Paper 124-26

Building an Integrated Information Back-plane

Mark Moorman, SAS Institute, Cary NC
Mark Riggle, SAS Institute, Cary NC
Ralph Hollinshead, SAS Institute, Cary NC

Abstract:

In today's world where customers are suppliers, and competitors are partners; where understanding the cause and effect relationships of business is key to survival, is it really smart to have standalone data warehouse systems? The problem with uniting these information systems is how do you integrate all of these disparate business applications together? When a contract means one thing to Sales and something different to Finance; where customer is aligned along one hierarchy in Distribution and another in Marketing; where cost is allocated differently in Traffic than it is in HR, how do you bring them into one integrated system? The answer isn't always easy, but the value is great. This paper will discuss how to build an Integrated Information Backplane (IIB). By properly designing the IIB you will be able to build a system that can grow and accommodate any number of business applications. This paper will cover

Data modeling for integrating disparate systems
Metadata concerns for managing different conditions and hierarchies
Logical modeling for managed growth Data Mart interfaces to an Enterprise Data Warehouse
The Bound Entity method for data modeling

The material covered in the paper will help you

Better understand the challenges of integrating information systems
How to effectively use metadata to control system use
How to integrate a logical model with a physical model by using the Bound Entity method
Be able to build an IIB that will accommodate all of your systems.

Introduction:

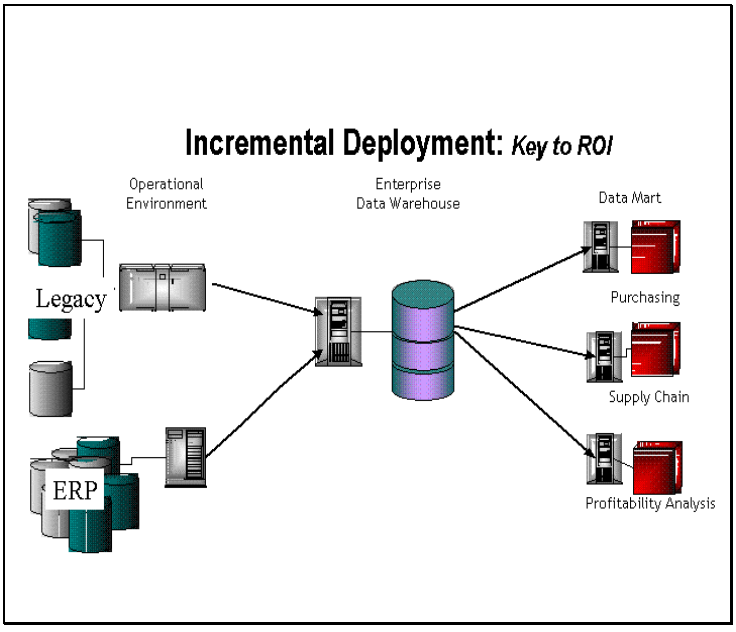
Building an information repository that can handle the business changes of today is a big challenge. The problems are to control the size of the data, while increasing the performance for dimensional reporting, and to manage the change. Dimensional reporting can be a problem because it is generally a full table scan along any number of dimensions. While it is possible to build a model that can perform by indexing along a specific dimension, conforming dimensions across different applications makes it difficult, and often impossible to implement this model at the enterprise level. Combine this with the fact that change is inevitable, and you have problems. To make things even more challenging for today's data warehouse architect, organizations want to integrate all sorts of data into a single model. There are two basic models in use today to handle these problems. These models may come in several variations, but they generally come down to an integrated dimensional model, or a relational warehouse, with dimensional data marts. There are pros and cons to each of these models. In order to best integrate warehouse data in organizations we have developed the Bound Entity Architecture (BEA). The BEA is an architecture that uses a 3rd normal form relational model for the integration, historical data, and the single version of the truth. It combines that with a dimensional model that conforms naturally conforming dimensions while allowing for non-conforming dimensions to drop through to the Enterprise Data Backplane (EDB). It ties the whole solution together with metadata, naming conventions, and a logical model that describes the most basic relationships.

Bound entity Architecture:

The hub and spoke is the basis for the bound entity architecture (BEA). The basic hub and spoke is a mixture of an enterprise data model with dimensional models for reporting. The BEA takes the value of both dimensional modeling for reporting, and the EDB for integration and management, and combines them into a single integrated architecture using metadata. The cornerstone of the BEA, is the binding activity. Each element of the business is modeled at the most basics relationship. It is then possible to take these relationships and build an EDB model that manages the data at a most basic form. In many organizations this is done to cleanse the data, consolidate the data from disparate systems, and to track history. The problem with these systems is they are not generally made an

active part of the analysis and reporting applications. The BEA addresses that by including navigational information of the EDB into the dimensional models so that elements with relationships outside the basic dimensions can be associated. For example, in analysis of procurement an employee (buyer) is usually a part of the dimensional model. Then with the basic conformed dimensions, this should mean that the procurement analysis system should be able to cross over to the HR system on the employee dimension and retrieve further information. This should be possible since these dimensions are already conformed. But what if you want to find out if that employee is a customer and you would like to find out their customer information. These dimensions are more difficult to conform, but if the EDB was modeled correctly (we will discuss that a little later), then you might be able to cross these dimensions at the EDB level, find the key for the customer dimension, and get information on that individual as it relates to customer. The basic Binding rules are as follows:

Basic Data model (3rd normal form)
Naming conventions (this includes suffixes that explain the type of data)
Binding rules for relationships at the 3rd NF model (customer and employee map to individual)
Dimensional binding (employee in procurement relates to an employee in HR)
Metadata attributes to track these relationships



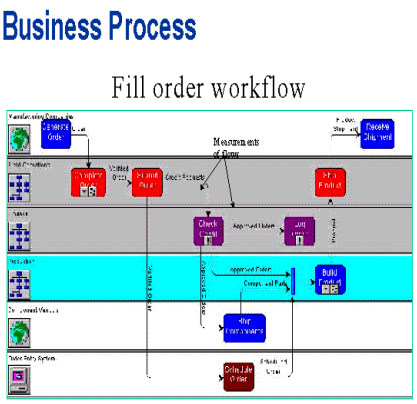
Dimensional Model:

The dimensional model is designed for the explicit purpose of reporting across business categories. In order to see a summarization of all sales by product and month last year, databases are designed such that the facts are aggregated, and keyed by their relationship to these basic business categories such as time, product and channel. The basic dimensional model is a star schema. The star schema is a nice way to represent this relationship of fact to dimension, and while star schemas can be built using any number of technologies from multi-dimensional databases to relational databases, the core model does not change. This is a very de-normalized model, and it uses the relationships of dimensions for the modeling. These models are most effective when they are used with a specific report in mind. Since the indexes and the aggregates can be most effectively designed to support that. A dimensional model is not as easy to use as the backplane to a corporations data warehouse. The problem with a dimensional model is that these relationships have to be described at the primary key level. This can be a problem when data is used at different levels of summarization. Take for instance an order header vs. order header line item. The issue is that the header is at one level of aggregation while the line item is at another. In order to conform those dimensions, the data has to be summed up or distributed down. That makes integration of changing dimensions rather difficult. In order to integrate data in a dimensional model, you have to conform the dimensions. That means that a dimension means exactly the same thing for any fact. At the surface, this seems reasonable. It is only that in practice this can become a massive challenge. Getting every organization to agree on exactly what a dimension means is a big task. There is a solution to this problem. Multiple dimension tables can be built to describe this data in multiple ways. The problem with this is the cost of management. Another way to manage an integrated information repository is to build an Enterprise Data Backplane.

Enterprise Data Backplane:

The term EDB is not really descriptive of the alternative to the conformed dimensional data warehouse, since a conformed dimensional model could certainly be an EDB. The model that we describe as an EDB is one that is based on a

relational 3NF (3rd Normal Form) model



of the data warehouse. A relational model allows more freedom for data modeling and design changes. Because it does not require applying relationships to the data, it can be designed at the most basic level. The data can be normalized, to reduce the amount of redundancy. It can also support many to many relationships. Which will probably exist at this level. Of course the draw back of a 3NF model is that it is not very useful for informational reporting. It generally requires joins and aggregations to get basic reports. To report on the data, data marts are built using dimensional models. The data marts are focused at a specific area, and can be designed for reports instead of integration. Many data warehouse architects believe that this model is more expensive because of the redundancy of the data, and in many cases it may be. In practice the redundancy may not be that great. Dimensional models require several levels of aggregation, and much redundancy to support conformed dimensions. By building the data marts for the specific business needs, performance can be increased.

Binding by business relationships:

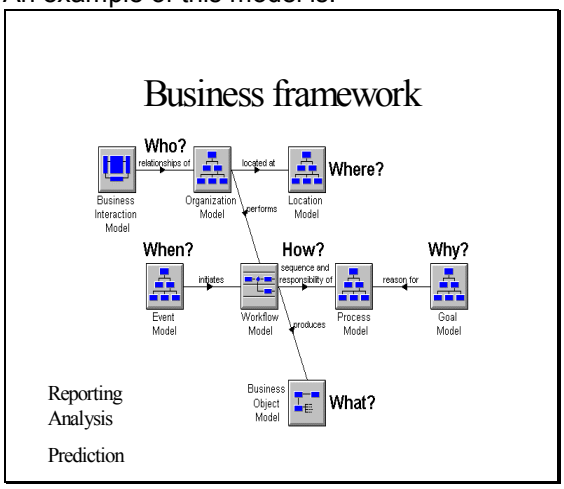
The problem with the BEA, is where do you start. The last thing that organizations need is to wait around for years until the full organization can be designed into an EDB. That is certainly not the thought behind the BEA. The BEA uses business models, and a basic concept of business activators to begin the building process. Business models provide the value, and the process flow that a solution hopes to achieve. To better integrate the solutions, start with the basic business flow. That would describe how information lives in the organization.

As an example of this, let's look at the sales order process above. This is the process that a sales order goes through at a very high level. Most of the process above could be broken into many smaller processes, but this process defines the delivery of goods through the order process to delivery. It is possible then to build reporting solutions that could help predict which customers are more likely to get credit, or which products are more likely to be in stock, or easier to deliver. Each arrow represents a process, and a potential money saving process. The important thing is to locate these areas that predict cost, revenue, or delay. These are the elements of interest for the EDB.

To maximize the solution a process can be defined to help focus attention on the desired outcome. Using a process that defines the business flow by drivers you can define the

Who
What
When
Where
Why
How

An example of this model is:

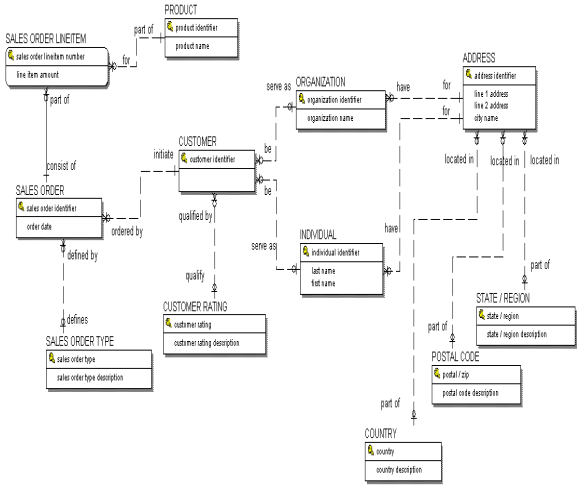


The model will help define the audience, the areas of concern, the relationships in time, the process. The driver will be the why or the business rule. The output will be the what or the results. So then an order generation process might look like this:

In this case the process starts with order generation, or presales activities. The order generation produces an actual order, which might also create a credit check, inventory check, or product development order. The completion of these processes creates a delivery, which will ultimately

create an invoice and a payment. Understanding this process is the key to building information systems. Using this as model, business questions can be answered by locating the areas in this process that affect cost or opportunity. The process also helps to understand the roles each entity has in a process. This model describes the facts by the measures at each process area, and by understanding what causes a split or delay; better business practices can be put in place. This creates a closed loop process with the actual transaction systems.

The activity of building a business model is very important to help organizations understand their relationships at the most basic levels. The politics are generally removed when the business is looked at as a process. From this business model it is pretty easy to build the basic EDB model.



Data modeling concepts:

Techniques for Developing an Enterprise Logical Model and Applications for an Integrated Data Warehouse Architecture

The Role of the Logical Enterprise Data Model and Modeling Techniques

The backbone of strong data architecture for the enterprise data warehouse begins with a logical enterprise data model. This logical model provides the blueprint necessary to develop physical data models at both the enterprise warehouse and data mart level. Since an enterprise warehouse is designed to serve the entire organization, its data structure will look very different from the data mart design that is optimized to answer business questions that are important to one particular business area. It is important, however, that both of these types of models are derived from the same foundation – the logical enterprise model. The following discussion outlines some important considerations when developing the enterprise model and how these considerations are relevant in a data warehouse architecture.

Define Business Relationships:

One of the most important aspects to developing a good logical model is understanding your data and the business relationships. Although the sales logical model is over-simplified, it serves to illustrate the idea of presenting a logical model that clearly shows the data contained in the entities and the relationships between them. For example, a *sales order* is “ordered by” a *customer* and a customer may be either an *organization* or and *individual.* Although, this example seems straightforward, there is a significant difference in a *sales order* that is “ordered by” a customer versus a *sales order* that is “delivered to” a customer, for example. Documenting these business relationships between entities is particularly important in the case of relationships to other entities like *organization* that may play many different roles in regards to other entities in an enterprise model. A *purchase order,* for example, may “originate from” one organization and be “intended for” another organization. In such cases, documenting these relationships adds to the clarity of the overall model.

Simplified Sales Logical Model

Define entities and attributes:

When defining the logical model, long descriptive names are important in order to convey the most meaning possible. Even more essential than the name, however, are the definitions of the entities and attributes. A *customer* in Marketing, for example, may mean "an individual or organization that has inquired about or purchased products from the company," In purchasing, however, a *customer* may mean only individuals or organizations that have actually purchased products. Coming to agreement on definitions for key entities such as *customer* and *product* can be time consuming but are critical in order to create an accurate model.

Model to use entities across subject areas:

Another fundamental concept of the enterprise data model is providing a single source for the development of decision support solutions. These consistent entity definitions are shared across business areas in order to provide for a common base for all functional areas of an organization. For example, the Human Resources subject area contains some of the same entities (organization, employee, address, individual) that appear in the Procurement subject area.

These types of entities should also be explicitly modeled for reuse across subject areas. For example, an *individual* may be both a *customer* and an *employee.* When modeling this information in an enterprise data model, it may be tempting to include both individual attributes (e.g., name, gender, ssn) in both the *customer* and *employee* entity. However, by creating separate entities for *individual, customer*, and *employee*, we allow the same occurrence of an individual to assume the role of either a *customer* or *employee* depending on the context. The same concept applies in the model for *address.* Rather than associating an address directly to a customer we associate the address to an individual or an organization. This allows a single occurrence of address for an organization (although in a real model you would probably include an "address type" as part of the primary key) and allows the organization to play the role of customer in one scenario and vendor, for example, in another scenario. It also allows for a *customer* to be defined as either an *individual* or an *organization.*

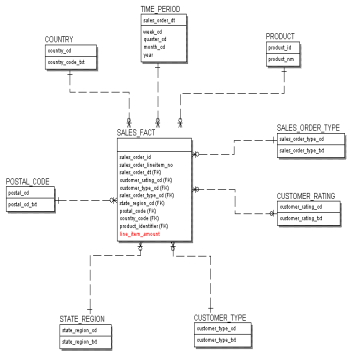
Although not applicable for data structures that will be queried directly, this level of normalization at the logical level is important in order to allow for the flexibility of different roles as well as capture the concept of a "single version of the truth."

Application of the Logical Enterprise Data Model in a Warehouse Environment

Developing a logical enterprise model requires a lot of time and effort. In an OLTP environment the translation of this logical model is more obvious and the benefits are, therefore, easier to see. However, this same blueprint can also be applied to defining data models that are optimized for querying and reporting. In this example, we will use the sales order logical model and translate it to a star schema that is optimized for a data mart environment. The level of granularity for this illustration will be at the sales order line item.

Once the reporting requirements and level of granularity are known, the process of moving from the normalized logical model to the star schema is fairly straightforward. Since the only numeric attribute associated with the *sales order line item* is the *line item amount*, this column becomes the measure for our fact table. Although this is a straightforward example, we see how the same entities in the logical model now serve as dimensions, which "bind" the fact table.

Star Schema Model Based on Logical Enterprise Model



The Benefits of an Enterprise Model Blueprint

Through the development of the logical model, we gain a better understanding of the business through common definitions of entities and attributes and modeling business relationships. By developing a logical enterprise data model, we are able to see the "big picture" and create data models, which will have the flexibility to accommodate different business areas of the organization.

Moreover, this same model can be used as a blueprint to develop various types of physical models depending on their purpose. As we saw in the sales model example, the logical model can serve as the basis for a star schema, dimensional design. Without the logical enterprise perspective, we risk modeling "stove pipe" data marts, which are not compatible with each other. However, if we use the enterprise data model as the starting point, we can easily create data marts with conformed dimensions. Since these data marts are all developed from the same base model, the table definitions, data types, and lengths will be consistent. Most importantly for the end user, this design is much more likely to return consistent results for decision support reporting.

Metadata for dimensional binding:

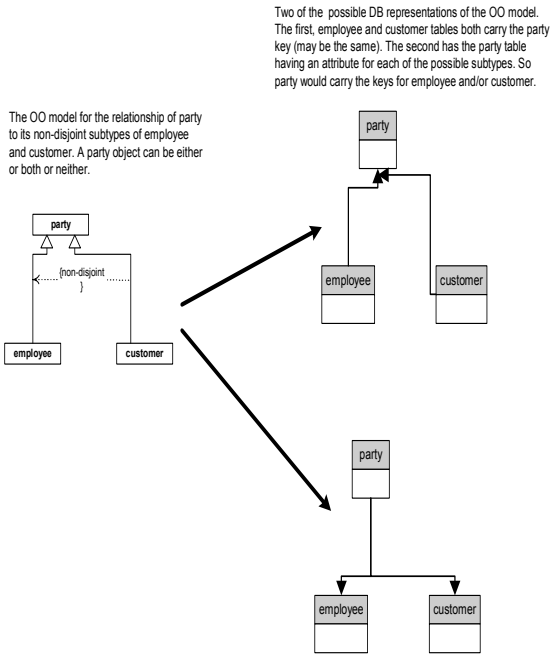
In order to tie these data models and process together and to add information about use, metadata is used. The metadata will help describe the data relationships. It is the technical metadata that drives the tools that interpret the data warehouse data models. The metadata contains the description of the parts and the relationships of those parts in the dimensional models. It indicates the conformed dimensions, fact table relationships and the restrictions on measures that are needed to correctly use and interpret the dimensional models in a business context. The user tools for examining and reporting on the data must be metadata aware in order to utilize the rules implied by that metadata. Hierarchy drill paths are an example of that metadata.

It is the commonality of a dimension between models that allows the combining of the information across those models. This is known as drill across or as cross navigation and the metadata needs to supply that connection.

Until recently there were competing major standards for the metadata representations of the permanent artifacts needed in a data warehouse environment. These metadata representations are called metadata models or metamodels. The Object Management Group (OMG) and the Metadata Colilition (MDC) both had standards relevant to data warehousing, but the OMG is now the carrier of the standard and they are in the process of merging their standard, the Common Warehouse Metadata (CWM) definition, and the MDC standard.

The CWM covers many areas but the part of the CWM that pertains to the metadata use in a dimensionally aware tool is called the OLAP model. It is a high level definition that currently does not support all desired aspects of dimensional models and as a standard it likely will not. A problem is that the meaning and the desired aspects of an OLAP tool are not standardized unlike say relational databases that have an ANSI SQL standard that dictates the model form. As far as the CWM is mainly concerned, variable tracebility and warehouse process control were the constraining requirements thus the OLAP model only needs to be rather general. There is enough in the metadata model however to allow navigation through hierarchies and across dimensions to other models. Of course tool vendors are free to extend the models as needed to supply the desired functionality and this occurs at the price of losing some interoperability. See http://www.cwmforum.org for the details of CWM.

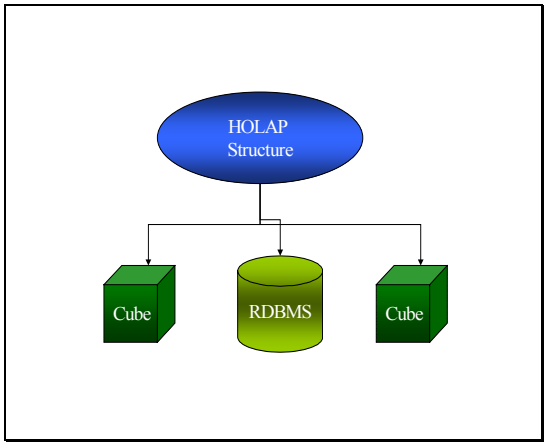
The notion of an extended drill across that utilizes information that connects 2 dimensions through some additional information is not standard. The following diagram shows an example of the deeper connection that may exist in a world model denoted by the object diagram on the left. Some employees may be customers and visa-versa but perhaps not with the same identifer. A common model piece connects the two and it can have several possible relational incarnations. Either one can be used to aid in an extended cross navigation, but the tool or application needs to be aware of that metamodel to use it properly.

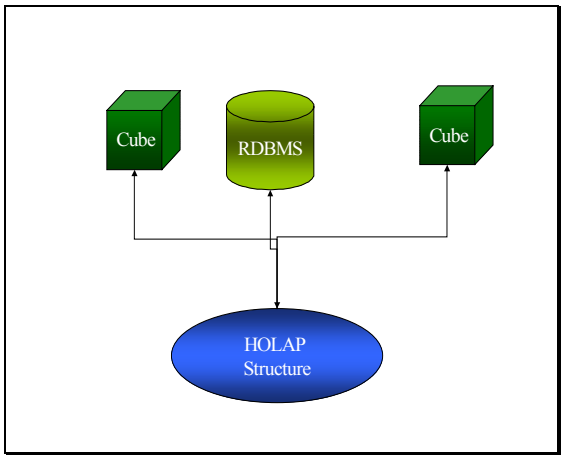


Rules of Engagement:

The question then is how do you build this using SAS Software. Current SAS technology makes all of this possible, but not easy. However, the future looks brighter. With the release of versions 8.2, and 9.0, changes to SAS will make building this solution easier. These changes will mostly come in the form of the Open Metadata Repository, which will help bridge the EDB and the OLAP portions for reporting.

In the current release of SAS it is possible to build this structure, and there are several ways to implement it. SAS/Warehouse Administrator® can be used to build the EDB, and the dimensional data marts. Using the Standards Column Add-in, the names can be standardized for binding. Extended attributes can be added to the metadata to keep the binding relationships. We recommend using a list for each relationship. This offers the ability to have multiple relationships from the conformed dimensions to the EDB. HOLAP can be used in several ways. Initially it might very well be used for the dimensional models themselves, but it will make up the dimensional backplane. A HOLAP structure is used to define the conformed dimensions. This structure encompasses every dimension, and has 1 empty cell. Generally HOLAP structures are defined from the top down. This means that the first interface is with the HOLAP structure, but with this solution, the HOLAP is from the bottom.





With the BEA, the HOLAP structure sits under the reporting OLAP cubes to maintain the relationship a dimension has to other relationships. Using this technology lets us create drill across in SAS.

Conclusion:

Building an integrated data warehouse is very difficult, but it is possible. By using the best of both an EDB and dimensional models the integrated informational backplane can work. The EDB ensures integration at the lowest level, one version of the truth, and historical data management. The dimensional model makes reporting and analysis easy. By binding these two models together with the BEA a complete system can be built. Basic modeling rules and metadata are used to tie the whole solution together. The final outcome will be an expandable solution that will make each new report easier.