Paper 118-25

Data Warehousing Design Issues for ERP Systems

Mark Moorman, SAS Institute, Cary, NC

ABSTRACT

As many organizations begin to go to production with large Enterprise Resource Planning (ERP) systems, the need for reporting and analysis on this information has grown. This paper will discuss the issues associated with designing a data warehouse for an ERP system. This paper is intended for people with a basic understanding of both data warehousing and ERP systems such as SAP AG, Oracle, PeopleSoft etc. This paper will cover the following topics.

- Design issues for a data warehouse from ERP
- Extraction using SAS/ACCESS® interface to SAP R/3 and SAS/ACCESS interface to Baan.
- Management using SAS/Warehouse Administrator™
- Loading issues for database design

Attendees should leave this paper with a firm understanding of how to extract, transform, and load data from the major ERP systems into a SAS data warehouse.

INTRODUCTION

In the not too recent past, corporations used homegrown systems for their backbone transaction processes. However, some recent dynamics have created a boom in the packaged applications market known as ERP. One of the major dynamics was the Y2K issue, but there were several other reasons for this increase.

- Modernize aging applications that were economically beyond salvage
- Address the need for new business requirements not supported by existing systems
- Address the Y2K problem
- Achieve better competitive advantage

These systems have created a unique dynamic in building data warehouses. In some ways they are easier, and in other ways they are much harder. With in house systems, the knowledge of how they worked, and how they were set up was generally readily available. The down side to the in house systems were that they grew incongruently. Generally they were created over time, and with different resources, so there was not always a clear integration line to them. The ERP systems offer the inverse of this. They generally create a level of generality and consistency across business lines, but usually they are not as well understood in house. This dynamic alone can account for most of the new issues with building data warehouses from ERP systems. The other issues are basic data warehousing issues.

BASIC DESIGN

The basic design of a data warehouse can be very intrusive. Some organizations spend the bulk of their time and money on just this, and almost plan failure. Other organizations plan very little, and often run upon failure after initial success. It is important to plan for the future, but with an eye on change. There are two basic flavors to a data warehouse.

1. Enterprise Data Warehouse (EDW): A data repository built around the data for the purposes of analysis and reporting.
2. Data Mart (DM): A data repository built around application needs for the purpose of analysis and reporting.

The enterprise data warehouse promises integration and growth but brings with it the headache and heartache of trying to be everything to everyone. The data mart design is easy at first, but can't satisfy different request. In a large corporation (and you probably are a large corporation if you are using an ERP) both are needed. The recommended design for large integrated warehouse for ERP systems is to use an EDW for the integration backplane with a DM at the report/analysis level. This may seem a bit redundant, but the value is well worth it.

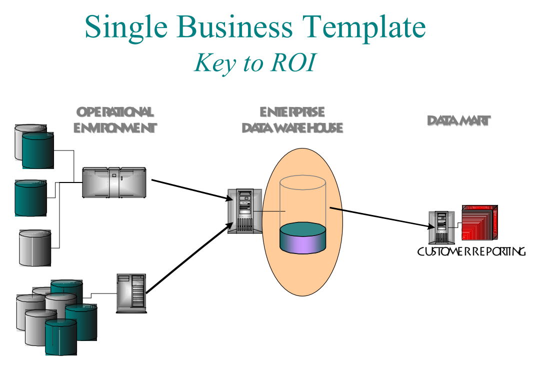


Figure 1.0 EDW to Data mart

The value of this design might not seem readily understandable, but the value comes in integrating several data marts. Today more and more companies are looking for new angles on information. This requires data that has historically been segregated, to be integrated into a single report or analysis type. One such question might be what kind of background makes for the most successful Account Rep. By mixing HR and Sales data, that question might be answered. It is also true that these operational systems are the heartbeat of a corporation. They can not be impacted drastically by reporting or data mart builds. With the EDW backplane all of the data needed can be extracted once, and those systems can be freed up to get back to business while the DM's can be populated from the EDW. One final issue is that this method can create another level of consistency. Reducing the number of extracts and transformations from the ERP will eliminate some of the problems with different results.

ENTERPRISE DATA MODEL

To create this consistency in the EDW, it is a good idea to design an Enterprise Data Model (EDM). This model is nothing more than a piece of metadata that describes entities and attributes about your corporation. This process is a very valuable exercise for many reasons, but specifically for the purpose of defining a context for everything. One of the fallacies about an EDM is that it is all-inclusive, and that is just not true. It is very possible to grow an EDM over time if it is designed properly. We use a very simple EDM design. It is no more than corporate entities and the attributes that best describe those entities. The purpose of this is to get a consistent model for multiple DM's. What this becomes is a definition of the columns that will be used in your EDW. It does not generally include the tables, because it might be added to several tables. It should however restrict at least somewhat columns that appear in the EDW. The EDM can be added to as needed, but the EDW should gather only those columns described in the EDM.

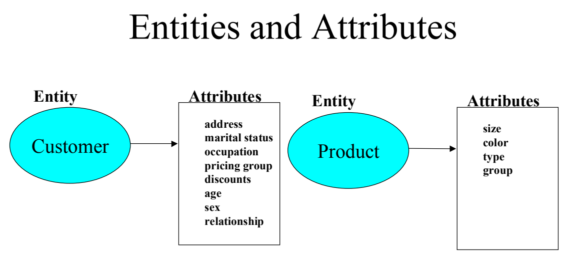


Figure 1.1 EDM designer

These corporate entities and attributes can then be further described for specific data as needed.

ENTERPRISE DATA WAREHOUSE

By using the EDM to define consistency across columns, it is possible then to define the Enterprise Data Warehouse (EDW) using these columns. For the purpose of this paper the EDW is used to describe the actual data representation with both tables and columns of the EDM. When extracting data from ERP systems, there are a couple of things to keep in mind. While ERP systems are designed to integrate transaction systems, they are not designed as Data Warehousing systems. Therefore, it will be important to extract this data and transform it. However, the transformation stage is shortened, and where possible we use structures from the ERP system. We designed the EDW around the entities in the EDM, but those entities in the EDM can reflect the ERP representation of data. This helps to maintain an understandable flow for the users of the data warehouse. Therefore, if your company uses SAP R/3, then it wouid probably be better to use the name material than to use product. With the EDM in place these naming issues are made moot. If you are integrating multiple systems, and would like to maintain different names for different data marts, then we recommend using a generic column name, and then labeling it with more ERP specific names. There are several basic issues that the EDW should support. Those covered here are

- Slowly changing dimensions
- Change data capture/integration
- Surrogate keys
- Hierarchies

There may be more issues than these, but these are the most requested from us. Let's look at each.

SLOWLY CHANGING DIMENSIONS

The EDW model has been designed to support slowly changing dimensions in several ways. Slowly changing dimensions are those values that exist in data that change over time, but are not considered transactions. Things like Customer Address, Age, and Marital Status may all be considered slowly changing dimensions. Depending on the type of reporting desired, these changes may have to be tracked different ways. There are 3 basic ways that the EDW supports slowly changing dimensions. These 3 ways may be referred to as types. The 3 types are:

- Type 1: treat current status of slowly changing dimension as old. This means that if someone gets married during the year, reporting on that person will show him or her as married for all historical reporting.
- Type 2: Treat current status as a new record. This means that if someone gets married during the year, that they have two master records indicating two different sales entities.
- Type 3: track status over time. This means that a customer's status can be given at any specific time.

Each Data Mart may require that slowly changing dimensions be tracked differently. The EDW is designed to support all three of these types. At the DM level it is going to be necessary to decide which columns will be tracked which way, but in order to support them all, the EDW should have some understanding of how to build that in. Our EDW is designed with a different key for each change that might occur on a record. This is an exact replication of the type 2 slowly changing dimension. This key is a composite key based on a unique identifier of the lowest key, and a current flag. It is then possible to pull type 1 dimensions by using the current flag. Another part of this composite key is a version or date. This makes it possible to support type 3. This composite key is added to the transactions as they are updated in the EDW. By selecting the whole composite, you will retrieve the data for type 2 dimensions. To select type 1 or type 3, only use the part of the composite that is needed. Our data model has a vector added to each of the tables with slowly changing dimensions on it. This vector is a group of single character columns that correspond to the columns on which you are tracking changes. If you are tracking changes on marital status, then there would be a single character column variable that would contain a 1 or 0 on that same record to indicate rather this column changed on this record. This is done, because there might be several columns on which changes are being tracked on this table, and each change produces a new record. Therefore, it is not possible to determine which column changed on that record. By using the vector it is now possible to determine which columns have changed. This can make for interesting analysis. It is now possible to count the number of changes to a dimension very easily, or to relate one change to another. To analyze how changes affect each other.

CHANGE DATA CAPTURE/INTEGRATION

Change Data Management can be defined as the total process of finding records that have changed in a transaction system, and applying those changes to a data warehouse, such that those changes might be tracked in a reporting system. This process will include two steps; one of which is Change Data Capture and the other is Change Data Integration. For this document, Change Data Capture can be defined as the process of finding records that have been changed in the transaction system.

The objectives of Change Data Capture are rather simple. The idea is to find all of the changed information as quickly as possible with the least amount of overhead. There are two basic reasons for Change Data Management.

- Faster Data Warehouse Loading
- Managing History

The first issue of Change Data Management is to reduce the load time of adding data to the Data Warehouse. To completely refresh the data in a Data Warehouse each time new data is added can be very expensive in wasted processing. The second reason is to manage history. Data Warehouses in many cases are used to keep history data. In those cases, a complete refresh does not work because all of the data needed is no longer stored in the transaction system. Therefore, Change Data Capture has to capture data for three different events.

- New data
- Change data
- Deleted data

There are also two distinct types of transaction data that require change management.

- Dimensional
- Transactions

These two data types will actually require different types of change data capture, because they are used very differently. Dimensional data are those values that describe data points such as customer and product hierarchies. As mentioned before it might be important to track the changes of this data for reporting and analysis purposes. It is also probable that this type of data will not have new data added as often as transactional data, but it will have changes more often. The transactional data is different however. Transactional data refers to the values that are accumulated such as orders, and invoices. This data generally has more new records than dimensional data, but has fewer changes. A change to this data is usually a correction of a problem, and therefore not often tracked. To track these changes it will be important to use some of the feature/functions of the ERP system. For our example we will use SAP R/3. The other systems offer similar functions.

Capturing Changes: SAP R/3 has several ways to track data. There is even a system specifically for checking changes in SAP R/3. However, due to complexity, and performance issues, it might not be active, or useful. There are five possible ways to capture changes to SAP R/3 data.

- SAP's change management system
- Change dates on the table
- Incremental Primary Key (transaction)
- Database triggers
- Comparisons to historical data

Each of these four ways has its good and bad points. Let's look at each individually.

SAP R/3 Change Management: This might be the best way to detect changes in the SAP R/3 system, but it offers the most problems also. The SAP R/3 system is rather complicated, and difficult to understand. The complexity of this system also can affect performance. Some companies might not turn it on for transaction type tables, because of the system impact.

Change dates on the Table: Many SAP R/3 tables have change dates columns on the records. This is especially true for the dimensional tables, but it is not true in all cases. In some cases, there is no change date. The other problem with this is that most tables are not indexed on the change dates. Therefore, this would require a total table read. This would affect performance failing one of the objectives of Change Data Management. It is possible to index these columns, but many companies might not feel comfortable doing that.

Incremental Primary Key (transaction): The incremental primary key is a way to find new changes to transaction data. Since transactional queries are only interested in new values, and since new values are generally applied with an incremental primary key, it is possible to fetch only rows greater than the last fetch's highest primary key. This works very well for transactional data, but it does assume that changes and deletions are not being tracked.

Database Triggers: Another way to track these changes is by using database triggers. Since SAP R/3 generally runs on a database that supports triggers, it is possible to put triggers on the R/3 tables that need changes tracked. The trigger would kick out the primary key for each of the rows changed. This key could then be used to join in the changes. The downside to this is that it happens outside of SAP R/3, and that may not sit well with some customers. It also requires database interfaces, which may differ from site to site.

Comparison to Historical Data: As a finally choice, the SAP R/3 table could be compared to the Data Warehouse table for changes. This has several downsides, and should only be used as a last resort. It reads every row and column of the transaction table and the data warehouse table, so it fails the performance objective. It also requires that the databases in the transaction system and the data warehouse be defined exactly alike.

In our approach to this we are using several of these methods in concert. It might be best to use the SAP R/3 Change system for those tables that it is turned on for, use the change dates for the tables that have them, and use the incremental primary keys for the tables that have those. This would leave the database triggers and comparisons as last resorts.

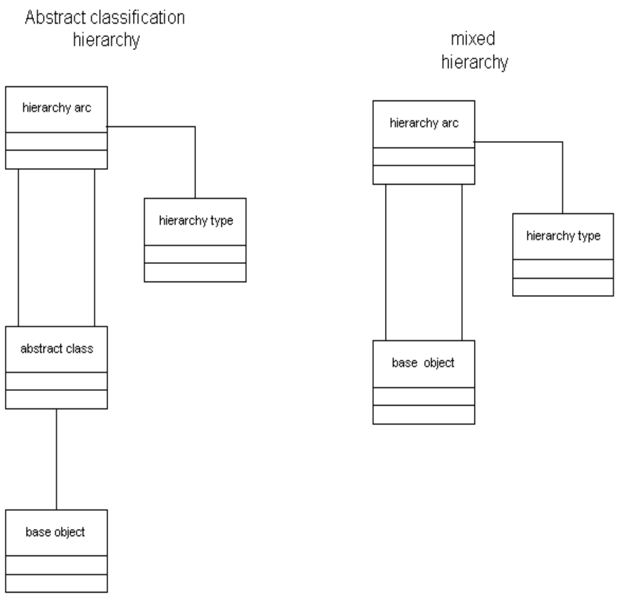
SURROGATE KEYS

Surrogate keys are an interesting issue when considering ERP systems. Most data-warehousing specialist will recommend using surrogate keys to conform data from different systems. Surrogate keys are new keys that replace the transaction keys from the ERP. They are generally defined incrementally, and are used to help integrate systems that might have duplicate keys. The issue with surrogate keys and ERP systems is that most ERP systems use complex composite keys. These composite keys create a rather difficult problem. At what level do you define the surrogate keys? We decided to define them at the primary level, then use the surrogate's primary keys to rebuild the composite keys. The problem with this is determining the primary key level. That is not always as easy as it seems. Since different modules might define keys as lower levels. We use the lowest level defined in the module of the ERP system that we are using.

HIERARCHIES

Hierarchies are those classifiers with relationships. They can be very difficult to represent at the EDW level. Similar to the problem with slowly changing dimensions. Hierarchies have to be decided at the DM, but they can be used differently by different DMs. To make this problem even more difficult to understand, each of the ERP systems includes several ways to store hierarchies, and some can be very confusing, while others are rather limiting. SAP R/3 for instance uses at least two different hierarchy structures for Materials. The first is a very simple character string with a length of 18. The first 5 characters make up the parent hierarchy level. The second 5 characters make up the second level of hierarchy, and the last 8 characters are the lowest level of hierarchy. This is simple enough, but very limiting. The other system that they use is much more complex, but more scalable. This system uses classes, and parent child relationships to build an arc system. We have taken this basic idea, and created a more general design. This design for representing hierarchies has a single place to hold all hierarchy information for any entity. An instance of the hierarchy pattern should be used in each place of the EDW where a hierarchy is needed. The hierarchy arc table has the following fields with the following names so those later exploitation pieces can be parameterized.

- PARENT – type depends on object but default to C20
- CHILD – as above
- TYPE – default to C20 – this may be unused for some hierarchies.
- VALID_FROM – start date where this is a valid arc
- VALID_TO – end date where this is a valid arc (do our part for the 10K problem, use 9/9/9999 for still valid, or missing)
- Other needed fields required by the hierarchy modeled.
- Language component should be pushed to the hierarchy type if needed.



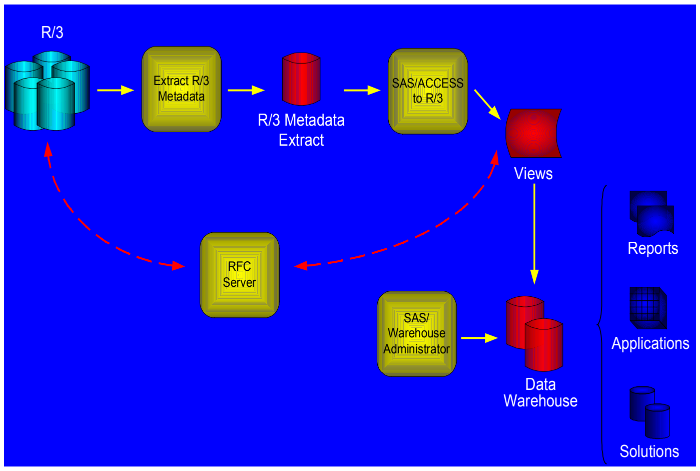
With this design, we are able to build simplified hierarchy solution for each entity.

EXTRACTING ERP INFORMATION USING SAS/ACCESS®

Currently there are SAS/ACCESS® products for SAP R/3, and Baan. There are others in the works for the likes of PeopleSoft and Oracle. Most if not all ERP systems store data in an RDBMS such as Oracle, SQL Server7, or DB/2. While SAS offers access to these base systems, there is a layer of metadata that exist at the application level that is necessary to make sense of this data. These SAS/ACCESS® products designed specifically to make use of this metadata. With these products it is possible to read the business rules that tie these RDBMS tables together. While these products do the same thing, they do it very differently.

SAS/ACCESS TO SAP R/3®

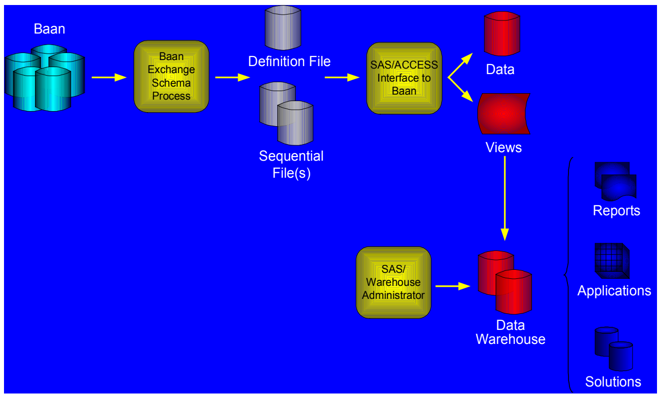
For access to SAP R/3, SAS extracts the SAP R/3metadata into a SAS view. There is a GUI shipped with the access product that allows searching and basic navigation of the SAP R/3 data. Once a piece of data is requested, an RFC server is activated to retrieve that data directly from SAP R/3.



This solution works quite well, because it allows access to all of the SAP R/3 data and even supports ABABI programs. ABABI is the programming language for SAP R/3

SAS/ACCESS TO BAAN®

The mechanism for Baan does the same thing, only different. The access to Baan product uses the Baan exchange schema process to create an exchange file and definition file.



The exchange file can then be read into a SAS view using the definition file.

WAREHOUSE ADMINISTRATOR

The SAS/Warehouse Administrator is a great tool for building and deploying Enterprise Warehouses. One of the strengths of this tool is the tight integration with the ERP access products. From SAS/ACCESS for SAP R/3, there is an export option, which will export the SAP R/3 view directly into the warehouse environment. This is an extremely valuable tool since we use the access views as the ODDs in our warehouse environment. It exports the short descriptions as labels, and the long descriptions as notes. This is very helpful, since the notes now describe in detail what each column is. SAS/ACCESS to Baan® has a plug-in to the SAS/Warehouse Administrator. This makes it easy to include the Baan views as the ODDs. The warehouse environment includes ODDs that are direct replicas of the ERP tables being used. It has Subject areas that are directly related to our business entities described earlier. These Subject areas are made up of the master and hierarchy data structures described earlier. There is a Data Group for all the DMs. This Data Group has a Data Group for each of the business areas for which we have included DMs. We break up the DMs into business areas. Business areas are logical groupings of DMs such as Sales and Human Resources. By using this logical grouping it is possible to share a DM. Finally, there is a Data Group for the specific business application under each of the business areas. This Data Group might contain the DM, or it may just contain the reports as Information Marts. The actual DM might be at the business area level, or it might be at the specific application area.

CONCLUSION

With the growth of ERP applications, it is important to make sure that you are getting the most benefit possible from this investment. It is important to build a different structure for the reporting and analysis to fully recognize the value of this information.

ACKNOWLEDGMENTS

I would like to thank and acknowledge many people, but most specifically thanks a credit goes to
Kerrie Brannock - for an unlimited knowledge of SAP R/3, and Change Data Integration design
Mark Riggle - for all kinds of wise council, but most specifically for the data models used in this paper.

CONTACT INFORMATION

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

Mark Moorman
SAS Institute
SAS Campus Drive
Cary, NC 27513
Work Phone: (919)677-8000x6522
Fax☹919)677-4444
Email:mark.moorman@sas.com
Web:www.sas.com