

# Data Integration Blueprint and Modeling

Techniques for a Scalable and  
Sustainable Architecture

Anthony David Giordano



The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

© Copyright 2011 by International Business Machines Corporation. All rights reserved.

Note to U.S. Government Users: Documentation related to restricted right. Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corporation.

IBM Press Program Managers: Steven M. Stansel, Ellice Uffer

Cover design: IBM Corporation

Editor in Chief: Mark Taub

Marketing Manager: Stephane Nakib

Publicist: Heather Fox

Acquisitions Editors: Bernard Goodwin, Michael Thurston

Development Editor: Michael Thurston

Managing Editor: Kristy Hart

Designer: Alan Clements

Project Editor: Betsy Harris

Copy Editor: Karen Annett

Senior Indexer: Cheryl Lenser

Senior Compositor: Gloria Schurick

Proofreader: Language Logistics, LLC

Manufacturing Buyer: Dan Uhrig

Published by Pearson plc

Publishing as IBM Press

IBM Press offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales

1-800-382-3419

corpsales@pearsontechgroup.com

For sales outside the U.S., please contact:

International Sales

international@pearson.com

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both: IBM, Global Business Services, DataStage, Cognos, Tivoli. Microsoft, Excel, PowerPoint, Visio are trademarks of Microsoft Corporation in the United States, other countries, or both. Oracle and Java are registered trademarks of Oracle and/or its affiliates. UNIX is a registered trademark of The Open Group in the United States and other countries. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.

Library of Congress Cataloging-in-Publication Data

Giordano, Anthony, 1959-

Data integration : blueprint and modeling techniques for a scalable and sustainable architecture / Anthony Giordano.

p. cm.

ISBN-13: 978-0-13-708493-7 (hardback : alk. paper)

ISBN-10: 0-13-708493-5 (hardback : alk. paper)

1. Data integration (Computer Science) 2. Data structures (Computer science) I. Title.

QA76.9.D338G56 2010

005.7'3—dc22

2010041861

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc  
Rights and Contracts Department  
501 Boylston Street, Suite 900  
Boston, MA 02116  
Fax (617) 671-3447

First printing December 2010

ISBN-13: 978-0-13-708493-7

ISBN-10: 0-13-708493-5

# Contents

<b>Preface</b>	<b>xix</b>
<b>Acknowledgments</b>	<b>xxii</b>
<b>About the Author</b>	<b>xxiii</b>
<b>Introduction: Why Is Data Integration Important?</b>	<b>1</b>
<b>Part 1 Overview of Data Integration</b>	<b>5</b>
<b>Chapter 1 Types of Data Integration</b>	<b>7</b>
Data Integration Architectural Patterns	7
Enterprise Application Integration (EAI)	8
Service-Oriented Architecture (SOA)	9
Federation	12
Extract, Transform, Load (ETL)	14
Common Data Integration Functionality	15
Summary	16
End-of-Chapter Questions	16
<b>Chapter 2 An Architecture for Data Integration</b>	<b>19</b>
What Is Reference Architecture?	19
Reference Architecture for Data Integration	20
Objectives of the Data Integration Reference Architecture	21
The Data Subject Area-Based Component Design Approach	22
A Scalable Architecture	24
Purposes of the Data Integration Reference Architecture	26
The Layers of the Data Integration Architecture	26
Extract/Subscribe Processes	27
Data Integration Guiding Principle: “Read Once, Write Many”	28
Data Integration Guiding Principle: “Grab Everything”	28
Initial Staging Landing Zone	29

Data Quality Processes	31
What Is Data Quality?	31
Causes of Poor Data Quality	31
Data Quality Check Points	32
Where to Perform a Data Quality Check	32
Clean Staging Landing Zone	34
Transform Processes	35
Conforming Transform Types	35
Calculations and Splits Transform Types	35
Processing and Enrichment Transform Types	36
Target Filters Transform Types	38
Load-Ready Publish Landing Zone	39
Load/Publish Processes	40
Physical Load Architectures	41
An Overall Data Architecture	41
Summary	42
End-of-Chapter Questions	43
<b>Chapter 3 A Design Technique: Data Integration Modeling</b>	<b>45</b>
The Business Case for a New Design Process	45
Improving the Development Process	47
Leveraging Process Modeling for Data Integration	48
Overview of Data Integration Modeling	48
Modeling to the Data Integration Architecture	48
Data Integration Models within the SDLC	49
Structuring Models on the Reference Architecture	50
Conceptual Data Integration Models	51
Logical Data Integration Models	51
High-Level Logical Data Integration Model	52
Logical Extraction Data Integration Models	52
Logical Data Quality Data Integration Models	53
Logical Transform Data Integration Models	54
Logical Load Data Integration Models	55
Physical Data Integration Models	56
Converting Logical Data Integration Models to Physical Data Integration Models	56
Target-Based Data Integration Design Technique Overview	56
Physical Source System Data Integration Models	57
Physical Common Component Data Integration Models	58
Physical Subject Area Load Data Integration Models	60
Logical Versus Physical Data Integration Models	61
Tools for Developing Data Integration Models	61
Industry-Based Data Integration Models	63
Summary	64
End-of-Chapter Questions	65

---

<b>Chapter 4</b>	<b>Case Study: Customer Loan Data Warehouse Project</b>	<b>67</b>
	Case Study Overview	67
	Step 1: Build a Conceptual Data Integration Model	69
	Step 2: Build a High-Level Logical Model Data Integration Model	70
	Step 3: Build the Logical Extract DI Models	72
	Confirm the Subject Area Focus from the Data Mapping Document	73
	Review Whether the Existing Data Integration Environment Can Fulfill the Requirements	74
	Determine the Business Extraction Rules	74
	Control File Check Processing	74
	Complete the Logical Extract Data Integration Models	74
	Final Thoughts on Designing a Logical Extract DI Model	76
	Step 4: Define a Logical Data Quality DI Model	76
	Design a Logical Data Quality Data Integration Model	77
	Identify Technical and Business Data Quality Criteria	77
	Determine Absolute and Optional Data Quality Criteria	80
	Step 5: Define the Logical Transform DI Model	81
	Step 6: Define the Logical Load DI Model	85
	Step 7: Determine the Physicalization Strategy	87
	Step 8: Convert the Logical Extract Models into Physical Source System Extract DI Models	88
	Step 9: Refine the Logical Load Models into Physical Source System Subject Area Load DI Models	90
	Step 10: Package the Enterprise Business Rules into Common Component Models	92
	Step 11: Sequence the Physical DI Models	94
	Summary	95
<b>Part 2</b>	<b>The Data Integration Systems Development Life Cycle</b>	<b>97</b>
<b>Chapter 5</b>	<b>Data Integration Analysis</b>	<b>99</b>
	Analyzing Data Integration Requirements	100
	Building a Conceptual Data Integration Model	101
	Key Conceptual Data Integration Modeling Task Steps	102
	Why Is Source System Data Discovery So Difficult?	103
	Performing Source System Data Profiling	104
	Overview of Data Profiling	104
	Key Source System Data Profiling Task Steps	105
	Reviewing/Assessing Source Data Quality	109
	Validation Checks to Assess the Data	109
	Key Review/Assess Source Data Quality Task Steps	111

Performing Source\Target Data Mappings	111
Overview of Data Mapping	112
Types of Data Mapping	113
Key Source\Target Data Mapping Task Steps	115
Summary	116
End-of-Chapter Questions	116
<b>Chapter 6 Data Integration Analysis Case Study</b>	<b>117</b>
Case Study Overview	117
Envisioned Wheeler Data Warehouse Environment	118
Aggregations in a Data Warehouse Environment	120
Data Integration Analysis Phase	123
Step 1: Build a Conceptual Data Integration Model	123
Step 2: Perform Source System Data Profiling	124
Step 3: Review/Assess Source Data Quality	130
Step 4: Perform Source\Target Data Mappings	135
Summary	145
<b>Chapter 7 Data Integration Logical Design</b>	<b>147</b>
Determining High-Level Data Volumetrics	147
Extract Sizing	148
Disk Space Sizing	148
File Size Impacts Component Design	150
Key Data Integration Volumetrics Task Steps	150
Establishing a Data Integration Architecture	151
Identifying Data Quality Criteria	154
Examples of Data Quality Criteria from a Target	155
Key Data Quality Criteria Identification Task Steps	155
Creating Logical Data Integration Models	156
Key Logical Data Integration Model Task Steps	157
Defining One-Time Data Conversion Load Logical Design	163
Designing a History Conversion	164
One-Time History Data Conversion Task Steps	166
Summary	166
End-of-Chapter Questions	167
<b>Chapter 8 Data Integration Logical Design Case Study</b>	<b>169</b>
Step 1: Determine High-Level Data Volumetrics	169
Step 2: Establish the Data Integration Architecture	174
Step 3: Identify Data Quality Criteria	177
Step 4: Create Logical Data Integration Models	180
Define the High-Level Logical Data Integration Model	181
Define the Logical Extraction Data Integration Model	183

Define the Logical Data Quality Data Integration Model	187
Define Logical Transform Data Integration Model	190
Define Logical Load Data Integration Model	191
Define Logical Data Mart Data Integration Model	192
Develop the History Conversion Design	195
Summary	198
<b>Chapter 9 Data Integration Physical Design</b>	<b>199</b>
Creating Component-Based Physical Designs	200
Reviewing the Rationale for a Component-Based Design	200
Modularity Design Principles	200
Key Component-Based Physical Designs Creation Task Steps	201
Preparing the DI Development Environment	201
Key Data Integration Development Environment Preparation Task Steps	202
Creating Physical Data Integration Models	203
Point-to-Point Application Development—The Evolution of Data Integration Development	203
The High-Level Logical Data Integration Model in Physical Design	205
Design Physical Common Components Data Integration Models	206
Design Physical Source System Extract Data Integration Models	208
Design Physical Subject Area Load Data Integration Models	209
Designing Parallelism into the Data Integration Models	210
Types of Data Integration Parallel Processing	211
Other Parallel Processing Design Considerations	214
Parallel Processing Pitfalls	215
Key Parallelism Design Task Steps	216
Designing Change Data Capture	216
Append Change Data Capture Design Complexities	217
Key Change Data Capture Design Task Steps	219
Finalizing the History Conversion Design	220
From Hypothesis to Fact	220
Finalize History Data Conversion Design Task Steps	220
Defining Data Integration Operational Requirements	221
Determining a Job Schedule for the Data Integration Jobs	221
Determining a Production Support Team	222
Key Data Integration Operational Requirements Task Steps	224
Designing Data Integration Components for SOA	225
Leveraging Traditional Data Integration Processes as SOA Services	225
Appropriate Data Integration Job Types	227
Key Data Integration Design for SOA Task Steps	227
Summary	228
End-of-Chapter Questions	228



<b>Chapter 10 Data Integration Physical Design Case Study</b>	<b>229</b>
Step 1: Create Physical Data Integration Models	229
Instantiating the Logical Data Integration Models into a Data Integration Package	229
Step 2: Find Opportunities to Tune through Parallel Processing	237
Step 3: Complete Wheeler History Conversion Design	238
Step 4: Define Data Integration Operational Requirements	239
Developing a Job Schedule for Wheeler	240
The Wheeler Monthly Job Schedule	240
The Wheeler Monthly Job Flow	240
Process Step 1: Preparation for the EDW Load Processing	241
Process Step 2: Source System to Subject Area File Processing	242
Process Step 3: Subject Area Files to EDW Load Processing	245
Process Step 4: EDW-to-Product Line Profitability Data Mart Load Processing	248
Production Support Staffing	248
Summary	249
<b>Chapter 11 Data Integration Development Cycle</b>	<b>251</b>
Performing General Data Integration Development Activities	253
Data Integration Development Standards	253
Error-Handling Requirements	255
Naming Standards	255
Key General Development Task Steps	256
Prototyping a Set of Data Integration Functionality	257
The Rationale for Prototyping	257
Benefits of Prototyping	257
Prototyping Example	258
Key Data Integration Prototyping Task Steps	261
Completing/Extending Data Integration Job Code	262
Complete/Extend Common Component Data Integration Jobs	263
Complete/Extend the Source System Extract Data Integration Jobs	264
Complete/Extend the Subject Area Load Data Integration Jobs	265
Performing Data Integration Testing	266
Data Warehousing Testing Overview	267
Types of Data Warehousing Testing	268
Perform Data Warehouse Unit Testing	269
Perform Data Warehouse Integration Testing	272
Perform Data Warehouse System and Performance Testing	273
Perform Data Warehouse User Acceptance Testing	274
The Role of Configuration Management in Data Integration	275
What Is Configuration Management?	276
Data Integration Version Control	277
Data Integration Software Promotion Life Cycle	277
Summary	277
End-of-Chapter Questions	278

<b>Chapter 12 Data Integration Development Cycle Case Study</b>	<b>279</b>
Step 1: Prototype the Common Customer Key	279
Step 2: Develop User Test Cases	283
Domestic OM Source System Extract Job Unit Test Case	284
Summary	287
<b>Part 3 Data Integration with Other Information Management Disciplines</b>	<b>289</b>
<b>Chapter 13 Data Integration and Data Governance</b>	<b>291</b>
What Is Data Governance?	292
Why Is Data Governance Important?	294
Components of Data Governance	295
Foundational Data Governance Processes	295
Data Governance Organizational Structure	298
Data Stewardship Processes	304
Data Governance Functions in Data Warehousing	305
Compliance in Data Governance	309
Data Governance Change Management	310
Summary	311
End-of-Chapter Questions	311
<b>Chapter 14 Metadata</b>	<b>313</b>
What Is Metadata?	313
The Role of Metadata in Data Integration	314
Categories of Metadata	314
Business Metadata	315
Structural Metadata	315
Navigational Metadata	317
Analytic Metadata	318
Operational Metadata	319
Metadata as Part of a Reference Architecture	319
Metadata Users	320
Managing Metadata	321
The Importance of Metadata Management in Data Governance	321
Metadata Environment Current State	322
Metadata Management Plan	322
Metadata Management Life Cycle	324
Summary	327
End-of-Chapter Questions	327

<b>Chapter 15 Data Quality</b>	<b>329</b>
The Data Quality Framework	330
Key Data Quality Elements	331
The Technical Data Quality Dimension	332
The Business-Process Data Quality Dimension	333
Types of Data Quality Processes	334
The Data Quality Life Cycle	334
The Define Phase	336
Defining the Data Quality Scope	336
Identifying/Defining the Data Quality Elements	336
Developing Preventive Data Quality Processes	337
The Audit Phase	345
Developing a Data Quality Measurement Process	346
Developing Data Quality Reports	348
Auditing Data Quality by LOB or Subject Area	350
The Renovate Phase	351
Data Quality Assessment and Remediation Projects	352
Data Quality SWAT Renovation Projects	352
Data Quality Programs	353
Final Thoughts on Data Quality	353
Summary	353
End-of-Chapter Questions	354
<b>Appendix A Exercise Answers</b>	<b>355</b>
<b>Appendix B Data Integration Guiding Principles</b>	<b>369</b>
Write Once, Read Many	369
Grab Everything	369
Data Quality before Transforms	369
Transformation Componentization	370
Where to Perform Aggregations and Calculations	370
Data Integration Environment Volumetric Sizing	370
Subject Area Volumetric Sizing	370
<b>Appendix C Glossary</b>	<b>371</b>
<b>Appendix D Case Study Models</b>	
Appendix D is an online-only appendix. Print-book readers can download the appendix at <a href="http://www.ibmpressbooks.com/title/9780137084937">www.ibmpressbooks.com/title/9780137084937</a> . For eBook editions, the appendix is included in the book.	
<b>Index</b>	<b>375</b>

# Preface

This text provides an overview on data integration and its application in business analytics and data warehousing. As the analysis of data becomes increasingly important and ever more tightly integrated into all aspects of Information Technology and business strategy, the process to combine data from different sources into meaningful information has become its own discipline. The scope of this text is to provide a look at this emerging discipline, its common “blueprint,” its techniques, and its consistent methods of defining, designing, and developing a mature data integration environment that will provide organizations the ability to move high-volume data in ever-decreasing time frames.

## Intended Audience

This text serves many different audiences. It can be used by an experienced data management professional for confirming data integration fundamentals or for college students as a textbook in an upper-level data warehousing college curriculum. The intended audience includes the following:

- Data warehouse program and project managers
- Data warehouse architects
- Data integration architects
- Data integration designers and developers
- Data modeling and database practitioners
- Data management-focused college students

## Scope of the Text

This book stresses the core concepts of how to define, design, and build data integration processes using a common data integration architecture and process modeling technique.

With that goal in mind, *Data Integration Blueprint and Modeling*

- Reviews the types of data integration architectural patterns and their applications
- Provides a data integration architecture blueprint that has been proven in the industry
- Presents a graphical design technique for data integration based on process modeling, data integration modeling
- Covers the Systems Development Life Cycle of data integration
- Emphasizes the importance of data governance in data integration

## Organization of the Text

The text is organized into three parts, including the following:

- **Part 1: Overview of Data Integration**

The first part of this text provides an overview of data integration. Because of the operational and analytic nature of integrating data, the frequency and throughput of the data integration processes have developed into different types of data integration architectural patterns and technologies. Therefore, this part of the text begins with an investigation of the architectural types or patterns of data integration.

Regardless of the type of architecture or supporting technology, there is a common blueprint or reference architecture for the integrating data. One of the core architectural principles in this text is that the blueprint must be able to deal with both operational and analytic data integration types. We will review the processes and approach to the data integration architecture.

The final concept focuses on a graphical process modeling technique for data integration design, based on that reference architecture.

To complete this section, we provide a case study of designing a set of data integration jobs for a banking data warehouse using the Data Integration Modeling Technique.

- **Part 2: The Data Integration Systems Development Life Cycle**

The second part of the text covers the Systems Development Life Cycle (SDLC) of a data integration project in terms of the phases, activities, tasks, and deliverables. It explains how the data integration reference architecture is leveraged as its blueprint, and data integration modeling as the technique to develop the analysis, design, and development deliverables. This section begins the next of a multichapter case study on building an end-to-end data integration application with multiple data integration jobs for the Wheeler Automotive Company, which will require the reader to work through the entire data integration life cycle.

- **Part 3: Data Integration and Other Information Management Disciplines**

The third part of this text discusses data integration in the context of other Information Management disciplines, such as data governance, metadata, and data quality. This part investigates the definition of data governance and its related disciplines of metadata and data quality. It reviews how both the business and IT are responsible for managing data governance and its impact on the discipline of data integration.

For metadata, this part provides an overview of what metadata is, the types of metadata, and which types of metadata are relevant in data integration.

Finally, this part reviews concepts of data quality in terms of the types, approaches to prevent bad data quality, and how to “clean up” existing bad data quality.

- **End-of-Chapter Questions**

Each chapter provides a set of questions on the core concepts in the book to test the reader’s comprehension of the materials. Answers to the questions for each chapter can be found in Appendix A, “Chapter Exercise Answers.”

- **Appendices**

Much of the supporting materials to the text can be found in the appendices, which include the following:

- Appendix A, “Chapter Exercise Answers”—This appendix contains answers to the questions found at the end of each chapter.
- Appendix B, “Data Integration Guiding Principles”—This appendix contains the guiding principles of data integration that were referenced throughout the book.
- Appendix C, “Glossary”—This appendix contains the glossary of terms used in the book.
- Appendix D, “Case Study Models”—This appendix can be found in the eBook versions of this book, or it can be downloaded from the book’s companion Web site ([www.ibmpressbooks.com/title/9780137084937](http://www.ibmpressbooks.com/title/9780137084937)). It contains the detailed data models, entity-attribute reports, subject area file layouts, data mappings, and other artifacts that were created and used throughout the book in the Wheeler case studies.

*This page intentionally left blank*

---

# Introduction: Why Is Data Integration Important?

Today's business organizations are spending tens to *hundreds* of millions of dollars to integrate data for transactional and business intelligence systems at a time when budgets are severely constrained and every dollar of cost counts like never before. There are organizations that have thousands of undocumented point-to-point data integration applications that require significant runtime, CPU, and disk space to maintain and sustain. Consider the cost of an average Information Technology worker at \$100,000; the larger the environment, the more workers are needed to support all these processes. Worse, a majority of these processes are either redundant or no longer needed.

This unprecedented rate of increased cost in data integration is felt especially in those organizations that have grown rapidly through acquisition. It is also observed where there is an absence of corporate-level strategy and operational processes regarding the management and maintenance of corporate data assets. Businesses are relying more heavily on analytic environments to improve their efficiency, maintain market share, and mine data for opportunities to improve revenue and reduce cost.

One of the main reasons for excessive cost within the data integration domain is the absence of a clear, consistent, and effective approach to defining, designing, and building data integration components that lead to a more effective and cost-efficient data integration environment. Having a well-documented environment with fewer data integration processes will ensure that both cost and complexity will be reduced.

The intent of this book is to describe a common data integration approach that can substantially reduce the overall cost of the development and maintenance of an organization's data integration environment and significantly improve data quality over time.



## Data Integration...An Overlooked Discipline

You can go into any bookstore or surf [www.Amazon.com](http://www.Amazon.com) on the Web and you will find volumes of books on Information Management disciplines. Some of these will be data modeling texts that cover all the different types of data modeling techniques from transactional, dimensional, logical, and physical types of models and their purposes in the process of data integration.

There are very few books that cover the architecture, design techniques, and methodology of the Information Management discipline of data integration. Why? Because data integration isn't sexy. The front-end business intelligence applications provide the "cool," colorful, executive dashboards with the multicolored pie and bar charts. Data modeling is a technology focal point for all data-related projects. But the processes or "pipes" that integrate, move, and populate the data have been largely ignored or misunderstood because it is simply hard, tedious, and highly disciplined work.

This emerging discipline has developed from the old programming technologies such as COBOL that moved data with traditional programming design patterns or from database technologies that move data with stored SQL procedures. It is a discipline that is in dire need of the same focus as data modeling, especially because data integration has consistently made up 70% of the costs and risks of all data warehousing and business intelligence projects over the past 15 years.

The cost of maintenance for these data integration environments can be staggering with documented cases of ongoing maintenance cost into the hundreds of millions of dollars. Most data integration environments are poorly documented, with no repeatable method of understanding or clear ability to view the data integration processes or jobs. This leads to unnecessary rework that results in massive redundancy in the number of data integration processes or jobs we see in many organizations. Every unnecessary or duplicative data integration process results in excessive data, increased maintenance, and staff cost, plus the dreaded word, *bad* when it comes to trust in and the measurement of data quality. Anytime an organization has competing data integration processes that perform the same task, it is inevitable that there will be different results, causing the user community to doubt the validity of the data.

As with any engineering discipline, when an organization uses an architecture-specific blueprint, with common processes and techniques to build out and sustain an environment, it reaps the benefits of adhering to that discipline. The benefits are improved quality, lower costs, and sustainability over the long term. Organizations that use a common data integration architecture or blueprint and build and maintain their data integration processes have reaped those benefits.

## Data Integration Fundamentals

Data integration leverages both technical and business processes to combine data into useful information for transactional analytics and/or business intelligence purposes. In the current environment, the volume, velocity, and variety of data are growing at unprecedented levels. Yet most

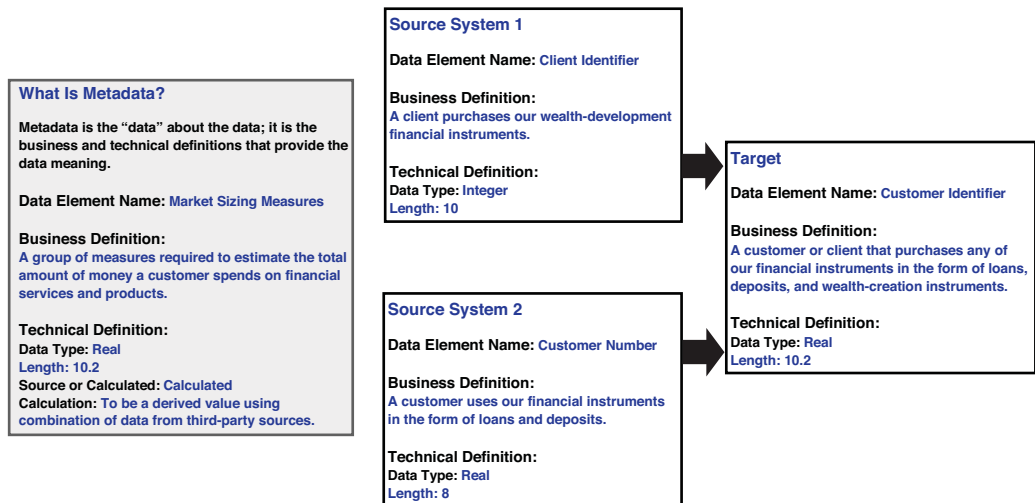
organizations have not changed the approach to how they develop and maintain these data integration processes, which has resulted in expensive maintenance, poor data quality, and a limited ability to support the scope and ever-increasing complexity of transactional data in business intelligence environments.

Data integration is formally defined as the following:

**Data integration** is a set of procedures, techniques, and technologies used to design and build processes that extract, restructure, move, and load data in either operational or analytic data stores either in real time or in batch mode.

## Challenges of Data Integration

Of all the Information Management disciplines, data integration is the most complex. This complexity is a result of having to combine similar data from multiple and distinct source systems into one consistent and common data store for use by the business and technology users. It is this integration of business and technical data that presents the challenge. Although the technical issues of data integration are complex, it is conforming (making the many into one) the business definitions or *metadata* that prove to be the most difficult. One of the key issues that leads to poor data quality is the inability to conform multiple business definitions into one enterprise or canonical definition, as shown in Figure I.1.



**Figure I.1** Example of integrating data into information

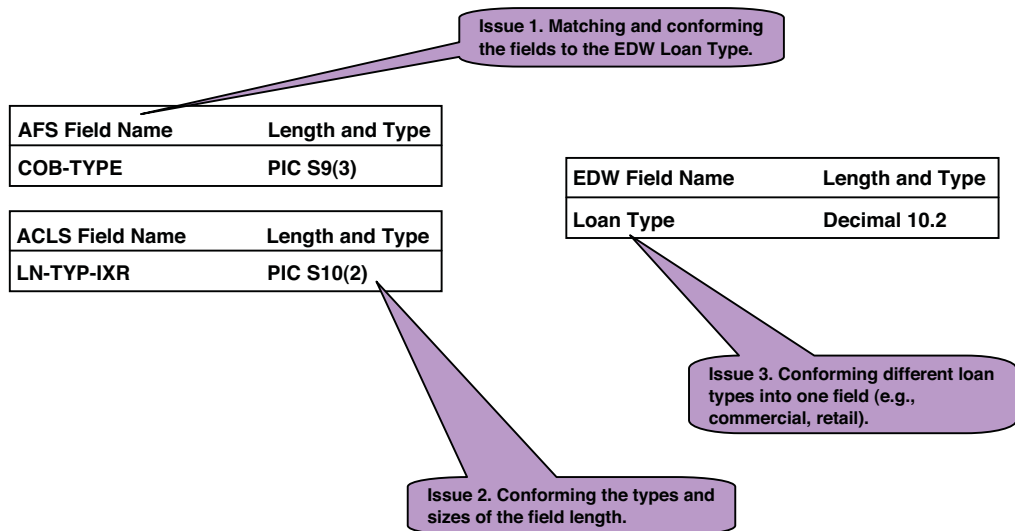
A major function of data integration is to integrate disparate data into a single view of information. An example of a single view of information is the concept of a bank loan.

For a bank (or other financial institution) to have a single view of information, they need to integrate their different types of loans. Most U.S. banks leverage packaged applications from vendors such as AFS for commercial loans and ACLS for retail loans for their loan origination and processing. To provide these banks a holistic view of their loan portfolios, the AFS-formatted loan data and ACLS-formatted loan data need to be conformed into a common and standard format with a universal business definition.

Because the major focus of this text is integrating data for business intelligence environments, the target for this loan type example will be a data warehouse.

For this data warehouse, there is a logical data model complete with a set of entities and attributes, one of which is for the loan entity. One of the attributes, “Loan Type Code” is the unique identifier of the loan type entity. A loan type classifies the valid set of loans, such as commercial loan and retail loan.

Figure I.2 demonstrates the issues caused by the complexity of simply integrating the Loan Type attribute for commercial loans (AFS) and retail loans (ACLS), into a common Loan Type field in the data warehouse.



**Figure I.2** Complexity issues with integrating data

In addition to discussing topics such as conforming technical and business definitions, this book covers core data integration concepts and introduces the reader to new approaches such as data integration modeling. This set of activities will help an institution organize its data integration environments into a set of common processes that will ultimately drive unnecessary cost out of their analytic environments and provide greater information capabilities.

*This page intentionally left blank*

# **A Design Technique: Data Integration Modeling**

This chapter focuses on a new design technique for the analysis and design of data integration processes. This technique uses a graphical process modeling view of data integration similar to the graphical view an entity-relationship diagram provides for data models.

## **The Business Case for a New Design Process**

There is a hypothesis to the issue of massive duplication of data integration processes, which is as follows:

If you do not see a process, you will replicate that process.
---

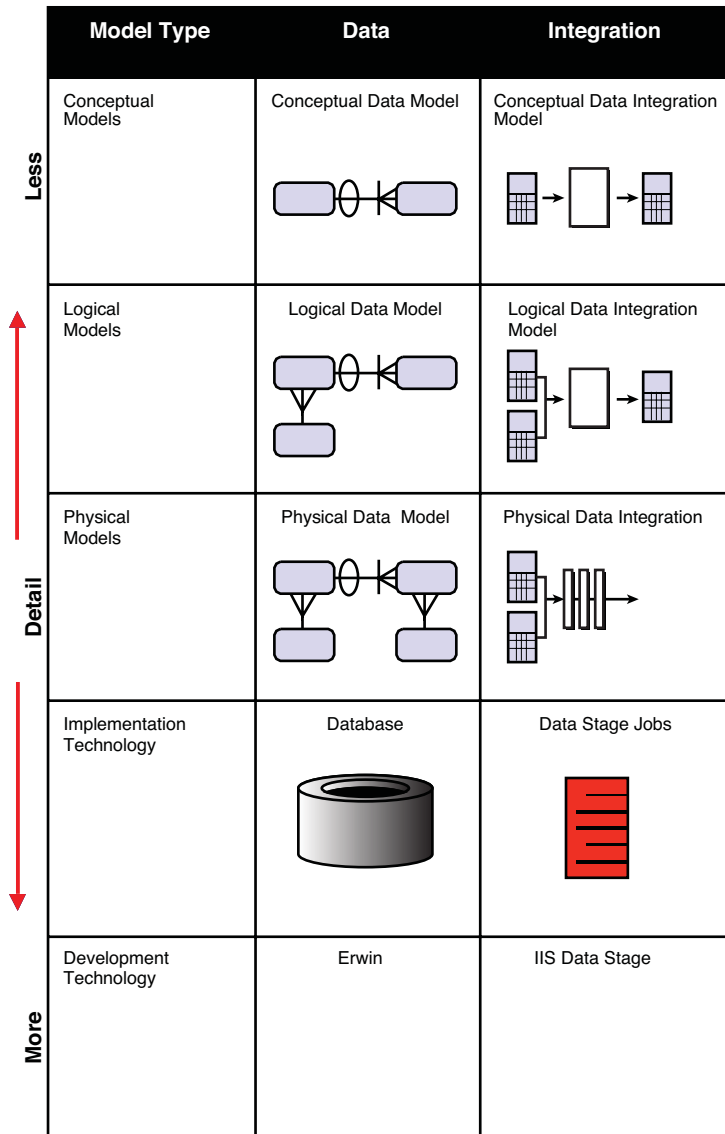
One of the main reasons why there is massive replication of data integration processes in many organizations is the fact that there is no visual method of “seeing” what data integration processes currently exist and what is needed. This is similar to the problem that once plagued the data modeling discipline.

In the early 1980s, many organizations had massive duplication of customer and transactional data. These organizations could not see the “full picture” of their data environment and the massive duplication. Once organizations began to document and leverage entity-relationship diagrams (visual representations of a data model), they were able to see the massive duplication and the degree of reuse of existing tables increased as unnecessary duplication decreased.

The development of data integration processes is similar to those in database development. In developing a database, a blueprint, or model of the business requirements, is necessary to ensure that there is a clear understanding between parties of *what* is needed. In the case of data integration, the data integration designer and the data integration developer need that blueprint or project artifact to ensure that the business requirements in terms of sources, transformations, and

targets that are needed to move data have been clearly communicated via a common, consistent approach. The use of a process model specifically designed for data integration will accomplish that requirement.

Figure 3.1 depicts the types of data models needed in a project and how they are similar to those that could be developed for data integration.



**Figure 3.1** Modeling paradigm: data and data integration

The usual approach for analyzing, designing, and building ETL or data integration processes on most projects involves a data analyst documenting the requirements for source-to-target mapping in Microsoft® Excel® spreadsheets. These spreadsheets are given to an ETL developer for the design and development of maps, graphs, and/or source code.

Documenting integration requirements from source systems and targets manually into a tool like Excel and then mapping them again into an ETL or data integration package has been proven to be time-consuming and prone to error. For example:

- **Lost time**—It takes a considerable amount of time to copy source metadata from source systems into an Excel spreadsheet. The same source information must then be rekeyed into an ETL tool. This source and target metadata captured in Excel is largely non-reusable unless a highly manual review and maintenance process is instituted.
- **Nonvalue add analysis**—Capturing source-to-target mappings with transformation requirements contains valuable navigational metadata that can be used for data lineage analysis. Capturing this information in an Excel spreadsheet does not provide a clean automated method of capturing this valuable information.
- **Mapping errors**—Despite our best efforts, manual data entry often results in incorrect entries, for example, incorrectly documenting an INT data type as a VARCHAR in an Excel spreadsheet will require a data integration designer time to analyze and correct.
- **Lack of standardization: inconsistent levels of detail**—The data analysts who perform the source-to-target mappings have a tendency to capture source/transform/target requirements at different levels of completeness depending on the skill and experience of the analyst. When there are inconsistencies in the level of detail in the requirements and design of the data integration processes, there can be misinterpretations by the development staff in the source-to-target mapping documents (usually Excel), which often results in coding errors and lost time.
- **Lack of standardization: inconsistent file formats**—Most environments have multiple extracts in different file formats. The focus and direction must be toward the concept of *read once, write many*, with consistency in extract, data quality, transformation, and load formats. The lack of a standardized set of extracts is both a lack of technique and often a result of a lack of visualization of what is in the environment.

To improve the design and development efficiencies of data integration processes, in terms of time, consistency, quality, and reusability, a graphical process modeling design technique for data integration with the same rigor that is used in developing data models is needed.

## Improving the Development Process

Process modeling is a tried and proven approach that works well with Information Technology applications such as data integration. By applying a process modeling technique to data integration, both the visualization and standardization issues will be addressed. First, let's review the types of process modeling.

## Leveraging Process Modeling for Data Integration

Process modeling is a means of representing the interrelated processes of a system at any level of detail, using specific types of diagrams that show the flow of data through a series of processes. Process modeling techniques are used to represent specific processes graphically for clearer understanding, communication, and refinement between the stakeholders that design and develop system processes.

Process modeling unlike data modeling has several different types of process models based on the different types of process interactions. These different model types include process dependency diagrams, structure hierarchy charts, and data flow diagrams. Data flow diagramming, which is one of the best known of these process model types, is further refined into several different types of data flow diagrams, such as context diagrams, Level 0 and Level 1 diagrams and “leaf-level” diagrams that represent different levels and types of process and data flow.

By leveraging the concepts of different levels and types of process modeling, we have developed a processing modeling approach for data integration processes, which is as follows:

*Data integration modeling* is a process modeling technique that is focused on engineering data integration processes into a common data integration architecture.

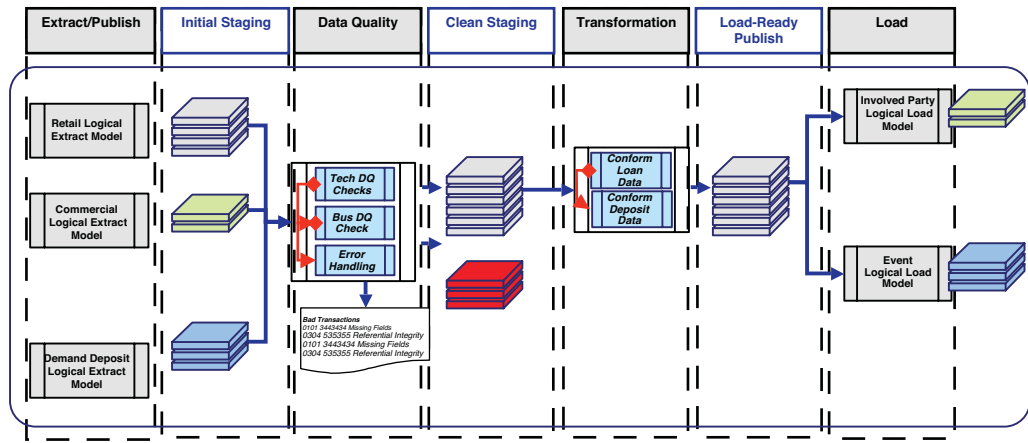
## Overview of Data Integration Modeling

Data integration modeling is a technique that takes into account the types of models needed based on the types of architectural requirements for data integration and the types of models needed based on the Systems Development Life Cycle (SDLC).

## Modeling to the Data Integration Architecture

The types of process models or data integration models are dependent on the types of processing needed in the data integration reference architecture. By using the reference architecture as a framework, we are able to create specific process model types for the discrete data integration processes and landing zones, as demonstrated in Figure 3.2.





**Figure 3.2** Designing models to the architecture

Together, these discrete data integration layers become process model types that form a complete data integration process. The objective is to develop a technique that will lead the designer to model data integration processes based on a common set of process types.

## Data Integration Models within the SDLC

Data integration models follow the same level of requirement and design abstraction refinement that occurs within data models during the SDLC. Just as there are conceptual, logical, and physical data models, there are conceptual, logical, and physical data integration requirements that need to be captured at different points in the SDLC, which could be represented in a process model.

The following are brief descriptions of each of the model types. A more thorough definition along with roles, steps, and model examples is reviewed later in the chapter.

- **Conceptual data integration model definition**—Produces an implementation-free representation of the data integration requirements for the proposed system that will serve as a basis for determining how they are to be satisfied.
- **Logical data integration model definition**—Produces a detailed representation of the data integration requirements at the data set (entity/table) level, which details the transformation rules and target logical data sets (entity/tables). These models are still considered to be technology-independent.

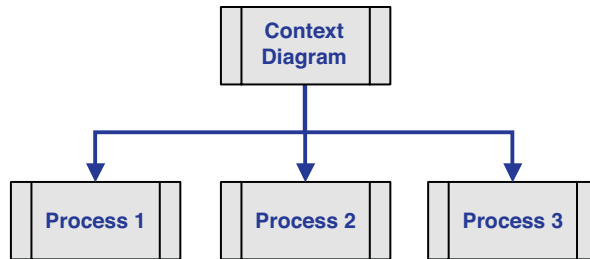
The focus at the logical level is on the capture of actual source tables and proposed target stores.

- **Physical data integration model definition**—Produces a detailed representation of the data integration specifications at the component level. They should be represented in terms of the component-based approach and be able to represent how the data will optimally flow through the data integration environment in the selected development technology.

## Structuring Models on the Reference Architecture

Structuring data models to a Systems Development Life Cycle is a relatively easy process. There is usually only one logical model for a conceptual data model and there is only one physical data model for a logical data model. Even though entities may be decomposed or normalized within a model, there is rarely a need to break a data model into separate models.

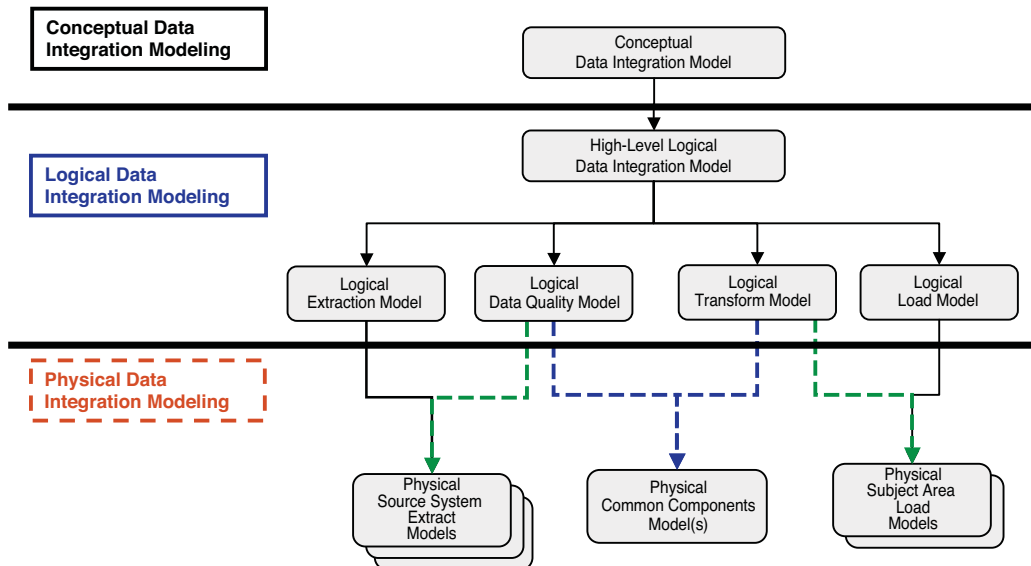
Process models have traditionally been decomposed further down into separate discrete functions. For example, in Figure 3.3, the data flow diagram's top process is the context diagram, which is further decomposed into separate functional models.



**Figure 3.3** A traditional process model: data flow diagram

Data integration models are decomposed into functional models as well, based on the data integration reference architecture and the phase of the Systems Development Life Cycle.

Figure 3.4 portrays how conceptual, logical, and physical data integration models are broken down.



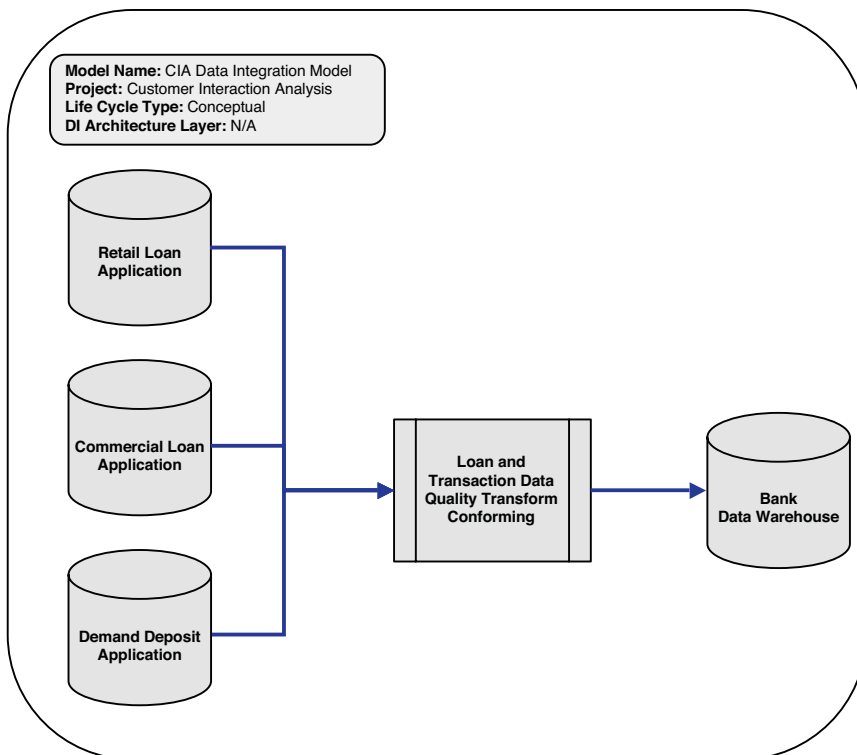
**Figure 3.4** Data integration models by the Systems Development Life Cycle

## Conceptual Data Integration Models

A conceptual data integration model is an implementation-free representation of the data integration requirements for the proposed system that will serve as a basis for “scoping” how they are to be satisfied and for project planning purposes in terms of source systems analysis, tasks and duration, and resources.

At this stage, it is only necessary to identify the major conceptual processes to fully understand the users’ requirements for data integration and plan the next phase.

Figure 3.5 provides an example of a conceptual data integration model.



**Figure 3.5** Conceptual data integration model example

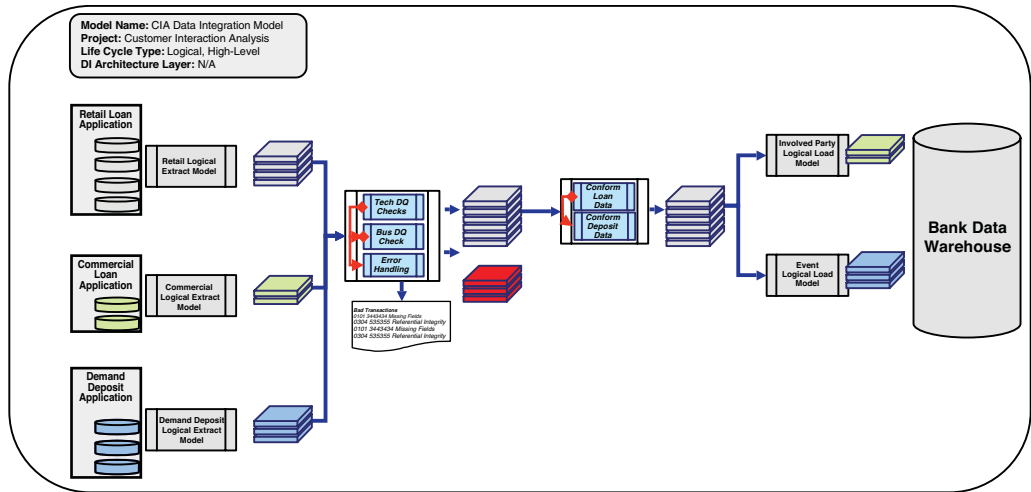
## Logical Data Integration Models

A logical data integration model produces a set of detailed representations of the data integration requirements that captures the first-cut source mappings, business rules, and target data sets (table/file). These models portray the logical extract, data quality, transform, and load requirements for the intended data integration application. These models are still considered to be technology-independent. The following sections discuss the various logical data integration models.

## High-Level Logical Data Integration Model

A high-level logical data integration model defines the scope and the boundaries for the project and the system, usually derived and augmented from the conceptual data integration model. A high-level data integration diagram provides the same guidelines as a context diagram does for a data flow diagram.

The high-level logical data integration model in Figure 3.6 provides the structure for *what* will be needed for the data integration system, as well as provides the outline for the logical models, such as extract, data quality, transform, and load components.



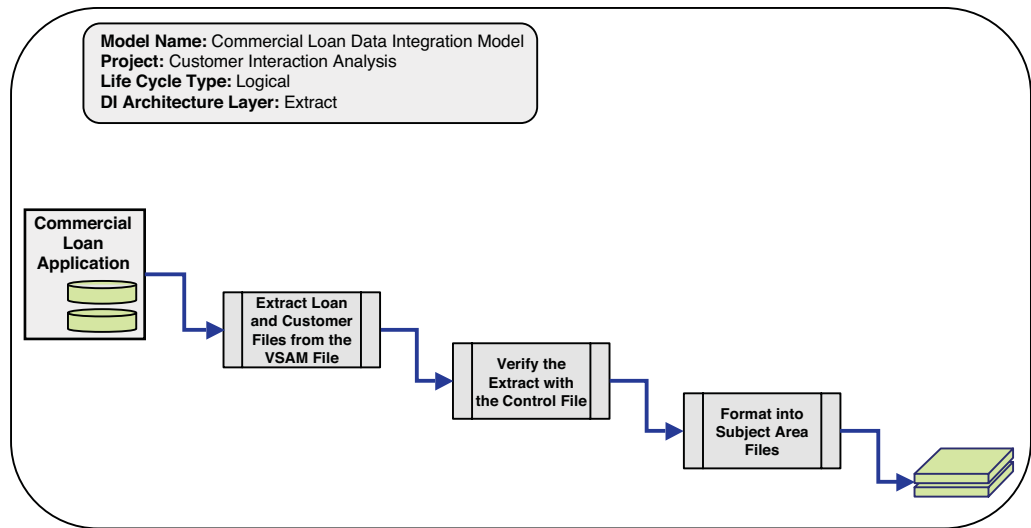
**Figure 3.6** Logical high-level data integration model example

## Logical Extraction Data Integration Models

The logical extraction data integration model determines what subject areas will need to be extracted from sources, such as *what* applications, databases, flat files, and unstructured sources.

Source file formats should be mapped to the attribute/column/field level. Once extracted, source data files should be loaded by default to the initial staging area.

Figure 3.7 depicts a logical extraction model.



**Figure 3.7** Logical extraction data integration model example

Extract data integration models consist of two discrete sub processes or components:

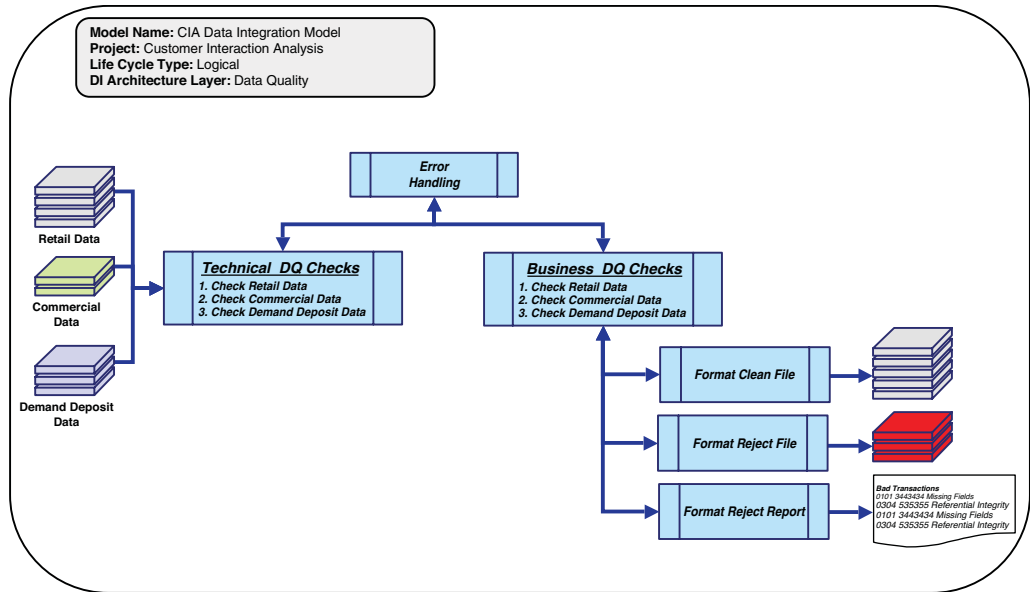
- **Getting the data out of the source system**—Whether the data is actually extracted from the source system or captured from a message queue or flat file, the network connectivity to the source must be determined, the number of tables/files must be reviewed, and the files to extract and in what order to extract them in must be determined.
- **Formatting the data to a subject area file**—As discussed in Chapter 2, “An Architecture for Data Integration,” subject area files provide a layer of encapsulation from the source to the final target area. The second major component of an extract data integration model is to rationalize the data from the source format to a common subject area file format, for example mapping a set of Siebel Customer Relationship Management Software tables to a customer subject area file.

## Logical Data Quality Data Integration Models

The logical data quality data integration model contains the business and technical data quality checkpoints for the intended data integration process, as demonstrated in Figure 3.8.

Regardless of the technical or business data quality requirements, each data quality data integration model should contain the ability to produce a clean file, reject file, and reject report that would be instantiated in a selected data integration technology.

Also the error handling for the entire data integration process should be designed as a reusable component.



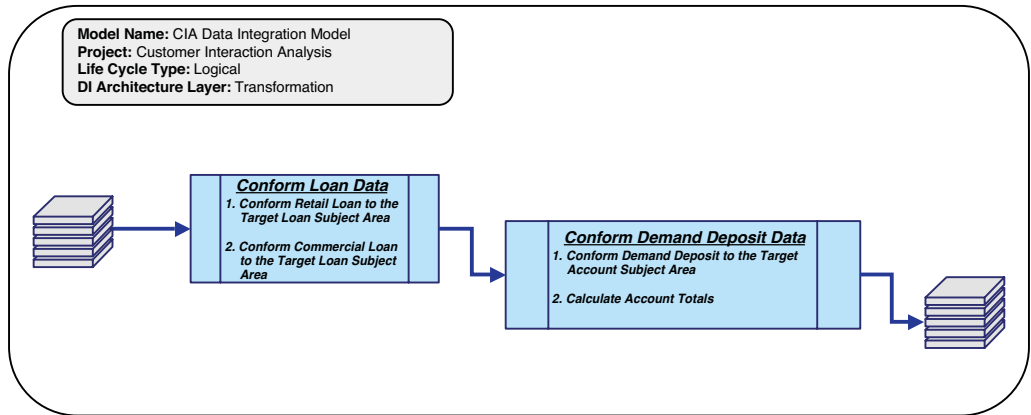
**Figure 3.8** Logical data quality data integration model example

As discussed in the data quality architectural process in Chapter 2, a clear data quality process will produce a clean file, reject file, and reject report. Based on an organization's data governance procedures, the reject file can be leveraged for manual or automatic reprocessing.

### Logical Transform Data Integration Models

The logical transform data integration model identifies at a logical level what transformations (in terms of calculations, splits, processing, and enrichment) are needed to be performed on the extracted data to meet the business intelligence requirements in terms of aggregation, calculation, and structure, which is demonstrated in Figure 3.9.

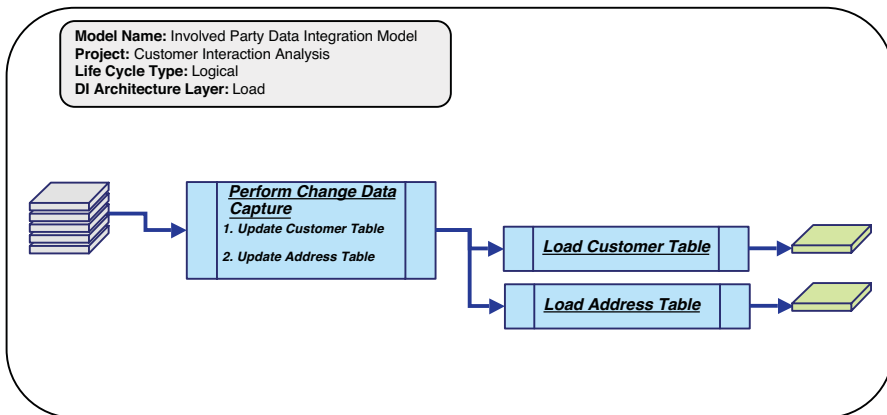
Transform types as defined in the transformation processes are determined on the business requirements for conforming, calculating, and aggregating data into enterprise information, as discussed in the transformation architectural process in Chapter 2.



**Figure 3.9** Logical transformation data integration model example

### Logical Load Data Integration Models

Logical load data integration models determine at a logical level what is needed to load the transformed and cleansed data into the target data repositories by subject area, which is portrayed in Figure 3.10.



**Figure 3.10** Logical load data integration model example

Designing load processes by target and the subject areas within the defined target databases allows sub-processes to be defined, which further encapsulates changes in the target from source data, preventing significant maintenance. An example is when changes to the physical database schema occur, only the subject area load job needs to change, with little impact to the extract and transform processes.

## Physical Data Integration Models

The purpose of a physical data integration model is to produce a detailed representation of the data integration specifications at the component level *within* the targeted data integration technology.

A major concept in physical data integration modeling is determining how to best take the logical design and apply design techniques that will optimize performance.

## Converting Logical Data Integration Models to Physical Data Integration Models

As in data modeling where there is a transition from logical to physical data models, the same transition occurs in data integration modeling. Logical data integration modeling determines *what* extracts, data quality, transformations, and loads. Physical data integration leverages a target-based design technique, which provides guidelines on how to design the “hows” in the physical data integration models to ensure that the various components will perform optimally in a data integration environment.

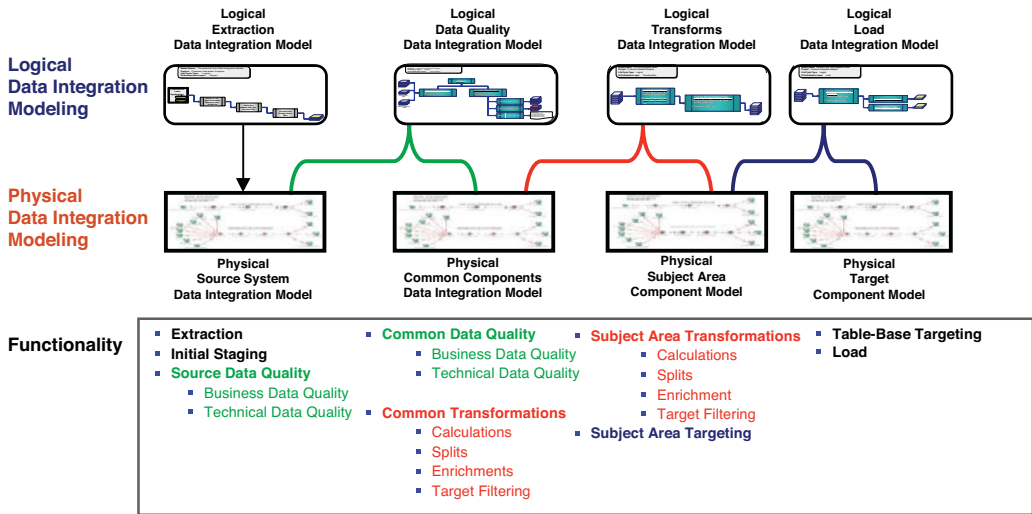
## Target-Based Data Integration Design Technique Overview

The target-based data integration design technique is an approach that creates physical data integration components based on the subject area loads and the source systems that populate those subject areas. It groups logical functionality into reusable components based on the data movement patterns of local versus enterprise usage within each data integration model type.

For example, in most data integration processes, there are source system-level and enterprise-level data quality checks. The target-based technique places that functionality either close to the process that will use it (in this case, the extract process) or groups enterprise capabilities in common component models.

For example, for source system-specific data quality checks, the target-based technique simply moves that logic to the extract processes while local transformations are moved to load processes and while grouping enterprise-level data quality and transformations are grouped at the common component level. This is displayed in Figure 3.11.





**Figure 3.11** Distributing logical functionality between the “whats” and “hows”

The target-based data integration design technique is not a new concept: Coupling and cohesion, modularity, objects, and components are all techniques used to group “stuff” into understandable and highly functional units of work. The target-based technique is simply a method of modularizing core functionality within the data integration models.

## Physical Source System Data Integration Models

A source system extract data integration model extracts the data from a source system, performs source system data quality checks, and then conforms that data into the specific subject area file formats, as shown in Figure 3.12.

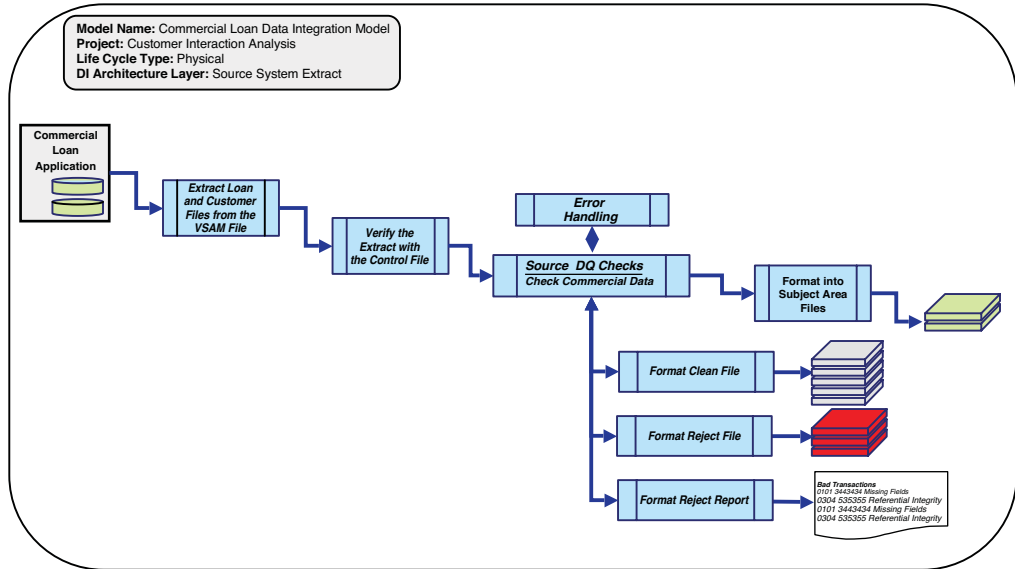
The major difference in a logical extract model from a physical source system data integration model is a focus on the final design considerations needed to extract data from the specified source system.

## Designing an Extract Verification Process

The data from the source system files is extracted and verified with a *control file*. A control file is a data quality check that verifies the number of rows of data and a control total (such as loan amounts that are totaled for verification for a specific source extract as an example).

It is here where data quality rules that are source system-specific are applied. The rationale for applying source system-specific data quality rules at the particular source system rather than in one overall data quality job is to facilitate maintenance and performance. One giant data quality job becomes a maintenance nightmare. It also requires an unnecessary amount of system memory to load all data quality processes and variables that will slow the time for overall job processing.

Cross-system dependencies should be processed in this model. For example, associative relationships for connecting agreements together should be processed here.



**Figure 3.12** Physical source system extract data integration model example

## Physical Common Component Data Integration Models

The physical common component data integration model contains the enterprise-level business data quality rules and common transformations that will be leveraged by multiple data integration applications. This layer of the architecture is a critical focal point for reusability in the overall data integration process flow, with particular emphasis on leveraging existing transformation components. Any new components must meet the criteria for reusability.

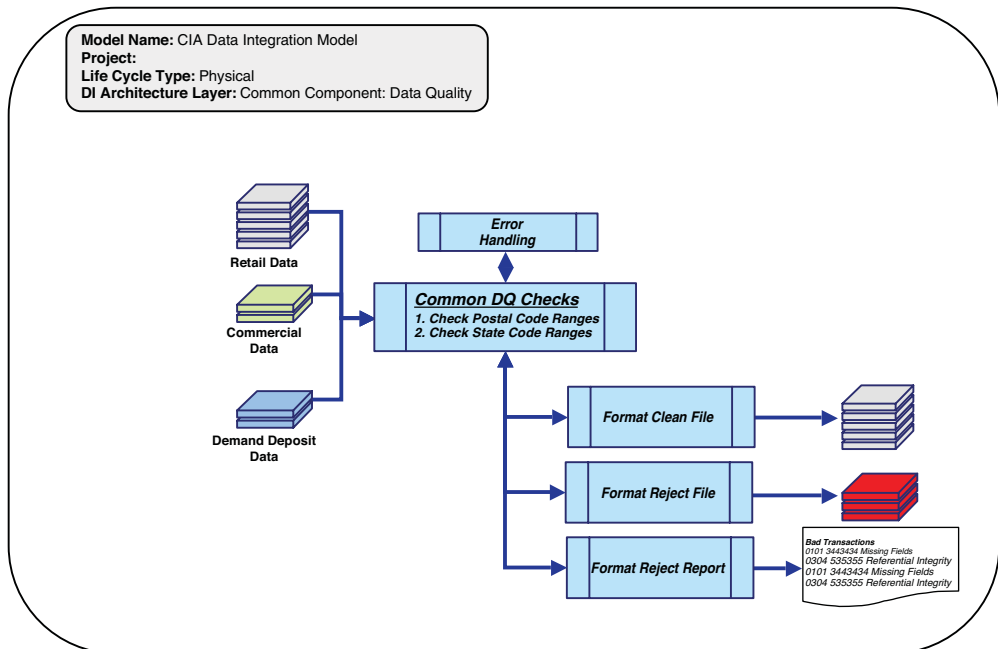
Finally, in designing common component data integration models, the process flow is examined on where parallelism can be built in to the design based on expected data volumes and within the constraints of the current data integration technology.

## Common Component Data Quality Data Integration Models

Common component data quality data integration models are generally very “thin” (less functionality) process models, with enterprise-level data quality rules. Generally, source system-specific data quality rules are technical in nature, whereas business data quality rules tend to be applied at the enterprise level.

For example, gender or postal codes are considered business rules that can be applied as data quality rules against all data being processed. Figure 3.13 illustrates an example of a common data quality data integration model.

Note that the source-specific data quality rules have been moved to the physical source system extract data integration model and a thinner data quality process is at the common component level. Less data ensures that the data flow is not unnecessarily constrained and overall processing performance will be improved.

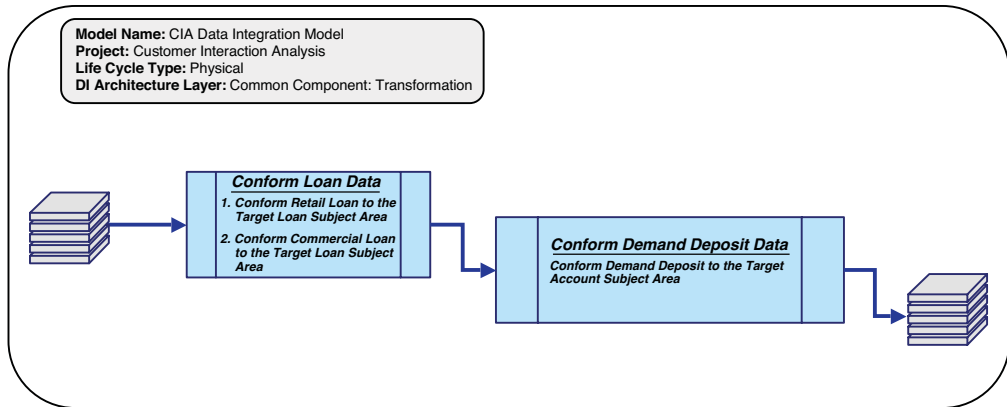


**Figure 3.13** Common components—data quality data integration model example

### Common Component Transformation Data Integration Models

Most common transforms are those that conform data to an enterprise data model. Transformations needed for specific aggregations and calculations are moved to the subject area loads, or where they are needed, which is in the subject areas that the data is being transformed.

In terms of enterprise-level aggregations and calculations, there are usually very few; most transformations are subject-area-specific. An example of a common component-transformation data integration subject area model is depicted in Figure 3.14.



**Figure 3.14** Common components—transform data integration model example

Please note that the aggregations for the demand deposit layer have been removed from the common component model and have been moved to the subject area load in line with the concept of moving functionality to where it is needed.

### Physical Subject Area Load Data Integration Models

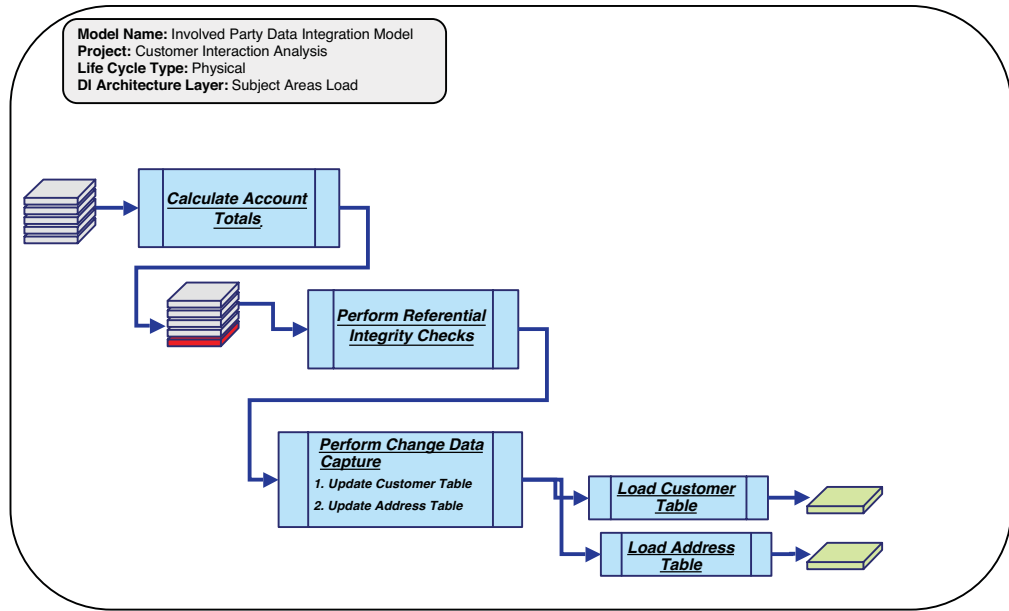
A subject area load data integration model logically groups “target tables” together based on subject area (grouping of targets) dependencies and serves as a simplification for source system processing (layer of indirection).

A subject area load data integration model performs the following functions:

- **Loads data**
- **Refreshes** snapshot loads
- **Performs Change Data Capture**

It is in the subject area load data integration models where primary and foreign keys will be generated, referential integrity is confirmed, and Change Data Capture is processed.

In addition to the simplicity of grouping data by subject area for understandability and maintenance, grouping data by subject area logically limits the amount of data carried per process because it is important to carry as little data as possible through these processes to minimize performance issues. An example of a physical data integration subject area model is shown in Figure 3.15.



**Figure 3.15** Physical subject data area load data integration model example

## Logical Versus Physical Data Integration Models

One question that always arises in these efforts is, “Is there a need to have one set of logical data integration models and another set of physical data integration models?”

The answer for data integration models is the same as for data models, “It depends.” It depends on the maturity of the data management organization that will create, manage, and own the models in terms of their management of metadata, and it depends on other data management artifacts (such as logical and physical data models).

## Tools for Developing Data Integration Models

One of the first questions about data integration modeling is, “What do you build them in?” Although diagramming tools such as Microsoft Visio® and even Microsoft PowerPoint® can be used (as displayed throughout the book), we advocate the use of one of the commercial data integration packages to design and build data integration models.

Diagramming tools such as Visio require manual creation and maintenance to ensure that they are kept in sync with source code and Excel spreadsheets. The overhead of the maintenance often outweighs the benefit of the manually created models. By using a data integration package, existing data integration designs (e.g., an extract data integration model) can be reviewed for potential reuse in other data integration models, and when leveraged, the maintenance to the actual data integration job is performed when the model is updated. Also by using a data integration

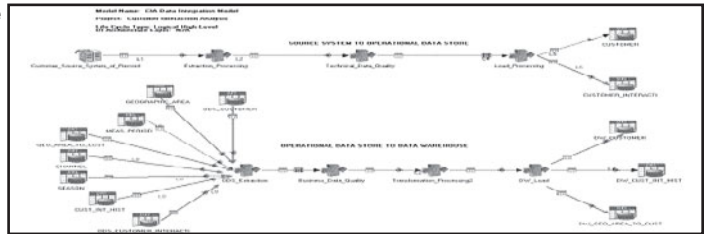
package such as Ab Initio, IBM Data Stage®, or Informatica to create data integration models, an organization will further leverage the investment in technology it has.

Figure 3.16 provides examples of high-level logical data integration models built in Ab Initio, IBM Data Stage, and Informatica.

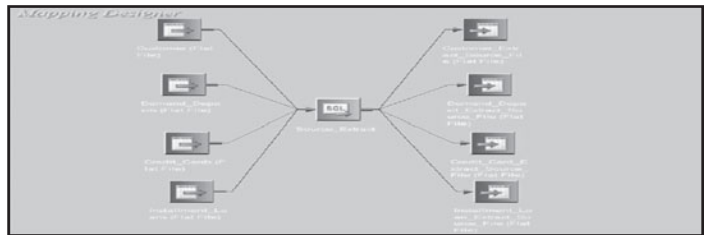
#### Ab Initio



#### IBM Data Stage



#### Informatica



**Figure 3.16** Data integration models by technology

Experience in using data integration packages for data integration modeling has shown that data integration projects and Centers of Excellence have seen the benefits of increased extract, transform and load code standardization, and quality. Key benefits from leveraging a data integration package include the following:

- **End-to-end communications**—Using a data integration package facilitates faster transfer of requirements from a data integration designer to a data integration developer by using the same common data integration metadata. Moving from a logical design to a physical design using the same metadata in the same package speeds up the transfer process and cuts down on transfer issues and errors. For example, source-to-target data definitions and mapping rules do not have to be transferred between technologies,

thereby reducing mapping errors. This same benefit has been found in data modeling tools that transition from logical data models to physical data models.

- **Development of leveragable enterprise models**—Capturing data integration requirements as logical and physical data integration models provides an organization an opportunity to combine these data integration models into enterprise data integration models, which further matures the Information Management environment and increases overall reuse. It also provides the ability to reuse source extracts, target data loads, and common transformations that are in the data integration software package’s metadata engine. These physical data integration jobs are stored in the same metadata engine and can be linked to each other. They can also be linked to other existing metadata objects such as logical data models and business functions.
- **Capture of navigational metadata earlier in the process**—By storing logical and physical data integration model metadata in a data integration software package, an organization is provided with the ability to perform a more thorough impact analysis of a single source or target job. The capture of source-to-target mapping metadata with transformation requirements earlier in the process also increases the probability of catching mapping errors in unit and systems testing. In addition, because metadata capture is automated, it is more likely to be captured and managed.

## Industry-Based Data Integration Models

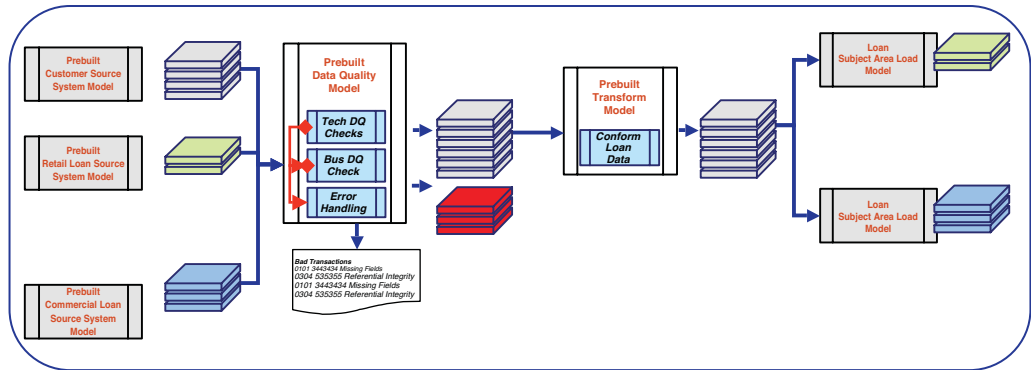
To reduce risk and expedite design efforts in data warehousing projects, prebuilt data models for data warehousing have been developed by IBM, Oracle, Microsoft, and Teradata.

As the concept of data integration modeling has matured, prebuilt data integration models are being developed in support of those industry data warehouse data models.

Prebuilt data integration models use the industry data warehouse models as the targets and known commercial source systems for extracts. Having industry-based source systems and targets, it is easy to develop data integration models with prebuilt source-to-target mappings. For example, in banking, there are common source systems, such as the following:

- **Commercial and** retail loan systems
- **Demand** deposit systems
- **Enterprise** resource systems such as SAP and Oracle

These known applications can be premapped to the industry-based data warehouse data models. Based on actual project experience, the use of industry-based data integration models can significantly cut the time and cost of a data integration project. An example of an industry-based data integration model is illustrated in Figure 3.17.



**Figure 3.17** Industry-based data integration model example

In the preceding example, the industry data integration model provides the following:

- Prebuilt extract processes from the customer, retail loan, and commercial loan systems
- Prebuilt data quality processes based on known data quality requirements in the target data model
- Prebuilt load processes based on the target data model subject areas

Starting with existing designs based on a known data integration architecture, source systems, and target data models, provides a framework for accelerating the development of a data integration application.

## Summary

Data modeling is a graphical design technique for data. In data integration, data integration modeling is a technique for designing data integration processes using a graphical process modeling technique against the data integration reference architecture.

This chapter detailed the types of data integration models—conceptual, logical, and physical—and the approach for subdividing the models based on the process layers of the data integration reference architecture. This chapter also provided examples of each of the different logical and physical data integration model types.

It covered the transition from logical data integration models to physical data integration models, which might be better stated as how to move from the “whats” to the “hows.”

Finally, the chapter discussed how this maturing technique can be used to create prebuilt, industry-based data integration models.

The next chapter is a case study for a bank that is building a set of data integration processes and uses data integration modeling to design the planned data integration jobs.



## End-of-Chapter Questions

**Question 1.**

Data integration modeling is based on what other modeling paradigm?

**Question 2.**

List and describe the types of logical data integration models.

**Question 3.**

List and describe the types of physical data integration models.

**Question 4.**

Using the target-based design technique, document where the logical data quality logic is moved to and why in the physical data integration model layers.

**Question 5.**

Using the target-based design technique, document where the logical transformation logic is moved to and why in the physical data integration model layers.

*This page intentionally left blank*

---

# Index

## A

absolute data quality  
  checkpoints, data integration modeling case study, 80  
accurate dimension (data quality), 332  
administration of metadata repositories, 324-325  
aggregation transformations, 37  
  in data warehouses, 120-122  
  defined, 373  
  where to perform, 370  
analysis. *See* data integration analysis  
analytic metadata, 318  
analytics layer (data warehouses)  
  aggregations in, 121-122  
  unit testing, 271-272  
Append Change Data Capture approach in physical design phase, 217-219  
application development cycle, data integration development cycle versus, 251-252  
architectural patterns  
  common functionality in, 15-16

EAI (Enterprise Application Integration), 8-9  
ETL (Extract, Transform, Load), 14-15  
federation, 12-13  
layers of, 26-27  
within overall architecture, 41-42  
physical load architectures, 41  
reference architecture  
  data integration modeling to, 48-49  
  defined, 19-20  
  modularity of, 22-24  
  objectives of, 21-22  
  purposes of, 26  
  scalability of, 24-25  
  structuring models on, 50  
SOA (Service-Oriented Architecture), 9-12  
assessing  
  data quality, 352  
  source data quality, 109-111, 130-134

audit phase (data quality life cycle), 335, 345-351  
  data quality measurement process, developing, 346-348  
  data quality reports, developing, 348-350  
  direct audits, 351  
  ongoing processing, 351

## B

best practices for data governance policies, 294  
build phase. *See* development cycle phase  
building metadata management repositories versus buying, 323-324  
business, relationship with Information Technology, 293  
business analytics centers of excellence, 302-303  
business case for data integration modeling, 45-47

- business data quality
    - checkpoints, 32
    - data integration modeling
      - case study, 77-80
      - packaging into common component model, 92-94
  - business extraction rules, 74
  - business intelligence
    - defined, 371
    - real-time analysis of, 12
  - business intelligence data
    - integration, 8
  - business metadata, 315
  - business users of metadata, 320
  - business-driven poor data quality, 32
  - business-process data quality
    - dimensions, 333-334
  - buying metadata management
    - repositories versus building, 323-324
- C**
- calculation transformations, 35-36
    - in data warehouses, 120-122
    - defined, 372
  - capturing metadata, 325-326
  - case studies
    - data integration analysis
      - conceptual data
        - integration model, building, 117-123
      - overview, 117-123
      - source data quality, assessing, 130-134
      - source system data profiling, 124-130
      - source/target data mappings, 135-144
    - data integration modeling
      - common component data
        - integration models, developing, 92-94
      - conceptual data
        - integration model, building, 69
    - high-level logical data
      - integration model, building, 70-72
    - logical data quality data
      - integration models, defining, 76-80
    - logical extraction data
      - integration models, building, 72-76
    - logical extraction data
      - integration models, converting to physical models, 88-90
    - logical load data
      - integration models, converting to physical models, 90-92
    - logical load data
      - integration models, defining, 85-86
    - logical transform data
      - integration models, defining, 81-85
    - overview, 67-69
    - physical data integration
      - modeling, converting logical models to, 88-92
    - physical data integration
      - modeling, determining strategy, 87
    - physical data integration
      - modeling, sequencing, 94-95
  - development cycle phase
    - prototyping, 279-283
    - unit testing, 283-287
  - logical design phase
    - data integration
      - architecture, establishing, 174-177
    - data quality criteria, identifying, 177-180
    - history data conversion, 195-197
    - logical data integration
      - models, creating, 180-197
    - source system
      - volumetrics, 169-174
  - physical design phase
    - history data conversion, 238-239
    - operational requirements, 239-240
    - parallel processing, 237-238
    - physical common
      - component data
        - integration models, designing, 230-232
    - physical data integration
      - models, creating, 229-236
    - physical data mart data
      - integration models, designing, 236
    - physical source system
      - data integration models, designing, 232-234
    - physical subject area load
      - data integration models, designing, 234-236
    - production support team, 248
    - scheduling data
      - integration jobs, 240-248
  - categories of metadata, 314-319
    - analytic metadata, 318
    - business metadata, 315
    - navigational metadata, 317-318
    - operational metadata, 319
    - structural metadata, 315-316
  - Change Data Capture (CDC), 38, 216-220
  - change management in data
    - governance, 310-311
  - chief data officers, 300
  - clean staging landing zone, 34, 372
  - coarse-grained SOA objects, 227
  - cohesion, coupling versus, 200-201
  - column analysis, 107-108
  - column metrics, 346

commenting in data integration jobs, 254

common component data integration models, 58-60
 

- completing code for, 263-264
- data integration modeling case study, 92-94

complete dimension (data quality), 332

complexity
 

- of data integration, 3-4
- of EAI (Enterprise Application Integration), 8-9
- of ETL (Extract, Transform, Load), 14-15
- of federation, 13
- of SOA (Service-Oriented Architecture), 11

compliance in data governance, 309

component-based physical designs
 

- creating, 200-201
- point-to-point application development versus, 203-205

conceptual data integration modeling, 51
 

- building model, 101-104
- data integration analysis case study, 117-123
- data integration modeling case study, 69
- defined, 49, 374

configuration management, 275-277
 

- Software Promotion Life Cycle (SPLC), 277
- version control, 277

confirming subject areas, 73

conforming transformations, 35

consistency measures of data quality, 347

consistent dimension (data quality), 332

constraints, 342

control file check processing, 74

converting logical data integration models to physical data integration models, 56, 203-210, 229-236

Core Data Elements List, 106

cost of data integration 1, 2, 22

coupling, cohesion versus, 200-201

cross-domain analysis, 108

current state inventory in metadata management, 322

**D**

data conversion in logical design phase, 163-166, 195-197

data discovery, source system
 

- data profiling, 104-108
- difficulty of, 103-104

data governance, 291-294
 

- change management, 310-311
- compliance in, 309
- data stewardship processes, 304-305
- in data warehousing, 305-309
  - defined, 292
- foundational processes, 294
  - best practices, 294
  - policy examples, 294
  - sample mission statement, 294
- importance of, 294
- metadata management, importance of, 321
- organizational structure, 294-304
  - business analytics centers of excellence, 302-303
- chief data officers, 300
- Data Governance Office (DGO), 300
- data quality audit and renovation teams, 300-301
- data stewardship community, 303-304
- data-related programs and projects, 302

Executive Data Governance Committee, 300
 

- relationship between business and Information Technology, 293
- responsibilities for, 293

Data Governance Office (DGO), 300

data integration
 

- architectural patterns
  - common functionality in, 15-16
- EAI (Enterprise Application Integration), 8-9
- ETL (Extract, Transform, Load), 14-15
- federation, 12-13
- layers of, 26-27
  - within overall architecture, 41-42
- reference architecture, 19-26
- SOA (Service-Oriented Architecture), 9-12
- benefits of, 2
- complexity of, 3-4
- cost of, 1, 2, 22
- data governance and. *See* data governance
- data modeling versus, 2
- data quality tasks in, 339-341
- defined, 3
- development cycle phase. *See* development cycle phase
- guiding principles
  - data quality, checking before transformations, 369
  - “grab everything,” 369
  - “write once, read many,” 369
- landing zones
  - clean staging landing zone, 34

- initial staging landing zone, 29-31
- load-ready publish landing zone, 39-40
- logical design phase. *See* logical design phase
- metadata, role of, 314
- physical design phase. *See* physical design phase
- process modeling, types of, 48
- processes
  - data quality processes, 31-34
  - extract/subscribe processes, 27-29
  - load/publish processes, 40-41
  - transformations, 35-39
- types of, 8
- volumetric sizing, 370
- data integration analysis
  - case study
    - conceptual data integration model, building, 123
    - overview, 117-123
    - source data quality, assessing, 130-134
    - source system data profiling, 124-130
    - source/target data mappings, 135-144
  - conceptual data integration model, building, 101-104
  - data quality development in, 339
  - scope, defining, 100-101
  - source data quality, assessing, 109-111
  - source system data profiling, 104-108
  - source/target data mappings, 111-115
- data integration applications, defined, 374
- data integration architecture
  - defined, 371
  - establishing in logical design phase, 151-154, 174-177
- data integration jobs. *See also* development cycle phase
  - completing code for, 262-266
  - defined, 374
  - job coding standards, 253-254
  - job scheduling for, 221-222, 240-248
- data integration layer (data warehouses)
  - aggregations in, 121
  - unit testing, 270-271
- data integration modeling
  - business case for, 45-47
  - case study
    - common component data integration models, developing, 92-94
  - conceptual data integration model, building, 69
  - high-level logical data integration model, building, 70-72
  - logical data quality data integration models, defining, 76-80
  - logical extraction data integration models, building, 72-76
  - logical extraction data integration models, converting to physical models, 88-90
  - logical load data integration models, converting to physical models, 90-92
  - logical load data integration models, defining, 85-86
  - logical transform data integration models, defining, 81-85
  - overview, 67-69
  - physical data integration modeling, converting logical models to, 88-92
  - physical data integration modeling, determining strategy, 87
  - physical data integration modeling, sequencing, 94-95
- conceptual data integration modeling, 51
  - defined, 374
  - development tools for, 61-63
- industry-based data integration models, 63-64
- logical data integration modeling, 51-55, 156-163, 180-197
- physical data integration modeling, 56-61
- to reference architecture, 48-49
- in SDLC (Systems Development Life Cycle), 49
- structuring, 50
- data integration process management, oversight of, 307
- data mappings, 111-115, 135-144
- data modeling, data integration versus, 2
- data profiling on source systems, 104-108, 124-130
- data quality, 329-330, 353
  - causes of poor quality, 31-32
  - check points, 32
  - checking before transformations, 369
  - common component data quality data integration models, 58-59, 92-94
  - defined, 31
  - framework for, 330-334
    - business-process data quality dimensions, 333-334

- key data quality elements, 331
- process types, 334
- technical data quality dimensions, 332-333
- guiding principles
  - aggregation
  - transformations, where to perform, 370
- data integration
  - environment volumetric sizing, 370
  - subject area volumetric sizing, 370
  - transformation
    - componentization, 370
- life cycle, 334-336
  - audit phase, 345-351
  - define phase, 336-345
  - renovate phase, 351-353
- logical data quality data
  - integration models, 53-54, 76-80
- oversight of, 305-306
- source data quality
  - assessing, 109-111
  - data integration analysis
    - case study, 130-134
  - where to check, 32-34
- data quality assessment and remediation projects, 352
- data quality audit and renovation teams, 300-301
- data quality criteria
  - defined, 371
  - identifying in logical design phase, 154-156, 177-180
- data quality elements, identifying, 336-337
- data quality measurement process, developing, 346-348
- data quality processes, 31-34
  - defined, 372
  - developing preventive processes, 337-345
  - types of, 334
- data quality programs, 353
- data quality reports, developing, 348-350
- data quality SWAT renovation projects, 352
- data stewardship community, 303-304
- data stewardship processes, 304-305
- data type validation, 109
- data validation checks, 109-110
- data volumetrics, defined, 374
- data warehouse database layer (data warehouses)
  - aggregations in, 121
  - unit testing, 271
- data warehouses
  - aggregations in, 120-122
  - calculations in, 120-122
  - capturing metadata, 325-326
  - data governance in, 305-309
  - development life cycle, 309
  - testing in, 266-275
    - integration testing, 272-273
    - system and performance testing, 273-274
  - types of, 268-269
  - unit testing, 269-272, 283-287
  - user acceptance testing, 274-275
- database development, data quality tasks in, 341-345
- database queries (data warehouses), aggregations in, 122
- data-related programs and projects, data governance role in, 302
- date format checks, 109
- date range validation, 110
- define phase (data quality life cycle), 334, 336-345
  - data quality elements, identifying, 336-337
  - preventive data quality processes, developing, 337-345
  - scope, defining, 336
- definitional dimension (data quality), 334
- deleted transactions, handling, 218-219
- delta processing, defined, 373
- design modeling. *See* data integration modeling
- design phases. *See* logical design phase; physical design phase
- development cycle phase, 251-253
  - configuration management, 275-277
    - Software Promotion Life Cycle (SPLC), 277
    - version control, 277
- data integration jobs, completing code for, 262-266
- data quality development in, 339
- data warehouse testing, 266-275
  - integration testing, 272-273
  - system and performance testing, 273-274
  - types of, 268-269
  - unit testing, 269-272, 283-287
  - user acceptance testing, 274-275
- error-handling requirements, 255
- job coding standards, 253-254
- naming standards, 255-256
- prototyping, 252, 257-262, 279-283
- development environment
  - preparation in physical design phase, 201-203
- development life cycle of data warehouses, 309
- development tools for data integration modeling, 61-63
- DGO (Data Governance Office), 300

direct audits, 351  
 direct measures of data quality, 346  
 disaster recovery for load-ready publish landing zones, 40  
 disk space requirements for initial staging, 30-31  
 disk space sizing, 148-150  
 distribution measures of data quality, 347  
 documenting nonstandard code, 254  
 duplicate key/field checks, 110

## E

EAI (Enterprise Application Integration), 8-9  
 encapsulation in reference architecture, 21-24  
 enrichment transformations, 36-38, 373  
 Enterprise Application Integration (EAI), 8-9  
 entity metrics, 346  
 error threshold checks, 110-111  
 error-handling requirements in development cycle phase, 255  
 ETL (Extract, Transform, Load), 14-15  
 evaluating reuse, 74  
 Executive Data Governance Committee, 300  
 Extract, Transform, Load (ETL), 14-15  
 extract sizing, 148  
 extract verification processes, designing, 57-58  
 extraction data integration models, 52-53, 72-76, 88-90  
 extract/subscribe processes, 27-29, 372

## F

federation, 12-13  
 file-to-file matching, 218  
 filters, target filters, 38-39, 373

fine-grained SOA objects, 227  
 foreign key analysis, 108  
 foreign key constraints, 342  
 foundational processes for data governance, 294  
   best practices, 294  
   policy examples, 294  
   sample mission statement, 294  
 FTP to target load architecture, 41, 373  
 functions, naming standards, 254

## G

governance. *See* data governance  
 “grab everything,” 28-29, 369  
 guidelines, defined, 294

## H

hard deletes, 218  
 high-level logical data integration model, 52  
   data integration modeling case study, 70-72  
   in logical design phase, 157-158, 181-183  
   in physical design phase, 205-206  
 history data conversion  
   in logical design phase, 163-166, 195-197  
   in physical design phase, finalizing, 220-221, 238-239  
 horizontal filtering, 38, 373

## I

improve phase (data quality life cycle), 335  
 inaccurate data, 32  
 inconsistent data definitions, 32  
 incorrect data, 342  
 indirect measures of data quality, 346

industry-based data integration models, 63-64  
 Information Technology, relationship with business, 293  
 initial staging landing zone, 29-31, 372  
 integration testing, 268, 272-273  
 invalid data, 31, 342

## J-K

job coding standards, 253-254  
 job log files, 254  
 job scheduling for data integration jobs, 221-222, 240-248  
 join transformations, 36-37, 373  
 Kernighan, Brian, 21  
 key data quality elements, 331

## L

landing zones  
   clean staging landing zone, 34  
   initial staging landing zone, 29-31  
   load-ready publish landing zone, 39-40  
 layers  
   of architectural patterns, 26-27  
   in reference architecture, 21  
 load/publish processes, 40-41  
   defined, 373  
   logical load data integration models, 55, 85-86, 90-92  
 load-ready publish landing zone, 39-40  
 load-ready staging area, defined, 373  
 log scrapers, 218  
 logical data integration modeling, 51-55  
   converting to physical data integration models, 56, 203-210, 229-236  
   defined, 49, 374



- high-level logical data
  - integration model, 52
  - data integration modeling
    - case study, 70-72
    - in physical design phase, 205-206
- logical data quality data
  - integration models, 53-54, 76-80
- in logical design phase, 156-163, 180-197
- logical extraction data
  - integration models, 52-53, 72-76, 88-90
- logical load data integration models, 55, 85-86, 90-92
- logical transform data
  - integration models, 54, 81-85
- physical data integration modeling versus, 61
- logical data mart data integration models in logical design phase, 192-195
- logical data quality data
  - integration models, 53-54
  - data integration modeling
    - case study, 76-80
    - in logical design phase, 159-160, 187-190
- logical design phase, 147
  - data integration architecture, establishing, 151-154, 174-177
  - data quality criteria, identifying, 154-156, 177-180
  - data quality development in, 339
  - history data conversion, 163-166, 195-197
  - logical data integration models, creating, 156-163, 180-197
  - source system volumetrics, 147-151
    - case study, 169-174
    - disk space sizing, 148-150
    - extract sizing, 148

- logical extraction data
  - integration models, 52-53
  - data integration modeling
    - case study, 72-76, 88-90
    - in logical design phase, 158-159, 183-187
- logical load data integration models, 55
  - data integration modeling
    - case study, 85-86, 90-92
    - in logical design phase, 162-163, 191-192
- logical metadata, 316
- logical transform data integration models, 54
  - data integration modeling
    - case study, 81-85
    - in logical design phase, 161-162, 190-191
- lookup checks, 110
- lookup transformations, 37, 373

## M

- management of metadata, 321-326
  - current state inventory, 322
  - importance in data governance, 321
  - life cycle, 324-326
  - planning, 322-324
- many-to-one data mapping, 114-115
- master data management (MDM), oversight of, 306
- measuring data quality, 346-348
- message publishing load architecture, 41, 373
- metadata
  - categories of, 314-319
    - analytic metadata, 318
    - business metadata, 315
    - navigational metadata, 317-318
    - operational metadata, 319
    - structural metadata, 315-316
  - defined, 313

- management of, 321-326
  - current state inventory, 322
  - importance in data governance, 321
  - life cycle, 324-326
  - planning, 322-324
  - oversight of, 306
  - in reference architecture, 319-320
  - role in data integration, 314
  - users of, 320-321
- missing data, 32, 342
- mission statements for data governance, 294
- modeling. *See* data integration modeling
- modularity
  - in physical design phase, 200-201
  - of reference architecture, 22-24

## N

- naming standards
  - for data integration components, 255-256
  - for variables and functions, 254
- navigational metadata, 317-318
- nonstandard code, documenting, 254
- null checks, 110
- numeric value range checks, 110

## O

- one-to-many data mapping, 113-114
- one-to-one data mapping, 113
- ongoing data quality processing, 351
- operational metadata, 319
- operational requirements
  - for data governance policies, 294
  - in physical design phase, defining, 221-224, 239-240

operational users of metadata, 321

optional data quality checkpoints, data integration modeling case study, 80

organizational structure in data governance, 294-304

- business analytics centers of excellence, 302-303
- chief data officers, 300
- Data Governance Office (DGO), 300
- data quality audit and renovation teams, 300-301
- data stewardship community, 303-304
- data-related programs and projects, 302
- Executive Data Governance Committee, 300

## P

parallel processing in physical design phase, 210-216, 237-238

patterns. *See* architectural patterns

percentage range checks, 110

performance testing, 269, 273-274

physical common component data integration models, 58-60

- data integration modeling case study, 92-94
- designing, 206-208, 230-232

physical data integration modeling, 56-61

- converting logical data integration models to, 56, 203-210
- data integration modeling case study, 88-92
- data integration physical design case study, 229-236

- defined, 49, 374
- determining strategy for, data integration modeling case study, 87
- logical data integration modeling versus, 61
- physical common component data integration models, 58-60, 92-94
- physical source system data integration models, 57-58
- physical subject area load data integration models, 60-61
- sequencing, data integration modeling case study, 94-95
- target-based data integration design, 56-57

physical data mart data integration models, designing, case study, 236

physical design phase, 199-200

- Change Data Capture (CDC), 216-220
- component-based physical designs, creating, 200-201
- data quality development in, 339
- development environment preparation, 201-203
- history data conversion, finalizing, 220-221, 238-239
- operational requirements, defining, 221-224, 239-240
- parallel processing, 210-216, 237-238
- physical data integration models, creating, 203-210, 229-236
- SOA-enabled framework, designing for, 225-228

physical load architectures, 41

physical source system data integration models, 57-58, 208-209, 232-234

physical subject area load data integration models, 60-61

- data integration modeling case study, 90-92
- designing, 209-210, 234-236

piped data load architecture, 41, 373

planning metadata management, 322-324

point-to-point application development, 203-205

policies

- data governance policy examples, 294
- defined, 294

poor data quality, causes of, 31-32

prebuilt data integration models, 63-64

precise dimension (data quality), 332

preparing development environment in physical design phase, 201-203

preventive data quality processes, developing, 337-345

primary key constraints, 342

prioritizing data elements, 106

process modeling

- defined, 374
- types of, 48

processes

- data integration modeling. *See* data integration modeling
- data quality processes, 31-34
- defined, 372
- developing preventive processes, 337-345
- types of, 334
- extract/subscribe processes, 27-29
- load/publish processes, 40-41
- transformations, 35-39
- calculations and splits, 35-36
- conforming transformations, 35
- defined, 35

- processing and enrichment transformations, 36-38
- target filters, 38-39
- processing transformations, 36-38
- production support team, determining, 222-224, 248
- profiling, 104-108, 124-130
- prototyping in development cycle phase, 252, 257-262, 279-283

## Q-R

- quality. *See* data quality; data quality processes
- quality measures of data quality, 347
- RDBMS utilities load architecture, 41, 373
- “read once, write many,” 28
- real-time analysis of business intelligence, 12
- record-level lookup checks, 110
- reference architecture
  - data integration modeling to, 48-49
  - defined, 19-20
  - metadata in, 319-320
  - modularity of, 22-24
  - objectives of, 21-22
  - purposes of, 26
  - scalability of, 24-25
  - structuring models on, 50
- renovate phase (data quality life cycle), 351-353
  - data quality assessment and remediation projects, 352
  - data quality programs, 353
  - data quality SWAT renovation projects, 352
- reports, developing data quality reports, 348-350
- requirements
  - defined, 294
  - disk space requirements for initial staging, 30-31

- for metadata user repository, 322-323
- operational requirements
  - for data governance policies, 294
  - in physical design phase, defining, 221-224, 239-240
- reuse, evaluating, 74
- Ritchie, Dennis, 21

## S

- Sarbanes-Oxley compliance, 309
- scalability of reference architecture, 24-25
- scheduling data integration jobs, 221-222, 240-248
- scope, defining, 100-101
  - conceptual data integration model, building, 101-104
  - in data quality life cycle, 336
- SDLC (Systems Development Life Cycle), data integration modeling in, 49
- security testing, 273
- Service-Oriented Architecture (SOA), 9-12
- simplicity in reference architectural layers, 21
- sizing for load-ready publish landing zones, 40
- SOA (Service-Oriented Architecture), 9-12
- SOA-enabled framework, designing for, 225-228
- soft deletes, 218
- Software Promotion Life Cycle (SPLC), 277
- source data quality, assessing, 109-111, 130-134
- source system data discovery data profiling, 104-108, 124-130
  - difficulty of, 103-104
- source system extract data integration models, 57-58, 264-265
- source system volumetrics, 147-151
  - case study, 169-174
  - disk space sizing, 148-150
  - extract sizing, 148
- source/target data mappings, 111-115, 135-144
- space requirements for initial staging, 30-31
- SPLC (Software Promotion Life Cycle), 277
- split transformations, 35-36, 372
- SQL load architecture, 41, 373
- staging areas. *See* landing zones standards
  - in data governance, 294
  - for data integration job coding, 253-254
  - defined, 294
- structural metadata, 315-316
- structuring data integration modeling, 50
- subject area files in reference architecture, 22-24
- subject area load data integration models, 60-61
  - completing code for, 265-266
  - data integration modeling case study, 90-92
- subject area volumetric sizing, 370
- subject areas, confirming, 73
- SWAT renovation projects, 352
- system and performance testing, 269, 273-274
- Systems Development Life Cycle (SDLC), data integration modeling in, 49

## T

- target data models, designing for Change Data Capture transactions, 218
- target database subject areas, confirming, 73
- target filters, 38-39, 373

target-based data integration  
 design, 56-57  
 target-based load design, 40-41  
 technical data quality  
 checkpoints, 32, 77-80  
 technical data quality  
 dimensions, 332-333  
 technical metadata, 316  
 technology users of metadata,  
 320  
 technology-driven poor data  
 quality, 31-32  
 testing in data warehouses,  
 266-275  
 integration testing, 272-273  
 system and performance  
 testing, 273-274  
 types of, 268-269  
 unit testing, 269-272,  
 283-287  
 user acceptance testing,  
 274-275  
 timely dimension (data quality),  
 332  
 tools for data integration  
 modeling, 61-63  
 transactional data integration, 8  
 capturing new/changed  
 transactions, 218  
 defined, 371  
 EAI (Enterprise Application  
 Integration), 8-9  
 real-time analysis of business  
 intelligence, 12  
 SOA (Service-Oriented  
 Architecture), 9-12  
 testing, data warehouse  
 testing versus, 267-268  
 transformations, 35-39  
 aggregation transformations,  
 where to perform, 370  
 calculations and splits, 35-36  
 checking data quality before,  
 369

common component  
 transformation data  
 integration models, 59-60,  
 92-94  
 componentization, 370  
 conforming transformations,  
 35  
 defined, 35, 372-373  
 logical transform data  
 integration models, 54,  
 81-85  
 processing and enrichment  
 transformations, 36-38  
 target filters, 38-39

## U

unique dimension (data quality),  
 332  
 unique key constraints, 342  
 unit testing, 268-272, 283-287  
 user acceptance testing, 269,  
 274-275  
 users of metadata, 320-323

## V

valid dimension (data quality),  
 332  
 validation checks, 109-111,  
 130-134  
 variables, naming standards, 254  
 version control in configuration  
 management, 277  
 vertical filtering, 38, 373  
 volumetric sizing  
 for data integration  
 environment, 370  
 defined, 374  
 in logical design phase,  
 147-151  
 case study, 169-174  
 disk space sizing, 148-150  
 extract sizing, 148  
 for subject areas, 370  
 volumetrics formula, 30

## W

Wheeler Automotive Company  
 case study. *See* data integration  
 analysis, case study  
 “write once, read many,” 369