Data Integration Blueprint and Modeling

Techniques for a Scalable and Sustainable Architecture

Anthony David Giordano

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

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

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

Appendix D is an online-only appendix. Print-book readers can download the appendix at www.ibmpressbooks.com/title/9780137084937. For eBook editions, the appendix is included in the book.

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). 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 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.

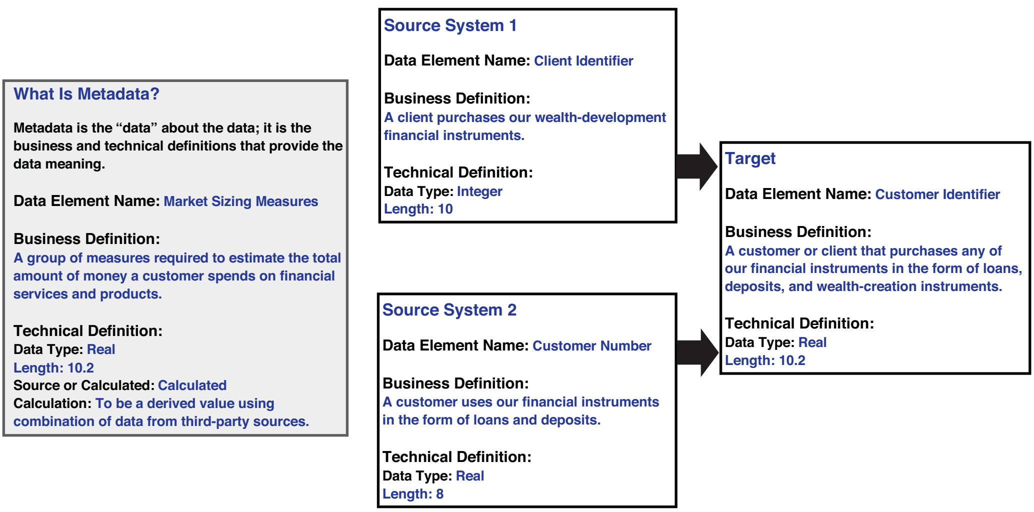

Figure I.1 Example of integrating data into information

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.

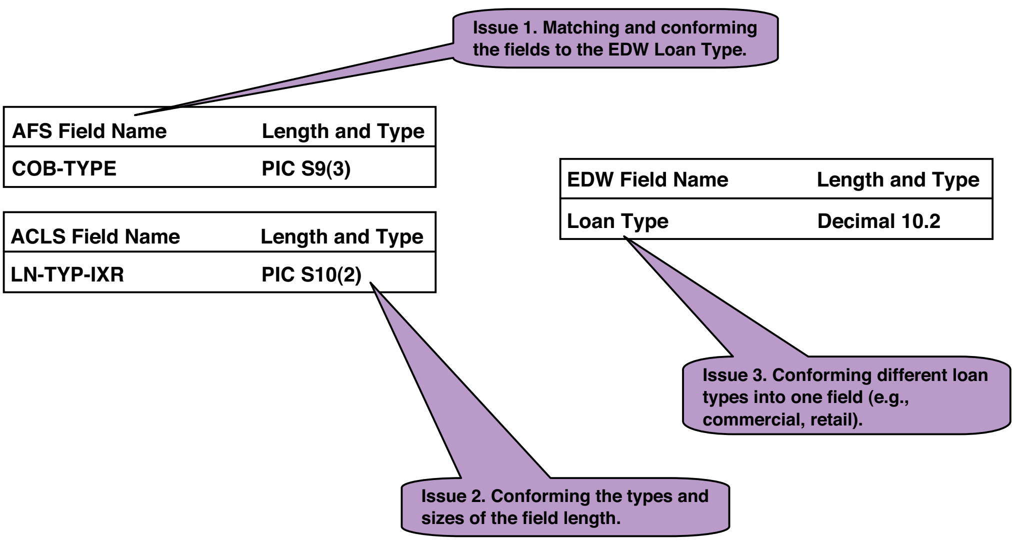

Figure I.2 Complexity issues with integrating data

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

CHAPTER 3

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.

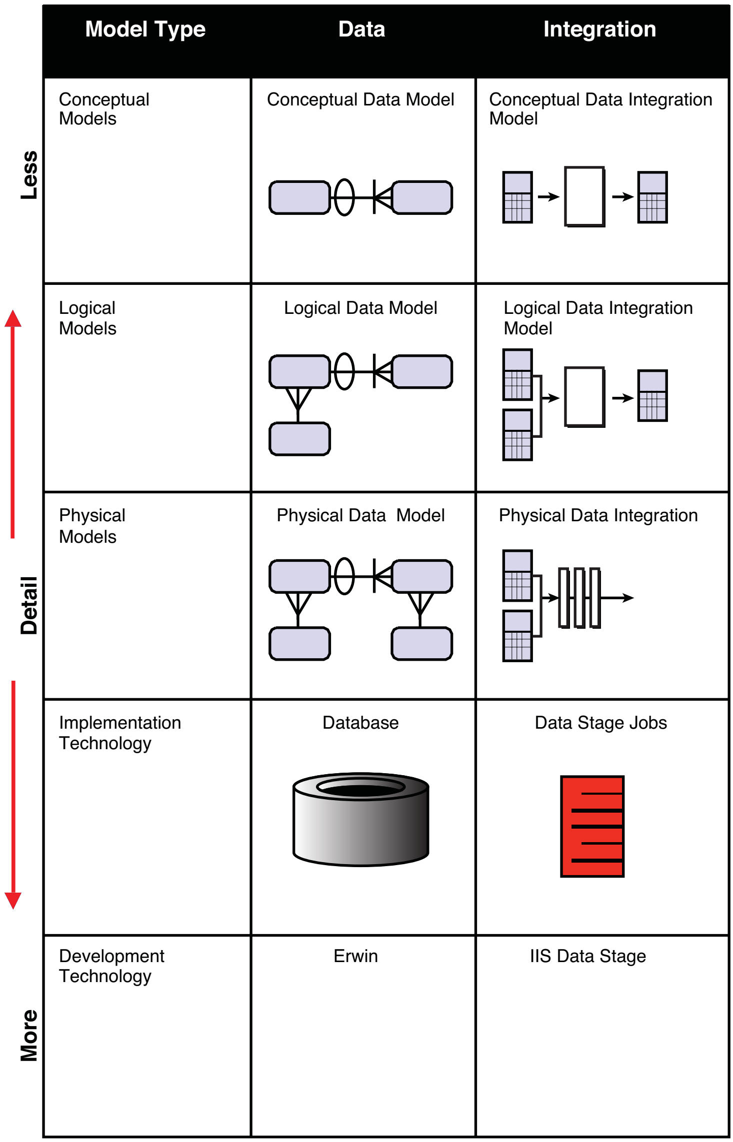

Figure 3.1 Modeling paradigm: data and data integration

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.

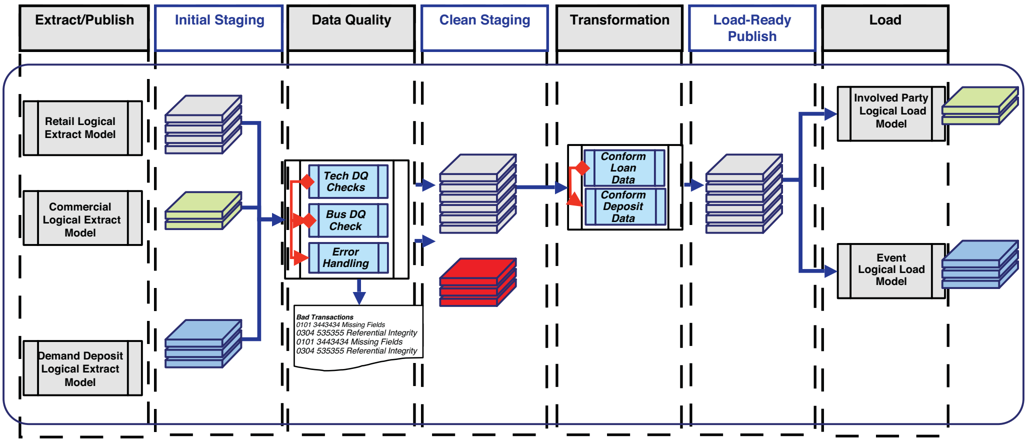

Figure 3.2 Designing models to the architecture

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.

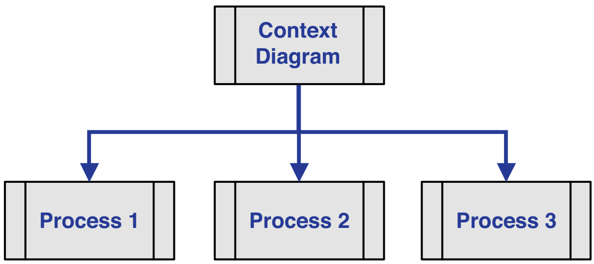

Figure 3.3 A traditional process model: data flow diagram

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.

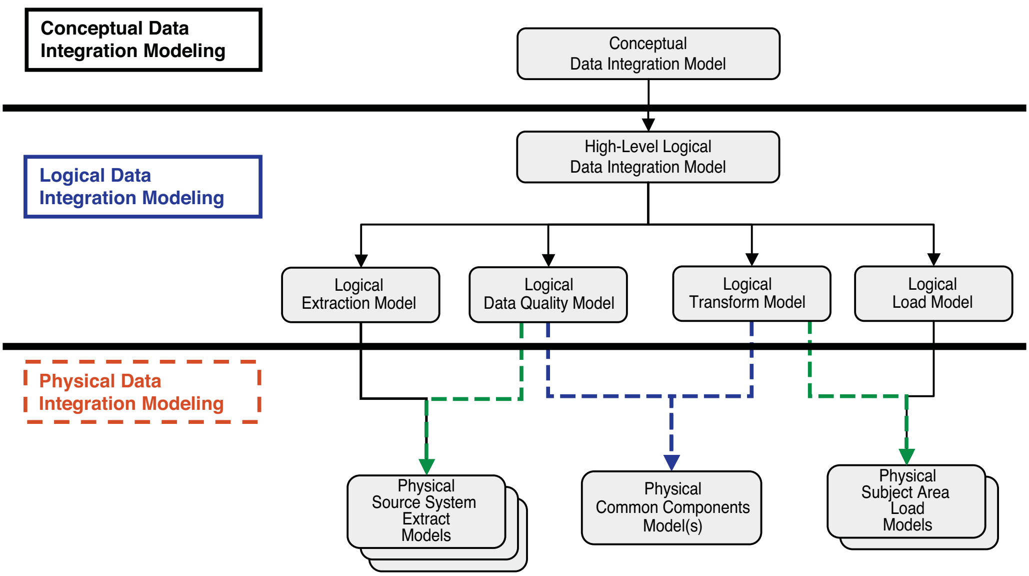

Figure 3.4 Data integration models by the Systems Development Life Cycle

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.

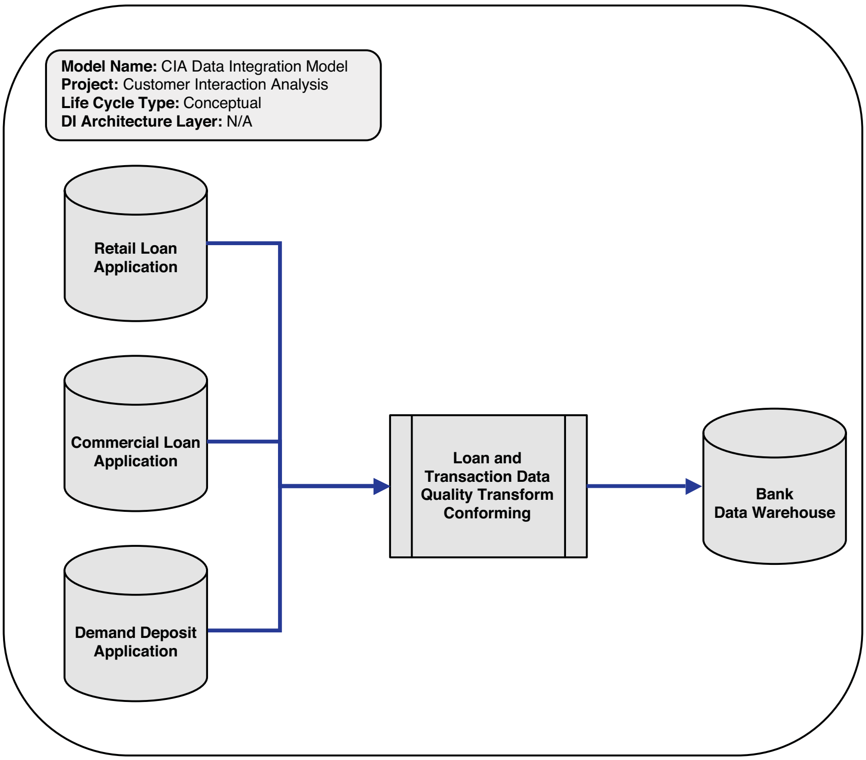

Figure 3.5 Conceptual data integration model example

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.

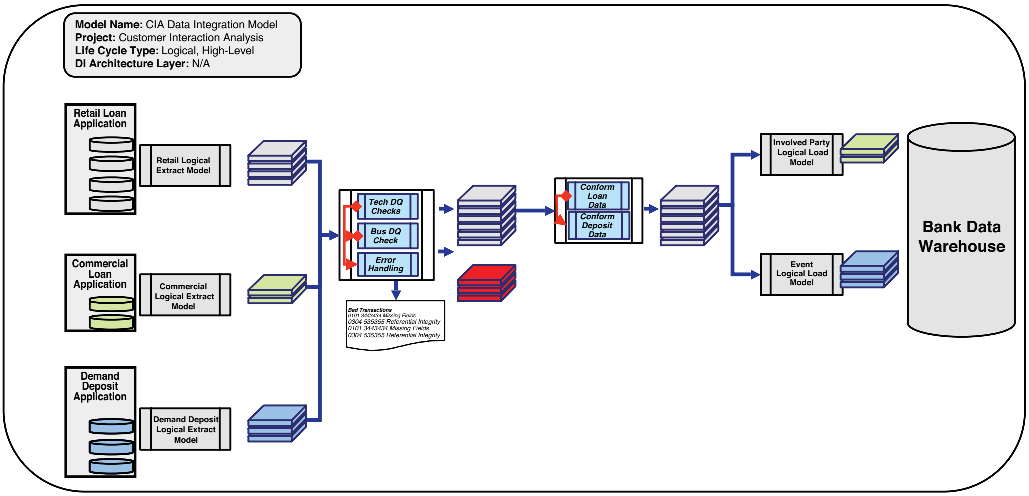

Figure 3.6 Logical high-level data integration model example

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.

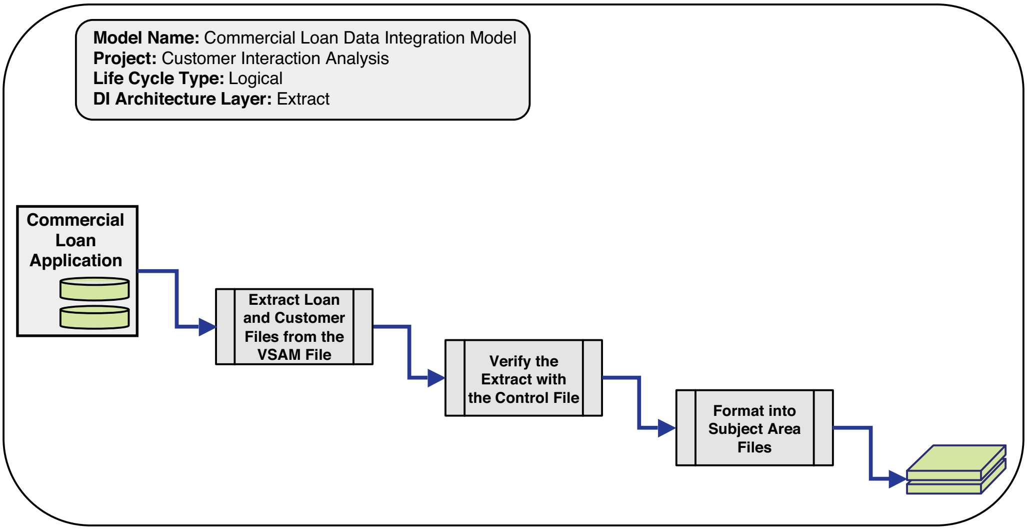

Figure 3.7 Logical extraction data integration model example

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.

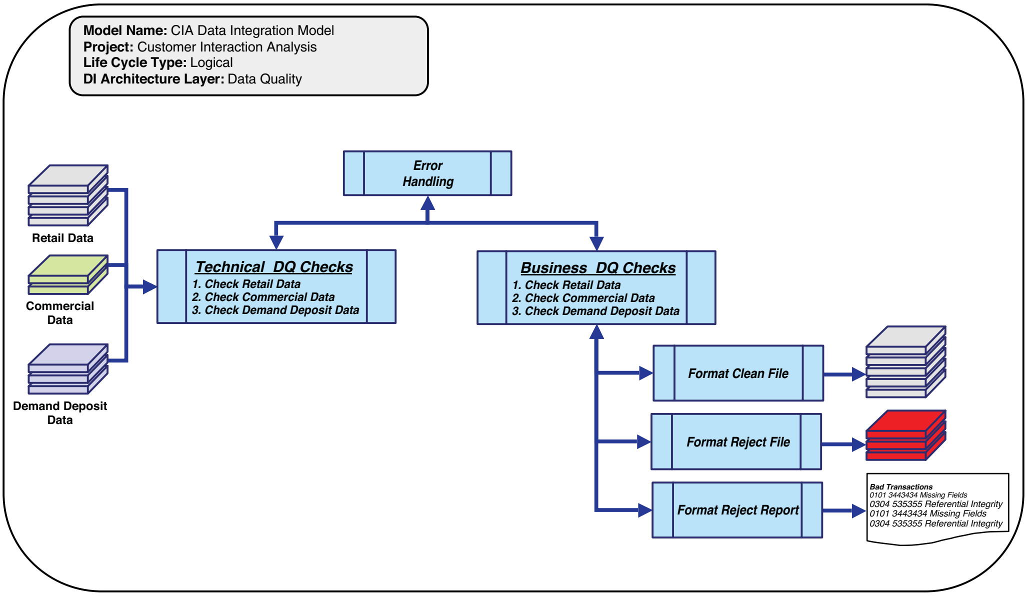

Figure 3.8 Logical data quality data integration model example

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.

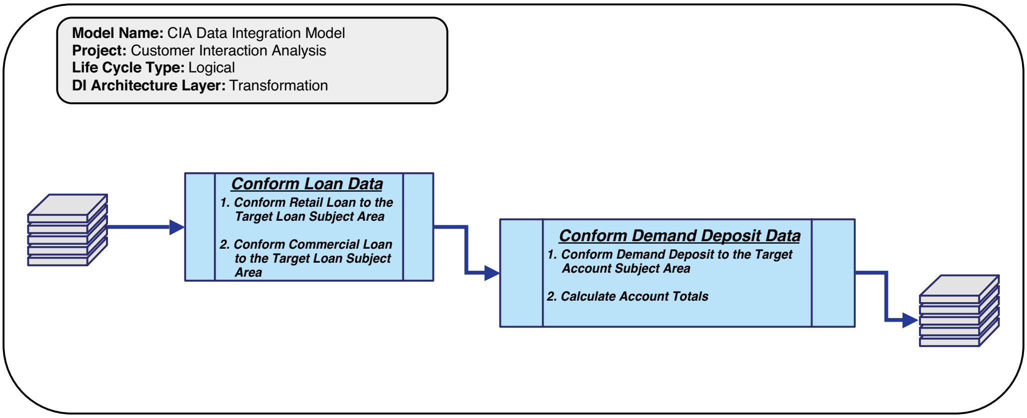

Figure 3.9 Logical transformation data integration model example

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.

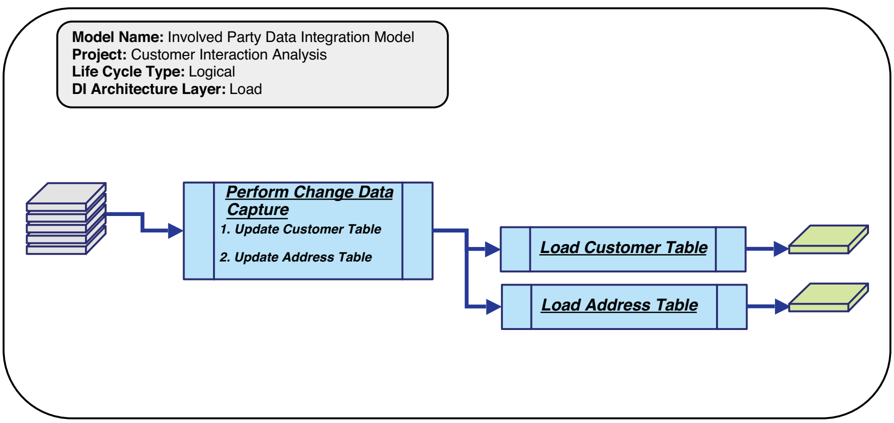

Figure 3.10 Logical load data integration model example

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.

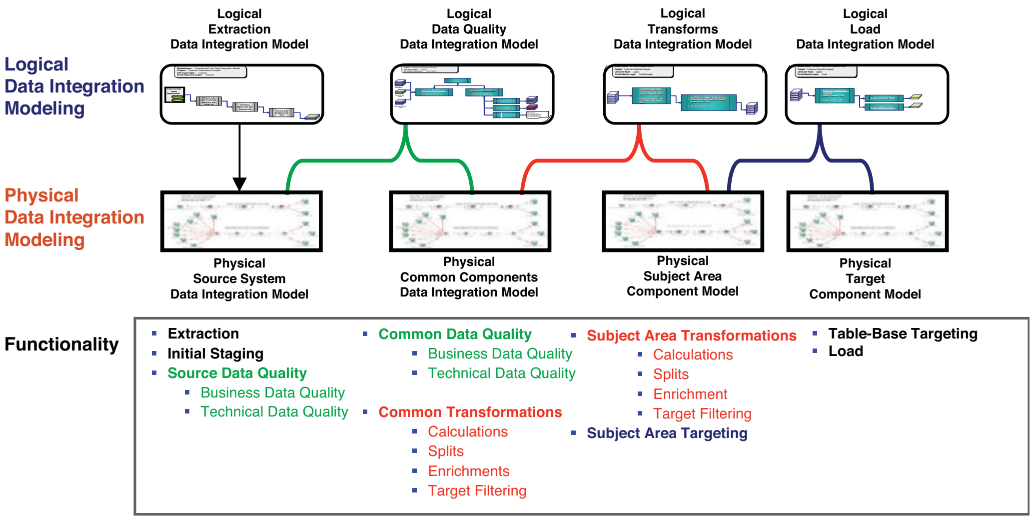

Figure 3.11 Distributing logical functionality between the "whats" and "hows"

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.

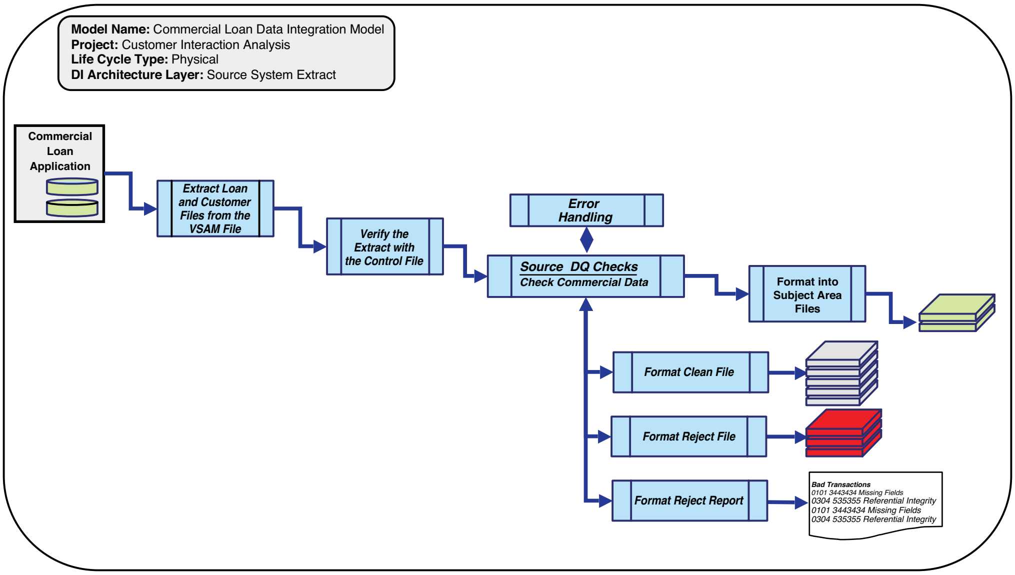

Figure 3.12 Physical source system extract data integration model example

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.

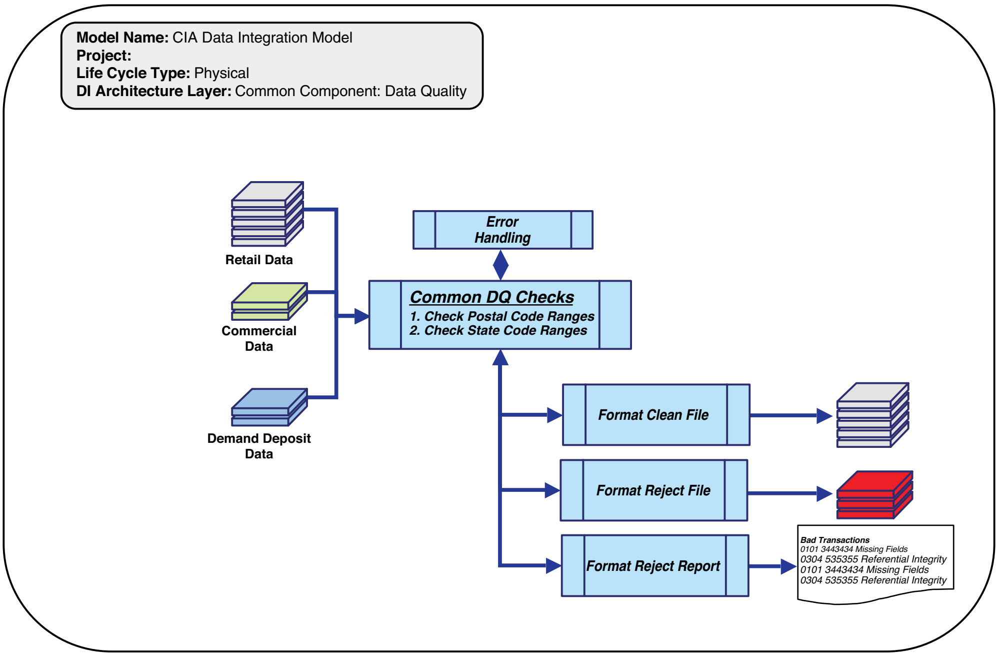

Figure 3.13 Common components—data quality data integration model example

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.

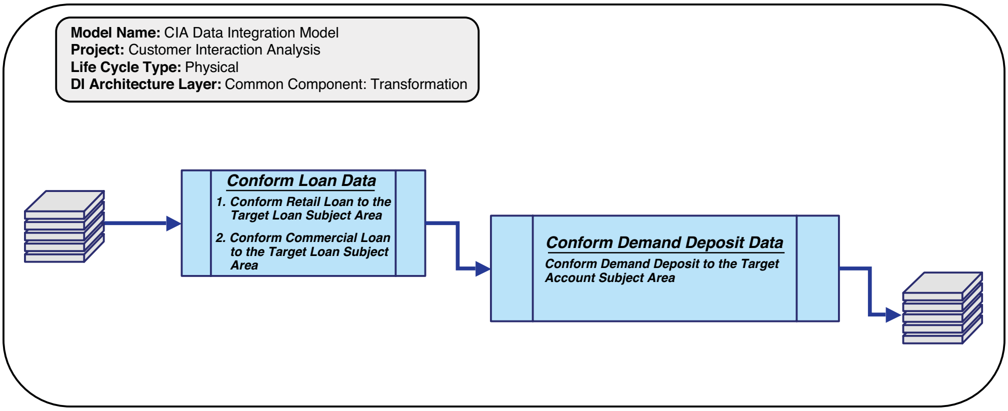

Figure 3.14 Common components—transform data integration model example

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.

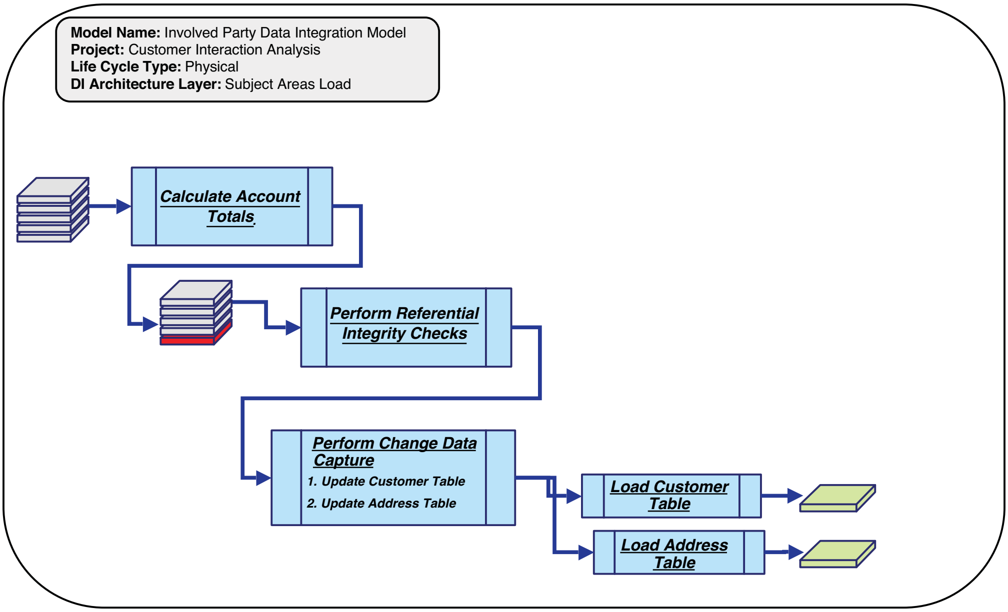

Figure 3.15 Physical subject data area load data integration model example

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.

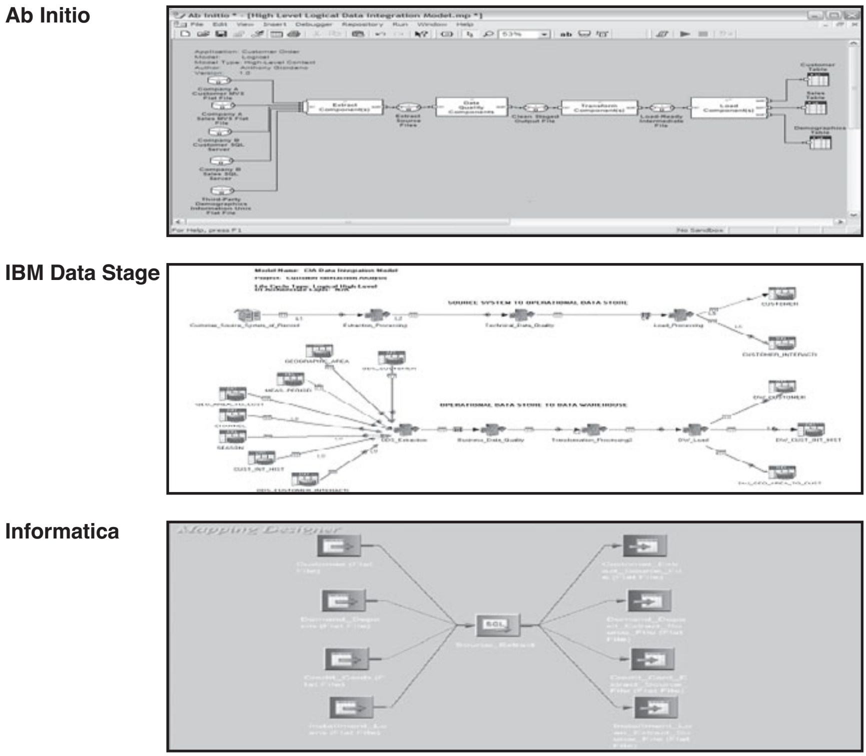

Figure 3.16 Data integration models by technology

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.

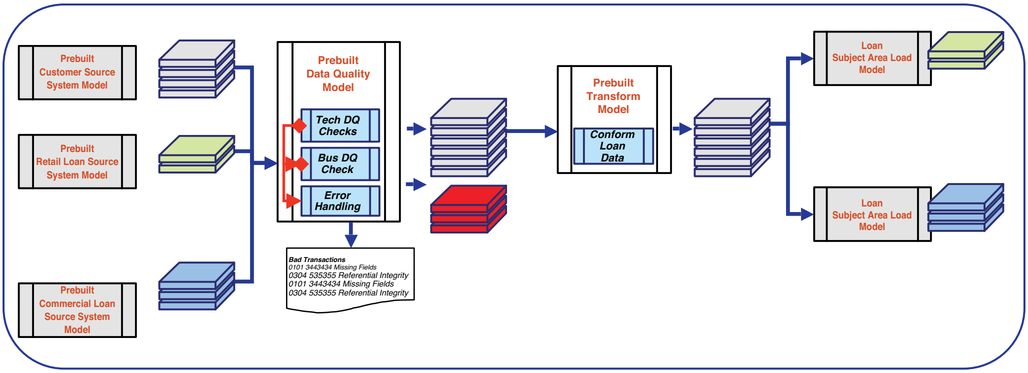

Figure 3.17 Industry-based data integration model example

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

B

C

M

N

O

Q-R

S

T

U

V

W