Data Integration Blueprint and Modeling

Techniques for a Scalable and Sustainable Architecture

Anthony David Giordano

CHAPTER 5

Data Integration Analysis

This chapter reviews the initial tasks for analyzing the requirements for a data integration solution, with the focus on the following:

- Scoping the target solution
- Confirming the source system information
- Determining the quality of the source data
- Developing the data mappings from source to target

This chapter also discusses how data integration analysis fits into an overall Systems Development Life Cycle (see Figure 5.1). The next several chapters detail how the data integration architecture and modeling techniques are integrated with analysis, logical design, technical design, build activities, tasks, and deliverables in addition to other key data integration analysis techniques and principles.

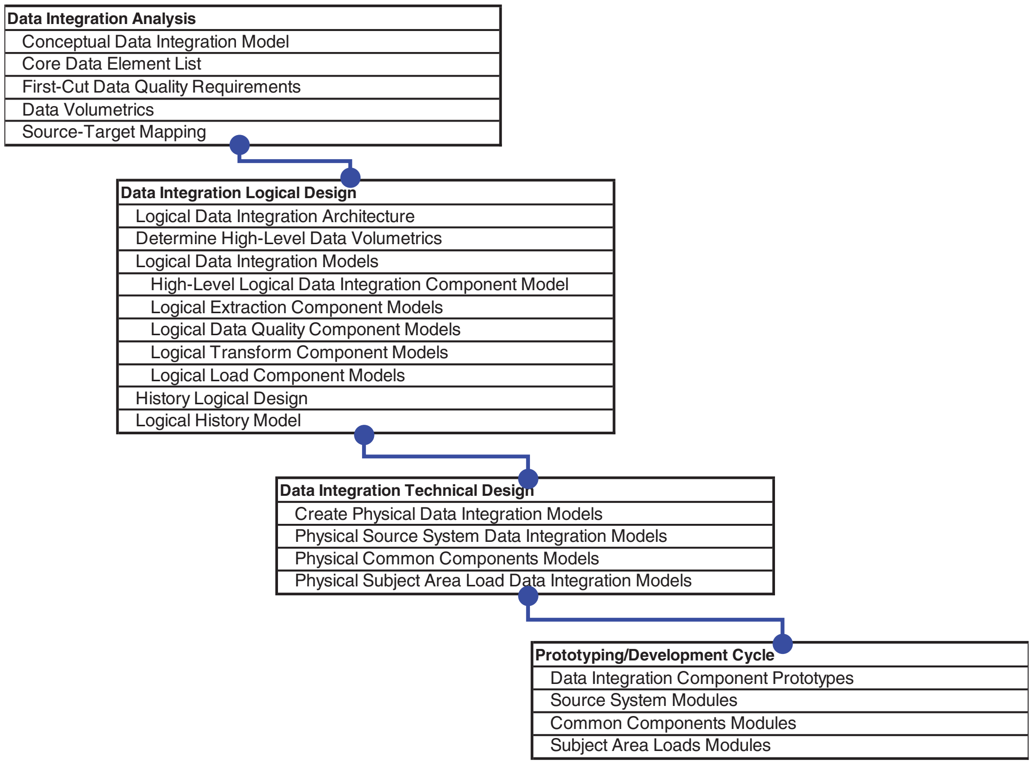

Data Integration Analysis
Conceptual Data Integration Model
Core Data Element List
First-Cut Data Quality Requirements
Data Volumetrics
Source-Target Mapping
Data Integration Logical Design
Logical Data Integration Architecture
Determine High-Level Data Volumetrics
Logical Data Integration Models
High-Level Logical Data Integration Component Model
Logical Extraction Component Models
Logical Data Quality Component Models
Logical Transform Component Models
Logical Load Component Models
History Logical Design
Logical History Model
Data Integration Technical Design
Create Physical Data Integration Models
Physical Source System Data Integration Models
Physical Common Components Models
Physical Subject Area Load Data Integration Models
Prototyping/Development Cycle
Data Integration Component Prototypes
Source System Modules
Common Components Modules
Subject Area Loads Modules

Figure 5.1 Data integration life cycle deliverables

Analyzing Data Integration Requirements

Traditional Systems Development Life Cycles define *analysis* as the phase that investigates a key business area or business problem as defined by the end-user community. It discerns the "whats" of a business problem.

The data integration analysis project phase scopes and defines the "logical whats" of the intended data integration processes or application.

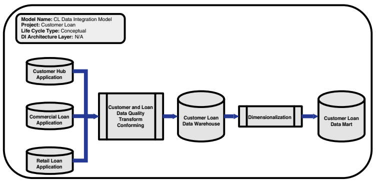
That first step in a data integration project is also the same step performed for any Information Technology project, which is defining the scope of the efforts and providing answers to the question "What do we need to do?" These activities are then aligned, sequenced, timed, and integrated into an overall project plan.

For a data integration project, defining scope means determining the following:

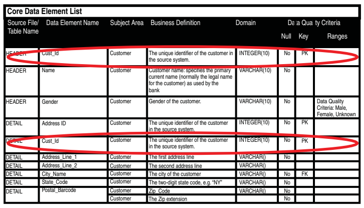
- *What* are the sources?
- *What* is the target (or targets)?
- *What* are the data requirements (fulfill business requirements if any)?

- *What* are the business rules needed to restructure the data to meet the requirements of the intended target(s)?

Once the scope is defined, understood, and agreed to, the data integration project team will need to analyze the sources of the data for the targets, investigate their data quality and volumes, and then map the source data fields to the intended target to produce deliverables, as illustrated in Figure 5.2.


Model Name: CL Data Integration Model
Project: Customer Loan
Life Cycle Type: Conceptual
DI Architecture Layer: N/A
Customer Hub Application
Commercial Loan Application
Retail Loan Application
Customer and Loan Data Quality Transform Conforming
Customer Loan Data Warehouse
Dimensionalization
Customer Loan Data Mart

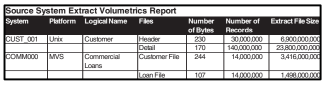
Analysis Deliverable



Core Data Element List

Source File/ Table Name	Data Element Name	Subject Area	Business Definition	Domain	Data Quality Criteria		
					Null	Key	Ranges
HEADER	Cust_Id	Customer	The unique identifier of the customer in the source system.	INTEGER(10)	No	PK	
HEADER	Name	Customer	Customer name: specifies the primary current name (normally the legal name for the customer) as used by the bank	VARCHAR(10)	No		
HEADER	Gender	Customer	Gender of the customer.	VARCHAR(10)	No		Data Quality Criteria: Male, Female, Unknown
DETAIL	Address ID	Customer	The unique identifier of the customer in the source system.	INTEGER(10)	No	PK	
DETAIL	Cust_Id	Customer	The unique identifier of the customer in the source system.	INTEGER(10)	No	PK	
DETAIL	Address_Line_1	Customer	The first address line	VARCHAR()	No		
DETAIL	Address_Line_2	Customer	The second address line	VARCHAR()			
DETAIL	City_Name	Customer	The city of the customer	VARCHAR()	No	FK	
DETAIL	State_Code	Customer	The two-digit state code, e.g. "NY"	VARCHAR()	No		
DETAIL	Postal_Barcode	Customer	Zip_Code	VARCHAR()	No		
		Customer	The Zip extension		No		

Analysis Deliverable



Source System Extract Volumetrics Report

System	Platform	Logical Name	Files	Number of Bytes	Number of Records	Extract File Size
CUST_001	Unix	Customer	Header	230	30,000,000	6,900,000,000
			Detail	170	140,000,000	23,800,000,000
COMM000	MVS	Commercial Loans	Customer File	244	14,000,000	3,416,000,000
			Loan File	107	14,000,000	1,498,000,000

Analysis Deliverable

Figure 5.2 Sample data integration analysis deliverables

To define the project scope for the data integration project and determine the requirements needed for the intended data integration processes, the following data integration solution requirements tasks must be performed:

1. Build a conceptual data integration model.
2. Perform source system profiling.
3. Review/assess source data quality.
4. Perform data mapping to source systems.

Building a Conceptual Data Integration Model

The first task in data integration analysis is to define the scope of the intended data integration process. The best scope management "tool" is a visual representation of the sources and targets. That visual representation is the conceptual data integration model.

How does a conceptual data integration model help define scope? A conceptual data integration model provides a high-level representation of how the data integration requirements will be met for the proposed system. It also provides that visual representation of how those requirements will be satisfied.

At this stage, it is only necessary to identify the planned source and target data stores and potential processes needed to fully understand the ramifications of the users' requirements for data integration in terms of the feasibility for the project. Things to review in developing a conceptual data integration model include the following:

- Identifying existing source system extractions that could be leveraged as potential sources

- Determining if existing data quality checkpoints in the environment could be reused
- Identifying existing target data stores for the target database

Figure 5.3 is the conceptual data integration model from the banking case study as sample output of the conceptual data integration modeling task that was developed in Chapter 4, "Case Study: Customer Loan Data Warehouse Project."

Please notice the differences and similarities in the models when the conceptual data integration model is developed for the Wheeler Bank case study in Chapter 4.

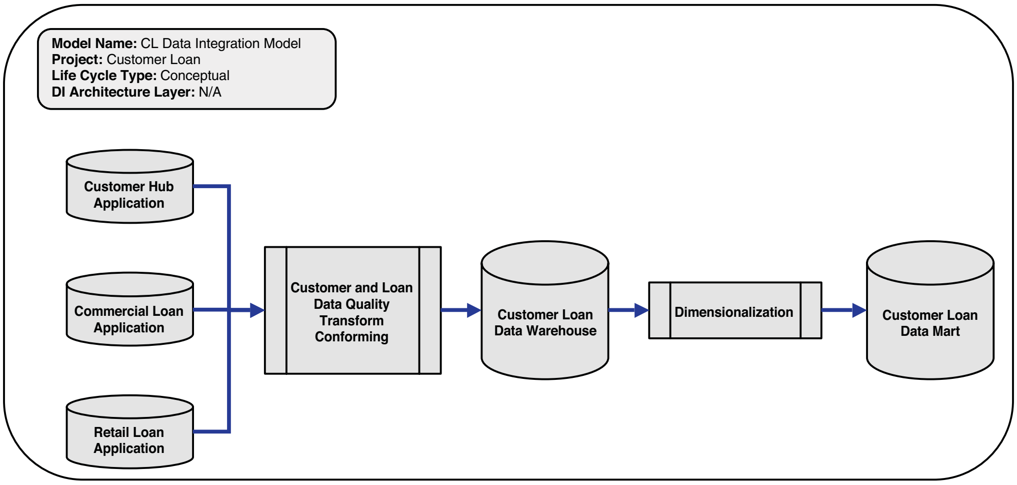

Model Name: CL Data Integration Model
Project: Customer Loan
Life Cycle Type: Conceptual
DI Architecture Layer: N/A
Customer Hub Application
Commercial Loan Application
Retail Loan Application
Customer and Loan Data Quality Transform Conforming
Customer Loan Data Warehouse
Dimensionalization
Customer Loan Data Mart

Figure 5.3 Data integration life-cycle deliverables

Again, a conceptual data integration model simply documents the scope of the proposed data integration application in terms of the high-level sources, targets, and business rules.

Key Conceptual Data Integration Modeling Task Steps

Building a conceptual data integration model requires these steps:

1. **Identify the major source data stores**—What are the expected systems that the data will be extracted from? How many files/tables are expected to be sourced from these systems? How wide are the files/tables (e.g., the number of columns)?
2. **Document initial volumetrics by source system**—What is the high-level estimate on the frequency and volumes of data from each source system?
3. **Review the data integration environment for reusable components**—If this is an existing data integration environment, are there extract components/jobs for the needed source system in place? Are there loads in place that can be extended and/or leveraged? Are there common data quality or transformation components/jobs that can be used?

4. **Define initial business rules**—What are the business rules in terms of data quality business rules and transformation business rules that can be documented at a high level?
5. **Identify the major target data stores**—What is the intended data store(s)? What are their subject areas, such as customer and product?

With the scope defined as well as the source systems and high-level business rules identified, it is critical to discover as much as possible about the sources' underlying data structures, data quality, frequency, and volumes. The next three tasks focus on that source system data discovery.

Why Is Source System Data Discovery So Difficult?

It used to be a foregone conclusion that a project manager would have to significantly pad their development and unit testing estimates due to data mapping issues. Those issues were due to a lack of understanding of underlying format and the data rules of the source systems, as well as the lack of rigor attached to the time and effort in performing source systems data discovery. This task was often overlooked due to the sheer magnitude of the difficulty.

Why is source systems data discovery so difficult? There are several reasons, including the following:

- **Undocumented and complex source formats**—Documentation for many systems are either out of date or undocumented. For example, many systems use old flat-file formats with unstructured file layouts with nested logic (hierarchies) built in with no easy method of understanding the number of layers. Documentation if it does exist is typically not kept up to date and has led to significant misunderstandings of the actual format of source systems.
- **Data formatting differences**—Often, data goes through an undocumented process that converts a field from one type to another while en route from one system to the source system being examined. For example, a calculation field defined as Packed Decimal is really Integer based on an undocumented transformation. This incorrect data formatting can cause an incorrect data mapping error, incorrect calculation, or even the data integration job to terminate.
- **Lack of client subject matter knowledge**—Often, the designers and developers of older transactional data systems are no longer available, leaving little to no documentation to aid in understanding the underlying data format and processing rules.
- **Bad data quality**—Often in source systems analysis, mapping issues can be a result of bad data quality, for example, a lack of primary or foreign keys. Referential integrity is often not enforced in the database, but in the ETL logic, which occurs for a multitude of reasons (e.g., performance). However, when these keys are not checked in the ETL logic or missed, leaving the mandatory key fields null, there are significant downstream technical data quality issues.

A series of data discovery techniques have been developed over time to analyze the data structures of the source systems to aid in discovering the underlying format and data rules of the source systems. The first of these techniques is data profiling.

Performing Source System Data Profiling

The first source system discovery task, data profiling, uncovers source systems' structural information, such as the data elements (fields or database columns), their format, dependencies between those data elements, relationships between the tables (if they exist via primary and foreign keys), data redundancies both known and unknown, and technical data quality issues (such as missing or unmatched key fields).

Data profiling as a formal data integration technique has evolved into a more formal and integrated function within the data integration discipline. It is simply impossible to build highly reliable data integration processes without a thorough understanding of the source data. In the past, data profiling was performed sporadically on data projects, often where a database administrator would run a series of SQL queries to look for data gaps. Both the technique and tools for data profiling have matured greatly in the past five years.

The following sections provide a brief overview of techniques and the tasks for performing data profiling.

Overview of Data Profiling

Data profiling uncovers critical source system information through the following:

- **Reviewing the data elements (fields or database columns) and their actual formats**—As discussed earlier, existing system documentation on the formats of the data is either inaccurate or outdated. Determining that a field is *Integer 7* rather than *VarChar 6* is invaluable in preventing mapping, coding, and testing issues.
- **Determining data dependencies and their actual relationships between the tables (if they exist via primary and foreign keys)**—For a host of reasons (performance for one), referential integrity is not enforced in most source systems. Determining and verifying that the data in the lookup tables matches the data in the main tables and that the primary key cascades into the detail tables is critical in maintaining referential integrity.

  Figure 5.4 provides an example of the types of data quality issues uncovered in data profiling.