Paper HW01

Planning for and Designing a Data Warehouse: A Hands-On Workshop

Gregory S. Nelson
ThotWave Technologies, Cary, North Carolina

ABSTRACT **1**

INTRODUCTION **2**

SCOPE OF THIS WORKSHOP 2
THE WORKSHOP DATA 3

STEPS TO PLANNING THE WAREHOUSE **3**

UNDERSTANDING THE REQUIREMENTS 3
Business Process Model *4*
UNDERSTANDING THE SOURCE DATA 5
Exercise #1: Data profiling *5*
DESIGN THE DATA WAREHOUSE DATA MODEL 6
Dimensional (Logical) Model *7*
Physical model *9*
Exercise #2: Data Modeling *11*
DEFINING THE MAPPING RULES 12
Exercise #3: Source to Target Mapping *13*
IMPLEMENT THE ETL CODING 14
EXERCISE SUMMARY 14

REFERENCES AND AUTHOR CONTACT INFORMATION **15**

REFERENCES AND RECOMMENDED READING 15
ACKNOWLEDGEMENTS 15
BIOGRAPHY 15
CONTACT INFORMATION 16

Abstract

In a previous hands-on workshop (Nelson, 2006), we demonstrated how to use Data Integration Studio to load a sample star-schema. We focused on the usage of the tool itself; however, today we will take a step back and better understand the DI Studio process by outlining a methodology for designing and implementing the ETL load, or the process of moving data from a source system (such as operational systems or a table in a database) into a target (usually a structure that supports analytics and reporting).

This workshop guides the participant through the methodology used to design the target database structure and transformations, create a mapping worksheet used to implement the ETL code, load the metadata, and create the process flows in DI Studio.

This paper connects the dots for those interested in getting started with DI Studio not only as a tool, but how practitioners think about the DI Studio process. Attendees should have a basic understanding of how SAS works; however, no prior experience or attendance at the previous workshop is required.

Introduction

Data warehousing is not about the tools. Rather, it is about creating a strategy to plan, design, and construct a data store capable of answering business questions. Good strategy is a process that is never really finished, rather it is a continuous cycle of enhancements.

While tools like Data Integration Studio work well for helping to design and load the target tables of your data warehouse, they cannot create a plan for the warehouse. Therefore, it is important to understand the steps that go before tool usage.

Ralph Kimball's 38 subsystems (Kimball, 2006) describe the things any ETL strategy must have. In a related paper (Nelson, 2006b); we categorized the subsystems of ETL into 6 key components. These 6 categories included:

- Design and data profiling
- Source data extraction
- Transformation and loading
- Change data capture
- Quality review, auditing and exception handling/management
- Integration with the production environment and business process components

Scope of this Workshop

In this workshop, we look specifically at the planning and design processes practitioners use during the construction of data warehouses. Here, we will go through the following steps. Note, in this workshop not every step has an accompanying exercise, but you will gain enough information to perform the work on your own.

1. Understand your requirements (Primary tool: requirements definition document)
    - Review the questions business users asked during the interview process
    - Review any existing reports or analysis
2. Understand your source data (Primary Tool: DataFlux dfPower Studio)
    - Understand the relationships that exist in the source data (Using Microsoft Visio)
    - Validate the data and its content
    - Perform a gap analysis on what you do and do not have in terms of data.
3. Design your data warehouse data model (Primary tool: Microsoft Visio)
    - Model out your new "schema"
4. Define the mapping rules (Primary Tool: Source-to-Target Mapping document)
    - Create rules for mapping data from the source tables to the target tables
5. Convert your Source to Target mappings into Source and Target metadata (Primary tool: SAS Data Integration Studio)

- In DI Studio, create source and target metadata. Then use the source to target mapping document to build the transformations

The Workshop Data

Please note for continuity the data used for the exercises below is the same data used in the previous hands-on workshop. However, even if you did not attend the workshop, you should be able to go back to your organization and use it today.

Steps to Planning the Warehouse

Understanding the requirements

During the planning and design phase of the data warehouse project, a Requirements Definition Document (also referred to as System Requirements or Functional Requirements Specification) needs to be created.

This document describes the end-user's expectations and needs, IT's expectations and needs, define what you want to try and tackle, create a scope of work or work plan, and lay out models designed to help you understand your problem, the questions you want answered and your data.

Requirements are simple—they describe what the system should do and are often expressed in those terms. For example,

> *The system should print in batch mode and upon request by the user.*

Requirements can be high level. Requirements are a guide through the process of knowing exactly where you are going so you do not get lost.

Well-written requirements can serve as a "contract" of sorts between you, the programmer, and the business because good requirements provide a guide for how you intend to test the system. While we certainly do not suggest that you enter into a "legalized" relationship with your end users, we do advocate the concept of "use cases". Wikipedia provides a nice summary of use cases:

> According to Bittner and Spence, "Use cases, stated simply, allow description of sequences of events that, taken together, lead to a system doing something useful" [Bittner and Spence, 2002]. Each use case provides one or more scenarios that convey how the system should interact with the users called actors to achieve a specific business goal or function. Use case actors may be end users or other systems. Use cases typically avoid technical jargon, preferring instead the language of the end user or domain expert. Use cases are often co-authored by business analysts and end users. Use cases are separate and distinct from use case diagrams, which allow one to abstractly work with groups of use cases.

Once you have a set of requirements, then you can go to the next step. Start gathering the data you have and create a well-structured model to get the answers you want. You can often derive the requirements from interviews with users, examining existing reports, and evaluating existing systems being replaced (if that is the case).

BUSINESS PROCESS MODEL

Goal:

- Define the purpose and decide on the subject(s) for the data warehouse
- Identify questions of interest

The first model that we will use is the "business process model." The business process model (Inmon calls this the high-level ERD or entity-relationship level) helps you understand what business questions you are asking.

Here you focus on what information is available (i.e.) the attributes and relations between them, not how the data should be accessed or organized, nor what it will be used for. For example, in Northwinds Trading Company database[1], try and answer some of these questions during the business modeling process:

1. *Who bought the products (customers and their structure)?*
2. *Who sold the product (sales organization, etc.)?*
3. *What was sold (product structure)?*
4. *When was it sold (time structure)?*
5. *What are the characteristics of the sale (discount, etc.)?*

In this exercise, use these as the requirements. The goal is to build a data structure that provides answers to these questions.

Understanding business processes and asking questions will help you understand how your business is organized. This then begins the development of your functional requirements. Figure 2 illustrates the main subjects found in the operational data.

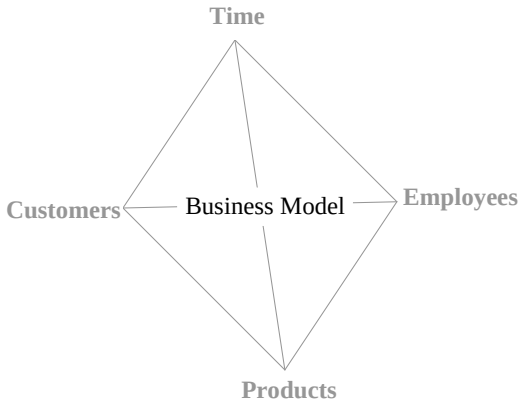

Figure 1. Main Subject Areas

Note the main subject areas are the "subjects" for analysis. The intersection of each line answers questions about our data.

[1] The Northwinds database is a sample database that ships with Microsoft Access.

Now is the time to consider data on our customers or our products already in the system. For example, ask the following question:

What was the total revenue last year from people who were single versus those married?

In order to answer that question, you need to figure out the level of granularity in the data—or how far down the data details go.

When considering a measure such as revenue or sales, it is important to consider what data is available. Does sales information exist for each of my products? For each of the sales reps? Does sales data exist for the last five years?

Think of data multi-dimensionally, as shown in Figure 2. Metrics, such as sales, do not exist in isolation, but rather in the context of layered dimensions, such as Products, Employees, Customers, and Time. Together these dimensions define what type of data is available. Therefore, sales revenue is the metric and it is qualified by the dimensions of Customers, Products, Employees (Sales Rep), and Time.

The other issue is one of *effective dating*. An effective date in your operational system tells you something about the customer on a given date. For example, your data does not have marital status coded for each of your customers. Therefore, even if you had the customer's marital status, you would not have known their marital status for the time period you requested the data.

Understanding the source data

Before spending too much time going off and programming the results, you need to *profile* the –data— generate listings, frequency counts and summary statistics on the fields in the source data systems. In addition, further understand the relationships between tables in your source systems in order to limit mistakes in defining your joins.

In the new world of GUI-based tools, you can get a quick understanding of your data before you make fundamental design decisions. DataFlux, a wholly owned subsidiary of SAS, has created a flagship product, DataFlux® dfPower® Software, that allows you to visually inspect your data quickly and without programming.

Via an intuitive point-and-click interface, DataFlux® dfPower® Software is designed for the business user or IT analyst to support data profiling. This solution allows you to connect directly to your data sources and automatically discover and correct data quality issues.

As a design time tool, dfPower helps you understand important information about the quality of your data and includes column-level statistics (e.g.) count, null & blank counts, min, max, etc., frequency distributions, including pattern counts and uniqueness useful in determining de-duplication/cleansing strategies; as well as relationship analysis and outlier analysis. Relationship analysis helps uncover redundant values across multiple tables (i.e.) highlights orphaned records.

EXERCISE #1: DATA PROFILING

To that end, let's begin exploring your data by profiling the data. In dfPower Studio, you will:

1. Make a connection to the Microsoft Access database (using ODBC)
2. Use dfPower profile to create a profile report for each of the following tables:
    - Categories

- Customers
- Employees
- Order Details
- Orders
- Products
- Shippers
- Suppliers

3. Look for potential data problems
    - Are there patterns in the data lead you to believe you might have to do some cleanup during the load processes? (i.e.) phone numbers with and without parentheses
    - How much missing data do you have?
    - Do you have unsold products in your product table?
    - Are all of the fields conformed? (i.e.) do “State” fields always use the two letter abbreviation
    - Are there outliers in the data? (i.e.) sales figures
    - Are there duplicates in the data? (i.e.) one customer may have two addresses and two customer IDs
4. Evaluate your requirements against what you see in the data
    - Do you have all of the data you need? For example, do you have gender in your customer records? Do you have zip+4 for mailing addresses or other geo-coding information for the mapping requirement?
    - Understand the relationships that exist in the source data (Using Microsoft Visio)
    - Validate the data and its content
    - Perform a gap analysis on what you do and do not have in terms of data

Design the data warehouse data model

Once you examine your data and feel comfortable that you have what you need in order to deliver on the requirements, you are ready to begin modeling the data.

The way that you process information in a transactional system is very different from the way you may want to analyze that data. In OLTP, or Online Transaction Processing, systems such as financial, order entry, work scheduling, and point-of-sale, you want very fast response times with no redundancy and only the most current data online.

In contrast, data required by decision support analysts has a lengthy time horizon, is redundant to the support of varying data views, is often summarized, and is non-updateable. In order to provide data to decision support analysts, relevant operational data is extracted from OLTP systems, cleansed, encoded (i.e.) data and dimensions made to be consistent, and summarized.

After being transformed into a format suitable for decision support, the data is uploaded into the data warehouse. The systems used to analyze data from the OLTP programs are called OLAP, or Online Analytical Processing. The data is organized very differently in these two scenarios.

OLTP uses a relational model in which all the data is broken out into separate tables to reduce redundancy and eliminate duplicate key information, among other things. The process of breaking information out into separate tables is known as data normalization.

OLAP, on the other hand, needs the data efficiently structured for end-user reporting and decision support applications. As a result, the data tends to be redundant. This process is called de-normalization.

Typically the way in which a transactional system (OLTP) is referred is by reviewing its ERD or entity-relationship diagram. Figure 1 illustrates the Northwind ERD. You can easily see the main subject areas and relationships. In Appendix A, you can find the entire ERD for the Northwinds database.

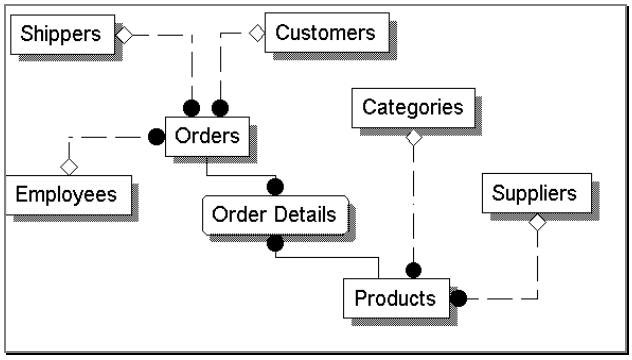

Figure 2. Main subject Area ERD

Figure 2 shows the relationships between tables in your database. Notice that the Order table is where most tables feed. This will be key in deciding which facts or subjects you want to model.

In the analysis tool (OLAP), you want the data to be relatively flat so you can calculate summary statistics without having to perform complex joins whenever you want a simple report. Table 1 shows an example of data that has been flattened out or de-normalized. Note that the Order ID field contains duplicate records, whereas in the relational model, this would be stored in a separate, normalized table.

Order ID	Order Date	Country	...	Salesperson	Extended Price
11044	20-Apr-95	Poland	...	Margaret Peacock	$591.60
11045	20-Apr-95	Canada	...	Michael Suyama	$37.50
...	...	...	...	...	...
11045	20-Apr-95	Canada	...	Michael Suyama	$1,272.00
11046	20-Apr-95	Germany	...	Laura Callahan	$722.00
11047	21-Apr-95	UK	...	Robert King	$480.37
11077	03-May-95	USA	...	Nancy Davolio	$26.00

Table 1. De-normalized data structure

Questions you can ask

DIMENSIONAL (LOGICAL) MODEL

During the analysis of the logical model, remind yourself of the functional requirements. What is it you really want from the information you have about your organization, and how is it structured?

Goal:

- Define your functional requirements
- Confirm the subject areas
- Figure out what the time dimension means
- Identify the granularity (how deep can we go) for your subject(s)
- Create "real" facts and dimensions from the subjects you identified

The next step is to define and agree upon your subject areas identified in your business model. You have developed some questions about the data. This led to categorizing these into your subject areas. Next, you need to understand what time information you have and how far down you want to go with each subject. Here you decompose the data subjects into data entities comprising facts and dimensions. Here you introduce the dimensional model.

A dimensional model is the proposed data modeling and design technique for the structuring of warehouse data. Ralph Kimball (Kimball, 1996) talks about the dimensional model (or start join schema) as a cube. The dimensions of the cube make up the dimensions of your warehouse. By using the image of a cube, the model can be better understood.

"When a database can be visualized as a 'cube' of three, four or even five or more dimensions," Kimball states, "people can imagine slicing and dicing that cube along each of its dimensions."

For example, in your data you know you have customers, products, employees, and time. If you visualize this as a dimensional model, you would first visualize the three primary subjects: customers, products and employees in a cube.

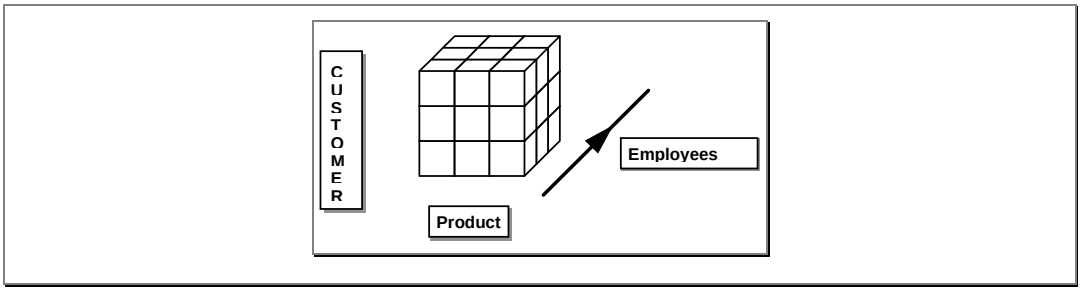

Figure 3. 3-D Model of our Main Subjects

The three dimensional example can be extended to four dimensions by adding a time dimension that indicates the month of the year in which a sale was made. Visualizing a fourth dimension is more difficult.

Imagine 12 boxes depicting the months of the year into which the cube is placed. When the box is placed in the JANUARY box, the cells contain information for JANUARY. When in the FEBRUARY box, the cells contain information for the month of FEBRUARY, and so on.

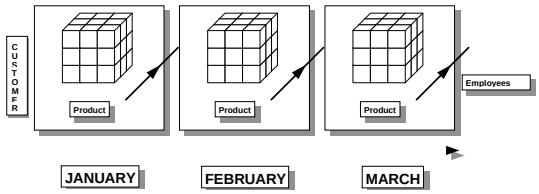

Figure 4. 4-D Model with the time dimension

This paradigm can be extended to five or more dimensions.

Kimball suggests that the data loaded into the warehouse is one of two types: a fact or a dimension. A fact is something that can be measured; they are numeric and generally continuous (e.g.) sales, and are used to calculate a statistic. Dimensions, on the other hand, are pieces of information that categorize things. Examples here include, regions, sales people, fiscal quarter, and so on.

In your data, the metric is sales revenue. You have four dimensions as depicted in Figure 4. At the intersection of two or more dimensions is a fact. For example, consider a two-way intersection: *product B sold to customer 2*.

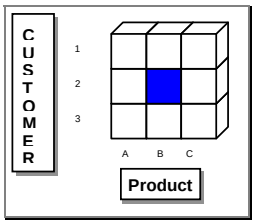

Figure 5. 3-D Model of the Main Subjects

If you calculate a summary statistic, such as sum of sales revenue, you could answer, *how much money did customer 2 spend on product B?*

Obviously, there are more complex questions for your warehouse that involve three or more dimensions. This is where the multi-dimensional database plays a significant role. The ability for users to slice-and-dice the data is done at a dimension level. *Ranging* is when users want to restrict their view to attributes on a dimension.

In this scenario, a user may want only a few values from each dimension (e.g.) I want to know how much revenue was generated by two of my employees from my two largest customers in the beverage and produce categories. Figure 6 shows this situation.

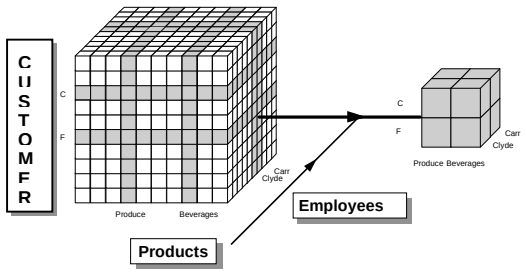

Figure 6. 3-D Model of our Main Subjects

In sum, the logical model helps you think about how your data is organized and whether or not you have the granularity in your data to answer the question(s). The next phase is to develop the physical model.

PHYSICAL MODEL

The physical model in the warehouse design gets into the nitty-gritty of the data available and how should it be stored. The physical design of your data is called a schema. A schema for your data warehouse can be represented by one or more design constructs such as:

- Entity-relationship model
- Star schemas
- Snowflake schemas

- Fact-constellation schemas
- Persistent multidimensional stores
- Summary tables

Others have talked about the use of such modeling techniques (e.g.) Betancourt, 1996; Kimball, 1996; McGuff, 199 and Red Brick, 1995; Raden, 1995, you can review those articles for your own edification.

Goal:

- Define the actual storage architecture
- Decide on how the data is to be accessed and how it is arranged

In considering the data warehouse data model, it is useful to first review the types of tables found in a data warehouse. The three types of tables are:

- Primary Data Tables
- Descriptor Tables
- Characteristics Tables

Primary Data Tables contain both metrics and attributes, and contain the data end-users are seeking. In Figure 7, the Primary Data Table contains the attributes: ProductID, CustomerID, EmployeeID and the metric Sales. In large data warehouses, the full-text attribute description is not stored in the Primary Data Table, but rather in a Descriptor Table. Numeric element ID codes are stored in the Primary Data Table. These numeric ID codes index faster, yield smaller indices, and provide faster "WHERE" clause matching.

Descriptor Tables often contain only two columns, the attribute ID code and the common-English description of the attribute. There is a one-to-one relationship between the ID and the description. These tables replace the ID codes used in queries with common business terms familiar to the user.

In Figure 7, there are three Descriptor Tables which map ProductID, CustomerID and EmployeeID codes to their respective user-understandable business terms. In smaller warehouses, where load performance and storage concerns are less of a problem, text descriptors may appear in the Primary Data Tables. This increases data comprehensibility.

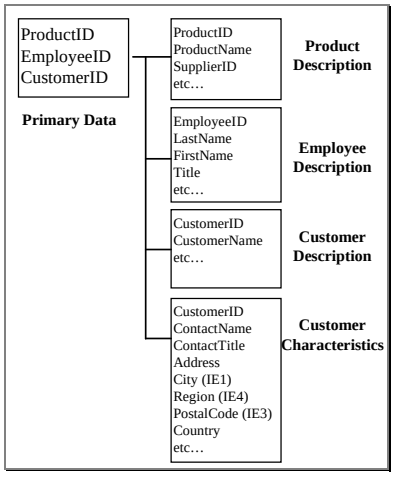

Figure 7. Data model depicting data warehouse tables

Characteristics Tables contain additional information about an attribute and can be used to segment data in an ad-hoc manner. Each column in a Characteristics Table represents an additional attribute used as a filtering criterion in queries.

In Figure 7, there is a single Characteristics Table that contains additional attributes related to the Customer attribute. Using this Characteristics Table, Sales could be segmented by the location of the customer, or any other attribute in the Characteristics Table.

Your task is to develop the databases to help you answer your questions. The schema you use will guide your programming tasks to that end. The star-schema and the snowflake are the most widely use schemas.

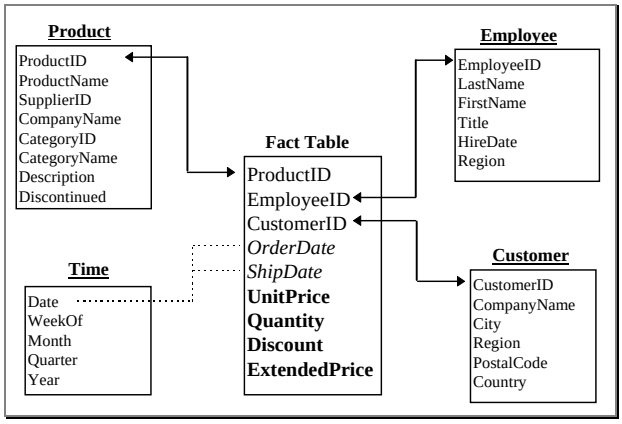

Figure 8. Possible Star-schema model for the Northwinds Trading Company system.

The star schema model in Figure 8 is based on a central Fact Table (Primary Table) surrounded by any number of Dimension Tables (Descriptor and Characteristic Tables). The Fact Table has few columns or variables but many observations. These are linked by Dimension Tables that are typically short and wide, that is, many columns, few observations.

The snowflake schema is similar to the star schema, but it normalizes your data by breaking out the lookup tables (called Outrigger Tables). One example of the data in the snowflake schema is shown in Figure 9.

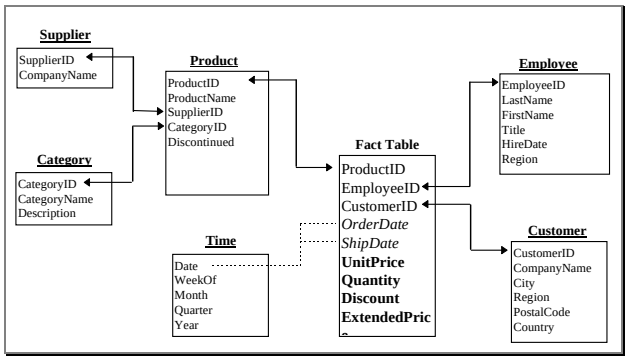

Figure 9. Possible Snowflake schema model for the Northwinds Trading Company database.

Here the snowflake schema is normalized even more than the star schema by breaking out the categories and supplier information into outrigger tables. The normalization of the dimension tables reduces storage overhead by eliminating redundant data values in the dimension table. Normalization comes at the cost of complex queries which are often more difficult to use and implement and require more processing at run-time.

EXERCISE #2: DATA MODELING

Now that you have an understanding of how you want to model your data, you can go through the exercise of modeling using a data-modeling tool. While we would not necessarily consider Microsoft Visio product a

strong data-modeling tool, it is something that most organizations have and does a good job at helping users visualize their data model.

Here you will perform the following tasks:

1. Reverse engineer the data model found in Microsoft Access
2. Create a new tab with your new data model for your data warehouse.
    - Create the Fact Table and a few of the Dimension Tables.
        I. PRODUCT_DIM_TABLE
        ii. EMPLOYEE_DIM_TABLE
        iii. CUSTOMER_DIM_TABLE
        iv. DATE_DIM_TABLE
        v. ORDER_FACT_TABLE
    - After you create a database model diagram, the work of refining the diagram begins. You can add and customize tables and views, create relationships, and customize columns and data types.
        i. Tables
        ii. Columns
        iii. Relationships

The resulting data model should look like the figure below.

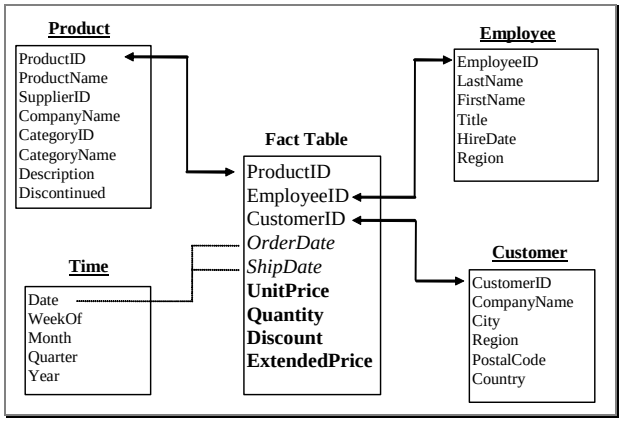

Figure 10. Our target data model.

Defining the mapping rules

Now that you have a solid understanding of you data and you know what the target structure will look like, you are ready to begin to design the transformations. One of the most useful tools for doing that is the source to target mapping document.

This document can either be a text-based document with tables or it can be a spreadsheet. In this document, you will map your target tables and for each of the fields in the target table, we will describe how they are to be populated.

If the data is coming from a source tables (column), then we'll say which column(s) contribute to that field and what business rules should be applied when we load the data. Examples of these transformations could include something as simple as extracting an area code out of the phone number field or converting sales figures to US dollars.

In addition to mapping data from the source systems, sometimes, you have to create new fields. New fields can be completely new, for example, surrogate keys, or they can be derived.

The following screen shot provides an example of a source to target mapping document.

1.1 Name target table

Target Table Information		**Source Table Information**	
Target Table Name	Target table name (E.g. ODSEPOS.options, etc.)	**Source Table Name(s)**	Source table name
		Are there any flat file sources?	! ! ! ! !
RDBMS System / Version	! ! ! ! !	If flat files are they delimited or fixed?	! ! ! ! !
Libref Information	Libref name	Libref Information	Libref name
Will there be any flat file targets ?	! ! ! ! !	Flat files have occurs or redefines	! ! ! ! !
Estimated Number of records	! ! ! ! !	Estimated number of records	! ! ! ! !
Update Strategy	Update as update	Update strategy	Update as update
Refresh Frequency	! ! ! ! !	Refresh frequency	
Target Columns Supply the name of each target column. Attributes <number(8,2) <varchar(30) Whether the column is key Primary key <PK> Foreign key <FK>	**Source Columns** What source table(s) and column(s) are needed for this Transformation? Note: If this target definition requires source columns from multiple tables, describe the relationship between the two (or more) source tables. RDBMS primary key / foreign key relationship or a logical relationship.		Transformation Pseudo Code If a transformation is required for this target column, provide a simple explanation or pseudo code for the transformation. Note: For aggregates, specify grouping rules. If no transformation is required, indicate <*direct map from source*>

Target Field Name	**Attribute**	**Key** **Type**	**Source Database and Table Name**	**Source Field Name**	**Attribute**	**Source Tables Relationship and Join Logic**	**Transformation Requirements** **Pseudo Code to include filters**
SAS field name in bold source field name in regular text Description and Key field if applicable.	Char(xx), Num (xx), Date (xx), DateTime (xx), etc.	PK					Psuedo code and comments or pertinent information.

EXERCISE #3: SOURCE TO TARGET MAPPING

The best way to understand the source to target mapping document, is try to complete one.

In this exercises, you will perform the following tasks:

1. For each of the following tables, create one of these mapping references.
    - PRODUCT_DIM_TABLE
    - EMPLOYEE_DIM_TABLE
    - CUSTOMER_DIM_TABLE
    - DATE_DIM_TABLE

- ORDER_FACT_TABLE

2. As you go through this exercise, pay particular attention to:
    - How you handle surrogate keys
    - Conversion from character to numeric (example date fields)
    - Calculation of derived variables
    - Conformed values
    - Enriched values (adding zip+4 or gender)
    - Handling of slowly changing dimensions (Type I, II or III)

Implement the ETL Coding

Once you have all of the tasks above completed, you are ready to open up your tool of choice. For SAS programmers this could mean DI Studio or it could mean BASE SAS. Regardless, the effort you have put into designing for and planning your warehouse will pay you dividends as you understand your data you have identified any anomalies or problem areas in the data and you have a clear picture of what the transformation rules you should apply when loading the target data model.

For a detailed experience in walking through this process, please refer to the sister hands on workshop cited in the beginning.

Exercise Summary

In the steps above, you started with your requirements documents and worked your way backwards from the reports you intended to develop. You validated that you had the data, it was accurate, had the expected values and relationships between other data that you intended. From there, you designed your new data model and then, using the source to target mapping document, you defined the business rules or transformations that would be implemented when you did your ETL.

It is at this point, where you then take your mapping document and data model to DI Studio for implementation. While it may be tempting to just pop open the DI Studio interface and start programming, you will find if become proficient at thinking through these steps it will save you time and aggravation down the road.

This approach seems rigorous, however it can be very agile with the end users it will become second nature to you and you will appreciate this “thinking through” process before you commit yourself in the tools.

We hope you continue your education by going through the loading of the data warehouse explained in the sister hands-on workshop.

References and Author Contact Information

References and Recommended Reading

Kurt Bittner and Ian Spence (2002). Use Case Modeling. Addison Wesley Professional, 2-3.

Grasse, D. and Nelson, G. "*Base SAS vs. Data Integration Studio: Understanding ETL and the SAS tools used to support it*". Invited Paper presented at the SAS Users Group International Conference. San Francisco, CA. March, 2006.

Kimball, Ralph. The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, 1996

Kimball, Ralph. "*The 38 Subsystems of ETL: To create a successful data warehouse, rely on best practices, not intuition.*" Intelligent Enterprise." December 4, 2004.

Kimball, Ralph and Conserta, Joe. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. John Wiley & Sons, 2004

Kimball, Ralph, Laura Reeves, Margy Ross, and Warren Thornthwaite. The Data Warehouse Lifecycle Toolkit: Tools and Techniques for Designing, Developing, and Deploying Data Warehouses John Wiley & Sons, 1998

Kimball, Ralph and Ross, Margy. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition). John Wiley & Sons, 2002

Nelson, G. "*A Pragmatic Programmers Introduction to Data Integration Studio: Hands on Workshop*". Hands on Workshop presented at the SAS Users Group International Conference. San Francisco, CA. March, 2006.

Nelson, Gregory S. "*Implementing a Dimensional Data Warehouse with the SAS System*." Invited Paper presented at the SAS Users Group International Conference. San Diego, CA. March, 1999.

Acknowledgements

I would like to thank Richard Phillips, Danny Grasse and Jeff Wright for their "thot-ful" and insightful comments on this manuscript.

Biography

Greg Nelson, President and CEO

Greg has recently started his third decade in the SAS eco-system as a programmer, analyst, architect, and teacher. Greg is the President and CEO of ThotWave Technologies where he supports customers in a variety of industries. Prior to ThotWave, Mr. Nelson spent several years in consulting, media and marketing research, database marketing, and large systems support. Mr. Nelson holds a B.A. in Psychology and PhD level work in Social Psychology and Quantitative Methods.

Contact information

Your comments and questions are valued and encouraged. Contact the authors at:

Greg Nelson greg@thotwave.com

ThotWave Technologies, LLC

2504 Kildaire Farm Road

Cary, NC 27511

(800) 584 2819

http://www.thotwave.com