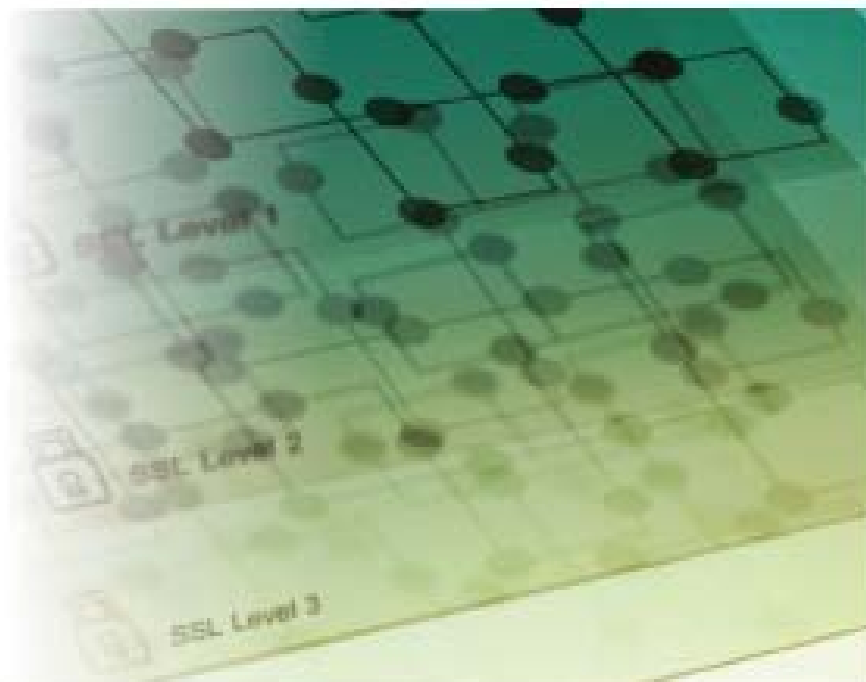


# Getting Started With

**Hands-on  
exercises  
included**

# InfoSphere Data Architect

Ideal for application developers and administrators



by:

**ERIN WILSON**

**SAGAR VIBHUTE**

**CHETAN BHATIA**

**RAHUL JAIN**

**LIVIU PERNIU**

**SHILPA RAVEENDRAMURTHY**

**ROBERT SAMUEL**

**DB2 ON CAMPUS** BOOK SERIES



GETTING STARTED WITH

# InfoSphere Data Architect

ERIN WILSON, SAGAR VIBHUTE, CHETAN BHATIA,  
RAHUL JAIN, LIVIU PERNIU,  
SHILPA RAVEENDRAMURTHY, ROBERT SAMUEL

---

FIRST EDITION

**First Edition (June 2011)**

**© Copyright IBM Corporation 2011. All rights reserved.**

IBM Canada  
8200 Warden Avenue  
Markham, ON  
L6G 1C7  
Canada

---

## 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.

## 6 Getting started with InfoSphere Data Architect

---

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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

### COPYRIGHT LICENSE:

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

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## 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](http://www.ibm.com/legal/copytrade.shtml)" at [www.ibm.com/legal/copytrade.shtml](http://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

<b>Preface .....</b>	<b>13</b>
Who should read this book? .....	13
How is this book structured? .....	13
A book for the community .....	14
Conventions .....	14
What's next? .....	15
<b>About the Authors.....</b>	<b>17</b>
<b>Contributors .....</b>	<b>19</b>
<b>Acknowledgments .....</b>	<b>20</b>
<b>PART I – OVERVIEW AND SETUP .....</b>	<b>21</b>
<b>Chapter 1 – Introduction to IBM InfoSphere Data Architect .....</b>	<b>23</b>
1.1 What is IBM InfoSphere Data Architect? .....	23
1.2 System requirements .....	25
1.3 Obtaining DB2 Express-C.....	25
1.4 Obtaining InfoSphere Data Architect .....	25
1.5 Installing InfoSphere Data Architect .....	26
1.6 Applying the license to IBM InfoSphere Data Architect .....	29
1.7 Launching IBM InfoSphere Data Architect .....	32
1.7.1 Touring the workbench .....	34
1.7.2 Touring the Data Perspective and its views .....	38
1.7.3 Manipulating views .....	39
1.7.4 Resetting the default views for a perspective .....	40
1.8 Exercises .....	40
1.9 Summary.....	41
1.10 Review questions.....	41
<b>Chapter 2 – Data Modeling Overview .....</b>	<b>43</b>
2.1 The data model design life cycle .....	43
2.2 Organizing the data model.....	45
2.3 Creating the student information management system .....	45
2.4 Summary.....	46
2.5 What's next? .....	46
<b>PART II – MODELING YOUR DATA .....</b>	<b>47</b>
<b>Chapter 3 – Logical Data Modeling .....</b>	<b>49</b>
3.1 Logical data modeling: The big picture .....	50
3.2 Creating a logical data model .....	51
3.2.1 Creating a logical data model with the workbench .....	51
3.2.2 Creating entities with the diagram .....	53
3.2.3 Adding relationships .....	61
3.3 Working with glossary models .....	65
3.3.1 Best practices for naming standards and glossary models .....	67

---

3.3.2 Creating a glossary model .....	68
3.4 Working with naming standards .....	70
3.4.1 Analyzing to check compliance with naming standards .....	71
3.5 Constraints .....	72
3.6 Exercise .....	72
3.7 Summary .....	74
3.8 Review questions .....	74
<b>Chapter 4 – Domain Models .....</b>	<b>77</b>
4.1 Domain models .....	77
4.1.1 Creating a blank domain model .....	78
4.1.2 Atomic domains .....	79
4.1.3 List domains and union domains .....	81
4.2 Associating domain model elements with logical data model elements .....	81
4.3 Exercise .....	83
4.4 Summary .....	83
4.5 Review questions .....	83
<b>Chapter 5 – Physical Data Modeling .....</b>	<b>85</b>
5.1 Physical data modeling: The big picture .....	86
5.2 Creating a physical data model from scratch .....	87
5.3 Transforming a logical data model to a physical data model .....	87
5.4 Working on your physical data model .....	90
5.4.1 Anatomy of your model .....	90
5.4.2 Storage modeling in DB2 .....	92
5.5 Refining the physical data model .....	95
5.5.1 Rearranging columns in a physical data model .....	95
5.5.2 Creating roles within the physical data model .....	96
5.5.3 Adding a user ID to the physical data model .....	98
5.5.4 Validating the physical data model .....	98
5.6 DDL generation .....	100
5.6.1 Generating the DDL script from the database object .....	100
5.7 Exercise .....	103
5.8 Summary .....	104
5.9 Review questions .....	104
5.10 What's next? .....	104
<b>PART III – ITERATIVE DESIGN: REPORTING, REVISING, AND ANALYZING .....</b>	<b>105</b>
<b>Chapter 6 – Generating Reports, Importing, and Exporting .....</b>	<b>107</b>
6.1 Reporting, importing, and exporting: The big picture .....	108
6.2 An insight into reporting .....	109
6.3 Generating a BIRT report .....	109
6.3.1 Generating a basic physical data model report .....	109
6.3.2 Setting up the reporting environment .....	110
6.3.3 Adding data objects to a report .....	112
6.3.4 Grouping data in a report .....	116

---

6.3.5 Adding dynamic text to a report.....	119
6.3.6 Generating a report configuration from a template.....	121
6.4 Generating XSLT reports.....	122
6.5 Importing and exporting with IBM InfoSphere Data Architect.....	123
6.5.1 Exporting with the workbench.....	123
6.5.2 Importing a data model with the workbench.....	124
6.6 Exercise .....	125
6.7 Summary.....	126
6.8 Review questions.....	126
<b>Chapter 7 – Reverse-Engineering .....</b>	<b>129</b>
7.1 Reverse-engineering: The big picture.....	130
7.2 Reverse-engineering with the workbench.....	131
7.2.1 Reverse-engineering from DDL.....	131
7.2.2 Reverse-engineering from a database .....	133
7.3 Making changes to the new physical data model .....	134
7.4 Compare and merge your changes .....	135
7.4.1 Comparing and merging changes with the database .....	136
7.4.2 Advantages of the compare and merge functions.....	140
7.5 Exercise .....	140
7.6 Summary.....	141
7.7 Review questions.....	141
<b>Chapter 8 – Model Mapping and Discovery .....</b>	<b>143</b>
8.1 Mapping models: The big picture.....	143
8.1.1 Managing metadata with mapping models .....	144
8.1.2 Further managing naming standards with mapping models.....	145
8.2 Building mappings within the workbench.....	146
8.2.1 Creating a blank mapping model.....	146
8.2.2 Adding mappings to mapping model .....	147
8.3 Types of mapping .....	152
8.4 Adding expressions and filters to the mapping model .....	153
8.5 Generate scripts that you can deploy .....	155
8.6 Export mapping models in CSV format.....	155
8.7 Exercise .....	156
8.8 Summary.....	156
8.9 Review questions.....	156
<b>Chapter 9 – Analyzing Data Models .....</b>	<b>159</b>
9.1 Analyzing data models: The big picture.....	159
9.2 Analyzing data models with the workbench.....	159
9.2.1 Analyzing logical data models with the workbench .....	159
9.2.2 Analyzing physical data models with the workbench .....	160
9.2.3 Fixing errors and warnings in the Problems view .....	161
9.3 Modifying the preferences for model analysis .....	161
9.4 Summary.....	162

## 12 Getting started with InfoSphere Data Architect

---

9.5 Exercise .....	162
9.6 Review questions.....	162
<b>Chapter 10 – The Data Management Life Cycle .....</b>	<b>165</b>
10.1 Managing your data .....	165
10.1.1 The data management life cycle.....	166
10.1.2 Integrating IBM InfoSphere Data Architect with other products .....	167
10.1.3 Shell-sharing with other Eclipse-based products .....	168
<b>References.....</b>	<b>171</b>
<b>Resources.....</b>	<b>171</b>
Web sites .....	171
Books .....	173
Contact emails .....	173

---

## 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 fundamentals of data modeling using IBM InfoSphere® Data Architect, an Eclipse-based tool that can help you create data models for various data servers. By using the IBM InfoSphere Data Architect interface, you can design and deploy data models to a number of environments, and you can even integrate it with other Eclipse-based products.

### How is this book structured?

The book is structured as follows:

- *Chapter 1* introduces you to IBM InfoSphere Data Architect and gets you up and running with the InfoSphere Data Architect workbench (user interface).
- *Chapter 2* introduces you to the basic concepts of data modeling and the project that you will complete as you work through the exercises in the book.
- *Chapters 3, 4, and 5* walk you through the data modeling process:
  - *Chapter 3* teaches you about logical data modeling, and it shows you how to start creating your data models. You learn about entities, attributes, relationships, glossary models, and naming standards.
  - *Chapter 4* helps you get familiar with domain models. In particular, you will learn how to create unique data types that can help you specify what data should be masked to keep personal information private.
  - *Chapter 5* introduces you to physical data modeling. In this chapter, you transform your existing logical data model into a new physical data model, which you will then use to generate a DDL script that you can use to actually deploy the data model.
- *Chapters 6, 7, 8, and 9* acquaint you with the iterative design process:
  - *Chapter 6* walks you through the process of creating reports within IBM InfoSphere Data Architect. You learn how to draft both BIRT and XSLT reports to share with your larger data modeling team.
  - *Chapter 7* describes how reverse-engineering works within the workbench. You learn how to create physical data models from DDL scripts and use

existing database connections so that you can make changes and deploy them to the server.

- *Chapter 8* introduces mapping models and how they help you integrate different data models and data sources.
- *Chapter 9* covers how to analyze your data models to ensure that they are valid, in order to ensure that they conform to common best practices and design standards or do not cause errors once the models are deployed to the server.
- *Chapter 10* describes how IBM InfoSphere Data Architect fits in with the greater data management capabilities from IBM, and how you can integrate this product with other IBM offerings to further design, develop, and manage your data models throughout the entire data life cycle.

Exercises are provided with most chapters. There are also review questions in each chapter to help you learn the material.

## 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](mailto:db2univ@ca.ibm.com) with the subject "IBM InfoSphere Data Architect 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.

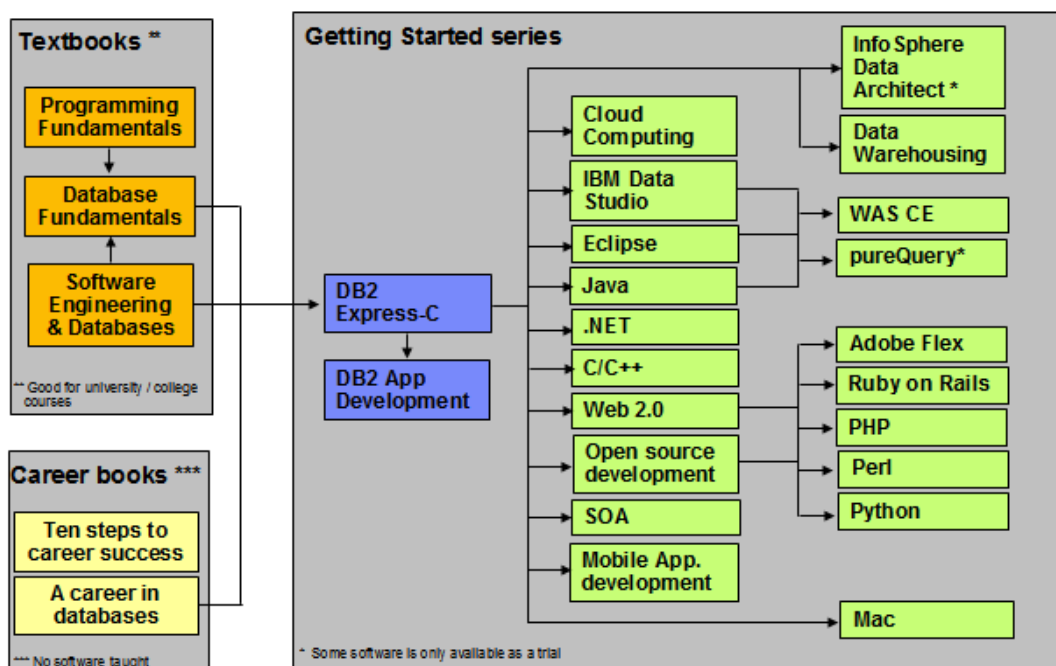
Italics are also used for variable names in the syntax of a command or statement. If the variable name has more than one word, it is joined with an underscore. For example:  
**CREATE TABLE *table\_name***

## 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 IBM Data Studio for DB2*

The following figure shows all the different eBooks in the DB2 on Campus book series available for free at [ibm.com/db2/books](http://ibm.com/db2/books)



The DB2 on Campus book series





---

## About the Authors

**Erin Wilson** is an information developer working at IBM's Silicon Valley Laboratory. As information development lead for InfoSphere Data Architect, she works to document the information needed most by data architects. She has worked with several Eclipse-based products in the InfoSphere Optim™ Data Lifecycle portfolio, specializing in data modeling and warehousing, and she is particularly knowledgeable about DB2-based environments. In addition to product documentation, she has contributed to and narrated for product demos in the data lifecycle portfolio. Before joining IBM, Erin graduated from Purdue University with a degree in Professional Writing. A lifelong lover of computers and technology, she spends her spare time learning more about programming and web development.

**Sagar Vibhute** began his career in 2008 with IBM. Presently he is a part of the JCC development team with the India Software Lab (ISL) in Bangalore. He has previously worked as a part of the Continuing Engineering team for InfoSphere Data Architect. He holds a Masters Degree in Information Technology from IIIT-Bangalore. In his spare time he likes to play the guitar or cycle through the countryside.

**Chetan Bhatia** has been working with IBM since 2008. He is currently working as a developer for InfoSphere Data Architect. He has a Masters in Computer Applications and has around 11+ years of experience in the software development field. He has worked on Web development platforms and is currently working with Eclipse Framework-based product development. He enjoys learning new technologies and new gadgets. He also enjoys swimming and reading.

**Rahul Jain** started his professional career with IBM in June 2008. He is a software developer and currently working as a part of the InfoSphere Data Architect development team in ISL Bangalore. Prior to that, he was working with the Continuing Engineering team for InfoSphere Data Architect in ISL Bangalore. He completed his Master Degree in Information Technology from IIIT-Bangalore and Bachelor Degree in Chemical Engineering. His favorite pastime is listening to music and driving his car on the highways.

**Liviu Perniu** is an Associate Professor in the Automation Department at Transilvania University of Brasov, Romania, teaching courses in the area of Data Requirements, Analysis, and Modeling. He is an IBM 2006 Faculty Award recipient as part of the Eclipse Innovation Awards program, and also one of the authors of *Database Fundamentals* book which is also part of the DB2 on campus book series.

**Shilpa Shree R.** is a BE graduate in the branch of Electronics and Communications. She has 6 years of IT experience in Java and J2EE. She is currently working as a System Analyst at iGate Global Solution Ltd.

**Pauljayam Sp Robertsamuel** has been with IBM for more than two years. He has a degree in Physiotherapy [Rehabilitation Medicine]. However, he changed interests and now is currently positioned as a Level 3 product support engineer for InfoSphere Data Architect.

His domain experience also includes Geographical Information Systems. His hobbies include reading and swimming.

## Contributors

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

Contributor	Company/University	Position/Occupation	Contribution
Yun Feng Bai	IBM China Software Development Laboratory	Staff software engineer, InfoSphere Data Architect	Technical review
Raul F. Chong	IBM Canada Labs – Toronto, Canada	Senior DB2 Program Manager	DB2 on Campus Book Series overall project coordination, editing, formatting, and review of the book.
Don Clare	IBM Silicon Valley Laboratory	Software developer, InfoSphere Data Architect	Technical review
Steve Corcoran	Aviva UK Health – Eastleigh, United Kingdom	Data modeler	Technical and content review
Joe Forristal	IBM – Ireland	Business Intelligence Analyst	Technical review
Leon Katsnelson	IBM Toronto Lab	Program Director, IBM Data Servers	Technical review
Hemant Kowalkar	IBM Silicon Valley Laboratory	Software developer, InfoSphere Data Architect	Technical review
Tao Li	IBM China Software Development Laboratory	Staff software engineer, InfoSphere Data Architect	Technical review
Wei Liu	IBM Silicon Valley Laboratory	Software developer, data tools	Technical review
Diem Mai	IBM Silicon Valley Laboratory	Software developer (installation), Data Studio tools	Technical review
Lu Qiu	IBM China Software Development	Staff software engineer, InfoSphere Data Architect	Technical review

	Laboratory		
Robin Raddatz	IBM Austin	Software developer, SQL and XQuery tools	Technical review
Neil Wang	IBM	Software developer, InfoSphere Data Architect	Technical review
Minghua Xu	IBM Silicon Valley Laboratory	Software developer, InfoSphere Data Architect	Technical review
Joseph Yeh	IBM Silicon Valley Laboratory	Software developer, InfoSphere Data Architect	Technical review

## Acknowledgments

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

**Jin Leem**, IBM Silicon Valley Laboratory, who designed the graphic to explain logical data modeling in Chapter 3.

**Natasha Tolub** for designing the cover of this book.

**Susan Visser** for assistance with publishing this book.

**Kathryn Zeidenstein** and the rest of the team that wrote the *Getting Started with IBM Data Studio for DB2* book, whose work provided the framework for Chapter 1 of this book.

## **PART I – OVERVIEW AND SETUP**



# 1

## Chapter 1 – Introduction to IBM InfoSphere Data Architect

IBM InfoSphere® Data Architect is an enterprise-level data modeling tool. It is a comprehensive development environment that you can use to model diverse and distributed data assets and also to find and establish relationships between those assets.

In this chapter you will learn about the following concepts:

- What is IBM InfoSphere Data Architect?
- How to use the product
- System requirements
- How to install IBM InfoSphere Data Architect

### 1.1 What is IBM InfoSphere Data Architect?

Businesses today own vast amount of data and decisions are made based on data. The challenge for IT is clear: provide understanding of the data, improve data quality and consistency, and keep data design aligned with business intent and requirements. IBM InfoSphere Data Architect is a collaborative data design solution that helps you discover, model, relate, and standardize diverse and distributed data assets. Some of the key features and benefits of IBM InfoSphere Data Architect are listed below:

- InfoSphere Data Architect discovers the structure of heterogeneous data sources by examining and analyzing the underlying metadata. Using an established Java Database Connectivity (JDBC) connection to the data sources, InfoSphere Data Architect explores their structures using native queries. With the user interface, users can easily browse through the hierarchy of data elements, facilitating an understanding of detailed properties for every element.

- InfoSphere Data Architect can create logical, physical, and domain models for DB2®, Informix®, Oracle, Sybase, Microsoft SQL Server, MySQL, and Teradata. Elements from logical and physical data models can be visually represented in diagrams using Information Engineering (IE) notation. Alternatively, physical data model diagrams can use the Unified Modeling Language (UML) notation. InfoSphere Data Architect enables data professionals to create physical data models from scratch, from logical models using transformation or from the database using reverse-engineering.

IBM InfoSphere Data Architect can also create and work with both logical and physical multidimensional data models. As with logical and physical data models, you can create these models from scratch or reverse-engineer them from existing data sources.

**Note:**

Multidimensional modeling is not covered in this book. If you want to learn more conceptual information about multidimensional modeling visit the IBM InfoSphere Data Architect information center:  
[http://publib.boulder.ibm.com/infocenter/rdahelp/v7r5/topic/com.ibm.datatools.dimensiona.ui.doc/topics/c\\_ida\\_dm\\_container.html](http://publib.boulder.ibm.com/infocenter/rdahelp/v7r5/topic/com.ibm.datatools.dimensiona.ui.doc/topics/c_ida_dm_container.html)

- Most development projects are iterative in nature, so it is important to be able to design incrementally and manage changes and their impact seamlessly. InfoSphere Data Architect allows users to do just that. Impact analysis lists all of the dependencies on the selected data elements. Advanced synchronization technology compares two models, model to database or two databases. Changes can then be promoted within and across data models and data sources. Use with Optim™ Database Administrator to manage complex DB2 changes and data migrations without disruption.
- InfoSphere Data Architect enables architects to define and implement standards that increase data quality and enterprise consistency for naming, meaning, values, relationships, privileges, privacy, and traceability. Define standards once and associate them with diverse models and databases. Built-in, extensible, rules-driven analysis verifies compliance to naming, syntax, normalization, and best practices standards for both models and databases.
- Whether you are working on a small team where each member plays multiple roles or in a large distributed team with clearer delineation of responsibilities, you can use InfoSphere Data Architect as plug-in function to a shared Eclipse instance or share artifacts through standard configuration management repositories like Rational® Clear Case or Concurrent Versions System (CVS).



## 1.2 System requirements

Make sure that your computer meets the system requirements before you install IBM InfoSphere Data Architect V7.5.3. You can find the system requirements for IBM InfoSphere Data Architect at the following URL:

<http://www-01.ibm.com/support/docview.wss?uid=swg27019867>

This document includes hardware, software, and data source requirements.

**Note:**

If you are extending an existing Eclipse environment, you must use version 3.4.2 of Eclipse and the IBM JDK 1.6 SR 7.

## 1.3 Obtaining DB2 Express-C

This book uses DB2 Express-C, the free version of DB2, for most exercises. To obtain and install DB2 Express-C:

1. [Download](#) DB2 Express-C from [ibm.com/db2/express](http://ibm.com/db2/express)
2. Install DB2 Express-C. You can review videos showing how to do this in [db2university.com](http://db2university.com)
3. Set up the SAMPLE database included with DB2 so that you can deploy the data models that you create as a result of this book.

**Note:**

For more information, refer to the eBook [Getting started with DB2 Express-C](#) which is part of this book series.

## 1.4 Obtaining InfoSphere Data Architect

If you are a student, instructor, or researcher, you can get a free copy of IBM InfoSphere Data Architect through the IBM Academic Initiative. You must be affiliated with a university, and you must meet one of the following criteria:

- You are a student that is studying data architecture
- You are an instructor that is teaching data modeling and architecture
- You want to use the product to conduct research

To learn more about the IBM Academic Initiative, visit the following URL:

<https://www.ibm.com/developerworks/university/academicinitiative/>

To buy IBM InfoSphere Data Architect, visit the product page and click the **Ready to buy** button to download the product from Passport Advantage:

<http://www-01.ibm.com/software/data/optim/data-architect/>

You can find the Passport Advantage® part numbers that you need in the following download document. On this page, select the operating system that you will use to run the product:

<http://www-01.ibm.com/support/docview.wss?uid=swg24028103>

If you want to try the product for free for 30 days, you can download a trial of IBM InfoSphere Data Architect from ibm.com. Visit the following URL to download a trial copy of the product:

<http://www.ibm.com/developerworks/downloads/r/rda/index.html>

## 1.5 Installing InfoSphere Data Architect

IBM InfoSphere Data Architect can be installed by using the Launchpad interface, or silently, which means that you create a response file of your chosen installation options, then run that response file. Silent installations are useful for larger installations in which installation must be pushed out to many machines.

This chapter focuses on the installation that uses the Launchpad interface. It assumes you do not have the IBM Installation Manager installed. This means that installing IBM InfoSphere Data Architect starts in the 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.

### Note:

As explained in the Data Security, Privacy, and Lifecycle Management information center ([http://publib.boulder.ibm.com/infocenter/idm/docv3/index.jsp?topic=/com.ibm.datatools.rda.install.doc/topics/c\\_plan\\_imover.html](http://publib.boulder.ibm.com/infocenter/idm/docv3/index.jsp?topic=/com.ibm.datatools.rda.install.doc/topics/c_plan_imover.html)), 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. IBM 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.

This chapter assumes that you have administrative privileges on the computer on which you are installing the product.

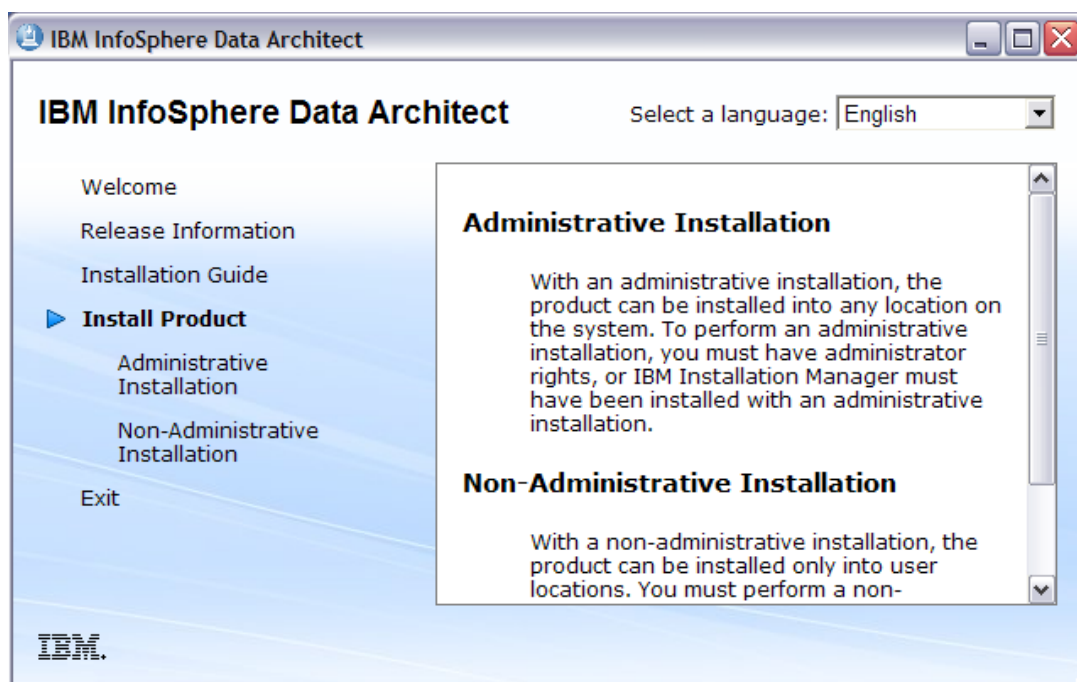
Follow these steps to install the IBM InfoSphere Data Architect product:

1. After you extract the contents of the compressed package, start the launchpad:

- Windows: Run the **setup.exe** file, located in the directory where you extracted the image.
- Linux or UNIX: Run the **setup** command from the root path where you extracted the image.

The launchpad opens.

2. In the left pane, select *Install Product*, as shown in *Figure 1.1*. The product installation page opens. You can now select which type of install that you want to run.

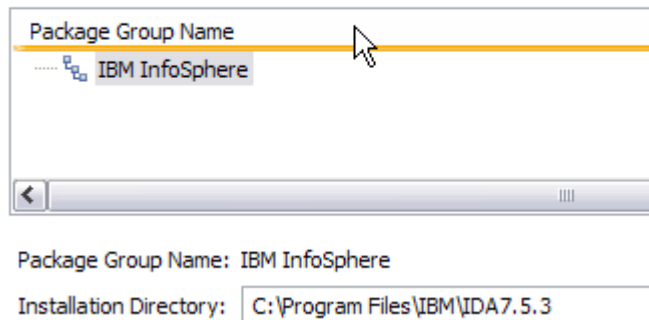


**Figure 1.1 – Click Install Product to launch the Installation Manager**

3. Select the *Administrative Installation* option. IBM Installation Manager opens.
4. Select the packages that you want to install. For this exercise, select the *IBM InfoSphere Data Architect* package. Click *Next*.
5. Review and accept the license agreement. Click *Next*.
6. Create a new package group and specify where common components will be installed:
  - a. Select the *Create a new package group* radio button as shown in *Figure 1.2*. Because you are installing the product on a machine that does not include any

existing package groups, you should select this option.

- ☐ Use the existing package group
- ☒ Create a new package group



Package Group Name: IBM InfoSphere

Installation Directory: C:\Program Files\IBM\IDA7.5.3

**Figure 1.2 – Create a new package group for IBM InfoSphere Data Architect**

- b. Specify the directory where the shared resources and Installation Manager will be installed. For the purpose of this exercise, you can use the default settings, but remember that you should choose a drive that has more space than you think you need just for IBM InfoSphere Data Architect in case you decide to shell-share with other Eclipse-based products.

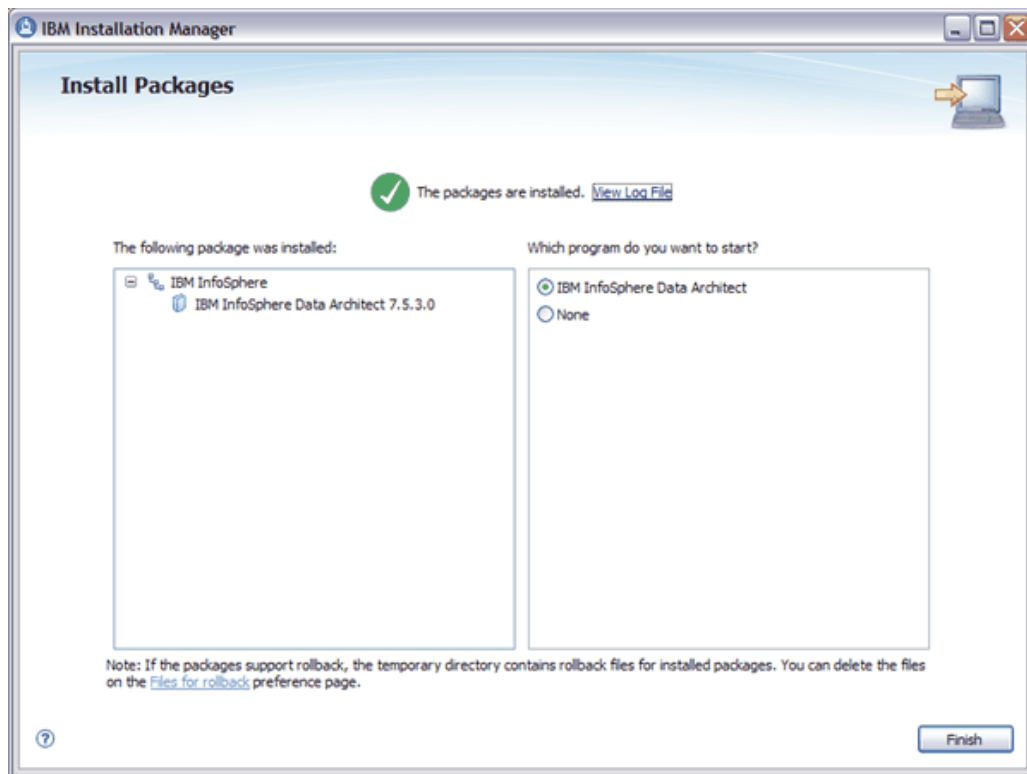
**Note:**

If you install other products that make use of Installation Manager, you cannot choose a directory for the shared resources.

- c. Click *Next* to proceed to the next screen.
7. If you already have Eclipse 3.4.2 on your machine, you can choose to *extend* that IDE instead of installing an additional copy. This adds the functions of the newly-installed product, but maintains your IDE preferences and settings. For this exercise, do not change any settings. Click *Next* to proceed to the next screen.
  8. Select any additional language translations that you want to install, then click *Next*.
  9. Select the features that you want to install. In this case, make sure that all of the check boxes are selected, then click *Next*.
  10. Specify how product documentation and help topics are accessed. By default, all help documentation is located in the IBM InfoSphere Data Architect information center. You can specify that all help documents are downloaded at installation time or that all help will be accessed from a central server. For this exercise, select the *Access help from the web* option, then click *Next*.

11. On the summary page, review your choices, then click *Install*.

IBM Installation Manager begins to install the product. The progress bar may pause occasionally; do not interrupt the progress of the installation or exit the installation until the product is fully installed. When the product is installed, you see the screen in *Figure 1.3*.



**Figure 1.3 – Congratulations! The product was successfully installed**

12. On the final page of the Install Packages wizard, select the *None* option to specify that you do not want to start InfoSphere Data Architect. If you are using a license, you will start IBM InfoSphere Data Architect after you apply the license to your product. Click *Finish* to exit the installation wizard.

13. Close the launchpad window by clicking the *Exit* link in the left pane.

IBM InfoSphere Data Architect is installed. If you have a license, you must now activate it in order to have access to the product after the 30-day trial period expires.

## 1.6 Applying the license to IBM InfoSphere Data Architect

InfoSphere Data Architect makes use of three different license types:

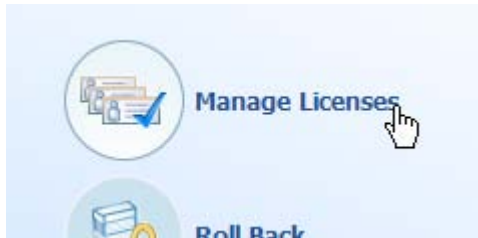
- ⤴ **Trial license:** You can download the product and use it with full functionality for free for 30 days. Use this option to evaluate the product to make sure it meets your data modeling needs.

If you use this option, you do not need to activate a license. The product is already bundled with a trial license.

- ⤴ **Floating licenses:** You can set up a Rational License server that stores and manages your floating licenses throughout your organization. To learn more about floating licenses and how to use them with InfoSphere Data Architect, see the following technote: <http://www.ibm.com/support/docview.wss?uid=swg21468395>
- ⤴ **Individual licenses:** You apply one activation kit to each installation of the product. These steps are outlined below.

If you have a floating or individual license that you would like to apply to the product, perform the following steps:

1. Open IBM Installation Manager:
  - Windows: Click *Start -> All Programs -> IBM Installation Manager -> IBM Installation Manager*.
  - Linux or UNIX: Open a terminal window with root privileges, then change to the directory for IBM Installation Manager. By default, this location is `/opt/IBM/InstallationManager/eclipse`. Run the **IBMIM** program.
2. Click the *Manage Licenses* button, as shown in *Figure 1.4*:

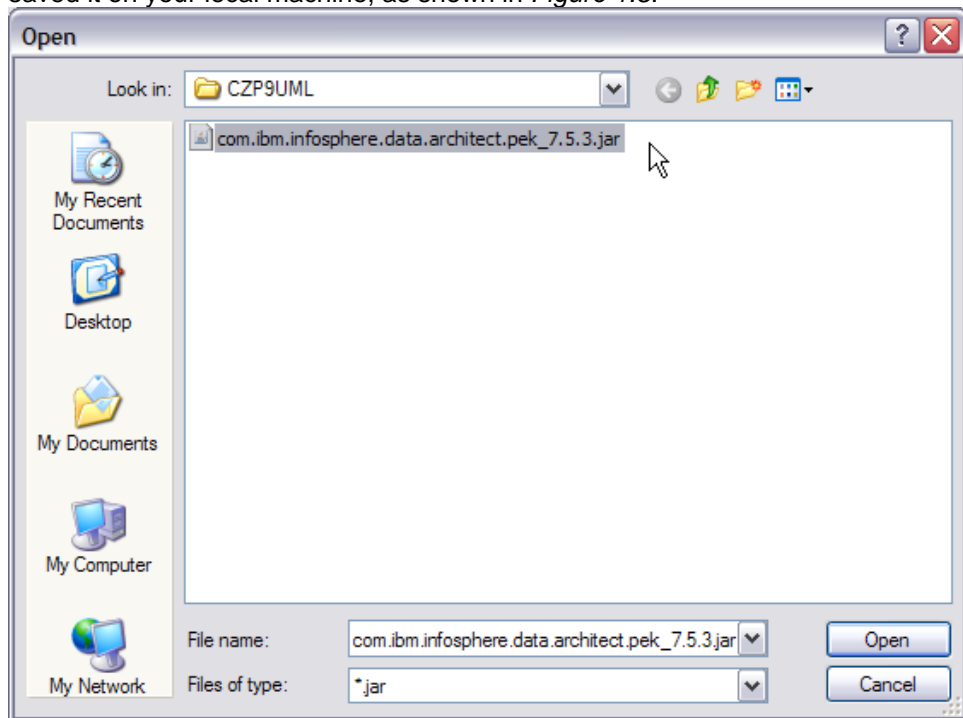


**Figure 1.4 – Select Manage Licenses in IBM Installation Manager**

The Manage Licenses page opens.

3. Import the product activation kit that you downloaded for IBM InfoSphere Data Architect:
  - a. Select *IBM InfoSphere Data Architect 7.5.3.0* from the list of installed packages.
  - b. Select the *Import Product Activation Kit* radio button, then click *Next*.

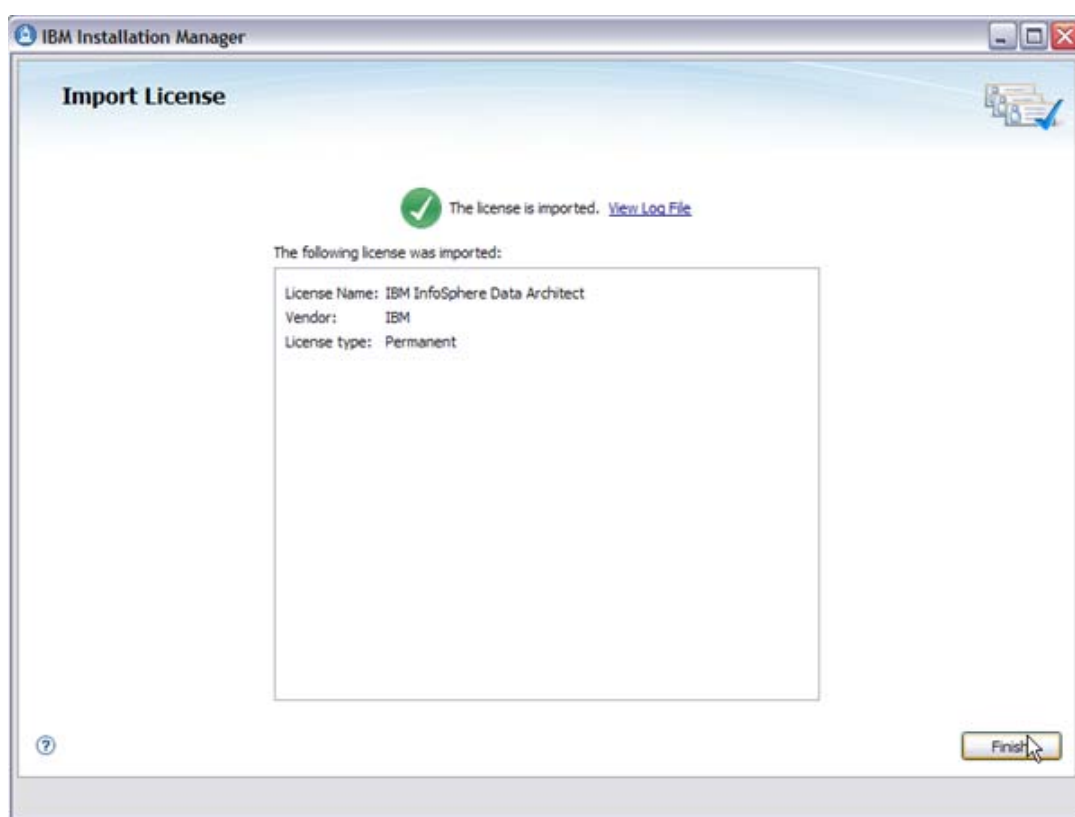
- c. Browse for and select the product activation kit **.JAR** file from where you saved it on your local machine, as shown in *Figure 1.5*:



**Figure 1.5 – Selecting the product activation kit**

- d. Click *Next* to proceed to the next page.  
e. Accept the license agreement, then click *Next*.  
f. On the Summary page, click *Finish*.

The license is applied to the product. The activation screen appears as shown in *Figure 1.6*:



**Figure 1.6 – The license is activated**

You can close IBM Installation Manager and start IBM InfoSphere Data Architect for the first time.

## 1.7 Launching IBM InfoSphere Data Architect

Launch IBM InfoSphere Data Architect via one of the following methods:

- Windows: Click *Start -> All Programs -> IBM InfoSphere -> IBM InfoSphere Data Architect 7.5.3.0*.
- Linux or UNIX: In a terminal window, type the following command:  
**`.product_install_directory/eclipse`**

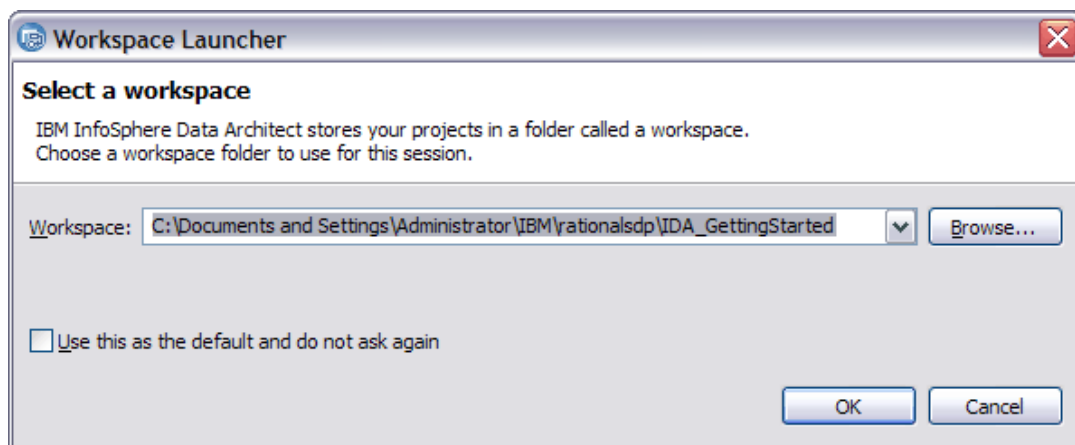
where *product\_install\_directory* is the directory in which you installed the product.

When you open the product, you can specify what workspace you want to use. Specify **IDA\_GettingStarted** as the name of your workspace, as shown in *Figure 1.7*.



**Note:**

A workspace is a location to save 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.



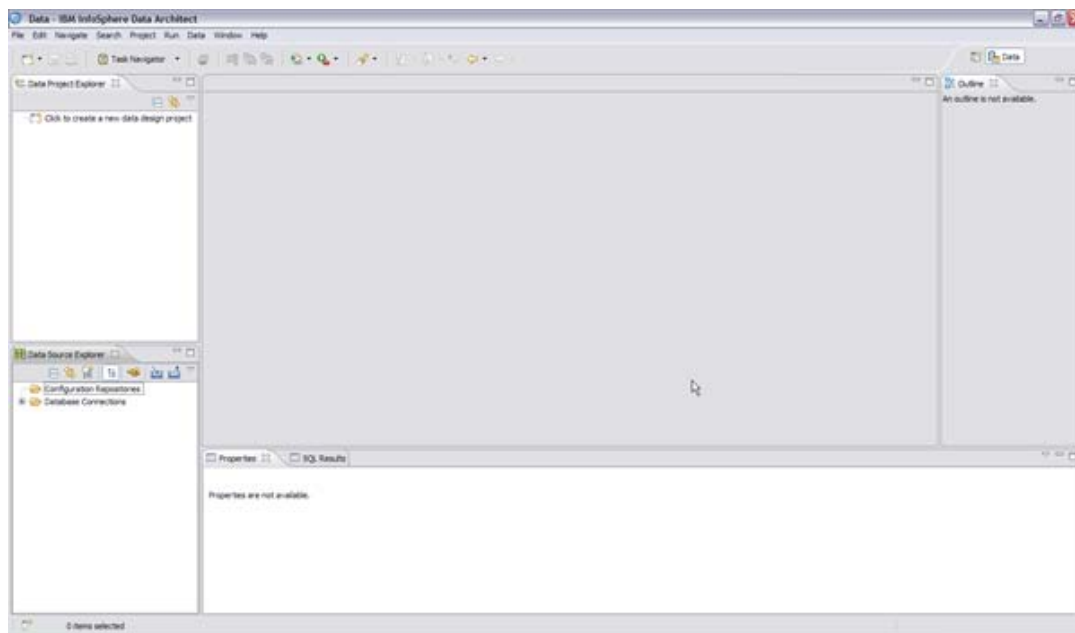
**Figure 1.7 – Selecting a workspace**

IBM InfoSphere Data Architect opens for the first time in the Data perspective, which includes the Task Launcher.

**Note:**

A perspective is an Eclipse concept. A perspective contains views and actions that are associated with particular tasks. A view shows you your resources, which are associated with editors. The default perspective for IBM InfoSphere Data Architect is the Data perspective, as shown in Figure 1.8. You can see the names of the various views there, including the Data Project Explorer and Outline.

## 34 Getting started with InfoSphere Data Architect



**Figure 1.8 – The default Data perspective in IBM InfoSphere Data Architect**

### 1.7.1 Touring the workbench

The term *workbench* refers to the desktop development environment, and is an Eclipse concept. 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.

#### 1.7.1.1 The Task Launcher

When you open an InfoSphere Data Architect workspace for the first time, you see a pane in the main editor view called the Task Launcher.

The Task Launcher is a view designed to help you get started with basic tasks within the workbench. For example, the Task Launcher within IBM InfoSphere Data Architect can help you create a physical data model for the first time.

Use the tabs at the top of the Task Launcher to get started with these tasks. IBM InfoSphere Data Architect has three tabs in the Task Launcher view:

- **Overview:** This tab contains links to general information about the product, as well as general “getting started” tasks, such as connecting to a database.

- *Design*: This tab contains common tasks that help you create data models and design your data. You can also learn more about other common tasks, or follow the links to tutorials on developerWorks.
- *Develop*: This tab contains common tasks that help you develop SQL or XQuery statements and stored procedures, and you can learn how to debug these statements.

The Task Launcher is shown in *Figure 1.9*.



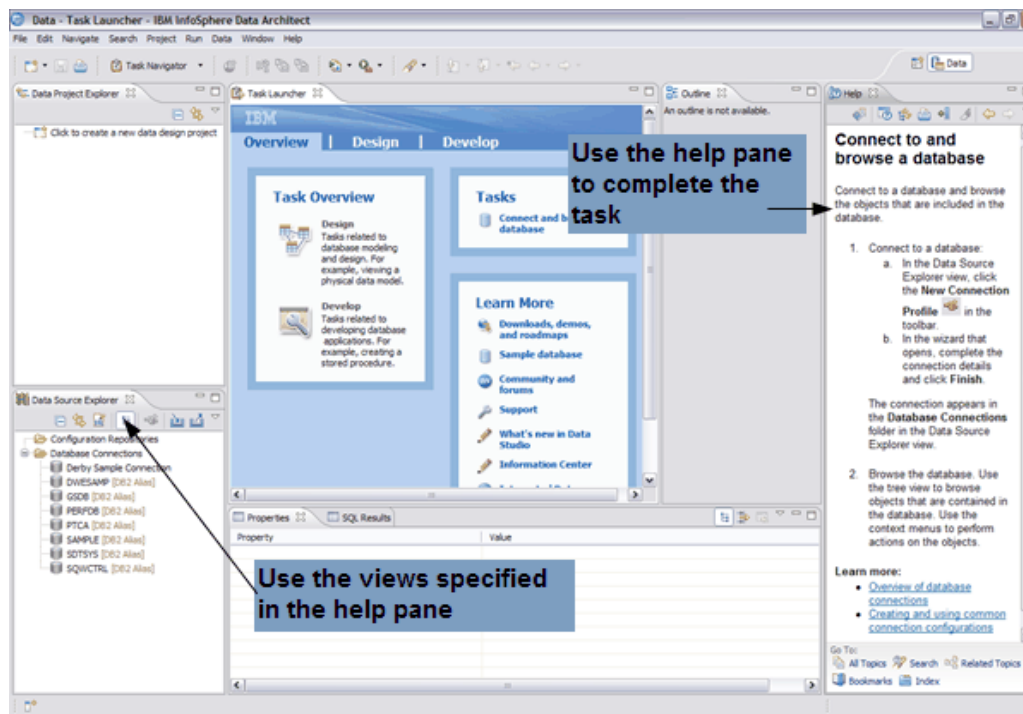
**Figure 1.9 – Getting to know the Task Launcher**

When you open a task in the Task Launcher, the perspective that is associated with that task automatically opens. A help panel also opens. Follow the instructions in the help pane to complete the task.

Let's explore the Task Launcher by connecting to the SAMPLE database connection that is included when you installed DB2.

Open the *Overview* tab of the Task Launcher to complete this task.

1. In the *Tasks* box, click the *Connect and browse a database* link. If the Data perspective is not open, the perspective opens, and the Help view opens to guide you through the task. The changes to the workbench are shown in *Figure 1.10*.



**Figure 1.10 – Completing a task with the Task Launcher**

2. Follow the instructions in the Help panel of the Task Launcher to complete the task and connect to the SAMPLE database that you set up when you installed DB2 Express-C.

Once you have completed the task, the connection appears in the Data Source Explorer view.

The Task Launcher is included with all Eclipse-based products in the Data Security, Privacy, and Lifecycle Management suite of products. If you shell-share with other Eclipse-based products, the tabs and tasks associated with those tabs will change, since you have more capabilities associated with your product environment. Shell-sharing is discussed in detail in Chapter 10, as well as the larger Data Security, Privacy, and Lifecycle Management portfolio of products.

**Note:**

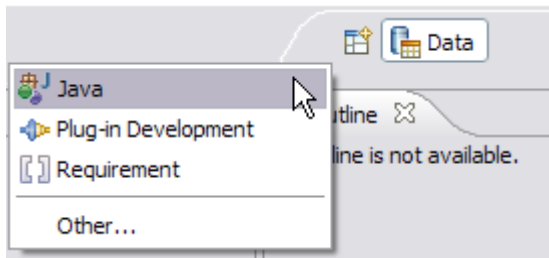
If you do not want to see the Task Launcher when you open the workspace, simply close the view. If you want to open the Task Launcher in the future, simply open it from the main menu: Help -> Optim Task Launcher.

### 1.7.1.2 Perspectives

Each workbench window contains one or more perspectives. *Perspectives* contain views and editors and control what appears in certain menus and toolbars 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 Data perspective.

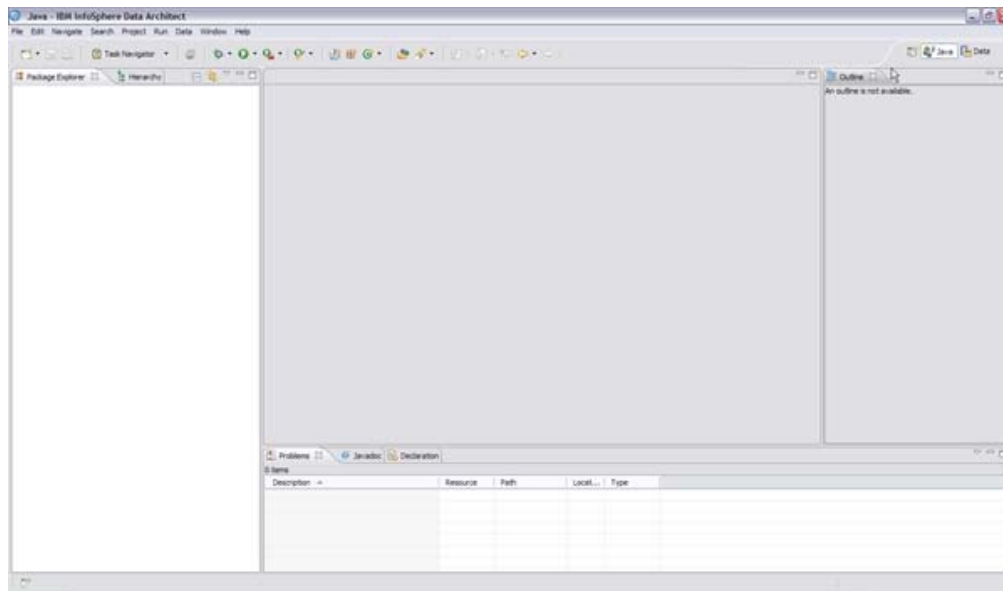
Let's look at the Java perspective for fun.

One way to open a different perspective is to click the icon shown below in *Figure 1.11* and select *Java*. An alternate way to open a perspective is to click on *Window -> Open Perspective*.



**Figure 1.11 – Opening a different perspective**

As you can see by comparing *Figure 1.8* and *Figure 1.12* (below), the Java perspective has a different task focus (Java development) than the Data 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 1.12 – The Java perspective**

Click on the Data perspective to switch back again so that you can learn more about the capabilities of the Data perspective.

**Note:**

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

### 1.7.2 Touring the Data Perspective and its views

Because most of the work you'll do in this book is in the Data perspective, make sure the Data perspective is open.

As we described earlier, *views* are the windows that you see in the workbench, such as Data Source 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.

The views shown in the Data perspective are described in *Table 1.4* below.

View	Description
Data Project Explorer	This view is used by a data architect or a data or information modeler when creating models. It shows data design projects (which you will use to store logical and physical data models) and

	data development projects.
Data Source Explorer	This view allows you to connect to and view databases and the objects that are within them. It automatically displays detected databases, but you can add new database connections. You can reverse-engineer from this view by dragging and dropping data objects from the Data Source Explorer into the Data Project Explorer.
Properties	This view shows the properties of the object that is currently selected in the workspace. For some objects, you can use this view to edit properties, such as making changes to the database objects that are selected in the Data Source Explorer. When an object is highlighted in either a diagram or the Data Project Explorer, the Properties view changes to show the properties of that object.
SQL Results	This view shows the results after you run SQL or XQuery statements. You can also use this view to obtain sample data from a selected table within a database.
Outline	This view shows an outline of a structured file that is currently open in the editor area and lists its structural elements. For example, if you were editing a data diagram, you would see the overall diagram in the Outline view, and when you zoom in or out in the diagram, a shaded box displays to show you where you are in the context of the larger diagram.

Table 1.4 – Views in the default Data perspective

### 1.7.3 Manipulating views

The basic view controls are shown in *Figure 1.13*.

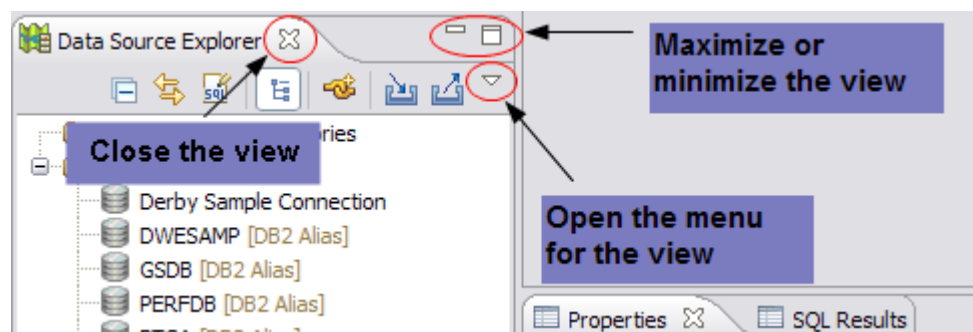


Figure 1.13 – View controls

To close a view, click on the X to the right of the view name as shown in *Figure 1.13*. There's no need to panic if you close a view accidentally. If you close a view, simply go to *Window -> Show View* and select the view that you want to open. If you don't see the view that you want to open, click *Other* and select the view.

#### 1.7.4 Resetting the default views for a perspective

You should play around with the views and perspectives in the workbench. If you're not familiar with Eclipse, it can seem strange to open and close views. If you want to reset the perspective to the default settings, reset the perspective by clicking *Window -> Reset Perspective*.

**Note:**

The Reset Perspective option only resets the current perspective. If you want to change a different perspective, open *Windows -> Preferences*, then select *General -> Perspectives*. Select a perspective and click the *Reset* button. The next time the perspective is opened, it is restored to the default layout.

### 1.8 Exercises

In this set of exercises, you will install IBM InfoSphere Data Architect and get comfortable with the workbench and Eclipse tools.

1. If you have not yet installed DB2 Express-C, install it.
2. If you have not installed InfoSphere Data Architect, install it, following the instructions in this chapter.
3. Familiarize yourself with the workbench. Perform the following tasks:
  - Switch to the Plug-In Development perspective.
  - Switch back to the Data perspective.
  - Close the outline view.
  - Minimize and maximize some of the view windows.
4. Explore the product documentation. The documentation for IBM InfoSphere Data Architect is in the information center at the following URL:  
<http://publib.boulder.ibm.com/infocenter/rdahelp/v7r5/index.jsp>
5. Read the [product overview](#).



6. Watch and complete the lessons in the [Understand the workbench environment](#) tutorial.
7. View the [Introduction to IBM InfoSphere Data Architect](#) demo on developerWorks.

## 1.9 Summary

IBM InfoSphere Data Architect provides tooling support to help you create data models that discover, visualize, relate, and standardize your data. You can create logical, physical, dimensional, and domain models for DB2, Informix, Oracle, Sybase, Microsoft SQL Server, MySQL, and Teradata source systems. It allows for an iterative design process, which lets you update and refine your data models as the needs of the business change, with minimal impact to your production environment.

There is an active discussion forum for IBM InfoSphere Data Architect at the following URL: <http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1796>

This forum can provide informal support.

This chapter also covered how to install the product and navigate the Eclipse workbench. You learned how to open different perspectives and manipulate the views in a perspective.

## 1.10 Review questions

1. What open-source platform is IBM InfoSphere Data Architect built on?
2. What are *perspectives* in an Eclipse-based product?
3. What is the default perspective for IBM InfoSphere Data Architect?
4. In which Eclipse view do the results of SQL operations appear?
5. How do you open a view within a perspective?
6. True or false: You view database connections within the Data Project Explorer view.
7. True or false: The default perspectives, projects, and data objects are unique to each workspace.
8. True or false: You cannot specify which features are installed with the product.
9. True or false: To extend an existing Eclipse environment to work with IBM InfoSphere Data Architect, you must have the same Eclipse and IBM JDK level.



# 2

## Chapter 2 – Data Modeling Overview

When you create data models, you follow a certain process. This book introduces some best practices and concepts to ensure that the models you create within the workbench adhere to good data design principles.

In this chapter, you will learn about the following concepts:

- The data design life cycle
- How data models are typically organized
- The information management system that you will create as a result of this book

**Note:**

For more information about Data Modeling, review the eBook [Database Fundamentals](#) that is part of this book series.

### 2.1 The data model design life cycle

With this book, you will learn how to create data models that can be used to create databases. The data model design life cycle helps you conceptualize and develop your data model, using an iterative design process.

Data modeling is done through two methods of engineering:

- ✧ *Forward-engineering:* Data models and databases are built from scratch, without a pre-existing model to work from. This type of modeling is best done for new systems or when modeling new business processes that are not yet addressed by your current systems.
- ✧ *Reverse-engineering:* Existing data models and databases are used to create models within the workbench. The workbench is capable of creating physical data models from existing databases and schemas, and you can use the transformation tool to create logical data models to further refine your projects.

**Note:**

The primary focus of this book is on forward-engineering. Reverse-engineering is also introduced to provide a full perspective of both data modeling and the capabilities of the product.

More information about reverse-engineering is provided in Chapter 7 of this book.

To develop and deploy databases in the workbench, you should use the following process:

1. **Gather requirements for the project.** Speak with business process owners and determine what information is vital to the business. You can use this information to draft entities or tables and attributes or columns.
2. **Use IBM InfoSphere Data Architect to create a logical data model.** The logical data model is the first part of the data modeling process, where you gather requirements and create entities and attributes. You will eventually use the logical data model to create a physical data model.

As you design the logical data model, you can work with data diagrams to visualize the model and help you show business process owners how information is linked together. You refine this data model over time, improving the design as you speak with business process owners and further define what information must be modeled.

3. **Use IBM InfoSphere Data Architect to transform the logical data model into a physical data model.** Work with the physical data model to draft physical storage, table spaces, indexes, or views. You can also use the physical data model to secure parts of the database that you create by implementing roles and users to deploy on the new database.

As you refine your physical data model, you can continue to work with existing data diagrams that were transformed from the logical data model, or you can create new data diagrams to visually model your storage requirements.

4. **Create a mapping model that defines the relationships between your data models and the target database.** Mapping models contain the information that defines the relationship between the source schemas that you create and their targets. This includes transformations, join conditions, filters, sort conditions, and annotations. Mapping models can find or document the relationships, and you can use a mapping model to generate statements that query or manipulate your data sources by using these relationships.

**Note:**

As a best practice, make sure that your data models are stable before you create mapping

models. If you change relationships or the names of data objects, the mappings that were previously created are destroyed, and you will encounter errors.

5. **Generate DDL that will deploy the database.** Once the database is deployed, applications can be created to connect to and read from the database, or you can use other software to further optimize or read from the database.
6. **Refine your model as the needs change over time.** Business requirements sometimes change or mature, and you must adapt the data models as needed to ensure that you are still obtaining or recording relevant information.

Here are a few other things that you can do with IBM InfoSphere Data Architect:

- Create reports that business process owners can use to understand the data that is stored in the database.
- Import or export data models to and from the workbench.
- Compare and synchronize changes between data models, and analyze the impact that your changes will have on updated data models before you deploy the changes.
- Reverse-engineer from an existing data source.
- Analyze data models to ensure that common data modeling rules are followed and enforced.

## 2.2 Organizing the data model

Data modeling involves structuring and organizing data. Data modeling can define and organize the data and can impose (implicitly or explicitly) constraints or limitations on the data placed within the structure.

## 2.3 Creating the student information management system

You will use this book and the workbench to create a sample student information management system for a university. The student information management system will store information about students, such as student ID number, grades, and address.

You will follow the data design life cycle to design and then implement this student management information system. You will create a logical data model, which is then used to create a physical data model. After the physical data model is created, you can make further updates to the system or deploy it.

The next part of this book focuses on the data modeling part of the data design life cycle. You will create a logical data model, based on requirements from the process owners at a fictional university. Once the logical data model is created, you can transform the data model into a physical data model. After the physical data model is created and you have designed the tables, columns, and other physical data objects, you will generate a DDL script to deploy your physical data model, creating a database. Then, you will create a mapping model, which allows you to integrate your data models with other data models, creating a cohesive database system that can track all aspects of the students' information.

In the third part of the book, you will learn how to generate reports with the workbench. These reports can be used by business process owners to analyze the data that can be found through the design of your data models. Finally, you will learn how to reverse-engineer from an existing database or schema, so that you can make further iterative changes to existing data objects and deploy the changes. Learn how to analyze your data models in the final chapter to ensure that your models are valid before you place them into the production environment.

## **2.4 Summary**

You learned a basic overview of data modeling and how you can use IBM InfoSphere Data Architect to draft and deploy data models. When you create these models, you start with a logical design, and then you can draft the physical design.

The data design process is iterative, and IBM InfoSphere Data Architect makes it easy for you to continue to work with your existing data models and data sources. Reverse-engineering makes it easy for you to model with existing data sources, and you can continually refine existing resources to extend them, improve performance, or respond to changing requirements.

## **2.5 What's next?**

Now that you know the basics of data modeling, you should start to draft a design. Using this book, you will become acquainted with the workbench and how to create complete data models, from start to finish, that you can deploy and refine.

## **PART II – MODELING YOUR DATA**





# 3

## Chapter 3 – Logical Data Modeling

Logical data models are just the first step in the data design life cycle. Logical data models help you capture the requirements of the business, and you start creating the future design of your schemas and databases before you design the physical storage and deploy the model.

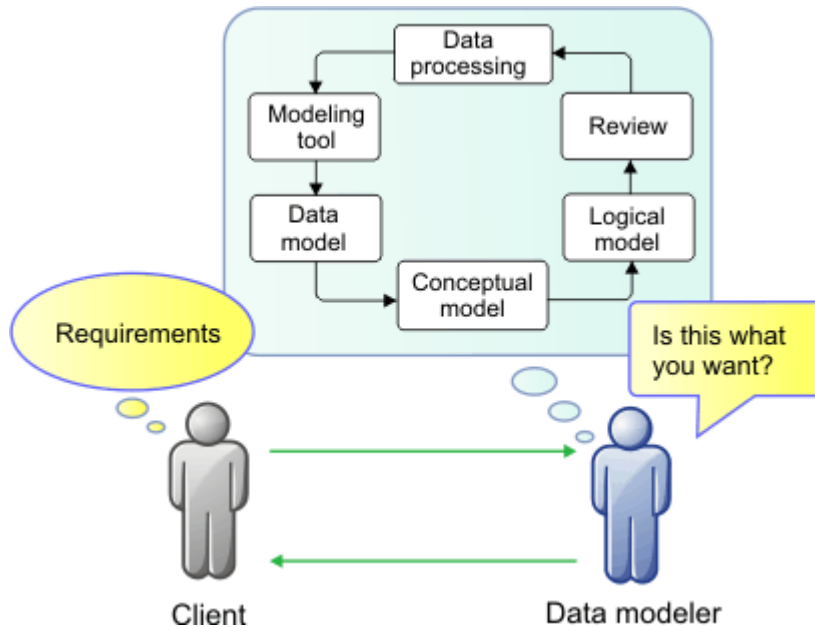
This chapter covers the following concepts:

- How to design a logical data model
- Creating diagrams to visually display the design of your models
- Developing glossary models and naming standards for your organization
- Applying those glossary models and naming standards to your data models

**Note:**

For more information about Logical Data Modeling, review the eBook [Database Fundamentals](#) that is part of this book series.

### 3.1 Logical data modeling: The big picture



**Figure 3.1 – Use InfoSphere Data Architect as your data modeling tool**

When you create a logical data model, you take the first step in developing a database or schema that addresses business needs. A *logical data model* is a model that isn't specific to a database vendor. The logical data model describes things about which an organization wants to collect data, and it documents the relationships between these things. Logical data models are organized hierarchically and contain objects such as packages, entities, attributes, and other relationship objects.

Before you create a logical data model, you must gather the requirements for the model. A business user's description of his or her requirements may be incomplete. Often, business users or clients may give you only some pieces of a whole. You may be given a very vague description of a business process: "The business must track the sales of our products." You, as a data modeler, must understand all of the requirements of the project in order to gather the missing pieces and construct a complete model.

After you gather initial requirements, you present the outcome to the client, preferably by presenting a diagram, and you work with the client to negotiate the relevant and insignificant aspects, as you can see in *Figure 3.1*. After negotiations, you (the data modeler) modify the logical data model and present the revised model to the client. Through this iterative process, you draft and develop a complete logical data model.

The development of the logical data model is introduced in this chapter.

## 3.2 Creating a logical data model

One of the most difficult problems you will encounter as you create a logical data model for a project is to come to agreement on what business process should be modeled. You should work with your client to determine the needs of the business that the project will satisfy. To create a logical data model of a business process, you should use the following process:

1. Analyze the various needs of the business. Start with the most important business process. Document the analysis, so that you and other data modelers can understand the goals and future plans of the project.
2. Work with the project owners to determine what data they want to identify and why the data is important. You can use this to identify preliminary data objects that are common across several processes, data sources, or the various grains of each data object.
3. Meet with project owners to resolve ambiguous requirements.
4. Make a preliminary plan to meet the requirements. Create a conceptual model, drafting the initial design of the logical data model. You will use this plan to draft your data model.

### 3.2.1 Creating a logical data model with the workbench

You meet with business process owners for the university. They ask you to help them create a student information management system that collects student information. After further meetings, you decide to start small. You will create a logical data model that tracks basic student information, such as student ID number, grades, and classes.

To learn how to create a logical data model with the workbench, you will create a small logical data model, named **STUDENT\_INFO\_SYSTEM**.

#### 3.2.1.1 Creating a new data design project to store your models

Before you can create a logical data model, you need to create a new *data design project*, where your files are stored. Use this data design project to store all of the data models for this project. You can have multiple data design projects within a workspace, and each data design project can contain multiple data models.

To create a new data design project:

1. Click on *File -> New -> Data Design Project*. The New Data Design Project window opens.

#### Note:

Since your workspace is more than likely empty, you can also click the *Click to create a new data design project* link. However, this link is not available once you create your first data design project.

2. In the *Project* field, specify a descriptive name for your project. Type **University Info System**.
3. Click *Finish*.

The University Info System data design project is created in the Data Project Explorer. In the data design project, there are folders that store the files in your project.

### 3.2.1.2 Creating your first logical data model

Currently, the project is empty, so you must create a new logical data model. You will use this logical data model to start creating the student information management system.

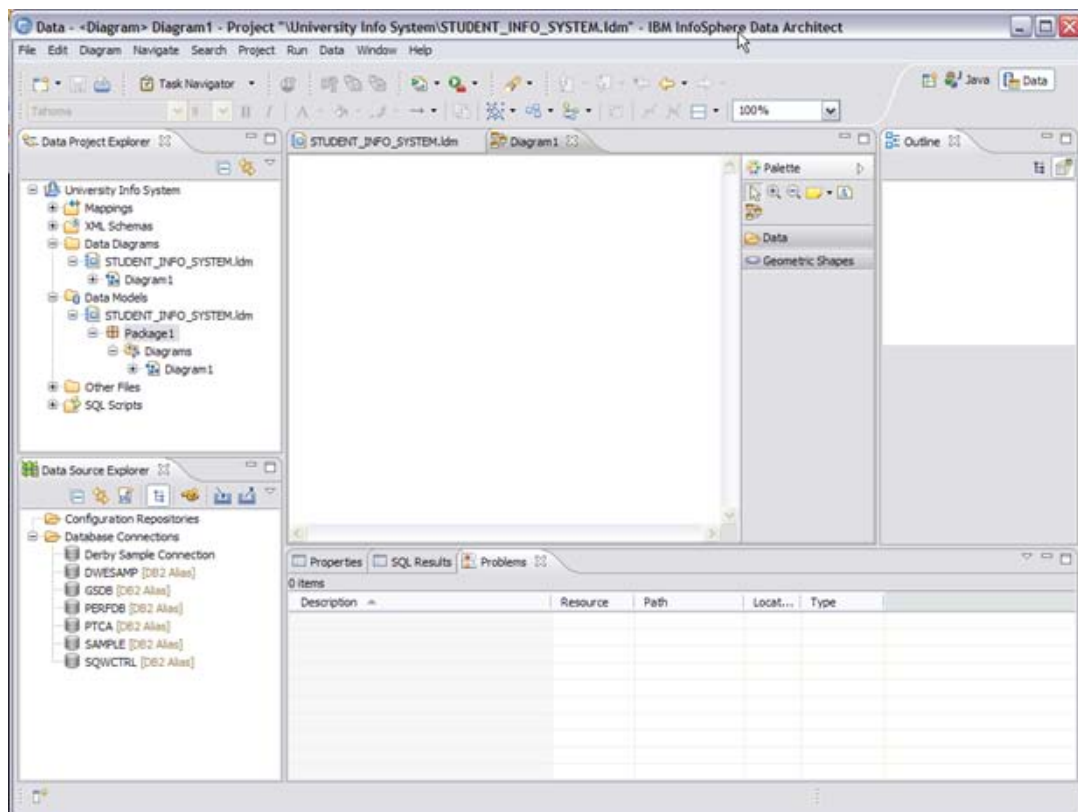
Create a new logical data model:

1. Open the New Logical Data Model window:
  - a. In the Data Project Explorer view, expand the *University Info System* data design project.
  - b. Right-click on the *Data Models* folder, then choose *New -> Logical Data Model*.

The New Logical Data Model window opens.

2. Complete the fields of the New Logical Data Model window:
  - a. Specify the following text in the *File name* field: **STUDENT\_INFO\_SYSTEM**.
  - b. Click *Finish*.

The logical data model is created, and the data model is opened, including a blank diagram. You can use the diagram editor to create a visual representation of your logical data model. *Figure 3.2* shows your new data design project and the new, empty logical data model.



**Figure 3.2 – Creating a blank logical data model**

### 3.2.2 Creating entities with the diagram

To work with a logical data model, you should work with the visual representation of the model, the diagram. Each logical data model can contain multiple diagrams, each representing a small part of the greater logical data model.

#### 3.2.2.1 Creating diagrams

You can use diagrams to visualize and edit objects that are contained in data design projects.

As you create objects within the diagrams of the logical data model, the changes are automatically made to the logical data model, making it easier to not only visualize your data objects and the relationships between those objects, but also saving the time needed to manually create these relationships.

Data diagrams are a representation of an underlying data model. You can create diagrams that contain only a subset of model objects that are of interest.

### 3.2.2.2 The Data section of the palette

The Data area of the palette contains data model objects. When you add or modify data model objects by using the palette or the diagram surface, you modify the underlying data model.

### 3.2.2.3 Renaming data objects from the Properties view

By default, *Diagram1* is the name of this diagram, but you should change it to something that you can easily remember or differentiate from other data models in the project. To rename the diagram and the top-level package, perform the following steps:

1. Open the *Diagram1* diagram in the diagram editor. When you open or select any object, the properties of that object are displayed in the Properties view. In this case, you will edit the diagram properties of the Diagram1 diagram.
2. Open the *General* tab of the Properties view, and change the text in the *Diagram name* field to **STUDENT\_INFO**.
3. Select the Package1 package in the Data Project Explorer. The package is located under the **STUDENT\_INFO\_SYSTEM.ldm** data model file.
4. Open the General tab, and change the text in the *Name* field to **STUDENT INFO SYSTEM**.

The data object is automatically renamed. When you click a blank area in the diagram, the name of the diagram editor changes to **STUDENT\_INFO**, and the logical data model is updated in the Data Project Explorer view. An asterisk (\*) appears next to any data object in the Data Project Explorer or the data object editor that has been modified. The asterisk will disappear when you save the file or project.

**Note:**

You can modify the display characteristics of the diagram with the Properties view. For example, you can change the following options:

- Use the Format tab to change the foreground or background colors of entities or foreign keys, or change the line color of an implicit foreign key relationship.
- Use the Documentation tab to document the purpose of the diagram.
- Use the Appearance tab to format the fonts and text color.

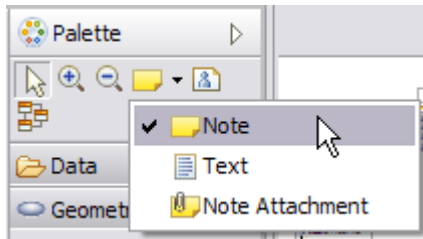
All of these options change the diagram properties, but do not modify the underlying data model. You can also save image files of diagrams for archival or presentation purposes.

### 3.2.2.4 Adding notes and text boxes to the diagram

Add notes or text boxes to the diagram to help you document information about the data model. Notes and text boxes are also useful to call out information or fully explain what complex diagrams are intended to address.

First, let's add a note to the diagram that labels the diagram.

1. Click the *Note* icon in the palette of the diagram, as shown in *Figure 3.3*.

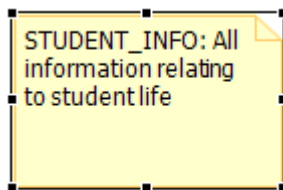


**Figure 3.3 – Clicking the Note icon**

2. Click a blank area in the diagram editor. The note appears on the diagram.
3. In the note, type the following note:

**STUDENT\_INFO: All information relating to student life.**

The note appears on the diagram, as shown in *Figure 3.4*.



**Figure 3.4 – Creating a note in the diagram**

To provide additional information to a data diagram, you can add text boxes. Let's add a text box to note that the logical data model is still being drafted.

1. Right-click an empty space in the diagram, then select *Add -> Text*. A text box appears on the diagram.
2. In the text box, type:  
**NOTE: This logical data model is still being drafted.**
3. Make the text red and bold:
  - a. Open the *Advanced* tab of the Properties view.
  - b. Expand the *Styles* property.
  - c. Change the *bold* property to *true*.
  - d. Change the font color to red.
4. Save your work. Click *File -> Save All*.

### 3.2.2.5 Designing the data diagram and logical data model

An *entity* is a logical data model object that store information about significant areas of interest. Each entity contains *attributes*, which store details about the entities. For example, you can create a Sales entity, with ProductKey, Quantity, UnitPrice, and OrderKey attributes to describe the sales for various products. When the logical data model is transformed into a physical data model, the entities can become tables, and attributes become columns.

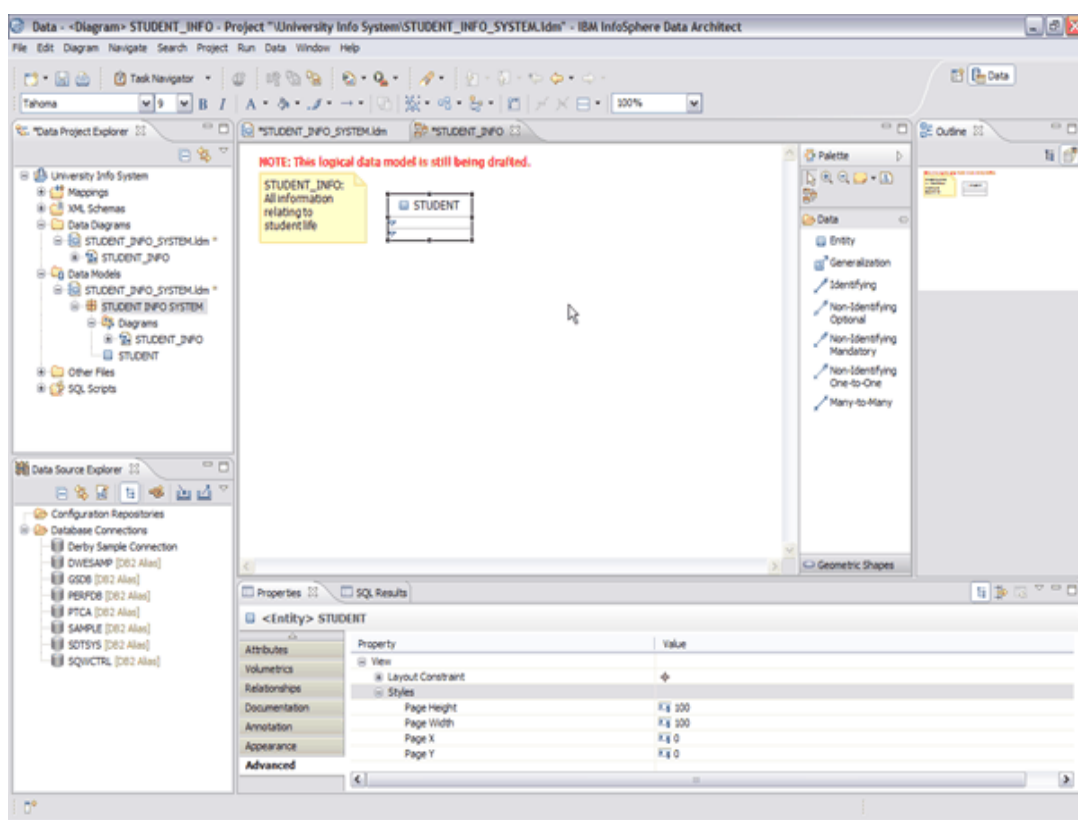
When you create entities and attributes, be as specific as you can. Insert all attributes you need, all domain constraints, and all other aspects you consider in helping you obtaining a model closer to real world. Remember that the logical data model will be transformed into a physical data model.

Let's use the data diagram to add a new entity to the logical data model:

1. In the Palette view, expand the *Data* section, then drag and drop an entity onto a blank space in the diagram editor. The new entity is created in the model and within the diagram, and you can immediately give the entity a new name.
2. Name the entity **STUDENT**.

The entity is created in the diagram, and the corresponding data object is created in the logical data model. When you add or modify logical data model objects by using the palette or the diagram surface, the underlying logical data model is modified.





**Figure 3.5 – Creating the STUDENT entity**

You can also add entities to the diagram via two other methods:

- ✧ Using the context menu: You can right-click in the diagram editor to add data objects to a diagram. To add a new entity, you can right-click an empty area of the editor, then select *Add Data Object > Entity*.

You can also use this method to add other data objects to existing data objects; for example, you can add attributes to entities by using this method.

- ✧ Using the pop-up bar to create entities: You can use the white space on the diagram editor to add data objects to diagrams. When you hover over a data object or an empty area of the diagram, a pop-up bar appears, as shown in *Figure 3.6*:




**Figure 3.6 – Using the pop-up bar to create data objects in the diagram**

You can also use this method to add other data objects to existing data objects; for example, you can add primary keys to entities by using this method.

### 3.2.2.6 Adding attributes to the STUDENT entity

The entity is currently empty. Add attributes to the entity. Attributes of an entity can include the primary key and non-key attributes. All of these attributes will store information about each student at the university.

Let's start with the STUDENT entity. At the university, each student receives a unique ID number that is 10 characters long. You will create an attribute to store the student ID number of the students at the university. To add attributes to the STUDENT entity, perform the following steps:

1. Select the STUDENT entity in the STUDENT\_INFO diagram. The Properties view displays the properties of the STUDENT entity.
2. Open the *Attributes* tab of the Properties view.
3. Add the student ID (STUDENT ID) attribute to the entity:
  - a. Click the *New* button (  ). A new attribute is created in the Properties view and the diagram.
  - b. Give the new attribute the following properties:
    - *Name*: STUDENT ID
    - *Primary key*: Select this check box.
    - *Type*: CHAR
    - *Length/Precision*: 10

You have created the STUDENT ID attribute. The STUDENT ID attribute is of the CHAR(10) data type, making it long enough to store the student ID numbers of the students at the university. When you select the check box in the *Primary key* column to mark the STUDENT ID column as the primary key, the *Required* property is automatically selected, too, because a primary key cannot be a null value.

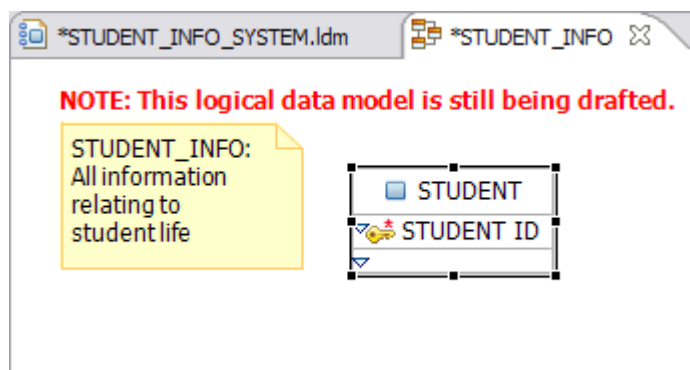


Figure 3.7 – Creating the STUDENT ID attribute

**Note:**

If you don't know what data type to choose, keep in mind the following considerations:

- If an attribute or column is designed for arithmetic calculations, use a numeric data type.
- Use text data types for all other attributes for the following reasons:
  - Text data types are more flexible; for example, you can use constraints to reduce storage size of data.
  - You can better control the input of text data types because DB2 offers more functions for string data types.

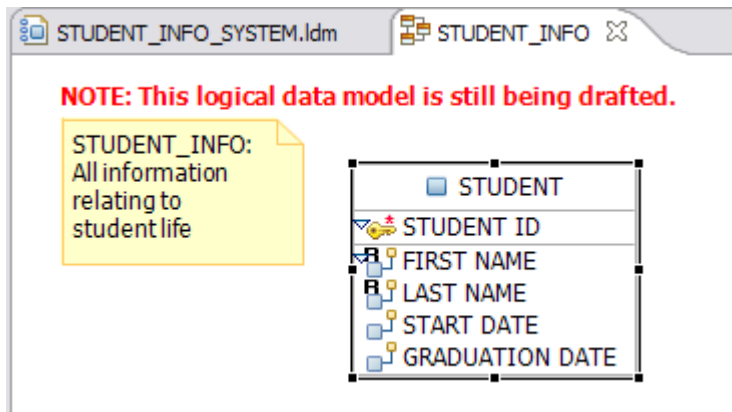
However, sometimes you should use a numeric data type if you want the primary key to also be numeric (for example, surrogate keys are also numeric).

Note that when you add attributes to the entity, the diagram is also updated. The STUDENT ID entity is now displayed in the key compartment of the STUDENT entity in the diagram, with an icon next to it to indicate that STUDENT ID is the primary key of the entity.

Follow the same process outlined above to add more other attributes to the STUDENT entity. Add the following attributes to the STUDENT entity:

4. Add the FIRST NAME attribute. The attribute has a data type of VARCHAR(20) and is required.
5. Add the LAST NAME attribute. The attribute has a data type of VARCHAR(50) and is required.
6. Add the START DATE attribute. The attribute has a data type of DATE.
7. Add the GRADUATION DATE attribute. The attribute has a data type of DATE.
8. Save your work.

You have used the diagram editor to create a complete entity, as seen in *Figure 3.8*. Now let's learn how to use the Data Project Explorer to create another entity.



**Figure 3.8 – Complete STUDENT entity**

### 3.2.2.7 Creating entities in the Data Project Explorer

Let's use the Data Project Explorer view to create a new entity, GRADE to store information about student grades.

To create entities with the Data Project Explorer view:

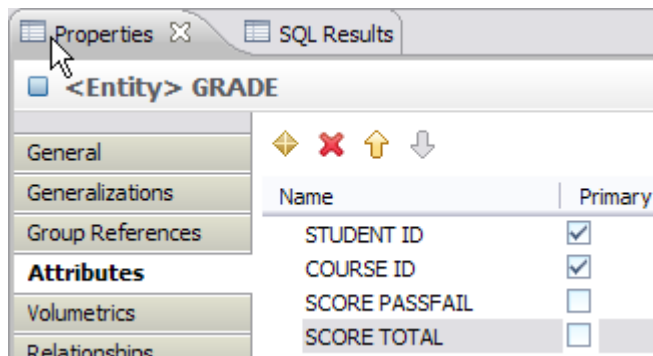
1. Locate the *STUDENT\_INFO\_SYSTEM.Idm* data model in the Data Project Explorer by expanding the folders in the project: *University Info System -> Data Models -> STUDENT\_INFO\_SYSTEM.Idm*.
2. Create the GRADE entity:
  - a. Right-click the *STUDENT INFO SYSTEM* package under the logical data model, then select *Add Data Object -> Entity*.
  - b. Name the new entity GRADE by making sure the entity is selected in the Data Project Explorer, then open the *General* tab of the Properties view and change the name and label to **GRADE**.

The entity is created under the *STUDENT INFO SYSTEM* package. Note that the entity does not appear in the diagram editor. This is because you did not create the entity within the diagram. Only objects that you create or drag to the diagram appear in the diagram editor. To better visualize how the data is connected, add the entity to the diagram.

3. Add the new entity to the diagram:
  - a. Select the GRADE entity in the Data Project Explorer.
  - b. Drag the entity from the Data Project Explorer to a blank space in the diagram editor. When you release the mouse button, the entity is added to the diagram.
4. Select the GRADE entity.
5. Add the following attributes to the GRADE entity:

- STUDENT ID: Use the same properties that you used to create the STUDENT ID attribute for the STUDENT entity.
- COURSE ID: Use the CHAR(8) data type. This is also a primary key.
- SCORE PASSFAIL: Use the BOOLEAN data type. This attribute is required.
- SCORE TOTAL: Use the DECIMAL data type, with a length/precision of 3 and a scale of 0. This attribute is required.

You have created the GRADE entity. In some cases a composite key is needed. A *composite key* is a primary key that is made up of more than one attribute. Composite keys help make sure that your primary keys are always unique without duplicating existing attributes. For example, to make the GRADE entity unique, you need a composite primary key: STUDENT ID and COURSE ID. This is shown in *Figure 3.9*.



**Figure 3.9 – Creating a composite key in the Properties view**

**Note:**

When you delete objects from the diagram, they are not deleted from the data model. Use the Data Project Explorer to remove data objects from your data models.

### 3.2.3 Adding relationships

Now that you have created the two entities in this diagram, you need to specify how the entities are related. Since the information in this model is related to the student, the STUDENT entity is the parent entity.

Just as you can create entities within the diagram, you can also specify the relationships between data objects by using the diagram editor.

Relationships types are identifying and non-identifying. In an *identifying* relationship, one of the child entities is a dependent entity. A dependent entity is an entity whose existence is based on the existence of another set of entity. A *non-identifying* relationship is one in which both entities are independent (also known as strong relationship set).

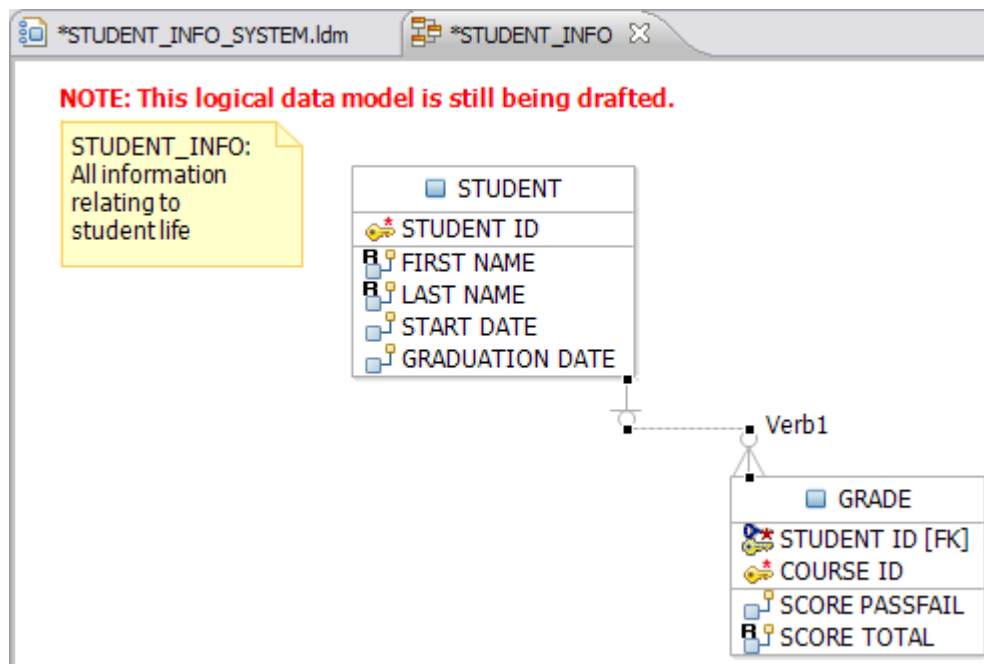
Create a non-identifying optional relationship between the STUDENT entity and the GRADE entity:

1. Display all key and non-key attributes in each entity:
  - a. Click on a blank part of the diagram. The properties of the diagram display in the Properties view.
  - b. On the *Filters* tab of the Properties view, select the *Show key* and *Show non-key* check boxes.
2. Use the diagram to create a *Non-Identifying Optional* relationship from the STUDENT entity to the GRADE entity:
  - a. In the palette, under the *Data* section, select the *Non-Identifying Optional* data object.
  - b. In the diagram editor, select the STUDENT entity, drag the mouse cursor to the GRADE entity, then release the mouse button.
  - c. Since the STUDENT ID attribute exists in both entities, you are asked to specify how to address the attribute. In this case, choose the *Replace with migrated FK attribute/column* option to ensure that both attributes are the same.

The relationship is created in the model and depicted in the diagram. Now you should describe the relationship. Remember that the diagram is just a visual representation of the metadata in the model; therefore the relationship is also created in the logical data model.

When you “describe” this relationship, you explain with verb phrases how these entities interact in the real world. In this case, you might explain that each student has more than one grade, because they take more than one class. Also, a single grade is given to each student for each class. Therefore, the cardinality of the relationship is a *one-to-many* relationship.

3. Describe the relationship, using the Properties view:
  - a. Select the relationship object on the data diagram, as shown in *Figure 3.10*:



**Figure 3.10 – Selecting the relationship object**

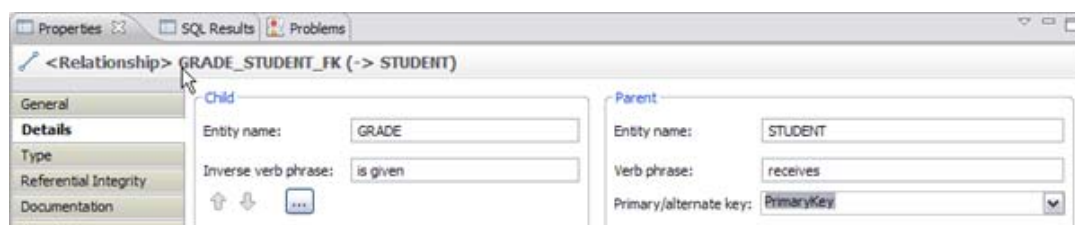
The properties of the relationship object display in the Properties view.

- b. Open the *Details* tab of the Properties view.
- c. Enter verb phrases to describe the relationship between the parent and child entities:
  - GRADE: **is given**
  - STUDENT: **receives**

The relationship is defined in the Properties view, as shown in *Figure 3.11*.

#### Note

All foreign key attributes are automatically constructed. You are not prompted to migrate the key attribute if you do not have a corresponding attribute of the same name in the child entity.



**Figure 3.11 – Defining the relationship within the Properties view**

In the Properties view, it is possible to choose the type of relationship, participation, and cardinality.

4. Explore the other tabs in the Properties view to see what other information you can define:
  - a. Open the *Type* tab. Make sure that the following options are defined:
    - *Relationship Type*: Non-identifying
    - *Existence*: Optional
    - *Cardinality*: One or more
  - b. Use the *Referential Integrity* tab to specify what should happen to the child entity when you update the primary key of the parent entity of the relationship. Make sure that the *Delete* option is set to *CASCADE* in the *Parent Action* group.

The *Referential Integrity* tab should look like the image in *Figure 3.12*:



**Figure 3.12 – Specifying a CASCADE action on the child GRADE entity**

*Table 3.1* specifies what will happen when you specify each action. You draft these requirements in the logical data model, and when you deploy the physical data model, these actions will occur when the referenced data objects are updated.

Action	Definition
NONE	No action occurs. The delete or update action is rolled back.
RESTRICT	The parent entity is not deleted, nor its key updated, because there are dependent entities that rely on this entity.
CASCADE	If the modified parent entity has a foreign-key relationship to the other entity in the relationship, the foreign-key reference to the dependent entity is updated. If the entity is deleted, and the delete action is set to CASCADE, the dependent entity is removed from all relationships (in other words, deleted).
SET_NULL	If the modified parent entity has a foreign-key relationship to the other entity in the relationship, each nullable foreign-key attribute of each dependent entity is set to NULL. This option can only be specified if all foreign-key attributes allow NULL.



	values.
SET_DEFAULT	When a parent entity is altered, each nullable foreign key attribute is set to its default value. This option can only be specified if all foreign-key attributes have default values.
DO_NOT_ENFORCE	<p>This generates an informational constraint for DB2 systems. The code that is generated is not enforced.</p> <p>An informational referential constraint is a constraint that DB2 does not enforce during normal operations. Use these constraints only when referential integrity can be enforced through other means, such as when you retrieve data from other sources. These constraints might improve performance by allowing the system to automatically rewrite the query.</p>

**Table 3.1 – Definitions for parent and child entity actions**

5. Save your work.

### 3.3 Working with glossary models

Sometimes, when working with various people on the same project, data object naming varies. Where one data architect might refer to sales entities as **SALES\_x**, another data modeler might prefer **SLS\_x**. Over time, the system becomes difficult to understand, and the inconsistency can spread across systems, causing confusion, longer application development times, or even the inability to make the connection between two fields that mean the same thing in two different systems.

To avoid this problem, you should develop a set of naming standards early in the data design process. *Naming standards* help you and other people in your organization look at your data in a consistent manner, so that you all get the same meaning. With IBM InfoSphere Data Architect, you can create a set of rules that all data modelers should follow, and the workbench can automatically create names of some of the data objects (such as relationships) for you.

A *glossary model* is a model that describes the names and abbreviations that an organization allows for data objects. Data object naming standards promote a common understanding of data, since you use the same name conventions across the entire organization. You can use glossary models to enable sharing of data across organizational boundaries and reduce data redundancy through the consolidation of synonymous and overlapping data elements.

When you name a data object, you need to take into account two items: semantic rules and format. Semantically, glossary model objects include prime words, class words, and modifiers:

- *Prime word*: A word that represents the business concept about which data is being collected. Prime words are nouns that describe the subject area of the data. For example: LOAN, EMPLOYEE
- *Class word*: A word that identifies a distinct category or classification of data. For example: RATE, NAME
- *Modifier*: A word that further qualifies or distinguishes the prime and class words. It helps make the object name clear and unique, and it can help restrict the meaning of the class and prime words. For example: NEXT, FIRST

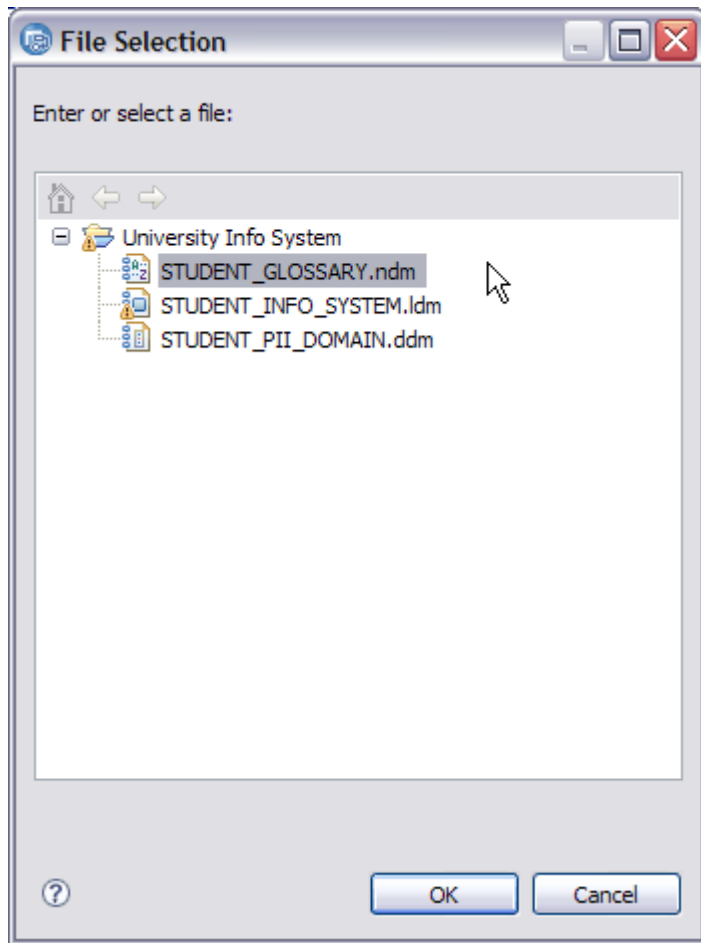
You can use IBM InfoSphere Data Architect to help you define these semantic rules or *naming standards*. You will learn more about how to specify some of these options for future data modeling projects in *Section 3.4, Working with naming standards*.

To enforce the naming conventions that are specified in a glossary model, the glossary model must be referenced by the project that contains the models to be analyzed. To reference an existing glossary model in a project, perform the following steps:

**Note:**

You do not need to perform these steps, since you are creating a glossary model that the project will reference in an upcoming section. This step is only necessary if you did not add the glossary model to the project when you created it, or if you imported a glossary model from another product.

1. Select the data design project in the Data Project Explorer. The properties for the data design project display in the Properties view.
2. Open the *Naming Standard* tab, then click the *Add* button. The File Selection window opens.
3. Expand the data design project, and then select the glossary model from the list of files, as shown in *Figure 3.13*.



**Figure 3.13 – Selecting a glossary model**

The glossary model is referenced by the data design project.

### 3.3.1 Best practices for naming standards and glossary models

Here are some best practices you can keep in mind when designing your naming standards:

- Choose the singular form of the word for entity names.
- Do not use numbers for the first character of an entity name.
- Choose short, unambiguous names, and try to use no more than one or two words.

When creating glossary models, consider the following best practices:

- Create a separate enterprise standard project that contains any glossary or domain model that should be shared throughout the organization. You will learn more about domain models in Chapter 4.

- Create a glossary model within a specific project for legacy models that might have terminology that was created before the enterprise glossary and may use different definitions.
- Give the glossary model a meaningful name.
- Create multiple glossaries to logically group terms into categories that make sense from a business perspective.
- Create nested glossaries to provide a logical grouping of terms within a specific business subject area.
- Create a sub-glossary that contains class words only to reduce the time you spend looking through an entire glossary to understand these terms.
- If possible, use a spreadsheet to enter business terms more efficiently. Copy terminology to a spreadsheet application such as Lotus® Symphony or Microsoft Excel to quickly update and edit terminology and then copy it back into InfoSphere Data Architect.

### 3.3.2 Creating a glossary model

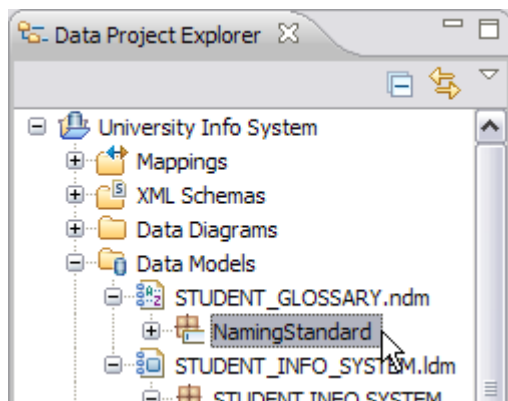
Let's create a glossary model to create a student glossary. This glossary model will determine how information about students is specified throughout the data models in the project.

In order to create a glossary model, perform the following steps:


1. Select *File -> New -> Glossary Model*.
2. Make sure that the destination folder is set to the University Info System data design project.
3. Specify **STUDENT\_GLOSSARY** as the file name.
4. Make sure that the glossary is added to the naming standard definition for this project by selecting the *Add to project properties as project naming standard* check box. Click *Finish* to create the glossary model.

The **STUDENT\_GLOSSARY.ndm** glossary model is opened, and you can find it in the *Data Models* folder of the data design project.

5. Specify a name for the root package of the glossary model:
  - a. Select the *NamingStandard* data object under the **STUDENT\_GLOSSARY** glossary model, as shown in *Figure 3.14*.



**Figure 3.14 – Selecting the NamingStandard package**

- b. In the *General* tab of the Properties view, change the package name to **STUDENT\_GLOSSARY\_ROOT**.
6. Specify a name for the empty glossary. Select the *Glossary1* glossary, and change the glossary name to **STUDENT\_GLOSSARY**.
7. Create the ID Number glossary item:
  - a. In the *Contained Words* section, click the *New* button (  ). A new word, Word1 is added to the glossary.
  - b. Select the new word, and change the name to ID Number.
  - c. In the *Abbreviation* field, specify ID.
  - d. In the *Type* field, specify *PRIME*.
8. Complete the glossary model by creating the following words:
  - a. Create the following prime words:
    - Address (abbreviated ADDR)
    - Date (DATE)
    - Department (DEPT)
    - Name (NAME or NM)
    - Pass/fail (PASSFAIL or PF)
    - Social security number (SSN)
    - Total number (TOTAL or TTL)
  - b. Create the following class words:
    - Course (COURSE or CRS)
    - Email (EMAIL or EML)

- First (FIRST or FT) – also a modifier
- Graduation date (GRADUATION or GRDN)
- Instructor (INSTRUCTOR or INSTR)
- Join date (JOIN)
- Last (LAST or LT) – also a modifier
- Score (SCORE or SCR)
- Start date (START)
- Student (STUDENT or SDT)
- Student activity (ACTIVITY or ACT)

9. Save your work.

In the next section, *3.4 Working with naming standards*, you will learn how to modify the naming standards so that when you validate the logical data model, your entities and attributes are mostly in compliance with the items in the glossary model.

**Note:**

You are not required to fix warnings in the Problems view to create a valid, usable data model. In this case, the warnings let you know that the current names of the data objects are not in line with your glossary model. You can take steps to either update the naming standard options or edit the names of the data objects to be compliant with the glossary model.

### 3.4 Working with naming standards

Now that we've created a glossary model, let's set up the naming standard that the data design projects in your workspace should adhere to. After working with university officials, you decide that it makes the most sense to display descriptive class words before prime words when you name attributes and columns. You decide to go into the workbench preferences to set the naming standard.

To specify the naming standards for physical and logical data models and glossary models, open the Preferences window:

1. Click *Window -> Preferences*. The Preferences window opens.
2. Expand the *Data Management* node and select *Naming Standard*.
3. Make sure the *Logical* tab is open.
4. In the Entity section, specify the following order:
  - a. Modifier (optional)
  - b. Class word (optional)

- c. Prime word (optional)
  - d. Modifier (optional)
  - e. Modifier (optional)
  - f. Modifier (optional)
5. In the Attribute section, specify the following order:
  - a. Modifier (optional)
  - b. Class word (required)
  - c. Prime word (required)
  - d. Modifier (optional)
  - e. Modifier (optional)
  - f. Modifier (optional)
6. In the *Physical-Table/Column* tab, specify the same options as in steps 4 and 5.
7. Click *OK* to save the preferences.

### 3.4.1 Analyzing to check compliance with naming standards

After you set the naming standards, you should check to see what data objects are in compliance with the naming standards and glossary model items that you have specified. Use the Analyze Model wizard to specify what rules to check for, and then use the Problems view to seek out and correct any data objects that do not comply with your naming standards.

To analyze the logical data model:

1. Locate the STUDENT INFO SYSTEM package in the logical data model:  
*University Info System -> Data Models -> STUDENT\_INFO\_SYSTEM.Idm -> STUDENT INFO SYSTEM.*
2. Right-click on the package and select *Analyze Model*. The Analyze Model wizard opens.

Since we are analyzing a logical data model, every rule under the *Logical Data Model* rule category is selected.

3. Deselect the check box next to the *Logical Data Model* rule category, then select the *Naming Standard* rule. Selecting this rule allows you to narrow the model analysis to make sure that data objects are in compliance with the naming standard rules only. This narrows the amount of errors or warnings you see in the Problems view.
  4. Click *Finish* to analyze the model.

**Note:**

If you see warnings in the Problems view, you are not required to fix warnings to create a valid, usable logical data model. Warnings simply let you know when certain analyzed items might be out of compliance or do not conform to standard data modeling rules, but it will not impact the functionality of the model.

### 3.5 Constraints

Now that you have created a couple of entities, you should create some constraints. *Constraints* are used to enforce business rules for data.

First, let's create a constraint that checks to make sure that the student's graduation date is later than the student's start date. This check constraint will make sure that your data is valid when it's entered into the database.

To add a constraint to the STUDENT entity:

1. Locate and select the STUDENT entity in the Data Project Explorer: *University Info System -> Data Models -> STUDENT\_INFO\_SYSTEM.Idm -> STUDENT INFO SYSTEM -> STUDENT*.
2. Right-click on the STUDENT entity, then select *Add Data Object -> Constraint*.
3. Name the constraint DATE\_CK.
4. Complete the information on the *General* tab of the Properties view to create a constraint:
  - a. Specify *SQL* as the language.
  - b. In the *Expression* field, type **GRADUATION\_DATE>START\_DATE**.
  - c. In the *Transform As* field, specify *Check Constraint*. The check constraint simply checks that the expression is true.
5. Save your work.

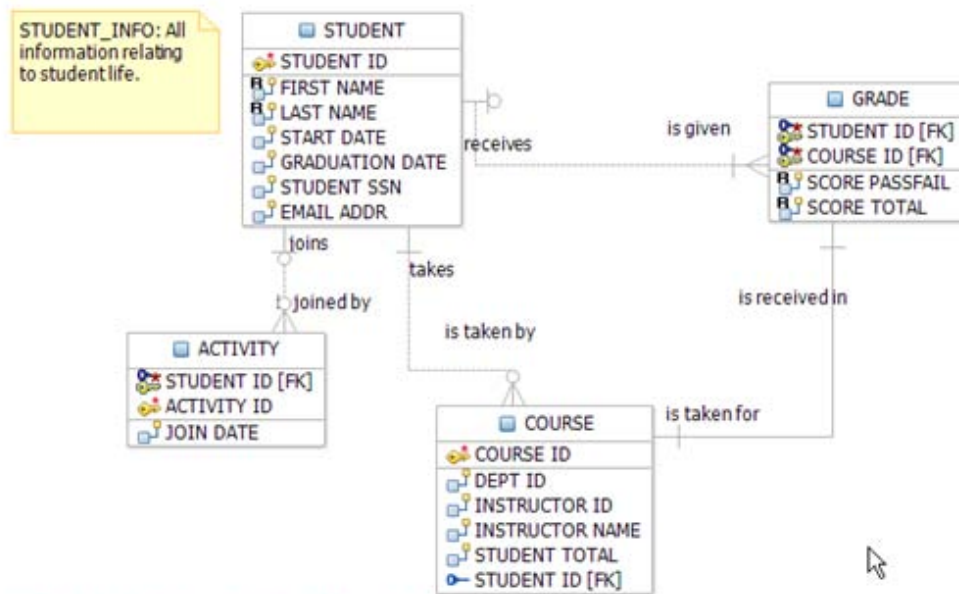
### 3.6 Exercise

Use the process outlined above to expand upon the existing logical data model. Determine the best data type for each attribute, and document each relationship with descriptive verbs. Consider how each entity is related to other entities in the model.

1. Add another entity, ACTIVITY, to describe the clubs and student organizations that each student belongs to. You will use this entity as you complete the other chapters in this book. The entity should include the following required attributes:
  - STUDENT ID: Stores the student ID number. This is part of the composite key.



- ACTIVITY ID: Stores the 5-character ID number of the club or organization. This is part of the composite key.
  - JOIN DATE: Stores the date that the student joined the organization.
2. Link the STUDENT and ACTIVITY entities with a *Non-identifying optional* relationship, with a *Zero or more* cardinality. Use the *Details* tab to define the relationship of the parent and child entities, and add descriptive verbs to the relationship.
  3. Now create one more entity for the logical data model, COURSE. Create the following required attributes:
    - COURSE ID: Stores the 5-character ID number of the course. This is the primary key of the entity.
    - DEPT ID: Stores the 4-character ID of the department that runs the course. For example, a course within the English department would have the ENGL prefix.
    - INSTRUCTOR ID: Stores the 5-character ID of the instructor that is teaching the course.
    - INSTRUCTOR NAME: Stores the last name of the instructor that is teaching the course.
    - STUDENT TOTAL: Stores the number of students registered for the course.
  4. Link the STUDENT and COURSE entities with a *Non-identifying mandatory* relationship, with *One or more* cardinality. Again, use the details tab to describe the relationship between the tables with verbs. Then, link the COURSE and GRADE entities with an *Identifying* relationship, with a cardinality of *exactly one*. Document descriptive verbs for each end of the relationship.
  5. Remember to save your work.
  6. Make sure the diagram reflects the logical data model. When complete, your logical data model should look similar to the image in *Figure 3.15*.



**NOTE:** This logical data model is still being drafted.

**Figure 3.15 – The completed STUDENT\_INFO diagram and logical data model**

Now that you have created a small logical data model, define a domain model for this and other data models for the project. A domain model lets you define custom data types for your organization. In the next chapter, you will learn about domain models and how to apply them to your logical data model.

### 3.7 Summary

This chapter discussed the concept of logical data modeling, using IBM InfoSphere Data Architect to complete your task. You learned how to create data models with both the data diagram and the Data Project Explorer.

The chapter also introduced topics like working with naming standards and working with glossary models. Naming standards help everyone in your organization look at the same words and data objects in the same way, while deriving the same meaning from them. The words will have a consistent representation throughout IT systems. Naming standards promote a common understanding of data.

Glossary models are used to enable sharing of data across organizational boundaries and reduce data redundancy. Create glossary models to better enforce your naming standards.

### 3.8 Review questions

1. To ensure that an attribute can contain only the values 'B' or 'G', which data object should you choose?

- A. Constraint
  - B. Primary key
  - C. Glossary model
  - D. Relationship
2. Which of the following is not a definition of an entity?
- A. An entity is a collection of real-world objects or concepts that have many common properties.
  - B. An entity is an instance of real-world objects.
  - C. An entity is a set of entities of the same type that share the same properties.
  - D. An entity is an item that exists and is distinguishable from other items.
3. Which of the following data types is a better option for a person's phone number attribute?
- A. NUMERIC
  - B. INTEGER
  - C. CHAR
  - D. LONG
4. Draw entities and relationship sets for each of the following.
- A. Each garage may house one or more cars. Each car may stay in one and only one garage.
  - B. Each student may learn from one or more teachers. Each teacher may educate one or more students.
  - C. An employee may manage one or more employees. An employee must be managed by one employee.
  - D. An employee may belong to one company's branch. A company's branch must have one or more employees.
5. Resolve the many-to-many relationship set between students and teachers. Think of additional attributes for the new entity.
6. Given the following entities within a logical data model, a graphical representation of the relationship and entity sets has to be provided:
- a company's branch;
  - employees and their relatives of the company's branch;

Draft a data model diagram.

7. In a University logical data model, choose the best relationship name from STUDENT to GRADE:
  - A. Each STUDENT must be the receiver of one or more GRADES.
  - B. Each STUDENT must be the sender of one or more GRADES.
  - C. Each STUDENT may be the receiver of one or more GRADES.
  - D. Each STUDENT may be the producer of one or more GRADES.

# 4

## Chapter 4 – Domain Models

In this chapter, you will learn how to create a domain model. *Domain models* describe domain data types and their constraints that are specific to an organization. You can define domain models as part of the logical data model, or you can create a separate domain model that can be used throughout all logical and physical data models for a project.

You will create a domain model that can be used to model what information should be secured in a test environment – in particular, you will secure the social security numbers, last names, and email addresses of each student. The test system will be populated with realistic data, and the privacy rules that you specify ensure that sensitive information is still protected. Then, you will associate this domain model with the data objects in the logical data model.

**Note:**

The domain models that you create in InfoSphere Data Architect can be shared with the Optim Data Privacy Solution suite of products. To learn more about the Optim Data Privacy Solutions, visit this URL:

<http://www-01.ibm.com/software/data/optim/protect-data-privacy/>

### 4.1 Domain models

Domain models describe the domain data types and their constraints that an organization allows. You can create organization-specific data types within a domain model (in other words, data types that describe aspects of a certain domain). Using a domain data type instead of the base data type ensures that you maintain consistency across an organization, and you can reuse the common data type definitions to make your team more efficient.

Domain data types can be stored in a domain model or as part of a logical data model. You can associate a domain data model with a data design project so that the domain data types are available for all of your logical and physical data models. Using atomic domains instead of base data types ensures consistency among team members.

A domain model consists of a group of domain data types. A domain data type represents a base data type that can be restricted by adding constraints. You can use the domain model to define the domain data types for attributes or columns within logical or physical data models.

Domain data types are based on base data types. For example, you can define domain data types for common descriptors, such as gender or grade. IBM InfoSphere Data Architect can be used to create domain models and atomic domains. These models can be shared among all team members who are working on modeling tasks. IBM InfoSphere Data Architect provides the tooling that helps you create and manage domain models. You can create a domain model from a wizard, either from an existing template or from scratch. Use the Data Project Explorer to add domain objects, then use the Properties view to modify them .

The following list details some best practices for domains:

- Group all domains into one or more domain models, so that these domains can be reused in future projects.
- Create a separate enterprise standard project that contains any glossary or domain model that is universally shared throughout the organization.
- Create a domain model within a specific project for legacy models that might have domains that were created before any enterprise domains if they do not conform to the newer enterprise standard.
- Give the domain model a meaningful name. You might want to include “domain” as part of the model name to easily identify the domain model in the Data Project Explorer view.
- Create subpackages (separate domain packages under the root domain package) to provide a logical grouping of domains within a specific business subject area.

Now that you have created the base logical data model, you have been asked to also store sensitive information within the logical data model, including social security numbers and email addresses. Before you expand the logical data model to include this information, you decide to create a domain model that will secure this sensitive information. Then, you will create new attributes in the STUDENT entity and use the domain model data types to define the attributes.

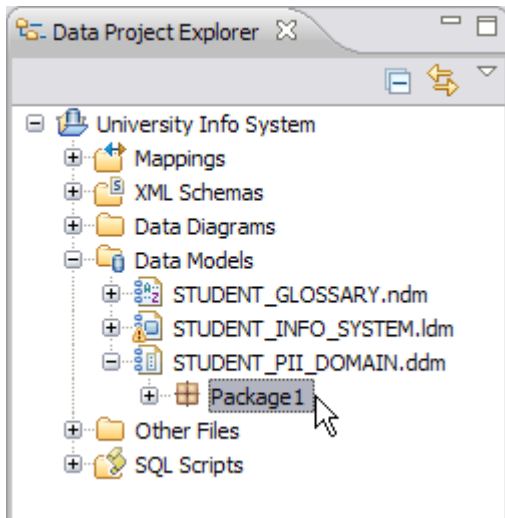
First, let's define a domain model to protect personally identifiable information. This domain model will help you ensure that the personal information of students is protected.

#### **4.1.1 Creating a blank domain model**

Let's create the *STUDENT\_PII\_DOMAIN* domain model:

1. Right-click the *Data Models* folder in the Data Project Explorer, then select *New -> Domain Model*. The New Domain Model window opens.

2. Specify the *File name* for the domain model: **STUDENT\_PII\_DOMAIN**.
3. Make sure that the *Add to project properties* option is selected, then click *Finish*. The **STUDENT\_PII\_DOMAIN.ddm** domain model is created in the *Data Models* folder of the data design project.
4. Specify the name for the root package of the domain model:
  - a. Select the package underneath the **STUDENT\_PII\_DOMAIN** domain model, as shown in *Figure 4.1*.



**Figure 4.1 – Selecting the STUDENT\_PII\_DOMAIN domain model**

The properties for the package display in the Properties view.

- b. Specify the following name in the *Name* field: **STUDENT\_PII\_ROOT**
4. Save your work.

The domain model and package are created and displayed in the Data Project Explorer. Now that you have created an empty domain model, you can add an atomic domain, list domain, or union domain. In this exercise, you will add an atomic domain to secure the social security numbers of each student.

#### 4.1.2 Atomic domains

*Atomic domains* are custom data types that you create to maintain consistency across your organization. You create domain-specific rules, then use these rules to define other objects within your data models.

For example, you could define a domain model for grades to ensure that no values outside an appropriate range are entered. You can only accept values between 1 and 10 on a 10-point scale.

You can also create a domain model to specify when data is private and requires masking, such as a student or a teacher identifier.

By default, data masking is not enforced, but to ensure privacy of sensitive data elements, you should enforce data privacy.

With an atomic domain, you typically specify privacy rules for four areas:

- ⤴ Classification: In a test environment, how should this data be classified? Is it personally identifiable information, or is it confidential?
- ⤴ Enforcement: In a test environment, is privacy for this atomic domain required, not required, or simply a best practice?
- ⤴ Privacy policy type: What type of information do you want to mask in a test environment?
- ⤴ Privacy policy: What type of masking should a test system use?

#### 4.1.2.1 Creating an atomic domain

You decide that you should first secure student social security numbers. To do this, you will create a new atomic domain, SSN, that will display a random social security number to mask that sensitive piece of information.

This atomic domain creates a unique data type within the data design project, and you can use this special data type for various models throughout your project. This domain model can be used throughout the design process to define the policies that you should apply to the student information stored in this model. This will help you ensure consistency across your data models.

Let's create the SSN and STUDENT\_ID atomic domains:

1. Right-click on the STUDENT\_PII\_ROOT package, then select *Add Data Object -> Atomic Domain*. A new atomic domain object is created in the Data Project Explorer under the root domain package.
2. Name the atomic domain object **SSN**.
3. Define the base data type of the SSN atomic domain object in the *General* tab of the Properties view. Since social security numbers contain both numbers and dashes, you should specify a base data type of CHAR(11).
4. Specify the data privacy properties for this atomic domain data type:
  - a. Open the *Data Privacy* tab of the Properties view.
  - b. Specify the following settings to mask the students' social security numbers:
    - *Classification*: Personally Identifiable Information
    - *Enforcement*: Required
    - *Privacy Policy Type*: Social Security Number (SSN)



- *Privacy Policy*: Random SSN with valid source area number
5. Create the STUDENT\_ID atomic domain with the following options:
    - *Base data type*: CHAR(10)
    - *Classification*: Personally Identifiable Information
    - *Enforcement*: Required
    - *Privacy Policy Type*: Shuffle
    - *Privacy Policy*: Random Shuffle

Specifying these options allows you to specify to your teams that any attributes or columns that make use of this domain model should be masked in test environments. This helps reduce the chance that any data that can identify students at the university is not inadvertently given out.

#### 4.1.3 List domains and union domains

A list domain is one whose values are whitespace-separated, such as **AvailableSizes** (10 12 14). These are a type of atomic domain.

A union domain has values that are a union of atomic values or list values or union values. For example, you can create a union domain, **DressSize**, that allows a value to be either an integer from 2 to 18 or one of the string values **small**, **medium**, **large**.

#### Note:

You do not create a list domain or atomic domain in this book. The definitions are provided here for informational purposes only.

## 4.2 Associating domain model elements with logical data model elements

Once you have created elements within a domain model, you should associate them with the elements in the logical data model. By associating the domain model with a logical data model (and later, the physical data model), you are taking the first steps toward identifying the business requirements that are necessary to privatize the information, so that your policies can be shared with other teams and systems. Then, you can export this domain model to Optim Designer (part of the Optim Data Privacy solution), where you can replace the data in entities and columns that are associated with this domain model with fake (or secured) data. A common application of this is to create a test database and populate it with realistic data and test how data objects associated with the domain model are secured.

#### Note:

Masking data should only be done on test systems.

For the purposes of this book, you should associate the SSN atomic domain with a corresponding attribute in the STUDENT entity. To secure the social security numbers of the students, you will specify the special data type that you created within the SSN atomic domain. You should also create an atomic domain to secure the student ID numbers of each student.

To specify specialized, atomic domain data types for these attributes:

1. First, create a new attribute for the STUDENT entity, and name it **STUDENT SSN**.
2. Select the attribute in the Data Project Explorer. The properties for the STUDENT SSN attribute display in the Properties view.
3. Change the data type for the STUDENT SSN attribute:
  - a. Open the *Type* tab of the Properties view.
  - b. Click the ellipsis (...) next to the *Data Type* field. The Select a Data Type window opens.
  - c. Select the SSN data type from the STUDENT\_PII\_ROOT package, as shown in *Figure 4.2*.

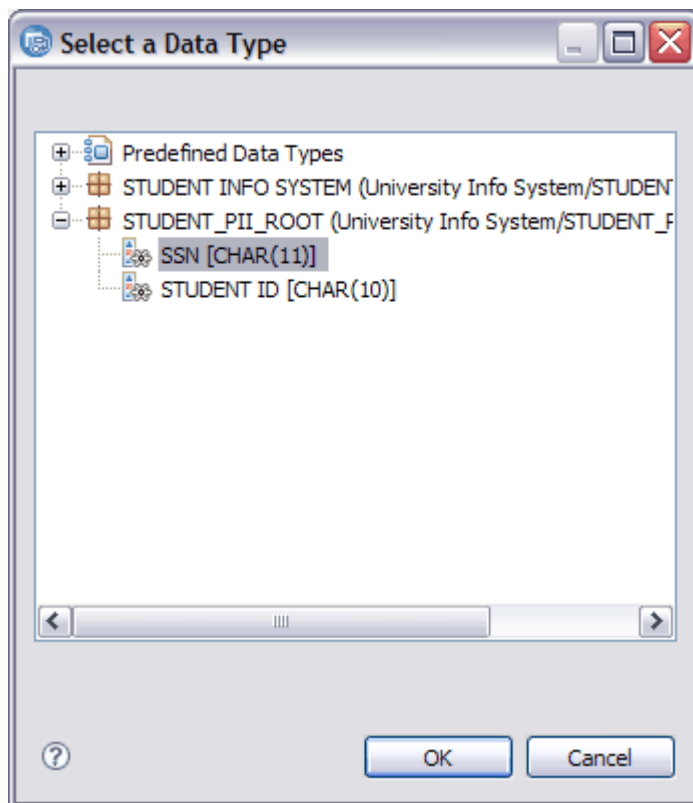


Figure 4.2 – Selecting the atomic domain data type

The domain data type is specified in the *Data Type* field of the SSN attribute.

4. Change the data type for the STUDENT ID attribute:
  - a. Open the *Data Type* tab of the Properties view.
  - b. Click the ellipsis (...) next to the *Data Type* field. The Select a Data Type window opens.
  - c. Select the STUDENT\_ID data type from the STUDENT\_PII\_ROOT package.
5. Save your work.

### 4.3 Exercise

Complete the STUDENT\_PII\_DOMAIN domain model.

1. Create the following atomic domain elements:
  - EMAIL: The base type is VARCHAR(128). This is personally identifiable information, and the privacy for this element is required. To mask the data, a random email address should be used.
  - LAST\_NAME: The base type is VARCHAR(128). This is personally identifiable information, and the privacy for this element is only a best practice. When masking the data, use a random shuffle as the privacy policy.
2. Create the EMAIL ADDR attribute in the STUDENT entity, and associate the EMAIL atomic domain element with that attribute.
3. Associate the LAST NAME attribute with the LAST NAME atomic domain.
4. Save your work.

### 4.4 Summary

This chapter summarized how to create a domain model within IBM InfoSphere Data Architect. You learned how to associate this domain model with an existing logical data model and to identify information that should be privatized when you or other teams are testing your models and systems.

### 4.5 Review questions

1. What is the definition of a domain model?
2. What information is stored in a domain model?
3. True or false: Domain models help you identify the data privacy policies for your enterprise.
4. True or False: You can define your own data types with a domain model.

5. What is a nested domain?

# 5

## Chapter 5 – Physical Data Modeling

In this chapter, you will learn about *physical data models* and how they differ from logical data models. You will also learn how to transform a logical data model into a physical data model. By the end of the chapter, you will learn how to create and use a physical data model to create database-specific DDL statements that can turn your data modeling exercises into a functional schema.

## 5.1 Physical data modeling: The big picture

*Physical data models* are data models that are specific to a database, and they are based on database specifications. With a physical data model, you can model storage that includes partitions, table spaces, or indexes, in addition to other storage objects.

You can model data by making use of specific properties available within a specific database and database platform. For example, physical data models will vary depending on whether they are modeled for DB2 systems, Oracle systems, or SQL Server systems. IBM InfoSphere Data Architect supports many different database system types and versions, and you can use a physical data model to create the database-specific data objects for these systems.

**Note:**

InfoSphere Data Architect only supports storage modeling for DB2 for Linux, UNIX, and Windows, DB2 for z/OS®, and DB2 for i5/OS® databases. You can still create tables, columns, and other related objects within the workbench, but data objects from other database vendors, such as table spaces and buffer pools, are not supported.

For this book, you will be working with models that are designed for a DB2 system.

Typically, when you work with a physical data model, you use the following task flow:

1. Create a physical data model, either from scratch, template, transforming from a logical data model, or importing from another system.
2. Refine the physical data model by adding storage objects, indexes, and views.
3. Generate the DDL to deploy the database and/or schema.
4. Run the DDL script that creates the data objects on the server.

So you might be asking yourself, “What are the differences between a logical and physical data model?” *Table 5.1* summarizes the differences.

Logical data models	Physical data models
Do not correspond to any database vendor	Always correspond to a database vendor and database version
Based on a higher level of abstraction; more generic	More specific; closer to the design of an actual database
Physical storage is not taken into account	Used to model physical storage for supported database vendors
Does not require database vendor construction knowledge; requires to know how the model will be used	Requires knowledge about database vendors and how models are constructed for each platform; requires information about how data

	will be stored
--	----------------

**Table 5.1 – Differences between logical and physical data models**

## 5.2 Creating a physical data model from scratch

Ideally, you would first create a logical data model before you create a physical data model. This helps you keep the logical and physical designs separate, allowing you to modify data models at the logical level before you design the physical storage aspects of a database or schema.

However, it is possible to create a blank physical data model within the workbench.

**Note:**

The steps below are provided for reference only. You do not need to perform these steps to continue with the design of the student information system. In the next section, you will learn how to transform to the physical data model.

To create a blank physical data model:

1. Click *File -> New -> Physical Data Model*. The New Physical Data Model wizard opens.
2. Complete the wizard. You can create a blank physical data model, or you can reverse-engineer an existing database, schema, or DDL script. Reverse-engineering is covered in a later chapter.
3. Use the workbench to add data objects, such as tables, schemas, or columns. Right-click on various objects within the data model to see which data objects you can add from the context menu, or create a diagram to visualize these data objects and their relationships.

## 5.3 Transforming a logical data model to a physical data model

Instead of creating a physical data model from scratch, you will use the existing logical data model (**STUDENT\_INFO\_SYSTEM.ldm**) that you created in the previous chapters and transform it to a physical data model. The transformation process creates the physical design of a database or schema for you, which saves you the work of manually creating tables, columns, and other data objects.

After you transform from the logical data model, you will add more specific properties to the physical data model and bring it one step closer to a functional, deployable data model.

**Note:**

The data types that are mapped from one data model to another depend on what you have specified in the Preferences window. To ensure that data types are mapped correctly between the logical data model and the new physical data model:

1. Open the Preferences window from the main menu by clicking *Window > Preferences*.
2. Expand the Data Management node and find the *Transform > Data Type Map* page.
3. Expand the Data Type Map node in the Preferences window to specify the options for data type mapping between logical and physical data models and the various database vendors.

To transform a logical data model into a physical data model:

1. Select the **STUDENT\_INFO\_SYSTEM.ldm** file in the data design project.
2. From the main menu, click *Data -> Transform -> Physical Data Model*. The Transform to Physical Data Model wizard opens.
3. On the Target Physical Data Model page, select the *Create new model* radio button, then click *Next*.
4. Complete the Physical Data Model File page:
  - a. Specify the *University Info System* data design project as the destination folder.
  - b. Specify *DB2 for Linux, UNIX, and Windows* as the database type.
  - c. Select *V9.7* as the database version.
  - d. Click *Next*.
5. On the Options page, in the *Schema name* field, specify **STUDENT\_LIFE**, then click *Next*. The Options page is shown in *Figure 5.1*.



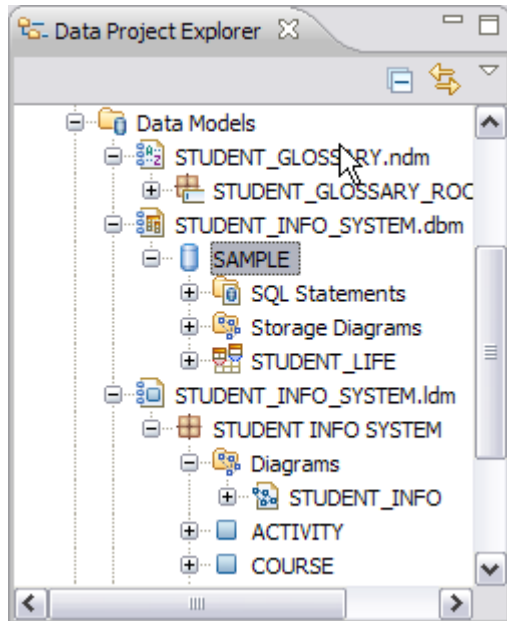
**Figure 5.1 – Completing the Options page**

**Note:**

As a best practice, you should select the *Generate traceability* option. This option allows you to identify dependencies between the logical and physical data models, which helps you perform impact analysis when refining your data models.

6. On the Output page, click *Finish*. The physical data model is created in the *Data Models* folder of the data design project, and the file opens in the editor view.
7. Give the new database a more descriptive name.
  - a. Select the Database object under the physical data model.
  - b. In the *General* tab of the Properties view, specify **SAMPLE** as the new name.
8. Save your work.

Figure 5.2 shows the data objects and folders that are created after you complete the transformation process.

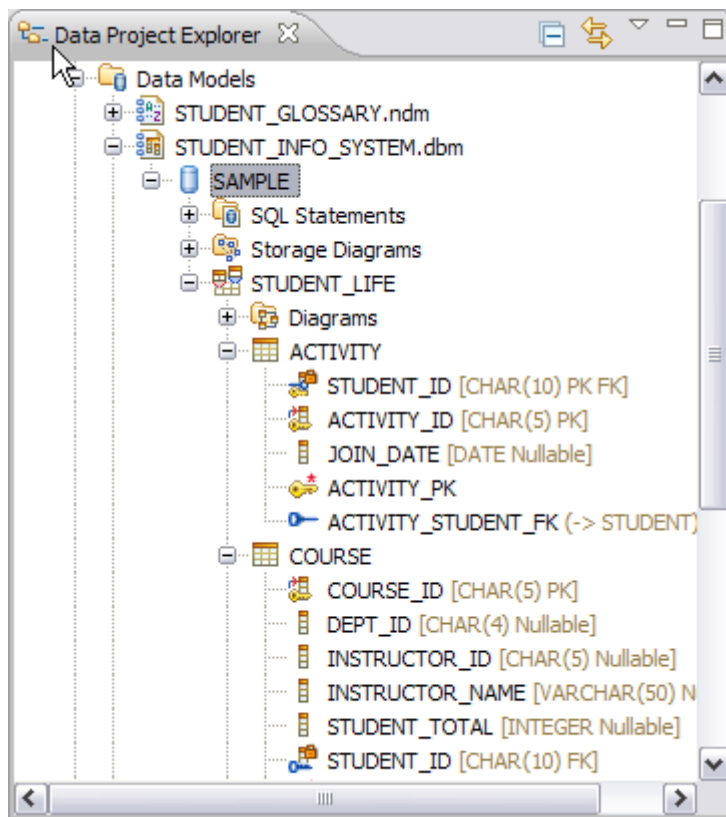


**Figure 5.2 – New physical data model and data objects**

## 5.4 Working on your physical data model

### 5.4.1 Anatomy of your model

Figure 5.3 shows you what a physical data model looks like.



**Figure 5.3 – The physical data model, displayed in the workbench**

Every physical data model has the following characteristics:

- File extension: The file extension at the end of each physical data model file name is *.dbm*.
- Schema and database design:
  - The structure is more or less the same as a logical data model.
  - As you have learned previously, logical data models contain a single package, where you can store subpackages, entities, and diagrams. Physical data models, on the other hand, store information in databases. Each database can have multiple schemas, and each schema can contain multiple tables.
- Diagrams:
  - When you transform a diagram from a logical data model, your design is intact, including notes, formatting, and relationships. The only difference is in the icons used to depict the new tables, columns, and other data objects that you design.

- As with logical data models, you can create physical data model diagrams to help you visualize and understand the design of complex data models.
- Storage diagrams: You can create diagrams to help you model the storage of your DB2 databases. This includes partitions, table spaces, buffer pools, and other storage objects.
- SQL statements: The *SQL Scripts* folder stores SQL scripts that you create within the physical data model.

If you explore the physical data model, you will see that even the atomic domain elements that you specified are carried over into the new physical data model.

### 5.4.2 Storage modeling in DB2

Storage modeling for DB2 involves creating *table spaces*, *partitions*, and *partition groups* of a database setup.

**Note:**

With IBM InfoSphere Data Architect, you can only model the storage of a DB2 for Linux, UNIX, and Windows database, or DB2 for z/OS database.

For more information about the DB2 storage model, refer to the book [Getting started with DB2 Express-C](#), that is part of this book series.

This chapter does not cover how to model storage in detail, because the schema we are creating is very small. However, this section does explain the basic concepts behind storage modeling and the information that you should keep in mind when you design the storage for complex data models.

#### 5.4.2.1 Table spaces

The data for databases is stored in *table spaces*. Table spaces are database vendor-specific.

**Note:**

To learn more about table spaces in DB2 for Linux, UNIX, and Windows, visit the following URL:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.dbobj.doc/doc/c0004935.html>

To learn more about table spaces in DB2 for z/OS:

[http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z.doc.gloss/src/gloss/db2z\\_gloss.htm](http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z.doc.gloss/src/gloss/db2z_gloss.htm)

Table spaces are typically managed in one of two ways:

- *System-managed space*: The operating system will manage the table spaces. Containers are defined as operating system files, and they are accessed using operating system calls. This means that the operating system handles the following actions:
  - Buffering of input and output
  - Space allocation
  - Automatic extension of the table space
- *Database-managed space*: The database manages the table spaces. Containers are defined as files, which will be fully allocated with the size given when the space is created, or as devices. When you use DB2 to manage your table spaces, it will handle the following actions:
  - Buffering of input and output (as much as the allocation method and operating system allows)
  - Extending containers when using ALTER TABLESPACE
  - Releasing unused portions of database-managed containers

Each table space has at least one container. A *container* belongs to a single table space (but a table space can have more than one container). You can only add a container to a system-managed table space when the database is partitioned, and the partition does not yet have a container allocated for that table space. When you add a container to a table space, the data is distributed across all of the containers.

When you specify a table space, you need to consider the following settings (found on the *General* tab of the Properties view of the table space):

- *Page size*: Defines the size of pages that are used for the table space. For DB2, the supported sizes include 4K, 8K, 16K, and 32K. The page size limits the row length and column count of the tables that can be placed in the table space. Table spaces are limited to 16,777,216 pages. Choosing a larger page size increases the capacity of the table space.
- *Extent size*: Specifies the number of pages that will be written to a container before the next container is used. The database manager cycles repeatedly through the containers as data is stored. This setting only has an effect when there are multiple containers for each table space.
- *Prefetch size*: Specifies the number of pages that are read from the table space when data prefetching is performed. Prefetching draws data that is needed by a query before they are actually referenced by the query. This allows the query to have the data ready when data input or output is performed. Prefetching is selected by the database manager when it determines that it needs to use sequential input and output, and prefetching the data will improve performance.

**Best practice:**

Set the prefetch size as a multiple of the extent size value and the number of table space containers. For example, if your extent size is 32 and there are 4 containers, then the prefetch size should be 128, 256, or other multiples of 128. If a table is used often, create a separate table space for that table to improve performance.

- *Overhead and transfer rate*: Determines the cost of input and output during query optimization. Values are measured in milliseconds, and they should be the average for all containers. *Overhead* is the time that is associated with input-output controller activity, disk seek time, and rotational latency. The *transfer rate* is the amount of time that is needed to read a page into memory. These values can be calculated based on hardware specifications.

**5.4.2.2 Buffer pools**

A *buffer pool* is associated with a single database, and it can be used by more than one table space. When you create a buffer pool for a table space, you must ensure that the table space size and the buffer pool page size are the same for all of the table spaces that the buffer pool services. A table space can use only one buffer pool.

Adequate buffer pool size is essential to good database performance, because it reduces the amount of input and output by the disk, which is the most time-consuming database operation. Large buffer pools also have an effect on query optimization, because more of the work can be done in memory.

**5.4.2.3 Performance implications of storage modeling**

When you design the storage of a schema or database, you should focus primarily on utilizing buffers and maximizing the amount of parallelism for input and output. Use the following task flow to ensure that your storage models are efficient:

1. Determine the constraints given by the table designs. You might have to use more than one regular table space.
2. Consider whether having the tables in large table spaces with different settings is likely to increase performance.
3. Design a tentative table space.
4. Consider using buffer pools, which might help you improve upon your table space design.
5. Allocate containers to the table spaces.
6. Refine the model, and verify that your design is solid by thoroughly testing and benchmarking the model.

As a general rule, start out with the simplest design. Only allow complexity if and when performance is impacted.

## 5.5 Refining the physical data model

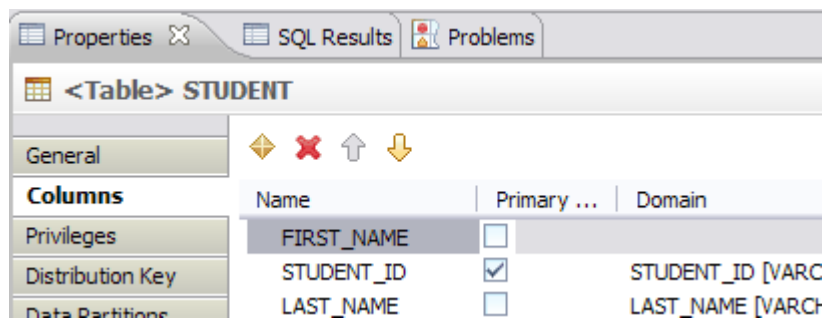
Now that we have transformed a logical data model into a physical data model, we should make some final refinements before we generate the DDL to deploy the database. First, we will rearrange the columns of the STUDENT table into a more logical order. Then, we will create a role and add a user to the database.

### 5.5.1 Rearranging columns in a physical data model

When you created the logical data model, you were more concerned with modeling *what* attributes and data characteristics must go into each entity. Let's rearrange the columns of the STUDENT table to make the object more easily understood to an end user.

To rearrange the columns, perform the following steps:

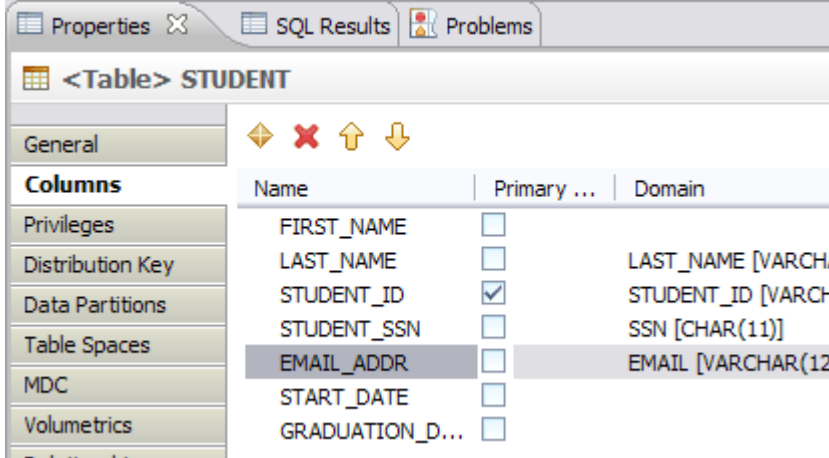
1. Select the STUDENT table under the STUDENT\_LIFE schema. The properties of the table display in the Properties view.
2. Open the *Columns* tab of the Properties view.
3. Rearrange the columns to identify vital student information first:
  - a. Make the FIRST\_NAME column the first column in the table. Select the FIRST\_NAME column, then press the Move Column Up arrow (↑) to place it first in the table. The columns should now look like the image in *Figure 5.4*.



**Figure 5.4 – Rearranging columns in a table**

- b. Make the LAST\_NAME column the second column in the table by moving it under the FIRST\_NAME column.
- c. Make the STUDENT\_SSN column the fourth column in the table.
- d. Make the EMAIL\_ADDR column the fifth column in the table.

The columns should be in the order displayed in *Figure 5.5*.



<Table> STUDENT			
General			
Columns	Name	Primary ...	Domain
Privileges	FIRST_NAME	<input type="checkbox"/>	
Distribution Key	LAST_NAME	<input type="checkbox"/>	LAST_NAME [VARCHAR]
Data Partitions	STUDENT_ID	<input checked="" type="checkbox"/>	STUDENT_ID [VARCHAR]
Table Spaces	STUDENT_SSN	<input type="checkbox"/>	SSN [CHAR(11)]
MDC	EMAIL_ADDR	<input type="checkbox"/>	EMAIL [VARCHAR(12)]
Volumetrics	START_DATE	<input type="checkbox"/>	
	GRADUATION_D...	<input type="checkbox"/>	

**Figure 5.5 – The proper order of the columns in the table**

4. Save your work.

### 5.5.2 Creating roles within the physical data model

Roles simplify the administration and management of privileges. A *role* is a database object that groups together one or more privileges and can be assigned to users, groups, PUBLIC, or other roles. Roles provide several advantages that make it easier to manage privileges in a database system:

- *Security administrators* (SECADM) can control access to their databases in a way that mirrors the structure of their organizations. They can create roles in the database that map directly to the job functions in their organizations.
- *Users* are granted membership in the roles that reflect their job responsibilities. As their job responsibilities change, their membership in roles can be easily *granted* and *revoked*.
- The assignment of privileges is simplified. Instead of granting the same set of privileges to each individual user in a particular job function, the administrator can grant this set of privileges to a role that represents that job function and then grant that role to each user in that job function.
- A role's privileges can be updated and all users who have been granted that role receive the update. The administrator does not need to update the privileges for every user on an individual basis.
- The privileges and authorities granted to roles are always used when you create views, triggers, materialized query tables (MQTs), static SQL and SQL routines, whereas privileges and authorities granted to groups (directly or indirectly) are not used. Because roles are managed inside the database, the DB2 database system can determine when authorization changes and act accordingly.



- All of the roles assigned to a user are enabled when that user establishes a connection, so all privileges and authorities granted to roles are taken into account when a user connects. Roles cannot be explicitly enabled or disabled.
- The security administrator can delegate management of a role to others.

All DB2 privileges and authorities that can be granted within a database can be granted to a role. For example, a role can be granted any of the following authorities and privileges:

- DBADM, SECADM, LOAD, and IMPLICIT\_SCHEMA database authorities
- CONNECT, CREATETAB, CREATE\_NOT\_FENCED, BINDADD, CREATE\_EXTERNAL\_ROUTINE, or QUIESCE\_CONNECT database authorities
- Any database object privilege (including CONTROL)

A user's roles are automatically enabled and considered for authorization when a user connects to a database; you do not need to activate a role by using the SET ROLE statement. For example, when you create a view, a materialized query table (MQT), a trigger, a package, or an SQL routine, the privileges that you gain through roles apply.

A role does not have an owner.

After speaking with your information system team, it is determined that you should create a role for application developers, so that they can connect to your system. This role that you create can be used to grant certain elevated privileges to the application developers so that they can deploy stored procedures, routines, and other data objects that easily access your schemas and databases.

To create the APP\_DEV role:

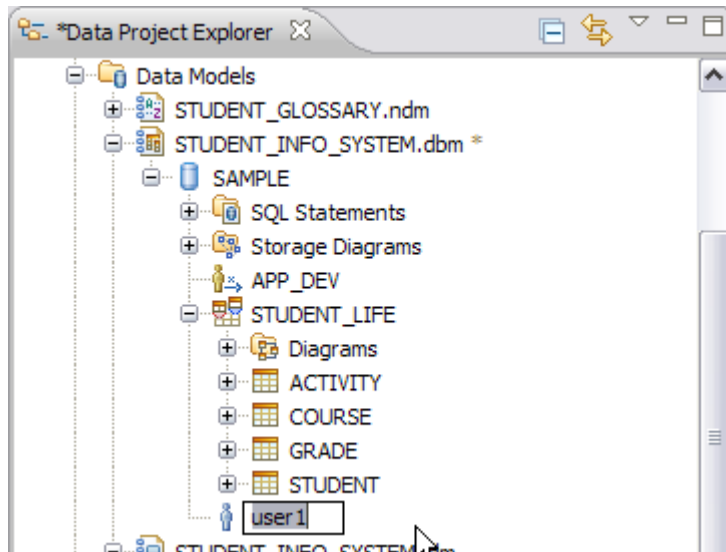
1. Right-click on the SAMPLE database object, then select *Add Data Object -> Role*. A new role is created in the Data Project Explorer.
2. Name the role **APP\_DEV**.
3. Add the privileges to the role:
  - a. Open the *Privileges* tab of the Properties view. Make sure the *Database* tab is selected.
  - b. Click the New (🔹) button. The Grant New Privileges window opens.
  - c. Select the BINDADD, CONNECT, CREATE\_EXTERNAL\_ROUTINE, and CREATE\_NOT\_FENCED\_ROUTINE options, then click *OK*. The database privileges are added to the role.
4. Save your work.

The role is created within the physical data model. Once you deploy the DDL that you generate in section 5.6 *DDL generation*, you can assign users to this role.

### 5.5.3 Adding a user ID to the physical data model

To ensure that specific users can access the database once it is deployed, you should add those user IDs to the physical data model. You can also add other users, but let's start with the default user1 ID that is created for most DB2 databases:

1. Right-click on the SAMPLE database and select *Add Data Object -> User*. A new user object is created in the physical data model.
2. Specify **user1** as the name of the user. An example of this is shown in *Figure 5.6*.



**Figure 5.6 – Specifying a user**

3. In the *Privileges* tab of the Properties view, make sure that you are in the *Database* section.
4. Click the New ( ) button. The Grant New Privileges window opens.
5. Select all of the privileges, then click *OK*.
6. Save your work.

The privileges are added to the user1 user ID. Now that user will have full access to the database.

### 5.5.4 Validating the physical data model

Before you deploy the physical data model, you should validate it. When you validate the model, the workbench checks the model to ensure that valid DDL can be generated from the data model. The validation process also ensures that your model not only adheres to the design standards that you have created, but that it adheres to the common data modeling standards. These “best practice” rules are built into the product, making it easier for you to ensure that your model is well-designed before you deploy it.

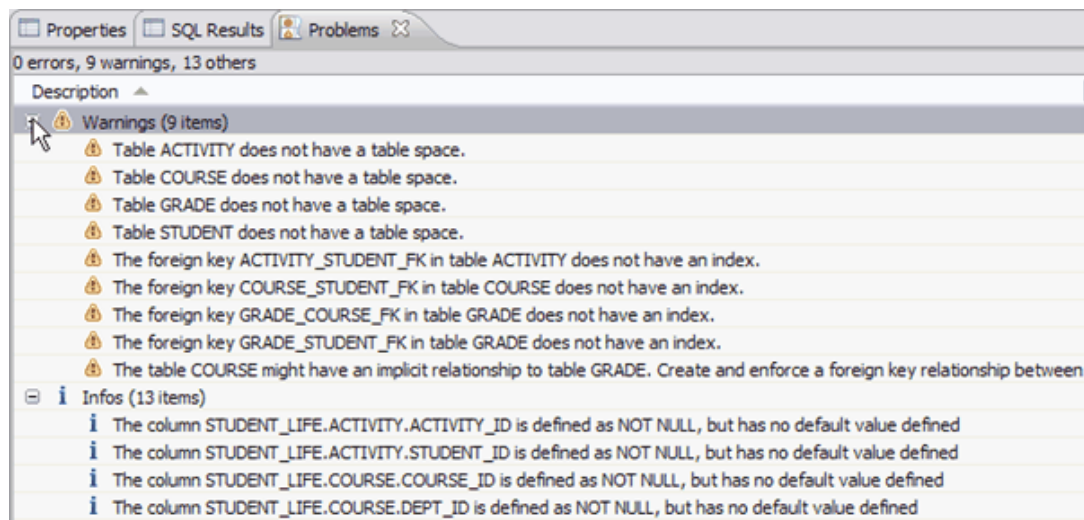
To validate the physical data model:

1. Right-click on the SAMPLE database in the Data Project Explorer, then select *Analyze Model*. The Analyze Model wizard opens. By default, all of the *Physical data model* rules are selected.
2. Deselect the *Dimensional Modeling* and *Naming Standard* options, then click *Finish*.

**Note:**

Dimensional modeling is not a topic that is covered in this book. For more information about dimensional modeling and its benefits, visit the IBM InfoSphere Data Architect information center at <http://publib.boulder.ibm.com/infocenter/rdahelp/v7r5/index.jsp>.

The model is analyzed, and you see a list of warnings and informational messages in the Problems view, as shown in *Figure 5.7*.



**Figure 5.7 – List of warnings in the Problems view**

For the purposes of this exercise, you do not have to take any steps to correct warnings and informational messages; these messages simply let you know of potential issues that can impact performance or alert you of where SQL statements might fail. For example, when information is INSERTed for the data objects in *Figure 5.7*, the update might fail, because no default value is explicitly set for the columns that are set to NOT NULL. Since your data model is relatively small and will not go into a real production environment, you do not need to correct these warnings by creating table spaces and indexes.

However, it is considered a best practice to evaluate each warning to ensure the best performance of your schemas and databases.

## 5.6 DDL generation

Now that you have generated a complete physical data model, you are aware of the process of creating, editing, and updating data models. Now that you've completed this process, how do you deploy the database to a test or production environment?

IBM InfoSphere Data Architect is capable of creating database-specific *data definition language (DDL)* scripts from the model that you have created. After you generate the DDL script, you can run it on the server, and users can access the model as part of the database.

### 5.6.1 Generating the DDL script from the database object

You generate the DDL script from the database object of the physical data model. The Generate DDL wizard will create the SQL statements to deploy your database on a server.

Let's create a DDL script to deploy your new database:

1. Connect to the SAMPLE database by right-clicking on the connection in the Data Source Explorer and selecting *Connect*.

**Note:**

If you are not deploying the DDL as part of the DDL generation, you do not need to connect to the database as the first step.

2. Right-click on the SAMPLE database in the Data Project Explorer, then select *Generate DDL*. The Generate DDL wizard opens.
3. On the Options page of the wizard, make sure that all of the options are selected, then click *Next*.

Table 5.1 explains the options on the Options page:

Option	Explanation
Check model	The model is validated before the script is created.
Fully qualified names	Any object is referenced by its full name. For example, the STUDENT table of the STUDENT_LIFE schema is referenced as STUDENT_LIFE.STUDENT.
Quoted identifiers	Specifies if you need to address object within quotes. This is done if you want to consider "Table A" and "TABLE A" as two different objects.
DROP statements	Includes DROP statements at the beginning of the DDL script to drop objects if they are

	already present.
CREATE statements	Includes CREATE statements in the DDL to create or re-create data objects.
COMMENT ON statements	Includes the statements from the <i>Documentation</i> tab of the Properties view of the data objects.
IN TABLESPACE clause	Takes info from the storage model or the default table space and adds it to the DDL script.

**Table 5.1 – Available model elements that you can add to the DDL script**

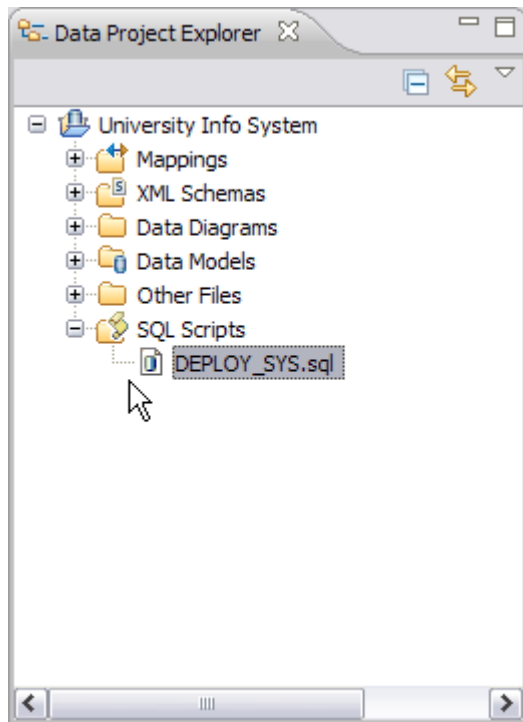
4. Keep the default settings on the Objects page, then click *Next*.
5. On the Save and Run DDL page, specify the following file name:  
`DEPLOY_SYS.sql`.

**Note:**

The DDL script is displayed in the *Preview DDL* section of the page. Use this section to verify that your script contains all of the SQL that you need to deploy the information system to the server. The first part of the SQL makes sure that none of the data objects exist on the server. Then, the SQL will re-create these data objects and make sure that the role you have specified is, in fact, created on the server.

6. Select the *Run DDL on server* option, then click *Next*.
7. Complete the Select Connection page by selecting the SAMPLE database.
8. Review your choices on the Summary page. You cannot change the options on this page. Once you are ready to generate the script, click *Next*.
9. On the Select Connection page, select the SAMPLE database connection, then click *Next*.
10. On the Summary page, click *Finish*.

The DDL script is generated and saved in the *SQL Scripts* folder of the data design project, as shown in *Figure 5.8*.



**Figure 5.8 – The DDL script, saved in the *SQL Scripts* folder**

The DDL script also runs on the server. This deploys your new database and schema to the DB2 system, where it can be populated with data, and application developers can create applications to query and report on the data. To verify that the schema has deployed, open the Data Source Explorer view, and expand the SAMPLE node and database. Locate and expand the *Schemas* folder and look for the STUDENT\_LIFE schema, now created in the database. This is shown in *Figure 5.9*.

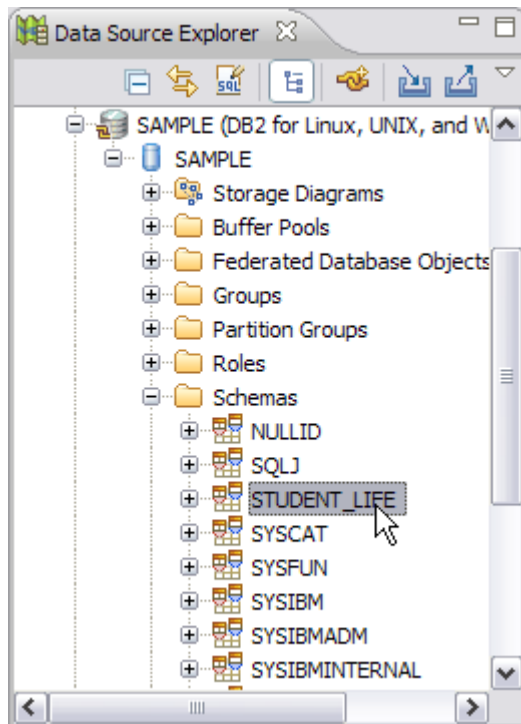


Figure 5.9 – Newly-deployed schema

## 5.7 Exercise

Generate a new DDL script for the physical data model:

1. On the Options page of the Generate DDL wizard, make sure the following items are not selected:
  - *DROP statements*
  - *COMMENT ON statements*
  - *Use domain if exists*
2. On the Objects page of the Generate DDL wizard, only generate statements for the following objects:
  - *Tables*
  - *Privileges*
  - *Primary key constraint*
  - *Roles*
3. Complete the Save and Run DDL page of the wizard:
  - a. Specify **EXERCISE\_5\_7.sql** as the file name.

- b. Select the *Open DDL file for editing* option.
4. Complete the wizard.
5. Select both SQL files that you have created (DEPLOY\_SYS.sql and EXERCISE\_5\_7.sql), then right-click and select *Compare With -> Each Other*. The compare editor opens. What are the differences in the files?

## 5.8 Summary

In this chapter, you learned about physical data models. Physical data models are the next step in the data design process; they take the logical design and help you draft the physical storage (tables, columns, indexes, and other data objects) for the database and server.

You also learned about data definition language (DDL) files and how they can be used to deploy the data models that you create with IBM InfoSphere Data Architect. DDL scripts contain the queries that are necessary to drop, create, or modify data objects.

## 5.9 Review questions

1. What is a physical data model?
2. Physical data models can model one or all of the following objects:
  - A. Tables
  - B. Roles
  - C. Schemas
  - D. All of the above
3. From which objects of the physical data model do you generate DDL scripts?
4. True or false: You cannot preview the DDL script before you create it in your workspace.
5. What is the purpose of storage modeling?
6. What storage model objects can you create in a physical data model?
7. True or false: You must have a logical data model before you create a physical data model. You cannot create a physical data model from scratch.

## 5.10 What's next?

Congratulations! You have designed and deployed a complete data model.

In the next chapter, you will learn how to generate informative reports about your data models.



# **PART III – ITERATIVE DESIGN: REPORTING, REVISING, AND ANALYZING**



# 6

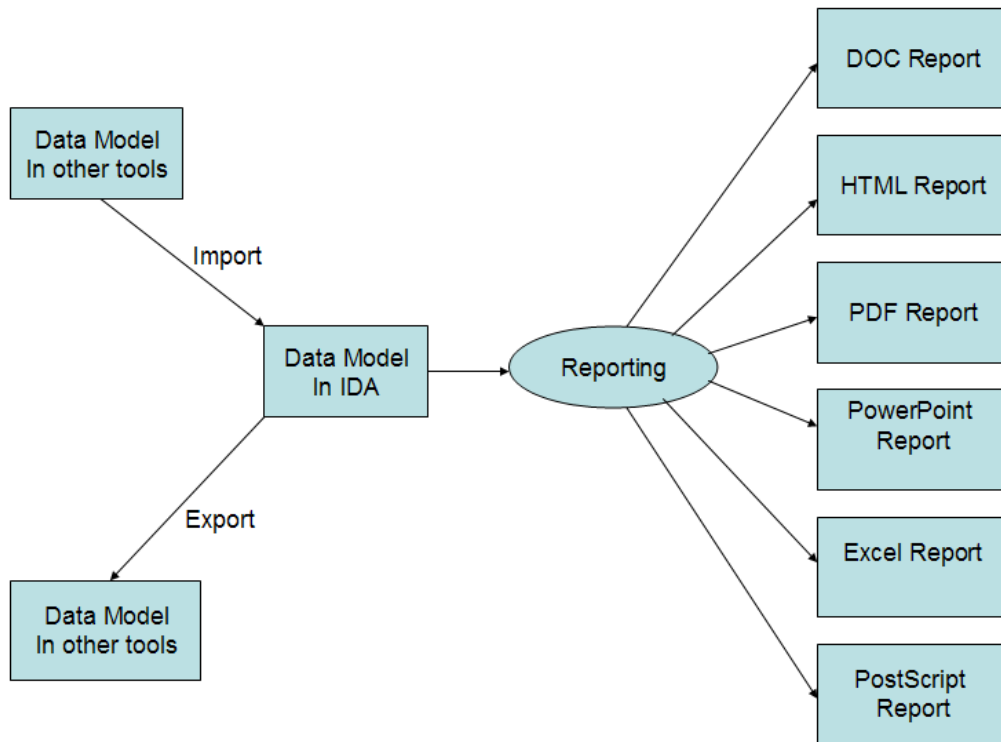
## **Chapter 6 – Generating Reports, Importing, and Exporting**

Reporting is an important feature of IBM InfoSphere Data Architect. The information provided by reports can be copied, printed, and distributed as one physical document. Reports are used to provide compliance information in many organizations.

The Import/Export feature in IBM InfoSphere Data Architect helps you ensure that your data models are compatible with other data modeling, design, and reporting tools. The Import/Export feature allows you to import existing data models from compatible data modeling tools and export to compatible data modeling tools and formats.

## 6.1 Reporting, importing, and exporting: The big picture

When you create reports within the workbench, you can provide information on all or part of a model; that is, a list of objects and their relationships. IBM InfoSphere Data Architect provides a wide range of built-in reports or templates for your logical, physical, domain, and glossary models. *Figure 6.1* illustrates the big picture of reporting and import/export.



**Figure 6.1 – Reporting, importing, and exporting in the workbench**

The left side of *Figure 6.1* illustrates the import and export capabilities of IBM InfoSphere Data Architect. The *import* feature of IBM InfoSphere Data Architect allows you to convert the data model from another tool into either a physical data model or logical data model in IBM InfoSphere Data Architect. The *export* feature of IBM InfoSphere Data Architect allows you to convert either a physical data model or logical data model in IBM InfoSphere Data Architect to the format that is supported by other tool that you want to export the data model to.

The right side of *Figure 6.1* illustrates the reporting capabilities of IBM InfoSphere Data Architect. You can generate reports for your logical, physical, domain, and glossary models in 6 different formats: HTML, PDF, PowerPoint, Excel, Word (.doc), and PostScript.

**Note:**

You can only import or export physical or logical data models or domain models.

## 6.2 An insight into reporting

The reporting feature of IBM InfoSphere Data Architect allows you to generate reports that help you analyze a data model or a package. There are various built-in templates available in IBM InfoSphere Data Architect which can be used to generate a report. You will see the information related to various built-in templates in IBM InfoSphere Data Architect in the next section.

Reporting in IBM InfoSphere Data Architect is done via two methods:

- Business Intelligence Reporting Tool (BIRT) reporting
- XSLT reporting

*BIRT reports* are a widely used reporting format in IBM InfoSphere Data Architect. It gives greater flexibility and more options for the output format of a report. It supports multiple output formats.

*XSLT reports* only support PDF output.

IBM InfoSphere Data Architect supports various built-in templates to generate reports for your logical, physical, glossary and mapping models. The file extension for a BIRT report template is **.rptdesign**, and for an XSLT report, the file extension is **.xsl**.

### Note:

You can also create your own report templates. You will create one later in the Exercise section of this chapter.

## 6.3 Generating a BIRT report

Let's learn how to create a BIRT report by using an existing data model. We will use the existing **STUDENT\_INFO\_SYSTEM.dbm** physical data model to create a report in HTML format with the *Physical Data Model* report type.

First, you will see how easy it is to generate a physical data model report. Then, you will learn how to customize your reports by using the Report Design perspective to specify the data objects for which to create a report.

### 6.3.1 Generating a basic physical data model report

When you create a report model template, you can re-use them to produce reports that contain the same information and basic formatting. This helps you provide a consistent experience when presenting model reports to business process owners.

Use the Report Configurations window to create a report of your existing physical data model:

1. Click **Run -> Report -> Report Configurations**. The Report Configurations window opens.
2. Double-click the *BIRT Report* option in the left pane. A new report configuration is created and opens in the Report Configurations window.
3. Specify the options for the report:
  - a. In the *Name* field, specify **SDT\_Info\_Sys**.

- b. If it is not already selected, select the *Built-in* radio button, then click *Browse*. The Select Built-In Report window opens.
  - c. Expand the *Physical Data Model Reports* node, then select *Physical Data Model Report*. Click *OK* to add the report to the configuration.
  - d. In the *Report Data* section, in the *Data Sources* field, select the *Data Source* option., then click the *Add* button. The Load Resource window opens.
  - e. Click the *Browse Workspace* button, then expand the *University Info System* node and select the *STUDENT\_INFO\_SYSTEM.dbm* physical data model. Click *OK*.
  - f. Click *OK* to add the physical data model to the report configuration.
  - g. Click the *Workspace* button to specify a location to save the report. The Select Report Output Location window opens.
  - h. In the *File name* field, specify *SDT\_Info\_Sys\_Rpt*, then click *OK*.
4. Click *Report* to generate the report.

The report is generated and opens in your default browser. As you can see, it's a somewhat generic report, and it contains all of the information about your physical data model.

IBM InfoSphere Data Architect also allows you to customize your report designs, making it even easier for you and your teams to make sense of the data model. Let's create a new report configuration to showcase important information about the columns in your physical data model.

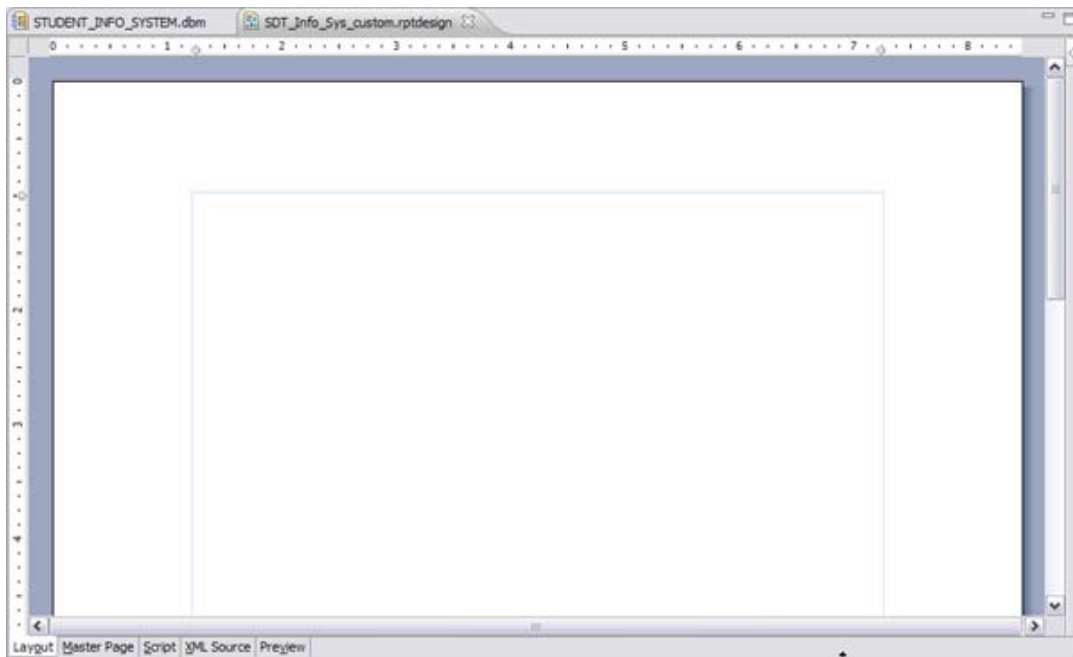
Before you create the report configuration, you will set up the workbench environment to design the report. You will work in the Report Design perspective to create a report that displays the information about the physical data model in a report that you can then share with other members of your team. This report is useful to analyze the structure of the data model and place it in a more visual, understandable format.

### 6.3.2 Setting up the reporting environment

First, let's set up the report environment to design the report:

1. Open the Report Explorer view:
  - a. Click *Window -> Show View -> Other*. The Show View window opens.
  - b. Expand the *Reporting* node, then select the Report Explorer option. The Report Explorer view opens, and you see a list of report templates.
2. Copy the *Blank physical Data Model Report (BlankPhysicalModel.rptdesign)* template to the *Other Files* folder:
  - a. In the Report Explorer view, expand the *Physical Data Model* node, then right-click *Blank physical Data Model Report (BlankPhysicalModel.rptdesign)* and select *Copy*.
  - b. In the Data Project Explorer, expand the *University Info System* folder, right-click the *Other Files* folder and select *Paste*.

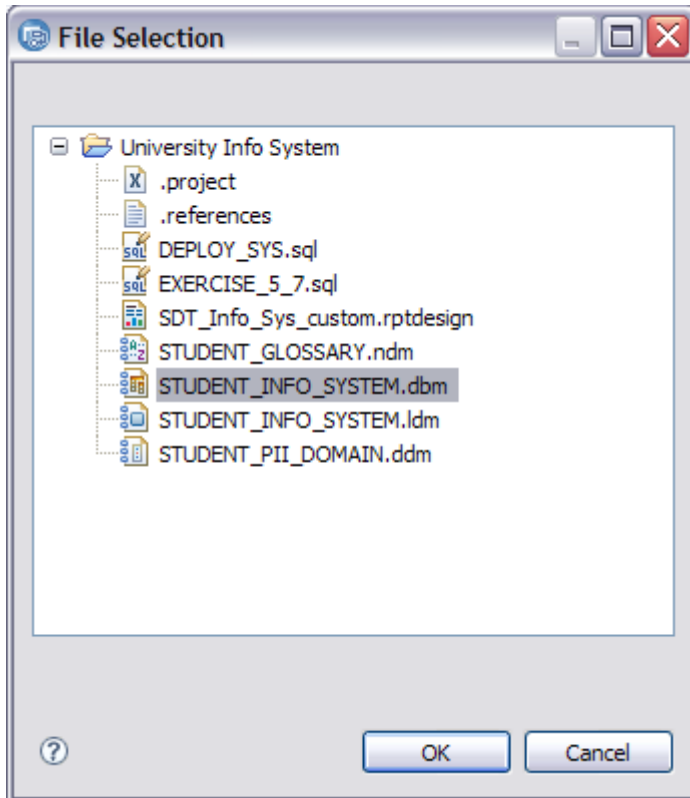
- c. In the Data Project Explorer, right-click *University Info System -> Other Files -> BlankPhysicalModel.rptdesign* and click *Rename*. The Rename Resource window opens.
  - d. In the Rename Resource window, specify **SDT\_Info\_Sys\_custom** for the new resource name and click *OK*.
3. Open the Report Design perspective:
  - a. From the main menu, click *Window -> Open Perspective -> Other*. The Open Perspective window opens.
  - b. In the Open Perspective window, select *Report Design*. The perspective opens. The Palette, Data Explorer, Outline, and Property Editor views are empty until you open a report design in the Report Editor view. Not all of the views are visible. For example, the Navigator and Outline views are configured so that only one view is visible. Click the tab to display the other view.
4. Open the **SDT\_Info\_Sys\_custom.rptdesign** file in the Report Editor view:
  - a. Open the Navigator view, located in the bottom half of the left pane by default.
  - b. Expand the University Info System folder and double-click the **SDT\_Info\_Sys\_custom.rptdesign** file. The report opens in the Report Editor, as shown in Figure 6.2.



**Figure 6.2 – Opening the report in the report editor**

5. Add the **STUDENT\_INFO\_SYSTEM.dbm** physical data model as a BIRT data source:
  - a. Locate the Data Explorer, located in the top half of the left pane by default.

- b. In the Data Explorer, expand the *Data Sources* folder, then right-click the *Data Source* object and click *Edit*. The Edit Data Source - Data Source window opens.
- c. On the *Models* page, in the *Instance models* section, click *Add*. The Load Resource window opens.
- d. In the Load Resource window, click *Browse workspace*. The File Selection window opens.
- e. Select **STUDENT\_INFO\_SYSTEM.dbm** under the *University Info System* node and click *OK*. This is shown in *Figure 6.3*.



**Figure 6.3 – Selecting the physical data model**

- f. Click *OK* to load the data model resource.
  - g. Click *OK* to save the data source.
5. Save your work.

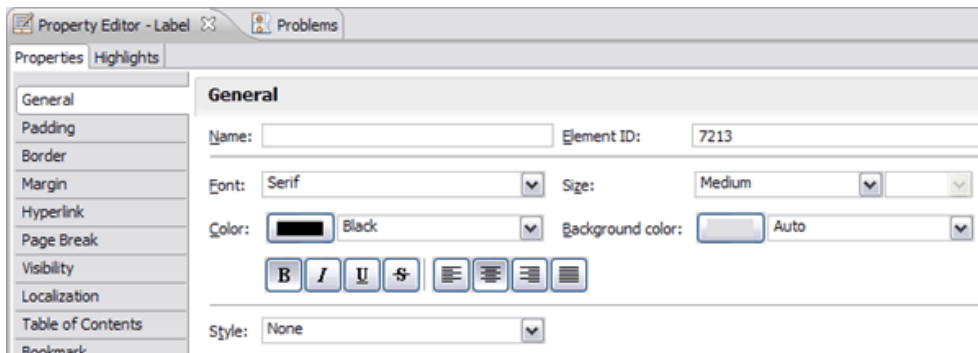
The **STUDENT\_INFO\_SYSTEM** physical data model file is added as a BIRT data source. The report environment is now set up for you to start drafting your BIRT reports.

### 6.3.3 Adding data objects to a report

You have a blank report. In order to report on the information in the data model, you should add data objects to this report. Let's start by creating a table that lists the columns in the tables of your schema:



1. First, add a title to the report:
  - a. Open the Palette view by clicking the *Palette* tab, found in the top half of the left pane by default.
  - b. From the Palette view, select and drag a *Label* element into the *Layout* page of the Report Editor.
  - c. Double-click the label in the Report Editor and enter the following text for the label: **Table and column report.**
  - d. In the Property Editor, select the *General* tab.
  - e. Set the font weight to *bold*.
  - f. Align the text in the center of the label. Click the *Center* button. The General tab should look like the following image:



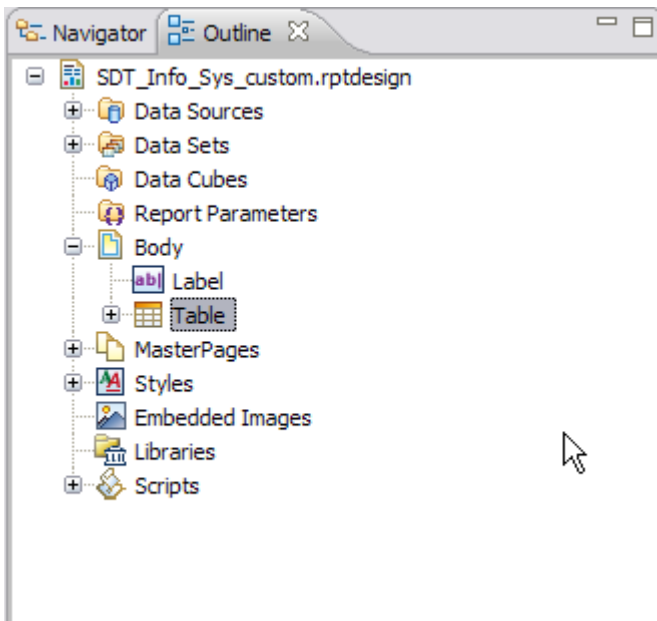
**Figure 6.4 – Specifying text properties for a label**

2. Add a BIRT report table to the report:
  - a. From the Palette view, drag a *Table* element into the *Layout* window after the report title. The Insert Table window opens.
  - b. Set the number of columns to 2.
  - c. Specify *Column* from the list of data set options. A two-column table is inserted into the report. The table contains a header row, a detail row, and a footer row. The Column data set is associated with the report table.

**Note:**

The Column data set is a predefined data set in the report design. In the Data Explorer view, you can see the data sets that are defined for the report design.

3. Specify a name for the report table, and update the format of the table:
  - a. Open the Outline view, found in the bottom half of the left pane by default.
  - b. Click *SDT\_Info\_Sys\_custom.rptdesign -> Body -> Table*.



**Figure 6.5 – Selecting objects in the Outline view**

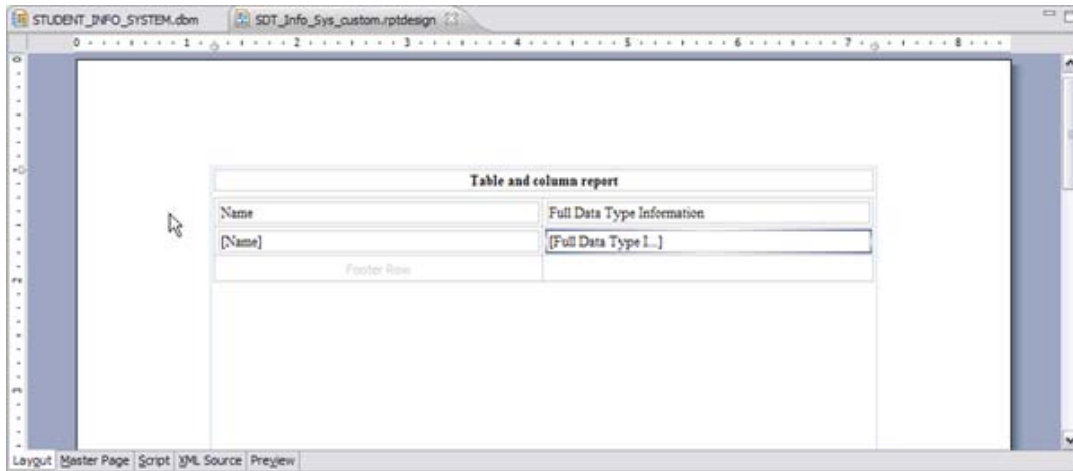
- c. In the Property Editor view, select the *General* tab.
- d. In the *Name* field, type **Column Report**.

**Note:**

The format of a BIRT report table can be set at the table level. You can override the properties within the table by selecting the element and updating the properties.

4. Add the *Name* and *Full Data Type Information* elements from the *Column* data set to the table:
  - a. Open the Data Explorer view.
  - b. Expand the *Data Sets -> Column* node and drag the *Name* element to the left column of the detail row in the table.
  - c. Expand the *Data Sets -> Column* node and drag the *Full Data Type Information* element to the right column of the detail row in the table.

The report should now look like the image in *Figure 6.6*.

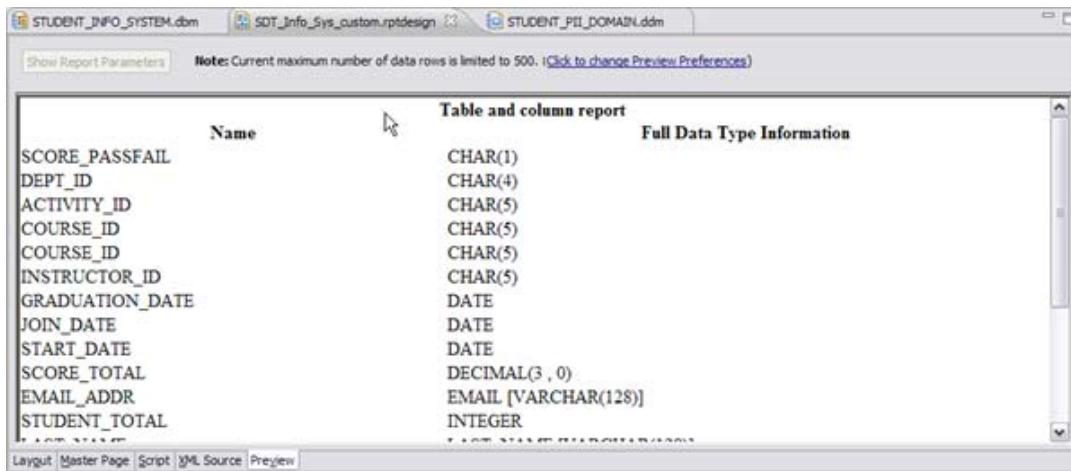


**Figure 6.6 – Adding elements to a report**

5. Set the table property to sort the data in the table by the data type, then by column name:
  - a. Open the Outline view.
  - b. Expand the *SDT\_Info\_Sys\_custom.rptdesign* -> *Body* node, then select *Table – Column Report*. The *Table – Column Report* element opens in the Property Editor view.
  - c. In the Property Editor, select the *Sorting* tab at the top of the view. Click *Add*. The *New Sort Key* window opens.
  - d. In the *Key* field, select *Full Data Type Information* and click *OK*.
  - e. Click the *Add* button to add another sort key.
  - f. In the *Key* field, select *Name* and click *OK*.

You have created two sort keys for the table. When you generate the report later in this chapter, the information in the table will be sorted by the data type of each column and then by the name of each column.

6. Preview your report. At the bottom of the Report Editor view, click the *Preview* tab. Your report should look like the image in *Figure 6.7*.



**Figure 6.7 – Previewing a report**

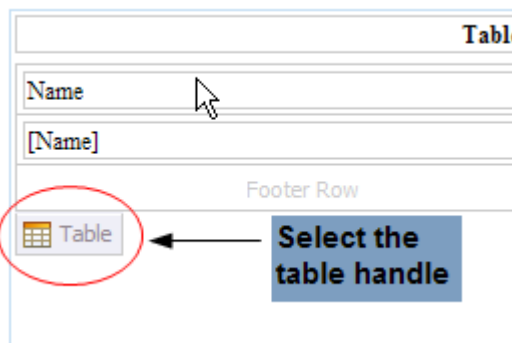
You have created a basic BIRT report. Use the *Preview* tab to check the design of your reports before you run them.

Now, you should further modify this report to specify what tables these columns belong to. You will nest a table within the existing table in order to group and sort the columns to organize them according to the report design that you specify.

### 6.3.4 Grouping data in a report

Let's group the columns by table, maintaining the existing design of sorting the columns by data type and name. To group the data in the report:

1. Click the *Layout* tab at the bottom of the Report Editor.
2. Select the table. When you hover over the table, the *Table* handle appears, as shown in *Figure 6.8*. Select this handle.

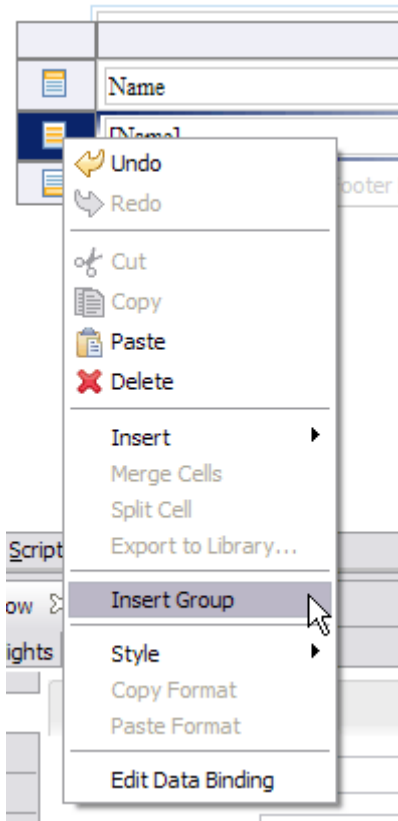


**Figure 6.8 – Click the *Table* handle to select the table**

The guide cells for the table appear to the left and top of the report table.

3. Add a group to sort the detail rows of the table by source table:

- a. Right-click the detail row guide cell and click *Insert Group*. The detail row guide cell is the middle guide cell, as shown in *Figure 6.9*.

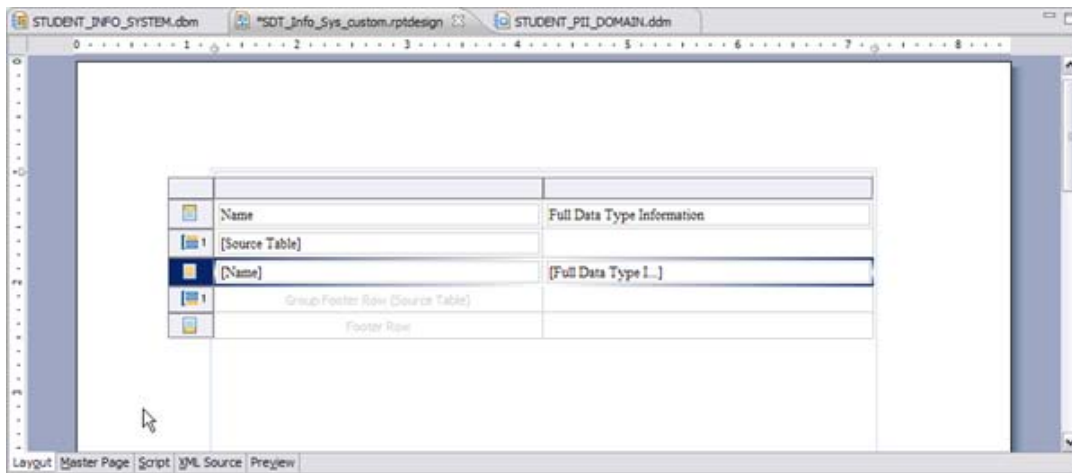


**Figure 6.9 – Insert a group with a detail guide cell**

The New Group window opens.

- b. Specify **GroupByTable** as the name.
- c. Specify *Source Table* as the *Group On* option.
- d. Click *OK* to insert the group.

The table now contains a group to sort the columns by table. The report layout should look like the image in *Figure 6.10*.

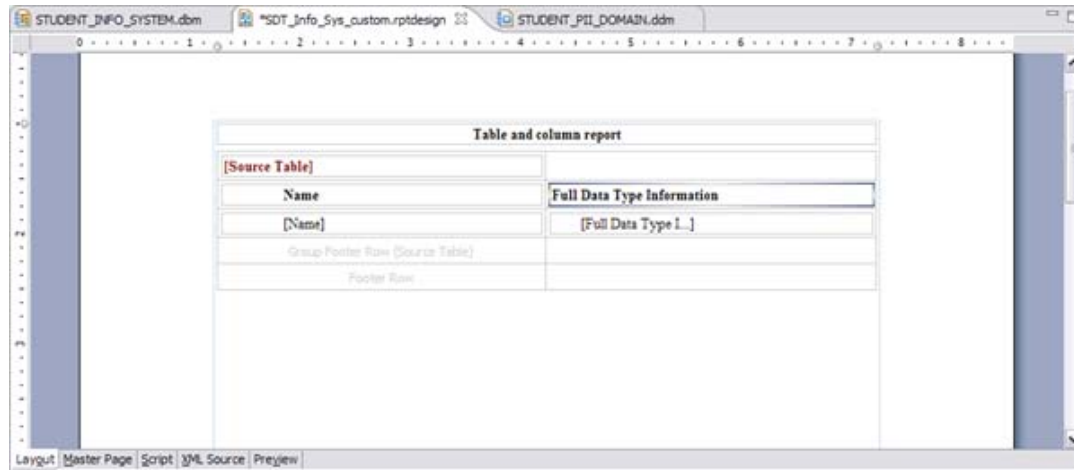


**Figure 6.10 – Adding groups to the report**

4. Format the *[Source Table]* element:
  - a. In the Report Editor, select the *[Source Table]* element. The properties of the element display in the Property Editor view.
  - b. In the *Properties* tab of the Property Editor, make sure the *General* tab is open.
  - c. Change the font color to *Maroon*, and set the font weight to bold.
5. Format the *[Full Data Type Information]* element:
  - a. In the Report Editor, select the *[Full Data Type Information]* element.
  - b. In the *Properties* tab of the Property Editor, open the *Padding* tab.
  - c. In the *Left* field, type 20.
6. Format the *[Name]* element:
  - a. In the Report Editor, select the *[Name]* element.
  - b. In the *Properties* tab of the Property Editor, open the *Padding* tab.
  - c. In the *Left* field, type 40.
7. Create a header row:
  - a. In the Report Editor, right-click the guide cell for the table header row that contains the *[Full Data Type Information]* element.
  - b. Select *Insert -> Row -> Below*. A group header row is inserted into the report table.
  - c. In the Report Editor, move the *Name* and *Full Data Type Information* labels from the first header row to the new row that you created. Drag the *Name* label to the left column, and drag the *Full Data Type Information* label to the right column.
  - d. Delete the newly empty header row. Right-click on the guide cell for the first header row, then click *Delete*.

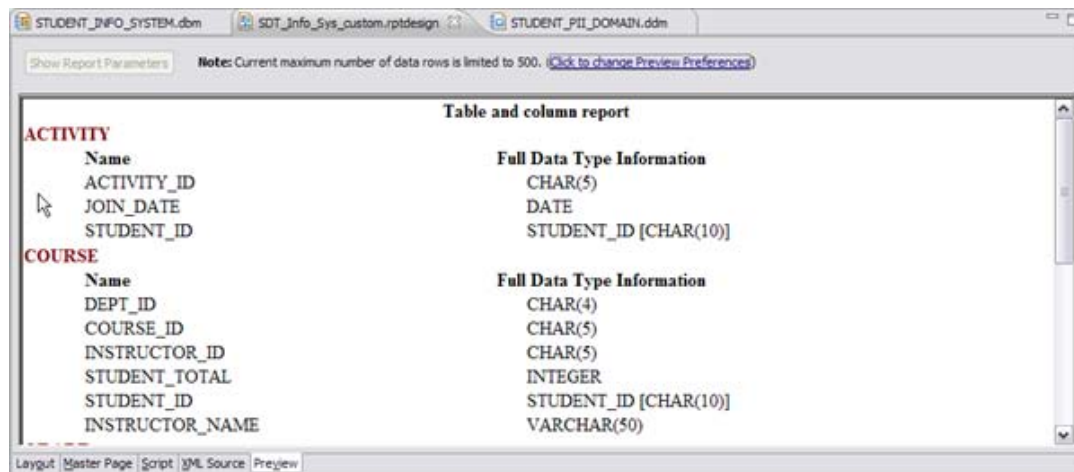
- e. Format the *Name* label. Select the *Name* element. In the *General* tab, set the font weight to *Bold*, and in the *Padding* tab, set the left padding to 40.
- f. Format the *Full Data Type Information* label. Select the *Full Data Type Information* label. In the *General* tab, set the font weight to *Bold*.

In the *Layout* tab of the Report editor, the table looks like the image in *Figure 6.11*.



**Figure 6.11 – Grouping and formatting data in a report**

8. Preview the report. Open the *Preview* tab of the Report Editor. The report should look like the image in *Figure 6.12*.



**Figure 6.12 – Previewing a modified report**

### 6.3.5 Adding dynamic text to a report

Now that you have created a basic report, you should improve the appearance of the report with dynamic text and formatting. A dynamic text label lets you clearly label elements of your data model, and the name of each data object displays dynamically according to text that you specify.

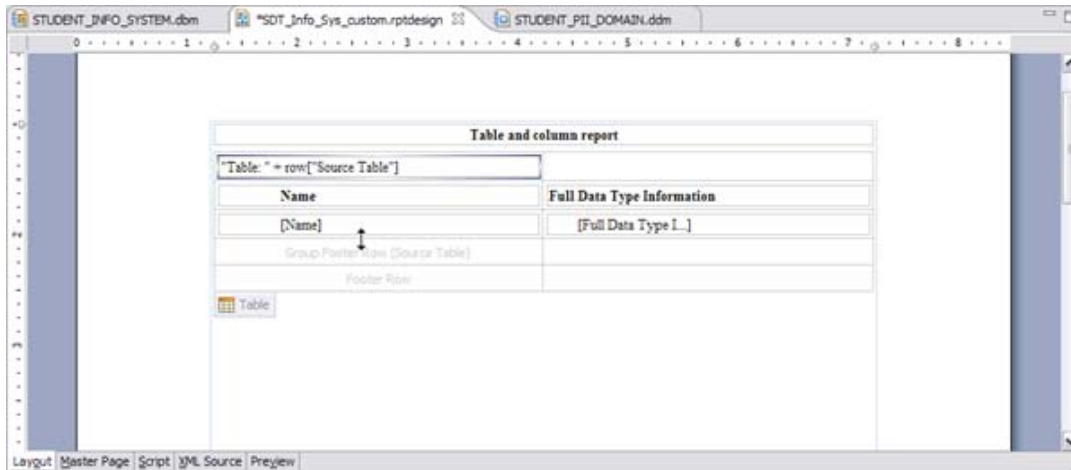
In the report that you have created, you should clearly label tables and columns, so that when you share this report with other members of your team (or business process owners), they will be able to clearly understand the design of your physical data model.

To format the report:

1. Open the *Layout* tab of the Report Editor.
2. Create dynamic text for the table header:
  - a. Delete the *[Source Table]* element from the group header row.
  - b. From the Palette, drag a *Dynamic Text* element into the header cell where you deleted the *[Source Table]* element. The Expression Builder window opens.
  - c. In the Expression Builder window, add the following string to the *Expression* field, then click *OK*:

**"Table: " + row["Source Table"]**

The expression is added to the dynamic text element, as shown in *Figure 6.13*.

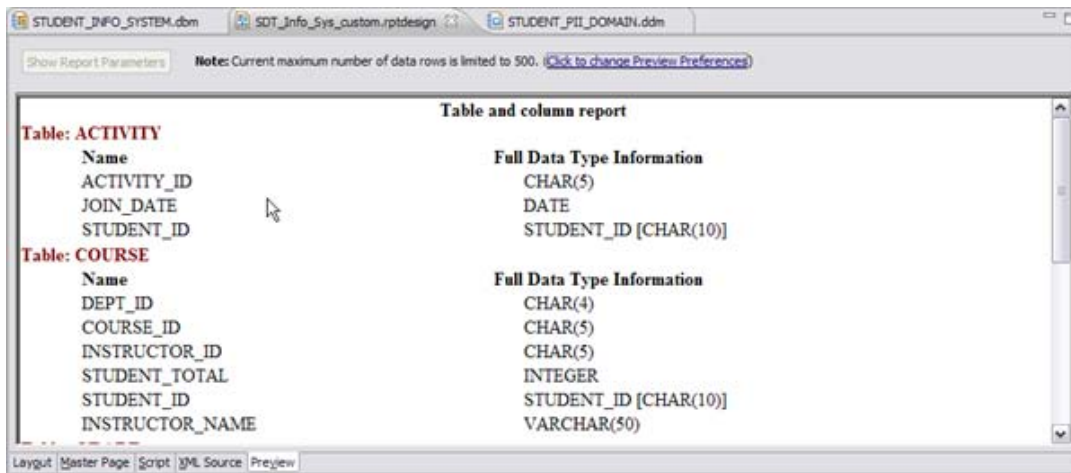


**Figure 6.13 – Adding a dynamic text element to a BIRT report**

3. Format the new dynamic text label:
  - a. Make sure the label is selected in the Report Editor.
  - b. In the Property Editor view, open the *General* tab. Set the font weight to *Bold*, and set the font color to *Maroon*.
4. Preview your report.

The report should look like the image in *Figure 6.14*.





**Figure 6.14 – Previewing the formatted report**

This design helps make it easier for your team to identify the various objects within your data model. Information is conveniently sorted.

### 6.3.6 Generating a report configuration from a template

Now, let's create a report configuration so that you can share your report with your team. You can also use this report configuration repeatedly, so as you make changes to the model, you can create new reports with minimal effort.

To generate a report from a BIRT template:

1. Open the Navigator view.
2. Select the report and create the configuration:
  - a. Right-click the `SDT_Info_Sys_custom.rptdesign` file and select *Report As -> Report Configurations*. The Report Configurations window opens.
  - b. Double-click the *BIRT Report* node to create a new report configuration.
  - c. In the *Name* field, specify `SDT_Info_Sys_custom`.
  - d. Select the *Location* radio button, then click the *Workspace* button to locate your report design in your workspace. The Select Report window opens.
  - e. Expand the *University Info System* node, then select the `SDT_Info_Sys_custom.rptdesign` template that you just created. Click *OK* to add it to the report configuration.
  - f. Specify a location for the report output. For example, specify `C:\Temp\IDA_Reports\Student_Info_Sys.html`.
  - g. Click *Apply* to save the configuration.
3. Click *Report* to run the report configuration and generate a report.

The physical data model report configuration runs. Since you created an HTML report, your default browser opens the physical data model report. The files for the report are also saved to the location that you specified.

This report contains some of the information about your tables and columns. The HTML report is shown in *Figure 6.15*.

Table and column report	
<b>Table: ACTIVITY</b>	
Name	Full Data Type Information
ACTIVITY_ID	CHAR(5)
JOIN_DATE	DATE
STUDENT_ID	STUDENT_ID [CHAR(10)]
<b>Table: COURSE</b>	
Name	Full Data Type Information
DEPT_ID	CHAR(4)
COURSE_ID	CHAR(5)
INSTRUCTOR_ID	CHAR(5)
STUDENT_TOTAL	INTEGER
STUDENT_ID	STUDENT_ID [CHAR(10)]
INSTRUCTOR_NAME	VARCHAR(50)
<b>Table: GRADE</b>	
Name	Full Data Type Information
SCORE_PASSFAIL	CHAR(1)
COURSE_ID	CHAR(5)
SCORE_TOTAL	DECIMAL(3, 0)
STUDENT_ID	STUDENT_ID [CHAR(10)]
<b>Table: STUDENT</b>	
Name	Full Data Type Information
GRADUATION_DATE	DATE
START_DATE	DATE
EMAIL_ADDR	EMAIL [VARCHAR(128)]
LAST_NAME	LAST_NAME [VARCHAR(128)]
STUDENT_SSN	SSN [CHAR(11)]

**Figure 6.15 – The report displays in your default Web browser**

Note that the physical data model report does not actually display the data within the database. This type of report is useful to analyze the structure of your model with other members of your team.

#### Note:

The report configuration is saved to your workspace, as well. If the design of your model changes, simply run the report configuration by clicking *Run -> Report -> Report Configurations*, and select the report configuration that you want to run.

## 6.4 Generating XSLT reports

Let's produce one more report that contains all of the information about the logical data model that you have already created. This report will gather all of the information about the logical data model so that you can present detailed information to your team.

To create an XSLT report:

1. Switch to the Data perspective by clicking *Window -> Open Perspective -> Other*. The Open Perspective window opens. Select the *Data* perspective.
2. From the main menu, click *Run -> Report -> Report Configurations*. The Report Configurations window opens.

3. Double-click the *Data Model Classic XSLT Report* option to create a new XSLT report configuration.
4. Specify the following options for the new configuration:
  - a. In the *Name* field, specify **SDT\_Info\_Sys\_XSLT**.
  - b. In the report input *Location* field, click *Browse* and select the *STUDENT\_INFO\_SYSTEM.Idm* logical data model.
  - c. In the report input *Element* field, click *Browse* and select the *STUDENT INFO SYSTEM* package.
  - d. Select the *Built-in* radio button in the *Report* section, then click *Browse*. The *Select Built-In Report* window opens.
  - e. Expand the *Logical Data Model (Classic XSLT Reports)* node, then select *Classic Sample Logical Model Report*. Click *OK* to add the report type to the configuration.
  - f. In the report output *Location* field, specify a file path and name. For example, specify **C:\Temp\IDA\_Reports\SDT\_Info\_Sys\_XSLT.pdf**.
5. Click *Apply* to save the configuration.
6. Click *Report* to run the configuration.

The report runs. After the report is generated, it opens. The report contains all of the information about the logical data model and describes all of the properties of your data objects. The diagram that you have created with IBM InfoSphere Data Architect is also included in the report, making it easy for non-technical members of your team to understand the relationships between the data objects.

## 6.5 Importing and exporting with IBM InfoSphere Data Architect

The import and export capabilities of IBM InfoSphere Data Architect allow you to import and export data models from IBM InfoSphere Data Architect to and from other modeling tools, such as IBM Cognos Framework Manager, IBM Rational Rose, or IBM Rational Software Architect.

When you import or export data models in and out of IBM InfoSphere Data Architect, the metadata from the data models is converted to a format that can be understood by the other modeling tools.

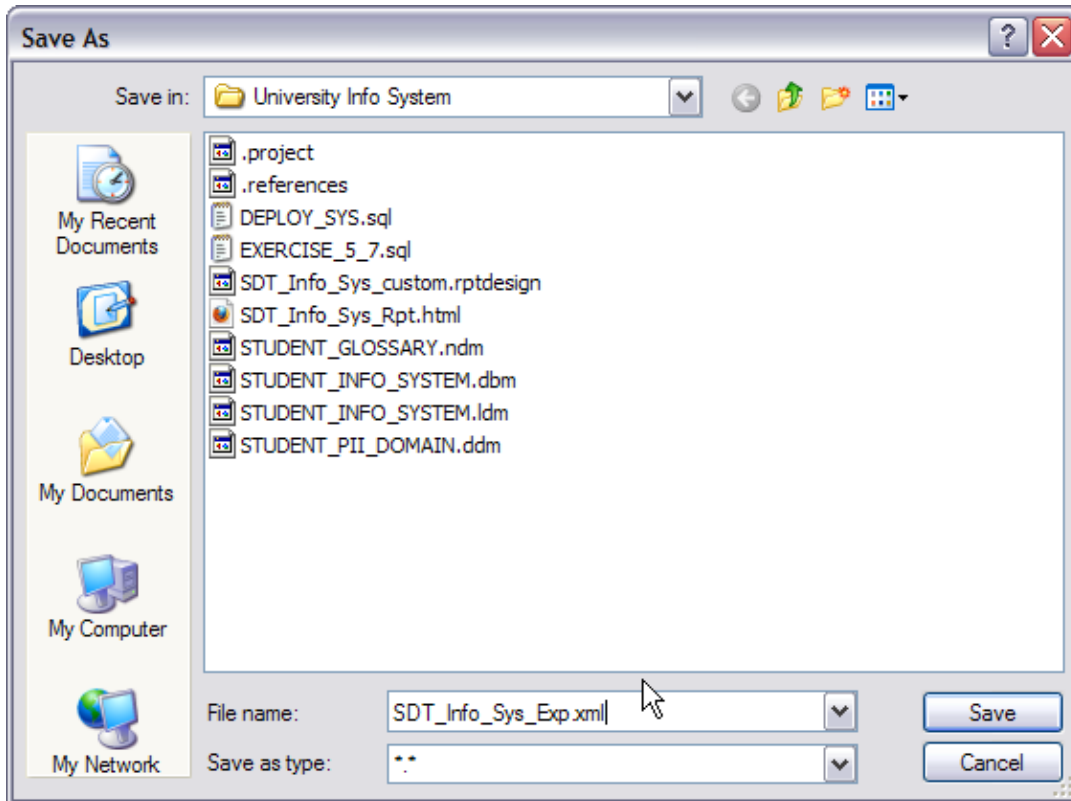
### 6.5.1 Exporting with the workbench

Let's use IBM InfoSphere Data Architect to export a logical data model to an XML file, so that when the data model is populated with information, business process owners can import the XML model file into tools that can use it to generate reports.

To export the data model:

1. From the main menu, click *File -> Export*. The *Export* wizard opens.
2. Expand the *Data* node and select *Data Model Export Wizard*, then click *Next*.
3. Complete the *Select the Model* page of the wizard:
  - a. In the *Model format* field, specify *W3C XML Schema 1.0 (XSD)*.

- b. Click *Browse* next to the *Model* field, and select the *STUDENT\_INFO\_SYSTEM.Idm* logical data model file.
- c. Click *Browse* next to the *Export to* field, and then specify that the exported data model should be saved to your data design project as *SDT\_Info\_Sys\_Exp.xml*. An example of this is shown in *Figure 6.16*.



**Figure 6.16 – Exporting the data model and saving it to your project**

- d. Click *Next*.
4. On the Select the Options page, click *Next*. The data model is exported.
5. Click *Finish* to save the exported data model and exit the Data Model Export wizard.

The exported model can be found in your data design project, under the *XML Schemas* folder.

**Note:**

Since you are exporting to an XML file, the diagram included with this model is not exported.

### 6.5.2 Importing a data model with the workbench

Although the exported model is already in your workbench, let's practice importing models. IBM InfoSphere Data Architect can read information from a number of other data modeling tools, making it easy for you to further refine and improve your data models with the workbench.

Let's import the data model into the workbench:

1. From the main menu, click *File -> Import*. The Import wizard opens.
2. Expand the *Data* node and select *Data Model Import Wizard*, then click *Next*.
3. Complete the Select the Model page:
  - a. In the *Model Format* field, select *W3C XML Schema 1.0 (XSD)*.
  - b. Click the *Browse* button next to the *Model* field, then select the *SDT\_Info\_Sys\_Exp.xml* file.
  - c. Click the *Browse* button next to the *Target project* field, then select the *University Info System* data design project.
  - d. In the *Model type* field, select *Logical only*. This will generate a logical data model file, which you can revise and then transform into a physical data model.
  - e. In the *File name* field, specify *SDT\_Info\_Sys\_Imp*.
  - f. Click *Next*.
4. On the Select the Options page, click *Next*.

The data model is transformed and imported into the data design project.

5. Click *Finish* to exit the Data Model Import wizard.

The model is located in your *Data Models* folder. When you expand the logical data model, you can see that your data model is complete.

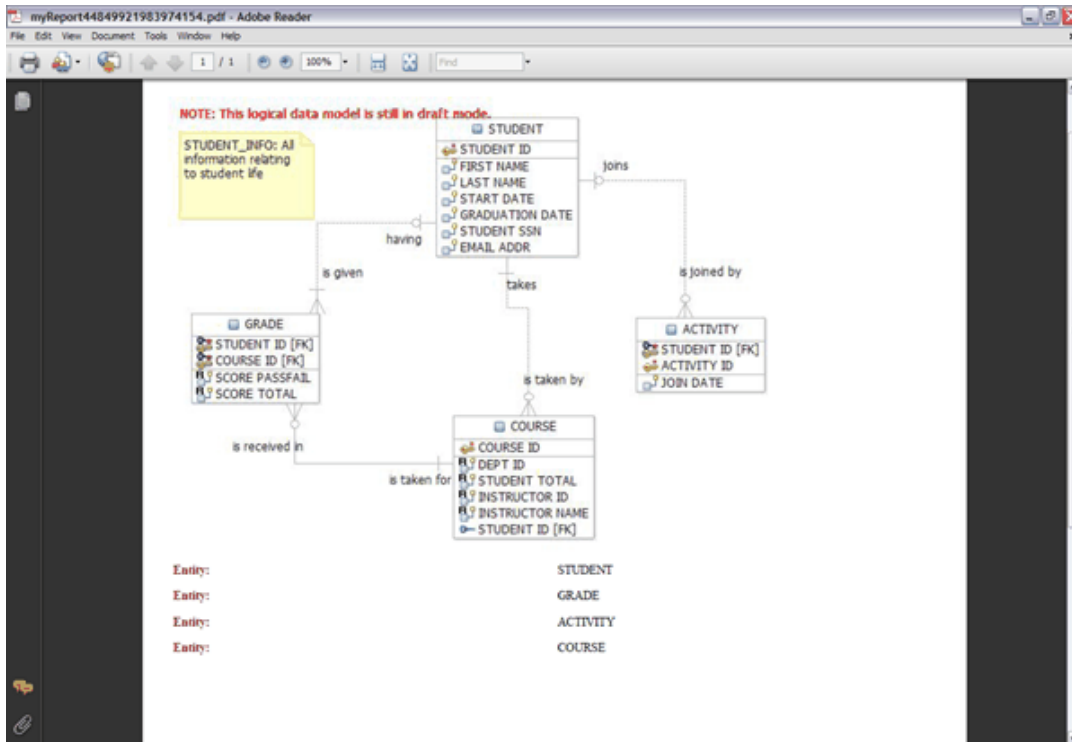
## 6.6 Exercise

Create a customize template, *SDT\_Info\_Sys\_Diag.rptdesign*, which will capture the diagram in the *STUDENT\_INFO\_SYSTEM.ldm* logical data model, including information about the entities in the diagram. Copy and modify a *Blank Diagram Report* to draft this report.

1. Insert a table with 1 column and 2 detail rows, and specify a diagram as the data set.
2. In the first detail row, insert an image of the diagram, and specify that you want to insert a *dynamic image*. Click the *Select Image Data* button to specify the data set binding for the row.
3. Click the *Add* button to add a new data binding that will let you add an image to the report. Specify the best data type to use for this row, in this case, *Blob*, then click the button next to the *Expression* field to open the Expression Builder window.
4. Use the Expression Builder window to specify the following expression for the data set row:
 

```
dataSetRow["Image"]
```
5. Once you've added the expression, select the check box next to the Column Binding data set, then click *OK*, then *Insert* the image in the table.
6. Nest (insert) a table in the second detail row, with 1 column and 2 detail rows and no data set.
7. In the new table, insert a grid with 2 rows and 1 column in the first detail row.

8. Insert a label, **Entity**, in the left column of the grid.
9. Use the Data Explorer to insert a *Entity* -> *Name* object in the right column of the grid.
10. Switch back to the Data perspective, then create a report configuration for the report template. Make sure that the data source of the template points to the **STUDENT\_INFO\_SYSTEM.ldm** logical data model file, so that the diagram within the logical data model can be pulled into the report. Specify that you want a PDF output, then run the report. The report should look like the image in *Figure 6.17*.



**Figure 6.17 – Completed diagram report**

## 6.7 Summary

In this chapter, you learned the fundamental concepts of creating reports for data models in IBM InfoSphere Data Architect. You learned how to generate different types of reports using various built-in templates available in the workbench.

You also learned how to import and export data models within IBM InfoSphere Data Architect. This feature helps you to convert different data models, so that they can be imported from or used with other tools.

## 6.8 Review questions

1. Which BIRT report output format is not supported in the workbench?
  - A. HTML

- B. PDF
  - C. Plain text
  - D. Microsoft Excel
2. Which of the following data models you can import or export in the workbench?
- A. Physical data model
  - B. Glossary model
  - C. Mapping model
  - D. All of the above
3. Which of the following validity check is not supported when you export data models?
- A. Basic validity check
  - B. Normal validity check
  - C. No validity check
  - D. Both A and C
4. What is the file extension for a BIRT report template?
- A. .xsl**
  - B. .xsldesign**
  - C. .rptdesign**
  - D. None of the above
5. Which report output format is supported by XSLT reports?
- A. PDF
  - B. Microsoft Excel
  - C. Plain text
  - D. HTML





# 7

## Chapter 7 – Reverse-Engineering

Often, companies have large schemas with multiple tables that are deployed to a database before you start your work as a data architect. If you need to make changes to these schemas, it takes a large amount of time and effort to manage these changes without affecting existing constraints or losing data.

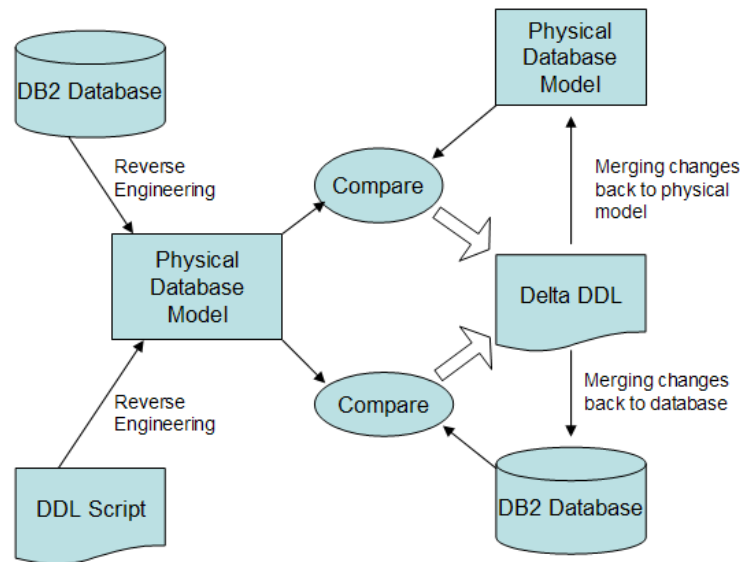
However, you can create a physical data model from the existing set of database objects. After you reverse-engineer an existing database, making changes and deploying the set of changes in the model back to the database becomes an easy process. This is what we will be exploring in this chapter.

In this chapter you will learn about:

- Reverse-engineering from an existing database in the workbench
- Comparing and merging changes to your data models
- Generating the DDL to deploy your changes

## 7.1 Reverse-engineering: The big picture

We have seen how objects in the physical data model can be converted into objects in a database when you generate a DDL script that deploys them. *Reverse-engineering* is the opposite of this process: We convert an object in the database into an object in the physical data model. *Figure 7.1* illustrates the big picture of reverse-engineering.



**Figure 7.1 – An overview of reverse-engineering**

The left side of *Figure 7.1* illustrates that the process of reverse-engineering can be carried out from either a database or a DDL script. The reverse-engineering process creates a physical data model (.dbm) file in the Data Project Explorer of the workbench. This physical data model captures the information about the data objects in the database. Once you reverse-engineer to a physical data model, you can modify the data model, transform to a logical data model, or work with it as you would any other data model.

Once you make changes to the physical data model, you should deploy the changes to the database. Use the compare function in the workbench to match the physical data model against the database and show the differences between the model in development and the model in production. Then, you can analyze the impact that your changes will have to an existing data model. Finally, use the workbench to merge the changes and generate a delta DDL file that can deploy the changes to the database. When you run this script on a database, you make the necessary modifications to the database object without losing any existing data.

### Note:

Although you can reverse-engineer from a DDL script or an existing database, you have more options when you reverse-engineer from a database. Since the database stores a lot of metadata in catalog tables, IBM InfoSphere Data Architect can make good use of that metadata to better populate the physical data model.

## 7.2 Reverse-engineering with the workbench

When you reverse-engineer a database, the workbench reads the details of the objects in the database and converts them to their equivalent representation in the physical data model. This simply means that if you reverse-engineer a schema that contains one table and one index, then the reverse-engineered physical data model will contain one schema with one table and one index.

To understand what happens when you reverse-engineer from a DDL script, consider the following example: A DDL script creates a schema that contains a table and an index in the database. When you reverse-engineer from the DDL script, the workbench creates an equivalent representation of the schema, the table, and the index in the physical data model. We will see the process in more detail in the next section.

### 7.2.1 Reverse-engineering from DDL

#### 7.2.1.1 Creating a sample DDL script

First, let's create a sample DDL script from which to reverse-engineer. This DDL script creates the following database objects in the SAMPLE database connection:

- A database, named *SAMPLEUNIV*
- A schema, named *UNIVERSITY*, that belongs to the *SAMPLEUNIV* database
- Two tables in the schema, named *DEPARTMENT* and *STUDENT*

To create the sample DDL script:

1. Right-click on the *SQL Scripts* folder of your data design project, then select *New -> SQL or XQuery Script*. The New SQL or XQuery Script window opens.
2. Complete the New SQL or XQuery Script window:
  - a. Specify the *University Info System* project.
  - b. In the *Name* field, type `SAMPLE_DDL.sql`.
  - c. Make sure that the following radio button is selected: *SQL and XQuery Editor (for scripts that contain one or more SQL and XQuery statements)*
  - d. Click *Finish*.

The Select Connection Profile window opens.

3. Select the SAMPLE database connection profile. Since you aren't deploying the script, you do not need to worry about a connection at this point.

A blank SQL file is created and opens in the SQL Editor view.

4. Enter the following SQL statements to create the SAMPLEUNIV database with a UNIVERSITY schema and DEPARTMENT and STUDENT tables:

```
CREATE DATABASE SAMPLeUNIV;
CREATE SCHEMA "SAMPLeUNIV"."UNIVERSITY";
CREATE TABLE "UNIVERSITY"."STUDENT"
  (STUDENT_ID CHAR(10) NOT NULL,
```

```
STUDENT_FIRSTNAME CHAR(20) NOT NULL,
STUDENT_LASTNAME CHAR(20) NOT NULL,
STUDENT_ADDRESS CHAR(100) NOT NULL,
STUDENT_PHONE CHAR(10) NOT NULL,
DEPARTMENT_ID INTEGER NOT NULL);
CREATE TABLE "UNIVERSITY"."DEPARTMENT"
(
    DEPARTMENT_ID INTEGER NOT NULL,
    DEPARTMENT_NAME CHAR(20) NOT NULL,
    DEPARTMENT_ESTDATE DATE);
ALTER TABLE "UNIVERSITY"."STUDENT" ADD CONSTRAINT
"STUDENT_PK" PRIMARY KEY(STUDENT_ID);
ALTER TABLE "UNIVERSITY"."DEPARTMENT" ADD CONSTRAINT "DEPARTMENT_PK" PRIMARY
KEY(DEPARTMENT_ID);
ALTER TABLE "UNIVERSITY"."STUDENT" ADD CONSTRAINT "STUDENT_DEPARTMENT_FK"
FOREIGN KEY(DEPARTMENT_ID) REFERENCES
"UNIVERSITY"."DEPARTMENT" (DEPARTMENT_ID);
```

5. Save your work.

The SQL script is created and saved in your *SQL Scripts* folder.

**Note:**

You do not have to run this file to reverse-engineer from it. You will reverse-engineer from this script in the next section.

### 7.2.1.2 Reverse-engineering from the DDL script

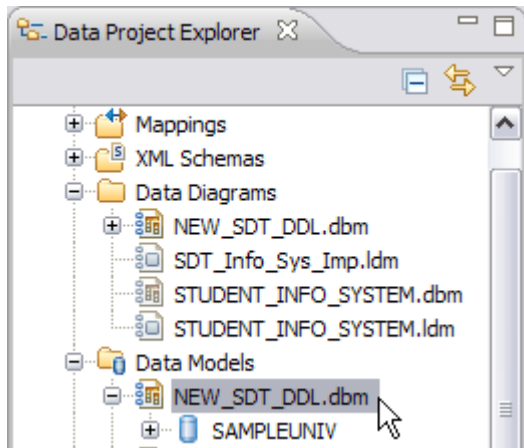
Let's use this new DDL file to reverse-engineer to a physical data model.

To reverse-engineer the DDL script:

1. Create a new physical data model by clicking *File -> New -> Physical Data Model*. The New Physical Data Model wizard opens.
2. Complete the Model File page of the wizard:
  - a. Specify the *University Info System* data design project as the destination folder.
  - b. In the *File name* field, specify **NEW\_SDT\_DDL**.
  - c. Specify *DB2 for Linux, UNIX, and Windows* as the database type.
  - d. Specify *9.7* as the database version.
  - e. Select the *Create from reverse engineering* radio button.
  - f. Click *Next*.
3. On the Source page, select the *DDL script* option, then click *Next*.
4. On the Script File page, click *Browse*, then select the **SAMPLE\_DDL.sql** file. Click *Next*.
5. On the Options page, click *Next*.

6. On the Reverse Engineering Status page, verify that the script file was successfully reverse-engineered, then click *Finish* to create the physical data model.

The **NEW\_SDT\_DDL.dbm** physical data model is created in the Data Project Explorer, as shown in *Figure 7.2*.



**Figure 7.2 – New physical data model, created from a DDL file**

If you explore the physical data model, you see the schema and tables that were specified in the DDL script.

### 7.2.2 Reverse-engineering from a database

The process of reverse-engineering from a database is pretty similar to that of reverse-engineering from a DDL script, except that you can choose what objects to reverse-engineer from the database. For example, a database might contain multiple schemas, and you want to keep your schema models separate from one another. Simply select the schema or schemas that you want to reverse-engineer, and modify them before you deploy the changes.

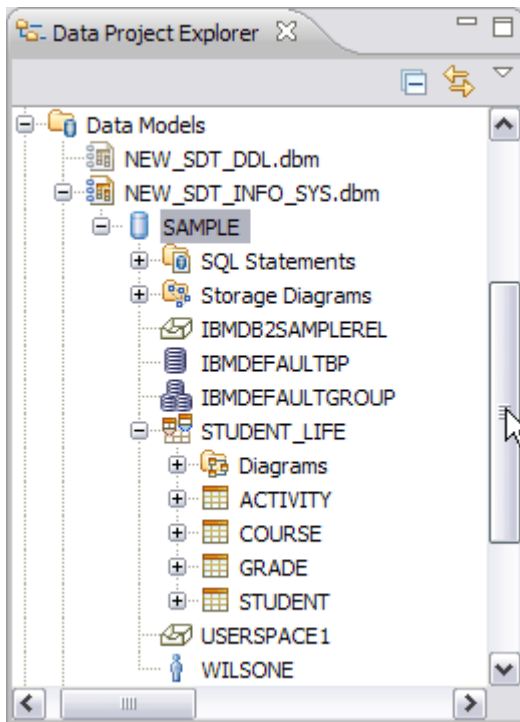
Although you already have an existing copy of the **STUDENT\_LIFE** schema, contained in the **STUDENT\_INFO\_SYSTEM.dbm** physical data model file, let's use the schema deployed on the database to reverse-engineer and make changes, so that we can later compare and deploy those changes to the **SAMPLE** database.

To reverse-engineer from a database:

1. From the main menu, click *File -> New -> Physical Data Model*. The New Physical Data Model wizard opens.
2. Complete the Model File page:
  - a. Specify the *University Info System* data design project as the destination folder.
  - b. In the *File name* field, specify **NEW\_SDT\_INFO\_SYS**.
  - c. Specify *DB2 for Linux, UNIX, and Windows* as the database type.
  - d. Specify *V9.7* as the database version.

- e. Select the *Create from reverse engineering* radio button, then click *Next*.
3. On the Source page, make sure the *Database* radio button is selected, then click *Next*.
4. On the Select Connection page, select the SAMPLE database connection profile, then click *Next*.
5. On the Select Objects page, select the STUDENT\_LIFE schema, then click *Next*.
6. On the Database Elements page, keep the default selections, then click *Next*.
7. On the Options page, click *Finish*.

The new physical data model is created in the Data Project Explorer (as shown in *Figure 7.3*) and opens in the editor view.



**Figure 7.3** – A new physical data model, reverse-engineered from an existing database

### 7.3 Making changes to the new physical data model

Let's modify the NEW\_SDT\_INFO\_SYS.dbm physical data model file to include some new columns and an index. In this scenario, the business process owners have asked you to include columns to store the main phone number of each instructor of each course and the phone numbers of students. They've also requested an index to help sort student names to make accessing the data easier.

After you make these changes, you can use the compare and merge features of the workbench to generate DDL to deploy the changes to the SAMPLE database.

To make changes to the physical data model:

1. Locate the COURSE table in the data design project: *University Info System -> Data Models -> NEW\_SDT\_INFO\_SYS.dbm -> SAMPLE -> STUDENT\_LIFE -> COURSE.*
2. Create the INSTRUCTOR\_PHONE column in the COURSE table:
  - a. Right-click on the COURSE table and select *Add Data Object -> Column*. A new column object is created under the COURSE table.
  - b. Name the new column **INSTRUCTOR\_PHONE**.
  - c. In the *Data Type* field for the INSTRUCTOR\_PHONE column, specify CHAR.
  - d. In the *Length* field, specify 10.
3. Use the steps outlined in step 2 to create the STUDENT\_PHONE column in the STUDENT table. Use the same data type.
4. Create a new index to sort student information by last name:
  - a. Right-click the STUDENT table and select *Add Data Object -> Index*. A new index is created under the STUDENT table.
  - b. Name the index **STUDENT\_LAST\_NAME\_IDX**.
  - c. Open the *Columns* tab of the Properties view, and click the ellipsis button (...) next to the *Key columns* field to modify the key columns. The Select Columns window opens.
  - d. Deselect the FIRST\_NAME column, and select the LAST\_NAME column. Click *OK* to modify the key column.
5. Save your work.

The columns and index are created and saved in your physical data model file. Now you can use this modified physical data model to compare with the target database, then generate delta DDL to merge your changes in the SAMPLE database.

## 7.4 Compare and merge your changes

The *compare* function helps you find differences between two versions of a model. The *merge* function helps you deploy the changes into an existing model. With this view, you can also *analyze the impact* that your changes will have upon existing models. This action helps you find dependencies among your data objects, so that you can understand how the changes you make to one data object will impact related objects.

### Note:

Since the changes we have made to this small model are minor, there are no dependent objects in this model that would be impacted. As a best practice, you should always analyze the impact of your changes before you generate the DDL to deploy your changes.

If, in the future, you would like to analyze the impact of your changes, you must first copy the changes from one data model to another. Then, you can use the **Analyze Left Impact** or **Analyze Right Impact** buttons to analyze the impact of your changes. The impacted objects are shown in the Impacted Objects view, and you can drill down through the changes to see what data objects will be impacted by your

changes.

To learn how to compare and merge with the workbench, let's use the `NEW_SDT_INFO_SYS.dbm` physical data model and compare it to the `SAMPLE` database. Since you have already modified the physical data model, you can now compare it to the database and generate a DDL script to deploy your changes.

**Note:**

You can also compare two models within a workspace. This task is not covered in this book, but is similar to the process of comparing to a source database.

### 7.4.1 Comparing and merging changes with the database

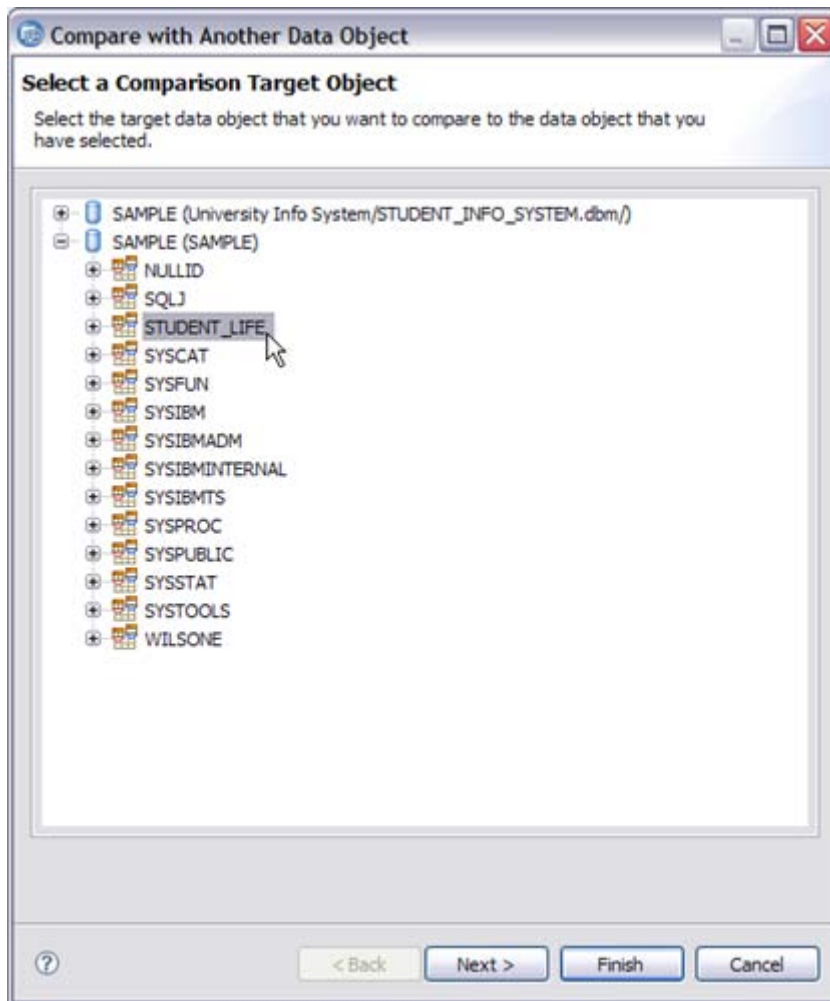
You can compare a data model with the source database. The compare feature is available for the following objects:

- ▲ The source database
- ▲ Logical data models
- ▲ Physical data models
- ▲ Glossary models
- ▲ Domain models

Let's compare the new physical data model with its source database:

1. Locate the `STUDENT_LIFE` schema in the Data Project Explorer: *University Info System -> Data Models -> NEW\_SDT\_INFO\_SYS.dbm -> SAMPLE -> STUDENT\_LIFE*.
2. Right-click the `STUDENT_LIFE` schema in the Data Project Explorer and select *Compare With -> Another Data Object*. The Compare with Another Data Object wizard opens.
3. Expand the `SAMPLE (SAMPLE)` database, then select the `STUDENT_LIFE` schema of the source database, as shown in *Figure 7.4*:





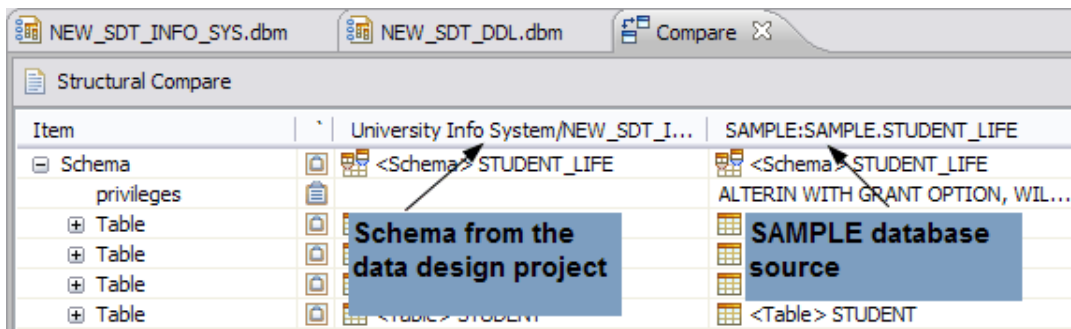
**Figure 7.4 – Selecting the source database**

Click *Next* to proceed to the next screen. The Filtering Criteria page opens.



4. Select specific data objects to compare:
  - a. Click the *Deselect All* button to deselect all of the data objects.
  - b. In the *Objects to include in comparison* field, select the *Columns* object. All of the objects that are related to columns are also selected.
  - c. Select the *Index* object.
5. Click *Finish* to run the comparison.

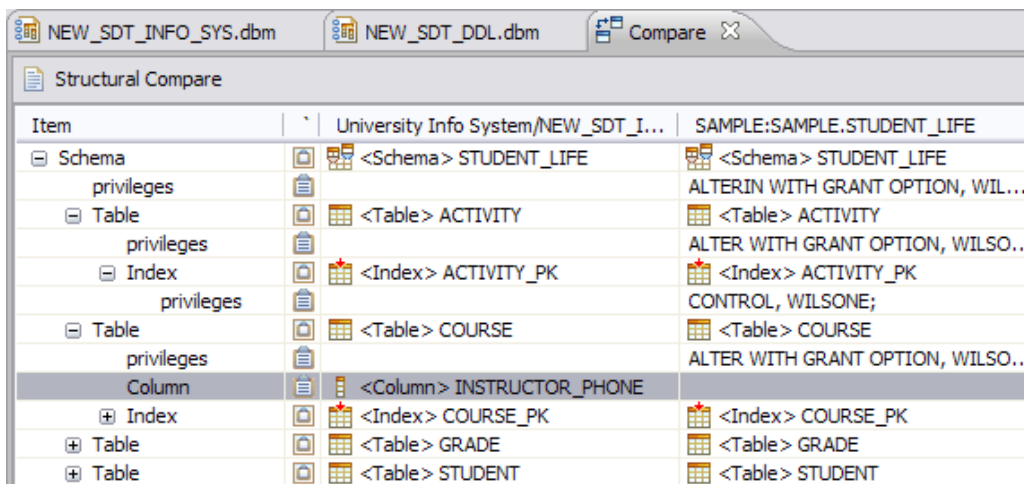
The schema object of the physical data model is compared to the STUDENT\_LIFE schema in the SAMPLE database. The Compare editor opens.

The Structural Compare pane lets you navigate through the differences in the data design project schema and source schema. The data design project schema is found in the left side of the pane, and the source schema is found in the right side of the pane. This is shown in *Figure 7.5*.




**Figure 7.5 – Comparing the schemas**


4. Use the Structural Compare pane to navigate through the differences between the two schemas. Click the *Next Difference*  or *Previous Difference*  buttons to see the differences between the schemas.
5. Use the Structural Compare view to locate the `COURSE.INSTRUCTOR_PHONE` column, as shown in Figure 7.6. Note that the Structural Compare view shows that the object does not exist in the source schema.



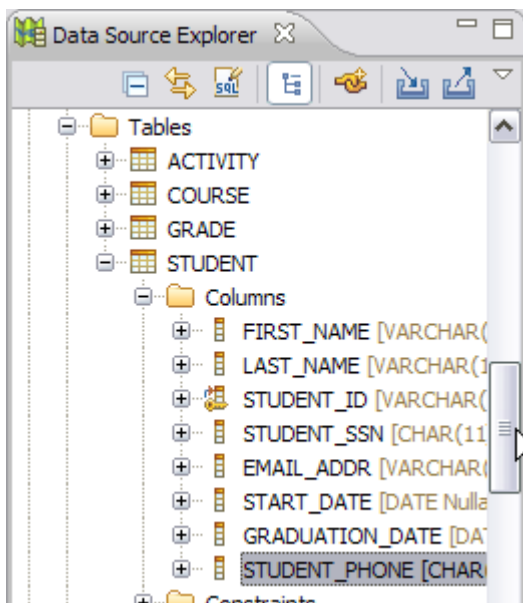
**Figure 7.6 – Locating the new column**

The Property Compare pane lets you compare the differences between each data object. The left side of the pane contains the information about the object in the data model. The right side of the pane contains the information about the object in the source database, if it exists.

6. Right-click the `INSTRUCTOR_PHONE` column and select *Analyze Left Impact*. The Impacted Objects view opens. Since no other data objects reference this column, and since there are no dependencies, there is no impact displayed in the Impacted Objects view.
7. Use the Property Compare view to specify that you want to copy the new column to the source schema. Click the *Copy from Left to Right*  button.
8. Use the Structural Compare view to locate the `STUDENT_PHONE` column.

9. Right-click the STUDENT\_PHONE column and select *Analyze Left Impact*. Since no other data objects reference this column, and since there are no other dependencies, there is no impact displayed in the Impacted Objects view.
10. Use the Property Compare view to specify that you want to copy the STUDENT\_PHONE column to the source schema.
11. Locate the new STUDENT\_LAST\_NAME\_IDX index, then analyze the impact and copy the change to the source schema.
12. Generate the DDL to deploy your changes. The *Generate Right DDL*  button generates the DDL that you need to deploy the changes to the database. The Generate DDL wizard opens.
11. Complete the Save and Run DDL page:
  - a. Make sure that the *University Info System* data design project is specified in the *Folder* field.
  - b. In the *File name* field, specify `DELTA_SDT_CHANGES.sql`.
  - c. Select the *Run DDL on server* check box to run the DDL script on the server and deploy your changes.
  - d. Click *Next*.
12. On the Select Connection page, select the SAMPLE database connection, then click *Next*.
13. Review the information on the Summary page, then click *Finish*.

The DDL runs on the server, and your new columns and index are created in the SAMPLE database. Check the SQL Results view to ensure that the SQL statements ran successfully. Refresh the database in the Data Source Explorer, then verify that the new columns and index exist. One of the new columns is displayed in *Figure 7.7*.



**Figure 7.7 – Viewing the changes in the Data Source Explorer**

### 7.4.2 Advantages of the compare and merge functions

Without the help of IBM InfoSphere Data Architect, comparing and merging updated data models is a time-consuming process, since you would have to manually design all of the queries. Manually introducing changes can cause many errors, especially in databases with large number of existing schemas, tables, and constraints.

IBM InfoSphere Data Architect accounts for existing constraints and dependencies when it carries out the compare and merge process, making it easy for you to manage changes to your databases and schemas.

## 7.5 Exercise

1. Create the following DDL script, named **EXERCISE\_7\_5.sql**:

```
CREATE SCHEMA "EXERCISE" ;
CREATE TABLE "EXERCISE"."TABLE1"
(
    "TABLE1_ID" INTEGER,
    "TABLE1_NAME" VARCHAR(20),
    "TABLE2_ID" INTEGER);
CREATE TABLE "EXERCISE"."TABLE2"
(
    "TABLE2_ID" INTEGER,
    "TABLE2_NAME" VARCHAR(20),
    "TABLE3_ID" INTEGER);
CREATE TABLE "EXERCISE"."TABLE3" (
    "TABLE3_ID" INTEGER,
    "TABLE3_NAME" VARCHAR(20));
CREATE UNIQUE INDEX "TABLE1_ID_INDEX" ON "EXERCISE"."TABLE1"
("TABLE1_ID");
CREATE UNIQUE INDEX "TABLE2_ID_INDEX" ON "EXERCISE"."TABLE2"
("TABLE2_ID");
CREATE UNIQUE INDEX "TABLE3_ID_INDEX" ON "EXERCISE"."TABLE3"
("TABLE3_ID");
```

2. Reverse-engineer the DDL script and create a new physical data model, **EXERCISE\_7\_5.dbm**. On the Options page of the New Physical Data Model wizard, make sure that you create an *overview* diagram and *infer implicit relationships* between the data objects of the data model. Document your results.
3. Use the Properties view of the EXERCISE diagram to display the key and non-key compartments of the objects in the diagram. Use the *Layout* tab to change the layout to a *Hierarchical -> Top to bottom* layout. The overview diagram of the new physical data model should look like the image in *Figure 7.8*.

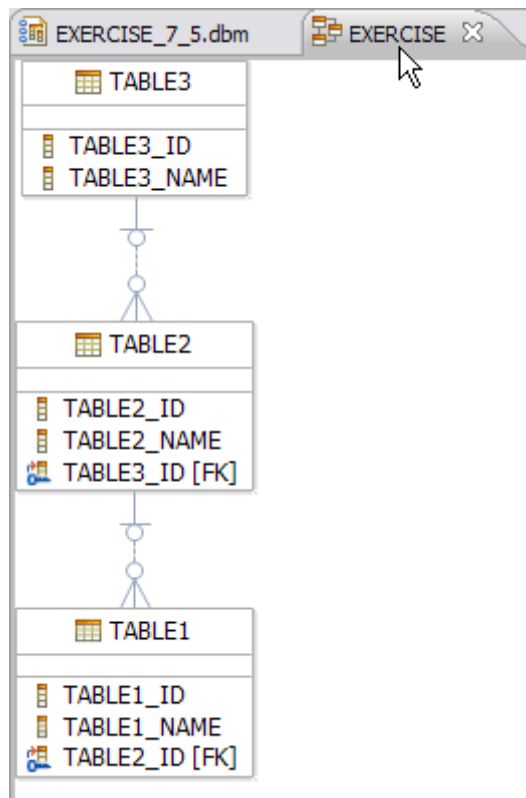


Figure 7.8 – Viewing the EXERCISE diagram

## 7.6 Summary

We have seen how the process of reverse-engineering helps us in creating a physical data model from an existing set of tables and schemas in our database or from a DDL script. We have also seen how we can choose the type of database elements which we want to incorporate into the reverse engineered model when reverse-engineering from a database. We have seen how the compare and merge process can help us manage the changes to an existing database without losing any existing data.

## 7.7 Review questions

1. True or false: To reverse-engineer a DDL script, it can only have the .DDL file extension.
2. True or false: When reverse-engineering from a DDL script, the workbench allows you to specify the type of database elements that you want to reverse-engineer.
3. What are the different types of database elements that can be reverse-engineered into a physical data model?
4. True or false: You can only compare and merge changes on a physical data model and its ancestor in the database.
5. What types of data models can you compare?



# 8

## Chapter 8 – Model Mapping and Discovery

Information can come from different data sources, and integration of such data is a common requirement in any enterprise. When an existing data source has to be integrated with an acquisition data source, understanding the metadata is very crucial.

*Metadata* is data about the data; it stores the data type of the data, purpose of the data, and other identifying information. Understanding metadata helps to visualize how the different data sources are related to each other and creating mappings and relationships amongst them. *Mapping models* help you map this metadata among a variety of data sources.

IBM InfoSphere Data Architect provides the following metadata integration tools:

- Define relationships and mappings between data sources
- Set transformation rules
- Discover relationships
- Generate scripts which could be deployed to a database

### 8.1 Mapping models: The big picture

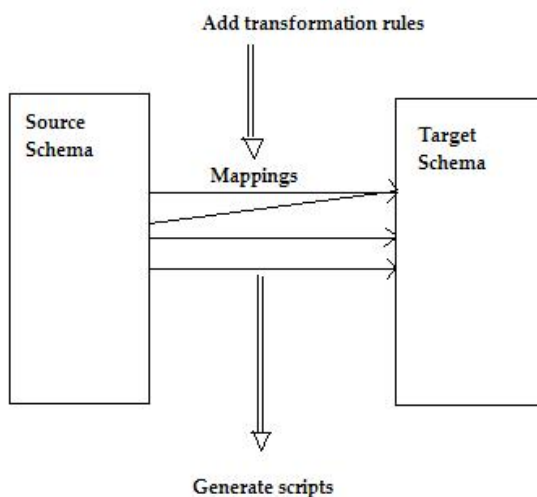


Figure 8.1 – The big picture

A *mapping model* defines relationships between two different schemas: a source schema and a target schema. The mapping model is stored as a mapping specification language (MSL) file. The source model can be either a logical data model or physical data model. The target model can be a `logical data model`, `physical data model`, or an XML schema file (XSD).

Mapping models are a part of information integration modeling. *Information integration modeling* provides a set of functions that can help you manage metadata. To integrate a new data source with existing enterprise data, you need to understand how the data in the new source is related to data in the existing environment. You can build a process that copies data from the new data source into the existing source. By defining mappings and relationships, you can manipulate a large set of disparate metadata.

### 8.1.1 Managing metadata with mapping models

IBM InfoSphere Data Architect helps you manage the metadata about your models in the following ways:

- **Discover relationships:** The workbench can help you discover methods that show you how one or more source schemas might be related to a target schema. The relationships can be based on the metadata or any available data samples.
- **Map relationships:** IBM InfoSphere Data Architect visually maps your source and target data models and the relationships that are discovered or manually specified. The mapping that you create describes how you can combine and transform data that is represented by one or more source models into some form that is appropriate to a target model. The results are saved in a mapping model.
- **Build expressions:** You can define criteria, such as functions, join conditions, filter conditions, and sort conditions, that you can add to the mapping model. Use these expressions in scripts that you deploy.
- **Generate scripts:** You can generate a script from the mapping model that can be used to transform and filter data from mapping model-compliant sources to mapping model-compliant targets.

Mapping models help us integrate different models from various sources by defining relationships amongst them. Transformation rules can be added to the relationships and mappings created between schemas.

A *mapping* is a dependency between two data structures that is not implemented in the data source. A mapping model stores the summary of these mappings.

Assume that this student information system is one that is designed to work with multiple campuses. For example, some universities have a main campus, and they also make use of smaller, satellite campuses, typically within the same geographical area. Assume that the `NEW_SDT_INFO_SYS.dbm` physical data model models some of the data from a smaller satellite campus for a major university, but you want to seamlessly integrate that information.

Since you reverse-engineered the physical data model in Chapter 6, you can create a mapping model to map some of its metadata to the corresponding objects in the `STUDENT_INFO_SYSTEM.dbm` physical data model file.



### 8.1.2 Further managing naming standards with mapping models

You can use a glossary or naming model to discover relationships between source and target schemas. You learned about glossary models in Chapter 3 of this book. The naming rules that you defined earlier in the data design process can be used to identify the relationships between different schemas.

**Note:**

You will not perform the steps outlined in this section. They are provided for reference only.

You already created a glossary model to track the names of entities, tables, and other data objects in your data models. You could use this glossary model to ensure that all data models conform to the naming standards that you have defined for your organization. If the mapping model discovery cannot correctly determine which objects should be mapped, then perhaps some of these objects are out of compliance.

To determine if models are in compliance with naming standards with a mapping model:

1. Open the Enterprise Naming Standard window to attach a naming model to the mapping model:
  - a. Open the mapping editor by opening an existing mapping model.
  - b. Right-click in the mapping editor.
  - c. Select *Discover Relationships -> Enterprise Naming Standard*. The window opens.
2. Select the schema to which you want to attach a naming standard.
3. Click the *Add* button next to the *Naming model list* pane. The Select Naming Model Files window opens.
4. Select a glossary model (with the file extension of **.ndm**) and click *OK*.
5. Repeat the steps above for any remaining schemas, then click *Finish*.
6. Discover the relationships based on the best fit:
  - a. Right-click in the mapping editor and select *Discover Relationships -> Find Best Fit*. The Discover Relationships window opens.
  - b. Select the schemas for which you want to discover the relationships, then click *Finish*.

The models are analyzed, and the workbench determines which data objects should be related in the mapping model.

7. Approve or reject the mappings.
8. Create new mappings if necessary.
9. Save your work.

## 8.2 Building mappings within the workbench

You can create mapping models from source logical and physical data models in your data design projects. Target models for your mapping models can be logical or physical data models or XML schemas.

### Restriction:

You cannot generate scripts from a source that is a logical data model. When you create mappings between logical data models and other sources or targets, it is only for publishing purposes; that is, you can only report on the mappings.

### 8.2.1 Creating a blank mapping model

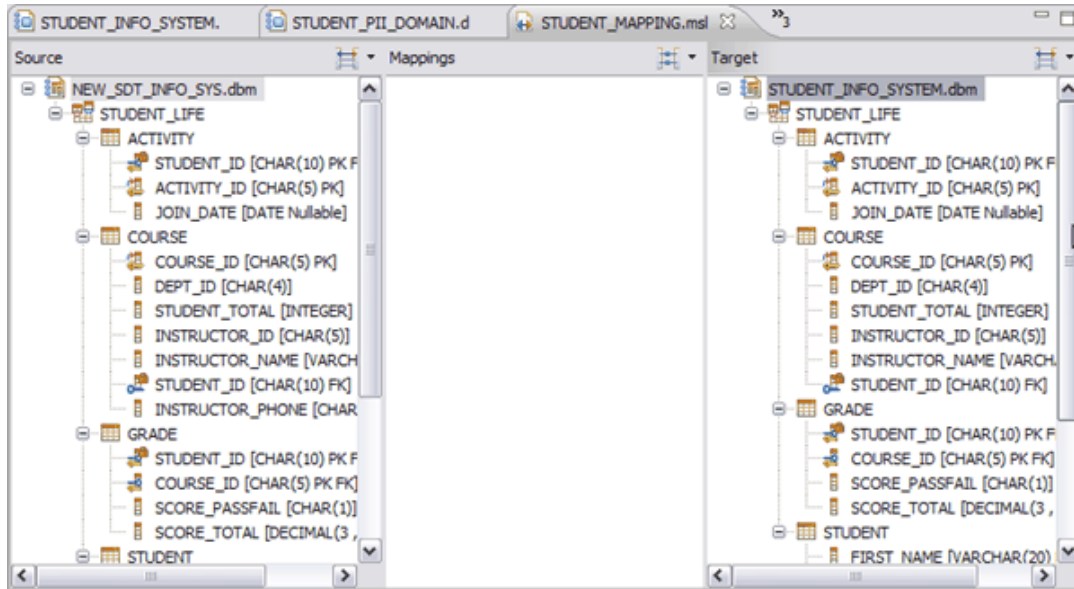
Let's create a new mapping model to map the objects in the **NEW\_SDT\_INFO\_SYS.dbm** physical data model to the **STUDENT\_INFO\_SYSTEM.dbm** physical data model:

1. Click *File -> New -> Mapping Model*. The Mapping Editor wizard opens.
2. Complete the Create Mapping page:
  - a. Make sure that the *University Info System* data design project is specified in the *Destination folder* field.
  - b. In the *File name* field, specify **STUDENT\_MAPPING.msl**.
  - c. Make sure that the *Enable script generation for this model* option is selected, so that you can create SQL scripts to deploy your mappings.
  - d. Click *Next*.
3. Complete the Specify the Mapping Source page:
  - a. Click the *Add* button. The Select a Mapping Source window opens.
  - b. Select the **NEW\_SDT\_INFO\_SYS.dbm** physical data model file, then click *OK*. The mapping source file is added to the *Mapping source files* list.
  - c. Click *Next*.
4. Do not change any of the options on the Specify the Mapping Source page, then click *Next*.
5. Complete the Specify the Mapping Target page:
  - a. Click the *Add* button. The Select a Mapping Target window opens.
  - b. Select the **STUDENT\_INFO\_SYSTEM.dbm** physical data model file, then click *OK*. The mapping target file is added to the *Mapping target files* list.
  - c. Make sure that all of the objects in the *Mapping target schemas* list are selected.
  - d. Click *Next*.
6. Verify your options on the New Mapping Model Summary page, then click *Finish*.

The mapping model file is created in the *Mappings* folder of your data design project, and it opens in the editor view.

The mapping editor has three panes, as shown in *Figure 8.2*:

- **Source:** This pane contains the source file and schema.
- **Mappings:** This pane will display mappings and mapping groups within the mapping model.
- **Target:** This pane contains the target file and schema.



**Figure 8.2 – Getting to know the mapping editor**

## 8.2.2 Adding mappings to mapping model

You can create mappings between the source and target schemas from the mapping editor. You can create mappings in two ways:

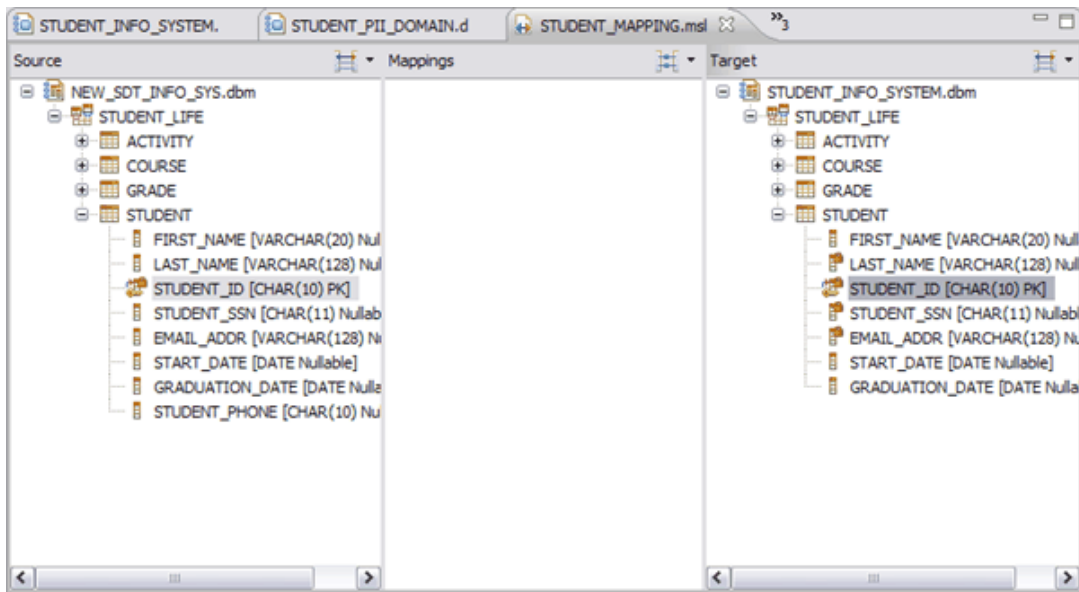
- **Manually add mappings:** Select objects in the *Source* and *Target* panes, then create a mapping between them.
- **Let the workbench discover mappings:** Run the Discover Relationships wizard to determine the best mappings between data objects.

### 8.2.2.1 Manually creating a mapping

When you create a mapping manually, you ensure that each source object is mapped to its appropriate, corresponding object in the target data model. This can be a time-consuming process, but in the case of small data models, it is sometimes best to use this method.

Let's experiment with manually creating mappings and create two mappings:

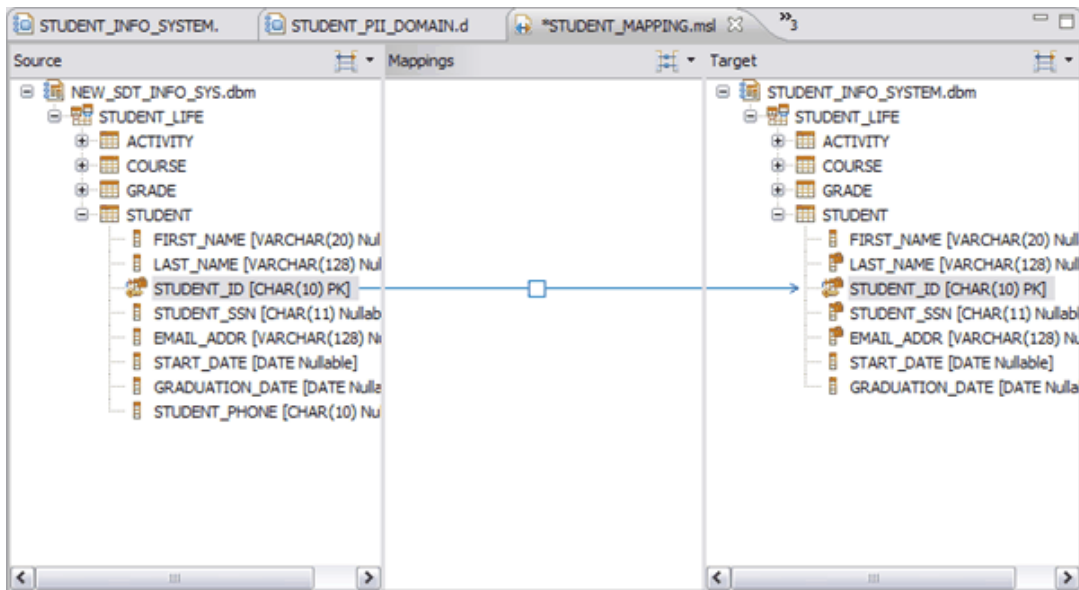
1. Map the STUDENT.STUDENT\_ID columns:
  - a. Left-click to select the STUDENT.STUDENT\_ID column in the *Source* pane.
  - b. Left-click to select the STUDENT.STUDENT\_ID column in the *Target* pane. The selected columns are shown in the image in *Figure 8.3*.



**Figure 8.3 – Selecting source and target objects**

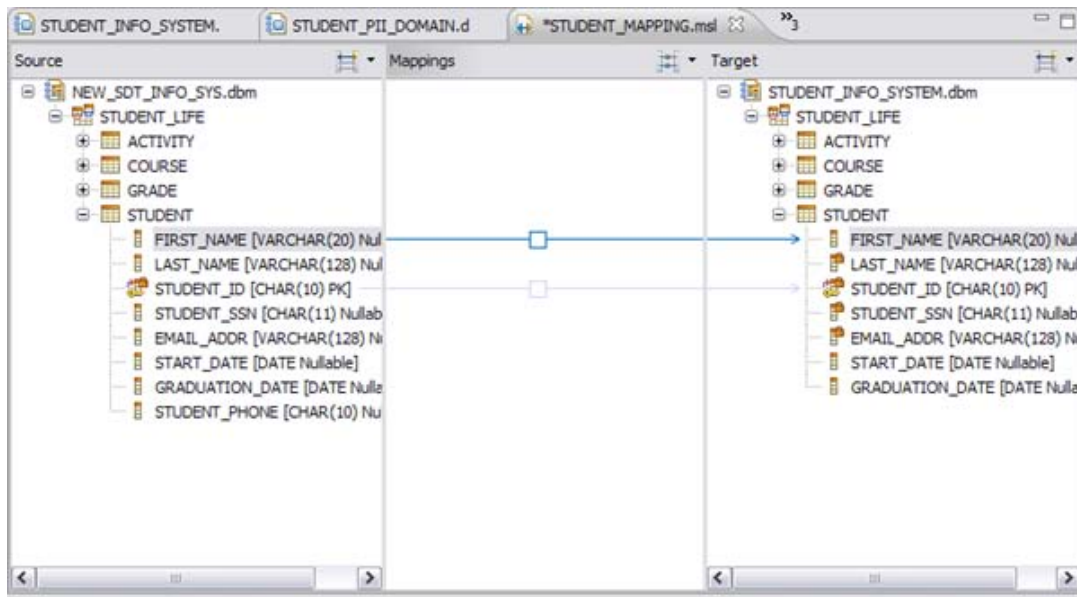
- c. Right-click anywhere in the mapping editor, then select *Create mapping*.

The mapping is created and displayed in the mapping editor, as shown in *Figure 8.4*.



**Figure 8.4 – Newly-created mapping in the mapping editor**

2. Map the STUDENT.FIRST\_NAME columns using the above process. The mapping editor should now look like the image in *Figure 8.5*.



**Figure 8.5 – Creating another new mapping**

3. Save your work.

#### 8.2.2.2 Automatically discovering relationships

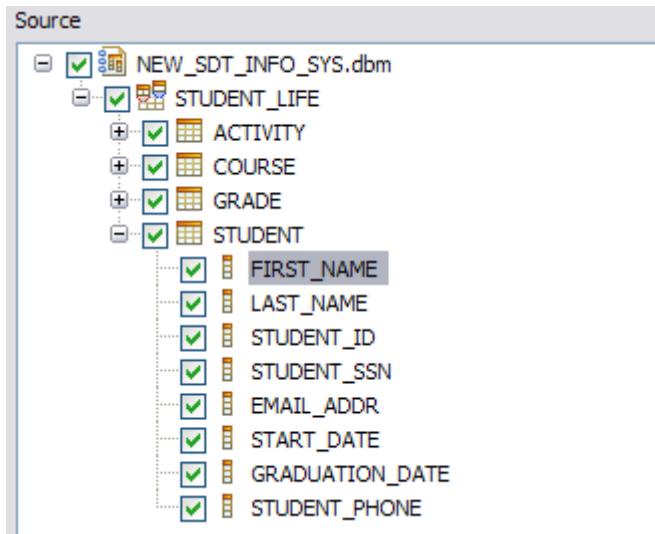
Identifying the relationships between disparate data sources is quite tedious and time consuming when you are working with large data models. In addition, the subtle relationships are not always obvious to identify. You can use the Discover Relationships wizard within the workbench to perform this task for you, based on certain algorithms.

##### Note:

You can modify how relationships are discovered from the Preferences window. Click *Window -> Preferences*. The Preferences window opens. Expand the *Data Management* node, then select *Mapping Editor -> Discover Relationships* to define how relationships are discovered.

Use your existing mapping model to discover the relationships between objects in both data models:

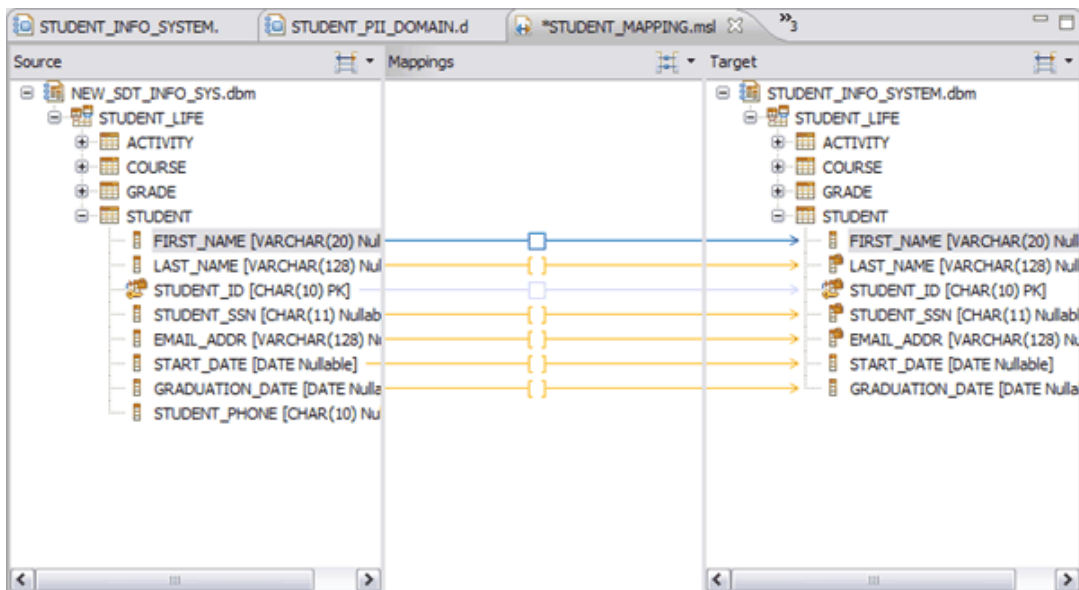
1. Right-click anywhere within the mapping model, then select *Discover Relationships -> Find Best Fit*. The Discover Relationships window opens.
2. In the *Source* pane, select the check box next to the `NEW_SDT_INFO_SYS.dbm` data model. Make sure that all objects under the physical data model are selected, as shown in *Figure 8.6*.



**Figure 8.6 – Selecting all objects in the Source pane**

3. In the *Target* pane, select the check box next to the `STUDENT_INFO_SYSTEM.dbm` data model. Make sure that all objects under the physical data model are selected.
4. Click *Finish*.

The discovery process runs, and you see a series of mapping lines that show potential relationships between the objects in the two data models. This is shown in *Figure 8.7*.



**Figure 8.7 – Mapping lines, connecting data objects between models**

### 8.2.2.3 Accepting and rejecting discovered relationships

The relationships that were discovered are only potential relationships. Before they become officially part of the mapping model, you must approve or reject the mappings.

Relationships within a mapping model are color-coded. The color codes are determined by the settings in the Preferences window, in the Mapping Editor page (found under the *Data Management* node).

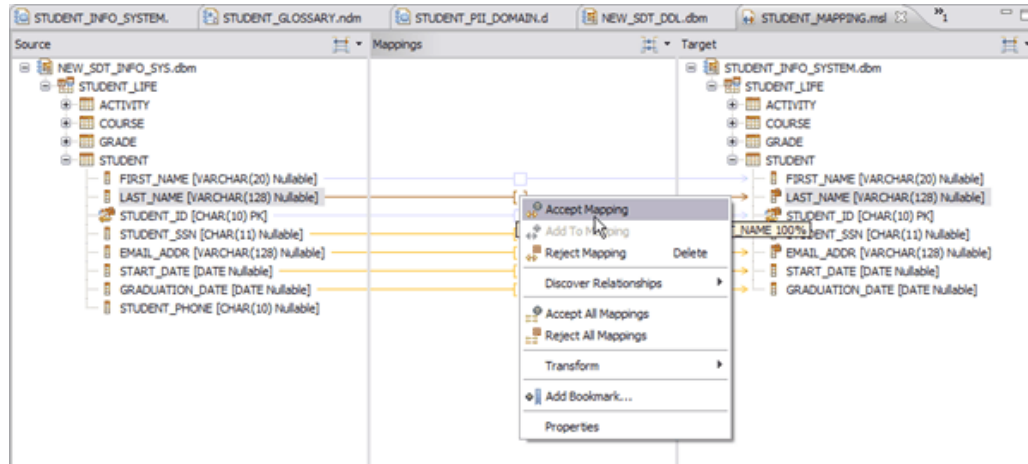
Note that the relationships you created are in blue or light blue, depending on whether the relationship is selected. Within the workbench, these are considered to be accepted mappings and are an approved part of the mapping model. Discovered mappings are in gold if they are not selected, and they appear to be brown if you select them.

You select mappings to approve or reject them. If you reject a mapping, the mapping line disappears from the mapping model.

First, you will approve a couple of relationships manually. Then, you will inspect the remaining relationships and determine whether to accept the mappings.

To accept or reject discovered relationships:

1. Select the mapping for the STUDENT.LAST\_NAME relationship. The mapping line changes to a brown color, and the discovered mapping displays in the Properties view.
2. Right-click on the mapping and select *Accept Mapping*, as shown in *Figure 8.8*.



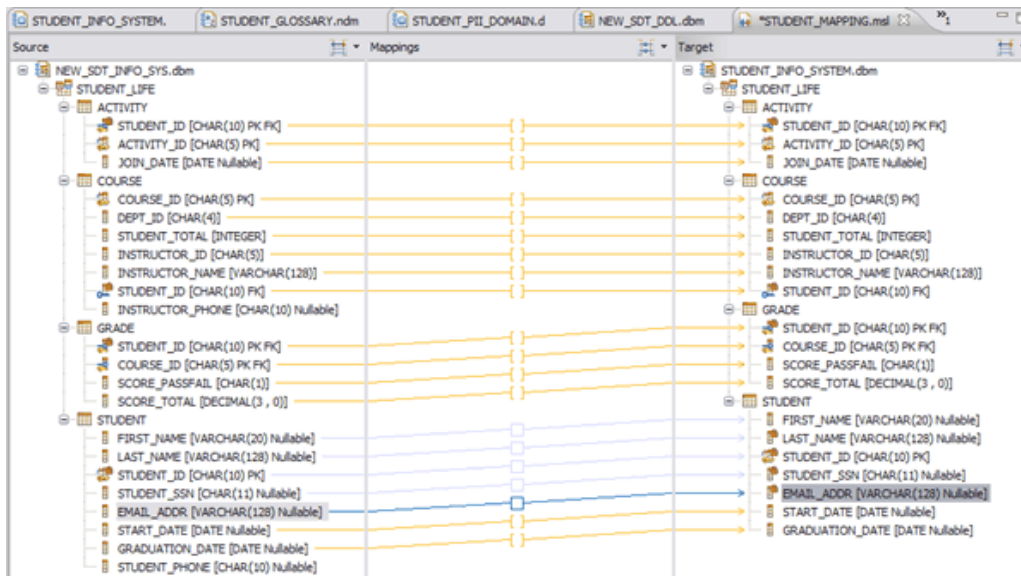
**Figure 8.8 – Manually accepting a relationship**

The mapping line color changes to blue.

3. Use the method outlined in steps 1 and 2 above to approve the mapping for the STUDENT.STUDENT\_SSN and STUDENT.EMAIL\_ADDR mappings.
4. Save your work.

At this point, you probably realize that with large data models, this can be a tedious process. Now that you are familiar with how to accept or reject mappings manually, you can view the remaining mappings and determine whether to accept all of the proposed mappings.

5. Expand all of the table nodes in the *Source* and *Target* panes. The editor should look like the image in *Figure 8.9*.



**Figure 8.9 – Viewing all of the proposed mappings**

6. View the mappings and verify that they should be mapped together. In this case, you would speak with your extended team at the satellite campus to determine what each column is intended to do, and then determine which columns should be mapped. Since you reverse-engineered the `NEW_SDT_INFO_SYS.dbm` physical data model, you know that the discovered relationships are, in fact, mapped correctly.
7. Right-click in the mapping editor and select *Accept All Mappings*.

The color of the mapping lines changes to light-blue, making them a valid part of the mapping model.

8. Save your work.

### 8.3 Types of mapping

Five types of mappings can be identified, depending on their properties and how they were created. Mappings are represented in the editor by lines that are drawn between the source and target nodes. Each mapping line contains a hub in the middle of the line.

The color of the line depends on the type of mapping. Discovered mapping lines contain hubs that appear as brackets. All other mapping lines contain square hubs.

- **Mappings:** A *mapping* connects one or more source columns to a single target column or an XML schema document (XSD) simple element or attribute.
- **Discovered mappings:** These are the mappings proposed from the Discover Relationships wizard. You can either accept or reject the individual mapping of each relationship.

Discovered mapping lines contain hubs that appear as brackets, until you approve or reject them. If you approve a discovered mapping, the hub changes to a square.

- **Constant mappings:** These mappings have a target element, but they do not have a source element. This can be used to select a transformation function like date or time function to assign



it to target element. You create constant mappings by right-clicking a target element in the Mapping Group Details view, then selecting *Create Constant Mapping* from the menu.

- **Mapping groups:** Mapping groups sort mapped elements by the target table (in a relational database) or by the element (in an XML schema). When you generate scripts for these mapping groups, one query is generated per mapping group.
- **Invalid mappings:** The mapping becomes invalid when you change the properties of data objects in the source or target model. When the mapping editor is opened, all of the mappings are re-validated. If the source or target object is no longer valid, then the mapping is invalidated and a red “X” will appear on the mapping line.

## 8.4 Adding expressions and filters to the mapping model

You can create transformation statements for each mapping that is created. The expression may be a filter, join, sort, or a transform function on a source column to get the desired target column value. The expression may change either the value or data type of the source column to be compatible with the target column. These expressions are SQL fragments and appear in the DDL that you generate from the mapping model.

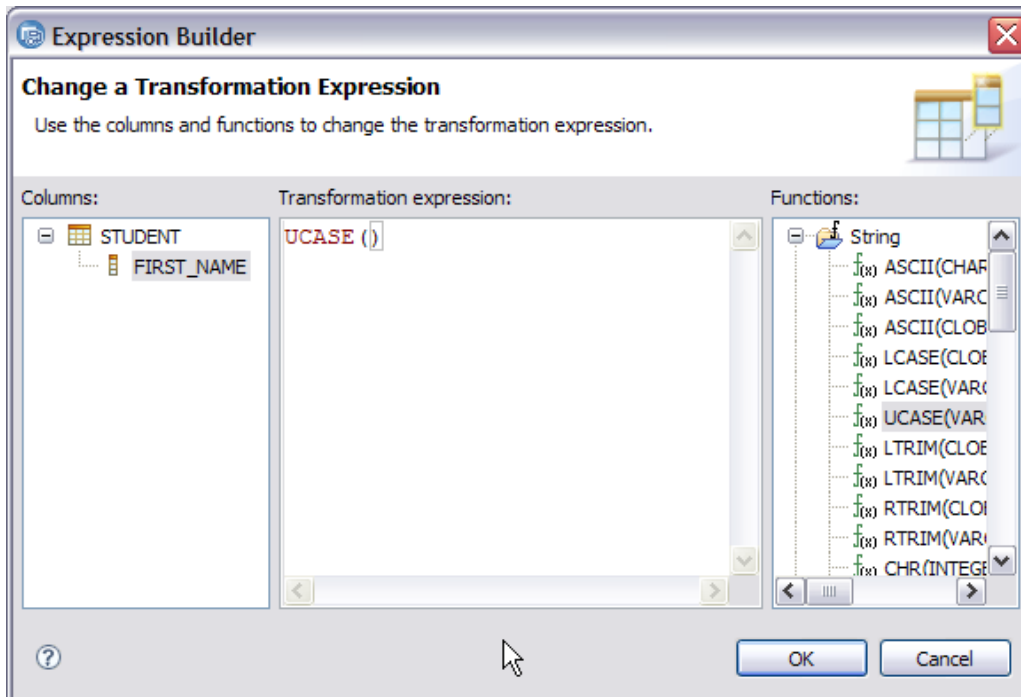
Let's create a mapping transformation expression to make sure that all of the rows in the STUDENT.FIRST\_NAME column of the **NEW\_SDT\_INFO\_SYS.dbm** physical data model appear in uppercase when they are consolidated with the information in the **STUDENT\_INFO\_SYSTEM.dbm** physical data model.

To create a transformation expression:

1. Select the mapping line between the two STUDENT.FIRST\_NAME columns. The properties of the mapping appear in the Properties view.
2. Click the *Expression Builder* button next to the *Transformation* field. The Expression Builder window opens.
3. Select a function. In this case, you will add a function to make sure that all first names stored in the database are stored in uppercase characters. Expand the *String* node and select the *UCASE(VARCHAR) – VARCHAR* function.

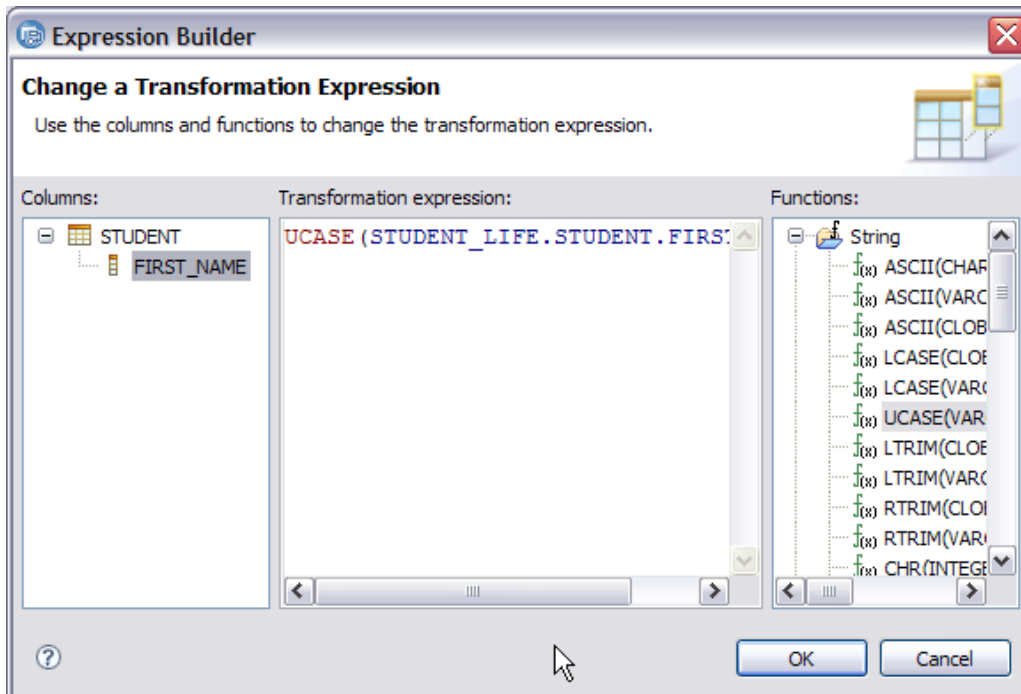
The function appears in the *Transformation expression* pane.

4. Place the cursor between the parentheses, as shown in *Figure 8.10*.



**Figure 8.10 – Placing the cursor to format the expression**

5. Add the STUDENT.FIRST\_NAME column to the expression by double-clicking the column. The expression should look like the image in *Figure 8.11*.



**Figure 8.11 – Creating a complete expression**

6. Click *OK* to add the transformation expression to the mapping model.

## 8.5 Generate scripts that you can deploy

You can generate a SQL script from the mapping model. This SQL script contains the SQL scripts necessary to account for all the mappings that you created in the mapping model. The script can then be deployed onto a database.

Let's generate a script to deploy the mapping model:

1. Locate the **STUDENT\_MAPPING.msl** mapping model in the Data Project Explorer: *University Info System -> Mappings -> STUDENT\_MAPPING.msl*.
2. Right-click on the mapping model file and select *Generate script*. The Generate Script wizard opens.
3. Complete the Generate Script Options page:
  - a. Make sure that the SQL query language is selected.
  - b. Make sure that the *Create a Select statement* query type is selected.
  - c. Select the *Create fully qualified names* check box.
  - d. Specify **STUDENT\_MAPPING.sql** as the script file name.
  - e. Click *Next*.
4. Review the script on the Summary page, then click *Finish* to generate the SQL script.

The script is generated and opens in the editor view. When you run this script on the server, the queries run and generate the mappings between the schemas in the physical data models.

## 8.6 Export mapping models in CSV format

You can export the mapping model into a comma separated file (using the **.csv** file extension). The CSV format is supported by other tools (including spreadsheet tools), and therefore, you can use this file to import the mapping model into other tools.

When you export a mapping model to a file in CSV format, the contents of the mapping model are transformed as follows:

- For one-to-one mappings, the resulting CSV file contains one row for each mapping.
- For many-to-one mappings, there are multiple rows, one for each source and target pair.

Export your mapping model to a CSV file:

1. From the main menu, click *File -> Export*. The Export wizard opens.
2. Expand the *Data* node, and select the *Export a mapping model to a file in CSV format* option, then click *Next*.
3. Complete the Mapping Model Export page:

- a. Click the *Browse* button next to the *Model to export* field. The Export Mapping Model window opens.
  - b. Select the `STUDENT_MAPPING.ms1` mapping model file, then click *OK* to add it to the wizard.
  - c. Click the *Browse* button next to the *Export to* field. The Save As window opens.
  - d. Specify `STUDENT_MAPPING.csv` in the *File name* field, then click *OK*.
4. Click *Finish* to create the CSV file in your workspace.

The CSV file is created and saved in the *University Info System* directory of your workspace path. For example, it could be saved to the following file path on a Windows machine:

```
C:\Documents and  
Settings\Administrator\IBM\workspaces\IDA_GettingStarted\University Info  
System\STUDENT_MAPPING.csv
```

You can import this file to other tools that support the `*.csv` file format.

## 8.7 Exercise

1. Create a mapping model of the `NEW_SDT_DDL.dbm` physical data model and the `STUDENT_INFO_SYSTEM.dbm` physical data model.
2. Run the Discover Relationships tool and approve or reject mappings within the mapping model.
3. Create new mappings as necessary.
4. Add a transformation function to one of the mappings.
5. Generate an SQL script from the mapping model.
6. Create a CSV file that can be used with other tools.

## 8.8 Summary

In this chapter, you learned about mapping models and how they can be used to manage various data sources and the relationships between data models. You learned how to create a mapping model and discover the potential relationships between the objects in your data models, and you also learned how to manually create the relationships between these objects. Then you generated a SQL file to create these relationships on the server, and you generated a CSV file to use the mapping model in other tools.

## 8.9 Review questions

1. What is a mapping model?
2. Why would you need to create a mapping model?
3. What are the different types of mapping?
4. Which of the following transformation functions can be added to a mapping group?
  - A. Filter

- B. Sort
  - C. Join
  - D. All of the above
5. True or false: You can map more than one source element to a single target element.



# 9

## Chapter 9 – Analyzing Data Models

IBM InfoSphere Data Architect is a free-form modeling tool, which means you can create a model that suits the needs of the business, and the workbench will not restrict the design of the model in any way.

When you design a model, however, you could possibly unintentionally create a model that violates the logical or physical data modeling rules or constraints which could introduce errors or do not conform to common data design best practices.

To help you identify issues with the model, use the Analyze Model wizard. In this chapter, you will learn the following concepts:

- How to analyze the data model
- How to change the preferences to influence how models are analyzed
- How to address errors and warnings after the model is analyzed

### 9.1 Analyzing data models: The big picture

You *analyze* a data model after you create it. The Analyze Model wizard validates the model in two major ways:

1. **Design Suggestions:** These are best practice suggestions or design constraints that are put in place by the architects so that the model follows a standard.
2. **Syntax Checks:** Syntax checks are validations validate the syntax of attributes and also checks if the valid relations are created between entities or tables.

Ideally, you analyze data models before you transform or deploy them. This ensures that you are not transforming or deploying invalid models, and you can determine what problems exist (if any) *before* these models go into production and cause potential issues.

### 9.2 Analyzing data models with the workbench

#### 9.2.1 Analyzing logical data models with the workbench

You can only analyze logical data models from the package node. Let's analyze the `STUDENT_INFO_SYSTEM.1dm` logical data model to ensure that it is valid.

To analyze a logical data model:

1. Locate the STUDENT INFO SYSTEM package within the logical data model: *University Info System -> Data Models -> STUDENT\_INFO\_SYSTEM.Idm -> STUDENT INFO SYSTEM*.
2. Right-click on the STUDENT INFO SYSTEM package and select *Analyze Model*. The Analyze Model wizard opens.

Since we selected a logical data model package, by default, all of the rule sets under the *Logical Data Model* node are selected. The rule sets in the *Logical Data Model* node include best practices, naming standards, and common design suggestions. Each rule set typically contains its own set of rules.

You can pick and choose which rules you want to adhere to when you validate your model, depending on what works best for your organization. Are naming standards important? Do you use domains? Does your data model have to be fully normalized?

3. Since you have already validated the naming standards, deselect the *Naming Standard* rule set.
4. Since you did not create a dimensional model, deselect the *Dimensional Modeling* rule set.
5. Click *Finish* to analyze the model.

The model is analyzed, and you see two warnings in the Problems view.

A *warning* specifies best practices, database limitations, or organizational requirements. Warnings do not make any model invalid by themselves. Warnings do not require changes to the data model before it is deployed. Warnings simply point out potential conflicts that could warrant design changes to your model. You should look at all warnings and act on them accordingly to make sure that any database limitations are addressed, and that you are adhering to organizational standards. You decide, however, whether to correct the warnings that relate to a data design best practice.

In this case, the two warnings in the Problems view note that the COURSE ID and STUDENT ID attributes in the COURSE and STUDENT entities are not unique. These attributes were added when you created relationships between the two entities. Therefore, you do not need to fix these attributes.

An *error* in the Problems view means that there is an issue with the data model that must be fixed. Errors will cause problems in a production environment, and you should fix them before you transform or deploy them to a production environment.

### 9.2.2 Analyzing physical data models with the workbench

Let's analyze the `STUDENT_INFO_SYSTEM.dbm` physical data model. You can analyze physical data models from the database or schema node.

To analyze a physical data model:

1. Locate the SAMPLE database object in the physical data model: *University Info System -> Data Models -> STUDENT\_INFO\_SYSTEM.dbm -> SAMPLE*.
2. Right-click on the SAMPLE database object and select *Analyze Model*. The Analyze Model wizard opens.

Since we selected an object within the physical data model, by default, all of the rule sets under the *Physical Data Model* category are selected. Just as in the logical data model analysis, the physical data



model analysis contains categories with their own rules that check for common best practices, naming standards, and design suggestions.

3. Because we did not create a dimensional model, deselect the *Dimensional Modeling* rule set.
4. Because we do not have table spaces for this physical data model, make sure that the tool does not validate table space options:
  - a. Select the *Index and Storage* rule set under the *Design Suggestions* node. The rules for the *Index and Storage* rule set display in the *Rules* pane.
  - b. Deselect the *Table space design check* and *Table with table space check* rules.
  - c. Deselect the *Table spaces* rule set under the *Syntax Check* node.
5. Deselect the *Naming Standard* rule set. Click *Finish* to analyze the physical data model.

You see 7 warnings and 13 informational messages about the physical data model. If you wish, you can navigate through the problems in the Problems view and fix the warnings.

### 9.2.3 Fixing errors and warnings in the Problems view

If you want to fix the errors or warnings in the Problems view:

1. View the entire message to see the design suggestions offered by the tool.
2. Double-click the message to navigate to the data object or objects affected by the warning message.
3. Update the objects according to the design suggestions offered in the Problems view.

#### Note:

You can fix some of the messages in the *Problems view* by right-clicking on the message, then selecting *Quick Fix*.

## 9.3 Modifying the preferences for model analysis

You can modify the categories and rules that are selected by default when you analyze your data models. If you perform these steps, you do not have to spend time selecting and deselecting the options that you always use in the Analyze Model wizard.

#### Note:

You will not actively modify these preferences in this section, but you will learn how to get to these options in the Preferences window.

To modify preferences for the Analyze Model wizard:

1. From the main menu, click *Window -> Preferences*. The Preferences window opens.
2. Expand the *Model Validation* node and select the *Constraints* page. Use this page to change the default settings for what categories and rules are selected when you validate your data models.

For example, if you do not implement a naming standard for logical data models, expand the *Logical data model* -> *Design suggestions* node and deselect the *Naming standard* category.

3. Once you have made your changes, click *Apply*, then click *OK*.

The next time you validate your data model, your preferred categories and rules are selected.

## 9.4 Summary

In this chapter you have learned how to analyze a model to make sure that it conforms to organizational standards or standard design syntax rules. You learned about warnings and errors in the Problems view and how to address them.

## 9.5 Exercise

Modify the default preferences for the Analyze Model wizard.

1. First, analyze your existing physical and logical data models without changing the default settings. Document your results.
2. Then, use the Preferences window to ensure that both physical and logical data models are not checked for naming standards, table spaces, or dimensional modeling rules.
3. Analyze your physical and logical data models, and document the results. Do not fix the warnings or errors.
4. Then, open the Preferences window and restore the default settings for the model validation tool.

## 9.6 Review questions

1. At what level of a data model can you invoke the Analyze Model wizard?
  - A. Model file
  - B. Diagram
  - C. Package, database, or schema
  - D. Entity or table
2. What type of validation occurs when you analyze your models?
  - A. Best practices
  - B. Model syntax
  - C. Valid design
  - D. All of the above
3. Warnings point out potential problems in the following area:
  - A. Database limitations and best practices
  - B. Organizational requirements

- C. Both A and B
  - D. None of the above
4. How do you change the default preferences for the Analyze Model wizard?
  5. When is the best time to analyze a model?
  6. What should you keep in mind when analyzing your data model or determining whether to address a warning?
  7. True or false: You must correct warnings in the Problems view.
  8. True or false: Your model is considered invalid if there are errors in the Problems view.



# 10

## Chapter 10 – The Data Management Life Cycle

### 10.1 Managing your data

The Data Security, Privacy, and Lifecycle Management portfolio is a unified solution set that facilitates management of data throughout all the stages of the software development cycle, from gathering requirements to project retirement. IBM InfoSphere Data Architect is a part of the Data Security, Privacy, and Lifecycle Management portfolio. Some of the popular products in the portfolio are:

- IBM InfoSphere Data Architect
- Optim Development Studio and pureQuery Runtime
- Optim Database Administrator
- Optim Performance Manager Extended Edition
- Optim High Performance Unload
- Optim Query Tuner

The following list contains some high-level descriptions of these products in the Data Security, Privacy, and Lifecycle Management family:

- **Optim Development Studio**

Optim Development Studio allows you to develop and administer databases. With Optim Development Studio, you can create, alter, and drop objects; manage database privileges; view distributions of data; and gather statistics on data objects. You can also develop and test SQL and XQuery queries, stored procedures, Web services, and Java data access layers.

Optim pureQuery Runtime is an optional feature that you can use with Optim Development Studio. Optim pureQuery Runtime is a high-performance data access platform that makes it easier to monitor, develop, optimize, secure, and manage data access. Use APIs to simplify best practices and enable collaboration between developers and database administrators, or monitor your SQL hot spots to gain insight to improve response times.

- **Optim Database Administrator**

Optim Database Administrator allows you to administer complex structural changes to databases and manage database migration scenarios. You can automate structural changes to better manage your test databases and speed up deployments, streamlining administrative tasks. Analyze the impact of database changes to minimize the risk to your deployment environment. You can also manage large environments more efficiently with Optim Database Administrator.

- **Optim Performance Manager Extended Edition**

Optim Performance Manager helps database administrators improve DB2 performance and availability. DB2 Performance Expert helps you identify, diagnose, solve, and prevent performance problems before they impact the business. Optim Performance Manager provides optimization and tuning recommendations and helps you analyze trends to plan for database growth. It monitors your applications and databases, identifying potential problems and managing your workload before it impacts performance.

- **Optim High Performance Unload**

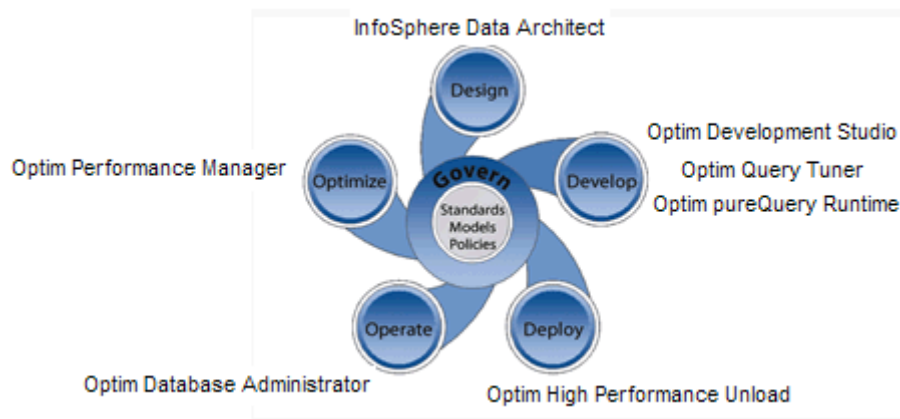
Optim High Performance Unload helps database administrators work with very large quantities of data with less effort and faster results. Optim High Performance Unload runs concurrently with production applications, maintaining system availability while minimizing the impact to production performance. Optim High Performance Unload can also fit unloads, backups, migrations, re-partitioning into tight or shrinking batch windows, reducing the risk of impact to your online production environment.

- **Optim Query Tuner**

Optim Query Tuner helps developers create efficient queries and build their tuning skills. It improves performance by providing expert advice about how to write high-quality queries and improve the design of the database.

### **10.1.1 The data management life cycle**

The products listed in the previous section, either individually or in collaboration, provide solutions in each phase of software development life cycle. *Figure 10.1* illustrates this life cycle that can be accomplished with IBM's Data Security, Privacy, and Lifecycle Management portfolio.



**Figure 10.1 – Data Security, Privacy, and Lifecycle Management tools**

IBM InfoSphere Data Architect, as discussed previously, is primarily used in the design phase of the integrated data management life cycle, although modifications to the design are commonplace throughout the life cycle of any project. You can use the workbench to manage these changes.

With the Data Security, Privacy, and Lifecycle Management portfolio, all of these products are presented to you as a single solution (see section 10.1.2 on Integration capabilities of IBM InfoSphere Data Architect and 10.1.3 on shell-sharing). With Data Security, Privacy, and Lifecycle Management, the entire product life cycle can be managed. Because the portfolio overcomes the problems of interoperability and compatibility, you can achieve the following improvements:

- Develop enterprise-ready applications faster
- Effectively manage data growth
- Optimize database application performance
- Protect data privacy in all stages of the product/application life cycle
- Simplify migrations and upgrades

### 10.1.2 Integrating IBM InfoSphere Data Architect with other products

In any software development team, whether large or small, you should use tools that can share data or information seamlessly. This makes it easier to reuse all of the information throughout the various stages of product development.

IBM InfoSphere Data Architect supports model-driven design by integrating with the following products:

- InfoSphere Business Glossary: Share enterprise glossaries across the business
- Cognos Business Intelligence: Author, share, and use reports from information in your database
- Rational Software Architect: Synch the design of applications and information
- Rational RequisitePro: Ensure that the model is complete for the requirements of the business
- WebSphere® Business Modeler Advanced: Transform data models to XML schema definitions for business processes

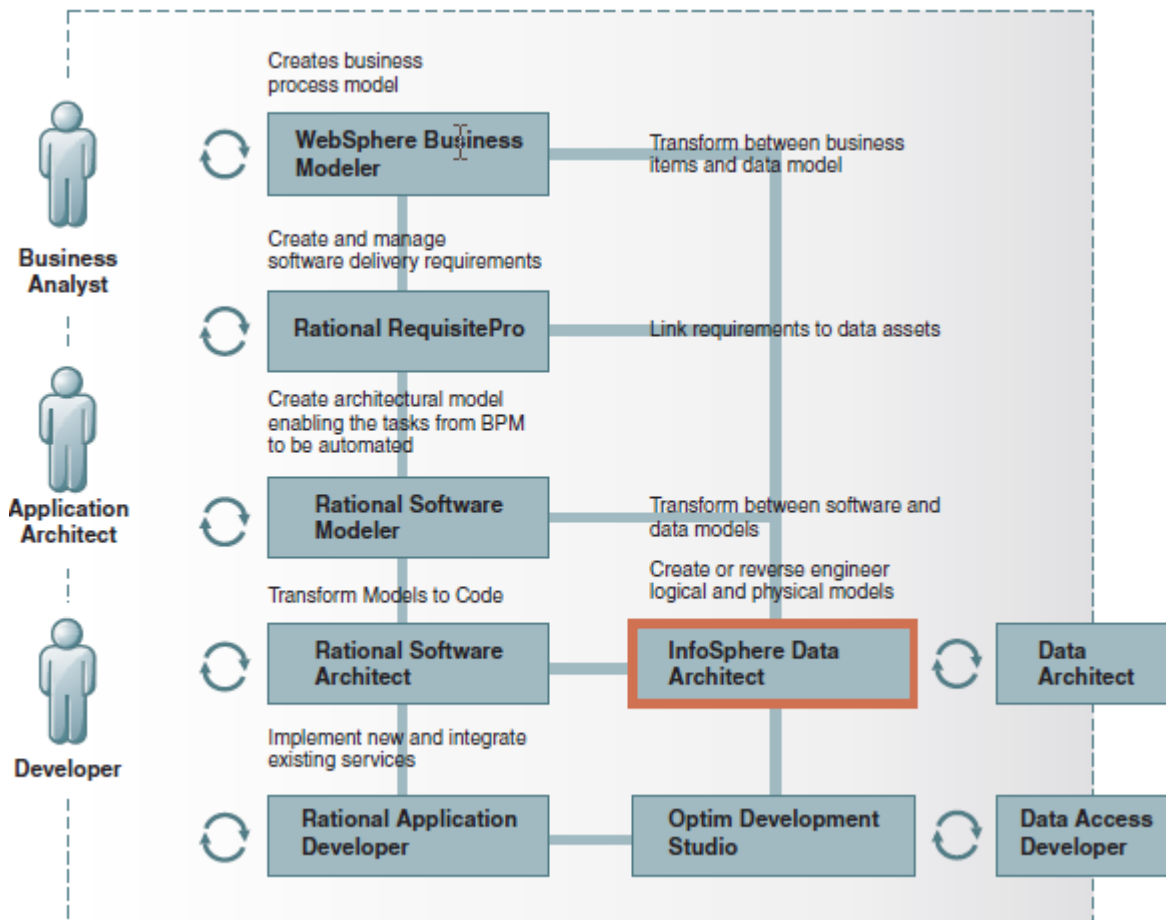


Figure 10.2 – Integrating InfoSphere Data Architect with other software development products

### 10.1.3 Shell-sharing with other Eclipse-based products

IBM InfoSphere Data Architect is built on top of the Eclipse platform, which is a popular open-source platform that you can use to develop applications. The Eclipse platform allows you to integrate with other Eclipse-based products.

*Shell-sharing* between Eclipse-based products basically means that you can share the core components of products so that they are not duplicated across each of the products. It eliminates the need for each product to have its own Eclipse platform. A simple analogy to understand shell-sharing is to compare it to a Microsoft Windows installation: all of the common components required by Windows Program Files are stored in the `C:\Windows` folder.

At a basic level, shell-sharing translates into disk space savings, and duplicate components are not installed. Shell-sharing achieves this result by sharing common and compatible features. Therefore, you can interact with all of the features of each of the products via a single user interface. If, for example, you shell-share IBM InfoSphere Data Architect and Optim Development Studio with each other, you can develop and architect databases from a single, common platform. This common platform is launched without regard to which of the products is launched.



You can shell-share any number of compatible products. For example, you can also integrate Rational Software Architect with IBM InfoSphere Data Architect and Optim Development Studio in order to use the features of Rational Software Architect.

**Note:**

For an example about how to shell-share InfoSphere Data Architect, Optim Development Studio, and Optim Database Administrator, see the following developerWorks article:

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



## References

- [1] IBM® InfoSphere Data Architect Information Center, October 2010.  
<http://publib.boulder.ibm.com/infocenter/rdahelp/v7r5/index.jsp>
- [2] LIVENGGOOD, E. *InfoSphere Data Architect Evaluation Guide*, white paper, October 2010.  
<http://download.boulder.ibm.com/ibmdl/pub/software/data/sw-library/studio/evaluation-guides/RDA75EvalGuide.pdf>
- [3] TSOUNIS, S., CHOU, D.P. *Best Practices for Rational Data Architect*, best practices paper, October 2008. <http://www.ibm.com/developerworks/data/bestpractices/informationmodeling/>
- [4] MACHADO, J., VOELLINGER, H. *Develop mapping models with IBM InfoSphere Data Architect*, developerWorks article, January 2010. <http://www.ibm.com/developerworks/data/tutorials/dm-1101mappingmodelsida/index.html>
- [5] PATEL, A. *DB2 basics: Table spaces and buffer pools*, developerWorks article, April 2010.  
<http://www.ibm.com/developerworks/data/library/techarticle/0212wieser/index.html>
- [6] GOPAL, V., BHAGAVAN, S. *Data modeling with InfoSphere Data Architect and Informix Dynamic Server*, developerWorks article, March 2009. <http://www.ibm.com/developerworks/data/tutorials/dm-0903datamodel/section8.html>
- [7] LIU, W. *Use InfoSphere Data Architect to define and enforce data object naming standards*, developerWorks article, October 2010. <http://www.ibm.com/developerworks/data/library/techarticle/dm-0701liu/index.html>

## Resources

### Web sites

DB2 Express-C web site:

[www.ibm.com/db2/express](http://www.ibm.com/db2/express)

Use this web site to download the image for DB2 Express-C servers, DB2 clients, DB2 drivers, manuals, access to the team blog, mailing list sign up, etc.

DB2 Express-C forum:

[www.ibm.com/developerworks/forums/dw\\_forum.jsp?forum=805&cat=19](http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=805&cat=19)

Use the forum to post technical questions when you cannot find the answers in the manuals yourself.

DB2 Information Center

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp>

The information center provides access to the DB2 online manuals. It is the most up to date source of information.

InfoSphere Data Architect Information Center

<http://publib.boulder.ibm.com/infocenter/rdahelp/v7r5/index.jsp>

The information center provides access to the InfoSphere Data Architect online manuals and help documentation. It contains tutorials, conceptual information, and tasks to help you get started with InfoSphere Data Architect.

developerWorks: DB2

<http://www-128.ibm.com/developerworks/db2>

This Web site is an excellent resource for developers and DBAs providing access to current articles, tutorials, etc. for free.

developerWorks: InfoSphere Data Architect

<http://www.ibm.com/developerworks/data/products/dataarchitect/index.html>

This Web site provides articles, tutorials, and other resources to help you learn more about data modeling with InfoSphere Data Architect.

InfoSphere Data Architect developerWorks forum

<http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1796>

This forum is an informal source of support. It is monitored by IBM professionals and InfoSphere Data Architect enthusiasts alike.

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.

planetDB2

[www.planetDB2.com](http://www.planetDB2.com)

This is a blog aggregator from many contributors who blog about DB2.

DB2 Technical Support

[http://www.ibm.com/software/data/db2/support/db2\\_9/](http://www.ibm.com/software/data/db2/support/db2_9/)

If you purchased the 12 months subscription license of DB2 Express-C, you can download fixpacks from this Web site.

InfoSphere Data Architect technical support

[http://www.ibm.com/support/entry/portal/Overview/Software/Information\\_Management/InfoSphere\\_Data\\_Architect](http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/InfoSphere_Data_Architect)

This is the support portal for InfoSphere Data Architect. Use this site to see the latest support alerts, troubleshooting information, and other documentation related to the product, or to contact IBM Support.

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

1. Free Redbook: DB2 Express-C: The Developer Handbook for XML, PHP, C/C++, Java, and .NET  
Whei-Jen Chen, John Chun, Naomi Ngan, Rakesh Ranjan, Manoj K. Sardana,  
August 2006 - SG24-7301-00  
<http://www.redbooks.ibm.com/abstracts/sg247301.html?Open>
2. Understanding DB2 – Learning Visually with Examples V9.5  
Raul F. Chong, et all. January 2008  
ISBN-10: 0131580183
3. DB2 9: pureXML overview and fast start by Cynthia M. Saracco, Don Chamberlin, Rav Ahuja  
June 2006 SG24-7298  
<http://www.redbooks.ibm.com/abstracts/sg247298.html?Open>
4. DB2® SQL PL: Essential Guide for DB2® UDB on Linux™, UNIX®, Windows™, i5/OS™, and z/OS®, 2nd Edition  
Zamil Janmohamed, Clara Liu, Drew Bradstock, Raul Chong, Michael Gao, Fraser McArthur, Paul Yip  
ISBN: 0-13-100772-6
5. Free Redbook: DB2 pureXML Guide  
Whei-Jen Chen, Art Sammartino, Dobromir Goutev, Felicity Hendricks, Ippei Komi, Ming-Pang Wei, Rav Ahuja, Matthias Nicola. August 2007  
<http://www.redbooks.ibm.com/abstracts/sg247315.html?Open>
6. Information on Demand - Introduction to DB2 9 New Features  
Paul Zikopoulos, George Baklarz, Chris Eaton, Leon Katsnelson  
ISBN-10: 0071487832  
ISBN-13: 978-0071487832

## Contact emails

General DB2 Express-C mailbox: [db2x@ca.ibm.com](mailto:db2x@ca.ibm.com)

General DB2 on Campus program mailbox: [db2univ@ca.ibm.com](mailto:db2univ@ca.ibm.com)

**Getting started with InfoSphere Data Architect couldn't be easier.**

**Read this book to:**

- Find out what InfoSphere Data Architect can do for you
- Learn how to create, deploy, and update data models
- Manage and analyze the impact of changes to your data models
- Discover how to forward- and reverse-engineer your data sources
- Learn more about data privacy, security, and lifecycle management
- Practice using hands-on exercises

InfoSphere Data Architect is a collaborative data design solution that you can use to discover, model, visualize, relate, and standardize diverse and distributed data assets. You can use InfoSphere Data Architect to leverage data model value across the data lifecycle to promote cross-lifecycle, cross-role and cross-organization collaboration and to align process service, application, and data architecture.

This book provides a comprehensive look at InfoSphere Data Architect and includes hands-on exercises using SQL and DB2 Express-C, the free version of DB2.

To learn more or download a free trial of InfoSphere Data Architect, visit [ibm.com/developerworks/downloads/r/rda/](http://ibm.com/developerworks/downloads/r/rda/)

To learn more or download DB2 Express-C, visit [ibm.com/db2/express](http://ibm.com/db2/express)

To socialize and watch related videos, visit [channelDB2.com](http://channelDB2.com)

This book is part of the DB2 on Campus book series, free eBooks for the community. Learn more at [db2university.com](http://db2university.com)

ISBN 978-0-9866283-9-9



Price: 24.99USD