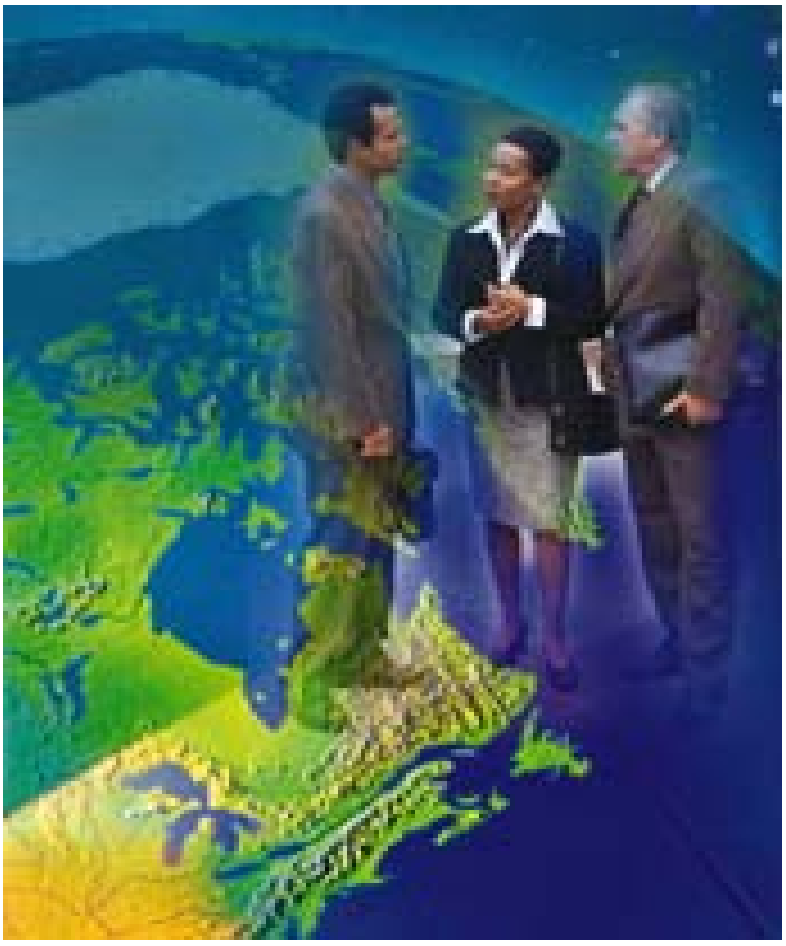
Spectrum IT Consulting, LLC

DATA WAREHOUSE DESIGN



October 19, 2003

Mailing Address:
Spectrum IT Consulting, LLC
9758 Windsor Way
Florence, KY 41042-9202
(859) 992-8969

Office by Appointment:
Florence Executive Centre
7430 U.S. 42
Florence, KY
(859) 992-8969

Purpose

Data warehouses are built for reporting, therefore **data warehouse databases are explicitly designed with reporting in mind**.

Benefits

One major characteristic of a data warehouse is that it reads your company's data from multiple locations and systems and it copies the data into a single location. Another major characteristic is that it arranges the data in a way that provides ease, flexibility, and speed of reporting. That is, if it is designed and implemented correctly.

Once Spectrum helps you collect and arrange your data entities in your data warehouse, you can use one or more reporting tools to produce your reports. You just need to define templates for your standard reports. Today's popular reporting tools are designed so that **end users can create these templates**. The templates can be built through drag-and-drop of any of your data entities. Your reports will contain data grids, and they can even contain charts, etc. When end users look at the reports on a computer screen they can flexibly filter and drill into segments of the data, producing "ad-hoc" reports, which is one of the coolest things about a data warehouse. Today's popular reporting tools allow users to look at reports from a remote location through any standard web browser, if you so desire.

Major Activities in Building a Data Warehouse

The major activities that we will discuss in this white paper include:

- Why to dedicate your data warehouse database to reporting
- Creating entity relationship diagrams
- Universally compatible database design
- Selecting and verifying data sources
- Selecting ETL software
- Designing and developing ETL programs
- Transforming data
- Minimizing needless complexity
- Minimizing data loading time
- Minimize query time
- Storing data
- Understanding dimension tables
- Understanding fact tables
- Understanding granularity
- Storing granular relationships
- Selecting data warehouse reporting software
- Other design decisions

Why to Dedicate Your Data Warehouse Database to Reporting

Data entry databases should not be used for reporting, conversely, **reporting databases should not be used for data entry.** Here's why: A database that is designed to be speedy for data-entry will be slow for reporting and vice versa. This is because they are optimized for speed through incompatible approaches. In a nutshell, data entry databases are fastest when they are "normalized". Data warehouse databases are fastest when they are "denormalized".

To elaborate, reporting systems need to report data quickly. To do so they allow redundancies in the data. These redundancies save the reports from spending time searching for the data when the report is requested. Naturally Spectrum's data warehouse design decisions usually follow this approach. For instance, we typically put a directly related collection of data entities into a single storage structure (i.e., table). One example of a related set of data entities is accounting years, accounting quarters, accounting months, and

days (or fiscal years, fiscal quarters, fiscal months, and days). Storage structures built like this are said to be "denormalized". This increases reporting speed and also increases the data storage requirements.

In contrast, data-entry systems need to update data quickly. To do so they minimize redundant data by pulling out the redundancies and storing them elsewhere. They link them through an ID. For instance, rather than storing the customer's name and address repeatedly with each order, they make a single entry of the name and address and they assign the customer an ID. They then store just the ID on each of the orders. This allows the system to update a subsequent change of address by modifying or inserting a single entry, rather than having to modify many, many orders from that customer. Storage structures built like this are said to be "normalized". This decreases data entry time and also decreases the data storage requirements.

These incompatible approaches result in some of the biggest differences between reporting storage structures and data-entry storage structures.

Creating Entity Relationship Diagrams

To create a solid understanding of entities and entity relationship diagrams, Spectrum holds workshops with **all of the groups that will receive reports from the data warehouse**. We seek **consensus** among all of the groups. Often we try to talk with even more groups within the organization we are helping; not just those that will receive reports. We do this to reduce the possibility of future rework if and when these other groups come on board.

We start the task of creating entity relationship diagrams by **first understanding the data entities required on the most detailed levels of reporting, then determining what levels of consolidation (i.e., roll-up) are required**.

In these workshops Spectrum discusses and documents the data entities that will appear on reports. Our standard documentation consists of a comprehensive, single definition of the meaning of each entity, as well as a few or all of the possible values in each entity. It also shows the relationships between the data entities and whether these relationships are one-to-one, one-to-many, etc. The following entity relationship diagram conveys some of this information:

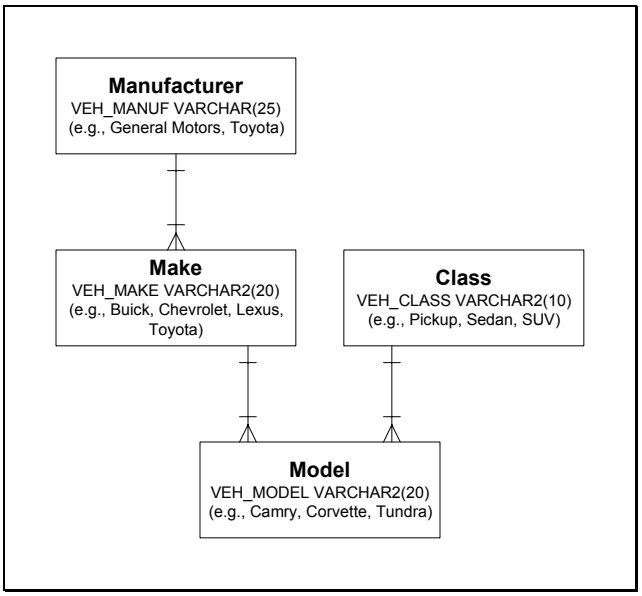


Figure 1: Entity Relationship Diagram

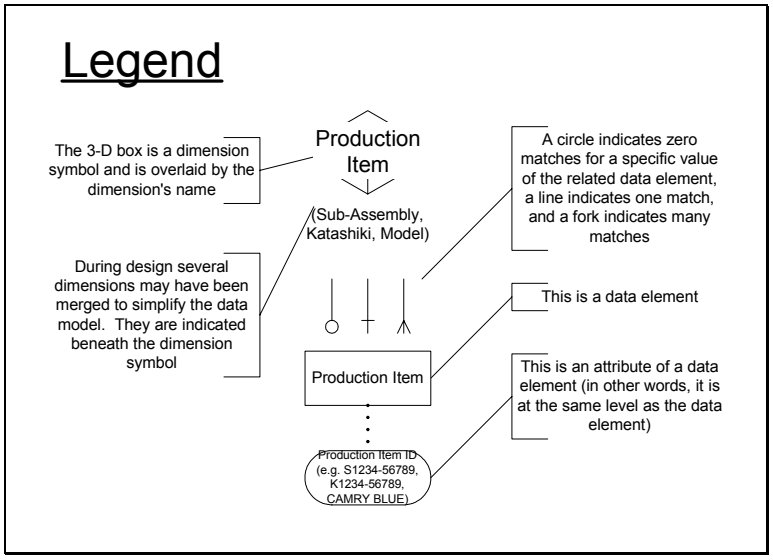


Figure 2: Entity Relationship Diagram Legend

One data entity that Spectrum's clients typically have is "Department". Sometimes there is a data entity under this that departments are broken into for finer tracking, such as a "Cost Center" or "Workgroup" data entity. Typically there will be a "Business Unit" data entity into which the departments roll up. There may also be a "Company" data entity into which the business units roll up. This would be diagrammed like this:

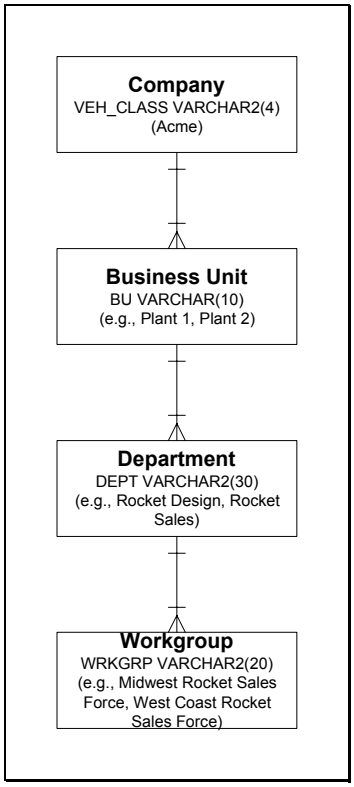


Figure 3: Entity Relationship Diagram

A full understanding of the data entities and their relationships with each other is **critically important to producing a quality database design**, and the design of a data warehouse database is no exception. After gaining this full understanding you will know whether properties are (a) properties of entities or (b) properties of relationships. The **properties of an entity should be stored in a table dedicated to that**

entity. The **properties of a relationship between entities should be stored in a table dedicated to that relationship**.

Universally Compatible Database Design

Unless there is a compelling reason to do otherwise, we **design a database that is universally compatible** with a large number of the best reporting software offerings. We use this approach because it retains compatibility for additional reporting software yet restricts the use of none. For instance, some reporting software assumes that when it uses data from multiple storage structures (i.e., tables), the storage structures should be joined using fields (i.e., columns) with matching names, and not using any others.

Even when the data eventually will be pulled into the proprietary database of a particular reporting software program, it is often best to first collect and organize the data in a neutral database. This leaves a copy of the data in the neutral database where it is available for (1) direct use by non-proprietary reporting software, (2) use as a source to feed data to other systems, and (3) use as a source to feed data to other proprietary reporting software.

Selecting and Verifying Data Sources

We help you verify your intended data sources. Our favorite approach is to manually extract subsets of data from the intended sources and manually prepare proof-of-concept reports using tools like Access, Excel, and Visio. The purposes of this are:

- Verifying that the expected data values are actually present
- Understanding the levels that the raw data are at
- Revealing how the data needs to be consolidated or allocated

Selecting ETL Software

There are many programs you can buy to assist you with your extraction, transformation, and loading (ETL) activities. The main questions to answer are:

- Can the particular ETL software program extract data from all of the sources for which you need it?
- Does the particular ETL software program meet your budget?

Designing and Developing ETL Programs

Often the reports you'll want will need **data sourced from different systems**. When data comes from different systems it usually needs to be brought to a consistent level and matched back together. This leveling and matching is part of the extraction, transformation, and loading (ETL) activities. Designing and developing ETL programs is **usually the most labor-intensive** part of a data warehouse project. It is labor-intensive for two reasons. One reason is that we need to figure out where to get the data for the data warehouse. The other reason is that we then need to figure out how to transform the data into the desired entities and entity relationships that we documented earlier.

Transforming Data

Data transformation **usually involves consolidation or allocation, and then matching**. Once this data is stored in a way the data warehousing tools can understand it, it then becomes straightforward for IS or even end users to design simple reports, and relatively easy to design more complex ones.

Minimizing Needless Complexity

Spectrum strives to **eliminate complexity that doesn't provide value**.

Minimizing Data Loading Time

It is best to reduce time-consuming processes, or at least **reduce any impacts on user response time**. One example is to eliminate full data loads by utilizing techniques that apply incremental changes to the data warehouse data, rather than constantly flushing the storage structures and performing full data re-

populations from the source systems. These techniques should also be applied to the initial load to verify that the techniques are, in fact, valid.

Minimizing Query Time

One example of a time-consuming process is the process of looking up the unit cost of an item in a transaction. This is time-consuming because it requires resolution of inequi joins, which is typical of effective-dated data. The process requires finding the latest effective-dated unit cost that is dated earlier than or equal to the date of the transaction. Finding the cost when loading data rather than when reporting will reduce user response time. The data loading process should find the unit cost, then store the transaction and the unique IDs (i.e., foreign keys) of the cost record.

Storing Data

There are two basic types of data in a data warehouse:

- Transactional data (such as sales orders)
- Entity data (such as customer addresses)

Understanding Dimension Tables

Some of the data in a data warehouse defines how data values are related to each other. For instance, some of the data might list zip codes and which state each one is located in. Other data might list calendar dates and which accounting month each one belongs to. The zip code list and the date list are each something often called “dimensions”. Entity data is stored in storage structures often called “dimension tables”.

If one of our fact tables stores data on sales quantities made each day in each zip code, we could simply report the raw data. The dimension tables allow us to look at the data in other ways too. We could use the zip code list to produce totals for each day and state. We could use the date list to produce totals for each accounting month and zip code. We could use both lists to produce totals for each accounting month and state.

Understanding Fact Tables

Transactional data is data collected from business transactions. Transactional data is stored in storage structures often called “fact tables”. Generally speaking, you can recognize transactional data because it usually contains some type of numeric data that reports will need to sum, divide, etc.

Separate fact tables are used for dissimilar types of transactions. For instance, shipments to customers are a different type of transaction, so they will be put in a different fact table. For instance, purchases that used purchase orders and purchases that used petty cash are similar types of transactions, so they will probably be designed to be stored in the same fact table. Note that although they will be stored on the same table, each of the entries would probably be stamped with its distinct transaction type.

Fact tables contain unique IDs (i.e., foreign keys) that tie them back to records in dimension tables, which were discussed previously. The IDs are used to link back to static or slowly changing entity data.

Understanding Granularity

The most granular entity in a dimension is the smallest data entity, for instance a day is more granular than a month.

Storing Granular Relationships

When relating fact tables to dimension tables, effort should be made to relate each transaction to the most granular data entities in each dimension. This allows you to obtain reports at that most detailed level and at any consolidated level. Naturally, if you store data at a consolidated level you will not be able to look at a more detailed level.

Other Design Decisions

A few of the other design decisions Spectrum will help you make:

- Frequency of updates
- How to capture changes to records that where previously sourced
- Deciding whether and how historical data will be retained (e.g., old sales data is reported using old sales territory boundaries) and/or restated (e.g., old sales data is reported using current sales territory boundaries)
- Deciding when and how to archive old data

