check out the behavior for binary data types. Create a new Web service operation
   called *checkBinary* with the following statement:

   ```
   SELECT BLOB(CAST(:input AS VARCHAR(255)) FROM
   SYSIBM.SYSDUMMY1
   ```

   Deploy the Web service with the new operation. Execute the operation by
   providing any string value as input. Observe the result. Try to find out the XML data
   type and explain why binary data is represented in this form. **Hint:** You can find the
   XML data type by examining the XML schema section in the WSDL document of
   the Web service.

10.10 Summary

In this chapter, you've learned about the architecture of Data Web Services, which
provides the ability to wrap Web services around business logic that is provided by SQL
statements, XQuery statements, or stored procedures. The services can be bound to either
SOAP or REST style bindings, providing the flexibility for a variety of clients to invoke and
consume the services. This chapter walked you through the process of creating a Data
Web Service that includes two stored procedures and two SQL statements and binding
them to both SOAP and simple HTTP protocols. The SOAP binding can easily be tested
using the Web Services Explorer. For information about testing other bindings, see
Appendix E.

10.11 Review questions

1. What are the three bindings supported for testing by the Data Studio Data Web
   Services explorer?

2. What does it mean when the Data Web Service name in the Data Project Explorer
   has an asterisk by it?

3. To see the SOAP request and response messages, which view do you need to
   open from the Web Services Explorer?

4. As a best practice, is it better to use named parameter markers or question mark
   for the SQL used in Data Web Services?

5. The approach of creating a Data Web Service based on existing database logic is
   called _____ development.

6. You create a new Data Web Service in:

    A. A Data Design project

    B. A Data Development project

    C. The Data Source Explorer

    D. SQL and XQuery Editor

    E. None of the above

7. Business logic for a Data Web Service can be provided by:

    A. SQL procedures

    B. XQuery statements

    C. SQL statements

    D. All of the above

    E. None of the above

8. Which transport protocol is used with a Data Web Service?

    A. FTP

    B. RMI

    C. HTTP

    D. SMTP

    E. None of the above

9. What is the Web Services Explorer used for?

    A. Browsing the Web

    B. Editing XML files

    C. Testing Web services

    D. Browsing the file system on a remote server

    E. All of the above

10. What are the three major steps in the development of a Data Web Service?

    A. Design, develop, deploy

    B. Create, deploy, test

    C. Model, develop, test

    D. Design, model, deploy

    E. None of the above

11

Chapter 11 – Getting even more done

In this book you've learned about how to use Data Studio to perform basic database administration and data development tasks with DB2. But there are a wide variety of tasks and management responsibilities involved in managing data and applications throughout the lifecycle from design until the time that data and applications are retired. IBM is helping organization manage information as a strategic asset and is focused on helping them manage their data across its lifecycle.

In this chapter, you'll learn more about some of tools and solutions from IBM that you can use to address the bigger challenges of managing data, databases, and database applications.

We encourage you to try these products. You may have access to some of the software as part of the IBM Academic Initiative program at www.ibm.com/developerworks/university/data/, or you can download the 30-day trial versions where available.

In this chapter you will learn about:

- The major phases in the data lifecycle and key tasks for each of those lifecycle phases.

- Why a lifecycle focused approach provides greater value than only focusing on specific tasks.

- Some of the products that address the challenges of data management and a summary of their capabilities.

- How these products can extend the Rational® Software Delivery Platform for data-driven applications.

11.1 Data lifecycle management: The big picture

Figure 11.1 illustrates the data management lifecycle phases and key value that a data lifecycle management approach can bring to those phases.
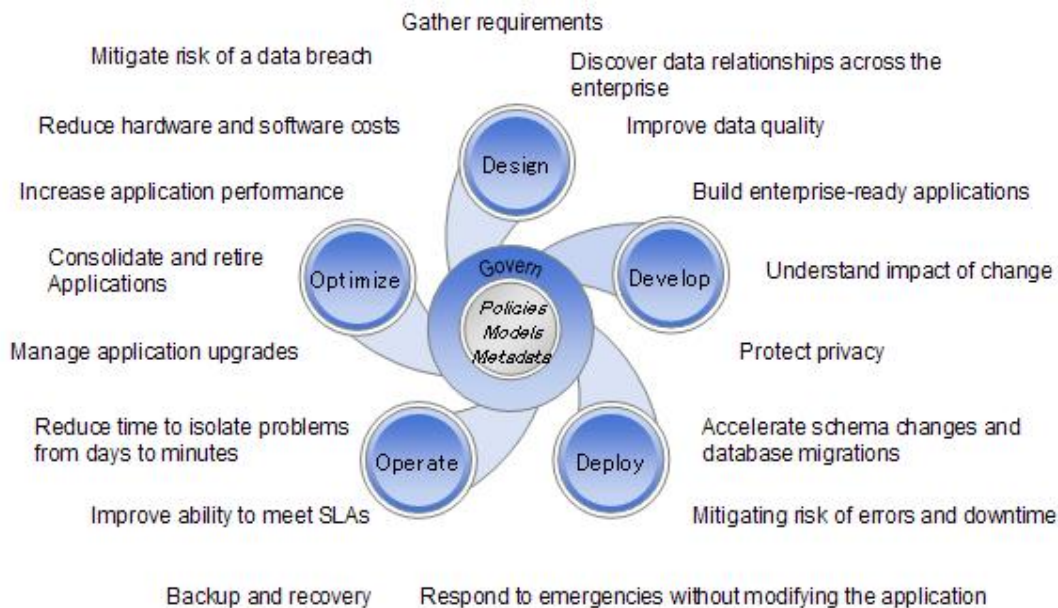
Figure 11.1 -- Managing data across the lifecycle can enhance value and productivity from requirements through retirement

As you can see in *Figure 11.1*, there are many considerations for effective management of data, databases, and data applications. As described by Holly Hayes in her developerWorks article entitled "*Integrated Data Management: Managing data across its lifecycle*", the main steps involved in the complete data lifecycle include:

- **Design** -- Discover, harvest, model, and relate information to drive a common semantic understanding of the business. You may need to interact with business users to track and gather requirements and translate those requirements into a logical design to share with application architects. A physical database model is generally used as the way to convert a logical design to a physical implementation that can be deployed into a database management system. If you are working with existing data assets (databases), you need to understand what tables already exist, and how they may relate to other tables or new tables you may want to create. In addition, you may wish to conform to a naming standard or enforce certain rules about what kind of data can be stored in a field or whether data stored in a field must be masked for privacy. All of these considerations happen during the design phase of the lifecycle.

- **Develop** -- Code, generate, test, tune, and package data access layers, database routines, and data services. This step is where the data access application is built. The data access may be part of a larger application development process, so it's important to collaborate closely with business developers and to ensure that application requirement changes are reflected back to the data architect or DBA for changes. In addition, developers may be responsible for ensuring that the data access they create (SQL, XQuery, Java, Data Web Services, etc.) not only returns

the correct result but also performs efficiently. Use of representative test data and test databases is often used. Because of regulations around how personally identifiable information such as social security numbers and credit card numbers can be handled, it's critical that developers who need test data are compliant with those regulations while still having representative test data.

- *Deploy* -- Install, configure, change, and promote applications, services, and databases into production. This phase includes a well-planned strategy for migrating databases (or database schema changes), data and applications into production. The goal is to do this as swiftly as possible and with the least amount of disruption to existing applications and databases to avoid affecting other applications and to do it without error. Deployment can also mean deploying changes.

- *Operate* -- Administer databases to meet service level agreements and security requirements while providing responsive service to emergent issues. This phase of the lifecycle is the bread and butter of a typical DBA's day. They authorize (or remove authorizations) for data access. They not only have to prepare for possible failures by ensuring timely backups, but they must also ensure that the database is performing well and they must be able to respond to issues as they arise. Because many failures can be difficult to isolate (that is, is a failure occurring in the database, the application server, the network, the hardware?), it's critical that all members of the IT staff have information to help them isolate the problem as quickly as possible so that the right person can fix the problem, whether it's the DBA, the network administrator, the application administrator or someone else.

- *Optimize* -- Provide pro-active planning and optimization for applications and workloads including trend analysis, capacity and growth planning, and application retirement including executing strategies to meet future requirements. This phase is where DBAs can really bring value to the business. It may take a backseat to the constant interrupt-driven needs of day to day operations, but it is a critical phase to ensure that costs are kept down and performance remains acceptable as the business grows and as more applications drive more users against the databases. It's critical that performance trends and data growth trends are analyzed and accommodated. A strategy for archiving old data is required for two reasons: 1) to manage data growth to ensure performance is not adversely affected and 2) to comply with regulations for data retention.

- *Govern* -- Establish, communicate, execute, and audit policies and practices to standardize, protect and retain data in compliance with government, industry, or organizational requirements and regulations. Not limited to a single phase, governance is a practice that must infuse the entire lifecycle. Governance can include the data privacy regulations mentioned previously as well as using techniques such as data encryption to guard against data breach or accidental loss. [1] In fact, data lifecycle management is just one aspect of Information Governance. IBM has collaborated with leading organizations to identify a blueprint and maturity

model for Information Governance. Find more information here: http://www-01.ibm.com/software/info/itsolutions/information-governance/

Although many products and technologies exist today to help with the phases of the data lifecycle, IBM is focusing on creating an infrastructure in which specifications made in one phase can be disseminated through other phases of the lifecycle and automatically maintained.

Why is this important? Although you may be in school or in a small development shop where there are very few people other than yourself managing data and applications, there are real problems as organizations grow and responsibilities become dispersed among different people and in different locations. For example, data privacy requirements identified in the design phase may get lost or forgotten as developers start pulling down data from production for testing purposes. It becomes more and more difficult to identify how a database schema change will affect the many applications that may be using the database. And not identifying dependencies properly can result in serious outages.

 With an integrated approach, the tools can actually help facilitate collaboration among roles, enforce rules, automate changes while identifying dependencies, and in general speed up and reduce risk across the lifecycle. This integrated approach cannot be achieved by unrelated tools. It requires common infrastructure and shared metadata such that actions in one tool are reflected down the line when another person uses their tool to support their particular responsibilities. InfoSphere Optim solutions for data lifecycle management are built to take advantage of such integrations. So, as an example, if the Data Architect defines a column as containing private data (such as credit card numbers or social security numbers), a developer who is viewing this table in their development tool should see the column marked as 'private' and be able to invoke proper masking algorithms should data be required for testing.

11.2 Optim solutions for data lifecycle management

Let's look at some of the data lifecycle management solutions from IBM, many of which are offered under the family name "InfoSphere"
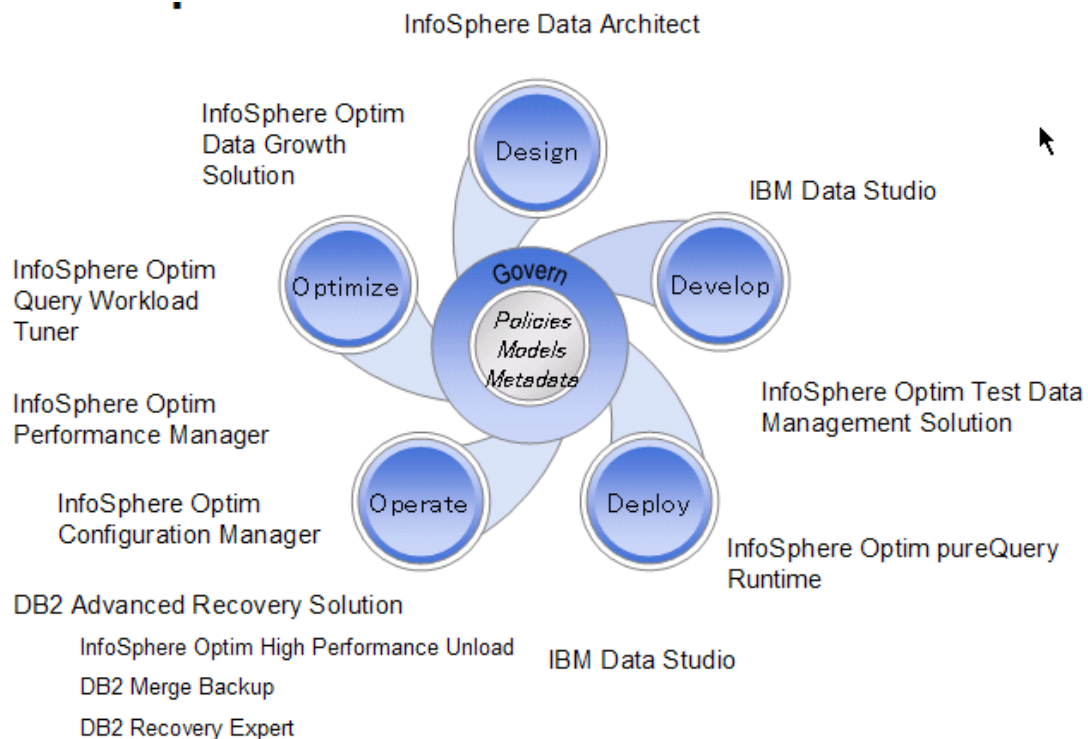
Figure 11.2 – InfoSphere Optim solutions for Data Lifecycle Management

Figure 11.2 shows some of the key products that help IT staff manage the various phases of the data lifecycle. We won't cover all the products in great detail here, but will cover a few key ones that you may wish to download and use to expand the capabilities of Data Studio as you learn more about working with DB2 Express-C and other databases.

11.2.1 Design: InfoSphere Data Architect

The data architect's key tool is InfoSphere Data Architect, used for discovering, modeling, relating, and standardizing data. Like any good data modeling offering, it supports logical and physical modeling and automation features for diverse databases that simplify tasks such as reverse engineering from existing databases, generating physical models from logical models, generating DDL from physical models, and visualizing the impact of changes. For warehouse development, it includes automatic discovery and annotation of facts, measures, dimensions and outriggers and denormalization into star, snowflake, and starflake schema.

InfoSphere Data Architect is more than a data modeling tool because it provides a framework for understanding your data, and can be used as a basis of understanding in other tools. Because it integrates with InfoSphere, InfoSphere Optim, and Rational offerings, as well as supporting heterogeneous environments, InfoSphere Data Architect is a central component in information governance strategies.

Figure 11.3 shows a screenshot of a model in InfoSphere Data Architect.
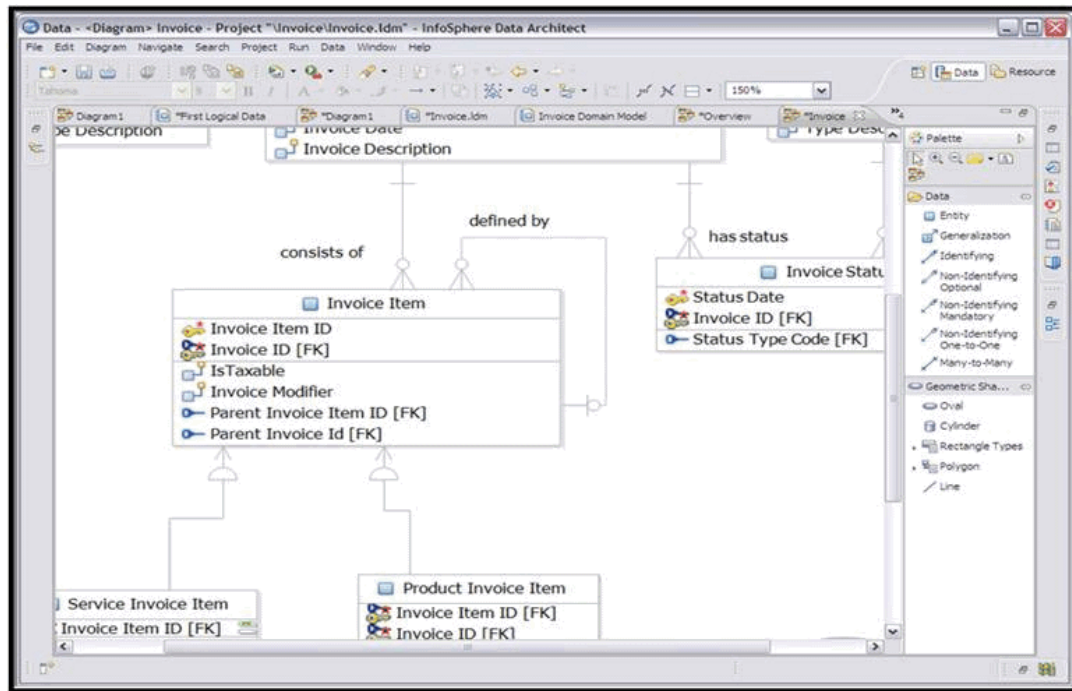
Figure 11.3 – InfoSphere Data Architect for data modeling

For more information about InfoSphere Data Architect, see the ebook *Getting Started with InfoSphere Data Architect* which is part of this book series.

11.2.2 Develop: Data Studio and InfoSphere Optim pureQuery Runtime

For data-oriented developers or DBAs, Data Studio contains all the database administration and data development capabilities a database developer needs. But is goes beyond the basics for Java development when used with InfoSphere Optim pureQuery Runtime

Data Studio ramps up Java development to new levels for heterogeneous databases. The data access layer generation includes support for the standard Data Access Object (DAO) pattern and leverages the pureQuery API, an intuitive and simple API that balances the productivity boost from object-relational mapping with the control of customized SQL generation. It also simplifies the use of best practices for enhanced database performance such as statement batching and static execution. InfoSphere Optim pureQuery Runtime is used with pureQuery data access layers.

For both pureQuery and other Java applications that might be using Hibernate, JPA, or some other framework, you can take advantage of the great features in Data Studio, as shown in *Figure 11.4*.
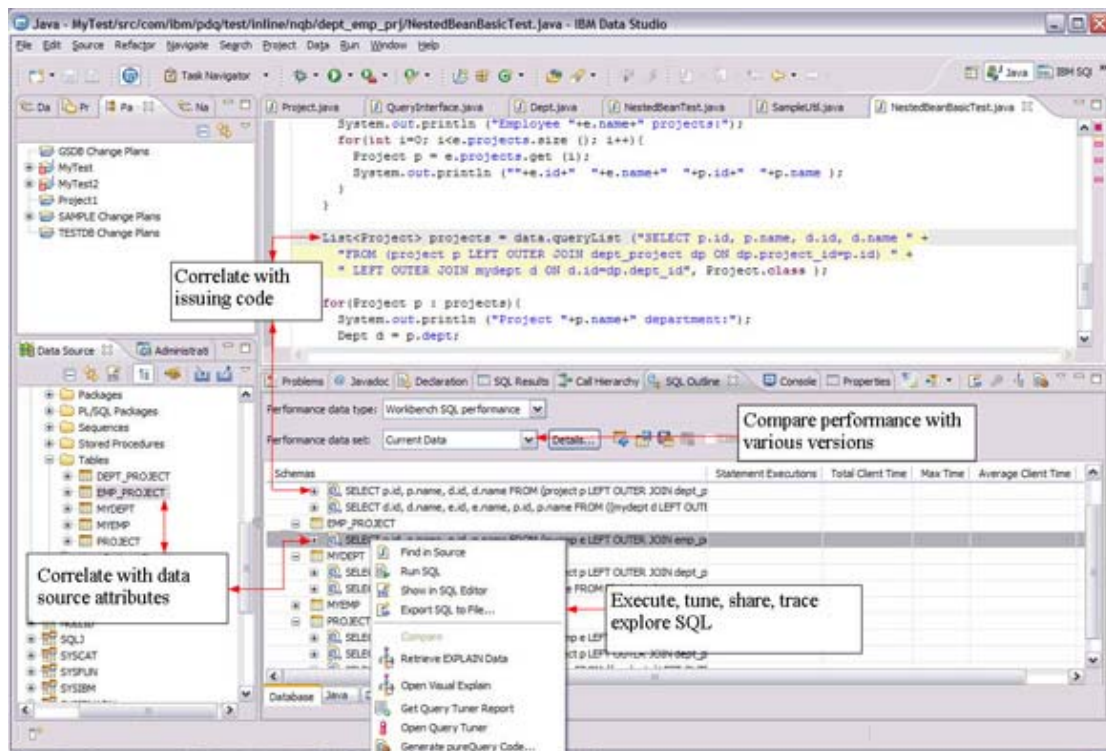
Figure 11.4– Data Studio has advanced features for Java database development and tuning

As shown in *Figure 11.4,* one great feature is the ability to correlate SQL with the data sources and with the Java source code, even if the SQL is generated from a framework. This can really help you understand the impact of changes, and can also aid DBAs and developers in identifying and tuning SQL when using output from the DB2 package cache during QA or production.

You can start learning about SQL performance issues by visualizing SQL "hot spots" within the application during development by seeing execution metrics around how many times a statement is executed and the elapsed times. Adding InfoSphere Optim Query Workload Tuner to your environment can help you tune you SQL by providing expert guidance and rationale to build your tuning skills.

In addition, because of the integration with other products, Data Studio helps developers be cognizant of sensitive data. For example, developers can readily identify sensitive data based on the privacy metadata captured in InfoSphere Data Architect. They can create test databases directly from test golden masters or can generate extract definitions for InfoSphere Optim Test Data Management and InfoSphere Optim Data Privacy solutions to create realistic fictionalized test databases. The physical data model is shareable among InfoSphere Data Architect, Data Studio, and InfoSphere Optim Test Data Management Solutions to enable collaboration and accelerate development.

Developers can spend considerable time isolating performance issues: first to a specific SQL statement, then to the source application, then to the originating code. Three-tier architectures and popular frameworks make this isolation more difficult as the developer may never see the SQL generated by the framework. Data Studio makes it easier to isolate problems by providing an outline that traces SQL statements back to the originating line in the source application, even when using Java frameworks like Hibernate, OpenJPA, Spring, and others. .

Be sure to read the *Getting Started with pureQuery* book of this series to read about the capabilities of Data Studio and pureQuery Runtime.

11.2.3 Develop and Optimize: InfoSphere Optim Query Workload Tuner

InfoSphere Optim Query Workload Tuner, is focused on enabling developers to tune queries and query workloads by providing them with advice on how to achieve better query performance [2]. See *Figure 11.5* for a screenshot.
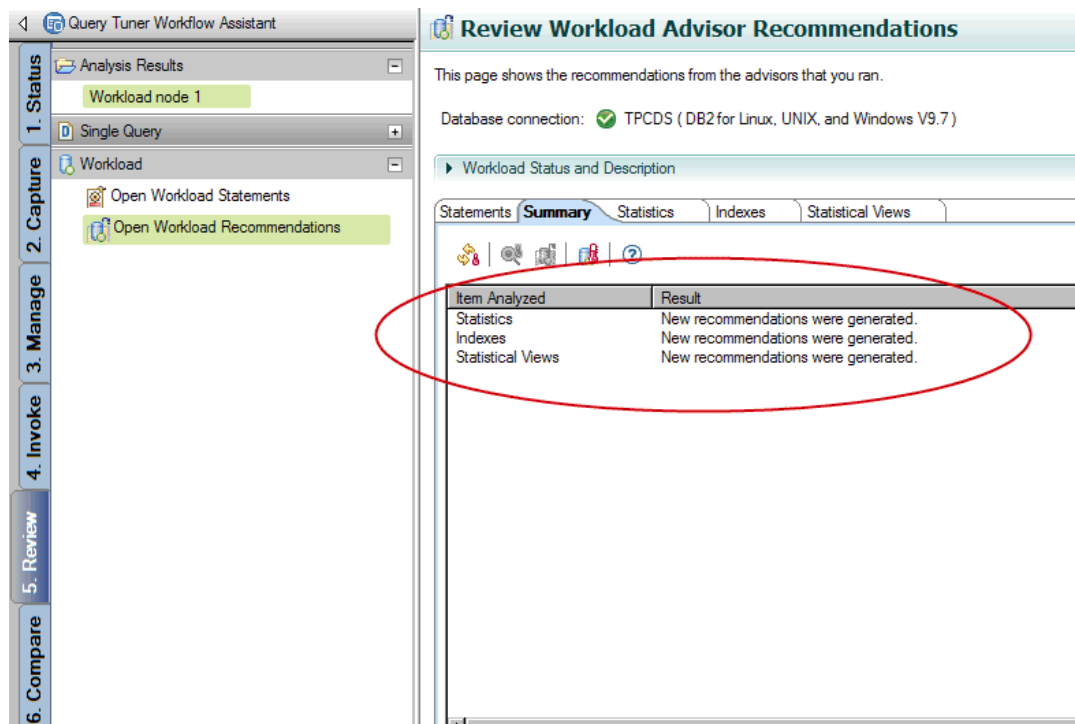
Figure 11.5 – InfoSphere Optim Query Workload Tuner provides help for tuning queries and workloads

InfoSphere Optim Query Workload Tuner provides advice for statistics, queries, access paths, and indexes. As you learned in Chapter 7, the tabs along the left side help step you through tuning steps, and the tool can format the queries for easy reading and include associated cost information, an access plan graph, access plan explorer, and access plan comparison.

InfoSphere Optim Query Workload Tuner can provide either single query analysis and advice , but can take as input an entire SQL workload (such as all SQL used in an order processing application) as input, which enables DBAs to determine for example what indexes or what statistics might provide the most benefit for the overall performance of the workload.

11.2.4 Deploy and Operate: Data Studio, InfoSphere Optim Configuration Manager, and DB2 Advanced Recovery Solution

Managing availability is often job number one for DBAs. When your database goes down and the data is unavailable to the end users, it can look bad for you and your organization. If you're supporting a business, that can have a direct impact on the bottom line.

The DB2 Advanced Recovery Solution is focused on reducing time to recover by aligning backup strategies with outage related Service Level Agreements providing faster methods to recovery.

DB2 Advanced Recovery Solution is comprised of the following:
- **InfoSphere Optim High Performance Unload** provides a high-speed unload utility as an alternative to the DB2 export feature. This product significantly reduces the time required to migrate databases, manage within batch windows, capture data for test environments, and move data without impacting production systems.

  Because unloads are so fast, you can use this as a means for migration, moving large amounts of data from one system to another or for backup,.

  The product is fast because it can o go directly to the data files, bypassing the database manager altogether. The tool does not interfere with or slow down production databases or impact CPU resources as it is completely outside of the database. It can also perform unloads from multiple database partitions, and it provides repartitioning capability in a single step for rapid data redistribution on the same or different system. This is particularly useful in warehouse environments where repartitioning can be very much a manual process.

- **DB2 Merge Backup** lets you avoid full database backups by merging incremental and delta backups into a full backup. Thus, it reduces resource requirements to maintain a full backup for large databases and it shortens recovery times on production servers ensuring full backups are always available when needed.

  This is very helpful where it just takes too long to take a full backup and in some cases it is not viable because of the amount of time it takes and because of the impact backup has to end users, who cannot access the database while it is being backed up. DB2 Merge Backup lets you have full backups available at a more consistent and up to date point in time. DB2 Merge Backup also gives you the capability to run the merge processing on a computer outside of your DB2 database, thus reducing the amount of resources being consumed on the production computer.

- **DB2 Recovery Expert** optimizes recovery processes. It helps organization minimize recovery times by isolating recovery to just the impacted objects without having to resort to full database recovery. This means that you can recover faster and with greater granularity than what is available with traditional database recovery. Think about the situation where you have multiple tables in a tablespace. If a user deletes some data from one of those tables by accident, you can identify which data was deleted and recover just that data rather than having to recover the whole tablespace.

  DB2 Recovery Expert log analysis capabilities enables single object recovery as well as recovery from data corruption caused by flawed applications. This allows DBAs to quickly restore or correct erroneous data using fewer resources. Log analysis also lets DBAs monitor activity to sensitive DB2 tables. DBAs can view data changes by date, users, tables, and other criteria. Administrators may institute tighter controls over the data to ensure that the data is no longer compromised.

Data Studio, as you have learned in this book, provides all the basic database administration capabilities required for managing a DB2 deployment. However, when managing many deployments, more capability is desirable to manage, synchronize, and govern configuration across hundreds of databases. **InfoSphere Optim Configuration Manager** discovers databases, clients, and their relationships and tracks configuration changes across them. It assists with application upgrades to determine that all clients have been changed correctly, and lets organizations quickly visualize and report on inventory and configuration changes. Such changes can also be correlated to performance degradation via contextual links in **InfoSphere Optim Performance Manager**

11.2.5 Optimize: InfoSphere Optim Performance Manager and InfoSphere Optim Data Growth Solutions

Organizations not only want their applications to run, but to run well. While DB2 provides advanced self-tuning features, it can only optimize within the constraints of the resources it has available. Organizations still need to monitor database performance to detect performance erosion that can lead to missed service level agreements and declining staff productivity. Beyond health monitoring in Data Studio, InfoSphere Optim Performance Manager provides 24 x 7 monitoring and performance warehousing to give DBAs and other IT staff the information needed to manage performance proactively to:

- Prevent problems before they impact the business

- Save hours of staff time and stress

- Align monitoring objectives with business objectives

In addition, when using Data Studio for query development, developers can use the InfoSphere Optim Performance Manager integration to see the actual performance improvements across iterations of query refinements and executions. Application metadata such Java packages and source code lines can be shard with InfoSphere Optim

Performance Manager to accelerate problem resolution. Problematic workloads can be easily transferred to InfoSphere Optim Query Workload Tuner for expert recommendations.

InfoSphere Optim Data Growth Solution is central to managing data growth, archival, and retention. Archiving in-active data support higher performing applications, yet archival must be managed to meet ongoing access requirements for customer support or for audit and compliance. InfoSphere Optim Data Growth provides such capabilities enabling data archival with ingoing access through the native application or though standard reporting or query facilities. Selective restore enables audit database to be created as needed.

11.2.6 Job responsibilities and associated products

In *Chapter 1*, we mentioned that Data Studio can provide a way to grow your skills horizontally across different database platforms. Data Studio can also serve as a stepping stone for the rest of the data lifecycle management products. Because the portfolio focuses on collaboration among different roles, these products can help you work with and learn tasks that can help you become a production DBA, a high performing Java developer, a Data Architect, or even a Data Governance officer.

Table 11.1 includes some job roles and appropriate products that can help with those roles.

Job	Related products
Developer (data access)	Data Studio, InfoSphere Optim Query Workload Tuner (Also helpful to learn Rational Application Developer for WebSphere Software)
Database Administrator (application-focused)	Data Studio, InfoSphere Optim Query Workload Tuner, InfoSphere Optim Test Data Management Solution and InfoSphere Optim Data Privacy Solution
Database Administrator (including data governance responsibilities)	Data Studio, InfoSphere Data Architect, InfoSphere Optim Performance Manager, DB2 Advanced Recovery Solution, InfoSphere Optim Configuration Manager, InfoSphere Optim Data Growth Solution, InfoSphere Optim Data Privacy Solution
Data Architect	InfoSphere Data Architect (also helpful to learn Rational Software Architect for WebSphere Software)

Table 11.1– Job roles and suggested software

11.3 Data Studio, InfoSphere Optim and integration with Rational Software

This section outlines how some of the key InfoSphere Optim products integrate with and extend the Rational Software Delivery Platform. The goal of InfoSphere Optim solutions for Data Lifecycle Management is to create an integrated approach to data management similar to what the Rational Software Delivery Platform does for the application lifecycle.

Therefore, the InfoSphere Optim solutions in general provide data-centric capabilities that can be used alone or installed with existing Rational products (assuming they are on the same Eclipse level).

> **Note:** Data Studio Version 3.1 is built on Eclipse 3.4. For more information about which products can shell share together, scroll to the Eclipse 3.4 section of the technote here:
> http://www.ibm.com/support/docview.wss?rs=2042&uid=swg21279139
>
> For known shell-sharing issues, see this technote:
> http://www.ibm.com/support/docview.wss?rs=3360&uid=swg27014124

This ability to install together and share artifacts enables better collaboration among various roles in the organization, as shown in *Figure 11.7*.
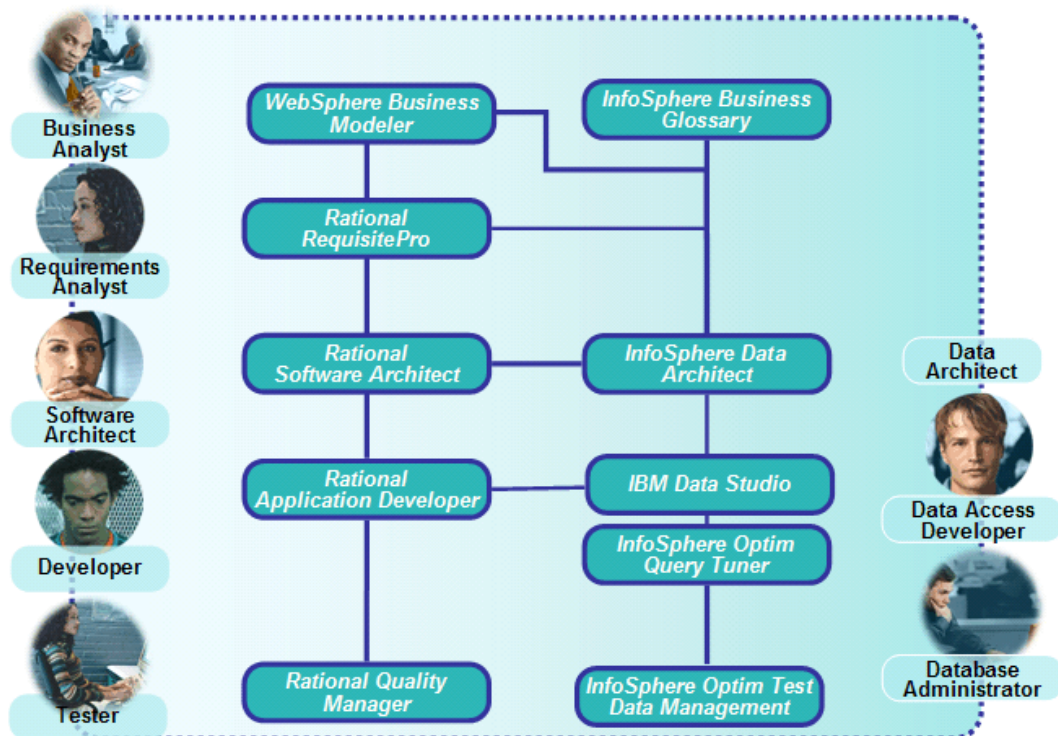
Figure 11.7– InfoSphere Optim Solutions extend Rational for data-driven applications

Let's look at a few particular examples shown in the above figure of how InfoSphere Optim solutions can help developers who are 'data-centric' extend the capabilities of Rational Application Developer for WebSphere Software (RAD).

Example 1: In the database modeling world we usually create logical and physical database models. In the software development world we usually create UML diagrams to portray our application architecture. You can take a UML diagram in Rational and convert that to a logical data model in InfoSphere Data Architect. You can also do the inverse and

take a logical data model in InfoSphere Data Architect and convert that to a UML application model in Rational.

Example 2: Many developers have RAD on their desktops. RAD extends base Eclipse with visual development capabilities to help Java developers rapidly design, develop, assemble, test, profile and deploy Java/J2EE™, Portal, Web/Web 2.0, Web services and SOA applications.

Adding Data Studio also takes your Java persistence layer development capabilities into high gear. Your Java editor will be enhanced with SQL Content Assist, which means that you can use CTRL-Space to see available tables or columns when building your SQL statements in Java. You can use all the other capabilities in Data Studio that can significantly reduce development and troubleshooting time, such as:

- Correlating a particular SQL statement with a particular line of Java source code. This can help you narrow in on problem SQL statements.

- Seeing which tables and columns are being used in the Java program and where, making it much easier to see if a schema change will impact the Java program.

- Searching through SQL statements for particular strings

- Gathering performance statistics on individual SQL statements in your Java program and even comparing performance with an earlier performance run.

In addition, by extending the development environment with InfoSphere Optim Query Workload Tuner, you can get query tuning advice, a task which often falls to the DBA or to a specialized performance management role. InfoSphere Optim Query Workload Tuner can help you avoid simple mistakes when writing queries so that the code is of higher quality and performance before moving into a test environment.

Furthermore, Data Studio can automate database changes. You may need to modify local development databases to reflect changing requirements, and being able to automate this process as well as back out those changes, without requiring the assistance of a DBA can be a great timesaver.

No two organizations are exactly alike and the responsibilities of people working in those organizations can vary significantly, even if they have the same job title. Thus, the modular nature of these capabilities and products make it easy for people to customize their desktops with the capability they need.

11.4 Community and resources

A great resource for learning more about Optim and its solutions is developerWorks, which includes a page from which you can find downloads, forums, technical articles, and more at https://www.ibm.com/developerworks/data/products/optim/.

You can also join the Optim fan page on Facebook to connect with others interested in Optim solutions and to get the latest announcements, at: www.facebook.com/pages/IBM-Optim/37213992975 or follow Optim on Twitter at www.twitter.com/IBM_Optim,

11.5 Exercises

1. Learn more about IBM InfoSphere Optim solutions for Data Lifecycle Management by visiting the Web page at: www.ibm.com/software/data/optim/ which is organized by *solution*. Click through at least two of the solutions listed on this page to see which products are used to *accelerate solution delivery* and *facilitate integrated database administration*.

2. View the demo on the DB2 Advanced Recovery Solution http://www.ibm.com/developerworks/offers/lp/demos/summary/im-optimbackuprecovery.html.

3. For a good introduction to InfoSphere Data Architect, see the video entitled *Introduction to InfoSphere Data Architect* on developerWorks at: www.ibm.com/developerworks/offers/lp/demos/summary/im-idaintro.html

11.6 Summary

In this chapter, we reviewed the concept of a data and data application lifecycle and some of the key tasks associated with the phases of that lifecycle. We described how an integrated approach to data management can make these tasks more efficient and less risky by facilitating collaboration among roles and automatically enforcing rules from one lifecycle phase to the next. We reviewed some of the IBM offerings for data lifecycle management and their key capabilities.

Finally, we closed with a description of how the InfoSphere Optim solutions can extend the capabilities in Rational for data-centric application development.

11.7 Review questions

1. What are the six phases of data lifecycle management, as described in this chapter? Which phase must be considered across all other phases of the lifecycle?

2. Which phase of the data lifecycle is most concerned with translating business requirements into a physical database representation? Which IBM product is primarily used in this phase?

3. Which products are used together to create a high performance data access layer using the data access object (DAO) pattern.

4. Which IBM InfoSphere Optim product is designed to help DBAs and developers improve SQL queries so that they can perform faster?

5. Which IBM InfoSphere Optim product is designed to help identify performance bottlenecks before they impact the business?

6. How does DB2 Merge Backup help organizations minimize downtime after a failure?

7. Which of the following is *not* a primary *goal* of an integrated approach to data lifecycle management:

A. Reduce risk

B. Improve collaboration among roles

C. Exchange metadata

D. Improve efficiency of development and deployment

E. Enforce rules and improve governance

8. When developing queries in Data Studio, what additional products can you use to help you test and tune the performance of the queries?   Choose the best answer below.

A. InfoSphere Data Architect and InfoSphere Optim pureQuery Runtime.

B. InfoSphere Optim Query Workload Tuner and InfoSphere Optim Performance Manager.

C. InfoSphere Optim Performance Manager and InfoSphere Optim High Performance Unload.

D. DB2 Merge Backup and DB2 Recovery Expert

E. None of the above

9. The integration of data-centric capabilities with the Rational Software Delivery Platform is important because (select all that apply):

A. It improves collaboration among people involved in application development lifecycle

B. It enhances application development with data-centric expertise and capabilities to improve productivity for data-centric development

C. It's important to install as many tools as possible into your Eclipse workbench

D. The similar look and feel can help grow skills across roles

E. None of the above

10. Which one of the following tasks is least likely to occur in the Optimize phase of the data lifecycle?

A. Capacity planning

B. Planning for application retirement

C. Controlling the growth of data by archiving data appropriately

D. Creating a Java data access layer

E. None of the above.

A

Appendix A – Solutions to the review questions

Chapter 1

1. The Data Studio client is built on Eclipse, which is an open source platform for building integrated development environments.

2. DB2 (all platforms) and Informix. Other databases are also supported. For a list of supported databases, see http://www-01.ibm.com/support/docview.wss?uid=swg27022147.

3. In Eclipse, *perspectives* are a grouping of views and tools based on a particular role or task. Integrated data management.

4. The default perspective is the Database Administration perspective.

5. True, Data Studio can be used at no charge with supported databases.

6. C. If you want to do .NET development, you must use the Visual Studio add-ins for DB2. (See http://www.ibm.com/software/data/db2/ad/dotnet.html for more information)

7. E

8. B

9. C

10. B, the results appear in a separate tab in the Properties view.

11. The default user ID of the default administrative user is *admin*.

12. True. The Data Studio web console can be viewed by itself within a browser or embedded within the Data Studio full or administration client.

13. B. Deploying Data Web Services is not supported from the Data Studio web console.

14. The Task Launcher is the default page that opens the first time you log into Data Studio web console.

15. E. No additional steps are required to start using Data Studio web console after you have added your first database connection.

Chapter 2

1. Select *Add to Overview Diagram,* and then select the list of tables you want to be shown in the diagram and click *OK.*
2. Schema
3. Sharing connection information with others by the ability to export and import connection information into a workspace.
4. Privileges tab.
5. D
6. B
7. A
8. D

Chapter 3

1. System-managed (SMS), database-managed (DMS), automatic storage.

2. Delimited (DEL), Worksheet format (WSF), and Integrated Exchange Format (IXF)

3. IXF, because structural information about the table is included with the export.

4. The two types of logging are circular and archive. Circular logging is only for uncommitted changes. To restore, you need to go to the last backup image. Archive logging logs all changes, committed and uncommitted and thus recovery can include changes made up to a specified point in time.

5. Recover is a combination of Restore and Rollforward.

6. A

7. B

8. A

9. C

10. B

Chapter 4

1. Yes, we can modify the default thresholds. You need to go to the Health Alerts Configuration page and select a database and then edit the threshold for each alert type.

2. The following alerts are supported -

- o Data Server Status - Creates an alert for different data server states including: Available, Unreachable, Quiesced, Quiesce Pending, and Rollforward.

- o Connections - An alert is generated when the number of connections to the database exceeds the threshold. This alert is disabled by default.

- o Storage - An alert is generated for the following situations:

  - Table Space utilization exceeds the threshold

  - Table Space container utilization exceeds the threshold. This alert is disabled by default.

  - Table Space container is inaccessible

  - Table Space is in Quiesced state

  - Table Space is Offline

- o Recovery - An alert is generated for the following situations:

  - Table Space is in Restore Pending or Rollforward Pending state

  - Table Space is in Backup Pending state

  - Table Space is in Drop Pending state

  - Primary HADR is disconnected

- o Partition Status – An alert is generated when the status of a partition is OFFLINE

- o Status of DB2 pureScale members – An alert is generated if any of the DB2 pureScale members is in any of the following states: ERROR, STOPPED, WAITING_FOR_FAILBACK, or RESTARTING.

- o Cluster Facility status of DB2 pureScale – An alert is generated if the pureScale cluster facility is in any of the following states: ERROR, STOPPED, PEER, CATCHUP, or RESTARTING.

- o Cluster Host Status of DB2 pureScale – An alert is generated if the DB2 pureScale Cluster Host status is INACTIVE

3. You can share alerts with others by adding your comments using the Comment button on the Alert List page and then sending an email to your colleague with those comments.

4. You need to configure the Data Studio web console in the Preferences in the Data Studio client and then click on any of the integration points such as the Health Summary, or Current Application

Chapter 5

1. Data Development Project and Data Design Project

2. *Default schema*, used to set the database CURRENT SCHEMA register. *Default path*, used to set the database CURRENT PATH register.

3. B

4. A

5. DB2 for Linux, UNIX and Windows (V9.7)

    DB2 for Linux, UNIX and Windows (V9.8)

    DB2 for z/OS (V10)

    DB2 for z/OS (V9)

    DB2 for i

    Informix

6. Yes

7. C

8. JDBC and Command Line Processor are the two available Run methods.

9. No

10. D

11. E

12. B

13. C

14. The SQL and XQuery Editor has the following features: SQL statement syntax and semantic validation, SQL statement execution preferences (Commit, Rollback), special registers and the ability to invoke Visual Explain, Query Tuning and Job Manager for a current script in the Editor.

Chapter 6

1. FALSE. A job does not contain the database information or the date and time that the job will run. This information is stored in a schedule. A job might have many schedules associated with it, but each schedule can only be associated with one job.

2. To send notifications you must first configure the web console with the details about your outbound SMTP mail server so that information can be sent to e-mail addresses.

3. C. All jobs. When you run a job directly you assign the databases to run the job on. Running the job directly also means that you do not need to set a date and time in a schedule.

4. The job manager supports the following types of jobs:

- SQL-only script
- DB2 CLP script
- Executable/Shell script

5. C. Not specify the user ID and password that will run the job for the databases. The user ID and password is specified in the database connection for each database.

Chapter 7

1. After a connection is set up, to enable the connection for query tuning, you need to:

    A. Switch to the IBM Query Tuner perspective.

    B. Connect to the database.

    C. Right click on the database name and select *Analyze and Tune > Configure for Tuning.*

    D. Configure the connection to create explain tables, either with the *Guided Configuration* or *Advanced Configuration and Privilege Management.*

2. The no-charged query tuner features with IBM Data Studio 3.1 are:

    A. Query formatter

    B. Access plan graph and visual explain

    C. Statistics advisor

    D. Query tuner report

3. The query tuner can be started from:

    A. The SQL and XQuery Editor

    B. A drop-down menu from the database connection in the Data Source Explorer view

4. Query tuning features can be tailored using:

    A. Global preferences

    B. Advisor options

5. Analysis results are stored in a query tuner project in the Project Explorer.

6. D

7. B

8. A

Chapter 8

1. You forgot to first Deploy the stored procedure with the *Enable debugging* option checked.

2. The default schema, which is administrator ID (such as db2admin).

3. SQL Results

4. Variable

5. Breakpoints

6. B

7. D

8. C

9. A

10. A

The answer to the exercise is that the line `SET p_in = 2;` should be `SET p_out = 2`

Chapter 9

1. Some advantages of using UDFs include: 1) Encapsulate reusable code and 2) extend the SQL language with user-defined logic.

2. Scalar UDFs return a scalar value as a result. Table UDFs return a relational table as a result.

3. One in which the result of the routine needs to be joined with an existing table

4. To encapsulate commonly used logic.

5.   UDF that receives multiple values as input and returns a scalar value as a result.

6. B

7. C

8. C

9. B

10. C

Chapter 10

1. HTTPGET, HTTPPOST, SOAP

2. The Data Web Service has been modified but not yet deployed.

3. Source view.

4. Named parameter markers

5. Bottom up

6. B.

7. D One in which the result of the routine needs to be joined with an existing table

8. C

9. C.

10. B.

Chapter 11

1. The five phases of the data lifecycle are: design, develop, deploy, operate, optimize and govern. Governance is the aspect that needs to be considered across all phases of the lifecycle.

2. The design phase is most is most concerned with translating business requirements into a physical database representation? The main product for doing this is InfoSphere Data Architect, perhaps in conjunction with other modeling tools such as Rational Software Architect for WebSphere Software.

3. IBM Data Studio and InfoSphere Optim pureQuery Runtime can be used together to create a high performance data access layer. You can develop with pureQuery on your own computer with Data Studio. To deploy a pureQuery application on another computer, you need to acquire InfoSphere Optim pureQuery Runtime. .

4. InfoSphere Optim Performance manager is designed to find performance problems before they become serious issues.

5. InfoSphere Optim Query Workload Tuner extends the basic query tuning capabilities in Data Studio with additional tools and advisors to help SQL developers and DBAs improve the performance of their queries.

6. DB2 Merge Backup minimize downtime by making it easier to create full backups (by merging incremental and delta backups), which can shorten downtime when it is necessary to recover the database.

7. The answer is C. Although metadata exchange is a key implementation approach to integrated tools, it is not a goal.

8. The answer is B. The integration with Data Studio and InfoSphere Optim Performance Manager enables you to see actual performance improvements and you develop and tune queries. InfoSphere Optim Query Workload Tuner provides you with the tools and advice to tune a single query or a set of related queries.

9. The answer is A, B, and D.

10. The answer is D. Although Java data access should be developed with efficiency and performance in mind, the optimize phase of the lifecycle generally reflects activities around optimizing existing applications and resources.

B

Appendix B – Advanced integration features for Data Studio web console

This appendix is included for those who would like to use the advanced integration features of Data Studio web console, such as embedding Data Studio web console in the Data Studio full client, using a repository database to store configuration data, enable multi-user configuration and privileges requirements for web console actions, and sharing database connections between Data Studio client and Data Studio web console.

B.1 Integrating Data Studio web console with Data Studio full client

By integrating the web client with the full client you can access the Data Studio web console health monitoring and job management features without leaving the Data Studio client environment. You can use the health pages of the web console to view alerts, applications, utilities, storage, and related information. You can also use the embedded job manager pages to create and manage script-based jobs on your databases, as well as schedule scripts as jobs directly from the SQL script editor.

To embed Data Studio web console you must install the product and point Data Studio full client to the web console URL. See *Chapter 1*

1. In Data Studio client, select *Window > Preferences* and then go to *Data Management > Data Studio Web Console* to configure the connection.

2. Enter the following information in the Preferences window, as shown in *Figure B.1*:

   Data Studio web console URL – This is the URL that you use to connect to the web console. The URL is of the form `http://<server>:<port>/datatools`, where <server> is the name or IP address of the computer on which you installed Data Studio web console, and <port> is the http or https port that you specified when you installed the product, see *Chapter 1*.

   User name – Enter the name of a user that has login rights on the web console. If you have not configured Data Studio web console for multi-user login you must log in as the default administrative user that you created when you installed the product.

   Password – Enter the password of the user that you specified.

Choose how to open the web console. By default, the web console opens embedded in the workbench. You can select to open the web console in an external browser instead.

> **Note:**
> If you choose to open the web console embedded in the Data Studio client, the web console will not have all the features that the web console opened in a web browser has. Only the job manager interface and health monitoring pages are included in the embedded interface. In addition, the Task Launcher and Open menu are not included in the embedded web console. To get the full featured web console interface you must open the web console in an external browser. However, these extra configuration tasks are normally not needed for the day to day use of web console.
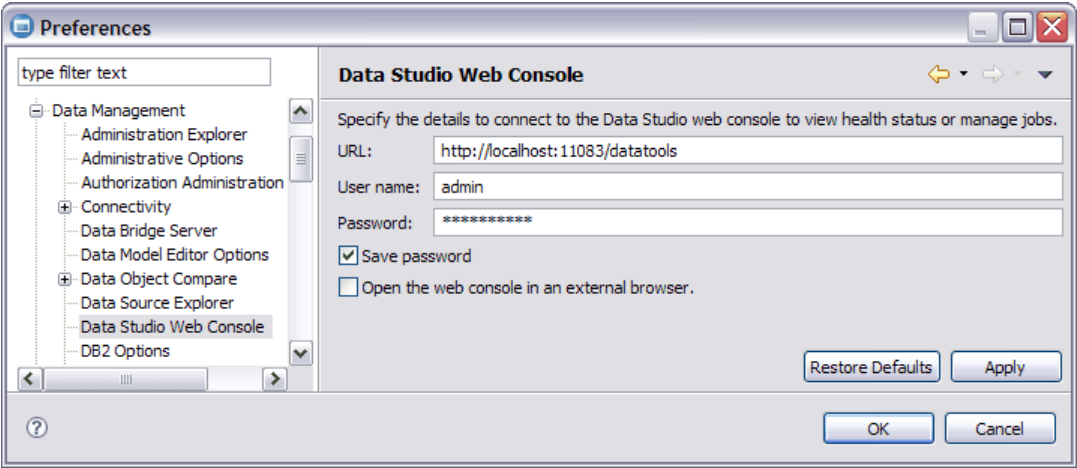
Figure B.1 – Configure Data Studio full client for Data Studio web console

3. Open the Data Studio web console. From within the workbench, you can open the Data Studio web console from the Administration Explorer, the Task Launcher, or the SQL and XQuery Editor.

User interface in the workbench	How to open the Data Studio web console
Administration Explorer	Right-click a database and select Monitor. Then select the monitoring options, such as Health Summary or Current Applications.
Task Launcher	Select the Monitor tab. Then select one of the monitoring tasks, such as View a health summary or View alerts list.
SQL script editor	Click ⏱ in the menu bar to open the job manager and to schedule the script that you are editing as a job.

Table 1 – Opening the web console from the Data Studio client

4. If prompted, enter the Data Studio web console URL and login information. The Data Studio web console opens on the selected page.

B.2 Using a repository database to store configuration data

When you install Data Studio web console, all user information and configuration settings such as database connections, alert settings, and job management settings are stored locally in system files.

By storing this data locally you can quickly get up and running with Data Studio web console in a test environment with only one user. If you plan to use Data Studio web console in a production environment, or if you plan to share the web console with other users, you need to set up a repository database.

Using a repository database provides significant advantages over storing the information locally:

▪   A repository database can be accessed by multiple servers to allow for clustered environments and scalability.

▪   A repository database provides tools to help maintain transactional integrity, back up and restore data, roll back data to a consistent state, replicate data, and more.

▪   A repository database provides access control through database privileges.

To set up Data Studio web console to use a repository database:

1. Create a DB2 for Linux, UNIX, and Windows database to use as the repository database. For information, see *Chapter 2.2.1 Creating a new database*.

2. Log in to Data Studio web console using the default administrative user ID.

3. If you have added database connections that you want to keep, you must manually export these, and then import them to the repository database. To do this, in the web console, click *Databases*, and then click *Export All* to export the existing database connections to a text file on your computer. Note that other settings, such as alert thresholds, notifications, and so on, are not retained when you move to a repository database. To retain these settings you must manually reconfigure these.

4. Set up the repository database:
    In the web console, select *Open > Setup > Configuration Repository* and then click *Select Repository Database* to select the relational database that you created in step 1 as a repository for the Data Studio web console data. Note that the user ID that you enter to connect to the repository database must have the authority to create new tables and schemas on the database. Test the connection to the database and then click OK to configure the repository

database by having Data Studio web console install the required schemas.

Figure B.2 – Configuring the repository database

5.   Import any existing database connections that you exported to a text file.
In the web console, click *Databases*, and then click *Import* to import the
database connections from the text file that you saved on your computer.

The Data Studio web console is now using the repository database to store database
connections, alert settings. You can now configure the Data Studio web console to run in
multi-user mode by configuring the product to use repository database authentication and
granting the users of the repository database access to the web console.

> **Note:**
>
> The repository database is not listed among the other database connections in the
> *Databases* page. You can only connect to one repository database at a time. To see the
> settings for the current repository database, select *Open > Setup > Configuration
> Repository* and then click *Select Repository Database*.

B.3 Enabling console security and managing privileges in the web console

If only one person will log in to Data Studio web console, you can continue using the
default administrative user that you have used to log in to the web console, but if more than
one user will share the web console and you want each user to have a unique log in user
ID you must set up the web console for multi user mode.

When you install Data Studio web console you enter the credentials for a default administrative user that is used for login to the server. However, for day-to-day use in a production environment where you should set up Console Security so that you can grant appropriate privileges to users, groups, and roles that are defined on the repository database.

You grant access to the repository database users in three steps:

1. Configure the web console for repository database authentication and grant the users of the repository database web console access. You will also give them web console roles such as Administrator or Viewer depending on if they will perform other web console administrative tasks or just view the information in the web console.

2. Grant the web console users database privileges to perform tasks such as setting alert thresholds on each of the connected databases.

3. Grant repository database privileges to web console users to perform tasks such as managing jobs for the connected databases.

B.3.1 Configure the web console for repository database authentication

The Data Studio web console comes with two main authentication mechanisms:

- **Default administrative user login**
  This is the install-time default security mechanism. User login is limited to the default administrative user ID that was created when you installed the product. Administrators can use this option to temporarily restrict user login without stopping the product.

- **Repository database authentication.**
  This is the multi-user security mechanism. Any user who can connect to the repository database can be given access to the web console. The user's credentials are authenticated directly against this database.

To use the repository database authentication mechanism:

1. Set up a repository database. For information, see *B.2 Using a repository database to store configuration data*.

2. In the web console, select *Open > Setup > Console Security.*
    Select *Repository database authentication* and click *Apply*. If prompted, log in with a database administrator user ID and password for the repository database.
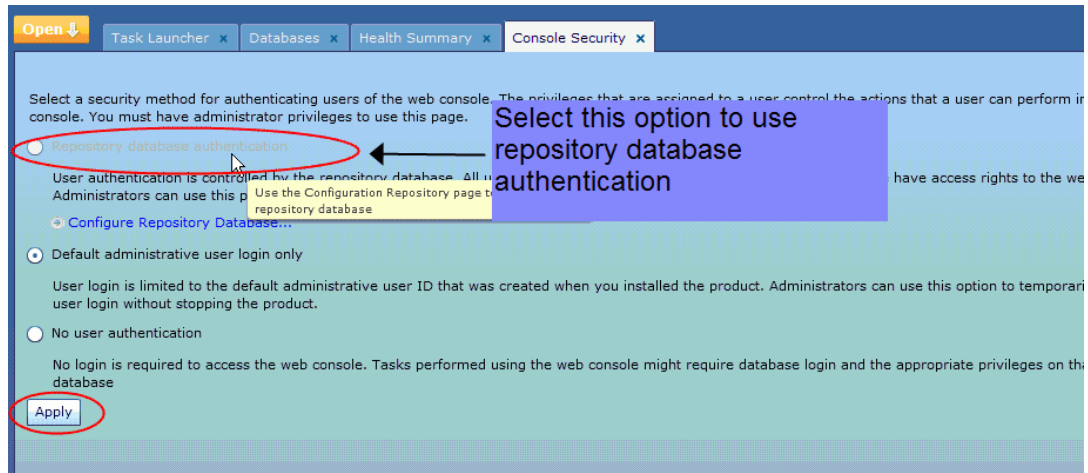
Figure B.3 – Selecting repository database authentication for the web console

3. At the bottom of the page you can now grant web console access rights to the
users of the repository database. Click *Grant*, and then type in the ID and select
the type for an existing user, group, or role.
You can then select the type of privilege to grant the ID on the web console:

   Administrator
   A user with the Administrator role can do any task in the web console, including
   setup tasks, such as configuring logs, managing privileges, and adding database
   connections.

   Operator
   This role is not used with Data Studio web console

   Viewer
   A user with the Viewer role has limited rights on the web console. The user can
   view all pages, but cannot add database connections, configure logs, or manage
   privileges.

Note:

The console security page only lets you configure access to the web console for users that
already exist on the repository database. To add users, groups, and roles to that database
you must have the appropriate access to the database. For information on how to add new
users to the repository database, see *Chapter 2.5.1 Creating users*.

B.3.2 Granting privileges to users of the web console

When you have granted web console access to the users, groups, and roles that need
access, you can configure **Can Do** privileges to perform tasks using the web console
individually for each user, group, or role of the repository database. The privileges are set
on either the database on which a task such as health monitoring is done, or on the

repository database for tasks that require modifying data on the repository database, such as job management.

The Can Do privileges on the connected database are required by default, and the Can Do privileges on the repository database are not required. For example, only users with the *Can Manage Alerts* privilege can modify the thresholds on the Health Alerts Configuration page, but any user can schedule a job on a database. To configure the privilege requirements for the web console, see *B.3.2.3 Enable and disable privilege requirements*.

B.3.2.1 Grant database privileges to the web console users

The Data Studio web console server typically monitors multiple databases from different organizational units in an enterprise, so it is important that the users and privileges boundaries defined in these databases are respected.

Data Studio web console lets you control the following privileges for users on each database:

Can Monitor
This privilege is not used by Data Studio web console.

Can Manage Alerts
This privilege gives the user the right to set the alert thresholds and enable and disable alert monitoring for a database. By default, this privilege is required.

To configure the web console to not require this privilege, see *B.3.2.3 Enable and disable privilege requirements*.

For example, there are two databases called *SALES* and *PAYROLL* defined in a system. Just because the DBA for *PAYROLL* is able to log in to the web console doesn't mean that she should have the ability to modify alert settings for the *SALES* database.

However, if a DBA for *SALES* would like to enable other DBAs to edit alert configurations for *SALES*, he can grant the Can Manage Alerts privilege to another DBA using the *Manage Privileges* page under *Open>Product Setup,* as shown in *Figure B.4* below.
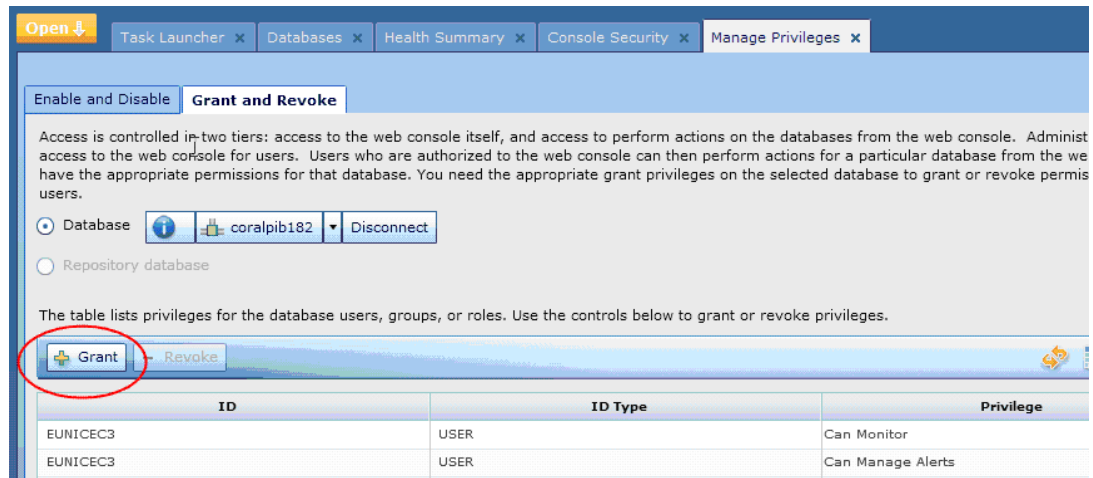
Figure B.4 – Granting user privileges on a database

B.3.2.2 Grant repository database privileges to the web console users

Two sets of privileges are set on the repository database, and apply to all connected databases. Both of these privileges handle job management:

Can Manage Jobs
Any user with web console access can create and schedule jobs. By default, this privilege is not required, and all web console users can manage jobs.

Can Run As Default User
When multiple databases are targets for a scheduled job the job will be run as the default user ID that is stored with the database connection for each database. If the privilege is enabled, the user that schedules the job must have the can Run As Default User privilege. By default, this privilege is not required, and all web console users can run jobs as the default user.

To configure the web console to require these privileges, see the next section. .

B.3.2.3 Enable and disable privilege requirements

Depending on your environment, you might want to be more or less strict in limiting the tasks that your users can perform. For example, you might want to allow all web console users to be able to configure alerts on a database, but at the same time only allow subset of your users to schedule jobs.

Data Studio web console lets you disable the privileges requirements for your databases and for the repository database.

For example, to allow all users of the web console to configure alerts on a database, you can disable the Can Manage Alerts privilege requirement under the Enable and Disable tab on the Manage Privileges page. See *Figure B.5*.
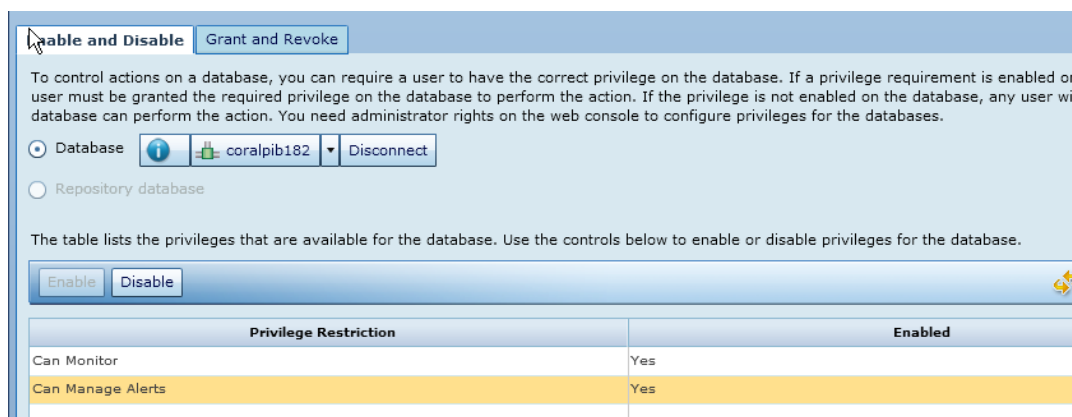
Figure B.5 – Enable and disable privileges requirements

B.4 Sharing database connections between Data Studio client and Data Studio web console

If you plan to connect more than one instance of Data Studio full client to the same Data Studio web console server it is advantageous to synchronize the database connection profiles on the clients to the database connections that are stored on the Data Studio web console. In this way the web console acts as a central repository for the database connections and is accessible to all client users that have configured their clients to access Data Studio web console for health monitoring or job management.

The database connection sharing is two-way. Just as the Data Studio client user can import existing database connections from the web console, that user can also automatically add existing database connection profiles to the web console by invoking the Current Application Connections, Current Table Spaces, and Current Utilities pages for the selected database. You can also manually add your database connections directly in the Data Studio web console directly from the *Open > Databases* page.

To synchronize the database connection profiles between two clients using the Data Studio web console as a database connection repository:

1. From the first Data Studio full client, in the Administration Explorer, select the database whose connection you want to add to the web console.

2. Right-click the database and select any monitoring task from the menu. You can select any one of these tasks:

    -   Application Connections

    -   Table Spaces

    -   Utilities

    Data Studio web console opens on the page that corresponds to the selected task and you are prompted for the login credentials to connect to the database.

3. Supply the login credentials.
    If the selected database does not exist in the Data Studio web console list of
    database connections, a new database connection for that database is added.

4. In Data Studio web console, select *Open* > *Databases* to see a list of the Data
    Studio web console database connections and to verify that the database was
    successfully added.

5. In the second Data Studio client, verify that your client is configured to work with
    the Data Studio web console. Select *Window* > *Preferences* and then go to *Data
    Management* > *Data Studio Web Console* to verify the configuration.

6. To import the shared database connections from the web console, from the
    Administration Explorer, click ⬛.

7. In the Import Connection Profiles wizard, verify that the URL for the Data Studio
    web console server is the same that you used when you configured Data Studio
    full client to connect to Data Studio web console, then click *OK*.
    The database connections that are defined in the Data Studio web console are
    imported to the Data Studio client, and are listed in the Database Connections
    folder in the Administration Explorer view.

For more information about managing database connections in Data Studio, see the
developerWorks article entitled *Managing database connections with the IBM Data Studio
web console* at http://www.ibm.com/developerworks/data/library/techarticle/dm-
1111datastudiowebconsole/index.html

C

Appendix C – Installing the Data Studio administration client

This Appendix is included for those people who would like to use the Data Studio administration client. As described in *Chapter 1*, the administration client is designed for DBAs who have no need for Java, Web services, or XML development capabilities and who like the smaller footprint provided by this package.

C.1 Before you begin

Please remember to read the system requirements before you download. It references important information like Java Runtime versions, Linux download tips, etc. For example, for machines that have never run Java, you will need a JRE of 1.6 or higher for the installer, which is InstallAnywhere (ISMP)-based. This allows the installer to launch.

Also, you can check out the discussion forum if you have questions.

To download the Data Studio administration client, find the link here:

http://www.ibm.com/developerworks/downloads/im/data/

After you register, you will see the page that lets you choose your platform – Windows or Linux, as shown in *Figure C.1.*

IBM Data Studio administration client

Downloads

To properly configure your download, please review the information below. Select the appropriate offering. When you are done, press the "Continue" button at the bottom.

Offering	Platform	Format
○ **IBM Data Studio administration client** Version 3.1 Languages: Chinese Simplified, Chinese Traditional, Czech, English, French, German, Hungarian, Italian, Japanese, Korean, Polish, Portuguese Brazilian, Russian, Spanish	Red Hat Linux SUSE Linux Enterprise Desktop(SLED) SUSE Linux Enterprise Server (SLES)	download
○ **IBM Data Studio administration client** Version 3.1 Languages: Chinese Simplified, Chinese Traditional, Czech, English, French, German, Hungarian, Italian, Japanese, Korean, Polish, Portuguese Brazilian, Russian, Spanish	Windows 7 Enterprise Windows 7 Professional Windows 7 Ultimate Windows Vista Business Windows Vista Enterprise Windows Vista Ultimate Windows XP Professional	download

Figure C.1 – Choose your platform from the download site

Choose your platform, register, accept the license, and download the package.

C.2 Installation procedure (assumes Windows)

1. After you unzip the package, double click on
   `ibm_data_studio_admin_client_v31_windows\install.exe` The
   screen shown in *Figure C.2* appears. Accept the license and click *Next*.
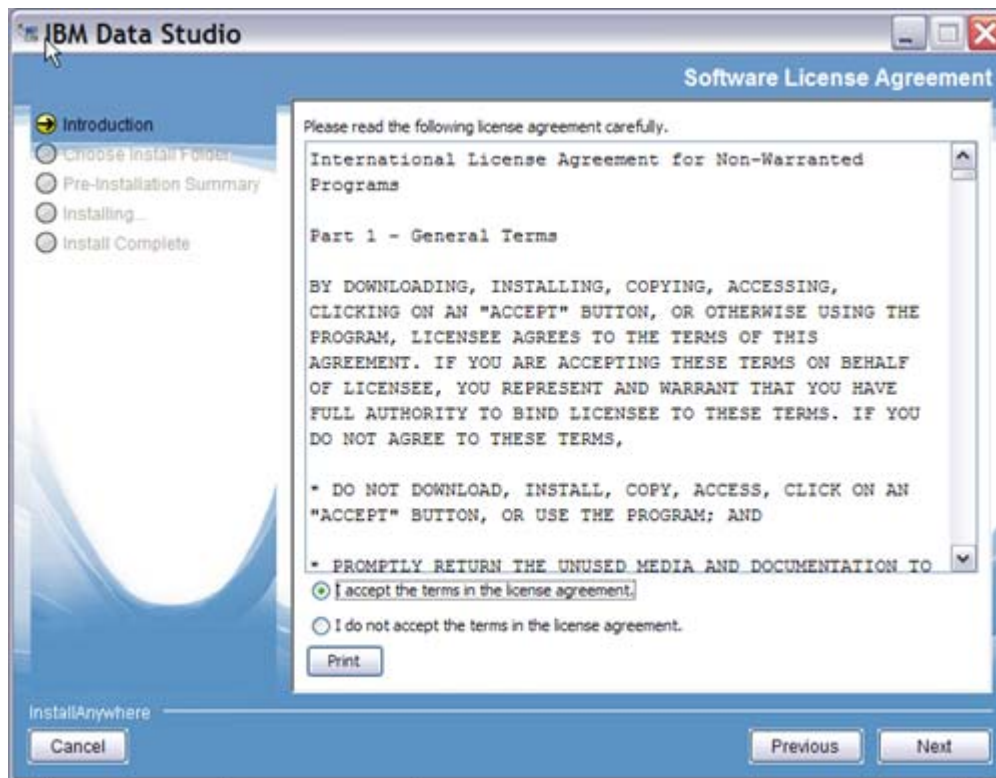
Figure C.2 – Accept the license agreement

2. Choose the installation directory or accept the default `C:Program Files\IBM\DSAC3.1` as shown in *Figure C.3* and click *Next*.
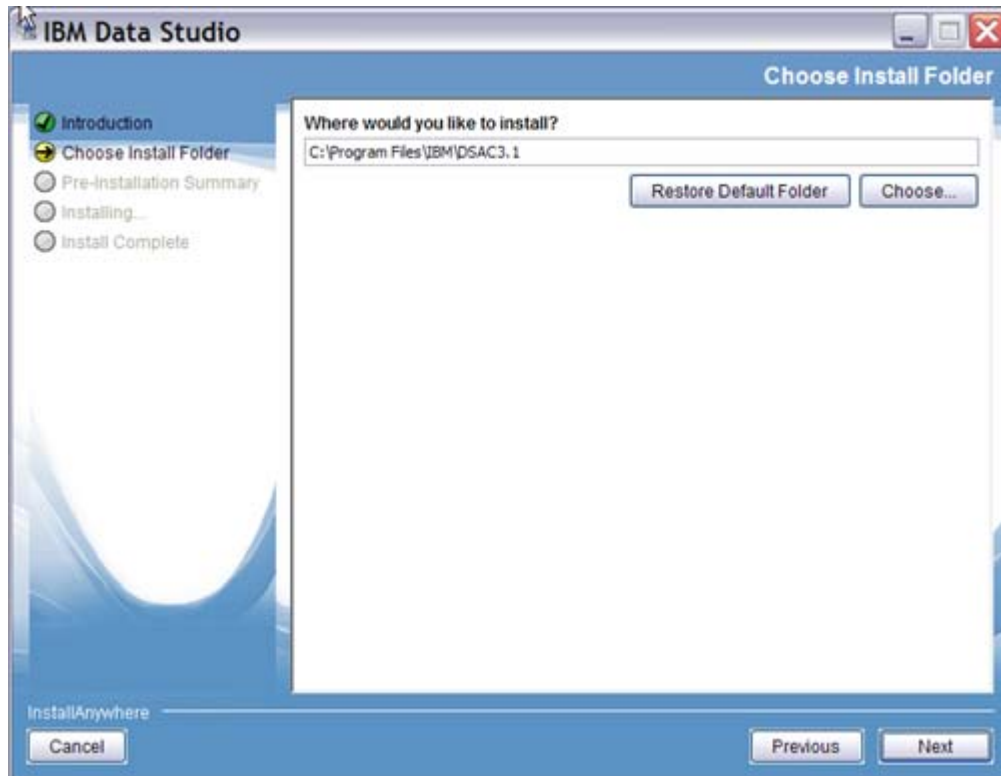
Figure C.3 – Choose an installation directory

3. If all goes well, you will see the screen shown in *Figure C.4* below, and you simply need to click *Done* to start Data Studio. (If you would rather start Data Studio later from the Start menu, simply uncheck the *Start IBM Data Studio* box.)
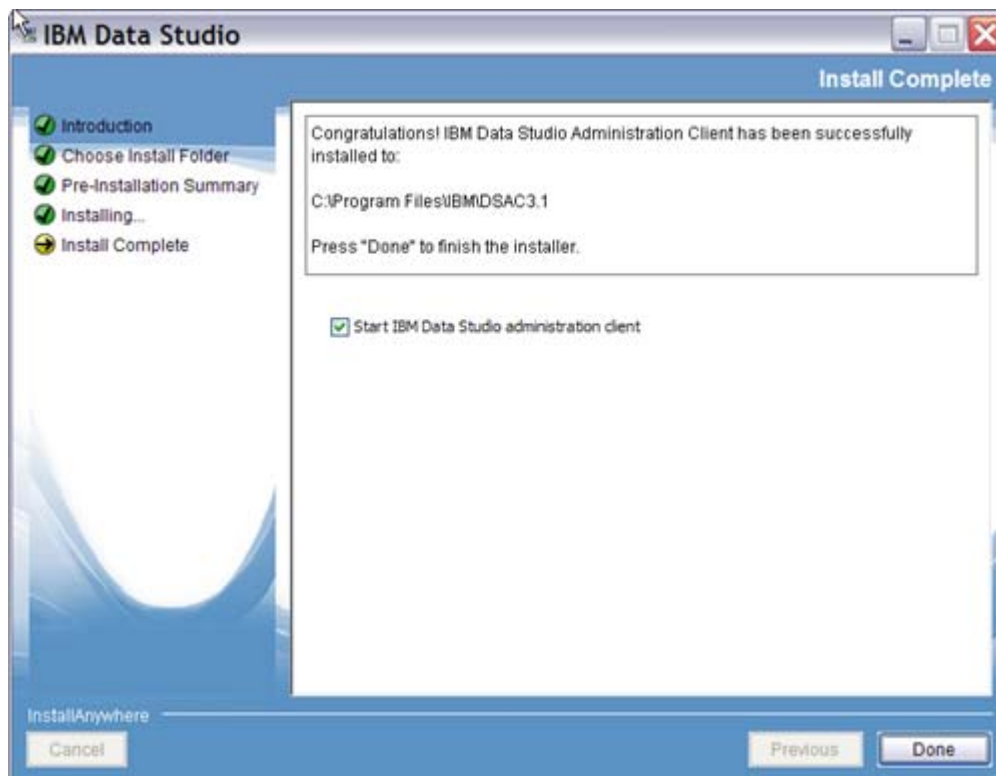
Figure C.4 – Choose your platform from the download site

    4.   The Task Launcher is shown below in *Figure C.5*.
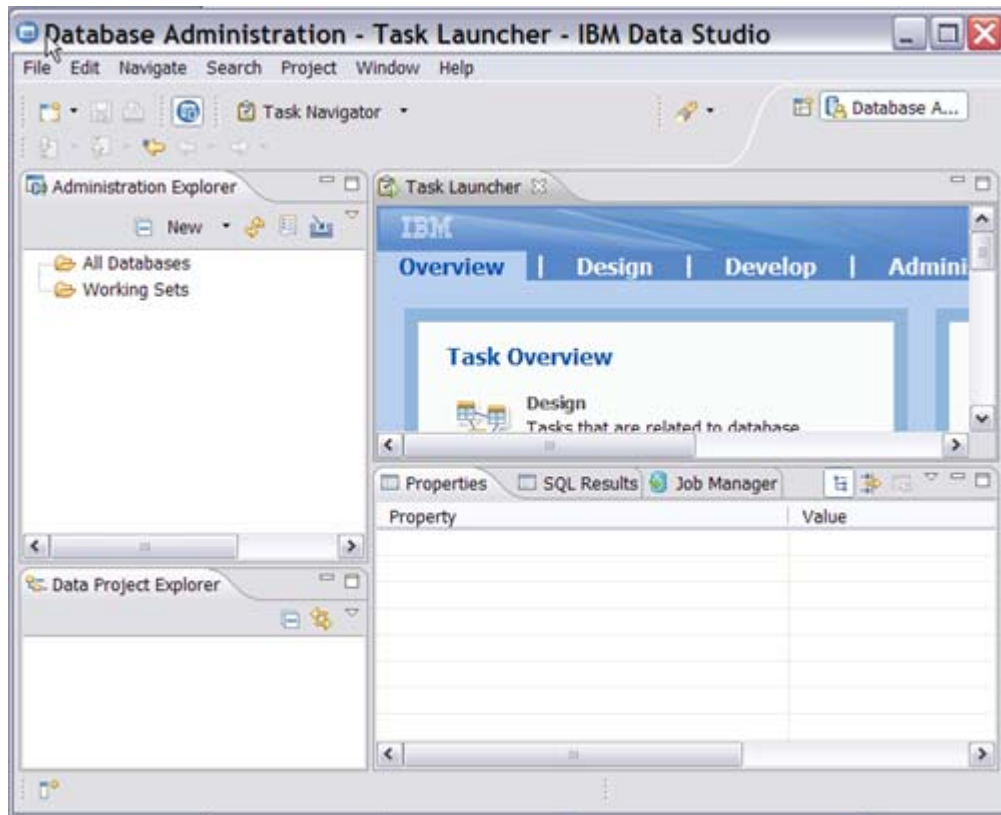
Figure C.5 – Welcome screen for Data Studio administration client

You are immediately launched into the Database Administration perspective in a default Workspace as shown in *Figure C.6*. (Note: You can use *File->Switch Workspace* if you have an existing project and workspace you want to use.)
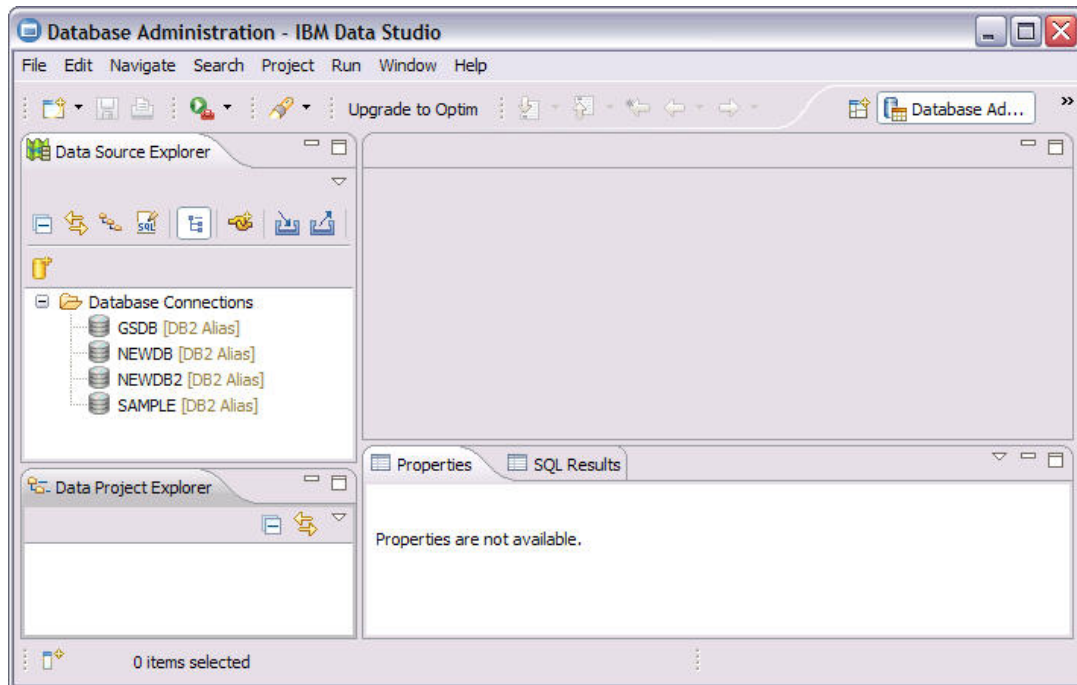
Figure C.6 – Default workspace for Data Studio administration client

Congratulations, you've successfully installed Data Studio stand-alone and are ready to get to work!

D

Appendix D – The Sample Outdoor Company

The Sample Outdoor Company is a fictional company used to help illustrate real-world scenarios and examples for product documentation, product demos, and technical articles. The sample database for the Samples Outdoor Company is used to illustrate many different use cases, including data warehousing use cases. This book uses only a subset of that database.

This appendix provides an overview of the schemas and tables that are used in many of the examples and exercises used in this book.

Note:

The sample database can be downloaded from the Data Studio Information Center at: http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/topic/com.ibm.sampledata.go.doc/topics/download.html

D.1 Sample Outdoors database data model (partial)

Figure D.1 shows the relationship among the tables used in the examples in this book.

GOSALES.PRODUCT_TYPE

PRODUCT_TYPE_CODE

PRODUCT_LINE_CODE [FK]
PRODUCT_TYPE_EN
PRODUCT_TYPE_FR
PRODUCT_TYPE_DE
PRODUCT_TYPE_NL
PRODUCT_TYPE_JA
PRODUCT_TYPE_SC
PRODUCT_TYPE_KO
PRODUCT_TYPE_CS
PRODUCT_TYPE_HU
PRODUCT_TYPE_PT
PRODUCT_TYPE_TC
PRODUCT_TYPE_DA
PRODUCT_TYPE_EL
PRODUCT_TYPE_FI
PRODUCT_TYPE_IT
PRODUCT_TYPE_NO
PRODUCT_TYPE_PL
PRODUCT_TYPE_RU
PRODUCT_TYPE_SV
PRODUCT_TYPE_ES
PRODUCT_TYPE_TH
PRODUCT_TYPE_MS
PRODUCT_TYPE_ID

PRODUCT_SIZE_LOOKUP

PRODUCT_SIZE_CODE

PRODUCT_SIZE_EN
PRODUCT_SIZE_DE
PRODUCT_SIZE_FR
PRODUCT_SIZE_JA
PRODUCT_SIZE_CS
PRODUCT_SIZE_DA
PRODUCT_SIZE_EL
PRODUCT_SIZE_ES
PRODUCT_SIZE_FI
PRODUCT_SIZE_HU
PRODUCT_SIZE_ID
PRODUCT_SIZE_IT
PRODUCT_SIZE_KO
PRODUCT_SIZE_MS
PRODUCT_SIZE_NL
PRODUCT_SIZE_NO
PRODUCT_SIZE_PL
PRODUCT_SIZE_PT
PRODUCT_SIZE_RU
PRODUCT_SIZE_SC
PRODUCT_SIZE_SV
PRODUCT_SIZE_TC
PRODUCT_SIZE_TH

CUST_COUNTRY

COUNTRY_CODE [FK]

COUNTRY_EN
COUNTRY_CS
COUNTRY_DA
COUNTRY_DE
COUNTRY_EL
COUNTRY_ES
COUNTRY_FI
COUNTRY_FR
COUNTRY_HU
COUNTRY_ID
COUNTRY_IT
COUNTRY_JA
COUNTRY_KO
COUNTRY_MS
COUNTRY_NL
COUNTRY_NO
COUNTRY_PL
COUNTRY_PT
COUNTRY_RU
COUNTRY_SC
COUNTRY_SV
COUNTRY_TC
COUNTRY_TH
FLAG_IMAGE
SALES_REGION_CODE [FK]
ISO_THREE_LETTER_CODE
ISO_TWO_LETTER_CODE
ISO_THREE_DIGIT_CODE
EURO_IN_USE_SINCE

PRODUCT_COLOR_LOOKUP

PRODUCT_COLOR_CODE

PRODUCT_COLOR_EN
PRODUCT_COLOR_DE
PRODUCT_COLOR_FR
PRODUCT_COLOR_JA
PRODUCT_COLOR_CS
PRODUCT_COLOR_DA
PRODUCT_COLOR_EL
PRODUCT_COLOR_ES
PRODUCT_COLOR_FI
PRODUCT_COLOR_HU
PRODUCT_COLOR_ID
PRODUCT_COLOR_IT
PRODUCT_COLOR_KO
PRODUCT_COLOR_MS
PRODUCT_COLOR_NL
PRODUCT_COLOR_NO
PRODUCT_COLOR_PL
PRODUCT_COLOR_PT
PRODUCT_COLOR_RU
PRODUCT_COLOR_SC
PRODUCT_COLOR_SV
PRODUCT_COLOR_TC
PRODUCT_COLOR_TH

GOSALES.PRODUCT

PRODUCT_NUMBER

BASE_PRODUCT_NUMBER
INTRODUCTION_DATE
DISCONTINUED_DATE
PRODUCT_TYPE_CODE [FK]
PRODUCT_COLOR_CODE [FK]
PRODUCT_SIZE_CODE [FK]
PRODUCT_BRAND_CODE [FK]
PRODUCTION_COST
GROSS_MARGIN
PRODUCT_IMAGE

PRODUCT_LINE

PRODUCT_LINE_CODE

PRODUCT_LINE_EN
PRODUCT_LINE_FR
PRODUCT_LINE_DE
PRODUCT_LINE_NL
PRODUCT_LINE_SC
PRODUCT_LINE_KO
PRODUCT_LINE_JA
PRODUCT_LINE_CS
PRODUCT_LINE_HU
PRODUCT_LINE_PT
PRODUCT_LINE_TC
PRODUCT_LINE_DA
PRODUCT_LINE_EL
PRODUCT_LINE_FI
PRODUCT_LINE_IT
PRODUCT_LINE_NO
PRODUCT_LINE_PL
PRODUCT_LINE_RU
PRODUCT_LINE_SV
PRODUCT_LINE_ES
PRODUCT_LINE_TH
PRODUCT_LINE_MS
PRODUCT_LINE_ID

CUST_CRDT_CHECK

CUST_CODE [FK]

CUST_SSN
CUST_CRDT_SCORE

GOSALESHR.RANKING

RANKING_CODE

RANKING_DESCRIPTION_EN
RANKING_DESCRIPTION_CS
RANKING_DESCRIPTION_DA
RANKING_DESCRIPTION_NL
RANKING_DESCRIPTION_FI
RANKING_DESCRIPTION_FR
RANKING_DESCRIPTION_DE
RANKING_DESCRIPTION_EL
RANKING_DESCRIPTION_HU
RANKING_DESCRIPTION_IT
RANKING_DESCRIPTION_JA
RANKING_DESCRIPTION_KO
RANKING_DESCRIPTION_NO
RANKING_DESCRIPTION_PL
RANKING_DESCRIPTION_PT
RANKING_DESCRIPTION_RU
RANKING_DESCRIPTION_SC
RANKING_DESCRIPTION_SV
RANKING_DESCRIPTION_TC
RANKING_DESCRIPTION_ES
RANKING_DESCRIPTION_TH
RANKING_DESCRIPTION_MS
RANKING_DESCRIPTION_ID

GOSALES.PRODUCT_BRAND

PRODUCT_BRAND_CODE

PRODUCT_BRAND_EN
PRODUCT_BRAND_DE
PRODUCT_BRAND_FR
PRODUCT_BRAND_JA
PRODUCT_BRAND_CS
PRODUCT_BRAND_DA
PRODUCT_BRAND_EL
PRODUCT_BRAND_ES
PRODUCT_BRAND_FI
PRODUCT_BRAND_HU
PRODUCT_BRAND_ID
PRODUCT_BRAND_IT
PRODUCT_BRAND_KO
PRODUCT_BRAND_MS
PRODUCT_BRAND_NL
PRODUCT_BRAND_NO
PRODUCT_BRAND_PL
PRODUCT_BRAND_PT
PRODUCT_BRAND_RU
PRODUCT_BRAND_SC
PRODUCT_BRAND_SV
PRODUCT_BRAND_TC
PRODUCT_BRAND_TH

GOSALESCT.CUST_CUSTOMER

CUST_CODE [FK]

CUST_FIRST_NAME
CUST_FIRST_NAME_MB
CUST_LAST_NAME
CUST_LAST_NAME_MB
CUST_ADDRESS1
CUST_ADDRESS1_MB
CUST_ADDRESS2
CUST_ADDRESS2_MB
CUST_CITY
CUST_CITY_MB
CUST_PROV_STATE
CUST_PROV_STATE_MB
CUST_PROV_STATE_CODE [FK]
CUST_POSTAL_ZONE
CUST_COUNTRY_CODE [FK]
CUST_PHONE_NUMBER
CUST_EMAIL
CUST_INFO
GENDER_CODE
CUST_AGE
MARITAL_STATUS_CODE
CUST_PROFESSION_CODE

GOSALESHR.RANKING_RESULTS

RANKING_DATE
EMPLOYEE_CODE [FK]
RANKING_CODE [FK]

RANKING_YEAR

GOSALES.PRODUCT_NAME_LOOKUP

PRODUCT_NUMBER [FK]
PRODUCT_LANGUAGE

PRODUCT_NAME
PRODUCT_DESCRIPTION

EMPLOYEE

EMPLOYEE_CODE

FIRST_NAME
FIRST_NAME_MB
LAST_NAME
LAST_NAME_MB
DATE_HIRED
TERMINATION_DATE
TERMINATION_CODE [FK]
BIRTH_DATE
GENDER_CODE [FK]
WORK_PHONE
EXTENSION
FAX
EMAIL

BRANCH

BRANCH_CODE

ADDRESS1
ADDRESS1_MB
ADDRESS2
ADDRESS2_MB
CITY
CITY_MB
PROV_STATE
PROV_STATE_MB
POSTAL_ZONE
COUNTRY_CODE [FK]
ORGANIZATION_CODE [FK]
WAREHOUSE_BRANCH_CODE

GOSALES.INVENTORY_LEVELS

INVENTORY_YEAR
INVENTORY_MONTH
WAREHOUSE_BRANCH_CODE [FK]
PRODUCT_NUMBER [FK]

OPENING_INVENTORY
QUANTITY_SHIPPED
ADDITIONS
UNIT_COST
CLOSING_INVENTORY
AVERAGE_UNIT_COST

GO_SALES_TAX

CUST_COUNTRY_CODE
CUST_PROV_STATE_CODE

SALES_TAX_RATE

Figure D.1- Sample Outdoors data model

D.2 Table descriptions

D.2.1 GOSALES schema

The GOSALES schema includes information about products and inventory.

D.2.1.1 GOSALES.BRANCH table

Row count: 29

The BRANCH table contains address information of each branch. Each branch has a collection of employees with different roles, including sales representatives operating from a regional base.

Not all branches have warehouses. The warehouse branch code is a repeating value of the branch code, identifying the regions covered by a particular warehouse.

D.2.1.2 GOSALES.INVENTORY_LEVELS table

Row count: 53730

This table shows inventory for all warehouses. Only 11 of the 29 branches have warehouses that maintain inventory.

D.2.1.3 GOSALES.PRODUCT table

Row count: 274

The company supplies sport gear for camping, climbing, and golfing. There are five product lines, further subdivided into 21 product types. There are a total of 144 unique products, or 274 products when including color and size.

D.2.1.4 GOSALES.PRODUCT_BRAND table

Row count: 28

Products of the same brand are associated by a style or price point.

D.2.1.5 GOSALES.PRODUCT_COLOR_LOOKUP table

Row count: 27

Product colors provide analysis by attribute. GO Accessories is the richest data source for attribute analysis including color and size.

D.2.1.6 GOSALES.PRODUCT_LINE table

Row count: 5

There are five product lines, with each covering a different aspect of outdoor activity. Each line is further subdivided into product types and products:

Camping Equipment

Mountaineering Equipment

Personal Accessories

Outdoor Protection

Golf Equipment

D.2.1.7 GOSALES.PRODUCT_NAME_LOOKUP table

Row count: 6302

This lookup table contains the name of each product.

D.2.1.8 GOSALES.PRODUCT_SIZE_LOOKUP table

Row count: 55

Product sizes provide analysis by attribute. The GO Accessories company is the richest data source for attribute analysis including color and size.

D.2.1.9 GOSALES.PRODUCT_TYPE table

Row count: 21

Each product line has a set of product types that define a functional area for outdoor equipment. The product type lookup table contains the names of 21 product types.

D.2.2 GOSALESCT schema

The GOSALESCT schema contains customer information.

D.2.2.1 GOSALESCT.CUST_COUNTRY table

 Row count: 23

This table defines the geography for the online sales channel to consumers. The addition of Russia and India make it different from the country table in the GOSALES schema. There are no sales regions for India or Russia.

D.2.2.2 GOSALESCT.CUST_CRDT_CHECK table

 Row count: 900

The customer credit check table contains the credit scores of consumers that make online purchases.

D.2.2.3 GOSALESCT.CUST_CUSTOMER table

 Row count: 31255

The customer table contains the name, address, and contact information of each customer. All customers in this table are online shoppers paying the retail price for items sold by the company or one of its partners.

D.2.2.4 GOSALESCT.GO_SALES_TAX table

 Row count: 94

The Sample Outdoors sales tax table contains sales tax rates at a country level, or state level if applicable.

Tax rates are for example only.

D.2.3 GOSALESHR schema

The GOSALESHR schema includes information used by the Sample Outdoors company Human Resources department.

D.2.3.1 GOSALESHR.EMPLOYEE table

 Row count: 766

The employee table contains the static information that repeats for each detail in the employee history table.

D.2.3.2 GOSALESHR.RANKING table

 Row count: 5

The ranking dimension contains text descriptions of an employee's ranking. Ranking is done annually and is one of the following values:

Poor

Satisfactory

Good

Very good

Excellent

D.2.3.3 GOSALESHR.RANKING_RESULTS table

 Row count: 1898

This fact table maintains ranking data for each employee. Rankings are published in the month of March based on the previous year.

E

Appendix E – Advanced topics for developing Data Web Services

This appendix shows you how to take advantage of more capabilities with Data Web Services, including the following topics:

Consuming Web services – using different bindings

Simplifying access for single row results

Handling stored procedure result sets

Using XSL to transform input and output results

Understanding Data Web Services artifacts

Selecting a different SOAP engine framework

E.1 Testing additional Web service bindings

You may have clients that require a different binding in order to consume the Web service response and which are not supported for testing in the Web Services Explorer. In this section, we'll review a couple of basic items you need to understand for using and testing these additional bindings, including more detail on the location of the WSDL and the default message XML schema. Then we'll explain how to use each of the following bindings:

- **SOAP over HTTP**: This is the binding described in *Chapter 10.* It is used with WSDL-based clients like SOAP frameworks, Enterprise SOA environments, and with service registries such as the WebSphere Service Registry and Repository. We include it here for completeness.

- **Web Access: HTTP GET**: This is used for quick access from Web 2.0 clients and for direct access from Web browsers.

- **Web Access: HTTP POST URL-encoded:** Used with more "traditional" HTML, such as for submitting HTML forms.

- **Web Access: HTTP POST XML**: Web 2.0, used by AJAX clients and JavaScript frameworks using the asynchronous HTTPXMLRequest JavaScript method from a Web browser.

- **Web Access: HTTP POST JSON**: Web 2.0, provides a direct way to parse messages into JavaScript objects.

All service bindings are based on HTTP and, for demonstration purposes, we use **cURL** as a lightweight, simple to use HTTP client.

> **Note:**
>
> cURL is a command-line tool for transferring files with URL syntax. Using the cURL command line, a URL must be used to define where to get or send the file that is specified in the command line. cURL is free software that is distributed under the MIT License and supports several data transfer protocols. cURL compiles and runs under a wide variety of operating systems. cURL uses a portable library and programming interface named libcurl, which provides an interface to the most common Internet protocols, such as HTTP(s), FTP(s), LDAP, DICT, TELNET, and FILE.
>
> Consult and download all documentation and binaries from the cURL Website at the URL address:
>
> http://curl.haxx.se/

E.1.1 Default XML message schemas

To test the SOAP over HTTP and HTTP POST (XML) binding you need to know the structure (XML schema) of the request message. This information is contained in the WSDL file, but you can also separately generate the XML schema for every operation in the Web service, as follows:

1. Right-click on the Web service operation from within the Data Project Explorer and select *Manage XSLT…. Figure E.1* shows this for the `RankEmployee` operation used in *Chapter 10.*
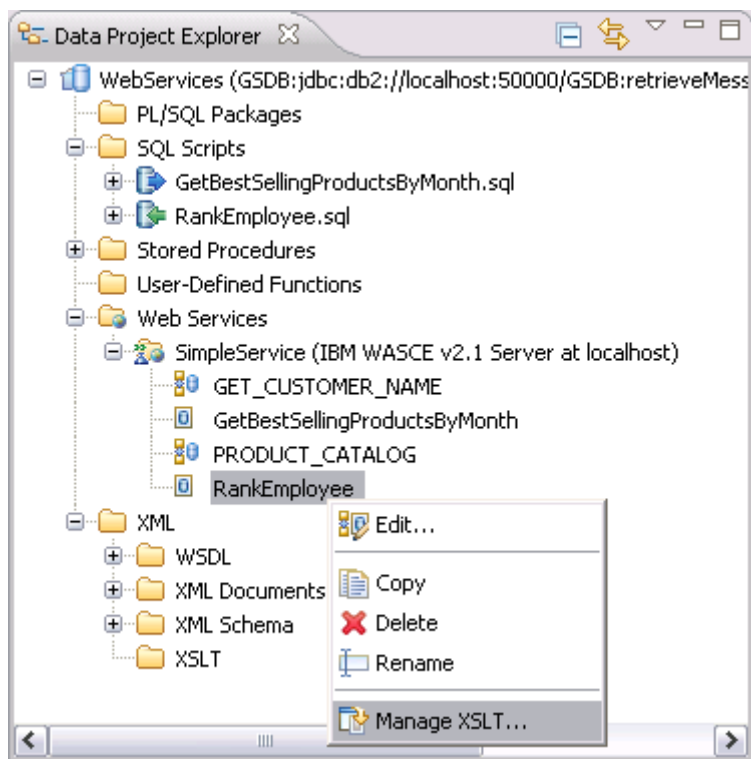
Figure E.1 – Selecting the Manage XSLT option

2. From the *Configure XSL Transformations* dialog, click on the *Generate Default* button. You will be asked for a location to store the XML schema file as shown in *Figure E.2*. Keep the default location, which points to your Data Development project folder. Keep the proposed name *SimpleService.RankEmployee.default.xsd.*
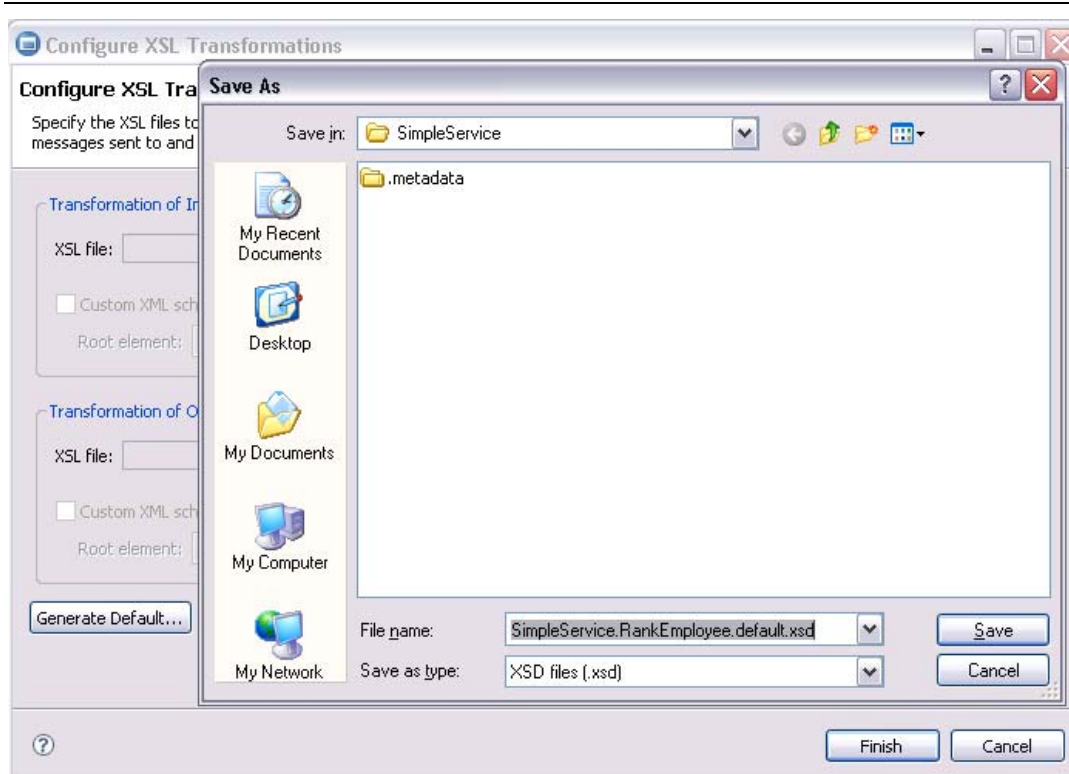
Figure E.2 – Saving the generated XML schema

3.   Click *Save.* Data Studio generates the XML schema for the selected operation. Exit the dialog and refresh the Data Development Project (right-clicking the project and selecting *Refresh…)*. Now a generated XSD file appears under the project's *XML -> XML Schema* folder. The XSD extension may not be displayed.

4.   Now you can use the Data Studio XML tools to create an XML instance document from the XML schema using the XML instance generator. Locate the generated XSD file in the *XML -> XML Schema* folder. Right-click the XSD file and select *Generate -> XML File …*

5.   From the *New XML File* dialog select a name and destination for the XML file instance. In *Figure E.3*, we select *SimpleService.RankEmployee.default.xml* as the file name, since we want to create the XML request message for the `RankEmployee` operation.

Figure E.3 – Selecting an XML file name and location

6. Click *Next.* In the next dialog shown in *Figure E.4*, you need to select the Root element for your XML message from the XML schema. In this case, there are two root elements available – *RankEmployee* and *RankEmployeeResponse.* Select RankEmployee as the root element name, since this represents the element for the

request message. Click *Finish*.
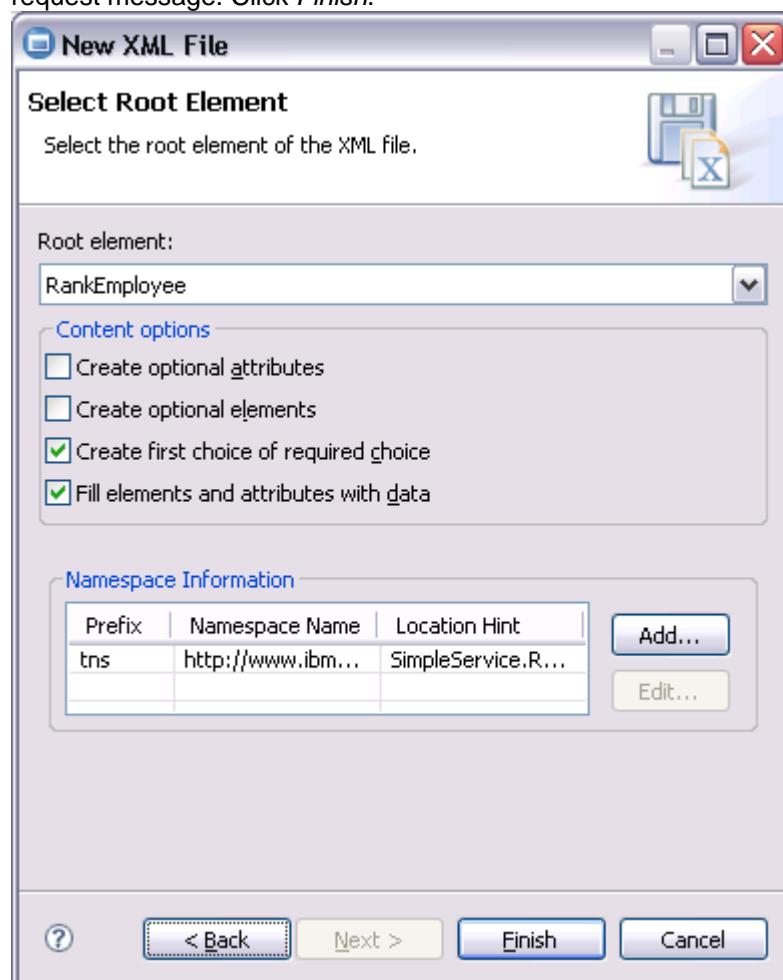
Figure E.4 – Select the root element from the XML schema

Note:

Data Studio always uses the operation name as the root element for the request message and the operation name with "Response" as the suffix for the response message.

---

7. Data Studio generates the XML instance document and opens it in the XML editor. As shown in *Figure E.5*, switch to the *Source* view by clicking the *Source* tab in the middle of the panel, and change the value of the *EMPLOYEE_CODE* tag to **10004** and the *RANKING* value to **Excellent**.

Figure E.5 – The generated XML instance document

> 8. Save the file. It appears in the *XML -> XML documents* folder after refreshing your Data Development project. When executing the SOAP binding for the `RankEmployee` operation with the Web Services Explorer, you can see that the generated SOAP request message content looks very similar, since both match the same XML schema.

Repeat these steps for all operations you want to test using cURL.

E.1.2 SOAP over HTTP Binding

The first service binding we look at is SOAP over HTTP. We described how to test this binding in *Chapter 10* using the Web Services Explorer. In this section, we show you how to re-create what the Web services Explorer did for you before.

Start by assembling the SOAP request message by creating a new XML file called `RankEmployeeSOAP.xml`. Into that file, copy and paste the SOAP request message from the source view of the Web Services Explorer (as shown in *Listing E.1*.)

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://www.ibm.com/db2/onCampus"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <q0:RankEmployee>
      <EMPLOYEE_CODE>10004</EMPLOYEE_CODE>
      <RANKING>Excellent</RANKING>
    </q0:RankEmployee>
  </soapenv:Body>
</soapenv:Envelope>
```

Listing E.1 - The SOAP request message

Invoke the SOAP binding using the cURL command. To do this, you need to know the SOAP over HTTP endpoint URL. Data Web Services (DWS) has the following rules to get to the SOAP endpoint URL:

```
http(s)://<server>:<port>/<contextRoot>/services/<ServiceName>
```

For the **SimpleService** example, the endpoint URL is:

```
http://server:8080/WebServicesSimpleService/services/SimpleServi
ce
```

The cURL command to send the request to the Web service should look like this:.

```
curl.exe -d @RankEmployeeSOAP.xml
          -H "Content-Type: text/xml"
          -H "SOAP-Action:
\"http://www.ibm.com/db2/onCampus/RankEmployee\""
          -v
http://localhost:8080/WebServicesSimpleService/services/SimpleService
```

> **Note:**
>
> Argument used:
>
> **-d @<filename>**
> Name of the file with the SOAP request message. This also forces cURL to use HTTP POST for the request.
>
> **-H**
> Additional header fields need to be specified for the request. The server needs to know the **Content-Type**, which is XML and the **SOAPAction** header, which can be found in the in the binding section for the SOAP endpoint in the WSDL document. **Note:** The SOAPAction String needs to be included in double quotes.
>
> **-v**
> The verbose switch to show detailed messages.
>
> **<url>**
> The URL to send the request to. This needs to be the SOAP over HTTP endpoint URL of your Web service. It can be found in the WSDL document or by using the Web Services Explorer.

The output of the command should look similar to what is shown in *Listing E.2*:

```
* About to connect() to localhost port 8080 (#0)
*    Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /WebServicesSimpleService/services/SimpleService HTTP/1.1
> User-Agent: curl/7.18.2 (i386-pc-win32) libcurl/7.18.2 OpenSSL/0.9.8h
libssh2/0.18
> Host: localhost:8080
> Accept: */*
> Content-Type: text/xml
> SOAPAction:"http://www.ibm.com/db2/onCampus/RankEmployee"
> Content-Length: 389
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: text/xml;charset=utf-8
< Transfer-Encoding: chunked
```

```
< Date: Sun, 28 Jun 2009 04:21:21 GMT
<
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope
xmlns:soapenv="http://sc
hemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" x
mlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><soapenv:Body><ns1:RankEmpl
oyeeResponse xmlns:ns1="http://www.ibm.com/db2/onCampus"
xmlns:xsi="http://www.w
3.org/2001/XMLSchema-instance"><row><RANKING_DATE>2009-06-
27T21:21:21.312Z</RANK
ING_DATE><RANKING_YEAR>2009</RANKING_YEAR><EMPLOYEE_CODE>10004</EMPLOYEE_C
ODE><R
ANKING_CODE>5</RANKING_CODE></row></ns1:RankEmployeeResponse></soapenv:Bod
y></so
apenv:Envelope>
```
Listing E.2 – The service response

If successful, the SOAP response message is displayed together with the HTTP header fields for the request and response.

> **Note:**
>
> SQL NULL values are represented via the `xsi:nil` attribute; that is, `xsi:nil="true"` indicates an SQL NULL value.

When using the SOAP binding, your request gets routed through the SOAP framework at the application server. Depending on the framework used, you can add additional configuration artifacts – like SOAP handlers or WS-* configurations – to your Web service.

But you can also use one of the more "simple" HTTP RPC (remote procedure call) bindings described in the following sections.

E.1.3 HTTP POST (XML) Binding

The HTTP POST (XML) binding is similar to the SOAP binding. The difference from the SOAP binding is that it does not get routed through a SOAP framework on the server side, and the messages are not following the SOAP specification. Only the "plain" XML payload is exchanged without the SOAP Envelope and SOAP Body tags. A simple HTTP POST request is used to send the XML request to the server.

You can reuse the `SimpleService.RankEmployee.default.xml` file you created before as the request message document.

You also need to know the REST endpoint URL in order to send your request. Data Web Services (DWS) has the following rules to get to the SOAP endpoint URL:

```
http(s)://<server>:<port>/<contextRoot>/rest/<ServiceName>/<Operati
onName>
```

To invoke the *RankEmployee* operation of the *SimpleService* example, your endpoint URL looks like this:

```
http://server:8080/WebServicesSimpleService/rest/SimpleService/Rank
Employee
```

> **Note:**
>
> The REST endpoint URL is used for all HTTP RPC bindings:
>
> - HTTP POST (XML)
> - HTTP POST (application/x-www-form-urlencoded)
> - HTTP POST (JSON)
> - HTTP GET
>
> You can enable or disable all HTTP Bindings for a Web service by checking or unchecking the *REST (Web access)* option in the *Deploy Web Service* dialog described in *Chapter 10.*

The cURL command to send the request to the Web service should look like this:

```
curl.exe -d @ SimpleService.RankEmployee.default.xml
         -H "Content-Type:text/xml;charset=utf-8"
         -v
http://localhost:8080/WebServicesSimpleService/rest/SimpleService/RankEmpl
oyee
```

E.1.4 HTTP POST (application/x-www-form-urlencoded) Binding

This binding can be used with HTML forms using the POST action. In this case, the parameters are sent as key/value pairs, where each pair is separated by an ampersand ('&").

You can also use cURL to test this binding. Create a new text file called `RankEmployeeUrlEncoded.txt`. The content of your file should look like this:

```
EMPLOYEE_CODE=10004&RANKING=Excellent
```

The cURL command to send the request to the Web service should look like this:

```
curl.exe -d @"RankEmployeeUrlEncoded.txt"
         -H "Content-Type:application/x-www-form-urlencoded"
         -v
http://localhost:8080/WebServicesSimpleService/rest/SimpleService/RankEmpl
oyee
```

The response message is the same as for the HTTP POST (XML) binding.

> **Note:**
>
> The HTTP POST (application/x-www-form-urlencoded) binding is listed in the WSDL file and can be tested using the Web Services Explorer as well. In case of the SimpleService the binding is called **SimpleServiceHTTPPOST**.

> **Note:**
>
> SQL NULL values are treated as absent. This means parameter values that are not present in the key/value string are set to SQL NULL. A parameter with an empty value is treated as an empty string.

E.1.5 HTTP GET Binding

This binding uses the HTTP GET verb with a URL to invoke the Web service operation. Since there is no content sent to the server, all input parameters must become part of the URL. This is done using the URL query string. Everything that follows the question mark "?' sign in a URL is specified as the query string. A query string consists of key/value pairs which are concatenated using the ampersand '&' character.

The cURL command that sends the request to the Web service should look like this:

```
curl.exe -v
http://localhost:8080/WebServicesSimpleService/rest/SimpleService/RankEmployee?
EMPLOYEE_CODE=10004&RANKING=Excellent
```

> **Note:**
>
> The HTTP GET binding is listed in the WSDL file and can be tested using the Web Services Explorer as well. In the case of the SimpleService, the binding is called **SimpleServiceHTTPGET**.

> **Note:**
>
> **Multi-byte characters in URL strings:**
>
> If your data contains multi-byte characters, you need to consider the following:
>
> - Multi-byte characters need to be provided in UTF-8
>
> - The UTF-8 bytes need be URL-encoded to follow the URI/URL specification. For example, if you have a parameter value in Chinese like 日本語 your URL must look like this:
>
> ```
> http://localhost:8080/JmaWebService/rest/WebService/Test?p1=%
> E6%97%A5%E6%9C%AC%E8%AA%9E
> ```

Application Servers and multi-byte UTF-8 characters in URLs:

You may have to perform some additional configuration steps at your application server to treat multibyte UTF-8 characters in URLs correctly.

Tomcat
With Tomcat, you need to add the attribute `URIEncoding="UTF-8"` to your `<Connector>` configurations in the `server.xml` file. More details can be found here:
http://wiki.apache.org/tomcat/FAQ/Connectors

WebSphere Application Server Community Edition (WAS CE):
WAS CE ships Tomcat as its Web container - but there is no `server.xml` file. Instead, there is a Tomcat configuration section in the `$WASCE_HOME/var/config/config.xml` file. You need to add `<attribute name="uriEncoding">UTF-8</attribute>` to the `<gbean name="TomcatWebConnector">` section. More details can be found here:
http://publib.boulder.ibm.com/wasce/V2.1.1/en/tomcat-configuration.html

Note:

SQL NULL values are treated as absent. This means parameter values that are not present in the key/value string are set to SQL NULL. A parameter with an empty value is treated as an empty string.

You can also easily test the HTTP GET binding with your Web Browser. Simply enter the URL into your browser to invoke the Web service operation. *Figure E.6* shows what the **RankEmployee** operation looks like when invoked with Firefox.
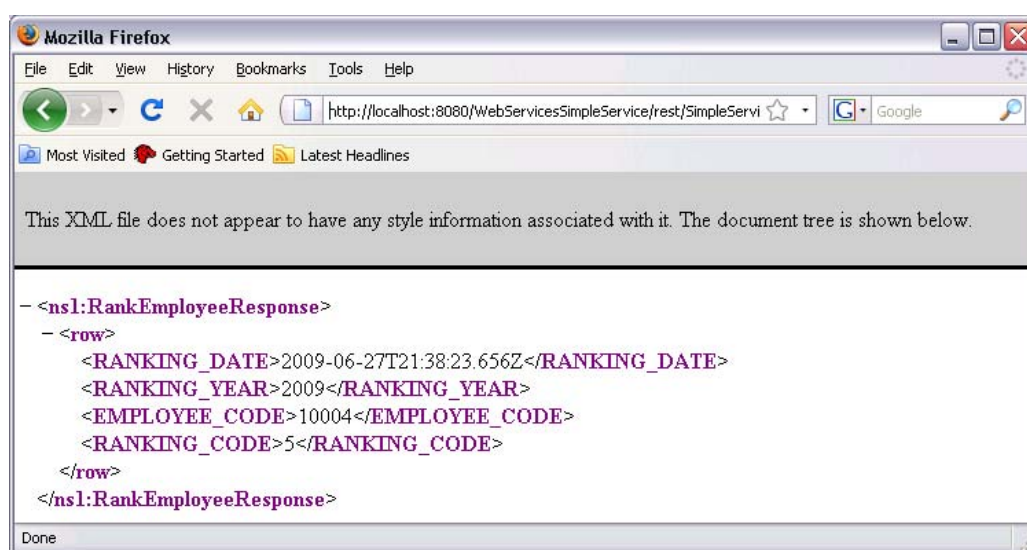
Figure E.6 – The service response in a Web browser window

E.1.6 HTTP POST (JSON) Binding

Finally, Data Web Services provides you with a simple JSON binding that can be leveraged from JavaScript applications –for example, when using AJAX with the **XMLHttpRequest** object. In order to test the JSON binding with cURL you need to create the JSON request message first.

The building rules for a Data Web Services JSON request message are as follows:

```
{"<operationName>":{"<parameter1>":<value1>,"<parameter1>":<value1>,…}}
```

Note:

JSON data type formatting:

The data type formats follow the JSON specification. Date, time and timestamp types are expected to be provided in XSD format: `xs:date`, `xs:time` and `xs:dateTime`. Binary data types are expected as base64 encoded strings. SQL NULL values are represented as JSON null.

Create a new file called `RankEmployeeJSON.txt`. The content of the file should look like this:

```
{"RankEmployee":
   {"EMPLOYEE_CODE":10004,"RANKING":"Excellent"}
}
```

The cURL command to send the request to the Web service should look this:

```
curl.exe -d @"GetBestSellingProductsByMonthJSON.txt"
        -H "Content-Type:application/json;charset=utf-8"
-v
http://localhost:8080/WebServicesSimpleService/rest/SimpleService/RankEmpl
oyee
```

The output of the command should look similar to what is shown in *Listing E.3.*

```
...
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Cache-Control: no-cache, no-store, max-age=0
< Expires: Thu, 01 Jan 1970 00:00:01 GMT
< Content-Type: application/json;charset=UTF-8
< Content-Length: 129
< Date: Sun, 28 Jun 2009 04:48:26 GMT
<
{"RankEmployeeResponse":[{"RANKING_DATE":"2009-06-27T21:48:26.203Z","RANKING_YEA
R":2009,"EMPLOYEE_CODE":10004,"RANKING_CODE":5}]}
```

Listing E.3 – The service response

The response is also formatted as JSON.

> **Note:**
>
> **Switching output format from XML to JSON:**
>
> For all HTTP RPC bindings, if the response should be returned as XML or JSON, you can specify the `_outputFormat` control parameter (the initial underscore character marks it as a control parameter) in the URL to define. For all bindings except HTTP POST (JSON), the output format is XML by default.
>
> Example (HTTP GET with JSON response):
>
> ```
> http://localhost:8080/WebServicesSimpleService/rest/SimpleService/RankEmployee?EMPLY
> EE_CODE=10004&RANKING=Poor&_outputFormat=JSON
> ```

E.2 Simplify access for single-row results

You can use an option called *Fetch only single row for queries* for query operations that you know will return only a single row to reduce the complexity of the service response data structure. It simplifies the message response by removing the `<row>` tag around the single row result. Since it's known that the query operation only returns one result row, the Data Web Services runtime can skip the row delimiter tag in the response.

The `RankEmployee` operation for example always returns a single row only. To remove the `<row>` tag:

1. From the Data Project Explorer, right-click the operation in your Web service and select *Edit*. The *Edit Operation* dialog opens as shown in *Figure E.7*.
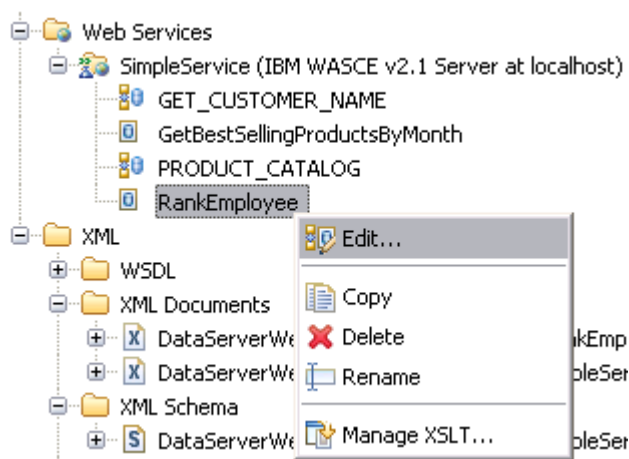
Figure E.7 – Select the Edit option for an operation

2. Check the *Fetch only single row for queries* option and click *Finish* as shown in *Figure E.8*.
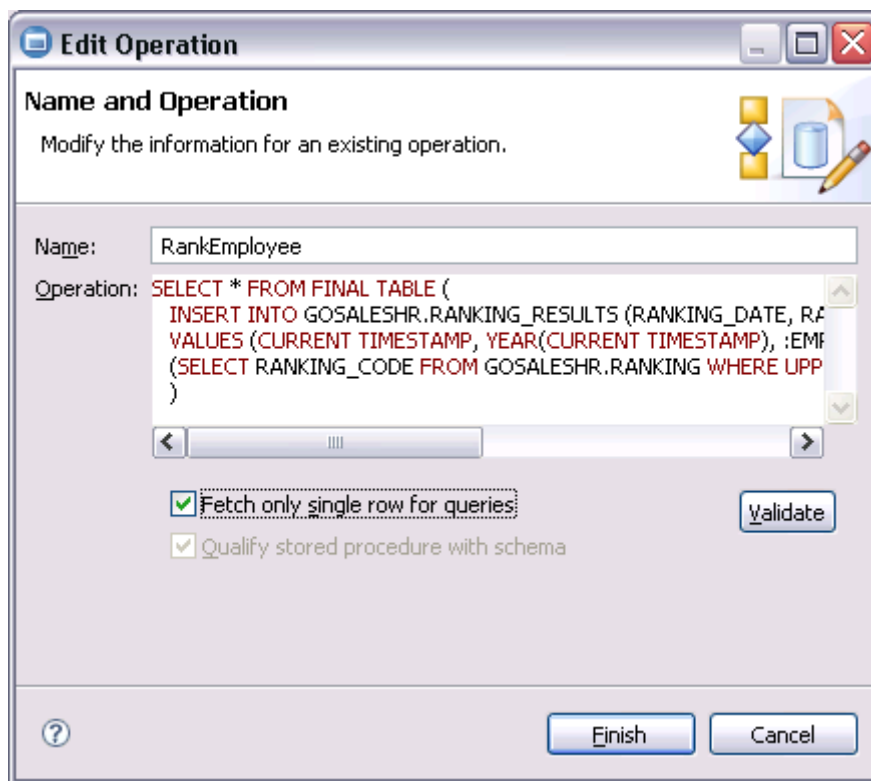
Figure E.8 – Check the Fetch only single row option

> 3. Re-deploy the Web service to propagate your changes to the application server, as described in *Chapter 10.*

When invoking the `RankEmployee` operation, you will see that there is no `<row>` tag, as shown in *Listing E.4*.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:RankEmployeeResponse xmlns:ns1="http://www.ibm.com/db2/onCampus"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
       <RANKING_DATE>2009-06-27T21:57:08.109Z</RANKING_DATE>
       <RANKING_YEAR>2009</RANKING_YEAR>
       <EMPLOYEE_CODE>10004</EMPLOYEE_CODE>
       <RANKING_CODE>5</RANKING_CODE>
</ns1:RankEmployeeResponse>
```

Listing E.4 – The RankEmployee response message without a <row> tag

Data Web Services also changes the XML schema for the response message in the WSDL accordingly.

E.3 Processing stored procedures result sets

As was discussed in *Chapter 10*, there are some special considerations for stored procedures that return result sets. Data Studio uses DB2 catalog information to generate the XML schema file for the operation's input and output messages. But the DB2 catalog does not contain metadata information about result sets returned by stored procedures.

Only the maximum number of result sets returned is known, which forces Data Studio to assign a very generic result set definition represented by the `anonymousResultSetType`, as shown in *Listing E.6*.

```
<complexType name="anonymousResultSetType">
  <sequence>
    <element maxOccurs="unbounded" minOccurs="0" name="row">
      <complexType>
        <sequence maxOccurs="unbounded" minOccurs="0">
          <any processContents="skip"/>
        </sequence>
      </complexType>
    </element>
  </sequence>
</complexType>
```
Listing E.6 – The anonymousResultSetType

You can see the reference to the `anonymousResultSetType` in the XML schema definition for the **PRODUCT_CATALOG** stored procedure response message, as shown in *Listing E.5*.

```
<element name="PRODUCT_CATALOGResponse">
  <complexType>
    <sequence>
      <element maxOccurs="1" minOccurs="0" name="rowset"
type="tns:anonymousResultSetType"/>
    </sequence>
  </complexType>
</element>
```
Listing E.5 – Reference to the anonymousResultSetType

The generic result set information can cause problems with Web service clients that rely on the message schema provided with the WSDL file – as you could see in *Chapter 10* with the Web Services Explorer, where the result set content was not displayed correctly (*Figure 10.25*).

Data Studio provides a way to circumvent this problem, but your stored procedure must match the criteria that it **always returns the same number of result sets with the same metadata information for every possible invocation.** If this is the case, you can add a more detailed result set XML schema. Follow these steps to add the additional result set information for the **PRODUCT_CATALOG** procedure:

1. From the Data Project Explorer, right-click the *PRODUCT_CATALOG* operation and select *Edit ...* to open the *Edit Operation* dialog.

2. Click *Next* to get to the *Generate XML Schema for Stored procedure* dialog and click the *Generate* button as shown in *Figure E.9.*
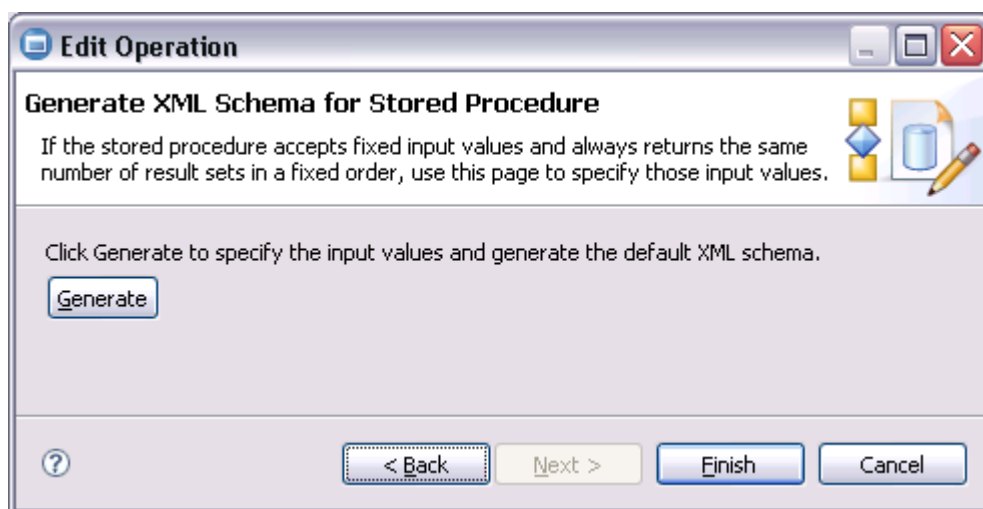
Figure E.9 – Generate XML schema for stored procedure

3. You will be prompted for input parameters in case the procedure has one or more input parameters defined. Use **Irons** as the value for the `PRODUCT_TYPE` parameter, as shown in *Figure E.10*.
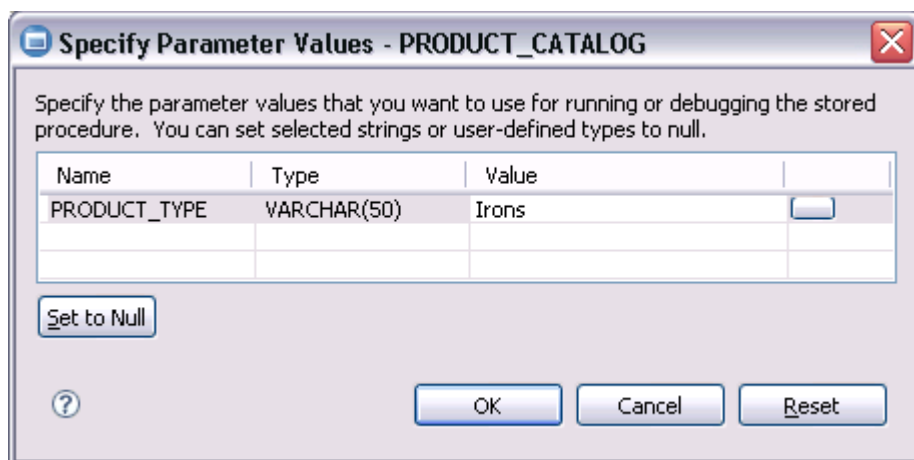
Figure E.10 – Provide stored procedure input parameter

4. Click *Finish* and re-deploy your Web service.

If compare the result from the Web Services Explorer shown in *Figure E.11* with that shown in *Figure 8.25,* you can see that the response is now displayed correctly.
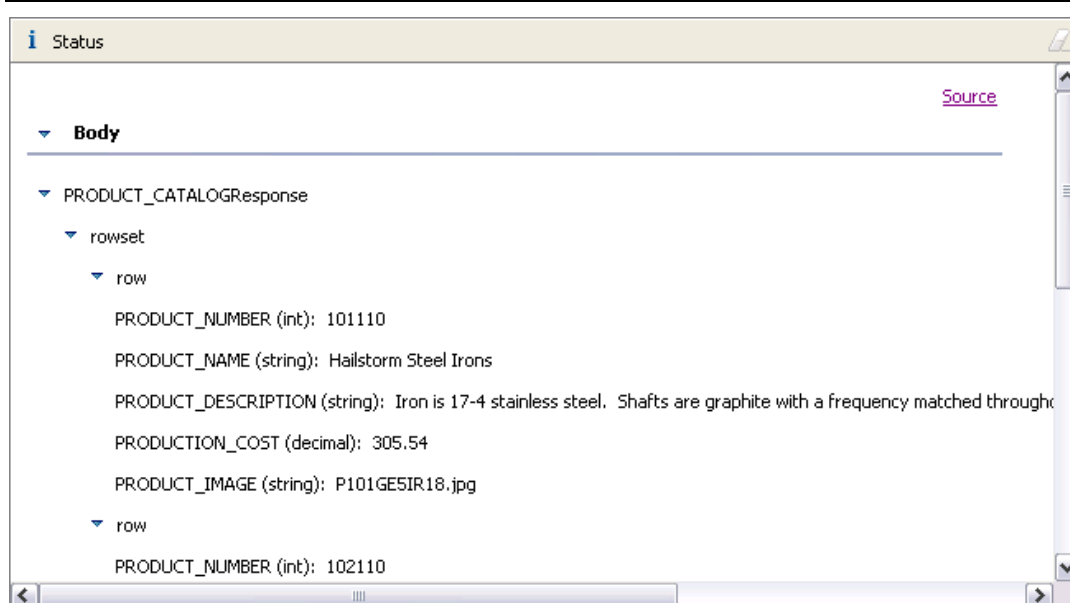
Figure E.11 – Stored procedure results now accurately mapped in the response

> **Note**
>
> The result set XML message did not change. The only difference is the more verbose XML schema for the operation response message.

If you look at the XML schema for the **PRODUCT_CATALOG** response message as shown in *Listing E.6*, you can see that the reference to the **anonymousResultSetType** is gone. Instead, there is now the actual column information for the result set.

```
<element name="PRODUCT_CATALOGResponse">
  <complexType>
    <sequence>
      <element name="rowset">
        <complexType>
          <sequence>
            <element maxOccurs="unbounded" minOccurs="0" name="row">
              <complexType>
                <sequence>
                  <element name="PRODUCT_NUMBER" nillable="true" type="xsd:int"/>
                  <element name="PRODUCT_NAME" nillable="true" type="xsd:string"/>
                  <element name="PRODUCT_DESCRIPTION" nillable="true"
type="xsd:string"/>
                  <element name="PRODUCTION_COST" nillable="true"
type="xsd:decimal"/>
                  <element name="PRODUCT_IMAGE" nillable="true" type="xsd:string"/>
                </sequence>
              </complexType>
            </element>
          </sequence>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
```
**Listing E.6 – Schema with verbose result set information**

E.4 Transform input and output messages using XSL

You can assign an Extensible Stylesheet Language Transformation (XSLT) to one or more of your operations to change the default XML input and output message format. This feature allows you to customize the format of the messages that the client sees; for example, to support industry standards or to changing or hiding default tag names in the request and response messages. You can even generate non-XML outputs – like HTML pages.

To give you some hands-on experience about stylesheets and what they can do for you, we show you an easy way to transform your Web service response into a service message that is written in HTML format. All Web browsers can interpret HTML code. As a result, you can invoke your Web service directly through a Web browser and receive a formatted response.

We use the `GetBestSellingProductsByMonth` operation and change its output into HTML. You can simply use the HTTP GET binding to invoke the operation from a Web browser and verify the HTML response.

E.4.1 Creating an XSL stylesheet

Data Studio provides an XSL editor as well as the ability to test your XSL script. To create a new XSL file:

1. Select *File -> New -> Other … -> XML.*
2. Select your Data Development Project as the parent folder and use the name *GetBestSellingProductsByMonthToHTML.xsl.*
3. Click *Finish* to create the XSL file. The new XSL file should appear in the *XML -> XSLT* folder of your project as shown in *Figure E.12*.
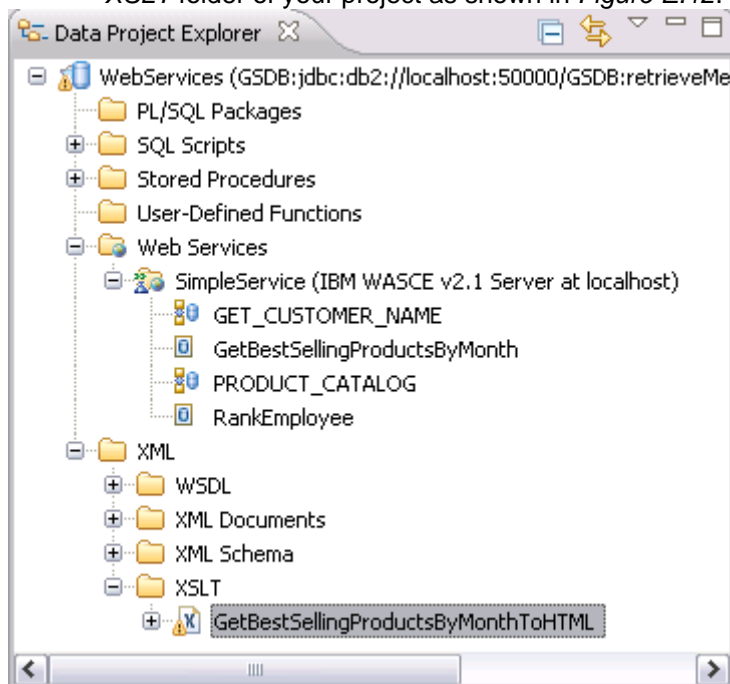
Figure E.12 – XSL document in Data Development project

    4. Double-click the file to open it with the XSL Editor. For testing purposes we use a rather simple XSL script shown in *Listing E.7*. Save the file.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<!-- use html as method to indicate that we generate HTML -->
<xsl:output method="html" encoding="UTF-8" media-type="text/html" />

<xsl:template match="/*">
<html>
 <head>
  <title>Best Selling Products</title>
  </head>
 <body>
  <table border="1">
   <tr bgcolor="#9acd32">
   <!-- use XML tag names of the first row as table header -->
    <xsl:if test="//row">
      <xsl:for-each select="//row[1]/*">
        <td style="width:150px">
          <b>
           <xsl:value-of select="local-name()" />
          </b>
         </td>
      </xsl:for-each>
    </xsl:if>
   </tr>
   <!-- iterate over all rows and fill the table -->
   <xsl:for-each select="//row">
    <tr>
     <xsl:for-each select="*">
      <td style="width:150px">
       <xsl:value-of select="text()" />
      </td>
     </xsl:for-each>
    </tr>
   </xsl:for-each>
  </table>
 </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Listing E.7 – XSL script transforming GetBestSellingProductsByMonth response

    5. To assign the XSL stylesheet, right-click at the *GetBestSellingProductsByMonth* operation and select *Manage XSLT…* as shown in *Figure E 13.*
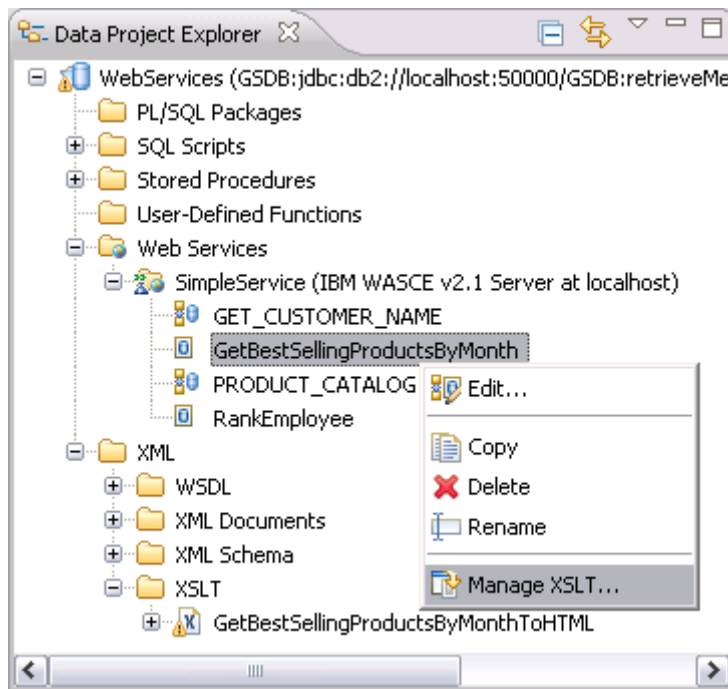
Figure E.13 – Select the "Manage XSLT…" option

6. Click the *Browse* button under *Transformation of Output Messages* and point to your XSL stylesheet, as shown in *Figure E.14*.
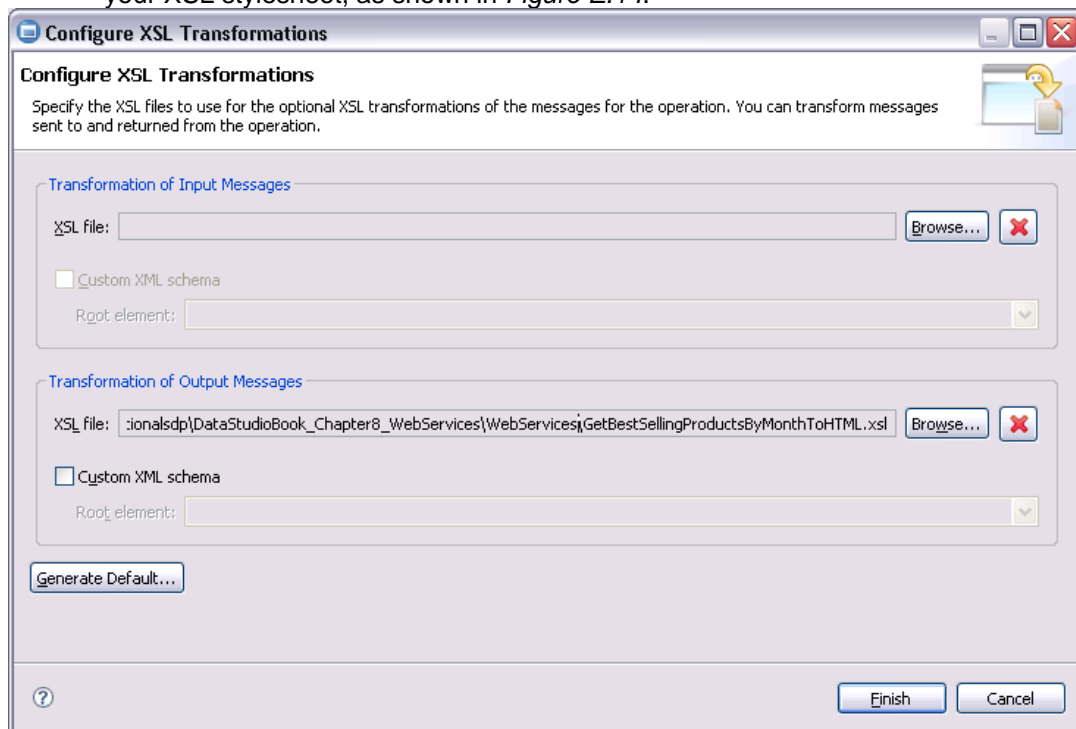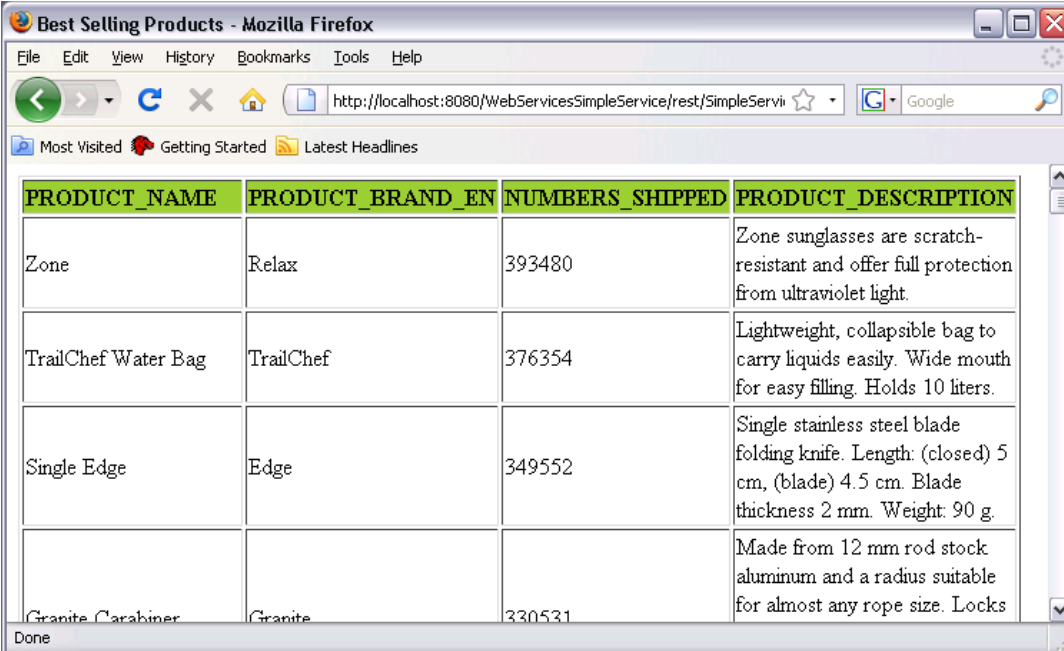
Figure E.14 – Configure XSL Transformation dialog

7.   Click *Finish* and re-deploy your Web service.

When invoking the **`GetBestSellingProductsByMonth`** operation now from a browser, you can see that the response is formatted as HTML. The URL to get the best selling products for April looks like this:

```
http://server:8080/WebServicesSimpleService/rest/SimpleService/GetBestSell
ingProductsByMonth?MONTH=4
```

The response is shown in *Figure E.15*.

Figure E.15 – Response transformed as HTML

Note:

When looking at the WSDL file you will recognize that the **`GetBestSellingProductsByMonth`** are missing in the SOAP binding. This is due to the fact that now HTML is produced, but a SOAP message needs to be XML.

E.4.2 Data Web Services XSL Extensions

The Data Web services runtime provides a few XSL extension functions that allow you to access the request URL, the request HTTP header fields, and set the HTTP response header fields, shown in *Table E.1*.

Extension function	Description

getHTTPRequestHeader(header)	Returns the value for a given HTTP request header
getHTTPRequestURL()	Returns the request URL
getHTTPRequestQueryString()	Returns the query string of the URL
setHTTPResponseHeader(header, value)	Sets the value for a given HTTP response header field
encodeJSON(value)	Encodes the string as JSON string – can be used to generate custom JSON output

Table E.1 – Available XSL Extension functions

The XSL stylesheet shown in *Listing E.8* demonstrates some of the extension functions.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
                xmlns:xalan="http://xml.apache.org/xslt"
                xmlns:java="http://xml.apache.org/xalan/java"
                exclude-result-prefixes="xalan java">
<xsl:output method="html" encoding="UTF-8" media-type="text/html" />

<xsl:template match="/*">
<html>
 <head><title>XSL Extension Test</title></head>
 <body>

   <table border="1">
       <tr bgcolor="#9acd32">
              <td colspan="2"><h2>Request URL</h2></td>
       </tr>
       <tr>
              <td colspan="2"><xsl:value-of
select="java:com.ibm.datatools.dsws.rt.common.XSLExtensions.getHTTPRequestURL()"/></td>
       </tr>
       <tr bgcolor="#9acd32">
              <td colspan="2"><h2>Request URL Query String</h2></td>
       </tr>
       <tr>
              <td colspan="2"><xsl:value-of
select="java:com.ibm.datatools.dsws.rt.common.XSLExtensions.getHTTPRequestQueryString()"/></td>
       </tr>
       <tr bgcolor="#9acd32">
              <td colspan="2"><h2>Request HTTP Header</h2></td>
       </tr>
       <tr>
              <td>Content-Type</td>
              <td><xsl:value-of
select="java:com.ibm.datatools.dsws.rt.common.XSLExtensions.getHTTPRequestHeader('Content-
Type')"/></td>
       </tr>
       <tr>
              <td>User-Agent</td>
              <td><xsl:value-of
select="java:com.ibm.datatools.dsws.rt.common.XSLExtensions.getHTTPRequestHeader('User-
Agent')"/></td>
       </tr>
       <tr>
              <td>Host</td>
              <td><xsl:value-of
select="java:com.ibm.datatools.dsws.rt.common.XSLExtensions.getHTTPRequestHeader('Host')"/></td>
       </tr>
       <tr>
              <td>Accept</td>
```

```
            <td><xsl:value-of
select="java:com.ibm.datatools.dsws.rt.common.XSLExtensions.getHTTPRequestHeader('Accept')"/></td>
        </tr>
        <tr>
            <td>Content-Length</td>
            <td><xsl:value-of
select="java:com.ibm.datatools.dsws.rt.common.XSLExtensions.getHTTPRequestHeader('Content-
Length')"/></td>
        </tr>
    </table>

    <table border="1">
        <tr bgcolor="#ffff44">
            <td colspan="2"><h2>GET_CUSTOMER_NAME RESPONSE</h2></td>
        </tr>
        <tr>
            <td>First Name:</td>
            <td><xsl:value-of select="//FIRST_NAME/text()"/></td>
        </tr>
        <tr>
            <td>Last Name:</td>
            <td><xsl:value-of select="//LAST_NAME/text()"/></td>
        </tr>
        <tr>
            <td>Phone Number:</td>
            <td><xsl:value-of select="//PHONE_NUMBER/text()"/></td>
        </tr>

    </table>

 </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Listing E.8 – XSL script to test extension functions

1. Using the steps described in the previous section, create a new XSL file with the name `TestXSLExtensions.xsl` and copy the information in *Figure E8* into that script.

2. Assign the `TestXSLExtensions.xsl` to transform the output message of the **GET_CUSTOMER_NAME** operation and re-deploy the Web service.

3. Now, you can execute the **GET_CUSTOMER_NAME** operation with HTTP GET using a Web browser. A URL to retrieve the information for a customer with the ID 126911 looks similar to this:

```
http://localhost:8080/WebServicesSimpleService/rest/SimpleService/GET_CUST
OMER_NAME?CUSTOMERID=126911
```

As you can see in *Figure E.16*, the response contains some information from the HTTP request – like the request URL, some HTTP request headers, and the result of the **GET_CUSTOMER_NAME** operation.
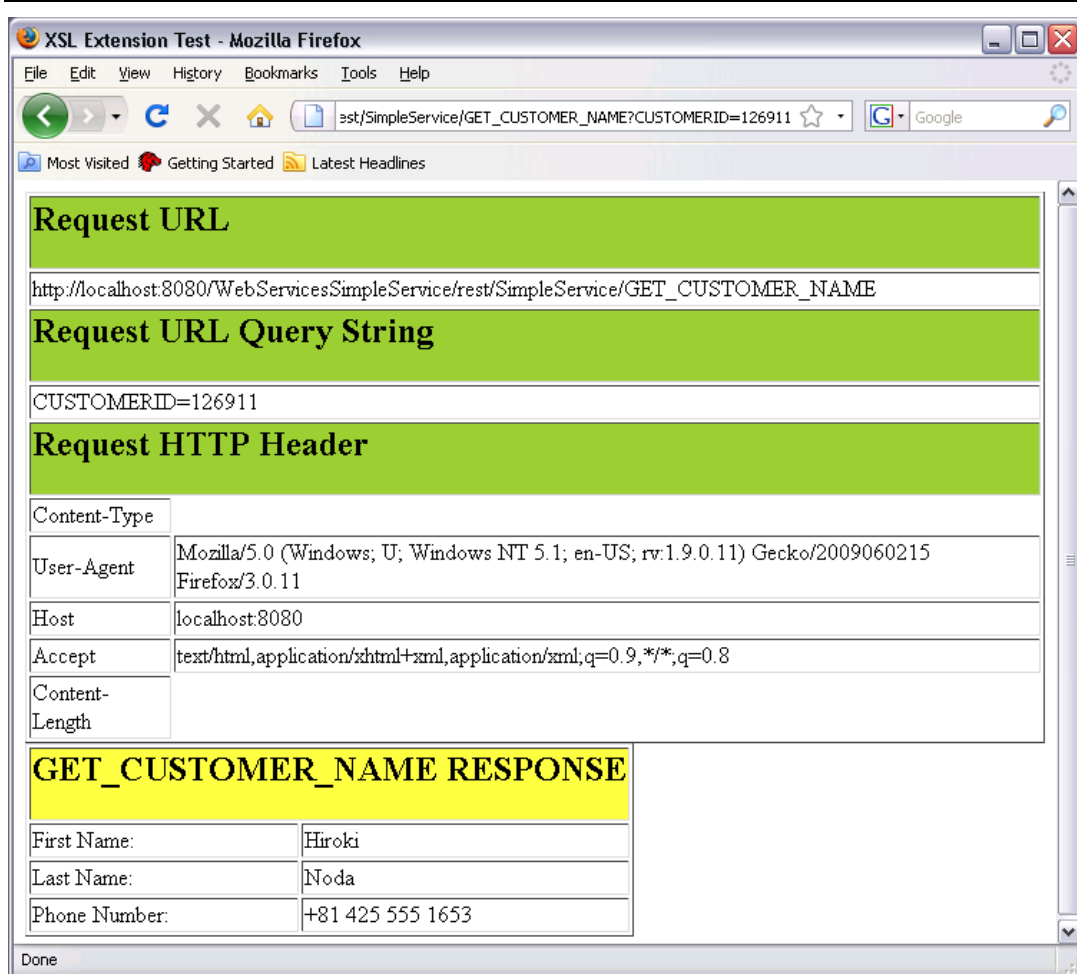
Figure E.16 – XSL extension functions provide additional information to the result

E.5 A closer look at the generated runtime artifacts

Explaining all artifacts in detail is beyond the scope of this book. Here is a brief glimpse at the files and structures. More information can be found in the documentation for JAVA EE, WTP, Servlets, WebSphere Application Server Community Edition, and the SOAP frameworks.

The Data Web services tools hook into the Web Tools Platform (WTP) framework, which is an Eclipse-based JAVA EE development environment. Data Studio contains WTP. The JAVA EE perspective is part of WTP.

http://www.eclipse.org/webtools/

Switch to the JAVA EE perspective to take a closer look at the generated runtime artifacts. As shown in *Figure E.17*, the Project Explorer shows three projects. One is your *WebServices* Data Development Project whereas the other two are JAVA EE projects representing the runtime artifacts for your *SimpleService*.
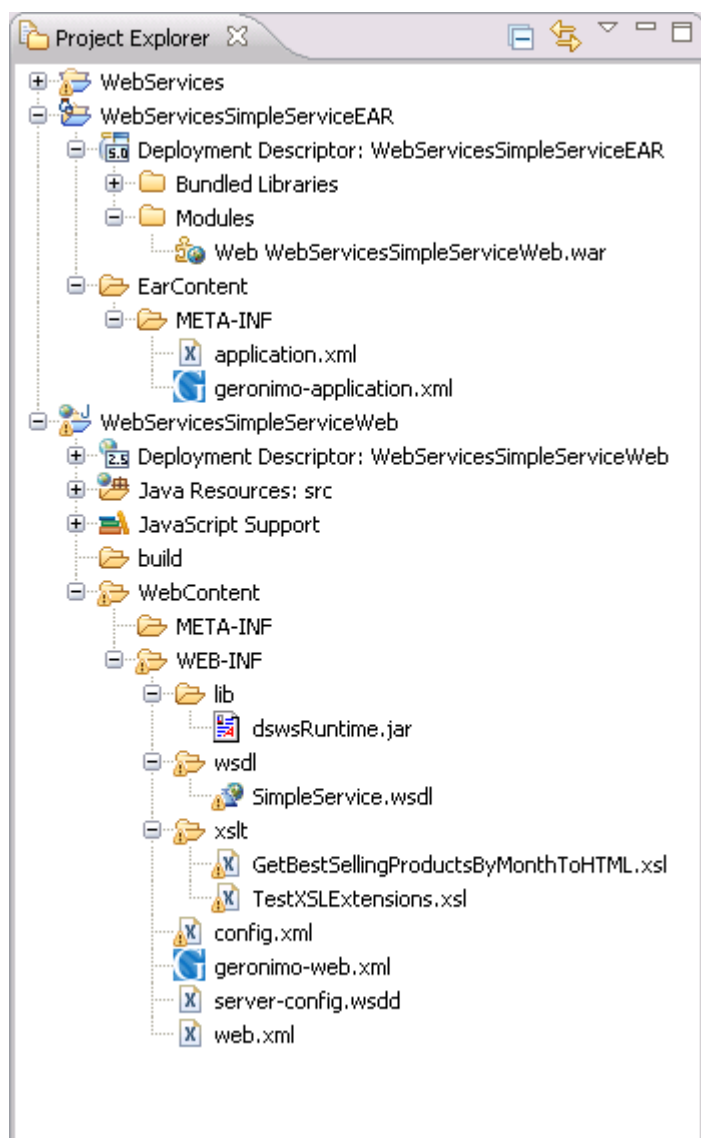
Figure E.17 – The generated Web service project in the JAVA EE perspective

Let's take a brief look at those two generated projects for the `SimpleService` Web service:

WebServiceSimpleServiceEAR
This project represents an "Enterprise Application Archive" (EAR). It can be seen as a container project for the actual Web service. You can see that the *WebServiceSimpleServiceWeb* is referenced under *Modules*. In addition, you can find configuration files to define settings like context root or data source definitions.

WebServiceSimpleServiceWeb
This "Web Application Archive" (WAR) project contains the actual Web service logic and configuration. The structure of the project follows the Servlet specification.

E.5.1 JAVA EE artifacts

File name	Description
WebServiceSimpleServiceEAR/EarContent/META-INF/application.xml	Contains configuration information about the contained modules, like the context root.
WebServiceSimpleServiceWeb/WebContent/WEB-INF/web.xml	Contains configuration information about the Web Application, including Servlet class names, URL mappings, resource references, security settings, etc.

Table E.2 – JAVA EE artifacts

E.5.2 SOAP framework artifacts

The configuration files for the SOAP engine vary depending on the selected SOAP framework.

File name	Description
WebServiceSimpleServiceWeb//WebContent/WEB-INF/server-config.wsdd	Deployment descriptor file for the Apache Axis 1.4 SOAP engine.

Table E.3 – Apache Axis 1.4 deployment descriptor file

E.5.3 WAS CE artifacts

The configuration files for the application server may vary depending on the selected application server. The WebSphere Application Server Community Edition Documentation can be found here: http://publib.boulder.ibm.com/wasce

File name	Description
WebServiceSimpleServiceEAR /EarContent/META-INF/geronimo-application.xml	WAS CE extension configuration file for Enterprise applications. It contains metadata about the data source configuration, required Java libraries, and other information.
WebServiceSimpleServiceWeb//WebContent/WEB-INF/geronimo-web.xml	WAS CE extension file for Web applications. It contains metadata about data source references, required Java libraries, and other information.

Table E.4 – WAS CE deployment descriptor files

E.5.4 Data Web Services artifacts

File name	Description
WebServiceSimpleServiceWeb//WebContent/WEB-INF/config.xml	DWS configuration file. You can find the mapping between operation names and SQL statements as well as references to XSL scripts and namespace declarations in here.
WebServiceSimpleServiceWeb//WebContent/WEB-INF/lib/dswsRuntime.jar	The generic DWS Java runtime library.

WebServiceSimpleServiceWeb//WebContent/WEB-INF/wsdl/SimpleService.wsdl	The generated WSDL file for your Web service.
WebServiceSimpleServiceWeb /WEB/WebContent/WEB-INF/xslt	A folder which holds the XSL stylesheet you assigned to your operations for input/output message transformation.

Table E.5 – Data Web Services artifacts

If you are familiar with the generated artifacts you can start to do some customization –for example, adding Servlets, JSPs, HTML pages, and advanced configuration like setting up authentication/authorization, security, etc.

E.6. Selecting a different SOAP framework

The supported SOAP framework depends on the selected application server. You may have the choice between multiple SOAP frameworks. For WAS CE, the following SOAP frameworks are supported:

- Apache Axis 1.4 (default) (http://ws.apache.org/axis/)

- Apache Axis 2 (http://ws.apache.org/axis2/)

- JAX-WS (http://jcp.org/en/jsr/detail?id=224)

You can select the SOAP framework by clicking the *artifact.soapEngine* property in the *Parameters* table of the *Deploy Web Service* dialog, as shown in *Figure E.18*.
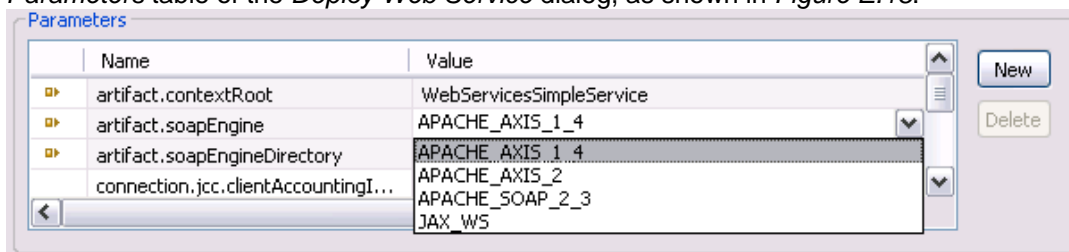
Figure E.18 – Selecting a SOAP framework in the Deploy Web Service dialog

The Data Web Services tools do not add any SOAP framework libraries to the Web application. It is expected that the SOAP engine libraries are present at the application server.

References

[1] HAYES, H. *Integrated Data Management: Managing data throughout its lifecycle,* developerWorks article, 2008; updated 2009. Originally published by IBM developerWorks at http://www.ibm.com/developerworks/data/library/techarticle/dm-0807hayes/. Reprinted by permission.

[2] LEUNG, C. et. al. *SQL Tuning: Not just for hardcore DBAs anymore*, IBM Database Magazine article, Issue 2, 2009.

Resources

Web sites

1.   Data Studio page on developerWorks:
     https://www.ibm.com/developerworks/data/products/datastudio/

     Use this web site to find links to downloads, technical articles and tutorials, discussion forums, and more.

2.   Team blog: Managing the data lifecycle:
     http://www.ibm.com/developerworks/mydeveloperworks/blogs/idm/

     Experts from IBM blog on subjects related to Integrated Data Management. Includes everything from latest news to technical tips.

3.   Data Studio forum on developerWorks
     http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1086&categoryID=19

     Use the forum to post technical questions when you cannot find the answers in the manuals yourself.

4.   Data Studio Information roadmap:
     http://www.ibm.com/developerworks/data/roadmaps/roadmap_datastudio.html
     Includes organized links to important information about the product.

5.   Data Studio Information Center:
     http://publib.boulder.ibm.com/infocenter/dstudio/v3r1/index.jsp

     The information center provides access to online documentation for Data Studio. It is the most up-to-date source of information.

6.   DB2 Information Center:
     http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp

     The DB2 Information Center provides access to online documentation for DB2 for Linux, UNIX and Windows and provides the background information you will need to understand the implications of using Data Studio for certain operations.

7. InfoSphere Optim Data Management solutions web site:
    www.ibm.com/software/data/optim/

    Use this web site to get an understanding of the solutions that are available from IBM for data lifecycle management

8. IBM Redbooks site:
    http://www.redbooks.ibm.com/

    IBM Redbooks are no-charge and are written by teams of people in intense, hands-on residencies on a wide variety of technical products and technologies.

9. alphaWorks:
    http://www.alphaworks.ibm.com/

    This web site provides direct access to IBM's emerging technology. It is a place where one can find the latest technologies from IBM Research.

10. planetDB2:
    http://www.planetdb2.com/

    This is a blog aggregator from many contributors who blog about DB2 and related technologies.

11. Data Studio Technical Support:
    http://www.ibm.com/support/entry/portal/Overview/Software/Information_Managem ent/IBM_Data_Studio

    If you have an active license from IBM for DB2 or Informix, you can use this site to open a service request. You can also find alerts here, as well as links to fixes and downloads.

12. ChannelDB2:
    http://www.ChannelDB2.com/

    ChannelDB2 is a social network for the DB2 community. It features content such as DB2 related videos, demos, podcasts, blogs, discussions, resources, etc. for Linux, UNIX, Windows, z/OS, and i5/OS.

Books and articles

1. ALLEN, G. *Beginning DB2: From Novice to Professional.* Copyright 2008 by Grant Allen.
   ISBN-13: 978-1-59059-942-6
   ISBN-10: 1-59059-942-X

2. HORNIBROOK, J. and N. Kolunovsky. *Best Practices: Writing and tuning queries for optimal performance.* developerWorks article. May 2008.
   http://www.ibm.com/developerworks/data/bestpractices/querytuning/

3. JULIEN, L., et. al. *Use database catalog filtering in IBM Data Studio to view, manage, and work with database objects efficiently. developerWorks article. September 2011.*
   http://www.ibm.com/developerworks/data/library/techarticle/dm-1109datastudiodbcatalog/index.html

4. JULIEN, L. et. al. Managing database connections using the Data Studio web console. developerWorks article. November 2011.
   http://www.ibm.com/developerworks/data/library/techarticle/dm-1111datastudiowebconsole/index.html

5. LIGHTSTONE, S, T. Teorey, T. Nadeau, *Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more*, Morgan Kaufmann Press, 2007.
   ISBN: 0123693896

6. PAUSER, M. *IBM Data Studio: Get started with Data Web Services*, developerWorks tutorial. Originally published November 2007, updated December 2008.
   http://www.ibm.com/developerworks/edu/dm-dw-dm-0711pauser-i.html

7. PAUSER, M. et al. *Deploy Data Web Services to a WebSphere Community Edition Web Server,* developerWorks tutorial. Originally published March 2008, updated January 2009.
   http://www.ibm.com/developerworks/edu/dm-dw-dm-0803pauser-i.html

8. PULLELA, K. et al. *Transform Data Web Services messages using XSLT in IBM Data Studio Developer*, developerWorks tutorial, July 2008.
   http://www.ibm.com/developerworks/edu/dm-dw-dm-0807pullela-i.html

9. RODRIGUES, V, et al. *Getting Started with pureQuery.* DB2 on Campus book series. December 2010.

10. TEOREY, T. S. Lightstone, and T. Nadeau. *Database Modeling & Design: Logical Design,* 4th edition*,* Morgan Kaufmann Press, 2005.
    ISBN: 0-12-685352-5

Contact emails

General DB2 Express-C mailbox: db2x@ca.ibm.com

General DB2 on Campus program mailbox: db2univ@ca.ibm.com

Getting started with IBM Data Studio couldn't be easier. Read this book to:

- **Find out what IBM Data Studio can do for you**
- **Learn everyday database management tasks**
- **Write SQL scripts and schedule them as jobs**
- **Back up and recover DB2 databases**
- **Tune queries and use Visual Explain**
- **Write and debug SQL stored procedures and routines**
- **Convert existing SQL or procedures to Web services**
- **Practice using hands-on exercises**

IBM Data Studio is replacing the DB2® Control Center and other tools for DB2. It is ideal for DBAs, developers, students, ISVs, or consultants because it's easy and free to use. IBM Data Studio can also be used with other data servers such as Informix®, and you can extend Data Studio with additional robust management and development capabilities from IBM to help accelerate solution delivery, optimize performance, protect data privacy, manage data growth, and more.

IBM Data Studio is part of the InfoSphere® Optim™ Data Lifecycle Management solutions from IBM that can help reduce the costs of managing data throughout its lifecycle, while enabling innovative and high performing new development. Get started with IBM Data Studio, and grow from there!

To learn more or download Data Studio, visit ibm.com/software/data/optim/data-studio/

To take online courses, visit db2university.com

To learn more or download DB2 Express-C, visit ibm.com/db2/express

To socialize and watch IBM Data Studio and DB2 videos, visit ChannelDB2.com

This book is part of the DB2 on Campus book series, free ebooks for the community. Learn more at db2university.com

ISBN 978-0-9866283-8-2

9 780986 628382 >

Price: 24.99 USD