Creating The Data Warehouse Data Model
From The Corporate Data Model

BY

W. H. Inmon
The point of departure for the building of the data warehouse is the data model. Without a data model it is very difficult to try to organize the structure and content of data in the data warehouse.

Many organizations have recognized the importance of the data model over the years and have invested the time and effort to build such a model. However, classical data modeling techniques make no distinctions between operational and decision support environments. Classical data modeling techniques merely try to gather and synthesize the informational needs of the organization. The result is the corporate data model.

The corporate data model is a very good place to start the process of building a data warehouse. However there is some amount of work that needs to be done on the corporate data model in order for it to be readied for the building of the data warehouse. A certain amount of transformation must occur to create the data warehouse data model from the corporate data model.

**THE CORPORATE DATA MODEL**

In order for the transformation from the corporate data model to the data warehouse data model to occur, the corporate data model must have been built and must be in a state of readiness. In particular the corporate data model must have identified and structured - at least - the following:

- the major subjects of the enterprise,
- the relationships between the subjects,
- the creation of an ERD (entity relationship diagram),
- for each major subject area:
  - the keys(s) of the subject,
  - the attributes of the subject,
  - the subtypes of the subject,
  - the connectors of one subject area to the next,
  - the grouping of attributes.

Figure 1 identifies the minimum components of the corporate data model.

![Diagram of data model components](image)
There may be much more to the corporate data model than that outlined in Figure 1. For example, the corporate data model may include analysis about processes as well. Process analysis typically consists of:

- functional decomposition,
- data/process matrices,
- data flow diagrams,
- state transition diagrams,
- HIPO charts,
- pseudocode, et al.

The process analysis is interesting but usually is only an adjunct to the corporate data model because the process analysis applies directly to the operational environment, not the data warehouse environment. It is the corporate data model that forms the backbone of design for the data warehouse, not the process analysis.

The corporate data model is usually broken into multiple levels - a high level and a mid level. The high level of the corporate data model contains the major subject areas and how they relate. Figure 2 shows a simple example of a high-level corporate data model.

In Figure 2 there are four subject areas - customer, account, order, and product. There is a direct relationship between customer and account, between account and order, and between order and product. Of course there are many indirect relationships that are inferred from the high-level data model. Only the direct relationships are shown. Note that the high-level data model does not contain any amount of detail at all. At the high level, detail only clutters up the model unnecessarily.

The next level of modeling found in the corporate data model is that known as the mid level of modeling. The mid level of modeling is the place where much of the detail of the model is found. The mid level of modeling contains keys, attributes, subtypes, groupings of attributes, and connectors. Figure 3 shows a simple example of a mid level model.
Figure 3 shows that for an order there is a certain amount of basic information, such as order date, who made the order, who received the order, and so forth. There are different types of order such as commercial orders and retail orders. Each order may have multiple occurrences of line items that have been ordered, and may or may not have specific order information such as who to call on delivery.

(Note: for a much deeper explanation of data modeling - conventions, techniques, methodologies - refer to INFORMATION SYSTEMS ARCHITECTURE: DEVELOPMENT IN THE 90's, QED, Wellesley, MA.)

There is a relationship between each subject area identified in the high level model and the mid level models. For each subject area identified, there is a single mid level model, as shown in Figure 4.
The corporate data model may cover a very wide scope. When it does, the corporate
data model is called the enterprise data model. Or the corporate data model may cover a
restricted scope. When this is the case, it is called simply the "corporate data model." Either case - that of the enterprise data model or the corporate data model - is adequate
for starting the process of creating the data warehouse data model.

Note that in many organizations the data model is not fleshed out to the same level of
detail. Some mid level models are completely designed and fully attributed. Other mid
level models are only sketched out, with little or no detail. The degree of completion of
the larger corporate data model is of little concern to the data warehouse developer
because the data warehouse will be developed a stage at a time. In other words it is
very unusual to develop the data warehouse on a massive frontal assault, where all mid
level models are developed at once. Therefore the fact that the corporate data model is
in a state of differing degrees of readiness is not of a concern to the data warehouse
developer.

Figure 5 shows that the data warehouse will be built one step at a time.

![Figure 5: Each major subject area has its own mid-level model.](image)

First one mid level model is transformed and readied for data warehouse design. Then
another mid level model is transformed and so forth.

**Making The Transformation**

Assuming that there is a corporate data model to begin with (and if there isn't, the
developer should investigate the possibility of generic data models such as those offered
by PRISM Solutions as a point of departure), the transformation process is ready to
commence. There are a few basic activities that make up the corporate data model to
data warehouse model transformation. Those activities are:

- the removal of purely operational data,
- the addition of an element of time to the key structure of the data warehouse if
  one is not already present,
- the addition of appropriate derived data,
- the transformation of data relationships into data artifacts,
- accommodating the different levels of granularity found in the data warehouse,
• merging like data from different tables together,
• creation of arrays of data, and
• the separation of data attributes according to their stability characteristics.

Each of these aspects of data model transformation will be discussed.

**REMOVING OPERATIONAL DATA**

The first task is to examine the corporate data model and remove all data that is purely operational. Figure 6 illustrates this activity.

In Figure 6 it is seen that most data found in the corporate data model finds its way into the data model for the data warehouse. However, some attributes of data such as call to the attention of upon delivery, phone on delivery, and after hours call apply only to the operational environment. These attributes should be removed before the data warehouse data model is built.

The removal of operational data is seldom a straightforward decision. The decision always centers around "what is the chance the data will be used for DSS?" Unfortunately circumstances can be contrived such that almost ANY data can be used for DSS. In other words, with a fertile imagination, almost any data qualifies as being applicable to DSS. A more rational approach is to say "what is the reasonable chance that the data will be used for DSS?"

The argument that can always be thrown up is that one never knows what is to be used for DSS. DSS always involves the unknown. On that basis, ANY and ALL data should be kept. However, the cost of managing volumes of data in the data warehouse environment is such that it is patently a mistake not to weed out data that will be used for DSS only in farfetched or contrived circumstances.

A tradeoff must be made here for the reasons discussed.
**Adding An Element Of Time To The Warehouse Key**

The second modification to the corporate data model that must be made is that of adding an element of time to the data warehouse key if one does not already exist. Figure 7 shows the addition of the element of time to the data warehouse key.

![Diagram](image)

Figure 7: Adding an element of time to the data warehouse data model.

In Figure 7 effective date has been added to a customer record. The corporate data model has specified customer and customer information. But in the warehouse snapshots of data are made. The effective dates of those snapshots are added to the key structure. It is worth noting that there are many different ways to take these snapshots and there are many different ways to add an element of time to the data warehouse key. The technique shown in the example is only one of many such techniques. (NOTE: refer to the PRISM SOLUTIONS TECH TOPIC on time dependent data for a complete discussion of this aspect of structuring data.)

If data identified in the corporate data warehouse already has an element of time as part of the key structure, then there is no need to add the element of time to the data warehouse key structure.

**Adding Derived Data**

As a rule data modelers do not include derived data as part of the data modeling process. Consequently, corporate data models do not contain derived data. The reason for the omission of derived data is that when derived data is included in the data model, that the data model will grow to ungainly proportions and the data model will never be complete.

The next transformation that must be made to the corporate data model is that of adding derived data to the data warehouse data model where appropriate. Figure 8 shows the addition of some derived data.
It is appropriate to add derived data to the data warehouse data model where the derived data is:
- popularly accessed, and
- calculated once.

The addition of derived data makes sense because it reduces the amount of processing required upon accessing the data in the warehouse. In addition, once properly calculated, there never is any fear in the integrity of the calculation. Said another way, once the derived data is properly calculated, there never is the chance that someone will come along and use an incorrect algorithm for the calculation of the data, thus enhancing the credibility of data in the data warehouse.

Of course, any time that data is added to the data warehouse the question must be asked - "is the addition of the data worth it?" The issue of volume of data in a data warehouse is such that every byte of data needs to be questioned. Otherwise the data warehouse will quickly grow to unmanageable proportions.

**Creating Relationship Artifacts**

The data relationships found in classical data modeling are for the operational environment. Those relationships assume that there is one and only one business rule underlying the relationship. For the assumption that data is accurate as of the moment of access (i.e., operational data), the classical representation of a relationship is correct. However, for a data warehouse there usually will be MANY business rules between tables of data. This is because data in a warehouse represents data over a long spectrum of time and there will naturally be many business rules over time. Thus the classical representation of relationships between tables as found in classical data modeling is inadequate for the data warehouse. Relationships between tables in the data warehouse are achieved by means of the creation of "artifacts".

An artifact of a relationship is merely that part of a relationship that is obvious and tangible at the moment the snapshot of data is made for the data warehouse. In other
words, when the snapshot is made that causes data to enter a data warehouse, what part of the relationship of the data on which the snapshot focuses is useful and obvious.

The artifact may include foreign keys and other relevant data. Or the snapshot may include only relevant data and no foreign keys.

This subject is one of the most complex subjects facing the data warehouse designer. An example is in order. Consider the simple data relationship shown in Figure 9.

Figure 9 shows that there is a relationship between a PART and a SUPPLIER. In the example, each PART has a primary SUPPLIER. This relationship is typical of that that might be found in a corporate data model. Integrity constraints dictate that if a SUPPLIER be deleted, that not PART exists that has the SUPPLIER as the primary source. The relationship represents an ongoing relationship of data that is active and accurate as of the moment of access.

Now consider how snapshots of data might be made and how the relationship information might be captured. Figure 10 shows some snapshots of PART and SUPPLIER data that might appear in the data warehouse.
The PARTs snapshot table is one that is created periodically - at the end of the week, the end of the month, etc. Much detailed information about a PART is captured at this time. One of the pieces of information that is captured is primary SUPPLIER as of the moment of snapshot. This then is an example of an artifact of a relationship being captured. Note that the relationship is accurate as of the moment of capture. No other implications are intended or implied.

Another type of snapshot is that shown for the capture of SUPPLIER information. Unlike PART information which changes rapidly - each time a PART is received or shipped - SUPPLIER data is much more stable. The SUPPLIER snapshot is made every month or every quarter. The artifact of the relationship that is captured with SUPPLIER is the last PART supplied. Note that if a SUPPLIER has provided multiple PARTs, that only the most recent one shows up here. This then is another example of an artifact of a relationship as found in a data warehouse.

Both of the snapshots discussed have one major drawback - both are incomplete. They show only the relationship as it exists as of some moment in time. Major events may have occurred that are never captured by the snapshots. For example, suppose the PART snapshot is made every week. A PART may have had three primary SUPPLIERS during the week, and the snapshot would never catch it. Or, for example, suppose the SUPPLIER snapshot is made monthly. A SUPPLIER may have made fifteen different shipments of many different PARTs, but the snapshot would show only the latest shipment made.

Snapshots are easy to make and are an essential part of the data warehouse, but they do have their drawbacks. If it is desired to capture a complete record of data, a historical record rather than a snapshot is required for data in the data warehouse. Figure 11 shows an example of historical data in a data warehouse.
In the PART history table, a shipment has been received at the loading dock. Upon receipt, relevant information is recorded. Among other things, the SUPPLIER of the PART is recorded. This then is another form of artifact relationship information being recorded inside a data warehouse. Assuming that ALL deliveries have a historical record created for them, the record of the relationship between the two tables is complete.

The same is true for the SUPPLIER history table. The historical record is written out whenever a SUPPLIER receives an order. At this point the artifact of the relationship is captured (among other things.) The totality of the orders received by the SUPPLIER provides a complete record of the relationship.

**CHANGING GRANULARITY OF DATA**

One of the features of a data warehouse is that of different levels of granularity. In some cases the level of granularity does not change as data passes from the operational environment to the data warehouse environment. In other cases the level of granularity does change as data is passed into the data warehouse. When there is a change in the level of granularity, the data warehouse data model needs to reflect those changes.

Figure 12 shows the changing of the level of granularity as data passes into the data warehouse.
Figure 12 shows that in the operational environment there is shipment activity data that is gathered each time a shipment is made. For reasons known to the data warehouse designer, data granularity is changed as the data passes into the data warehouse. Two summarizations of shipment data are made - the monthly summarization of total shipments, and the summarization of shipments made by source/location.

The issues in the changing of granularity (insofar as the data warehouse data modeler is concerned) revolve around:

- what period of time is the data to be summarized upon? (i.e., summarized by day? by week? by month? etc.)
- what elements of data are to be in the summarized table?
- will the operational environment support the summarized data elements? i.e., has the data warehouse designer specified data in the warehouse, which cannot be calculated from the operational data source).
MERGING TABLES
The next transformation consideration is that of merging corporate tables into a data warehouse table. Figure 13 illustrates such a merger.

In Figure 13 are three tables from an engineering/manufacturing environment. The tables are normalized. But as they are placed in the data warehouse environment they are merged together. The merger saves both space and performance.

The conditions under which a merger makes sense are when:
- the tables share a common key,
- the data from the different tables is used together frequently, and
- the pattern of insertion is roughly the same.

If any one of these conditions is not met, it makes sense to NOT merge the tables together.

CREATION OF ARRAYS OF DATA
The next transformation activity is the consideration of the creation of arrays of data in the data warehouse data model. Data in the corporate data model is usually normalized. This means that repeating groups are not shown as part of the data model. But under the proper conditions the data warehouse can and should contain repeating groups.
Figure 14 shows an example of a data warehouse data model containing arrays of data.

The corporate data model has shown that the accounts receivable record is created on a month-by-month basis. But as data goes into the data warehouse, it is organized into an array so that each month of the year is an occurrence of the array.

There are several benefits to this structuring of data. One is that by not having individual records of data for each month, which a certain amount of space is saved. In the data warehouse case, the value - ACCOUNT and YEAR appears once for each year, while in the case of the corporate data model, those values appear twelve times for each year. The savings of this space may not be trivial at all. In some cases it amounts to as much as 25% of the total space required for the table. In addition the data warehouse structuring of data requires one-twelfth the index entries as the corporate data model structuring of data.

The other advantage is the possibility of organizing all yearly occurrences of data in a single physical location. By having all occurrences in a single place, there is the possibility of performance being enhanced. Whether this turns out to be a significant factor depends on many factors, such as the usage of data, the dbms the data is organized under, the physical organization of data under the dbms, and so forth.
The creation of arrays of data is not a general-purpose option. Only under the correct circumstances does it pay off to create arrays of data in the data warehouse data model. Those conditions are:

- when the number of occurrences of data are predictable,
- when the occurrence of data is relatively small (in terms of physical size),
- when the occurrences of data are frequently used together, and
- when the pattern of insertion and deletion is stable.

One of the interesting aspects of the data warehouse is that since the key structure of the data in the warehouse always contains an element of time, and since the units of time occur predictably, then the techniques of arrays of data in a data warehouse table are peculiarly appropriate. Said another way, there is a strong affinity between the technique of creating arrays of data in a single table and the data warehouse.

**Organizing Data According To Its Stability**

The final transformation technique is that of organizing data in the data warehouse according to its propensity for change. The corporate data model makes little or no distinction in the rate of change of the variables contained inside a table. But a data warehouse is very sensitive to the rate of change of data within the warehouse. The optimal organization of data inside a data warehouse is where data in one table all changes slowly and data in another all changes rapidly.

Figure 15 shows that the corporate data model has gathered some data for customer. That data is then divided into three categories - data that never changes, data that sometimes changes, and data that often changes. The data warehouse data structure finally ends up with structures that are compatible, in terms of volatility.
An illustration of how this transformation works is shown in Figure 15.

**THE ORDER OF APPLYING THE TRANSFORMATION CRITERIA**

The order in which the transformation criteria are applied is as presented - with the removal of purely operational data first and with the grouping of data according to stability last. Of course, as with every design process there is a certain amount iteration that occurs. The order that the transformation criteria are applied is not set in concrete. However, as a general guideline, the criteria should be applied as presented.
SUMMARY
The corporate data model is the basis for building the data warehouse. But the corporate data model needs a fair amount of design activity before it is ready to be turned into a design for the data warehouse.

The data warehouse data model is created from the corporate data model by going through the following design activities:
- removing all purely operational data,
- adding an element of time to the warehouse key if one isn't already there,
- adding appropriate derived data,
- creating relationship artifacts,
- accommodating the granularity changes in warehouse data,
- merging tables where appropriate,
- specifying arrays of data where appropriate, and
- organizing data according to its stability.