The Fundamentals of Metric Driven Data Warehouse Design

By John M. Artz, PhD
Associate Professor of
Information Systems
The George Washington University

Disclaimer: This book needs a good re-read and a little patching in a few places. However, despite these, hopefully, minor flaws it should be fairly solid. So I am making it available in its current form until I have time to go over it and fix the flaws. If you encounter something that you think needs to be fixed right away please send me an email at jartz@gwu.edu.

Table of Contents

If Nature Had Been Comfortable, Mankind Would Never Have Invented Architecture - Oscar Wilde

Chapter 1: The Evolution of Data Management Paradigms for Information Processing

What is a Database?

This chapter will examine the history of data management paradigms and show how embedded in the two dominant database technologies of today (relational database and data warehousing) we can find the foundations of data management. It will answer the questions - 1) how are data warehouses different from relational databases; and 2) why are data warehouses and relational databases the foundations of data management.

Let's begin with the question - What is a Database ? A traditional definition is that a database is a collection of logically related data elements that may be organized in various ways to meet the multiple processing and retrieval needs of an organization or individual. And while that is a perfectly usable nuts and bolts view of a database, there is something essential to the nature of a database that is missed, entirely, by this definition. Consider the following somewhat exaggerated scenario.

I have a small degree program with about thirty students in it at any one time. In order to advise the students, I need to keep track of which courses they have taken and the grades they received in those courses. So I set up a small database on my laptop with students, courses and grades in it. As an added bonus I find that I can not only use it for advising, but I can use the information to clear students for graduation. The data is managed by Microsoft Access and it serves my processing and retrieval needs. It is unlikely that anyone would challenge the claim that this is a database.

Now, let us assume that my laptop has begun acting erratically. I am not sure what the cause is, and I do not have money in the budget for a new one so I have to figure out some way to preserve my student database. In a moment of inspiration, I decide to print the records out on labels which I can then affix to index cards. Further, I can use yellow cards for student information, blue cards for courses and pink cards for grades. Even though there is no budget for new equipment we always have an abundance of teaching assistants so I ask a few of them to organize the cards and put them in trays to facilitate access to the cards for advising. They do, and when I request the information on a student, they flip quickly through the cards and retrieve the information that I need. The question is – Is this still a database?

Most people would say yes. The data is the same logically related form that I had in the MS Access database and with the help of the teaching assistants it still meets my information and processing needs. I can still advise students and I can still clear them for graduation. Now, let's take this a step further. Suppose I normally keep the card file database in my office on the first floor, and on days that I do advising, I carry it up one floor to the advising office on the second floor. Unfortunately, one day, I am running a

little late and in my hurry I trip on a stair and drop the card tray. The cards slide all over the stairwell. Since there is already a long line of students waiting for advising, I decide to set up shop in the stair well. I recruit a couple more teaching assistants who sort through the cards in the stairwell and retrieve the cards that I need. Though the cards are in disarray and it does take a little longer to find the cards I need, I am still able to advise students and feel fairly confident that I could use the cards in the stairwell for any other processing tasks that I might have. It is not optimal, but it is workable. The question we return to is – Is it still a database?

Some people at this point might be reluctant to call my spilled cards in the stairwell a database. And certainly any of my teaching assistants locating cards in the stairwell would really be stretching it if they claimed, based on their efforts in the stairwell, that they had database experience. But the only thing that has changed is the time and cost required to retrieve records. If that is the key factor separating databases from non databases, then any database that slowed down as it accumulated more records would be in danger of becoming a non database. So let's push this a little further.

On the first floor, the stairwell opens onto a courtyard where students congregate between classes. A student, unaware of the situation, opens the door to the stairwell from the courtyard side allowing a burst of wind to sweep into the stairwell and blow a number of my cards from the stairwell into the courtyard. I don't have a spare teaching assistant to collect all of the cards in courtyard so I instruct them to look in the courtyard if they cannot find the yards they are looking for in the stairwell. This further slows down the process but seems to work nonetheless. Is it still a database?

Clearly this example could be taken to absurd extremes if those extremes have not been reached already but it goes to the heart of what we mean when we refer to something as a database. So let us go just one step further. It begins to rain. The cards in the courtyard are getting wet. More and more students come into the building through the stairwell to get out of the rain. The steps are getting wet and a little muddy from their shoes. The cards are getting wet and muddy. Yet they are still readable and can still be located. Is it still a database?

Some people would say it stopped being a database when I printed the data out on cards. Others would hold out claiming that as long as I can still use the data for the purposes for which it was intended, it is still a database. So the definition of what makes something a database is not completely clear. In fact, there is a lot of gray area.

Now let's consider a different scenario. In this scenario I begin with the same student database in MS Access but instead of fiddling with the physical representation of the data and the time to access it, I am going to fiddle with the domain which the database represents.

Again, I have a student database that I use to keep track of students and the courses that they have taken. As my degree program becomes more and more popular I have more and more students from other degree programs taking my courses as electives. Then in

order to capitalize on the growing demand for my courses, the school decides to offer a set of four classes to students who wish to have a certificate in my area of specialty. The certificate becomes increasingly more popular and a professional version of the certificate is made available to anyone, not just students, who wish to learn more about my specialty. Now, I have to keep track of degree seeking students, students from other degree programs, students seeking certificates and professionals seeking certificates. My student database is no longer adequate for keeping track of who has taken what. Nor is it adequate for varied new kinds of processing that I must do. Is it still a database?

Before we try to answer that question let's push things just a little further. Let's assume that a centralized university system has taken over the task of keeping track of which students have taken which classes and this system also performs other necessary functions such as clearing students for graduation or certification. I receive a data file from the central system every semester that I use to keep my student database current even though I now longer use it. Is it still a database?

We saw, in the first example, that a database can move gradually into non database status as the physical organization of the data deteriorates making it more difficult for the database to serve its purpose. In the second case we saw the logical organization of the data deteriorate. The entities in the domain changed until the database no longer represents the things of interest in the application domain. We also saw the purpose of the database change as its key functions were no longer necessary. So we are left with the question – can an organized collection of data that does not model anything of interest and does not serve any purpose really be a database?

We can push this question just a little further. Imagine you find a laptop sitting on a park bench. In order to find out who it belongs to, you turn it on and look for some sort of identifying information. But there is none. In fact, the only thing on the laptop other than a standard suite of office software is an MS Access database. So you open the database. In it you find a table entitled members. But instead of names and addresses all the members table has are some funny kind of nickname in one column and a rank that is impossible to make sense out of in the second column. There is an achievements table that has a level, some funny sort of code word for the achievement and what appears to be a city or county, presumably where that level of achievement or initiation took place. Finally, there is a payments table which consists of dates and amounts. The point is that there is no way to tell from the database or the laptop what the data refers too. Is it still a database? Is it possible to have a collection of electronic database managed by a database management where the data does not refer to anything and still call it a database?

Database as a Model

Our earlier definition that a database is a collection of logically related data elements that may be organized in various ways to meet the multiple processing and retrieval needs of an organization or individual is a little naïve. Perhaps a more sophisticated way to look at a database is to think of it as a model of some aspect of the world that we wish to know more about. We model that aspect of world in data and when we have questions about

that aspect of the world we interrogate the database rather than having to interrogate the world. This notion that the things in the database must correspond with things in the world is the reason why our student database seems to be less of a database when the underlying domain changes. It is also why the database on the laptop seems to fall short of being a database. In the case of the laptop, anyone who felt that it was still a database probably did so because they believed that at some point in the future we would discover the meaning of the data.

One final point in defining the nature of the database is that a database must be designed to serve a purpose of some kind. As a practical matter, database purposes are often implicit rather than explicit and this is often to the detriment of database design. However, consider the student database that we described earlier. When student records were centralized, the local student database became superfluous. Granted it still received updates from the central system. But as the central system began to provide more and more of the necessary features, the local student database was no longer useful. It has no purpose and hence it was less of a database. The notion of purpose is implicit in the original definition which asserted that a database must meet the multiple processing and retrieval needs of an organization or individual. However, it should be made explicit that the database should have an explicit purpose. So we can revise our original definition of database to say that a database is some aspect of the world modeled in data to serve an explicit purpose in providing information about that aspect of the world being modeled.

A database management system, by comparison, is a software product that supports a specially trained user with the task of creating, accessing, and maintaining data in a database. In today's world the overwhelming majority of databases are relational databases and the overwhelming major of database management systems are relational database management systems. Hence, it is easy to confuse the database with the database management system. In fact, people routinely learn how to use database management software and then consider themselves competent at creating relational databases. But the fallacy in this view can be easily exposed with a simple analogy.

Suppose a person wishes to learn how to write short stories so they take a class in creative writing. As they learn to write, they may write long hand. They may type their stories. Or they may use a word processor. Any of these means will do and, indeed, different writers have strongly held differing preferences for one of these media. Some even dictate their word into a recorder and have someone else type it. But if someone were to take a class in using a word processor, they would fall woefully short of being a writer. There is an intellectual skill required for writing and while the software may make that task easier, knowing how to use the software is not the same as having that intellectual skill. The same is true for databases. There is an intellectual skill required in designing a database, but knowing how to use database management software falls way short of that skill. One may know that you have to create or manipulate tables and may know how to create those tables, but it is the content of those tables that is the design challenge. Analogously, if one is using a word processor to write a story they may know that the story consists of paragraphs and they may know how to create or manipulate paragraphs in the word processor. But it is the contents of those paragraphs that makes

the story. Sometimes technicians who are familiar with a particular database management system feel that they can create a database if only someone were to tell them what the database should contain. That would, of course, be like someone who knows how to use a word processor thinking that they could write a story if only someone would tell them what to write. It is the intellectual skill of designing a database that we need to work on and that is why we need to step so far back from the technology.

The Evolution of Data Models

Going all the way back to the Pre Socratic Philosophers Heraclites and Parmenides we can see two starkly different views of the world that still persist today. Heraclites saw the world as in a constant state of flux. His famous observation that you cannot step in the same river twice captured his underlying belief that the fundamental nature of reality was a constant state of change. This makes sense intuitively. If I take the temperature or measure the flow volume of the river today, everything will be different tomorrow. Different temperature, different flow rate, different water, different river. And it isn't just the river that changes. The car I am driving today is not the same as the car I bought ten years ago even though it is the same make and model. It still has the same vehicle identification number and the same license plate. In fact, it still looks very similar. But the car I drive today requires more maintenance. It uses a little oil. The gas mileage isn't as good as it once was. The floor mats are worn out. The finish does not shine like it used to and the paint is nicked on the door panels. Each time I drive the car, it wears a little more, gets another nick, or is that many miles closer to needing maintenance again. So, we could take the view of Heraclites and say that you cannot drive the same car to work two days in a row.

Parmenides countered with the claim that if everything is in a constant state of change it is not possible to know anything. If I agree that I cannot drive the same car to work two days in a row, how can I ever draw conclusions about whether or not it is a good kind of car to buy. How can I ever figure out if it is better to change the oil every day, every month or every year. How can I accumulate knowledge about which noises mean maintenance is required and which noises are just a product of aging. So, in Parmenides' view, it is the same car despite any small changes that may occur and treating it as the same car allows me to accumulate knowledge about it. In fact, Parmenides was so adamant in his view that allowing change would destroy knowledge that he claimed change was actually an illusion. One of his followers, Zeno of Elea, offered the following paradox. If I shoot an arrow at a target, before it gets to the target it has to get halfway to the target. From there it has to get halfway again. Since there are infinitely many half ways that the arrow has to reach, it is not possible for the arrow to reach the target in a finite amount of time. Hence, the fact that it appears to reach the target must be an illusion. And, hence, all change must be an illusion.

Taken at the extremes both Heraclites and Parmenides look pretty silly. Saying that I cannot drive the same car to work two days in a row just doesn't square with my experience. Maybe it isn't the same car in every minute detail but it is close enough. Further, to say that the house I live in today is not the same as the house I lived in

yesterday and the friends I have today are not the same as the friends I had yesterday just doesn't seem to make sense. Yet, taking a step back, changes are occurring. My house changes in market value. It needs painting and repairs from time to time. So given enough time, it is not the same house. At what point did it go from being the same house to not being the same house? Similarly, Parmenides looks silly at the extreme. If the arrow never reaches the target, then, using the same logic, when I drive to work, I can never get to work. I can never finish mowing the lawn. If I take a class in database the class will never end. Something is very wrong with this view of reality.

But, if we stay away from the extremes these two views of reality and stay closer to the center we see that they reflect some fundamentally important features of reality. If things are static, then we can organize then into kinds. These kinds form categories which allow us to ask questions such as how many instances of a thing do we have in a particular category. How many of the instances in a category have a particular features? How likely is it that members of one category are members of another category?

Alternatively, if reality is in a state of flux we can ask how a thing is changing. Instead of assigning it to a category we watch it over time as it changes. We can ask questions such as how is it changing? What factors cause it to change? Are some factors more influential under certain circumstances than others? Can we influence the rate of change?

In research, we have names for the two different kinds of data that are gathered from these two different views of reality. The first is snapshot data and the second is longitudinal data. Snapshot data describes a situation at a moment in time and allows us to ask categorical questions such as those described above. Longitudinal data has a temporal component which means that data is collected over time. It allows us to ask questions of the second kind regarding how things change over time. If the data collected represents measurements, then we can express those changes over time more economically. But the distinction is not limited to research. We have the same dichotomy in databases which one should expect because, after all, databases are all about data. Typically, relational databases contain data of a categorical nature while data warehouses contain data of a temporal nature. And, since this distinction may not be obvious to users of relational databases some justification and explanation is in order.

Typically, relational database design begins with an information model in which entity classes are defined. Entity classes are nothing more or less than categories into which the data in the database is organized and that represent a relatively static view of the world. Let's say we have an entity model of a university containing entity classes for students and courses. To go back to Heraclites' view of reality we could argue that the same student never shows up for the same class twice. After each class session the student is, hopefully, more knowledgeable about the subject matter. The student may have greater or less enthusiasm for the subject matter. Other events may have occurred in their lives such as a job or a new girl friend or boyfriend. The course changes also. The instructor updates some of the material. Adjustments may be made to take into consideration the makeup of a specific course section. And yet, we ignore all of these changes to preserve the stability of our view of nature. The student takes a course and gets a grade.

We can see further support for the notion of a relational database as a categorical model of reality in Structured Query Language (SQL) which is the standard query language for relational databases. When we create a table in SQL we are creating a static template for instances of the entities in a category. When we select rows from the table we are asking for instances of entities in that category that have specific characteristics. We are, in fact, extracting a subcategory. In addition, the aggregate functions that you find in SQL are limited to the functions you would expect for analyzing categorical data. The count function tells you how many instances there are in a category. Sum, average, maximum and minimum allow you to determine limited distribution parameters of a numerical attribute of the instances of the category. But there are no correlation or temporal functions. You cannot easily answer questions such as - Is a certain kind of student more or less likely to get a high grade in a particular course than some other kind of student? You cannot easily ask if the average grade given for a course is higher or lower than it was in the past. That is not to say that you cannot get this information at all. It is simply to say that the categorical representation of the data and the limited analytical capabilities of SQL do not make it easy. Finally, relational databases are limited in their handling of dates and times. For example, if class attendance is recorded in the database, it would be difficult to determine if students are less likely to show up for class on the day before or after a holiday. So between the lack of temporally oriented functions such as moving average and the limitations of the date data types it is easy to see that relational databases are not well suited to representing temporal data.

A data warehouse, on the other hand, begins with a dimensional model representing a key business process. A common means of presenting a dimensional model is in a star schema where the facts representing measurements of the key business process are contained in a table at the center of the star while the surrounding dimension tables contain information on factors that might affect that measurement. We can think of the fact table at the center as the dependent variable and the dimensions as the independent variable. Time is always one of dimensions because dimensional models are inherently temporal. Let us assume, for the sake of this discussion, that we have a simple star schema in which the fact table contains sales data for a given product at a given store on a given day. The dimensions of this model would, of course, be product, store and day. However, we would not, in general, be interested in categorical questions. We would not ask – how many different products are there? Nor would be ask how many of the stores were over 100,000 square feet. We would ask if some stores produce more sales. We would also ask if certain products sold better at certain stores. The questions we would ask of a dimensional model are questions regarding the factors that affect the process that we are measuring. Since data warehouse products are not nearly as advanced as relational database products it is not possible to critique multidimensional query languages the way that SQL was critiqued. Further, many still feel that relational databases are perfectly adequate for dimensional data. So this analysis can only be taken so far based on the available products. Nonetheless, relational databases and data warehouses are based on quite different data models and contain quite different kinds of data. If one were to be more accurate we would call data warehouses multidimensional databases. Or we could call relational databases categorical databases and data warehouses longitudinal or

temporal databases. But names arise, not out of essence but out of convention and common usage. So we have what we have.

It bears mentioning, before moving on, that these are not the only two data models in existence. They are just the two most general. A data model is merely a collections of concepts and names for those concepts. The concepts are used to organize the data from a domain and allow someone to gain insight into the domain by examining the data organized according to the model. One of the first data models, appropriately called the hierarchical data model organized data into record types with hierarchical relationships. This was appropriate, at the time, because databases were viewed as electronic card trays and the purpose of those databases was not to deliver information but to support data processing. As database users became more interested in the information available from the database, the hierarchical model became less popular.

The network data model attempted to overcome some of the weaknesses in the hierarchical model by increasing the ease by which the data could be exploited. However, the network model was short lived because the relational data model provided a representation for categorical data that maximized the potential for exploitation. While the hierarchical and network data models were both models of how data is organized on a computer, the relational data model was the first model of data itself and hence the most general. It bears mentioning that the relational data model, while having improved incrementally, has not changed much conceptually in the several decades since it was introduced. There are few examples within the field of information systems of ideas that have lasted this long. The reason for this is that it is a conceptual model of data not a conceptual model of technology. Hence, unless our conceptual understanding of data changes radically, the relational data model has a long and promising future.

Contrast this with the variety of other data models that have appeared over those decades. The fuzzy data model, for example, allows one to represent inexact information. Whereas the relational data model requires single valued facts for attributes, the fuzzy data model allows inexact information. In a relational data base, one's height would have to be represented as a specific height, say 5 feet 11 inches. In a fuzzy database a person's height may be represented as slightly tall.

Whereas the relational database allows only facts, and single valued facts at that, the deductive data model allows rules of inference. Hence a user interrogating a deductive database may get all facts that satisfy a given query and all facts that can be inferred based on the facts in the database and the rules of inference.

We can think of the relational data model as a highly structured data model. All instances of an entity type must have the same attributes. For example, a student table may have an attribute called Name, but if the student has a nickname or an alias there is no room for it unless an attribute is created for all students. Semi structured data models such as XML allow the inclusion of data items that are not uniform across all instances of an entity and thus have great flexibility and greater representative power. However, this comes at the cost of exploitability. If one student has a nick name, another an alias, another a maiden

name, another a pen name, and so on, it would be hard to summarize this information in any useful way. When we design a relational database or a data warehouse, we superimpose order on the world and in doing so we increase our chances of understanding in the world in greater depth.

People often ask – What is the big deal with a data warehouse? Can you just think about a data warehouse as a large relational database containing a lot of historical data? The answer is – at some level you probably can, however, in doing so you will severely short change the expressive power of the data model. We have seen in this very short survey of data models that different data models allow you to represent things in the world differently and to answer different questions more easily. At the heart of a data warehouse is a dimensional data model which allows us to organize data in terms of processes that are measurable over time. Being able to measure processes allows us to improve processes. And that is the big deal with a data warehouse.

Summary

A data model is a conceptual construct that allows us to model some aspect of the world in data. Once a model is constructed we can interrogate the model in order to find out more about that aspect of the world that the model represents. Though there are, potentially, an unlimited number of data models, and in practice have been several, the most basic ones model data at its most basic level. That is they model data as belonging to categories as representing something changing over time. The relational data model is a basic model of categorical data. The dimensional model, which is the heart of a data warehouse, models processes as they change over time. In doing so, data warehouses allow us to improve processes. An as such, data warehouses represent a major step forward in data management.

Chapter 2: Introduction to Data Warehousing

The dimensional data model may be one of two primary data models for representing reality in a database. But data warehousing did not spring into the world fully formed as an essential concept. In fact the concepts used in this view of data warehousing still need substantial theoretical development. Instead, the concept of data warehousing has come to where it is today through a most circuitous route involving the evolution of ideas, technologies and even academic fields. The purpose of this chapter is to trace some of the key ideas in data warehousing from their source to the point where they converge today.

The emergence of the data warehouse concept was not a linear development of a single idea. Instead, a collection of ideas mingled, merged and influenced each other to bring data warehousing to where it is today. These ideas include some foundational ideas of Adam Smith and Frederick Taylor. They include ideas from information systems such as the truth database and decision support systems. And they include ideas from management such as process improvement and total quality management. It is difficult to determine exactly how each of these ideas influenced each other. But by looking at each of these ideas individually, we can see their contribution to the collection of concepts that make up data warehousing today. And by seeing each one of these developments we can better understand not only where data warehousing is today, but where it is likely to be going in the future.

The Classic View of Data Warehousing

The first and perhaps still that most widely cited definition of a data warehouse was first provided by William Inmon. “A data warehouse is a subject-oriented, integrated, nonvolatile and time-varying collection of data in support of management’s decisions. The data warehouse contains granular corporate data.” [Inmon, 2002, pg. 31] In this view a data warehouse was a repository of data collected in transaction processing systems and stored separately to facilitate analysis of the data rather than the processing of transactions. I will return to this definition later but for now we can think of the data warehouse as collection of historical data drawn from transaction processing systems. The data is extracted at periodic, probably monthly, intervals. It is time varying because each snapshot of data would represent a month’s activities. It is not volatile because it historical. It is subject oriented and integrated because the extraction process allows for corrections in design that may have been too late for the source systems. And it is used for decision making rather that for conducting the primary business of the organization. The first question we must address is – where did this idea come from? And while many ideas in information systems come from a coalescing of theory, practice and available technology, we can probably find the roots of data warehousing in the failure of a database concept that was popular in the mid 1970’s . For lack of a common descriptor, I will simple call this the Truth Database Concept.

The Failure of the Truth Database Concept

'Integrated databases' was a popular catch phrase in the 1970's and the most extreme form of this idea can be found in the Truth Database Concept. The Truth Database Concept simply claimed that there should be a single central database that represents the state of the organization at any moment in time. All sales, vendor payments, debt, inventory, etc. should be contained in a single database so that the financial state of the firm could be determined at any moment simply by issuing the appropriate query against the Truth Database. One can imagine an automatically refreshing display on the desk of the CEO that periodically queried the Truth Database and provided a succinct display representing the financial state of the firm with dials and bars much like the control panel of an airplane. With this information, the CEO could react quickly to any emerging events in the organization. Running low on cash would be like running low on gasoline and executives could track the status and intervene at exactly the right moment. It is a very seductive idea but suffers from two major flaws. First, it changes the focus of the CEO from strategic to operational concerns. And second, given the state of technology in the 1970's it was simply not practical.

If a company has online sales of 100 per minute, then that is 6000 transactions per hour. Limiting business to eight hours per day and five days per week that would yield 240,000 transaction each week or roughly a million transactions per month. So, if the CEO wanted to monitor sales activity, the query would require the summation of a million records if the query were executed at the end of the month. If this month were compared to last month it would require the summation of two million transactions. If a six month rolling average were computed it would involve summarizing six million transactions. But it doesn't stop there. Sales might be summarized in different ways, perhaps by product line, by sales person, or by the promotion that triggered the sale. The CEO may wish to monitor sales by time of day or day of week for staffing purposes. So more rollups need to be done in different ways.

And that is just the sales activity. All during this sales activity, customer payments are being recorded, inventory is adding and subtracting products items; vendors are being paid; employees are recording their time; internal accounts are being updated. And so on. It is easy to see why constantly running a set of queries that would summarize transaction data into a picture of the financial, quality or productivity health of the company would be a challenge even for today's computers and much greater for computers in the 1970's.

Further complications arise from the fact that database performance is usually determined by the indexing strategy. When a record is requested, the database management system does not search record by record through the database until it finds the record it is looking for. Instead it searches an index. Once it finds the record key in the index it can directly request that record by page number from secondary storage. But indexing is a double edged sword. Having an index speeds up locating a record, but slows down updates. Each time a record is added to an indexed table the index must also be updated. So indexes also slow down updates. Further, there are different kinds of indexes to address different performance needs. For example, an index that speeds access may not be the same as an index that speeds up a query that produces a report. If both kinds of indexes are created then, again, updates are slowed down because more indexes need to be updated with each

update to the database. All this is to say that transaction oriented databases are usually indexed to facilitate transaction updates to the database and not indexed to facilitate reporting, thus, further slowing down the mega queries that provide a picture of the health of the company. But, whether a company needs an online picture of the health of the company or just some analytical data to better understand how things are going, getting it from the Truth database just does not seem like a viable idea. So what is a company to do?

There is no easy solution to the problem of summarizing transaction level data in real time to produce metrics that reflect the health of the organization. One can observe that computers get faster over time and that memory capacities increase so this problem should go away. Yet, the level of detail and volume of transactions increases as well. So, time is not going to help much, at least for the foreseeable future. But this is a problem that does not really need to be solved. Transaction level data reflects the day to day operations of the firm and executives should be focusing more on the strategic issues. That is to say that the executives need information that reflects the trends in activities over time rather than a snapshot of the moment. And there is a reasonable solution to the problem of showing the activity trends of the firm over time. If the transaction level data were extracted on some regular basis, say monthly, from the transaction processing systems, summarized and stored at a higher level of summary in a separate database then analysts could use that summary data to produce reports showing trends in the firm's behavior without interfering with the transaction processing systems. This summary level data could accumulate over time and could be used for analysis of performance over time. And with this summary database, the concept of data warehousing was born.

On the face of it, it seems like we have found a solution. And, in part, we have. But there are still open questions. Relational databases are based on entity models of the organization. Since data in the data warehouse is largely summary data, how do we construct conceptual models that we can use to guide the design? Data in the relational databases is stored as records that correspond to entities and one of the cardinal rules of relational database design is that any given fact is represented only once in the database. But in the data warehouse summary data is stored. So a given fact may be recorded in several different summaries. On one hand this may be alright. The rule that any fact should be represented only once in a database prevents update anomalies. And the data warehouse does not really get updated. It gets an infusion of data periodically. But it is not updated. So maybe this is not a problem. And yet, the rule is also intended to prevent semantic disintegrity. That is the problem that occurs when a user executes a query and gets an answer but it is not the answer to the question they think they asked. This problem also occurs when facts are represented more than once and this is a problem for the data warehouse.

Further, ignoring these theoretical problems we have practical problems to consider as well. How is data stored? How is the data extracted? What summaries are needed? In a relational database, data is stored in records that correspond to entity occurrences. What do summary records correspond to? Although one could posit the existence of artificial entities such as total sales for a given product for a given week, this does not seem

particularly satisfying. After all the constructed entity could be weekday sales or sales for a ten day period. Some data might not be so easy to summarize. Employee data, for example, tends to change rather slowly. The data changes when somebody is hired, promoted, fired or retired. Sometimes addresses and phone number change. It is hard to imagine what meaningful summaries could be made of this data.

And even if we could determine some meaningful summaries, how should we store it. Should twelve months worth of sales data be stored in twelve separate records as relational database normalization would require. Or should it be stored in an array of twelve values? Since the data is not being updated or queries in a traditional relational database sense, do we still need to follow traditional database rules.

Finally, what kinds of queries or extracts should be allowed? Typical SQL queries attempt to locate a single record or a set of records that meet specified criteria. Data warehouse queries are usually requests for data summarized according to specified criteria. So an entirely different kind of query and extract mechanism may be needed.

So even though it may have appeared, superficially, that the summary data problem had been solved, in reality it had merely been traded for a whole set of theoretical problems. And in order to get a handle on these theoretical problems, we need to go back to the 18th economist Adam Smith. It is extremely unlikely that anyone pondering the problems associated with data warehousing even thought about Adam Smith, at least not in connection with data warehousing. But in order to see the evolution of ideas that eventually led to the modern concept of dimensional data modeling we need to invoke the ideas of the long dead economist.

Adam Smith

In 1776 Adam Smith published The Wealth of Nations in which he made a number of then new but now commonplace observations about a healthy and well functioning economy. One of those now famous observations was the division of labor. Laborers are most productive, according to Smith when work is divided up in such a way that workers can specialize their activities and focus on one task that they perform repetitively. Doing this one single task they can become more proficient at it. And a group of workers, each doing one specialized task repeatedly, can outperform a group of workers in which each worker does each task. This specialization of labor idea has become the cornerstone of manufacturing productivity. In Smith's words,

> "The greatest improvement in the productive powers of labour, and the greater part of the skill, dexterity, and judgment with which it is anywhere directed, or applied, seem to have been the effects of the division of labor." [pg. 3]

He goes on to say,

> "The division of labour, however, so far as it can be introduced, occasions, in every art, a proportional increase of the productive powers of labour." [pg. 5]

And,

> "This great increase of the quantity of work which, in consequences of the division of labour, the same number of people are capable of performing, is owing to three different circumstances; first, to the increase of dexterity in every particular workman; secondly, to the saving of the time which is commonly lost in passing from one species of work to another, and lastly, to the invention of a great number of machines which facilitate and abridge labour, and enable one man to do the work of many." [pg. 7]

The essence of this idea is that when one can focus on a task they can get better at it through specialization. There is no question that this principle works extremely well in a manufacturing environment. In fact, in Smith's original treatise he gave the example of a manufacturing case in which pins were being produced. The question is – does this apply to white collar work as well?

Consider an organization in which nobody has any specific duties. The only requirement is to do whatever needs to be done. So a person may be keeping the books one day, taking customer orders another day, interviewing potential employees the next day, and planning a five year strategy on yet another day. Imagine further that this company has hundreds and hundreds of employees. How well do you think this company would function? Not very well, most people would agree. And, in fact, in most companies today (with the possible exception of very small or very disorganized companies) most employees are assigned to functional units that allow them to focus their activities and make sure that all necessary tasks receive the needed attention. However, within functional units activities are often a little less structured. Most people would say that the nature of white collar work is that it cannot be too structured. It involves problem solving, communicating, coordinating and other activities that are not so well defined. White collar workers attend meetings, send email, write memos, interact with other employees and perform a whole host of activities that defy precise definition. So are the productivity advances that we have seen in manufacturing beyond the reach of white collar work? Perhaps not. But in order to see this we need to turn to Frederick Taylor, the father of what has come to be known as Scientific Management.

Frederick Taylor

In 1911, Frederick Taylor published his landmark book Principles of Scientific Management. In this book, Taylor introduced several key principles for improving the productivity of blue collar workers. Work should be studied, according to Taylor, in a scientific manner. Specific tasks should be analyzed and idealized. The analysis should be done scientifically which means that the analysts must be independent and objective.

Craftsmen, according to Taylor, cannot optimize their own work. This must be done by specially trained analysts. Work done by heuristics, on the job training, or tradition is ineffective. Work performed according to ideal task produces ideal productivity. Further, that productivity must be determined by objective measures and employees should be rewarded for performance.

At the time, most people were skeptically of Taylor's ideas. Craftsmen did not think it would be possible for non specialists to create ideal task structures. Nor did they think anybody would accept these tasks. And, yet, over time we have seen Taylor's ideas provide a tremendous benefit for manufacturing productivity. However, scientific management originally focused on production work. It did not address white collar work. As we will see in the chapter on design, the heart of dimensional modeling applies Taylor's principles of Scientific Management to white collar work.

Decision Support Systems

Skipping ahead more than 50 years since the observations of Frederick Taylor, we see the invention of the computer and the rise of information systems. Traditional information systems are viewed as a pyramid with transaction processing systems at the base, then management information systems, and finally decision support systems at the top. These three levels of information systems correspond to three levels of information in the organization. At the lowest level, transaction processing systems process operational data. Operational data represents the work of the organization being done. Sales are made. Bills are sent. And employees are paid. If the database is a model of the organization, it is through the transaction processing systems that the model is updated and it is through those updates that the database maintains consistency with the state of the organization. Moving up one level we find management information systems which process tactical data. Tactical or administrative managers are responsible for allocating the resources of the organization toward achieving the goals of the organization. So tactical reports typically present planned progress toward some budgetary or achievement goal versus actual progress. The heart of a management information system is the variance report. It could be a sales forecast versus actual sales or a project budget versus actual expenditures. Finally, the highest level of information system is a decision support system. Decision support systems are used to make strategic decisions about the future directions of the firm. Using historical data and projecting it into the future, a decision support systems uses models of the organization to anticipate future circumstances and allows an analysts to explore possible reactions to those future circumstances.

These three levels of information systems use different technology, have different objectives and require different skills to develop. Transaction processing systems generally require database management systems that support concurrency control or multi-user update management. Efficiency is generally more important than flexibility and concepts like transaction models, record locking and deadlock are important considerations. Management information systems use databases too, but more for their query languages and report writers than for concurrency control. Ease of report

production is more important than efficiency, and flexibility is needed in producing new reports to determine the source of problems when actual figures do not agree with budgeted, forecast or projected figures. Whereas a programmer working on a transaction processing systems would have to understand record locking, a programmer working on a management information system would have to understand SQL.

Decision support systems use historical data from the transaction processing system and analytical or statistical models of the firm to project future outcomes. Flexibility in data extraction and the expressiveness of the models are far more important than processing efficiency or reporting flexibility. A programmer working on a decision support system would have to know more about quantitative or statistical modeling than about query languages or record locking. And it is out of these analytical applications that the concept of OLAP or On Line Analytical Processing grew. The term OLAP is often used to distinguish between databases used primary for analysis and databases used primarily for transaction processing, the later being referred to as OLTP or On Line Transaction Processing databases. And it is from this idea of decision support and online analytical processing that the data warehouse concept would begin to take shape. It should be made clear, however, that while data warehousing grew out decision support systems, there are many decision support systems, such as statistical, modeling, and analytical applications, that are not data warehousing applications. How these applications differ from data warehousing and why this distinction is important will be addressed in Chapter 4. But, for now, if these applications are not data warehouse applications this raises the question – what kind of applications are data warehousing applications? The answer to that questions is that data warehousing applications focus on a measurable business process that we wish to improve. And that leads nicely into the next topic – process improvement.

Process Improvement

Before proceeding with the evolution of the data warehousing concept, we need to make a side trip into the concept of process improvement. This important concept was not directly in the evolutionary path between decision support systems and data warehousing. Instead, it would merge into these emerging ideas and lead to the conflicting views of data warehouse design that are discussed in detail in Chapter 4. So for now, it is best to think of process improvement as a collection of ideas that developed parallel to data warehousing and eventually began to influence its direction.

Conceptually, the way in which a process is improved is fairly straightforward. First, you figure out what the process is intended to achieve and how to measure the performance. Second, you perform the process and determine the results through measurement. Third, in the event that the measurement reveals a correctable deficit in performance, you make some adjustments in how the process is performed, or possibly in how it is measured, and then return to the first step.

Think of this simple process in terms of a thermostat. First, you set the thermostat at a temperature that you believe will be comfortable. The process here, which may not be

obvious, is the process of heating or cooling the room in order to produce a comfortable temperature. The effectiveness of the process is measured by the thermostat and by your subjective reaction to the temperature in the room. For example, when the temperature drops to a point below the assigned temperature on the thermostat, the furnace will turn on and warm the room up a bit. However, if you feel that the room is still not warm enough, you may reset the thermostat at a higher value. The point is that in order to get the room to stay at a comfortable temperature we need to establish some norms of performance, take measurements and make adjustments. Whether the process is heating a room or producing satisfied customers the essential process improvement model is the same. When we design a data warehouse we are, in essence, designing a data structure with which we can record measurements that tell us how well a process is being performed.

Let's take the thermostat example a step further in order to gain some additional insight into process improvement. In fact, we'll muddy up the process a bit. Let's say that you have several ways to affect the temperature of the room. Let's say you have a fireplace, a small space heater and windows that you can open or close to affect the temperature. Further, let's say that you can affect your comfort level by putting on or taking off a sweater. Finally, let's just observe that your body temperature fluctuates throughout the day so that a temperature at which you may be comfortable at one point in the day may leave you feeling chilly at a different point in the day.

Now, in an attempt to feel warm in the room you light a fire, turn on the space heater and put on a sweater. All this makes you too warm so you take off the sweater. As it turns out the room temperature had dropped a bit which is why it was chilly in the room and the thermostat caused the furnace to cut on. As the furnace warmed the room so did the fire and the space heater, and, now it is too warm. As the room cools off you get chilly again. Without the sweater you are more likely to feel the temperature of the room so you become very aware as the temperature drops again. But the thermostat is near the fire and the space heater so even though you are chilly the furnace does not cut back on right away. But the fire and the space heater do not adequately warm the room so you get chilly and turn up the thermostat and pull your sweater back on. The furnace cuts back on and warms the room to the new temperature where upon you are way too hot and have to take off the sweater. You also turn off the space heater. We could go on with this scenario but it is easy to see that this process is not likely to produce a comfortable room temperature, and this brings us to the first point about process improvement. In order to improve a process the process must be well defined.

What do we mean by a well defined process? This process is not well defined because it is not clear what we are trying to achieve and it is further unclear what factors are involved in achieve the result. On the first issue, it is unclear as to whether we are trying to keep the room at a stable temperature or trying to make it comfortable to be in. These are two, quite different objectives.

If the goal is to keep the room at a consistent temperature then we need only insure that the thermostat is working correctly. If the thermostat is not working correctly then we

need to replace it. If the room temperature is not consistent then perhaps we need to add more heater vents and possibly more thermostats in the room. However, assuming that we get the thermostat to work perfectly it may still not be the case that the room is comfortable. In order to make the room comfortable we may have to raise or lower the thermostat to a level that is perceived as comfortable to a person in the room. This level may not be consistent throughout the day so we may need a programmable thermostat that allows us to set the temperature at different levels during different times of the day. We may need an adjustable thermostat that allows a person in the room to raise or lower the temperature depending on how much clothing they are wearing. Keeping the temperature at a consistent level and keeping the room comfortable are two quite different processes. So, for purposes of this example, let's assume that the thermostat works perfectly well and the process we are trying to improve is to make the room comfortable for a person standing in the room.

Now that we know what we are trying to achieve, the next question is how do we measure it? One way to measure how comfortable the room would be to put a thermometer on the wall and measure the actual temperature. This is tempting because it is an objective measure. However, as we have already seen, the temperature in the room is not a good indicator of how comfortable the room may be to an inhabitant. The only way we can tell if a person is comfortable is to ask them. So for this example we would have to ask the person. We could just ask – are you warm enough and get a yes or no answer. But we would probably be better off asking them to rate the comfort level of room using a subjective scale such as Cold, Chilly, Comfortable, Warm, Hot. We could then convert Cold to 1, Chilly to 2 and so on. Other measurements are possible. We could require each person who enters the room to be wearing a t-shirt, shirt, sweater and jacket. We could then count the number of layers they remove. While this would provide us with a numerical measure, it has other problems. For example, a person might put up with a warm room rather than removing their shirt. The point here is that once we have clearly defined the process that we are trying to improve or regulate, then next step is to find some way of measuring our performance on improving or regulating this process.

Once we have clearly identified the process and figured out a reasonable way to measure the process our next step is to figure out how often to take the measurements. In data warehousing terminology, the interval at which we take measurements is called the grain.

Assume that we have a person in the room and we are trying to determine if they are comfortable. We could ask them once a year but it is likely that this interval would not capture as much information as we would need to regulate the temperature of the room correctly. We could ask them every second but that would be far too frequently and it is quite likely that the response that they gave one second would be exactly the same as the response they would give in the next second. So yearly is too coarse of a grain and every second is way too fine. Let's consider intervals in between. If we ask them daily the grain would still be too course. Temperatures fluctuate over the course of the day and valuable information might be lost. So an interval of an hour would seem to be frequent enough to capture daily fluctuations in temperature and not so frequent as to increase the data collection burden unduly. Hence, our grain is to interrogate our subjects hourly.

Our next step is to determine our independent variables. Let's say, for the sake of argument, that the thermostat is stuck and there is nothing that we can do about the temperature of the room. Then if a person in the room claims to be too warm or too cold there is nothing we can do about it. Hence, there is nothing we can do to improve the process of making inhabitants of the room comfortable. This brings up an important point about data warehouse design. There must be a variable that we can manipulates in order to affect that process that we are trying to improve. In the example we are considering here we are assuming that we can adjust the thermostat and in doing so make the people in the room more comfortable.

At the same time we must consider, even though we cannot control, what a person is wearing when they come into the room. If on person shows up in a sweater and a coat while another person shows up in a t-shirt they are going to perceive the comfort level of the room differently. Further, as we mentioned earlier, a person's body temperature fluctuates during the day so the same person may perceive a fixed temperature differently at different times of the day. If we wanted to get fancy we could also attempt to determine what a person was doing before they entered the room. A person who had been running or eating hot soup might perceive the room temperature differently than someone who has been loafing or eating ice cream. Since the person's clothing, the thermostat setting, and prior activities might all influence the person's comfort level we need to collect this information and these pieces of information become the dimensions of the process improvement problem.

Finally, we need to make sure that the person's perception of the comfort level of the room is not affected by anything that we have not considered. Earlier we allowed for other heat sources such as a fireplace or a space heater. If we are trying to improve the subjective comfort level of a person in our room by manipulating the thermostat, we need to control other factors that might influence that comfort level. For example, if we had a fire burning in the fire place and a person reported that the room was perfectly comfortable, we would not know if it was the thermostat setting or the fire that was making the room comfortable. So, no fires, space heaters, or opening and closing of windows. We must control any factors that might confound our results.

The notion of process improvement that began with Taylor's Scientific Management advanced further when applied to manufacturing processes through out the 20th century. Visionaries such as Edwards Demmings and initiatives such as Total Quality Management took the concept of process improvement to new levels of refinement. But those refinements did not influence white collar work in the same way that they affected blue collar work. As we will see in Chapter 4, there are competing views of data warehouse design. One carries the banner of process improvement while the other does not. The approach used in this book is based on the process improvement model and the various rationales for that position are explained in Chapter 4.

The Summary Database Becomes the Data Warehouse

As the Truth Database concept crumbled and gave way to a new a new vision consisting of a transaction oriented database and a summary database, we see the beginnings of the data warehousing concept beginning to emerge. As shown in Figure 1 data from transaction oriented systems would go through a process of extraction, purification and summarization after which it could be exploited for decision support. Often times, these summary databases were nothing more than spreadsheets, but we see the notion of separated data being extracted and used for analysis.

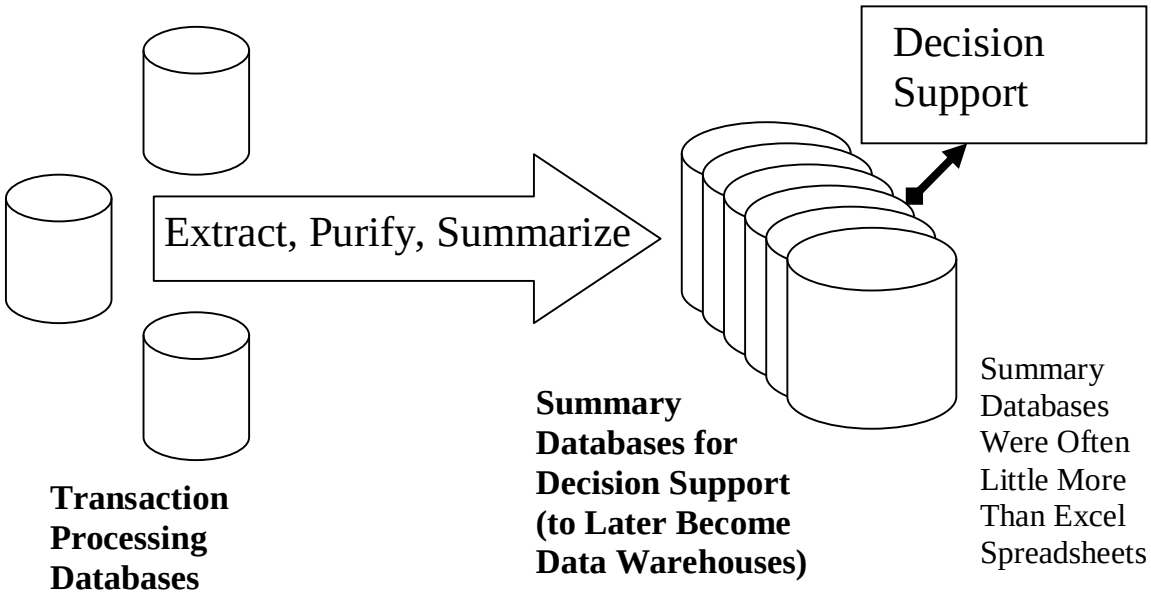

Figure 1 – Early Model of the Summary Database for Decision Support

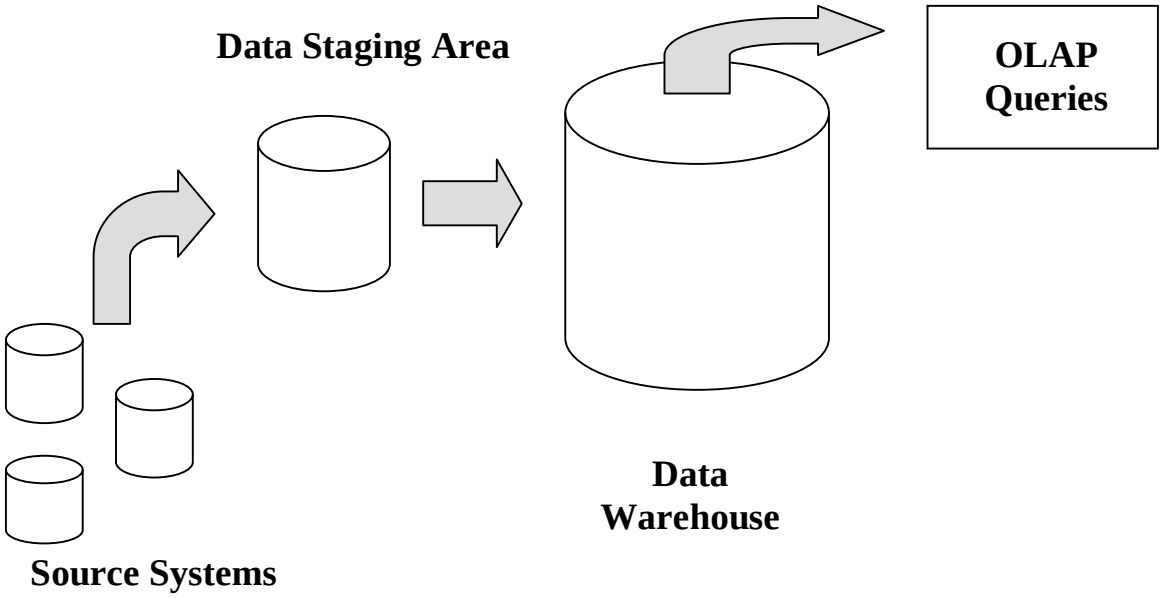

Figure 2 – Early Model of Data Driven Data Warehousing

We only need to take this concept one step further to see the birth of data warehousing. Instead of viewing summary databases as extracts we take the additional step of storing summary data over time. Now the summary database becomes a data warehouse. Decision Support Systems become replaced by online analysis which become generalized to online analytical processing or OLAP. The warehouse is viewed as a huge repository of historical data. And the extraction process evolves into data staging, a process that may involve a great deal more manipulation than extract, purify and summarize.

In Figure 2 we see the essential data warehouse model for data driven data warehousing. Data is acquired from source systems which are usually the transaction processing systems in the organization. However, data from existing source systems may have inconsistencies that need to be resolved. Some of the data may be incorrect. And certainly the level of data as it exists in the transaction processing system provides far more detail than is needed for analysis. So the extraction process becomes a data staging process where low level transaction data is summarized into intermediate values for analysis. This analysis would, to some extent, provide greater insight into the health and performance of the organization than was possible by examining detailed transaction data. So it appears that we are now modeling the organization at a different level. But the nature of that level is still a little unclear.

New Questions Arise

At this point in the evolution of ideas we have a data warehouse as a collection of old data used for analysis and exploited by a variety of software tools. But is there any

consistency to the concept. There seems to be more questions than answers. Is data warehouse design the same as relational database design. Can we use entity modeling to describe a data warehouse conceptually? What exactly are we modeling? Do relational databases and data warehouses differ only in the amount of data they contain and the level of summary of that data?

At the conceptual level entities do not seem to be the best vehicle for describing the data warehouse. The data isn't really categorical. It is temporal. Data in the data warehouse should be used to study the organization over time. But, if that is the case isn't a new conceptual model needed?

At the logical level tables seem, on one hand, to be adequate for storing data warehouse data. After all, a star schema has one central table at the center and dimensional tables surrounding it. On the other hand, tables should correspond to entities and facilitate NxN queries. But in a star schema not all tables are of equal importance. The fact table at the center seems to be far more important than the dimensional tables. So a flat table model does not seem adequate.

At the physical level, performance in a data warehouse is more greatly affected by the storage of intermediate summary tables than it is by indexing. So traditional physical design for relational database goes out the window. But in doing so it leaves a gap in physical design. So despite the fact that relational database theory seems to apply in many ways to a data warehouse, in many more ways it does not apply and we are left with a theory gap that must be filled.

In addition, queries against the data warehouse addressed questions that are quite different from the queries executed against a relational database. Relational databases provide data that represents a snapshot of the organization. But data warehouses, given their historical nature, are huge collections of temporal data that can reveal patterns of behavior over time. When a query to count the number of employees is executed in a relational data, the implied constraint is that user wants the number of employees at the current moment, not, for example, the number of employees last year or the total number of people who have ever worked for the organization. However, with historical temporal data you could ask questions about the number of employees in different time periods. You could look at trends and ask questions such as whether the number of employees is increasing or decreasing.

Finally, the update models are quite different. A relational database supports inserts, updates and deletes. If we hire a new employee, their record gets added to the database. If they get a raise, their record is updated. If they leave their record is deleted. Transactions are used to keep the database consistent with the organization that it is modeling. However, in a data warehouse, data is added in temporal chucks - say a month's worth of data at a time. Once inserted into the data warehouse, the data is a historical record and should never be changed unless it was recorded incorrectly. For example, if the monthly average of the number employees last year was 1000 then it remains as 1000 unless we discover somehow that this average was incorrect. If the monthly average of the number

of employees for this year is 1200 we don't change last year's value, we add a new value for this year. So the update model for a data warehouse is entirely different. All of these questions lead to an emerging need to better define the essence of the data warehousing concept and that is the topic of future chapters. But, before moving on with this concept we need to take a quick look at how the use of metaphors can help us sort out ideas in the short term while providing unfortunate baggage for the long term.

Some Bad Metaphors

Lakoff and Johnson, in their widely read book *Metaphors We Live By*, go to great length to reveal the extensive use of metaphors in our everyday language. This will be addressed in greater length in Chapter 11, but for now suffice it to say that we use metaphors routinely when we try to explain something we don't understand in terms of something we do understand. And this is useful when ideas are evolving. However, the down side is that we are sometimes stuck with metaphors that no long apply once an idea is more fully understood. And one could easily argue that the use of metaphors in language is a sign that the speaker does not fully understand what he or she is talking about. Data warehousing is one of those pesky metaphors. In the early days of data warehousing the metaphor was appropriate. Data was placed in a large storage repository where it could be located later. The most salient characteristic of this repository was that it was bulk storage like a warehouse. Other aspects of the warehouse metaphor also held. In a manufacturing environment one might go to the warehouse to get parts or subassemblies from which something would be made. And in the data warehouse one might retrieve bits of data from which a summary analysis would be made. But as we have acquired greater insight into the nature of the data warehouse this metaphor no longer seems appropriate. A much better designation would be multidimensional databases. The data in the warehouse is multidimensional by virtue of the dimensions surrounding the fact table. It is also temporal as opposed to the categorical data found in a relational database. The data warehouse could also be called a temporal data base. However, multidimensional seems to fit a little better because the data analysis done on a dimensional model has more in common with other multidimensional data than it does with other temporal data. So multidimensional database seems to be the best designation.

One attempt to get away from this inadequate metaphor used the term analytical database [McGuff and Kador, 1999] in place of data warehouse. Analytical databases serve an important function and are different from traditional relational databases in some important ways. In fact, analytical databases overlap substantially with the earlier concept of data warehousing. But, as we shall see later, analysis is not a primary function of a data warehouse and thus data warehouses should be distinguished from analytical databases.

In data mining, an area sometimes linked with data warehousing our understanding is also obscured by the common use of a bad metaphor. The term data mining suggests that we tunnel into a collection of data looking for nuggets of information that way that miners might tunnel into a gold mine looking for nuggets of gold. No only does this

metaphor misrepresent mining operations, it completely fails to provide any insight into the wide variety of data mining applications. Data mining will be addressed in some detail in a future chapter.

Summary

This chapter has provided an overview of some of the key ideas that came together to form out current understanding of data warehousing. These include ideas of Adam Smith and Frederick Taylor, along with evolving ideas in information systems and process improvement. These idea have come together to form the concept of a data warehouse that stores multidimensional data for the purpose of improving measurable business processes. As we will see in Chapter 4, this is not the only view of data warehousing and in that chapter we will see why this view has some very desirable features.

Bibliography

Inmon, W. et.al. (2000) Exploration warehousing : turning business information into business opportunity. John Wiley & Sons.

Inmon, W. (2002) Building the data warehouse. (3e) John Wiley & Sons.

Lakoff, G. and Johnson, M. (1980) Metaphors We Live By. University of Chicago Press.

McGuff, F. and Kador, J. (1999) Developing Analytical Database Applications. Prentice Hall.

Smith, Adam (1937) An Inquiry Into the Nature and Causes of The Wealth of Nations. The Modern Library.

Taylor, F. (1998) The Principles of Scientific Management. Dover.

Chapter 3: A Case Study – Theatre Attendance

Before getting into the technical aspects of data warehouse design a simple case study is needed in order to provide a concrete example from which we can draw the conceptual structure of data warehouse design. In this chapter we will develop a data mart for a movie theatre chain. The goal of the movie theatre is to have each showing of a movie attract as many viewers as it possibly can. Ideally, at each showing, every seat will be filled, but that is unrealistic. However, by careful selection of movies and show times; and by careful crafting of promotional efforts it is possible to maximize the number of viewers that attend a showing. As we go through this example we will examine some of the factors in data warehouse design and introduce some key concepts.

The Basics of Data Warehouse Design

We will get into the details of data warehouse design in Chapters 4 and 5. But for now we can quickly list the steps in the design process. Those steps are as follows:

1) Select a business process to model
2) Define the grain of the business process
3) Choose the dimensions of the process
4) Identify specific facts that will populate the fact table.

For our movie theatre, there are many potential processes to track. The theatre has concession sales, it schedules employees, and it maintains an inventory of items for sale at the concession stand that must turnover at a proper rate to avoid spoilage. But we are not going to be concerned about any of those problems. We are going to be concerned with filling as many seats in a theatre as possible for each movie showing. So the business process that we are concerned with is the process of getting viewers to attend a showing. Informally, this is sometimes referred to as the 'butts in seats' problem and it is not unique to movie theatres. It is also a problem for people who put on concerts or offer seminars, workshops or classes. It can be a problem for people who sell tickets for bus or airline seats and it can be a problem for people who provide bookings for hotels or resorts. The point of this digression is to say that while any given business may have a number of business processes to consider, very few of those processes are truly unique. Many of these processes can be described by generic models. The generic model issue will be addressed in further detail in Chapter 4.

This example also illustrates the first principle of data warehouse design. There may be many interesting questions that we may want to ask about how a business operates and there may be a number of interesting processes that we could track. However, in order to build an effective data mart we need to pick one and in doing so we exclude the others, at least for the time being. So we have to identify the most important problem or the area in which improvement will yield the greatest benefits.

This makes data warehouse design a little tricky from the very beginning because the business process that we are going to use to develop the dimensional model may not be a

process that is singled out in today's operations. For example, the theatre may have sales targets for concessions and movie tickets which it either meets or does not meet. If monthly sales are a little behind they may have a promotion of some kind to boost sales. But the exact relationships between promotions and sales are probably not very well understood. And they may not see this as a measurable process at all. We tend not to look at the things that we do as measurable processes that we can improve over time. We are more likely to look at them as efforts that were successful or not successful. So we cannot just walk into the offices of theatre management and ask for the business process that they would most like to improve. We are, in many ways, through the process of data warehouse design creating a measurable business process that does not really exist (at least explicitly and measurably) in today's operational environment.

So, assume that we have identified the process of using promotions to increase the number of people who attend a movie showing as the process we would like to improve. The next thing we need to consider is the grain of the process. The grain is the fineness of the measurements we will be taking. The grossest grain imaginable would be total ticket sales for the theatre chain per year. However, it is easy to see that a grain such as this would give us little insight into the factors that produced those sales. It is fairly easy to imagine that different promotions would be more or less effective on different kinds of movies. It is also fairly easy to see that different promotions would be more or less effective for different showings at different theatres. For example, early bird discounts would work for early afternoon showings where the local population is not all at work. Since we are promoting specific showings of specific movies we need to take the grain at least down to that level. So our grain will be attendance by movie, by theatre, by showing in response to a specific promotional activity. Selecting this grain gives us the dimensions of the process. They are: theatre, showing, promotion and movie. Figure 1 shows a preliminary dimensional model of this business process.

Next, we have to select the fact or facts that we will use to populate the fact table. We are measuring attendance at a showing of the movie so a natural measure would be the number of people who show up or the number of tickets we sell. But, not all theatres are the same size so selling twenty tickets to a theatre that has a capacity of twenty five could be considered a success, whereas selling twenty tickets to a theatre that has one hundred and fifty seats would not be a success. So perhaps we need to collect the percentage of filled seats also. Not all tickets are sold at the same price. Some are discounted. Some are free passes. Children and senior citizens pay less for tickets. So we should also collect the aggregate income for the showing. A promotion that offers half price admission would have to attract twice as many viewers to break even. So total sales for the viewing is also important. Finally, we should also include the cost of showing the movie. Some of this cost is fixed cost for the theatre and some is the cost of showing a particular movie. Not all movies cost the same to show so the cost of the movie has to be taken into consideration.

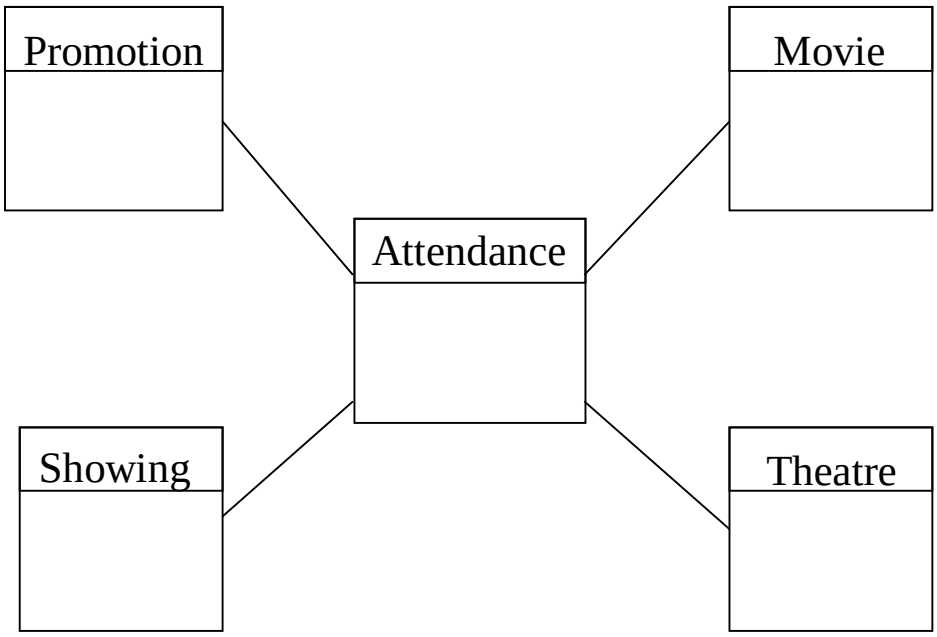

Figure 1 – Dimensional Model Theatre Attendance

Having to collect all of these facts about a specific viewing reveals another important principle of data warehouse design. We cannot just summarize data from the transaction processing systems in order to populate our dimensional model. Additional processing may be necessary. Next we look at each aspect of this design in greater detail.

The Process

The process is filling seats at the showing of a movie. In order to be appropriate for a dart mart this process must have several characteristics. First, it must be measurable in some way. If it is not measurable then we have no way of knowing how we are doing. And if we have no way of knowing how we are doing then there is no way to improve the process. This brings out the second feature of an appropriate process. It must be worthy of improvement. If we are just going to look at the results of any efforts that we might make to increase attendance, shrug and say "well that's interesting" then there really is no point in going to all the effort to create a dart mart. And it should be emphasized that what we are creating here really is a dart mart and not a data warehouse. It is a data mart because it addresses a single process. If we were to identify other processes such as improving concession sales, negotiating showing fees, or minimizing salary expenses we could construct dimensional models for those also. Hopefully, we could integrate those dimensional models and maybe even share some dimensions. Then we would have a full fledged data warehouse. But for now we have a dart mart.

A third aspect of the process is that the dimensions must influence the outcome. If the attendance at a showing is fifty percent of the seats no matter what we do to promote the movie, and no matter what movie we show nor what time we show it, then there really is no reason to create a data mart to track this process. And one of those dimensions has to be something that we can directly manipulate in an attempt to affect the outcome. We call this dimension a motivator and the dimensional model must have at least one motivator otherwise there is nothing we can do to improve the process. Without a motivator, all we can do is helplessly watch the data come in and see how attendance changes over time unable to change or affect it in any way. The other dimensions recognize the fact that manipulating the motivator will influence the outcome differently under different circumstances. So a given promotional strategy might have a different impact on a day time showing than on a night time showing. Or it may produce different results for a blockbuster movie than for a documentary. We call these dimensions mediators. A properly selected and well defined process will have at least one motivator and a couple of mediators.

The Dimensions

The dimensions of the process have already been mentioned. They are theatre, showing, movie, and promotions. The motivator is promotions so we will take that one first. The underlying premise of this model is that an effective promotional strategy will maximize attendance at a showing. In some cases, such as a weekend evening showing of a blockbuster movie it may seem that no promotions are necessary. But that is not true. There are TV ads, some national some local. There are schedules of showings printed in newspapers and on websites. There are posters displayed in the theatres showing what is currently playing or coming attractions. Some of this is standard and the theatre owner either has no control over it such as in the case of national TV ads or the promotions may be standard for every movie such as newspaper and website listings. Whatever affect these promotional activities may have on attendance is irrelevant because they are not variables that the theatre owners have any influence over.

What the theatre owner does have control over is local television and radio advertising, local newspaper advertising, and signs on buses or subways or on kiosks in shopping malls. The theatre management can also offer price reductions (general or targeted), free passes through radio stations or mall visits, or special offers to customers who have registered at the theatre's website. In order to track the effectiveness of promotions on attendance each showing must be covered by a single promotion package. That package may have multiple elements, but the more complex the package the more difficult it will be to determine which element of the package had the greatest influence on the attendance.

For the showing there is little additional information other than the time band, the showing room and the room capacity. Note here that the showing room is often called the 'theatre' but we are using the term theatre for a collection of showing rooms and this distinction is important because what brings viewers to a particular theatre may not at all be the same as what brings them to a specific showing.

For the movie dimension we certainly need the title of the movie and possibly the rating. Other attributes may not be so obvious and this brings up another aspect of data warehouse design. The designer should include all attributes that are likely to affect the performance of the promotion package. A given movie has several attributes that conceivably could affect this. Each movies has a genre (drama, romance, comedy, etc.). Each movie can be classified by audience specialization (popular, classic, art, foreign, etc). Each movie has a collection of stars, a director and a producer. How much of this information should be collected about every movie? This is a tough call. If we don't collect information on an important attribute we may miss out on important information. For example, it is conceivable that viewers of art or foreign films respond to information ads while viewers of classic films respond to discounts. So it would seem that we should gather as much information as possible. But this is not the solution either. The more information we gather, the more complex the data set becomes. The more complex the data set, the more difficult it is to exploit. The more difficult the data set is to exploit the less likely a given user of the data will fully analyze the data without getting confused. The less fully analyzed the data set, the less likely we discover important relationships. So more data can lead to less information. In some cases, a great deal of judgment and possibly some trial and error may be needed to find the ideal set of attributes.

Finally, the theatre dimension should include all attributes of the theatre that are likely to affect the performance of promotional package. As we saw with movies, this set of attributes may not be obvious. Potential attributes may include the number of showing rooms in the theatre, the theatre style (e.g. stadium, dining and bar or traditional), or how new the theatre is. However, it is also easy to see how the demographics of the neighborhood in which the theatre is located might affect promotion performance as might factors such as whether or not the theatre is located in a shopping mall. Again, too few attributes might leave out potentially important information while too many attributes might confuse the analyst or other users of the data.

The Facts

Certainly the fact table should contain the number the tickets sold since that is what we are attempting to influence with the promotions. But this is not the only possibility. We may also want to collect the amount of revenue generated. A theatre with one hundred seats selling for $9 each will generate $900 when full. That same theater during an discounted matinee where tickets sell for $5 will only generate $500. So we need to keep track of tickets sold and revenues generated. However, in order to keep track of revenue we need to make sure that the point of sale equipment in our ticket booth is capable of counting dollars as well as tickets sold. In addition, if some tickets are sold over the Internet we need to collect that information as well and merge it in with the information from the ticket booth. The point here is that revenue information is important in determining the effectiveness of promotions. At the same time revenue information may be very difficult to collect. So the fact table designer must weigh the cost of producing this information with the benefits it may provide.

There are other facts that may be useful to collect and store in order to facilitate analysis. For example, we may wish to store the ratio of seats filled to seats available (percentage filled) in the fact table. Even though this value could be computed from the number of tickets sold which is in the fact table and the room capacity which is in the showing dimension, there are two reasons for storing this ratio in the fact table. First, it will make analysis easier since each data cube that is constructed later will not have to realize this ratio as a computed value. Second, it is possible that the capacity of a room changes over time. The room may be remodeled as in the case of converting a traditional theatre to stadium seating. Or, in an older theatre some seats may be unusable due to damage, wear or vandalism.

Another fact that may be useful is the mean number of tickets sold for this room. Suppose the room has 100 seats and the mean number of tickets sold to a showing in that room is 50. Then if a showing managed to fill 60 seats we could determine right away this particular showing did better than the average for this particular room. Again, the mean can always be computed but storing the computed value may make analysis easier. As we saw when identifying the attributes for dimensions, too few facts might leave out potentially important information while too many facts might confuse the analyst or other users of the data. There is no hard and fast answer on whether or not these values or any other values should be included. There are benefits and drawback in both directions. But the design must consider those trade offs and determine which facts would provide the most benefit if included in the fact table.

Sample Reports

While it is tempting to think that once the dimensional model is defined, the work of the designer it over, this is not true. The designer must identify a series of stock reports and show how those reports are used by management to improve the process under consideration. Consider the report in Figure 2 for example. This report shows Revenue by Movie Type. And while this information may be interesting to management it is not clear how this report would be used in making management decisions. Should management schedule more romance movies to bring up the revenues in that genre. Or should they schedule fewer romance movies since they don't seem to bring in as much revenue? While reports of this type may help management get a better understanding of the current state of their business, they do not help management determine what should be done next.

Similarly with the report in Figure 3. This report shows how revenues are related to Promotion Type and may give management some insight into which promotions are the most effective. However, as we can easily see from the report, the bulk of their revenue comes from Matinee Discounts. So what should management do with this information? In both of these cases, this information is analytical in that it provides insight into how the business current operates. What it does not do, is to provide clues about how things might be done differently to improve attendance at showings.

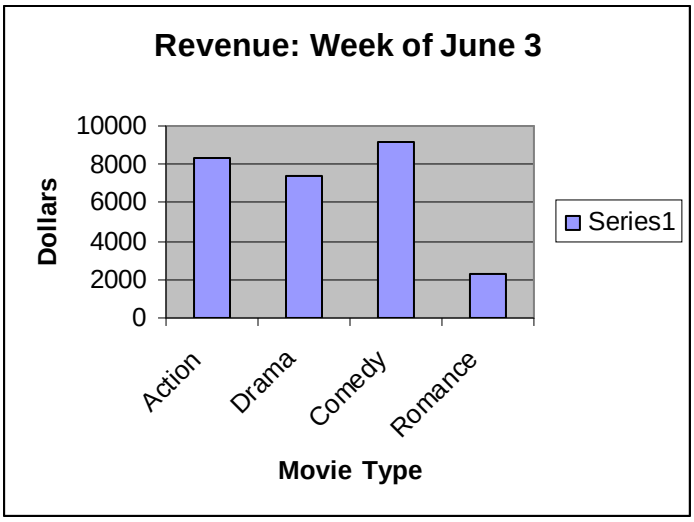

Figure 2: Revenue by Movie Type

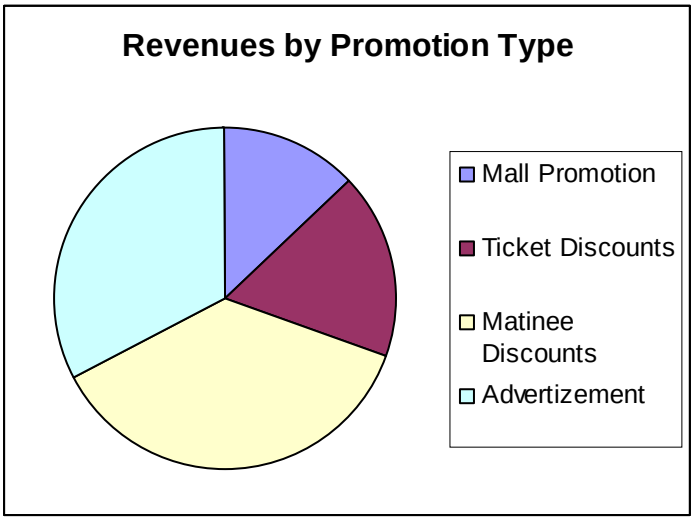

Figure 3: Revenue by Promotion Type

Figure 4 shows Actual Attendance at showings versus the Average Attendance. This report provides more insight. For example, we can see that Showing 3 improved quite dramatically. This suggests that something was done right in management's attempts to improve showing attendance. From this point a data cube can be constructed to drill down and find out why attendance improved at this showing. Was it the promotion? Did the type of promotion work particularly well with this movie or this theatre. Not only does this report focus management's attention on where improvement occurred, it reduces the scope of OLAP analysis to a specific area thus making the analysis more efficient. The designer should develop a suite of such reports and show how they can be used in the decision making process or in guiding the analysis which in turn would aid in the decision making process. Actually, much more in-depth analysis needs to take place and we need both reports and OLAP tools to do that analysis. But, for now, we will leave

the reporting requirements a little open ended and go into greater depth later in Chapter X on Advanced Analysis.

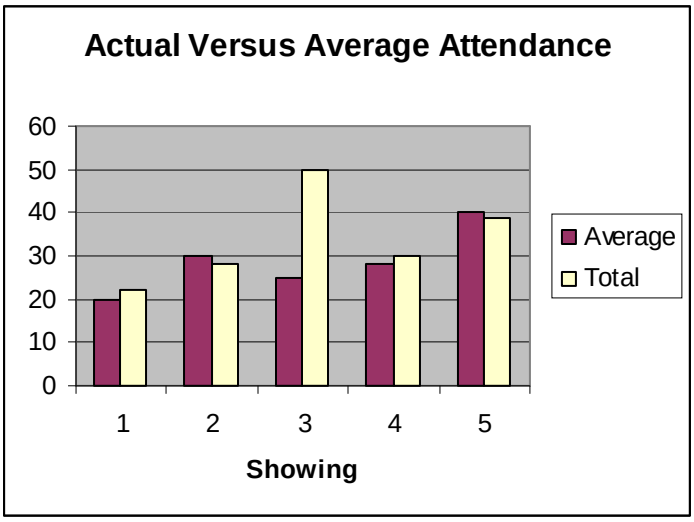

Figure 4: Actual versus Average Attendance

Process Improvement

The goal of this dart mart is the improvement of the process or promoting movies in order to increase attendance at a given showing. Hence, a regular suite of reports, such as those discussed above, should be run every month to show the performance of the promotional strategies. If something falls behind, a deeper analysis of the data through OLAP software may reveal the reason for the weakness. Further, strategies that worked in the past may not continue to work so the process must be monitored continuously. For example, the local demographics may change, the mix of available movies may change, or competitors may change their strategies. Some promotions may work for a while but then the public may loose interest in them. So data must be continuously collected and the process must be continuously monitored. However, at some point that process will be as under control as it can be for the foreseeable future. At that point the theatre owner may want to consider selecting another process to model. Perhaps they may want to model the movie selection process or the concession sales process. Or they may believe that their performance on other processes is adequate and potential improvements do not justify the effort of creating another data mart. Either way improvement is a continuous process and management must continue to monitor the modeled processes and continue to be vigilant in their efforts to find further means of improvement.

Is This Really the Way They Do Things?

One final question that we must ask before we leave this case study is – Is this really the way they do things in managing movie theatres. And the answer is - Probably not! Since the ability to collect data on business processes has been somewhat limited historically we do not see many businesses set up to take advantage of data warehouse technology. This is an important point because one cannot pursue data warehouse design believing that appropriate processes already exist in an organization and that the process of data warehouse design is merely the modeling of those existing processes. No doubt theatre managers know that promotional activities will improve the attendance rate at movie showing. They almost certainly know that different promotional strategies will perform differently for different movies at different theatres and different showing. What they have almost certainly not done is to think of this as a process that can be manipulated in an almost scientific manner and whose outcomes can be measured. So the process of data warehouse design begins with the identification or construction of measurable business processes followed by the measured needed to effectively ascertain their performance. Once these processes are identified they, in essence, structure the application area so that it can be improved. And that structure must in turn be superimposed on the application domain. That is the price of performance improvement. The cost is the cost of doing things differently and the benefit is the benefit of doing things better. “If nature were comfortable,” says Oscar Wilde, “man would have never invented architecture.” If businesses could improve reliably on their own, we could never had invented data warehousing.

Summary

This chapter has provided a case study of how data warehousing can be used to improve the process of selling seats to the showing of a movie in a theatre. The design process was discussed in a step by step fashion providing both insight into the design process and a concrete model that can be referenced in future chapters that examine the design process in more detail from a more conceptual perspective.

Fundamentals of Data Warehouse Design

Chapter 4: Data Push versus Metric Pull

In Chapter 1 we discussed some important distinctions between relational databases and data warehouses and their respective paradigms for modeling the world from different perspectives. The view of data warehousing that was presented in that discussion was not the only possible view. In fact there is a widely accepted competing view. In this chapter we will explore these two competing views which are being referred to in this discussion as Data Push versus Metric Pull. Not only will these two perspectives be compared, but justification will be provided for adopting the Metric Pull approach in this book. However, before we embark on drawing these distinctions, we need to reflect for a moment on why distinctions are important.

Why Distinctions are Important

In Chapter 1 we when through a somewhat exaggerated analytical example attempting to determine the exact nature of a database. By developing a definition of a database we can distinguish between things that are databases and thing that may look similar to a database but are not. Why care about such distinctions? Why not just let anyone call anything they want to a database? These distinctions are important because databases have some key features in common that make them databases. It is these features that we can use to define what constitutes a database and from these features we can extract theories that will help us to build better databases. By advancing the theory of databases we can develop more powerful databases and develop better models of some aspect of the real world. By developing better models we can gain greater insight and in turn manage that phenomenon to our advantage.

A simple example can illustrate this idea. Relational database theory asserts that in order for a database to be considered relational the data must be represented as a collection of tables. Normalization further requires that those tables have attributes with atomic values. So a storage mechanism that allows any type of record structure storing any type of object is not a relational database. Structured Query Language is based on the premise that the database for which the query is being written is relational. SQL does not allow tables to contain arrays or spreadsheets, for example. So by making distinctions between databases that are relational and databases that are not we can develop an advanced exploitation language like SQL. Relax the tenets of relational database and we loose SQL. Along with loosing SQL we loose the ability to exploit the information in the database. And with that we loose the purpose of most modern databases.

So when we make distinctions we can enforce restrictions on a definition that will allow us to advance the theory. And that is why distinctions are important. In this chapter we will make an important distinction between to competing conceptual models of data warehousing. And the model that is supported in this book is the model that provides the best theoretical basis for the advancement of data warehousing. The basis for this claim

will be explained for both the short term and the long term. But it is this potential for advancement of our understanding and the pragmatic benefits associated with that greater understanding that justifies making this distinction.

Why Ideal Types are Important

The databases that we encounter in the real world are far from ideal from a conceptual perspective. There are databases in which the tables are not properly normalized. There are databases containing documents, spreadsheets and images despite the relational requirement that attribute values be single valued facts. There are databases in which facts are recorded multiple times. There are databases which have tables that do not readily correspond to a well defined collection of entities in the real world. And yet the relational data model still requires that tables be normalized; that attributes be single valued facts; the facts be represented only once; and so on. So, why bother with the idealized view of databases represented in the relational model. There are two reasons. The first reason is a theoretical reason. Without an idealized model such as the relational data model, it would be nearly impossible to advance the theory of databases at the logical level. Without a theory of data at the logical level, databases would still be where they were in the pre relational days when database theory was limited to the physical level. At this level all theory would be implementation dependent and everything we know about databases would change each time the technology changed. So it would be difficult to accumulate any understanding of databases over time. The second reason is more practical. Although the relational model is not perfect, it does suggest a correct way to design a database. And although few databases adhere strictly to the relational model, at least designers know (or should know) why databases should be designed in concert with the principles of the relational mode. So when a design trade off is made, such as de normalizing a table for performance, it is possible for the designer to effectively evaluate the tradeoff.

Ideal types are not new nor unique to information systems. Plato suggested that idealized forms existed in a non material world accessible only by the intellect. These idealizations were important, according to Plato, in the pursuit of knowledge. One cannot make very many generalizations about a particular tree, for example, but when we create an ideal form of a tree we can make general statements that may or may not be true for a particular tree. Galileo introduced the notion of idealizations into physics. When students of physics work out problems involving an ideal spring or free fall in a vacuum they are working with idealizations. A particular spring made of an imperfect alloy or an object falling through air have complications that make it more difficult for us to advance our knowledge. More recently Max Weber introduced the concept of ideal types into social science for exactly the same reason. Ideal types cannot be found in the irregular world of particulars but without them it is difficult for us to advance our knowledge.

All this discussion of ideal types is to justify the fact that some important distinctions have been made up to this point regarding idealized notions of relational databases versus idealized notions of data warehouses. One might argue that in the world of real relational

databases and real data warehouses these distinctions do not exist. And that may very well be true. But in order for us to advance our knowledge about and understanding of databases we need to work with ideal types. So next we turn toward an ideal type of data warehouse, one constructed using metric driven design. The concept of metric driven design will be explained and this model will be justified as an ideal type. This ideal type is the basis for data warehouse design used throughout this entire book.

Leading up to Metric Pull

Data warehousing theory and technology have been around for well over a decade and for most of that time they have held out the promise of being the next hot technology. But, they have just not fully blossomed. Sometimes, it seems as though they "have a lot of potential and always will", as the saying goes. Granted web technologies eclipsed data warehousing for a good portion of that time. And related technologies such as data mining and business intelligence have taken some of the spotlight. Even OLAP tools seem to have gotten more attention than data warehousing even though the two are inextricably related. This raises the perplexing question - what keeps this important technology from realizing its potential?

There may, actually, be several answers to this question. It could be that the technology has not yet caught up to the theory. It could be that computer technology ten years ago did not have the capacity to deliver what the theory promised. Or it could be that the ideas and the products were just ahead of their time. Let's consider these answers for a moment. First, is the technology lagging behind the theory as was the case with relational database? This is actually a trick question. Unlike, relational databases the early concepts of data warehousing did not have much theory. So the technology could not lag the theory because the theory did not exist. Next, is it possible that computer technology ten years ago did not have the capacity to deliver on the promises of data warehousing? It is certainly true that computer technology ten years had a much lower capacity than it does today. But to blame technology for failing to deliver misses the point. The promises of data warehousing were unclear. Somehow if you collect a bunch of old data and analyze it, you can make better business decisions. This conceptualization is fraught with problems. What are the characteristics of the old data? What analysis needs to be done? Which decisions does that analysis serve? How do we know that the decisions are better? Finally, were the idea and products ahead of their time? The ideas were weak as was just shown. And the products are just starting to come into their own. So this cannot be the problem. The real answer, I believe, is that data warehousing is in the process of undergoing a radical theoretical shift and that this paradigmatic shift will reposition data warehousing to meet demands of the future. In order to explain this paradigm shift a personal anecdote will serve nicely.

In the field of Information Systems, a critical factor for remaining successful over time is to focus only on the most significant technologies. It is simply not possible for a single individual to keep up with every new technology and maintain any reasonable level of competence and expertise. Another critical factor is the ability to identify which

technologies are likely to be significant before the rest of the world realizes it. This gives you some lead time and avoids the problem of having to master a complex technology in short order once it has already become significant. Imagine that a completely new programming language begins to dominate the technical environment and you have to come up to speed in short order. You can do this by immersing yourself in the language, reading every book you can find on the subject, and writing all the standard test programs. Yes, you can do it. But you can't keep doing it. If you plan to reside in the technological realm for the long term you have to figure out a way to avoid having to do this. So I am always scanning the technological horizon for emerging technologies that promise great significance.

The most significant technological advance in recent memory was the introduction and evolution of web technologies. One could lay out the technological discontinuities on a timeline beginning with the introduction of the mainframe computer, following by the introduction of the personal computers and culminating with the introduction of web technologies. These technologies are significant because they resulted in radical changes in the technological environment and in the way that business was conducted. That is to say that both developers of the technology and users of the technology had to adjust to the impact of the new technology. Numerous other technologies along the way such as natural language processing, CASE tools, expert systems, object oriented databases and the like promised huge changes but those changes were never realized.

I wondered what it was about web technologies that allows them to have such significance whereas other very promising technologies failed to thrive. After reflecting on this question at length, it occurred to me that breakaway technologies were the result of a convergence of factors – a collection of things that needed to come together at the same time otherwise the significance would not reach a critical mass. At the risk of being overly simplistic those factors can be reduced to three. First, a technology has to come of age. That is the technology must be available to provide specific capabilities that were not previously available. Second, those capabilities must address a significant human need. That is, the things the technology allows you to do must be things that people feel a strong need to do. Finally, the technology must be sufficiently easy to use that the maximum number of people possible can exploit the technology to do things they need to do. This all sound pretty obvious but the mainframe computer may not have caught on the way that it did if businesses were not growing an in dire need of some way to get the vast amounts of information they were generating under control. The personal computer may not have caught on the way that it did if individuals within those corporations were not trying to wrestle control of corporate information away from centralized departments of data processing. And the web may not have caught on if the technologies did not come of age in an era of globalization.

Most people who are involved in any significant way with information systems were bowled over by the rise of web technologies. The mid to late 1990's were characterized by a constant struggle to keep up with these rapidly evolving technologies and many people were sadly left behind. It occurred to me that it would be prudent to attempt to

anticipate the next major technology and try to prepare a little ahead of time. So I began to ponder what then next killer app might be.

It occurred to me that Data Warehousing had a lot of potential for being a significant technology. I based this on several reasons. First, data warehousing technology was starting to come of age. We were seeing more software products on the market and OLAP capabilities were showing up all over the place. Second, relational databases had enjoyed several decades of dominance in data management and data warehousing seemed to be the next step. Third, after decades of information systems and the rise of web technologies the amount of data collected by businesses was just beyond comprehension. And most of that data was only being used in the most superficial of ways. Finally, the 20th century saw enormous productivity and quality advanced in manufacturing, but little of that knowledge on process improvement had been applied to white collar work. Data warehousing technology seemed to be a technology that could bring total quality management and process improvement to the office. So this convergence of factors boded well for data warehousing and I decided to develop a data warehousing course.

As I began to develop the materials for this course it did not take long to see that there were at least two distinct, and largely incompatible, views of the nature of a data warehouse. This distinction crystallized one day when a prospective student, who had several years of industry experience in data warehousing but little theoretical insight came by my office to find out more about the course I would be teaching. “Are you an Inmonite or a Kimballite,” she inquired, reducing the possibilities to the core issues. “Well, I suppose if you put it that way,” I replied, “I would have to classify myself as a Kimballite.”

The issue that she was trying to get at, in her somewhat informal way, was whether or not I viewed the dimensional data model as the core concept in data warehousing. I did, of course, but there is, I believe, a lot more to the emerging competition between these alternative views of data warehouse design. One of these views, which I will call the “data push” or “data driven” view of data warehouse design, begins with existing organizational data. This data has more than likely been produced by existing transaction processing systems. It is cleansed and summarized and is used to gain greater insight into the functioning of the organization. The analysis that can be done is a function of the data that was collected in the transaction processing systems. This was, perhaps, the original view of data warehousing and, as will be shown, much of the current research in data warehousing assumes this view. This is also, most likely, what she meant by the Inmonite view after data warehousing guru William Inmon. However, I will let William Inmon speak for himself and will simply refer to this view as “data push” or “data driven”.

The competing view, which I will call the “metric pull” or “metric driven” view of data warehouse design, begins by identifying key business processes that need to be measured and tracked over time in order for the organization to function more efficiently. A dimensional model is designed to facilitate that measurement over time and data is collected to populate that dimensional model. If existing organizational data can be used to populate that dimensional model so much the better. But, if not, the data needs to be

acquired somehow. The metric push view of data warehouse design, as will be shown, is superior both theoretically and philosophically. In addition, it dramatically changes the research program in data warehousing.

The Data Push View Of Data Warehouse Design

The classic view of data warehousing sees the data warehouse as an extension of decision support systems. Again, in a classic view, decision support systems "sit on top of" management information systems and use data extracted from management information and transaction processing systems to support decisions within the organization. This view can be thought of as a data driven or data push view of data warehousing because the exploitations that can be done in the data warehouse are driven by the data made available in the underlying operational information systems.

There are several advantages to this data driven model. First, it is much more concrete. You do not have to imagine the organization in terms of measurable business process. In fact, there is little if any design involved. The data in the data warehouse is merely defined as an extension of existing data. Second, it is evolutionary. The data warehouse can be populated and exploited as new uses are found for existing data. Finally, it requires much less up front analysis to determine how the data will be acquired. And there is no question that summary data can be derived, since the summaries are based upon existing data.

However, it is not without flaws. First, the integration of multiple data sources may be difficult. These operational data sources may have been developed independently and the semantics may not agree. It is difficult to resolve these conflicting semantics without a known end state to aim for. But the more damaging problem is epistemological. The summary data derived from the operational systems represents something, but the exact nature of that something may not be clear. Consequently, the meaning of the information that describes that something may also be unclear. This is related to the semantic disintegrity problem in relational databases. A user asks a question of the database and gets an answer, but it is not the answer to the question that the user thinks they asked. When the 'somethings' that are represented in the database are not fully understood, then answers derived from the data warehouse are likely to be applied incorrectly to known 'somethings'. This also, unfortunately, undermines data mining. Data mining helps us find hidden relationships in the data. But if the data does not represent something of interest in the world, then those relationships do not represent anything interesting either.

Research problems in data warehousing currently reflect this data push view. Current research in data warehousing focuses on: 1) data extraction and integration; 2) data aggregation and production of summary sets; 3) query optimization; and 4) update propagation. All of these issues address the production of summary data based on operational data stores.

A Poverty Of Epistemology

The primary philosophical flaw in the data push model is that it is based on an impoverish epistemology. That is to say, when you derive information from a data warehouse based on the data push model, it is unclear what that information means? How does it relate to the work of the organization? To see this issue, consider the following example. If I asked each student in a class of thirty for their age, then summed those ages and divided by thirty, I should have the average age of the class assuming that everyone reported their age accurately. If I were to generate a list of thirty random numbers between twenty and forty and take the average, that average would be the average of the numbers in that data set and would have nothing to with the average age of the class. In between those two extremes there are any number of options. I could guess the ages of students based on their looks. I could ask members of the class to guess the ages of other members. I could rank the students by age and then use the ranking number instead of age. The point is that each of these attempts is somewhere between the two extremes and the validity of my data improves as I move closer to the first extreme. That is, I have a measurement of a specific phenomenon and those measurements are likely to represent that phenomenon faithfully. The epistemological problem in data push data warehousing is that data is collected for one purpose and then used for another purpose. The strongest validity claim that can be made is that any information derived from this data is true about the data set, but its connection to the organization is tenuous. This not only creates problems with the data warehouse, but all subsequent data mining discoveries are suspect also.

A Poverty of Teleology

The second philosophical flaw in the data push model is that it is based on an impoverished teleology. Teleology is that branch of philosophy that is concerned with final causes or purposes. In situations in which the purpose we are pursuing is not know, it is difficult to know where or not we are doing the right thing and if we are making any progress. In order to see this issue, consider the following example. I decide that I need a vacation so I hop in my car and start driving. After a few days I return home again. Since I did not have a specific goal in mind for my vacation, it is difficult to know when it should be over. So I probably just drove around until I ran out of time or money. Since I did not have any specific purpose in mind for my vacation, such as getting some rest, having a change of scenery, or meeting new people, if somebody asks me if my vacation was successful I really have know way of knowing. The problem with my vacation is a teleological problem. I did not have an end state or purpose in mind so it was impossible to know if or when or how well I achieved it. We see this phenomenon in information systems development all the time. We start building something. We are not sure what we are trying to build so we keep working until we run out of time or money. And when we are asked if the project was successful we have no way of determining whether it was or not. Data push data warehouse design is a classic example of an impoverished teleology. We collect a bunch of old data in the hopes that it might be useful for something. Then, when somebody asks if the data warehousing development was successful or if it can be cost justified we have no way of answering those questions.

The Metric Pull View Of Data Warehouse Design

The metric driven or metric pull view of data warehousing begins by defining key business processes that need to be measured and tracked in order to maintain or improve the efficiency and productivity of the organization. Once these key business processes are defined, they are modeled in a dimensional data model and the further analysis is done to determine how the dimensional model will be populated. Hopefully, much of the data can be derived from operational data stores, but it is the metrics that are the driver, not the availability of data from operational data stores.

A relational database, as was discussed in Chapter 1, models the classes of entities or categories or things of interest to an organization. These things of interest may include customers, products, employees and the like. The entity model represents these things and the relationships between them. As occurrences of these entities enter or leave the organization, that addition or deletion is reflected in the database. As these entities change in state, somehow, those state changes are also reflected in the database. So, theoretically, at any point in time, the database faithfully represents the state of the organization. Queries can be submitted to the database and the answers to those queries should, indeed, be the answers to those questions if they were asked and answered with respect to the organization. To put a finer point on this, consider a user who queries a database and asks something like – how many employees are in the marketing department? Although it is not explicitly stated the queries is really – how many employees are in the marketing department, right now? If this user were to come back to the database and execute this query again a month from now they may not get the same answer, nor would they expect to.

A data warehouse, on the other hand, models the business processes in an organization to measure and track those processes over time. Processes may include sales, productivity, the effectiveness of promotions, customer satisfaction and the like. The dimensional model contains facts that represent measurements over time of a key business process. It also contains dimensions that are attributes of these facts. The fact table can be thought of as the dependent variable in a statistical model and the dimensions can be thought of as the independent variables. So the data warehouse becomes a longitudinal dataset tracking key business processes. This allows the organization to improve its processes and that is the key focus of a Data Warehouse. From an epistemological perspective, the data in the data warehouse corresponds to a measurable business process. From a teleological perspective the purpose of the data warehouse is to improve business processes. So unlike the data drive view of data warehousing, the metric driven view of data warehousing is well grounded philosophically.

Some Almost Fair Criticisms

Advocates of the data driven approach will point to important applications of data warehousing technology that are excluded by the metric driven approach. For example, consider an organization that collects historical sales data and uses it for forecasting future sales. Isn't that a valid and legitimate application? Or what about an organization that collects historical customer and sales data in order to get to know its customers better. That certainly sounds like a valuable use of data. Further, if a data warehouse has to be designed around a measurable business process, doesn't that just make data warehouse design that much more difficult and reduce the number of potential applications? While these are fair criticisms they should be considered within the larger context of advancing data warehousing theory and technology.

First consider the forecasting application. Historical sales data is used to forecast future sales. Sounds like a worthwhile thing to do, at least until you ask the question – how good are the forecasts? What if the forecasts are wildly off. Some products fall drastically below expected sales while other products sell out creating lost sales due to lack of inventory. How would you know this if you didn't track forecasted sales versus actual sales and how could you improve the forecasting if it were not defined as a measurable business process?

The customer information application has similar problems. What information is needed to get to know customers better. Is the answer to this question supposed to be intuitively obvious? Does everybody's intuition say the same thing? Is that intuition correct? How do you know? Given that some information is collected, how do you know if and when you have gotten to know the customers better? Is it when you can describe them demographically? Is it when you can characterize their buying behavior? Is it when you can define the key elements of customer satisfaction? And, what is the point of knowing the customer better? Do you want to increase satisfaction or increase sales or both? Finally, if you want to increase customer satisfaction or improve sales wouldn't it make more sense to define one of these as a measurable business process and design a data warehouse to model it?

The fact is that all of this does make a data warehouse more difficult to design. But the effort that goes into the design pays off in improved business processes. Imagine an organization who's motto is – We could do better, but its just too much effort. Is that an organization that you would want to have anything to do with?

Looking to the Relational Model for Guidance

We can see some indistinct parallels between the state of data warehousing and the state of database prior to the relational model. The relational model was introduced in 1970 but was not realized in a commercial product until the early 1980's. At that time there were a large number of non-relational database management systems. All of these products handled data in different ways because they were software products developed to handle the problem of storing and retrieving data. They were not developed as implementations of a theoretical model of data. When the first relational product came out, the world of

databases changed, almost overnight. Every non-relational product attempted, unsuccessfully, to claim that is was really a relational product. But the claims were not believed and the non-relational products lost their market share almost immediately.

Surprisingly, the relational model held its ground over the next couple of decades and is probably one of the few, if not the only, body of knowledge, in information systems, that has not changed much during that time. Certainly, relational database theory has advanced over the past two decades and the products have become more sophisticated. But the basic tenets of relational database theory have not changed, nor are they likely to. They are a theory of data and the nature of data is unlikely to change.

Similarly, there are a wide variety of data warehousing products on the market today. Some are based on the dimensional model and some are not. The dimensional model provides a basis for an underlying theory of data which tracks processes over time rather than the current state of entities. Admittedly, this temporal/dimensional model of data needs quite a bit of work, and possibly a new name, but the relational model did not come into dominance until it was coupled with entity theory, so the parallel still holds for the most part. We may never have an announcement, in data warehousing, as dramatic as Codd's paper in relational theory. It is more likely that a theory of temporal dimensional data will accumulate over time. However, in order for data warehousing to become a major force in the world of databases an underlying theory of data is needed and it will eventually be developed. Once it is developed, if developed properly, it will become along with relational database theory one of the few true theoretical pillars of databases and information systems.

Summary

Data warehousing is undergoing a theoretical shift from a data push model to a metric pull model. The metric pull model rests on a much firmer epistemological foundation and promises a much richer and more productive future for data warehousing.

Bibliography

Inmon, W. et.al. (2000) Exploration warehousing : turning business information into business opportunity. John Wiley & Sons.

Inmon, W. (2002) Building the data warehouse. John Wiley & Sons.

Kimball, R. (1996) The Data Warehouse Toolkit. John Wiley & Sons.

Kimball, R. et.al. (1998) The Data Warehouse Lifecycle Toolkit. John Wiley & Sons.

Thomsen, E. (2002) OLAP Solutions. John Wiley & Sons.

Chapter 5: Data Warehouse Design

This chapter will provide the conceptual structure for data warehouse design. Parallels between relational database design and data warehouse design will be exploited to provide a familiar three level model of design. The three level model will then be embedded in a concrete design process with well defined steps that overcomes the epistemological and teleological problems discussed in the previous chapter.

Relational Database Design

It is easiest to begin with relational database design and then draw comparisons between that and data warehouse design. This approach has two benefits. First, relational database is far better understood so we can leverage our understanding of that design process to gain insight into the data warehouse design process. Second, by making some important distinctions between relational database design and data warehouse design, we can see that data warehouses are not just relational databases with a lot of old data in them.

Traditionally, the relational database design process consists of three levels of design: conceptual design, logical design and physical design. At the conceptual level the designer determines the contents of the database by ascertaining the things of interest in the application domain. We call these 'things of interest' entities and to avoid the useless and counterproductive complexity of having a different table for each individual entity we group these entities into entity types and each table represents an entity type. We do this by abstracting the essence of the entity type and use that essence to enforce a uniformity on all occurrences. For example, in the university domain we may have an entity type called *Student* and we collect the same information about each student even though some students are full time and other are part time. Some students may have nicknames that we ignore while others may have multiple email addresses or multiple cell phone numbers. We identify a set of attributes common to all and of interest to the users of the database. We then enforce that uniformity on all students. A reasonable question to ask, at this point, is whether or not that entity class is real or constructed. If it were constructed, could it have been constructed differently? We will come back to this question later. But for now assume that we have determined the entity classes that we wish to represent in the database. These classes are documented in an information model. It bears mentioning again that these entity classes are categories and all data in a correctly designed relational database will be stored in categories. Typical queries will ask questions such as how many entities occurrences are there in a category or how many of the entity occurrences in a category have a certain characteristic.

The second step in the relational database design process is logical design. This is the stage where we convert entity classes into tables. We use a process called normalization to create tables that are, hopefully, free from update anomalies and capable of providing accurate information to a wide variety of users via a high level exploitation language such as SQL. In order to achieve this we must insure that each table corresponds to a single entity, that individual rows in that table correspond to specific occurrences of that entity,

that the entity occurrences are uniquely identified by the primary key and that all other attributes are facts about the occurrence identified by the primary key.

Update anomalies are updates to the database that cause the database to become inconsistent. For example, if a fact were represented more than once in a database it is possible to update one occurrence of the fact and not another. This would leave the database in an inconsistent state because a single fact cannot have multiple values.

Sometimes a uses poses a query to a data base and gets an answer, but the answer is not the answer to the question the user asked. We call this phenomenon semantic disintegrity. This is a serious problem for relational databases. The value of the database is derived from the value of the information in the database and its value in making better decisions. If the database delivers incorrect information the users will not trust the data and will not use it. Getting incorrect information from a database is somewhat like going to a restaurant and finding a bug in your food. You are much less likely to return unless there are no other choices.

Finally, once the logical tables are defined we have to be concerned about performance. The final stage of relational database design is physical design where we make decisions about the location of the data, if the database is distributed, and the indexing strategy. Physical design decisions will impact the performance of the database but not its delivery of information. All potential exploitation is determined in logical design. Physical design will determine the speed with which a query can be executed but will not affect the potential results of that query, nor will it have any affect on which queries can be realized.

We have (somewhat quickly) gone through the relational data process because it is fairly well understood and that understanding can be used to better understand the data warehouse design process. Next we turn to the conceptual, logical and physical stages of data warehouse design.

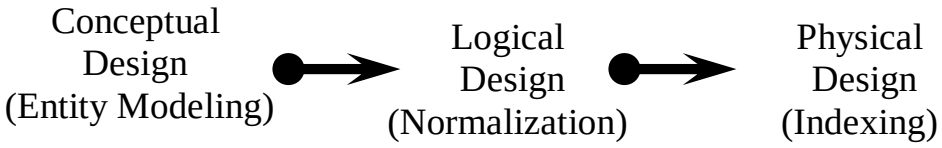

Figure 1: Relational Database Design

Data Warehouse Design

The stages of conceptual, logical and physical design apply to data warehouse design equally as well although the specifics at each stage vary from the specifics that occur at that stage of relational database design. This is good news because, if we learned

anything fundamental about database design from our experience with relational databases it should extend, in some fashion, to other kinds of databases. At the conceptual level we identify a key business process instead of an entity class. A key business process is a collection of intentional business activities that is measurable and worthy of improvement. There are three elements in that last statement that bear emphasizing. The activity must be intentional. It must be measurable. And it must be worthy of improvement. Let's consider each of this issues a little further.

We want to consider business activities that are intentional rather than secondary effects or side effects. There are two reasons for this. First, if we are going to all the trouble to design and build a data warehouse to model a business process, we would like it to be an important business process. We want it to be something critical to the success of the business like producing revenue or sales, cost control, customer satisfaction, employee retention or the like. Often times, the pursuit of one of these direct effects produces secondary effects such as getting more organized, developing greater discipline, or improving the effectiveness of decision making. And all of these secondary effects are, well, secondary. We need to focus on the primary intentional processes where effort is directed in order to improve the health and well being of the business.

The second element is measurability. We cannot know where we stand with a business process unless it is measurable. If we cannot know where we stand we cannot know if we are improving. And if we do not know whether or not we are improving, we cannot know if any specific set of actions or decisions were successful. Having a business process that is not measurable is like going on a diet but never stepping on a scale. Let's say, for example, that we wish to make our organization a better place to work. This certainly seems like a worthy goal, but how would we know when it is better? We could try to rely on anecdotal reporting or the subjective experiences of select people. But neither of these is truly reliable. We would have to develop a metric of some kind through which we could measure whatever we mean by "a better place to work". We may consider employee retention or turnover. We might survey employees on some regular basis. We may consider a less direct measure such as reducing the number of sick days or the number of complaints to human resources. How measures are chosen will be discussed in greater detail later. But there must be some way to measure the process that we are trying to improve or there is no way we can tell if we are, indeed, improving it.

Finally, the process we are considering must be worthy of improvement. Let us consider, for the sake of example, a company that produces a product for which the demand is relatively stable, has been stable for many years and is likely to continue to be stable for the foreseeable future. Does it make any sense for this company to develop a data warehouse to model and track sales performance? Probably not. If the sales have been stable and are likely to continue to be stable then there is not likely to be anything that the company can do to improve sales and there is little that the company needs to worry about short of becoming negligent and irresponsible that will dampen sales. In this case, the sales process is not worthy of improvement because sales are good. They are stable. And they are likely remain that way.

Before moving on let's consider one further example of worthiness. Let's say that the employees of the Acme Widget Company arrive at their desks between 7:30 and 7:40 each morning. The difference results largely form how soon they get to the parking lot and how far away they have to park. Getting there early is no guarantee that they will get to their desks on time because the farthest parking lots are opened first in order to manage the traffic flows. On the other hand if they arrive later there is more congestion in the parking lot. If they arrive even later they have to park are the far end of the closer parking lot which increases their walk. Some park far away so there cars don't get door dings. So Acme decides to build a data warehouse to model the time that employees arrive. We can measure the number of minutes after 7:30 that an employee arrives. So we have a measurable business process. We can consider the day of the week, the parking lot they parked in, and the department they work for. As a motivator we can provide incentives such as cash bonuses, time off, or awards of various kinds. So this looks like a workable dimensional model. The question is – is this process worthy of improvement. And the answer, hopefully obvious, is – of course not. This is silly. This process is not worthy of improvement.

If the employees arriving late is leading to missed sales calls then sales is the process that we need to model. If the problem is productivity then we need to figure out a way to measure productive and model that process. If the employees arriving as much as ten minutes late to their desks does not really affect anything measurable then the company should not be concerned about it.

Key business processes produce something and that something must be measurable. For example, sales is a key business process and it produces revenues. Revenues are measurable. We can count dollar sales or unit sales but either way the process of producing sales also produces a measurable result. We may decide to use profit as a measure of the process. One might argue that the process that produces revenue is not the same as the process that produces profit. This is true and reflects an important aspect of key business processes. They are not simply there for the taking. They are constructed in an attempt to improve the business.

Another example is customer satisfaction. Customer satisfaction can be measured in a variety of ways. We can send out customer surveys periodically and tally the results. We could count the number of complaint calls. We could track the number of returned products. Each of these reflects a slightly different process and a slightly different way to produce customer satisfaction. In one final example we could envision inventory control as a way to produce cost savings. Those savings could be measured in dollars. Or we could measure other factors that we believe are more directly related to inventory control and less directly related to cost savings. For example, we could measure the time which a product sits in inventory. We could measure the number of times that we need the produce but are out of it. The point is that we construct the process and identify the measures in order to improve something that we think needs improving.

While we are still on the topic of key business processes and conceptual data warehouse design, let's ask the larger philosophical question – are processes real or do we construct

them? So far we have talked about them as being constructed. But why is this significant? This is significant because if one believes that entity classes or process exist in the world then one will go looking for them. If one believes that key business processes are constructed then one will approach things differently. In the first case, a designer would look to the application domain for key business processes. In the second case the designer will not look to the domain but look to what the domain would like to achieve. If one views key business processes as inherent in the application domain then the database can only model the existing domain but cannot make it any better. If one views key business processes as constructs then it is possible to design a data warehouse that will actually improve the application domain as well as the process being modeled.

In the movie theatre case that was discussed in Chapter 3, it would be silly to say that the process of filling seats at a given movie showing did not exist. Certainly efforts were made to fill as many seats as possible. However, it is likely that the management of theatre did not conceive of these activities as being part of a collective whole where certain activities produced certain results and where those results could be measured in order to track the progress of those activities and improve the process. It is not until these activities are grouped together and identified as a key business process that the process actually exists despite how many hints may have existed in the way things were done previously.

The second stage of data warehouse design is the conversion of the key business process into a dimensional data model. This is discussed in detail in the next chapter so will not be addressed fully here. However, we can compare the logical design steps used in relational database design and data warehouse design. In the conceptual design step of relational database design we determine the categories that will be represented in the database by identifying entity classes. In the conceptual design step of data warehouse deign we determine the activities that will be represented in the data warehouse by identifying processes. In the logical design step of relational database design we insure that the information is represented correctly and that it will not be corrupted through update anomalies. The process of normalization insures, among other things, that the database will not fail to represent the domain accurately simply because an update produced an inconsistency. It also insures that a correctly written SQL query will return a correct result set. The data warehouse does not have the update anomaly problem since it is not updated with transaction data. But it does have the problem of representing the domain correctly. And it does have the problem of allowing the analyst to derive summary information correctly. To insure this we must insure that the facts in the fact table are indeed dependent variables and the dimensions represent independent variables that influence the dependent variables in some way. A relational database that is correctly designed at the logical level allows exploitation by a wide variety of users. A data warehouse that is correctly designed at the logical level also allows exploitation by a wide variety of users.

Moving to the level of physical design, it can also be said that once a logical design is developed for a relational database, all information that can be derived from that database is known. Physical design may improve performance, but physical design does not affect

the information that can be derived. Physical design in a relational database consists largely, although not exclusively of indexing. In a data warehouse we can make parallel observations. Once the dimensional model is constructed we know all information that can be derived from the data warehouse. Physical design may improve performance, but physical design does not affect the information that can be derived. In the case of data warehousing physical design focuses on the collection of summary datasets that will be computed in advance in order to improve the pivoting performance of data cubes. This process is currently handled by software and is not well understood by most designers. Having the software handle it is good, but has one significant drawback. In order to make good decisions about which summary sets provide the greatest boost to performance, the software would need to have a detailed history of the queries that were done and the cubes that we created. That is to say that software can only perform on historical data and the software will perform best when the historical data is both somewhat uniform and representative of future performance. A person, on the other hand, may be able to predict which queries or data cubes are likely to receive the most attention based on an understanding of the data and its uses. So, for the moment, physical design is left largely to the software, but it is likely that a theory of physical design will eventually emerge. Figure 2 summarizes the three steps in data warehouse design.

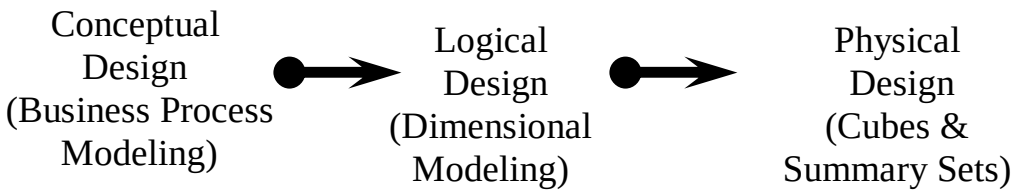

Figure 2: Data Warehouse Design

A Practical? Approach

Gausse and Weinberg tell a revealing story about an ad he once saw in a newspaper for a guaranteed cockroach killer. “Send us ten dollars”, the ad said, “and we will send you a guaranteed cockroach killer.” If you have ever lived in an apartment with cockroaches, you understand how impressive the claim is of a ‘guaranteed’ cockroach killer. So you send in your ten dollars and a few days later the product shows up in your mailbox. In the box are two flat ceramic tiles with instructions – “place the cockroach on one tile and smack it with the other.” Clearly, if you can get a cockroach to stand on one of the tiles you can smack it with the other tile and kill it. But how do you get the cockroach on the tile to begin with? This ‘guaranteed cockroach killer represents a class of solutions to

problems that sound good superficially but upon closer inspection do not really bring you any closer solving your problem. What you done, in essence, is to trade one difficult problem for another difficult problem without bringing yourself any closer to a solution.

To some extent the current view of data warehouse design, according to Ralph Kimball, is a guaranteed cockroach killer. The steps in the current view are:

1) Define a Business Process to Model
2) Define the Grain of the Business Process
3) Identify The Types Of The Measurements
4) Define the Conformed Dimensions
5) Define Conformed Facts Definitions
6) Identify Standard Suite of Reports to Track Critical Success Factors

Admittedly, these steps do provide important insights into the process of data warehouse design. They reveal the importance of business processes, grain, and measurement. They emphasize dimensions and the importance of conforming dimensions and facts. And this collection of steps also brings the importance of data exploitation to the foreground.

However, many questions are left unanswered. Where to do we find business processes? How do we know if the processes that we find in place are the correct processes to model in order to improve them? How do we construct them if they are not already in place? How do we select the grain? How do we measure things? How do we know which dimensions to conform? Doesn't this require that we anticipate other data marts and their dimensions? If we have never tried to improve a given process, how do we know what reports may be needed to track it? And so on. There is a great deal more to data warehouse design than is revealed in this procedural model. And while it is far better than not having a design model, it leaves much to be desired and hopefully research into the data warehouse design process will reveal more about it and some answers to these questions.

Nonetheless, we can, perhaps, take a small step further toward understanding the conceptual design process by taking a problem oriented approach and starting the process by defining a problem to solve.

Defining the Problem

A problem as defined by Gause and Weinberg is a gap, a gap between the perceived state of things and the desired state of things. Note that a problem does not necessarily mean that something is wrong. It means that the state of things falls short of expectations. So it may be that the current state of things is great, but there is potential to be even better. Whenever this gap exists, there is a problem.

The primary benefit of beginning the data warehouse design process by defining a problem is that it provides a focus for the design effort and a criterion for validating the design. If the data collected in the dimensional model will provide the information

needed to solve the problem then the design is a good one. If not, it is not a good design. A secondary benefit of beginning the design process with a problem is that it allows a cost benefit analysis for the data warehousing project. If the problem is solved there should be some economic gain from solving it. That economic gain can be translated into dollars. And those dollars can be compared to the dollars that will be spent to develop and maintain the data warehouse. If the dollars saved do not exceed the dollars spent, then there is not point in building the data warehouse.

It should be noted that not all problems can be solved by data warehousing technology. The weather may be disagreeable. Some employees may not get along with each other. The technological environment may be evolving too fast. The competition in a given industry may be to fierce. Only certain kinds of problems are appropriate for solution using data warehousing technology and those problems should have two specific characteristics. First, the solution of the problem should be quantifiable in economic terms. Second, the nature of the problem should be such that it can be solved by defining a measurable business process and improving that process. If either of those two criteria do not exist for a given problem then a data warehouse is probably not the solution.

Constructing a Key Business Process

Once we have identified an appropriate problem, the second step in data warehouse design is to define a measurable business process that has the characteristic that if we improve the process the problem should be solved. In many cases, this is much easier said than done. If the problem is that product sales are sagging or uneven, we can develop a model to track sales. However, if the problem is an excessive number of customer complaints, then identifying the appropriate process to solve the problem may be a lot trickier. Customer complaints may result from inferior products. They may result from poor service. They could result from misleading advertising. There could be many reasons. In order to solve this process we would have to first determine what is causing the high number of customer complaints and then identify a process to reduce the number of complaints.

This raises another important issue in using a problem oriented approach to data warehouse design, and that is the fact the problem we select may not be the right problem. As we move forward attempting to define the process, we may discover that we need to solve a different problem entirely. For example, we may see the problem as too many customer complaints. But after some further analysis we see that the problem is that the product we are producing has too many quality defects. If that is the case then we need to define a process for improving product quality. Alternatively, we may discover that the problem lies with customer service. Perhaps our customers have to wait too long on hold so they don't get the service they need and this leads to complaints. If this is the case then we need to develop a process for improving customer service response time. Another possibility is that they get through to customer service but the service they get is not up to their expectations. Perhaps the customer service personnel are inadequately

trained. If this is the case then we need to develop a process to improve the quality of customer service.

At this point we should know whether or not a data warehouse will help in solving the problem that we have identified. We need to define a measurable business process that we can improve over time. We need to be able to manipulate that process and measure the results of that manipulation. We need to have a reasonable expectation that if we do manipulate the process and measure the results that we can improve the process over time. Finally, if we improve the process over time the problem should be solved. If any of these are not true, then a data warehousing in probably not the solution to our problem. And we can determine that ahead of time without having to build the data warehouse to find out.

Defining the Dimensional Model

The definition of the measurable business process represents the conceptual stage of data warehouse design. We know what process we are monitoring. We know, to some extent, how we are going to measure it. We know what we can do to manipulate the process. And we know what the intervening factors are that may influence the success of our manipulations. In the logical stage of data warehouse design, we refine these ideas in a more rigorous way until we can represent them in a dimensional model. The dimensional model will center around a fact table that contains measurements of the process. One of the dimensions will represent the manipulations that we perform to improve the process. One of the dimensions should be time since dimensional models are always temporal models. And the remaining dimensions will be the intervening factors that may influence the success of our manipulations. This will be discussed at great length in the next chapter so we do not need to go into great detail here. However, at this point we have a well defined problem that is solvable using data warehousing technology. We have a measurable business process that, if improved, will solve the problem. And we a dimensional model representing that process. All we need do now is to demonstrate that if we were to build this data warehouse it is likely that we will solve the problem. And in order to do that we need to define a standard suite of reports and show how they could be used in the decision making process as we attempt to solve the problem.

Identifying a Standard Suite of Reports

The final step in conceptual data warehouse design is to develop a standard suite of reports and show how the information provided in these reports can be used by decision makers to improve the business process being modeled. These reports should have two important characteristics. First, they should be clearly derivable from the data contained in the dimensional model. And, second, the information they provide should serve to solve the problem upon which the data warehouse was based. At this point, if these two characteristics hold there should be no doubt that the data warehouse, if built as designed, will serve to solve the larger problem. This is important because it is not necessary to

build the data warehouse in order to determine if it will solve the problem. This can be determined analytically at the conceptual level.

The content of this standard suite of reports should focus on revealing the levels of the measurements in the fact table in response to manipulations of the motivator dimension. They should also reveal the effects of mediators on the level of the measurements. Finally, they should show how these effects occur over time. Much is made of the drill down and roll up capabilities of OLAP software and these capabilities are important. But it should be noted that these capabilities are important for more in depth analysis. That is, when the standard suite of reports reveals a problem, drill down and roll up capabilities can be used to ascertain the source of the problem. But for monitoring the progress of the measurable business process, it is the standard suite of reports that are needed.

Learning versus Performing

One final point needs to be made about the data warehouse design process. The process by which one designs a data warehouse given that they know what they are doing is quite different from the process by which one designs a data warehouse given that they do not know what they are doing. If the designer is knowledgeable in data warehouse theory and technology and they have designed numerous successful data warehouses in the past then it is likely that they will be able to easily identify an appropriate problem and a measurable business process that can be improved in order to solve that problem. This is no different than an experienced relational database designer who looks over an application domain and for whom entity classes pop out of the domain. It is a skill developed and refined by experience. Once a skilled data warehouse designer has identified the measurable business process, they can probably define the suite of reports without having to flesh out the dimensional model and they should be able to demonstrate how the information in these reports can be used to improve the process and solve the problem.

If, on the other hand, the data warehouse designer does not have any experience, it is likely they he or she will not fully grasp the process of identifying an appropriate problem. Nor will they have the skills needed to construct an appropriate measurable business process. The inexperienced designer will more likely have to carry the design forward to greater levels of detail in order to define the standard suite of reports because he or she will be unable to anticipate the implications of their design decisions. This is not a problem as long as the designer and everyone else involved in the process realizes that this is a learning process more than a design process. That is, the designer needs to learn how to design a data warehouse and the product of this learning process is unlikely to be a final product.

The process described in this chapter assumes some experience on the part of the designer. It is an idealized design process to which the designer should aspire. If the designer is less experienced, there will be a lot more learning, refining and redesigning needed that revealed here. In fact, if the designer and users are completely new to the

process, it may be a good idea to build a prototype of the design before committing to a full development effort.

Summary

This chapter has described the data warehouse design process at the conceptual level and has compared it with the relational database design process. The essence of conceptual design is to identify an appropriate problem, that is one that can be solved using data warehousing theory and technology, and then defining a measurable business process the improvement of which will solve the problem. Based on the measurable business process a standard suite of reports is identified showing how the information provided can be used to improve the process in question. Finally, an important distinction between learning and performing was provided. If the designer is new to data warehouse design they will be learning about the process as they are performing. Since the output of the learning process is knowledge rather than good design, first designs may need to be revised or tested before they are developed.

Gause, D. and Weinberg, G. (1989) Exploring Requirements: Quality Before Design. Dorset House Publishing.

Chapter 6: Dimensional Data Modeling

This chapter will introduce dimensional data modeling, the basic modeling approach used in data warehousing design at the logical level and show how the dimensional data model is superior to the entity relationship model for modeling measurable business processes. In addition, some data theory will be introduced in order to address the subtleties of dimensional data models. This data theory goes beyond the traditional treatments of dimensional data modeling and serves two purposes. First, it enhances the more traditional treatments of dimensional data modeling by showing how the selection of data types influences the information that can be derived from a dimensional data model. Second, it provides a basis for some speculations regarding the future of data warehousing that will be addressed in Chapter 12. Dimensional data modeling is a logical design technique used to model processes in the same way that normalization is a logical design technique used to model categories. Correct normalization leads to tables that can be easily exploited using a high level language and correct dimensional modeling leads to a star schema configuration of tables that can be easily exploited using an OLAP interface or a multidimensional data language.

Dimensional Data

A point in the physical world is identified by three dimensions height, width and depth. Values for these three dimensions are adequate for locating any point in 3-space. This concept of dimensionality and location in a multidimensional space is used metaphorically to describe data in which the values for certain independent variables will determine the values for the final dependent variable. So in the movie theatre example, if we know the theatre, the showing, and the movie, the value for sales or attendance is determined given that the showing has already occurred. So we can think of the dimensions of theatre, showing, and movie as determining the value for sales or attendance and we can think of all sales values as being located in a metaphorical three space, the values being determined by the dimensions. Of course we can only go so far with this analogy. Unless the dimensions also represent measures we cannot, for example, talk about the distance between two points in the multidimensional space.

So when we design a dimensional model we are attempting to identify the determinants of the fact value. Dimensional modeling is a logical design technique used in designing data warehouses that attempts to capture both facts that measure the process being modeled and the dimensions or attributes that affect the values of those facts. It differs from entity relationship modeling in that it focuses on measurable facts rather than categories and business processes are the unit of decomposition rather than things of interest to the organization.

In the physical world of height, width and depth we can locate any point by providing values for each of these dimensions. Metaphorically, we can think of a theatre ticket sales space where the number of tickets sold is determined by the values for showing, theatre, movie and promotion. So we can think of these attributes as the dimensions of this value. So when creating a dimensional model we have to determine the facts that we will use to

measure the process and dimensions which are the attributes needed to determine those facts.

Facts

The most important table in a dimensional model is the fact table. A fact, in this context is an observation of the performance of a business process. It is the facts in the fact table that will tell us how well we are doing in performing this process. They will tell us which factors influence performance and they will tell us if we are doing any better when we alter our activities in order to produce better results.

Facts should correspond to the grain of the fact table which, in turn corresponds to the dimensions of the fact table. This is a critically important but often overlooked point. Using the theatre example, if we are measuring attendance by showing, by movie, by theatre, and by promotion then we cannot have facts that only measure attendance by movie.

Kimball makes an important distinction between facts that are additive, semi-additive and non-additive. Since this represents the state of the art this distinction deserves some comment. First, additive facts are discrete numerical measures of activity such as dollar sales or units sold. Additive facts can be thought of as measures of flow past a point rather than a snapshot of a value taken at a point in time. Additive facts are additive across dimensions. Ticket sales are an example of an additive fact. Ticket sales can be additive across dimensions so we can talk about total sales for a day, total sales for a movie, total sales for a theatre, and so on. This is the strongest feature of additive facts. They can be summed or averaged across dimension and the results will be meaningful.

Semi-additive facts are measures of magnitude that represent the level of an attribute rather than a flow past a point. The simplest example of this would be an account balance. If I have a bank account that had a balance of $50 yesterday and $30 today, I could not add those two amounts and claim that I have $80 in the bank over the past two days. My account balance represents a magnitude and consequently is a semi-additive value. However, if I have a balance at Bank A of $50 and another account at Bank B with $30, it would be appropriate to say that I have $80 in the bank. On the other hand, I could say, in the first example, that I had an average of $40 over the past two day as I could say that I have an average of $40 in each of my bank accounts in the second example. So semi-additive facts can be summed in some of the dimensions but can only be averaged in others.

Another example of a semi-additive fact would be inventory balances. If I had 50 widgets in inventory yesterday and 30 widgets today, I could not say that I have 80 widgets in inventory over the past two days. However, I could say that had an average of 40 widgets in inventory over the past two days.

Let's consider a trickier example. If I am selling soup at a grocery store I can record total dollar sales and total cans sold. Both dollar sales and total cans will be additive. If, on the other hand, I am selling tickets at a movie, I can record total dollar sales, total tickets sold and total viewers. However, dollar sales and total tickets will be additive but total viewers will not be. If, for example, one viewer saw three movies that would represent three tickets but one viewer. If my goal were to get more people into my theatres, I could not just add up the number of tickets sold. So total viewers is an example of a fact that is not additive over any dimension.

Another more generic example that is probably not particularly relevant to data warehousing but really illustrates the property of being semi-additive is daily temperature. Daily temperature cannot be added in any dimension. Consider temperature readings taken yesterday and today or readings taken in New York and Texas. In neither case they be added in any meaningful way. The best you can do is to average them. And that is the key feature of semi-additive variables. There are dimensions across which they can only be averaged. In the case of account balances or inventory levels they can be added across spatial dimensions and averaged across temporal dimensions. But as we saw in the case of daily temperatures some semi-additive variables cannot be added across any dimension.

Let's consider one final example. Let's say that as customers are leaving the theatre we give them a customer satisfaction survey in which they rate their satisfaction with the movie, the concession stand, the sound and projection quality, and so on using a five point scale. Is this data additive or semi-additive? Well, if one argues that the survey measures utility then one might try to argue that it is additive. However, that is a pretty wobbly argument and the fact is that if two customers report a satisfaction level of five with the movie it would be a stretch to say that the movie generated ten units of satisfaction. So if we cannot add satisfaction ratings, can we average them. Again we get into trouble. Customer satisfaction is a kind of data called interval data and most experts in data analysis would caution against averaging this data. Reasons for this will be explained later. This example is raised to show that characterizations such as additive and semi-additive while somewhat revealing about the nature of data are inadequate and a much more sophisticated understanding of data is necessary for good data warehouse design. This issues is addressed again later in the section called data theory.

One final point about the fact table is that it is possible that facts in a fact table may be used in more than one fact table. If this occurs the fact definitions should be the same across all fact tables. That is facts with the same name should have the same meaning everywhere in the data warehouse. For example, in the theatre example, it is conceivable that in one circumstance total ticket sales may be computed by multiplying the number of tickets sold times the price of a ticket. In another circumstance discounts and free passes may be applied and ticket sales may be determined by the actual revenue taken in at the box office. If both values, total sales and actual sales (or total sales minus discounts) are needed then they should be given different names. Those names should be fully described so that anyone analyzing the data in the warehouse would know exactly what goes into a given number. Finally, standardizing fact definitions makes it possible to do more

sophisticated analysis. For example, one data mart may track the effects of promotions on ticket sales. Another data mart might track the effect of movie selection or showing times. If these data marts use the same definition for ticket sales then it would be possible to determine the relative impact of each factor on ticket sales. That is to say the theatre owners could determine whether promotions, movie selection or show times has the greatest influence on ticket sales.

Dimensions

Whereas the facts in the fact table represent dependent variables, the attributes in the dimension tables represent independent variables. The dimensions contain those attributes that are thought to affect the facts in the fact table. Dimensions can contain controllable variables which we will call motivators or they may contain uncontrollable variables which we will call mediators. All dimensional models must have at least one motivator. That is to say that there must be at least one controllable independent variable. Consider the theatre case in which we are using promotions to fill seats at a showing. The motivator is the promotion. That is something we can manipulate. The impact of a promotion may vary from one theatre to the next or from one showing to the next but the theatres and show times are taken as givens. They are mediators. It is important to have at least one motivator because that is variable that we manipulate to improve the process. Without a motivator we might be able to analyze the data and understand a little better how things works but we would not be able to do anything to effect improvement.

It is also important that dimensions be conformed. That means that if a dimension is used in two different dimensional models it is important that the dimension be designed so that the same logical dimension can participate in both models. There are two reasons for this. The first is that conformed dimensions join fact tables which allows for drill across applications. Assume in the theatre example that we decided to develop a second dart mart to increase concession sales. The motivator could be product or price and mediators could be theatre and showing. If the theatre and showing dimensions were conformed, or shared, with the attendance data mart we could do even further analysis by analyzing concession sales as a function of attendance. By going through the shared dimensions of theatre and showing we could analyze concession sales by theatre and by showing as a function of attendance which would provide even more insight into the process. This may seem like an obvious way to handle this and hopefully it is. However, another way to handle it would be to make attendance an attribute of showing. This would require that attendance data be computed separately and used to populate the showing dimension. In addition, it would not allow additional analysis across other dimensions of the attendance data. As you can see, sharing a dimension has great analytical value. So why not conform every dimension? Well this is easier said that done. Conformed dimensions must have the same grain and depending on how the various processes are being modeled this might be very difficult. One dart mart may be developed at one point in time while another data mart is developed later, perhaps much later. When the first dart mart was developed there were probably several choices for the grain. When the grain was selected future data

marts may not have been considered so conforming dimensions to accommodate future data marts may be difficult.

A second reason for conforming dimensions is that conformed dimensions are reusable thus spreading the cost of creating the dimension across multiple models. In the case of the theatre creating a dimension for the theatres might not be a large effort. However, if a company were to create a customer dimension it might be quite extensive. A customer dimension might be used in one data mart to track the effectiveness of promotions based on demographics and in another data mart to track the effectiveness of overdue bill collections. Since the customer dimension might well contain thousands or even millions of records it would make sense to create one customer dimension and reuse it in every dimensional model for which customer is a dimension.

Grain

The grain of the measurements is the precision with which the measurements are taken. For example, in the theatre case, audience is measured by showing, theatre, movie and promotion. Why not group several showings together and measure by the day or week. Why not group several movies together and measure audience by genre? The only reason why these groupings may not be appropriate is that changing the grain changes the dimensions and changing the dimensions may create problems in the analysis of the data and may prevent us from being able to improve the process. In order to see this, lets consider other potential grains for the theatre problem.

First, consider a more refined grain than measuring audience by showing. Let's consider measuring audience by the hour instead. Since a movie typically lasts for two hours, taking measurements each hour would double the number of measurements and double the size of the data mart. However, with this particular application, that would be the least of our problems. Since patrons of the movie theatre purchase tickets for a whole show, and we are modeling the performance of promotions on ticket sales, it really does not matter whether or not viewers walk out in the middle of a movie. So the refined grain does not buy us anything. Nonetheless, one might argue that information regarding how many people walk out of a movie may be useful for theatre management. This is true, as a hypothetical. That is if this information could be acquired for free then it might be useful to know. But can it be acquired for free? Well, we saw already that the volume of the data would double. But the data collection effort would more than double. We know how many tickets were purchased for a showing from box office receipts. But in order get patron counts at the end of the first hour we would have to send someone into the viewing room to count heads. Or we would have to post someone at the door to count the number of people leaving. So this finer grain would leave to dramatically increased data collection costs. But the costs to do not stop there. Having the grain be refined to hourly measurement rather than measurements by showing would almost certainly confuse anyone attempting to analyze the data. They would wonder why measurements are taken every hour rather than by the more obvious increment of showing. They would feel as though they did not fully understand the model and would be much more tentative in

their analysis and more timid about their conclusions or recommendations. As was mentioned before, anything that increases the confusion of analysts using the data decreases the exploitation of that data. And anything that decreases the exploitation of the data decreases the value of the data warehouse. So the costs of implementing this more refined grain are not even nearly outweighed by the increased cost.

So let's consider going the other way and making the grain more course. Let's say that we measure ticket sales by viewing room rather than showing. A given viewing room may have as many as a half a dozen showing per day so this would dramatically decrease data volume. But, we would also loose important information. The half dozen movies that would show in a given viewing room would almost certainly be different movies with different attributes. Some might draw a large crowd while others attracted only a few viewers. But, more importantly, these different movies would likely respond differently to different promotions so we would loose the most important piece of information of all – how did ticket sales for a give movie respond to a given promotional strategy. Hence, this grain would be too coarse and would render the data in the data mart as useless.

We could do this exercise of considering a more refined or more coarse grain on any of the potential dimensions and indeed the dimensional modeler should do a similar exercise in determining the grain. Too fine of a grain increases data volume and may increase data collection costs. It also may make the model more complex and confuse those exploiting the model thus reducing the value of the data mart. Too coarse a grain may lead to a loss of information that may yield the data mart useless in providing insights that would improve the process being modeled.

Grain determines dimensions and dimensions determine grain. So a reasonable question to ask is – should you start with the grain or the dimensions. There is no easy answer to this question and under different circumstances different starting points might be appropriate. However, as a general guideline it is best to begin with the dimensions because they are the independent variables. So begin with a set of dimensions and use those dimensions to determine the grain. Then consider the implications of coarser or refined grains. These different grains will lead to different dimensions and the implications of those different dimensions can be considered.

Legacy Accommodations

Data warehousing has been around for over a decade without having a solid theory of dimensional data as a foundation. Further, most of the work that was done in data warehousing over this time period was done by consultants and practitioners who were trying to create data warehouses to accommodate historical data from operational systems. Consequently, we find the literature peppered with accommodations for legacy systems. The following instances are often found both in the literature and in practice but are not, in general good data warehouse design. They are accommodations at best.

Non Additive Facts

Kimball provides two examples of non additive facts – a factless fact table and a text fact. A factless fact table records an event that occurred where the event did not produce a measurable activity but needed to be recorded. A text fact may describe an activity rather than measure it. Hopefully, in both cases the problems with these non additive fact tables are obvious based on previous discussions, but let's elaborate on each. First, consider the factless fact table. The name itself is a bit of an oxymoron. How can you have a fact table without any facts in it? Let's say that we are mailing surveys out to customers and we want to record whether or not the survey was returned. We can envision a fact table which contains a single flag, yes/no, or zero/one to indicate whether or not the survey was returned. We can flesh this out a little further and imagine that the dimensions are customer, survey, time and incentive. It begins to look like a regular data mart to improve the response of surveys. However, there are no facts in the fact table. The question is – what do we do with the data to improve the response rate? Presumably we would summarize the data to determine the response rate with respect to each dimension. And possibly summarize the customer dimension by demographics. But when we summarize the data into response rates we are creating facts on the fly. If we have to tally the data each time we summarize along a different dimension exploitation of the data mart is going to become difficult. What we should have done was to use the response rate as the measure of the process and not leave the data in a more primitive form. So a factless fact table probably indicates an incomplete design or a hurried implementation.

Text facts, as indicated above, describe an activity rather than measure it. Let's say that a salesman is making sales calls and we wish to record the contacts so we create a fact table that contains information about the contact. What are we going to do with the data? Presumable we will aggregate the data as we did with the factless fact table and derive some measure of performance. Perhaps we want to determine the number of contacts per salesman per day. Or maybe we would like to know the percentage of contacts that led to a sale. Whatever information we are interested in will be a measure, of some kind, and thus suggests that the fact table should contain that measure. The text fact table indicates, as did the factless fact table, that the design was incomplete and implementation proceeded too quickly.

One might argue, in both of these cases, that the designer might not know which measures are needed and implementation of a non additive fact table is necessary to find out which measures are needed. There is some validity to this point, however, at some point when the desired measures are determined the data mart will have to be rebuilt. And the question one has to ask is whether or not the expense of building the first data mart is justified or if it is possible to think through the problem a little more thoroughly and avoid the expense of having to rebuild.

Faulty Dimensions

Similarly legacy systems also have some faulty dimensions. These include degenerate dimensions and junk dimensions. A degenerate dimension is a key value in a fact table that does not expand into a dimensional table. In the movie theatre example assume that

each theatre had a unique number and that number was recorded in the fact table, but no theatre dimension was needed because there was no addition information about the specific theatre. In this case, theatre would be a degenerate dimension. It is degenerate because it does not expand into other attributes which can be used in further analysis. We can, of course, look at attendance by theatre but without knowing any other factors about the theatre we are limited in what we can discover about why attendance may vary from one theatre to another. In the case of a degenerate dimension we have to ask if it is really a dimension after all and if it is why can't we expand it with further information.

A junk dimension is a separate dimension consisting of a collection of attributes, codes, or flags from existing source systems that do not map into the conformed dimensions of the dimensional model. Junk dimensions are of limited value but are sometimes included in a dimensional model when the designer of the model fears loosing potentially useful information. After all, these attributes had value in the source system so why wouldn't they have value in the data mart? An all too common justification for including junk data in a dimensional model is that it doesn't hurt to have extra data and it is better to have it in case you need it than to not have it when you do need it. This is a faulty justification for several reasons. First of all, inclusion of unnecessary data confuses the users who exploits the dart mart. If they don't understand why some data items in the model are important they will just assume that they do not fully understand the model. They will be less bold in their analysis, again assuming that they do not fully understand the model. Anything that reduces the extent to which a user exploits the data in the model reduces the value of the data mart. And anything that reduces the value reduces the benefit. A second problem occurs when somebody does actual use this junk data in their analysis. Since it is junk data it is not tied in any fundamental way to the process being modeled. Hence it will be difficult to understand the meaning of any analysis that uses the junk data. Again this will confuse the user who exploits the data and will diminish the value of the data warehouse.

The problems just discussed, as mentioned earlier, result from the fact that there are many legacy data warehouses developed over the past decade or so which viewed a data warehouse largely as a repository for old data. The dimensional model added some theoretical underpinnings to data warehousing by asserting the need to model measurable business processes. This is just the beginning of a theory of multidimensional databases and much more theory is needed. The next chapter will begin to lay some foundation for such a theory by borrowing some ideas from data analysis. But before moving on to data theory we need to look at a currently unsolved problem in current data warehousing theory and implementation – the problem of changing dimensions.

An Unsolved Problem: Changing Dimensions

The theory of data warehousing is still emerging and there are a number of interesting problems that still need to be solved. One of these problems that has significant and obvious practical value is the problem of slowly changing dimensions. In general, dimensions should not change. If they do we loose our ability to track the influence of an

attribute over time. But sometimes this is unrealistic. For example, in the movie theatre case it is possible that we might refurbish theatres on some regular basis and that information should be captured in the theatre dimension. It is conceivable that the time since refurbishment may have an impact on attendance. If we had a customer dimension it is easy to see how certain attributes of the customer such as age and income would change over time and those changes would be relevant to our analysis and should be recorded. The difficult question is how do you handle changing dimensions? This is a difficult question to answer because the solution to slowly changing dimensions must meet three conflicting objectives. First, we would like to maintain the temporal continuity of our analysis. That means that we need to be able to track the influence of the stable attributes of a given dimension over time. Second, we would like to do this in a way that does not make our analysis inordinately difficult. And, third, we would like to keep our dimensions up to date with respect to the attributes that do change.

There are three potential solutions to the changing dimensions problem. First, we can just update the attributes that do change. This seems like a reasonable solution until we realize that updating attributes in a dimension makes historical analysis meaningless. We cannot go back, after updating a dimension, and look at the historical impact of those attributes simply because they no longer have their historical values. This fails the first goal since it eliminates our ability to maintain the temporal continuity of our analysis. A second solution would be to keep the old dimension record with its unique key and add a new dimension record with a new key. This would allow us to maintain the old information and the new information. However, if we wanted to do some analysis on the unchanging attributes that analysis would get very difficult violating the second objective. Let's say that the theatres in the movie case were numbered with integers (1, 2, 3, and so on). When we change a dimension we could create version numbers such as 1.1, 1.2 and so on. This would allow us to update the dimensions as often as we like but the numbering of keys would create no end of confusion in analysis. Imagine if we updated the showings dimensions using similar scheme and we wanted to create a data cube with theatres as one dimension and showing as another. Would we include theatres 1.2, 2.4, 3.2, and 4.1 along with showings 1.1, 2.1, 3.3, and 4.2. It is easy to see how this would become unwieldy very quickly. A variation on this theme would be to limit updates and have the dimension record have two sections – one for current data and one for data prior to the update. This would violate the third objective of keeping the dimensions up to date and the second in that it would make the analysis very awkward.

The only remotely reasonable solution, and admittedly not a very good one, would be to timestamp every dimension record. Each time a record is changed it would be time stamped somehow to indicate the timeframe for which it is valid. Then when data is extracted for aggregation, the extraction software would match up the appropriate values. This would have to be done in the software somehow because if it we left up to the user it would make exploitation far too complicated. This is not a particularly good solution because saying that time stamping must be handled in the software somehow only nods at a potential solution without really providing one. Nonetheless, the theory of dimensional data modeling is still fairly new. There are a number of areas in which work needs to be done and this is certainly one of them.

Summary

This chapter addressed dimensional data modeling in greater depth explaining the nature of facts and dimensions. A discussion of grain revealed that there is a reciprocal relationship between grain and dimensions. That is, a given grain will determine a set of dimensions and a given set of dimensions will determine the grain. Clearly some iteration in design is needed to find the right grain and the right dimensions to solve the problem at hand. In order to provide a counterpoint between the evolving theory of dimensional modeling and legacy data warehousing implementations some legacy design accommodations were discussed. Finally, an open problem in data warehouse design, that of slowly changing dimensions was introduced.

Chapter 7: Data Theory

When we create a database we are, in effect, modeling some aspect of the real world in data. Thus, the database designer is a modeler, of a sort, who has to choose the correct constructs in developing the model. Unfortunately, our understanding of those constructs and their implications is not very well understood at the present. In many cases the constructs that we have are quite limited. Bits and pieces of what we need to know can be found in the semantics of information modeling or the process of normalization within the relational data model. Statistics and data analysis hold further clues. It is possible that database design and statistics are two sides of the same coin; one focuses on the design of data sets while the other focuses on the exploitation of them. A poorly designed data set will be limited in its potential for exploitation and a well designed but largely unexploited dataset provides little, if any, value. And yet few database designers are knowledgeable in statistics beyond some basic parametric data analysis and few statisticians are even aware of the basic tenets of database design. So the purpose of this chapter is to begin laying out a basic theory of data. This theory will serve two purposes. First, it will provide further insight into the most effective way to design data warehouses which merits its inclusion in this book. And, second, it will serve as an outline of sorts for further study and development of an integrated theory of data.

First, a Word About Models

To see the utility of constructing models consider the following highly simplistic but very much on point example. Let's say that a person works at a rate of $5 per hour. One way in which we might pay this person is to give them $5 at the end of each hour they completed. While this might work it is not very practical. We would have to keep a stack of five dollar bills on hand and as our work force grew the process of handing out money at the end of each hour would get complicated, especially if the workers did not all start at the same time or did not get paid at the same rate. Perhaps a better approach might be to let a worker accumulate a number hours, say a weeks worth, and then pay them. So a worker might claim, at the end of the week, that he had worked forty hours. We could then give them $5 for the first hour, another $5 for the second hours and so on until we had handed him one $5 bill for each hour that he had worked. This sounds ridiculous and indeed it is. Even the most mathematically unsophisticated person should see that it would make much more sense to multiply the number of hours worked times the rate per hour and pay him the computed amount. And in doing this, we have created a mathematical model of wages due. This, of course, is so commonplace that nobody thinks of this as a mathematical model but it is and examining why we would construct such a model provides some important insights about using models.

First, this model provides an economical model of wages due. If the labor market varied the way the stock market does then computing wages would be a nightmare especially for anyone who did not have a computer. But people tend to get paid regular amounts at regular intervals which leads to a relatively high degree of predictability in how much is earned and how much is owed. Second, this model is efficient. That is computing pay for the week is much more efficient than simply laying out a five dollar bill for each hour

worked. Third, we interact with the model rather than interacting with the world, again due to the economy and efficiency of the model. It is much easier to compute pay for a time interval than to follow someone around and hand them a five dollar bill at the end of each hour. Fourth, the model is more reliable. If somebody works for forty hours at $5 per hour they should get paid $200. It would be hard to make an error with this. However, if you were following a person around and paying them at the end of each hour a variety of circumstances might occur that would result in an hour's pay being missed leading to disputes between employees and employers. Similarly, handing out a $5 for each hour one at a time might cause a person to loose count, again resulting in incorrect pay and all attendant problems. Finally, and perhaps most profoundly, the construction of this model superimposes order on the world. While it would certainly be possible to pay an employee a different amount each hour we tend not to do that because it would limit our ability to model wages mathematically. Having this mathematical model of wages earned not only makes it easy to compute pay for one employee, it allows us to compute aggregate pay by extending the model. For example, if we have X employees who get paid $5 and hour and Y employees who get paid $10 per hour we can not only compute but predict how much we will owe in wages before the work is even done.

This mathematical model of wages earned is so obvious and so commonplace that we tend not to think of it as a model. We tend to think of it as a direct representation of reality. But it is a model and seeing it as such provides insight into why we construct models. Yet, mathematical models are not the only kind of models. There are many kinds. One of those kinds of models is the data model and we construct data models for exactly the same reasons that we construct mathematical models.

Modeling the World in Data

The first problem we encounter when approaching the problem of how to model the world in data is the lack of clarity and consistency in the vocabulary that is commonly used to discuss this activity. What is the difference, for example, between an information model and a data model? What is the difference, for that matter between data and information? There are no end of attempts to define the differences between data and information but these are usually little more than proffered definitions with little conceptual foundation and certainly limited utility when one considers the vast inconsistencies that each attempt gives rise to. But let us not dally on that problem.

The Entity Relationship modeling technique is one of many techniques referred to generically as "information modeling". It is a technique for identifying categories of information and the relationships between those categories. So Entity Relationship appears to be an information model. That model agrees with Parmenides as discussed in the first chapter in that it adopts a static, categorical view of the world. It is, in fact, an ontology in that it makes assumptions about the very nature of reality. But ontology has taken on other meanings within information systems so we have make that observation cautiously.

Nonetheless, when we use the Entity Relationship modeling technique to model a domain we may come up with a more specific model that consists not of entities and relationships but of students and professors and courses, assuming that we had applied this technique to a university domain. This model, consisting of students and professors and courses would also be referred to as an information model. So we used an information model to create an information model. Should there be some difference in terminology? Perhaps one might be called a first order information model and the other a second order information model. Or one might be called a template model and the other called a derived model. There are many possibilities. But one thing is clear. The rule for constructing and applying these two models are quite different and using the same name for both of them just leads to confusion. But it gets worse. The information model consisting of students and professors and courses may well be a generic model of a university. This model may then be applied to specific university in order to design a database specific to its needs. This tailored model would be derived from the generic model and once again the rules for constructing and validating the derived model would be quite different from the rules for constructing and validating the generic model. If the generic model were considered a second order model would the derived model be considered third order? If so, what are the boundaries between the two? Is it possible to have a tenth or hundredth order model by making each one more specific?

Let's go through the same reasoning, although somewhat abbreviated with the relational model. We have the relational data model which defines relationships between data items. If we use the relational model to design a relational database for a specific application area, say the university consisting of students, professors and courses, then we have a data model of the university. This could be a generic model as before and could be further refined to model a specific university. Hence, we have the same levels of derivation that we had with the Entity Relationship model. So we have a great deal of confusion about the difference between data and information. More confusion about the difference between information models and data models. And yet even more confusion about the difference between levels of application of these models. No doubt there are database experts who could provide a nomenclature that would satisfy himself or herself. But there is no widely accepted nomenclature to clear up this problem from a conceptual perspective. And the questions do not stop there.

The Entity Relationship model and the relational data model are both categorical models of data. That is, they represent a world that is organized into categories based on type. It is reasonable to ask if there are any others. Certainly, the as yet unnamed model underlying data warehousing is not a categorical model. It is a process model. In fact, it models measurable business processes. While the dimensional data model does have a name the conceptual model does not. Nonetheless, we have at least two models based on different ontologies. This has already been discussed in some detail in Chapter 1. So we can move to the next question. Are there any other data models?

In order to answer this question, it is useful to take a step back and think of a data model in its most general sense as a representation of some aspect of the world that we interact with because it is easier than interacting directly with that aspect of the world. In the case

of relational databases we model some aspect of the world in information so that we can query that model to find out things about that aspect of the world without having to actually interact with that aspect of the world. So when an inventory manager, for example, looks at a report that summarizes inventory turnover and inventory costs he is really interacting with an information model of the inventory. He is not actually counting parts on shelves. The parts are represented in the database and the database is updated whenever the inventory changes. Thus, if he wishes to know how many of a given part are on hand, he does not go to the shelf or the bin. Instead he goes to the database. If he wants to know the current value of the inventory, he does not go to the warehouse, he goes to the database. This tendency to model the world in information and then interact with those models has become so pervasive that the distinction being made here probably seems hairsplitting. But it is not.

Now, let's consider another quite different example. Let's say that you go to an online bookstore to buy a book. You may look up your favorite author. You may browse the best sellers. You may take a much more rigorous approach and do an index search. Once you have located the book you are seeking you may read a couple of pages of it, check the price and then decide to purchase it. To purchase it, you put it in your shopping cart and go to the checkout. A few days later the book appears on your doorstep or in your mailbox. You did not actually visit a bookstore. You did not even visit a virtual bookstore. What you did was to interact with an information model of a bookstore and the result of the interaction was the book showing up on your door step. The extent to which we interact with information models of the world is increasing so dramatically that we don't even think of the information models as intermediaries any more.

As the role of data in modeling the world around us becomes increasingly more important we have to get a better understanding of data and that requires a theory of data. Taking a top down approach to defining a theory of data does not seem to be working. The existing theory of data models is a patchwork of inconsistencies. Perhaps a bottom up approach beginning with data types would be more productive. Clearly, a unified theory of data is way beyond the scope of this book but an attempt to begin laying down some data theory from the bottom up beginning with data types is not. In the relational data model the characteristics of data are not a major factor. There are data types so that data items can be declared as integers, floating point numbers or character strings. The selection of a data type has implications but not tremendous implications. If a number is an integer we cannot use it accurately in computations that require a decimal value. If the item is a character we cannot, or at least are not supposed to use it in a calculation at all. There are some specialized types such as currency and date time which make certain uses and calculations a little easier. But by and large the implications of data types in the relational data model are limited. And to the extent that we do consider data types, we consider them as computational data types as opposed to ontological data types. That is, integers and strings are based upon computer representations of data rather than descriptions of things that exist in the world.

Data Types

At the conceptual level we describe data items in the relational model as naming, descriptive, or referential. Naming attributes identify unique occurrences of entities and referential attributes show relationships between the current entity and another entity. Neither of these data items are particularly useful for analysis although the referential data items do provide linkages between entities which make correlations possible. The descriptive data items are facts about the entity occurrence referenced by the naming attribute. Some are useful for analysis, some are not. For example a person's address or phone number is probably not useful for analysis nor is their email address. But, if the descriptive attribute is categorical such as sex or numerical such as age or salary then it might be useful for analysis. The point here is that the type of the data item will have an impact on the analysis that can be done and, in general, the limitations on this analysis, in the relational data model, are rather severe. In the data warehouse, they are not.

In the data warehouse, much more sophisticated analysis is possible. So the selection of data types is much more critical. Ultimately, data warehouses will evolve into multidimensional information systems where dimensional models become the basis for mathematical and statistical modeling of the organization. Consequently, we need to begin integrating more data theory into the data warehouse design. So to begin this process let's quickly survey the types of data that we would recognize from a statistical perspective – nominal, categorical, ordinal and cardinal.

Nominal data, as the name implies, names or labels things. In the data warehouse, data items that name things can be keys into dimensions. So for example if there is a theatre dimension, as in the case study, the theatre identifier would be nominal data as would be the address. However, the address is descriptive data and as such is not generally useful in the data warehouse. The reason for this is that the address is not useful for purposes of analysis. In a relational database the address would be a useful data item because if may be used in generating a mailing label or a shipping order. But in the data warehouse we are modeling the process of filling seats for a moving showing and the address of the theatre as a descriptive attribute does not serve any analytical purpose.

One might argue that the location of the theatre could have an effect on the process of filling seats. For example theatres in lower income areas might perform differently than theatres in a higher income area. Or theatres in an area with a large elderly population may perform differently from theatres in an area with a largely working population. And these claim may be valid. However, address is not the variable to record this information. If this kind of analysis is useful then separate variables must be constructed to record this information. That is to say that nominal variables should not be used to construct pseudo categories. If we want to analyze the data according to categories then we need legitimate categorical variables not pseudo categorical variables. And that brings us to our next type of data item.

A categorical variable, as the name implies, is an attribute that associates an observation with a category. Categories are useful because membership in a category may influence

the process being measured. Hence, categories should also be meaningful. For example, assigning theatres to categories based on their location relative to other stores is a meaningful category because it is believable that theatres located in an enclosed mall might perform differently under certain conditions than theatres located in a strip mall or theatres in stand alone locations. It is not reasonable, on the other hand, to think that theatres whose street address begins with an 'S' would perform and differently than theatres whose street name begins with an 'M'. So this would not be a meaningful category. As was mentioned already it is important to define meaningful categories and it is also to important to avoid extraneous categories. Although it make seem like a good idea to include as many categories as you could possibly think of, just in case they turn out to meaningful, there is a cost associated with this. First, categorical attributes must be populated and the more categories one identifies the more data that must be collected when populating the dimensions. This in turn increases the cost of creating the dimension tables. However, if the dimensions are not large this is not the biggest problem. The biggest problem is that the more data items you have in the data warehouse the harder the analyst has to work to find relationships and the more likely the analyst is to get confused. Anything that confuses or inhibits the analyst will reduce the extent to which the data in the warehouse will be used in decisions making. Since the value of the data warehouse is a function of it use in decision making, anything that inhibits this use reduces the value of the data warehouse. An excessive number of categories may serve to confuse or inhibit the analyst and hence may ultimately reduce the value of the data warehouse.

One final caveat regarding categorical variables is that categories can take on numerical values and when they do those numerical values must be handled carefully. For example, if the numerical values 0 and 1 are used for gender, it is possible make sense out of an average gender. If the average gender is, say 0.45, then this means that 45% of the observations are Female and 55% are male. However, if an average zip code is 22101, it is hard to make any sense at all out of that average.

The next kind of data is ordinal data and is commonly found in a data warehouse that is tracking measurements such as customer satisfaction. Ordinal data is numeric data that can be compared. For example in a typical customer satisfaction survey a customer might be asked - How satisfied were you with the service you received? Possible responses might include Extremely Satisfied, Very Satisfied, Satisfied, Somewhat Dissatisfied, Very Dissatisfied. These responses could then be assigned numerical values say Extremely Satisfied = 5, Very Satisfied =4 and so on. At a minimum this data could be treated as categorical data and we could report 23% of our customers were Extremely Satisfied with the service they received, 36% were Very Satisfied and so on. However, ordinal data can also be compared. Being Extremely Satisfied is better than being Very Satisfied which, in turn, is better than just being Satisfied. But ordinal data gets tricky when you try to sum or average it. The reason for this is that even though being Extremely Satisfied is better than being Very Satisfied, how much better is it? Since Extremely Satisfied receives a 5 whereas Very Satisfied receives a 4, does that mean that Extremely Satisfied is 20% better than Very Satisfied? Probably not. Further, if Very Satisfied receives a 4 and Somewhat Dissatisfied receives a 2, does that mean that being

Very Satisfied is twice as good as being Somewhat Dissatisfied? Again, probably not. I say probably not because it is possible that this five point scale does indeed measure customer satisfaction levels uniformly however, it is also very unlikely. So what does one do with these numbers? Well, despite the fact that they should not be summed or averaged, people do it anyway. So it is not unusual to hear something like customer service satisfaction rose from an average of 3.8 in 2003 to 4.2 in 2005. That is to say that just because there are theoretical problems associated with averaging ordinal data that does not mean that people do not do it anyway. However, it would probably be better to say some thing like only 38% of our customers were Satisfied with our customer service in 2003 whereas 65% were Satisfied in 2005.

The next kind of data is cardinal data. Cardinal data can be used for comparisons. For example four dollars is twice as much money as two dollar whereas a customer satisfaction rating of four could not be considered to be twice as good as a rating of two. Money has this characteristic because it is ratio scale data. This subtle but important feature can be seen with a simple example. Cardinal data that does not have a meaningful zero is called interval scale data. This means that two equal distances on the scale represent the same amount. For example, 50° is 20° warmer than 30° and 70° is 20° warmer than 50°. So interval distance is meaningful. But 60° is not twice as warm as 30° so ratios are not meaningful. The ratios are not meaningful because the zero in the temperature scale is arbitrary. It does not represent an absence of heat. It is merely assigned at a relative location and is not consistent across scales. 0° Fahrenheit and 0° Centigrade are not the same temperature. So the zero is not meaningful. However, if you have no money in your pocket, it does not matter if you have no dollars or no pounds. You have zero money. Since the zero is meaningful in money, we can compare amounts in ratio. Hence we call it ratio scale data.

One final observation about cardinal data is that it can be continuous or discrete. Continuous data can take on any point on the number line whereas discrete data can only take on certain points, usually whole number, on the number line. Variables like age, height and weight are continuous variables because, presumably, they can take on any value within their allowable ranges depending on the accuracy of the measurement. If a person put on ten pounds increasing their weight from 120 pounds to 130 pounds, presumable at some point in their weight gain they weighed 124.343292 pounds, if only momentarily. However, discrete data can only take no limited values. One can buy 5 or 6 cans of soup. But one cannot buy 5.418 cans of soup. When you say something like – the average customer buys 5.37 cans of soup per month you are confusing continuous and discrete data. While this may be a useful characterization to indicate a level of purchasing, no body walks into a grocery store and comes out with 5.37 cans of soup.

In some cases such as money we blur the distinctions between continuous and discrete data. Money is not limited to whole numbers. If a can of soup costs $1.39 one can pay that amount whereas one cannot buy 1.39 cans of soup. But money is limited to a precision of two decimal points. If we say that they average consumer spends $7.4643 a month on soup it is again a characterization to indicate a level of purchasing, but nobody

actually spends $7.4643 because pennies are the smallest denomination of money that we have.

These different kinds of data are important in data warehouse design because different types of data will allow different analysis which, in turn, leads to different insights into the phenomenon being modeled. To more fully understand this we need to understand the various goals of our analysis. We will begin with the premise that our data warehouse models some aspect of the organization or application area. We can interrogate this model in order to more fully understand the organization or application area. This leads to a distinction between two types of models – descriptive and predictive. A descriptive model, as the name implies describes a domain. If one wants to more fully understand the domain they can interrogate the descriptive model. A predictive model, on the other hand, shows how the domain will change under certain circumstances. In other words, it predicts outcomes. If you want to know what is going on, a descriptive model is adequate. If you want to know what to do next a predictive model is more useful. But going from what is to what should be is a big step and predictive models are generally a lot more difficult to construct than descriptive models. To see this we need to make some further distinctions between classification and measurement, and association and correlation.

Classification, Associations and Relationships ←

When we classify an object we assign it to a group according to some system of classification or some principle. When we assign employees to departments, for example, we are, in effect, classifying them. Classification allows us to ask questions about individual employees such as – Does Bob work for the Marketing Department. It also allows us to asks questions about a category such as – How many people work for the Marketing Department. And we can compare categories with questions such as – Is the Marketing Department larger or smaller than the Research Department.

When we measure we attempt to determine the extent of a quality according to some standard. In the example above we classified employees according to their department. We could also measure employees according to their salary by asking - how much money does each person make? We could also determine the minimum, maximum, average and total salary for employees and again computer these values for any subset of employees such as those that work for a particular department. Since we can compute these values for each department, we could then ask if there is some relationship between department and salary such as – do the employees in the marketing department make more money than the employees in the research department. This leads to the idea of associations and correlations.

It is useful to be able to find relationships between variables because the value of one variable may affect the value of another. The weakest form of relationship is between two categorical variables. We call this association. For example one might ask if a person working for the marketing department is more likely to be a woman. This is an

association between two categorical variables. It is a weak relationship because the value of one does not predict the value of the other. It merely indicates a propensity for the value of the other. A stronger form of association can be found if one of the variables is categorical while the other is a measurement. For example, we could ask, as we did above, do the employees in the marketing department make more money than the employees of the research department? A stronger relationships could be defined between two measurements. For example, if an employee's salary increased with years of service we might find a strong correlation between these two variables. This would be particularly useful if that correlation was strong enough to have some predictive power. For example, if we could construct an equation so that given years of service we could predict salary that would be the best possible outcome. So when we choose the types of data that we collect we are also limiting the analysis that can be done. As we limit the analysis, we limit the extent of insight we can get into the phenomenon we are modeling and in turn limit the extent to which we can improve the process. Clearly it would not make sense to force measurements where categories are appropriate. But, at the same time, we want to select variables that will give us the greatest power possible given the nature of the underlying phenomenon.

Much More To Do

A statistician reading this survey of data types might well shrug and say – "So what? You have not really said anything new here." And that observation would be correct. What is new here is an attempt to integrate statistics with database design. Much of what was just said would be new to the average database or data warehouse designer. And if the previous sections were, instead, a short tutorial on normalization it would be new to a statistician. But this book is about data warehouse design and the primary audience is database designers. So the brief tutorial is aimed at them. But this just scratches the surface of data theory. There are some profound philosophical questions to be addressed. How are categories formed? How are processes defined? The former of these two questions is derived from a metaphysical problem know as the Problem of Universals. The second is derivative of the Problem of the Persistence of Identity and the philosophical problems associated with measurement. The point of this chapter was to show that a more rigorous approach to a theory of data is possible. Design decisions have implications and the practice of database design could be much improved if there were a theory of data that would allow the database designer to see what those implications may be. Here we saw the implications of the choice of data types on potential analyses. What are the implications of different types of categories, or different types of processes. In fact, what are the different types or categories and what are the different types of processes. Yes, there is still much more to do.

Summary

This chapter was an attempt to begin the development of a theory of data. First the notion of models was introduced followed by the idea of modeling the world in data. Then a brief theory of data types lifted from basic statistics was provided to show that the

selection of data type has implications for the exploitation of the data. More generally, design decisions have implications and a theory of data is need so that the database designer has more insight into those implications. But this only scratches the surfaces. Some addition problems were suggested that might point the way for further research into an integrated theory of data.

Chapter 8: Case Study: Using Web Log Data

Unlike the other case studies in this book that begin with a specific application, this case study begins with a simple dataset – the web log – and asks just where could we go given an initial dataset in an attempt to construct a data warehouse. It is an example of the most extreme bottom up approach to data warehouse design. That is, it shows how to start with a dataset, determine metrics that are derivable from that data set and then use those metrics to construct a data mart. For those who may be unfamiliar with the nuts and bolts of web technology, the web log is a data set that contains a record for each visit to a website. So when you go to your favorite website information is recorded in the web log not only of the initial page that you retrieve, but of every link you click on within that site there after. In their simplest form the records in the web log represent clicks on hyperlinks that point to the server maintaining the log. So a user clicks on a link. A request for a page is sent to a web server. And that web server records the request. It doesn't sound like much to go on. But it is. And we can take that minimal data set and do some pretty impressive things with it.

Cognitive Barriers in Data Warehouse Design←

This case studies also addresses a second and perhaps more subtle issue in data warehouse design and that it the problem of cognitive limitations on the part of both users and designers. Herbert Simon introduced a concept in problem solving called bounded rationality. The essence of this concept is that problem solvers automatically limit their potential solutions to those that are close enough to optimal with respect to their perception of the resources available to solve that problem. What this means from the perspective of database design is that when users are asked what information they would like from a database, they are unlikely to imagine useful information that they do not believe is possible to have given their current understanding of information that is readily available. So the traditional approach to database design of going to the potential users of the database and asking them what information they would like to have is of questionable value. It is often also difficult for database designers, who do not have a great deal of experience, to image what information might be possible if a data mart were constructed.

Another interesting perspective on this problem is provided by Gause and Weinberg. The Railroad Paradox [Weinberg, 1988, Gause and Weinberg, 1989] is an anecdote about a railway company used to illustrate an important point about information systems development. In this story a railroad train stops in Homeville each morning and takes commuters to work in Workville. In the evening the train picks up the commuters in Workville and returns them to Homeville. During the course of the day, the train passes through Homeville on the way to Workville but does not stop. Residents of Homeville would like to catch a mid afternoon train to Workville for shopping and other reasons so they write the railroad company and ask that the train stop on the 2:30 pm run. The railroad company sends representatives to Homeville to observe the station in question at 2:30 pm. While they are there they see no people waiting for the train, conclude that there is no demand, and decline the request. The point of the story, of course, is that demand

for the train does not exist until the train starts making a regular stop, at which time the demand for the train will develop.

In many cases the demand for information from a database or a data warehouse does not exist until the database or data warehouse is actually available. Once users begin using the available data to inform their decisions they can think of more ways in which to use the available data. But until the data is actually available, they cannot imagine how they would use the data and hence there is no demand for it. This is a serious problem for database designers because the information requirements must be derived in some way other than asking the potential users of the data. This case shows one way in which these limitations can be over come. We begin with a simple dataset and work our way up to a data warehouse.

The Web Log

A visitor to a web site requests a page by typing in the address of the page, in a web browser, or by clicking on a link that automatically requests that page. A message is sent to the web server, at that address, and the web server responds by returning the page. If the page contains graphics, the web browser sends additional requests to the web server to retrieve those files also. Each request that is processed by the web server is recorded in a file called the web log, which contains a record of all activity on the web site. The record typically contains the date and time, the IP address of the requestor, the page requested and the result. Web log records can be more elaborate but for the purposes of this bottom up example this simple structure is adequate.

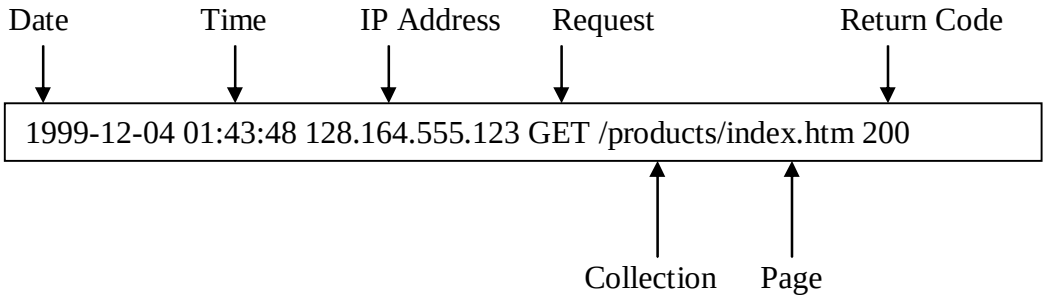

Figure 1: A Minimal Web Log Record

From this simple log record we can determine quite a bit about the traffic coming to the web site. For example, from the date field we can ask: 1) When do we get the most traffic?, 2) Are there peak days? 3) Is traffic to the web site uniform across a typical week or cyclical ?, 4) Is there more web traffic on weekends or weekdays? Answers to these

questions might help the webmaster in determining the best time to do maintenance or when to roll out new collections. From the time field we can ask: 1) Are there peak load time? 2) Do visitors come during the workday or in the evening, 3) At what point during the day is the traffic the lightest?, or 4) Is traffic spread out evenly throughout the day? Answers to these questions would be useful in determining when to do backups or when to take a server off line for maintenance.

Each request contains a path to the page from the root page of the web site. If the directories are set up so that the beginning of the path name corresponds to an identifiable collection of pages, then we can determine the number of visitors by page and by collection. For example if the URL for a given page is of the form http://domain/collection/subcollection/page.html then we can summarize traffic volumes by sub collection and page. From this information we can answer questions such as: 1) Which collections are the most heavily visited?, or 2) Which pages are the most heavily visited? Answers to these questions would be helpful in determining how to allocate development resources or which page designs are the most effective for attracting visitors.

Fields such as Date or Time can be combined with Collection to answer questions such as: 1) Are different collections visited more heavily at different times of the day?, 2) Are different collections accessed more on weekends?, or more generally, 3) Do traffic patterns vary by collection?

The type of request is usually a “GET”, however, other requests may indicate a form being submitted or a collection owner performing maintenance to the collection. Other operations might indicate a search engine probing the site or a hacker testing security. The Return Code indicates the final resolution of the request. A “200” means that the request was satisfied. A “401” indicates that the visitor did not have the appropriate authority to access the page. From these two pieces of information, we can answer further questions about the web site such as: 1) How many requests are visitors versus collection owners?, 2) Are there dead links?, 3) Are permissions set incorrectly?, or 4) Is there evidence of hacking?

A Web Log Dimensional Model

From just these six fields (Date, Time, Request, Collection, Page, Return Code) in a flat file format from the web log, the web master can determine quite a bit of information about the usage and traffic patterns on the web site. However, by viewing the web log as a data source for a data warehouse, it becomes an even richer source of data about the web site. Figure 2 shows a preliminary dimensional data model derived from the web log.

By converting the web log to a fact table and adding both Date and Time dimensions, it becomes possible to do additional analysis. For example, the Date dimension can identify specific days as holidays, weekdays, deadlines, sales or promotions or any other event that might be noteworthy for analysis. The Time dimension can identify time periods

such as daytime, lunchtime or evening, although this gets a little tricky since visitors may be in any time zone when they access a page.

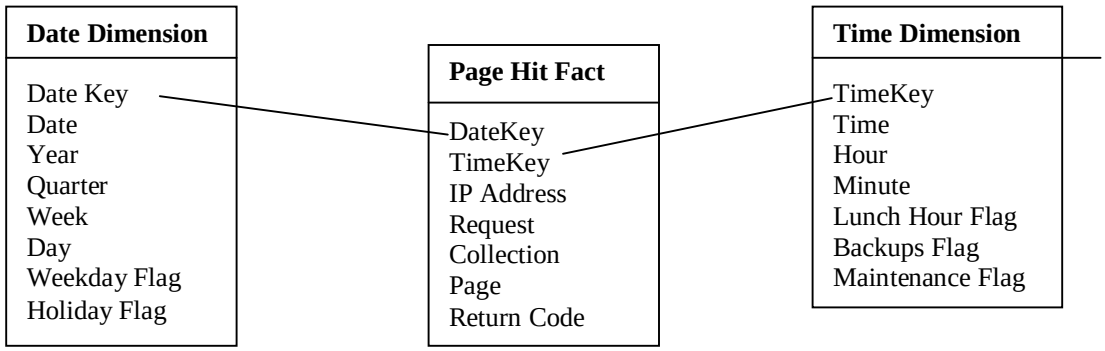

Figure 2: A Preliminary Dimensional Model Based on the Web Log

In the data staging process, the records in the web log that did not reflect page requests would be filtered out. This would include the requests for graphics files and would substantially reduce the volume of the web log. There may be other reasons why the graphics file requests would be left in, but for this example, the focus is on page hits.

With a little extra work on the dimensions, the model could be further expanded, as shown in Figure 3, to include dimensions for IP Address and Page.

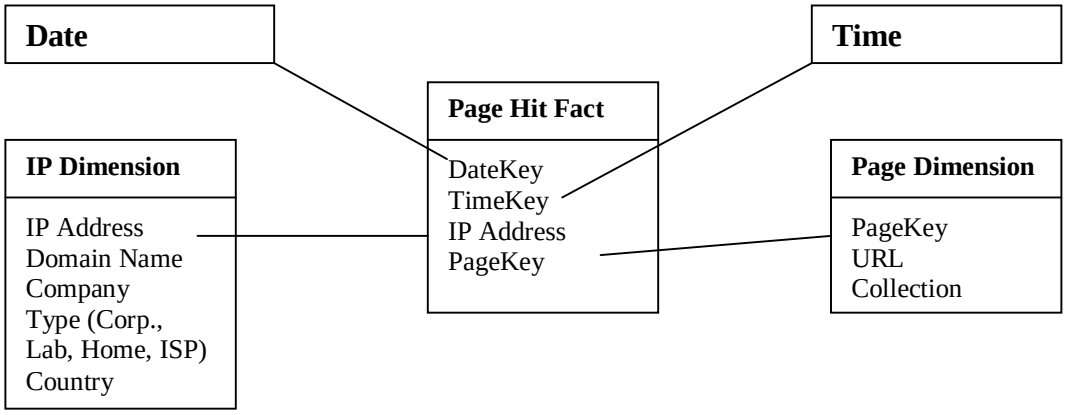

Figure 3: An Expanded Dimensional Model

The IP Address dimension allows the web log analyst to determine where the visitors are coming from or if certain collections attract visitors from different quarters. From this information it is possible to determine if the target audience is being reached. The IP Address dimension can be maintained at a variety of levels of granularity and completeness depending on the information needed. Keeping only three levels of IP address will usually identify a company or a service provider, but will not identify individual computers as four levels would. Three levels of IP address would require 16 million records, while four levels of IP address would require 4 billion records. Assuming a 100 byte record which provides ample room for the domain name, company name and country, this dimension table may get quite large. One option for reducing the size of this dimension table would be to snowflake (which is the dimensional model equivalent to normalizing) the table. This would reduce the duplicated character strings for domain name, company name and country to integer pointers into separate tables and reduce the storage requirements by up to 75%. Another option for reducing the size of this table would be to maintain records only for IP addresses that have actually visited the site. Even if a site has had ten million unique visitors, this is still only 0.25% of the 4 billion possible IP addresses.

This raises an important point about data warehouse design. Some dimensions, such as IP address can be very large and very costly to create. This is one of the reasons why conformed dimensions are important. Conformed dimensions can be reused in several data markets thus spreading the cost of developing a dimension over several applications. Further, enhancing the richness of a dimension may be useful for gaining greater insight into a process. But the cost of enhancing the richness may not be justified for a single data mart. Reusing the dimension makes it more worthwhile to make the dimension as rich as necessary to provide any needed insight into the factors that influence the process being modeled.

A second important observation about the preliminary and expanded dimensional models shown in Figures 2 and 3 is that they both contain fact less fact tables. The fact tables record events that occurred rather than measurements of a business process. It was mentioned in Chapter 6 that a fact less fact table might be an indication that the designer needs to think through the problem a little more thoroughly and this is a great example of exactly just that. Leave the fact table without a measure seems to leave the greatest flexibility but falls short of good dimensional modeling. Although there are a number of possible measures that we might derive from the data in these models, we need to pick one. For purposes of this case we will focus on dwell time.

Dwell Time

As was just mentioned, the models shown in Figures 2 and 3 contains fact less fact tables. That is, a record in the fact table indicates that an event occurred, but does not provide any measure of that event. One of the things that web designers like to know is how long a person stayed on a page. The time spent viewing a page is called dwell time and can be used as a measure of how useful the information on the page was to the visitor.

Specifically, the longer the visitor spent on the page, the more valuable the information. This can also be used in determining the effectiveness of advertising. The longer a visitor stays on a page, the more likely it is that they will see the advertisements on that page. Dwell time can be computed by sorting the web log by IP address, date and time. This new sequence will show the log records by visitor. Consider the three log records shown in Figure 4.

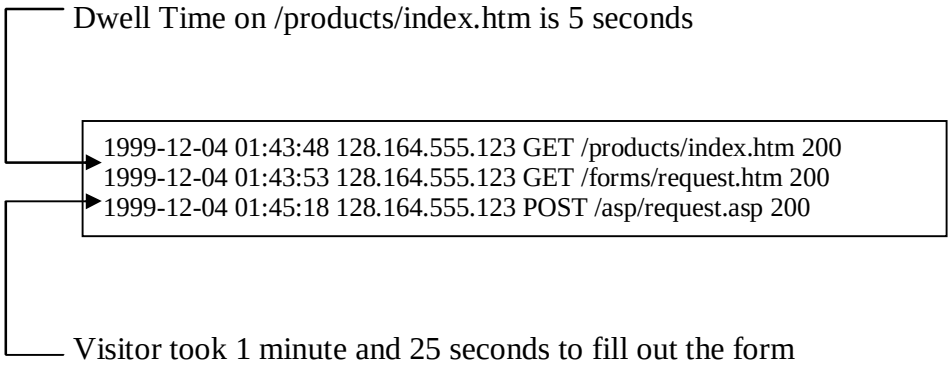

Figure 4: Computing Dwell Time

From just these three log records we can determine that the visitor came directly to the products page, which means that they book marked that page or came to it from another site. They found what they were looking for in 5 seconds and went to a form to order the product. It took a minute and twenty-five seconds to fill out the form and the visitor did not look at any further pages. We can compute dwell time for each page by subtracting the time on one log record from the time on the same visitors next log record. This computation is not foolproof, of course. You cannot tell how long the visitor lingered on the last page they visited. Nor can you be sure that the user did not get interrupted with a telephone call while they were looking at the page. In a shared lab, one person may visit a few pages and then leave. The next person who sits down might very well have the same IP address and appear to be the same user. Hence, it is prudent to establish a cutoff threshold. For example, if there is more than a few minutes between hits, it may be a good idea to treat it as a new visitor unless the page is a form or has a lot of text in which cases the threshold should be raised.

It takes a little more processing, in data staging, to compute dwell time, but now we have a dimensional model with a worthwhile measure rather than a fact less fact table. The addition of this fact allows us to do analysis to determine what factors affect dwell time. If dwell time is a function of the visitor, date or time, then there is little that can be done by the web developers to increase it. However, expansion of the page dimension allows the analyst to determine if dwell time is influenced by content, or page design. Figure 5 shows a dimensional model with the dwell time and an expanded page dimension to capture information about the design, content, and navigation scheme used on the page. This model is somewhat focused on determining what factors are likely to increase dwell

time. It is not the only grain that can be derived from the web log. But it is a useful one. It also points out the fact that fact tables are not easily populated directly from transactions. It is more likely that the transaction data will have to be summarized in data staging before it is useful in a dimensional model.

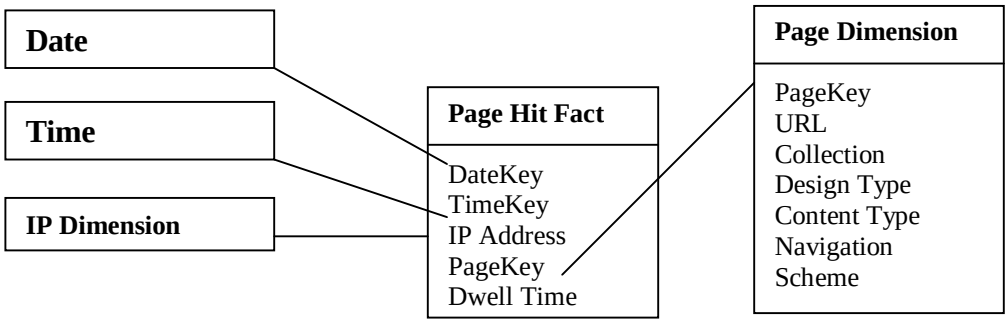

Figure 5: A Fact Table Including Dwell Time

Length Of Visit

Another useful measure of web site effectiveness is length of visit. Referring back again to Figure 4, it is easy to see how measures other than dwell time might be derived. For example, the total time between the first page request and the last page request is a minute and thirty seconds. We don't actually know how long the visitor dwelled on the last page, but this measure will have to suffice. We also know that the visitor viewed three pages, averaging thirty seconds per page. Now we can create a fact table in which we have three measures of visitor behavior – length of visit, pages visited and average dwell time per page. This model is shown in Figure 6.

Let us take a step back and think about what we have done so far. We started with a simple web log record and began to ask some questions that might be answered by analyzing the web log data. Those questions led to a plausible need for a data mart to store information acquired through the web log. But that first attempt led to a fact less fact table and suggested that we should push it further. So we thought of a metric, namely dwell time, that might be useful to determine. That metric then led us to think of another metric, namely visit length, and now we are on our way to a full fledged data warehouse with to metrics tracking the business processes of keeping a visitor on our web site.

Length of visit leads to a new dimensional model as shown in Figure 6. From this new dimensional model we can ask further questions regarding visitor behavior: 1) Does length of visit or pages visited vary by time of day? 2) Are visitors likely to surf longer in the evening or on weekends?, or 3) Do home visitors visit more pages than corporate visitors? If the web site has a well-defined target audience, answers to these questions might help the web site designers develop more effective collections. Notice that the

grain of this model is different. In measuring the length of time on the site we lost the page dimension. At the same time we gained three new metrics: Pages Visited; Length of Visit; and Average Dwell. The Average Dwell is computed by dividing the Length of Visit by the Pages Visited and can be used in conjunction with the dwell time computed in the previous model to ask questions regarding a specific pages such as – Is the dwell time for this page greater than or less than the average dwell time across all pages or across all visits.

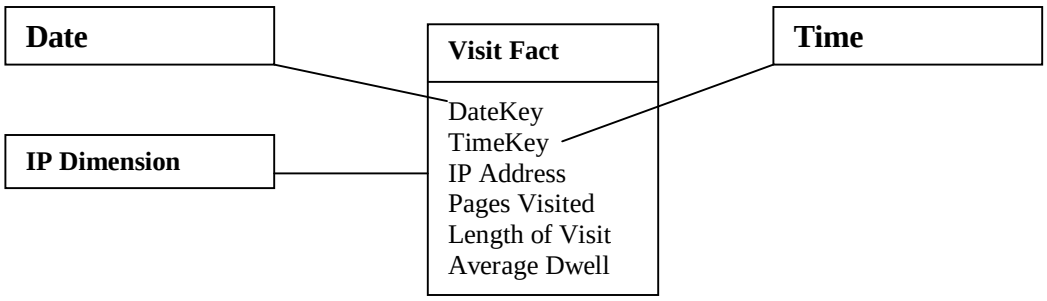

Figure 7: A Dimensional Model for Length of Visit

Conformed Dimensions

Conformed dimensions in a data warehouse, as we discussed in Chapter 6, are dimensions that are shared across dimensional models, providing two benefits. First, construction of dimensions (such as the IP Address dimension) can be costly and time consuming. Using these dimensions across multiple dimensional models spreads out the cost. Second, having shared dimensions allows analysis across fact tables, which may provide even further insight into the processes being analyzed.

If the Date, Time and IP Address dimensions are conformed, the Page Hit Fact table and the Visit Fact table can share these dimensions. The resulting integrated model is show in Figure 7. Now it is possible to answer questions that relate dwell time and pages to length of visit and number of pages visited. Note that the Visit Fact table contains the Average Dwell, but it does not contain the maximum, minimum, or any characteristics of the distribution. For this information the Page Hit Fact table is needed. Further, the Page Dimension tells which collections were visited, making it possible to see if visit length is related to the collection or attributes of the collection.

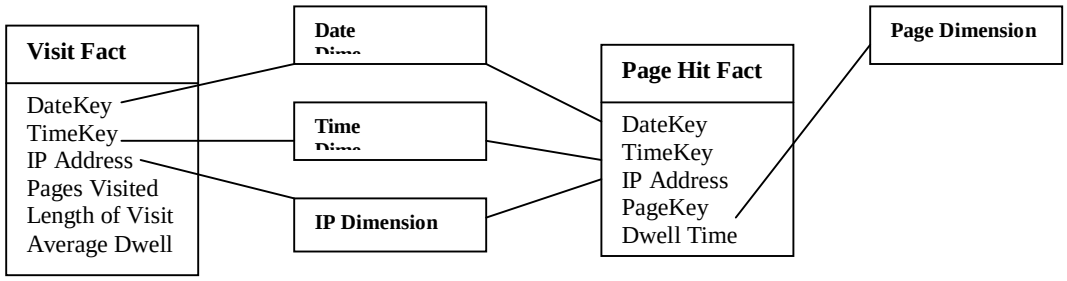

Figure 7: An Integrated Dimensional Model with Conformed Dimensions

Sizing The Web Log Data Warehouse

The Date dimension would have one record for each day of operation for the web site. At twenty-two bytes per record, the Date dimension would only require about 8K per year. The Time Dimension would require only 1440 bytes, so little additional storage is required to add these dimensions. The IP dimension is a little trickier to size since it requires quite a few assumptions. However, assuming 10 million unique visitors, and a 100-byte record, the IP dimension would require 1 billion bytes or 1 gigabyte. If this dimension were snowflaked, the size could be reduced by as much as 90%. However, this storage savings would have to be weighed against the increase in processing time to join the snowflaked tables. The Page dimension would also have a 100-byte record, largely due to the URL contained in the record. A site with 1000 pages would require 100,000 bytes for the Page dimension. A site with 10,000 pages would require 1,000,000 bytes or one megabyte for the page table.

To estimate the Visit Fact table and the Page Hit fact table, we will assume 100 hits per minute or 144,000 hits per day. Assuming the average visitor views ten pages, this becomes 14,400 visitors per day. The Visit Fact table would require 36 bytes per record. Assuming 14,400 visits per day, the Visit Fact table would grow at a rate of 518,400 bytes per day or roughly fifteen megabytes per month. Yet, the largest table would be the Page Hit fact table. The record size would be 32 bytes and with 144,00 hits per day, it would grow at a rate of 4.6 megabytes per day or 138 megabytes per month. At the end of one year the Visit Fact table would require 180 megabytes and the Page Hit Fact table would require 1.6 gigabytes.

Data Mining The Web Log

In Chapter 10 we will see some of the differences between OLAP and data mining. However, as a precursor to that we can see how data mining would provide a different analysis of the web log than the data warehouse that we have just designed. OLAP techniques largely address summarizing the measures in the fact table in order to

determine which attributes of which dimensions influence those measures. Other techniques such as data mining, allow the analyst to look for hidden relationships in the data. A data mining technique called visualization, presents voluminous quantitative data in a visual format making it easier to visually identify trends. Consider a storyboard that is visually encoded with darker shades of gray to depict heavier traffic such as the one shown in Figure 8. From this visualization we can make several observations regarding the traffic to the web site. First, we can easily see that Collection D is the most heavily visited collection. Second, Collection D is darker than the Home Page, which means that it has more visitors than the home page. If visitors were going through the Home Page to get to Collection D, then the Home Page would have at least as many visitors as Collection D. Hence, visitors must be arriving at Collection D via a book mark or from another site. If we have news, sales or other time critical information on the Home Page, it might make sense to move it to Collection D to make sure that visitors see it. Collections A and E are not very heavily visited, suggesting that the current set of top level categories should be reexamined. Collection E has very light traffic, raising questions regarding its value on the web site. Finally, Collection E has a sub page that is more heavily visited, again suggesting bookmarks or links from other pages. It may make sense to move this page directly off the home page. However, since we know that most visitors arrive via links, we may want to put a page at the current address that would automatically move visitors to the new address. The benefit of visualization techniques such as this one is that analysts can easily conceptualize traffic patterns without having to absorb large volumes of detailed data.

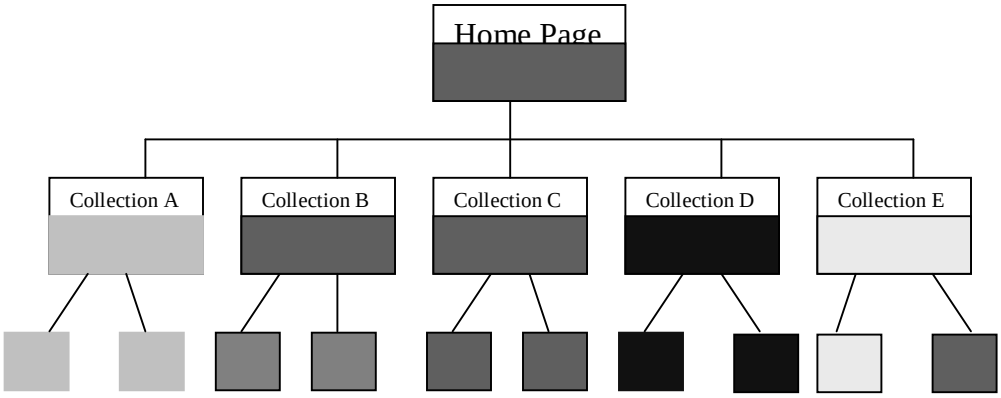

Figure 8: A Visualization of Traffic to the Web Site

A Marketing Research Data Warehouse

So far the dimensional models have focused on the behavior of web site visitors in an attempt to determine which factors make them dwell longer on a page or extend their

visit to the web site. With some simple enhancements it is possible to extend the existing dimensional models into a full-fledged marketing research data warehouse.

Many sites require their customers to log into the site. Prior to activating the login id, the visitor is asked to respond to a simple online survey form that provides some basic demographic information. This goes beyond the web log, but allows even more sophisticated analysis of visitor behavior. The program that processes the form can record the IP address, date and time along with the demographic information. With this information, the IP Address dimension from the previous model can be extended into a Customer dimension. If the site is designed in such a way that pages viewed can be translated into products viewed, then the Page dimension can be extended into a Product dimension allowing analysis of Customers viewing Products rather than Visitors viewing Pages. In fact, if sales are accepted online, sales events can be recorded in a sales fact table even further extending the scope of the analysis into a full market research application. Not only would it be possible to analyze customer behavior with respect to products, but it would also be possible to tie that behavior to web shopping behavior and ask questions like - Are visitors more likely to buy at certain times of the day such as lunchtime or certain days such as the first of the month or on week ends? Figure 9 shows the integrated dimensional models with two fact tables and conformed dimensions.

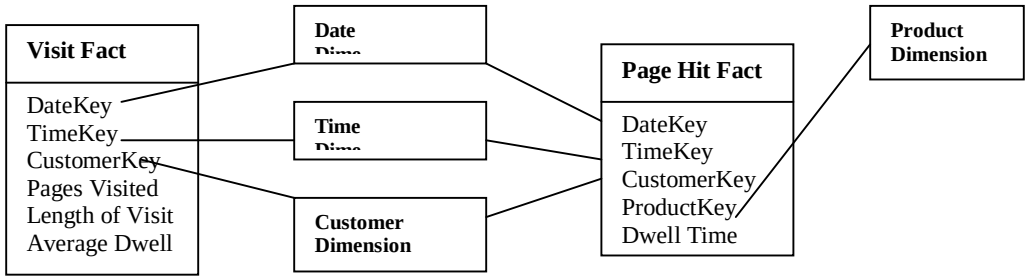

Figure 9: Integrated Market Research Dimensional
Model with Conformed Dimensions

The example presented here started with a dimensional model of the web log and evolved that model into a market research data warehouse. So as not to get too focused on the current models it is important to remember that the purpose of a data warehouse is to provide quantitative time varying information about the effectiveness of business processes. Consequently, a wide variety of other models could have just as well been evolved to analyze a wide variety of other activities including advertising, information dissemination, security, software distribution and the like. If the web site is an Intranet then customers become employees and the questions center around how well the web site meets the needs of its users. The possible models are as varied as the possible purposes of a web site. What is important to remember is that the web server generates a large

volume of time varying data and that dimensional models within data warehousing technology are the most effective means of exploiting that data.

Using The Web To Get To The Data Warehouse

Although the focus of this chapter was to show how data warehousing technology can be used to exploit the large volumes of data that are generated by a web site, it is important to remember that these technologies are truly synergistic. That is, web technologies can be used to advantage in data warehousing applications. A data warehouse that is not fully exploited does not return its full potential. Web technologies help make the data warehouse more accessible thus increasing its value.

The web can be used in a variety of ways to make the data warehouse more accessible. At a minimum, instructions for accessing the warehouse can be put on a company's Intranet along with tutorials and documentation of the available data. Similarly summary data sets can be published allowing users to download information into a spreadsheet or local database. In more advanced applications, server scripts can be written to extract customized summary sets from the data warehouse for further analysis on client machines. Finally, web friendly OLAP tools would allow anyone with proper permissions to access the data warehouse interactively extracting custom data sets from the warehouse and analyzing them using browser plug ins.

Initially web technologies and data warehousing technologies seem to be at opposite ends of the technology spectrum. Web technologies are unstructured, text oriented, and evolving more in technology than in theory. Data warehousing technology on the other hand is highly structured, measure oriented, and evolving more in theory than in technology. But these two diverse technologies are highly synergistic. The vast amount of data generated by a website provides the data needed to build a robust data warehouse. And the analytical technology of data warehousing provides the modeling and theoretical techniques needed to get insight into web visitor behavior.

Summary

This chapter began with a simple data set, called the web log, which is generated by a we server based on requests for web pages from a web site. The purpose was to provide a bottom up example to show how one can begin with a data set, start asking questions and evolve the data set into a data warehouse. First, the web log record was redefined as a fact less fact table. Then two metrics – dwell time and visit length – were defined and the fact less fact table evolved into a measure of the time a visitor spend on a page or at the site. Finally, by making the data collection a little more sophisticated it was shown that a full blown market research data warehouse could be developed.

Fundamentals of Data Warehouse Design

Chapter 9: Case Study – Customer Satisfaction

The more measurable business processes that are identified within an organization the more effective it can become at meeting its objectives in those application areas. One can imagine a theoretical end state whereby every significant process within the firm is modeled using a dimensional model and the aggregate of all those models represents a scientific model of that specific individual firm. But that day is still quite a way off. In the meantime, we can take on business processes one at a time as we recognize their need for improvement. The biggest challenge, of course, is to begin thinking about the work of the organization in terms of measurable business processes. And, yet, this is not as difficult as it may seem. One way to make it easier is to make it easier is to consider a collection of examples where data warehousing has been applied to a variety business application areas. In fact, this is exactly what Ralph Kimball did in his seminal book Data Warehouse Toolkit.

We have already seen two cases involving the specific processes of filling the seats in a movie theatre through promotions and a more generic example of using a dimensional model to track the effectiveness of a website. In this chapter we will take on customer satisfaction.

Some Conceptual Problems with Customer Satisfaction

Mark Twain once said that everybody talks about the weather but nobody ever does anything about it. It is an extreme case of something that is always on everybody's mind because it may affect them in significant ways but we often feel helpless because there is really little that we can do about it. Customer satisfaction is in some ways like the weather. It is something that every firm cares about, but what one can do about it is often elusive. It is for this reason that customer satisfaction makes a good case study.

But, there are also a number of other good reasons for looking at customer satisfaction. First, many data warehousing applications can use money in some way as a measure. Money is a fairly concrete measure and fairly easy to grasp. If we earn more money that is good. If we save more money that is good too. We accumulate more money either way and accumulated money is good because it can be converted into other things that more people believe are good such as higher salaries, more cash reserves, more money for research and development or increased stockholder wealth. But what do we accumulate when we increase customer satisfaction?

Let's say that we set up a process to improve customer satisfaction and we measure customer satisfaction by having customers fill out a form in which they indicate their level of satisfaction in interacting with the company. Let's say, for the sake of argument that the scale ranges from 0 to 10 where 0 indicates a truly horrible experience and 10 indicates a truly wonderful experience. Further, let us assume that the average for

customer satisfaction was 6.4 in a base line study and through a variety of policy implementations it was raised to 7.8 within three months. This sounds like an impressive accomplishment. We now have 1.4 more units of good feeling toward the company. But what does that mean? 1.4 units of increased good feeling is certainly not twice as good as 0.7 units of increased good feeling as we saw in the chapter on data theory because this is an interval variable not a ration variable. If we spent $140,000 to raise customer satisfaction from 6.4 to 7.8 we could not readily say that it costs $10,000 per 0.1 units of increased good feeling.

And why, exactly, are we trying to increase that good feeling? Do we believe somehow that more satisfied customers will make the company more profitable? If so, why don't we measure sales, or revenues or profits. Perhaps we believe that happier customers will make customer service a less stressful job thus reducing turnover and salary expenses. Perhaps it would be possible to correlate increased customer satisfaction with increased profits and then use profit as a measure of improved customer satisfaction. But profits increase or decrease due to a number of factors only one of which is customer satisfaction. So for now we will suspend these philosophical concerns and assume that increased customer satisfaction is a good thing and consequently thus process of producing customer satisfaction is worthy of improvement. Next we turn to the concept of customer satisfaction and try to nail it down a little more.

What is Customer Satisfaction?

To begin we need to ask the question – what do we mean by customer satisfaction? Clearly every company wants its customers to be satisfied. But what that means in an operational sense can vary greatly. One way to measure customer satisfaction would be to send out a survey through which the customer would answer questions which would indicate their level of satisfaction. But an entirely different way to look at customer satisfaction would be to count complaints or product returns. Clearly, if a customer returns a product they were not satisfied with it. And if a customer complains they were not satisfied with something. We could also look at repeat business. If a customer buys a product from a company and then makes another purchase within a short amount of time it would suggest that they were satisfied with the previous purchase.

Now consider the following numerical measures of customer satisfaction:: 1) Reported satisfaction level; 2) Number of complaints; 3) Number of returns; 4) Number of repeat purchases per time unit; or 5) Length of time until repeat purchase

These are far from the only ways in which customer satisfaction can be measured. But they give us enough examples to work with in order to explore the strengths and weakness of each. The main problem with reported customer satisfaction is that people may say one thing and do another. They may report that they are satisfied and never come back for another purchase. They may report that they are very dissatisfied and keep returning for additional purchases. There are many reasons for this. Some people do not want to admit negative experiences. Others will complain no matter what. Some people

may feel one way when they are filling out the survey and have an entirely different impression when it is time to make another purchase. In addition, some people will take the time to fill out satisfaction surveys whereas others will not. They group who does actually fill out the surveys may or may not represent the full set of customers. This is not to say that reported customer satisfaction is not a reliable measure. It is merely to show that it is far from perfect.

Counting the number of complains has some benefits over reported satisfaction in that we know exactly what we are measuring. If we normally receive a thousand complaints per month and we do something to reduce that number to eight hundred per month then we can understand the improvement that we have made. However, if we do something that increases customer satisfaction from 7.8 to 8.1 it is not clear what was improved. Measurement of satisfaction is a subjective assessment of a poorly understood phenomenon. Measurement of the number of complaints is an objective assessment of a more clearly understood phenomenon. And, yet, counting the number of complaints is not perfect either. For example, if a customer is so fed up with a product or service they may not complain. Instead, they may simply take their business elsewhere. So if the product or service gets bad enough, complaints may decrease while sales are plummeting.

Counting the number of returns has similar benefits to counting the number of complaints. If somebody returns a product you can assume that in most cases they were dissatisfied with it. Further, a return is a specific action. It can be objectively counted and the phenomenon being measured is well understood. Or is it? In order to look into the phenomenon of returns a little more deeply we need to ask why somebody would return a product. It may be that the product is simply not what they expected. In this case it may be a problem with advertising or product quality. If the product was purchased online it could be that the product was returned because it took too long to be delivered and the customer purchased the product elsewhere in the meantime. In this case the delivery process is at fault. But there could be other reasons. The customer may have returned the product realizing after they ordered it that they could not afford it or did not really want it. So product returns can represent a wide variety of situations. When a product is returned we ask the customer why they are returning the product and then use those answers to get to the bottom of the problem. But if we do that we are assuming that returns are a problem and we want to get to the bottom of the problem. So we might take this approach if we believe that product returns are a problem for our company and we need to set up a data mart to monitor the process of keeping product returns below a certain acceptable level. But what if the number of returns are not a problem? What if we assume that a certain percentage of returns are merely a cost of doing business and keeping the customers happy and the thing we really care about is customer retention. In that case we might want to monitor customer retention measures such as the number of repeat purchases per unit of time or the length of time until a repeat purchase.

While there is nothing inherently wrong with any of these measures, there is nothing inherently right with any of them either. If we think thing that customer complaints are a problem then we must believe that we are doing something to cause those complaints. And that leads us to the next problem in data warehouse design. We not only need to

have a measurement of a process, we need dimensions of that measurement with at least one dimension that we can manipulate to improve the process. So let's consider the factors that may affect customer complaints.

There is always a time dimension in a data warehouse because the data is inherently temporal. So we will consider the number of complaints per unit of time. For the moment we do not need to specify whether that unit of time is one day, one week or whatever. We can come back to that later. Another dimension of the complaint would likely be product or service. Presumably the customer is complaining about something and that something is something the customer purchased. Notice how this last assumption automatically limits the scope of the dimensional model. It is conceivable that the customer may be complaining about a advertisement that they saw on the television. They may be complaining about an environmental policy or hiring practice. But, in order to measure and improvement something we necessarily have to exclude other things. So for purposes of this example we are only considering customer complains regarding products or services. The next dimension is customer. That is the source of the complain. They question we need to ask ourselves here is whether we want to record complains by individual customer or by some category of customer.

Counting complains by individual customer does not make sense for two reasons. First, it is unlikely that we will get multiple complains from an individual customer. Thus, it would be difficult to track this measurement over time. Second, we will have to develop policies to reduce customer complaints and policies should not be made for individual customers. So we need to have categories of customers instead. The question is – what categories are appropriate – and this leads us into a whole new vein of analysis. We could use standard demographics or we could create categories of our own. Whatever categories are determined should group customers that behave in a similar manner are most likely to respond similarly to our initiatives to reduce their complaints.

We could also consider the nature of the complaint. A complaint about a misleading advertisement is different from a complain about product quality. So a dimension that keeps track of the attributes of the complain will allow us to tell if we are reducing one kind of complain while not affecting another. So, thus far we have four potential dimensions: time, product, customer, and complaint. These are all mediators and we need at least one motivating dimension. What can we do to reduce complaints? If there is nothing that we can do then there is no point in building a data warehouse. The data in the warehouse will simply tell us how many complaints we have as we sit by idly unable to do anything about them. So we need some sort of portfolio of complain reduction strategies. We may work on our advertisements. We may have training classes for customer service representatives. We may pursue product improvement initiatives. As we kick off these new efforts we need to assign each complaint that comes in to the strategy that was implemented to reduce that complaint and that gives us our final and motivating dimension.

The same is true of returns. If we believe that returns are a problem we could develop a dimensional model designed to monitor returns and, ultimately, reduce returns. But let us

pause for a moment and consider the fact that we could easily reduce returns by making it a company policy that we do not accept returns under any circumstances. We could reduce the number of complaints by making it difficult to get through to the complaint desk on the telephone and by not reinforcing complainants by responding to them. However, both of these actions would likely lead to a loss of customers so just reducing complaints and returns may not be the thing we want to do. Perhaps allowing returns and addressing complaints is merely a part of doing business and what we are really interested in is having satisfied customers. It is important to note that reducing complaints or reducing the number of returns may very well be the think we want to do. The doubts about these objectives are being raised for two reasons. First, we need to see that it is not always that obvious what we are really trying to do. Sometimes we see dimensional models that are worked out to solve a particular problem and assume that these models will solve the problem that we have. This is simply not the case. Although there are important commonalities across companies there are also important unique problems. Before jumping to the solution we need to understand what the problem is. If shoddy workmanship or misleading advertising is causing a lot of returns then perhaps we do need to focus on returns. However, in the case we are discussing we will dismiss returns and complaints as not being representative concerns and focus directly on customer satisfaction.

All of this is simple to make the point (or perhaps belabor it) that while satisfied customers is certainly what every firm wants, the difficulty of defining customer satisfaction in a meaningful and measurable way is quite difficult. So for the purposes of this case we are going to make some fairly significant assumptions in order to design a customer satisfaction data mart. And those assumptions will not apply to every case so any given customer satisfaction dart mart may look very different from the one presented here.

→

The Customer Satisfaction Data Mart

The World Class Wine Company sees itself as a purchaser of fine wine rather than a vendor of fine wine. That is, they have an established customer base with fairly well known preferences for wine in terms of price, value and variety. So WCWC looks for opportunities to purchase wine that will maximize the satisfaction of its customers.

More Conceptual Flaws in Customer Satisfaction

Is the customer the best judge?

Do you really know your customer?

Gerson, R. (1993) Measuring Customer Satisfaction: A Guide to Managing Quality Customer Service. Thomson.

Hayes, B. (1992) Measuring Customer Satisfaction: Development and Use of Questionnaires. ASQC Quality Press.

Fundamentals of Data Warehouse Design

Chapter 10: Problems with Multidimensional Data: Up To and Beyond OLAP

One of the questions that frequently arises in discussing data warehousing is – why do we need new software products to manage the data warehouse? Why can't we just use the relational database software we currently have? After all databases such as MS Access and MS SQL Server provide some fairly sophisticated data management capabilities while spreadsheets such as MS Excel provide some pretty fancy graphing capabilities. In addition we have recently seen the inclusion of support for pivot tables in both of these product. Why is something further needed?

A similar question could be asked about relational databases. After all if one has some data in a text file, a fairly straight forward Visual Basic program could be written to summarize the data or provide an answer to any specific query. Of course, as the queries grew more numerous and the dataset grew more complex it would be easy to see how this approach would not work. Then add the problems of data independence and multi-user update management and it is easy to see, before going too far, why the Visual Basis text file solution is not really a serious solution to data management. However, if one is naïve with respect to the demands of data management applications, the Visual Basic alternative may appear viable.

In this chapter we will see why relational databases and spreadsheets are not viable solutions for data warehousing problems. This will be done through a series of increasingly difficult examples each of which shows the limitations of current technology. Even products such as MS SQL Server Analysis Services which provides capabilities far beyond those of a relational database is not a complete solution to the problem. So the chapter will conclude with some ideas for advancing products to provide even greater support for management and exploitation of multidimensional data.

Using an MS Excel Spreadsheet for Multidimensional Data

Imagine a spreadsheet with some data from the theatre example.

Showing	Theatre	Movie	Promotion	Tickets Sold
1	1	42048	25	15
2	1	17729	113	17
3	1	42048	25	21
4	1	66432	25	18
5	1	17408	22	13

Figure 1 – Multidimensional Theatre Data

This small amount of data would not be useful for any kind of analysis but serves to illustrate several important points. The first of which is that this data does not mean

anything to anyone who is not intimately familiar with the codes used for the items represented in the spreadsheet. We will return to this issue later. If we wanted to compute the tickets sold by promotion in MS Excel, we would have to select the entire dataset, sort it by promotion, create a row for the summary value at each break in the data, and use the @sum function to compute the total tickets sold for that promotion. Then, if we wanted to compute a similar total by movie we would have to go back and remove our summary rows, resort the data by movie and go through the prescribed procedure again. There are some shortcuts that can be taken, but in general, summarizing multidimensional data in a spreadsheet is awkward.

Next consider the problems of complexity and volume. A typical spreadsheet might hold an average of around a dozen columns before the data begins to scroll off the page. A dimensional model with one fact and four dimensions will need a minimum of five columns to represent the smallest amount of dimensional data. If we add just one attribute from each dimension and one additional fact from the fact table we are already up to ten columns and approaching the complexity limits of the spreadsheet. So the more complex a dataset the more awkward the analysis. But complexity grows by tens of columns at most whereas volume grows by hundreds or thousands of rows. Suppose the above sample had a thousand rows instead of five. The manipulations described above would require scrolling down through a thousand rows to select the data to be sorted, inserting possibly dozens of summary rows, computing those summary values and removing those summary rows in order to do the next summarization. With a couple dozen columns and a thousand rows this has become an difficult job. If the number of rows increases to the tens of thousands, the job of analysis becomes unwieldy. The limits in MS Excel spreadsheet is 65,536 rows by 256 columns. While this is far greater capacity than the average analyst could handle, it is not nearly enough capacity for a data mart. In fact, it is not even in the ballpark.

To be fair, spreadsheets do have some nice features such as charting capabilities and pivot tables which facilitate the analysis of multidimensional data. And some might argue that the manipulations described above could be automated with VBA so the analyst would not have to do these manipulations by hand. These are fair observations. But, despite the fact that spreadsheets have some nice features they are not an adequate vehicle for analyzing high volume complex multidimensional data. So lets consider loading our data in a relational database such as MS Access and see if we fair any better.

Using an MS Access Database for Multidimensional Data

Next assume that the data in Figure 1 is in a table in MS Access. Does this make things any easier? Well to some degree it does. If we wish to determine the tickets sold by promotion we could use the SQL *Group By* statement and easily compute those values. Assuming that the table is named TicketSales, the SQL would be as follows.

```
Select Promotion, Sum(TicketsSold)
From TicketSales
```

Group By Promotion

This is much easier but we are not out of the woods yet. We still have a number of problems to consider. First, what if we wanted to have names in our summary data instead of codes? Second, what if we wanted to see the data in graphical form. Third, what if we were doing some exploratory analysis and needed to continually look at different slices of the data. And fourth, what do we do if we had a very large volume of data. Each question will be considered in turn.

What if we wanted to have names in the summary data instead of codes? Well, presumably, in a relational database, we have the data in a dimensional model where the facts are in one table as shown in Figure 1 while the dimensions are in dimension tables. We will assume that the dimension tables are named after the columns. Then we could convert the table in Figure 1 to a table with names in it by joining the fact table to the dimension tables. The SQL for look something like the following:

```
Select Showing.Name, Theatre.Name, Movie.Name, Promotion.Name,
TicketSales.TicketsSold From TicketsSold
Where Showing.Id = TicketsSold.Showing
And Theatre.Id = TicketsSold.Theatre
And Movie.Id = TicketSoldMovie
And Promotion.Id = TicketsSold.Promotion
```

If we wanted to summarize TicketSales by Promotion using promotion name we could use the following SQL:

```
Select Promotion.Name, Sum(TicketsSold)
From TicketSales, Promotion
Where TicketSales.Promotion = Promotion.Id
Group By Promotion.Name
```

Now, in both cases, the SQL has gotten a little more complicated, and we can pause for an important point. There isn't anything that you can do with SQL and a relational database that you could not do with Visual Basic and a flat file. It would just take a great deal longer using Visual Basic and a flat file. In some cases it would take so long that you wouldn't even consider certain queries. In addition, the likelihood of introducing errors in the summarization VB programs are fairly large. So while it is true that you cold do anything with VB and a flat file that you could do with SQL it is not practical. We can make the same point comparing OLAP tools with SQL. There isn't really anything you can do in an OLAP tool that you could not do with a relational database (sometimes you may also need a spreadsheet). However, it would be awkward, time consuming and error prone. So OLAP just makes it a little *easier* to analyze multidimensional data. And the issue of a spreadsheet brings us to the second point from above. What if you wanted to see the data in a graphical form.

Data visualization is an important component of data analysis. Sometimes we can see patterns in visual data that we cannot see in numerical data. This will be explored more fully in Chapter 11 on data mining. However, for now, we ask the question – what if we want to see the data in a graphical format? One way we can handle this is to export a dataset into a spreadsheet and use the charting capabilities of the spreadsheet. This seems like a reasonable solution as long as we only want to chart a single dataset. But if we want to do some kind of iterative or exploratory analysis this process would be cumbersome. First we would have to write a query to create the slice of data that we wanted to chart. Then we would have to export the resulting dataset into a spreadsheet. Finally, we would have to chart the data in the spreadsheet. For one dataset this might now be such a chore. But to do it over and over again would be a time consuming challenge.

The vary nature of multidimensional data is that it is analyzed by summarizing along one dimension and then another. When summarization along one dimension proves fruitful we may then want to look at that dimension in greater detail. For example, if the summary of sales by promotion shown above yielded something interesting we might wish to 'drill down' and see how promotions fared by theatre. The SQL to do this drill down is

```
Select Promotion.Name, Theatre.Name Sum(TicketsSold)
From TicketSales, Promotion
Where TicketSales.Promotion = Promotion.Id
Group By Promotion.Name, Theatre.Name
```

If that drill down was not revealing (that is promotion success did not vary by theatre) then we might try drilling down by showing. The SQL for that drill down is

```
Select Promotion.Name,  Showing.Id, Sum(TicketsSold)
From TicketSales, Promotion
Where TicketSales.Promotion = Promotion.Id
Group By Promotion.Name, Showing.Id
```

As we can see from these two drill down queries, this interactive analysis that shifts back and forth between levels of detail is difficult using a traditional relational database and that is why OLAP tools are necessary. But before we get ahead of ourselves we need to look at an intermediate development on the road to OLAP tools - the pivot table.

Crosstab Queries

A pivot table is a tool available both in MS Excel and MS Access that provides a limited capability for multidimensional analysis. In this way, it is the forerunner of OLAP tools and understanding pivot tables brings us one step closer to understanding OLAP tools. However, in order to understand pivot tables we need to take one step further back and understand the predecessor to pivots tables – the cross tab query.

A cross tab query is different from a tradition SQL query in some important ways that show a significant divergence from the traditional relational database model. Consider the table in Figure 1 again. This is a valid relational table representing the showing of a movie in a specific theatre at a specific time under the influence of a specific promotion yielding a specific audience size. If we were interested in summarizing by promotion and movie we could use the following SQL query

```
Select TheatreData.Promotion, TheatreData.Movie, Sum (TicketsSold)
From TheatreData
Group By TheatreData.Promotion, TheatreData.Movie
```

This would yield the table in Figure 2.

Promotion	Movie	Tickets Sold
22	17408	13
25	42048	36
25	66432	18
113	17729	17

Figure 2 Ticket Sales Summarized by Promotion and Movie

This is still a valid relational table and can be used in an SQL query to produce yet another valid relational table. However, the data is not in a form that makes it easy to find specific combinations of promotion and movie. Nor is it in a form that makes it easy to see trends in the data. A better form for this data would be to put it in a two dimensional matrix with promotions down the size and movies across the top. And an extension to SQL called a Crosstab query will allow us to do exactly that as the table in Figure 3 shows.

TicketSales_Crosstab					
Promotion	**Total Of TicketsSold**	**17408**	**17729**	**42048**	**66432**
22	13	13			
25	54			36	18
113	17		17		

Figure 3: Ticket Sales Cross Tab

Here we see a two dimensional matrix with Promotions down the side and Movies across the top. This is a handy way to summarize information and it is possible to construct this matrix fairly easily via the Cross Tab wizard in MS Access or, as was just mentioned,

through an extension to SQL. For example, the SQL to create the Cross Tab in Figure 3 is as follows:

```
TRANSFORM Sum(TicketSales.TicketsSold) AS SumOfTicketsSold
SELECT TicketSales.Promotion, Sum(TicketSales.TicketsSold) AS [Total Of
TicketsSold]
FROM TicketSales
GROUP BY TicketSales.Promotion
PIVOT TicketSales.Movie;
```

This benign and helpful query has significance far beyond the data summarized in the query. First, this table represents a significant violation of relational database theory. In a relational database, any well formed relational table operated on by valid relational operators should produce another well formed relational table. This closure property of relational operators is important because it means that the results of one query can be used in another query producing valid results about the universe of discourse. In simpler terms we could say that if you create a view, in a relational database, and that view is created using valid relational operators, then that view can be treated as though it were another table and any queries that include that view will produce valid results.

But the table shown above cannot possibly be a valid relational table. In a valid relational table, the table itself represents a class of entities. The rows in the table represent instances of those entities and the columns represent facts about those instances. What entity does the table in Figure 3 represent? The answer is that it does not represent an entity. Instead it represents a summary set of data derived from a valid relational table. This may seem like a picky point but the analytical power of a relational database relies heavily on the closure property. So why would we want to compromise (or some would say further compromise) SQL and allow invalid tables? And the answer to that gives us some insight into the transition from relational database to data warehouses and highlights some of the important differences.

A Cross Tab query is a two dimensional matrix containing summary information derived from the underlying tables. In this sense we can see it as the first data cube (or at least a two dimensional matrix). And in it we see the transition from data tables (relational database representation) to data cubes (OLAP representation). These early non relational queries recognized the limitations of relational databases for browsing summary information. Before long they gave way to pivot tables and pivot charts. Pivot tables allowed for multidimensional cross tab queries that could be manipulated through a graphical user interface rather than by iteratively writing SQL queries and Pivot Charts provided the same capabilities in a graphical fashion. With Pivot Tables and Pivot Charts we see the first software tools for manipulating multidimensional data. So why do we need OLAP? Well, we are getting ahead of ourselves once again. We need to take a breath and look at Pivot Tables a little more closely.

Pivot Tables

Going back to the table displayed in Figure 1 and the subsequent SQL examples using that table we can see that it is possible to summarize that data with respect to any of the attributes by writing the appropriate SQL Group By statement. Similarly, we could summarize the data with respect to any two attributes with a similar SQL Group By statement. However, as we have seen, reading a table of data summarized on two attributes is a little difficult whereas the Cross Tab query presents this same data in a more convenient form. In Figure 3 we saw a Cross Tab query that summarized the data by Movie and Promotion. What if we wanted to see the data summarized by Showing and Promotion? Well, we could go back to the Cross Tab wizard and create a new Cross Tab queries. Or if we were a little more bold, we could attempt writing the SQL. And this strategy might work if there were only a few ways to roll up the data.

But what if there were four dimensions and each dimension had four attributes. There would be sixteen attributes and the number of two dimensional summaries would be the combinations of sixteen attributes taken two at a time or xxxxx . [The calculation for this is [(16-2!)/2! = xxxxx] A data mart with only sixteen attributes would be a tiny data mart indeed. And yet having to go through the Cross Tab wizard xxxxx times to see the summary data using two attributes would be daunting. If you were bold and were attempting the SQL your chances of writing xxxxx Cross Tab queries without an error are not in your favor. Worse yet, some of those Cross Tab queries could be error free syntactically and provide erroneous information because the chances of confusing attributes over the course of xxxx queries are quite high.

So a Pivot Table provides a graphical user interface for summarizing a multidimensional dataset along selected pairs of attributes without having to use the Cross Tab wizard or write the tricky SQL. Thus the Pivot Table is a blessing of a labor saving device for summarizing multidimensional database. But it utility does not stop there. Suppose we had used a Pivot Table to construct the Cross Tab query in Figure 3. Seeing that promotion number 54 was particularly successful we may wish to see more detail. For example, we may with to summarize the data at a lower level of detail by adding another attribute such as weekday or showing that may provide some insight into the successfulness of promotion 54. The Pivot Table provides a convenient user interface for this drill down operation also. So, in short, a Pivot Table is a user friendly graphical representation of a multidimensional dataset which provides the capability for the user to easily view summarizations of the data along various collections of attributes.

So far we have been discussing the capabilities of MS Access which provides extraordinary easy of use capabilities at the cost of some fairly severe limitations on database size. MS Access databases can support databases up to maximum of a few hundred thousand records. Even if this limit were to raise into the millions of records, it would not be adequate for the volumes of data managed by a data warehouse which could easily be in the billions or even trillions of records. For this volume of data we would need a production quality database such as SQL Server.

Going back to the scenario with Cross Tabs we would need SQL Server support for creating data extracts (as we did with the Cross Tabs) and then further support for analyzing the data (as we had with the Pivot tables). This is exactly what SQL Server Analysis services provides. The extracting capability allows us to define data cubes and the data browser allows us to look at the data cube from a variety of perspectives for analysis. But as we move to a full fledged OLAP product we see further features to support multidimensional data and further capabilities for addressing data volume. The next section will explain some of the key features of SQL Server Analysis Services and show how they represent evolving software support for multidimensional data. The section after that will explain some of the limitations and will provide some suggestions regarding where the software products should go next.

SQL Server Analysis Services

The data cube is a generalization of the pivot table and is used to represent multidimensional data. SQL Server Analysis Services, in a nutshell, provides support for data cubes. It provides capabilities to define conformed dimensions. It supports hierarchical dimensions. It provides the capability to extract a data cubes from appropriately defined and related relational tables. And it provides the capability to browse cubes using drill down and roll up operations. Since SQL Server Analysis Services uses a combination of relational tables and data cubes, it is refereed to as a MOLAP approach to data warehousing. This acronym bears some explaining.

As was mentioned before, in Chapter 2, when the Truth Database concept failed data was divided between two repositories – 1) transaction oriented operational data and 2) summary data collected for the purposes of analysis. To distinguish between there two kinds of data a new acronym was introduced – OLAP. OLAP or Online Analytical Processing was distinguished from OLTP or Online Transaction Processing. Despite the confusion automatically introduced by more acronyms, this distinction did serve to distinguish the transaction oriented databases (OLTP) from the summary data databases (OLAP). But the next question to arise was – how should this summary data be stored? Should it be stored in relational tables or should it be stored as summaries in pivot tables and data cubes?

Abraham Maslow once said, “When the only tool you have is a hammer, everything looks like a nail.” This observation can be easily extended to databases. When the only tool you have is a relational database, everything looks like a collection of tables. And indeed the first data warehouses were little more than relational databases containing tables of summary data. And these databases represented a relational version of OLAP or ROLAP. Advocates of the second approach thought that multidimensional data should be stored as multidimensional data and thus supported the idea of storing data cubes. This notion of Multidimensional OLAP eventually took on the acronym of MOLAP.

There are benefits and draw backs with both of these approaches. If you take the relational approach, the underlying data is kept at the most granular level. Hence, any summary can be derived from the underlying data. The drawback is the time it takes to

produce that summary. A second drawback of the pure relational approach is that the data may be too complex residing in a large number of relational tables. One response to this second problem was to define limiting relational schema such as the star and snowflake schema which would reveal the inherent relationships between facts and dimensions.

The MOLAP approach is based on the premise that relational database are inherently the wrong storage representation for dimensional data. The basis of this argument lies in the fact that the primary operations on dimensional data are drill down and roll up. These operations can be computationally intensive on data if the volume is large and the number of dimensions are many. Data stored in cubes facilitate analysis. On the other hand, cubes are less flexible. Every time data is stored in a summarize fashion, they are other summaries that are no longer possible without going back to the source data.

SQL Server Analysis Services uses a hybrid OLAP approach known as HOLAP. This means that data is stored in relational tables as well as in data cubes. The functionality of Analysis Services provides more advance Pivot Tables by providing support for conformed and hierarchical dimensions as well as some more advanced summarization capabilities. But for all its glory, Analysis Services is little more than a glorified Pivot Table sitting on top of a much more powerful relational database engine. This is no small thing given the capacity limitations of MS Access. Without the power of SQL Server to manage data, Analysis Services would be more like an advanced spreadsheet than a data warehousing tool. And yet we see some capabilities in Analysis Services that point to possible future capabilities with much greater promise. And those we turn to next.

Future Capabilities

Analysis Services provides more than a graphical user interface for manipulating data cubes. In addition, it supports a multidimensional query language called MDX which allows data cube manipulation through SQL like queries. In its current state MDX is somewhat crude but it points to two important future directions for utilizing multidimensional data. The first is that data analysis can be driven by a programming language that issues MDX statements to produce higher more sophisticate analytical capabilities that are easily available through a graphical user interface. For example, consider the simple case described above in which xxxxx possible two dimensional summaries could be derived from the underlying data set. Going through each of these xxxx summaries by hand using a graphical user interface would be very time consuming. Further, with each summary the analyst would have to look at the data to see if there were any interesting results. So this is not only time consuming, it is very error prone. Consider, as an alternative, a program that generates each summary one at a time and scans the summary data for interesting results or patterns. This would completely automate one aspect of the data analysis making it more cost effective and less error prone. So having a multidimensional query language that can be used in conjunction with a host language program expands the possibilities for data analysis by orders of magnitude.

This example, hopefully, points to further possibilities for even more advanced analysis. For example, one can see that in order to do truly sophisticated analysis the data cube may eventually become the unit of analysis. When this happens we will need a more succinct language for specifying the operations on data cubes. One possibility would be some sort of a dimensional algebra. In order to see the potential of this idea we need to back up a little and look to the relational model for guidance.

In a relational database the unit of analysis is the table which represents a set of entity occurrences that form a group or a category of entities. We can operate on that table to derive a new table which represents either a subset of the original set (using the where clause to restrict the query), an expansion of the original sets (using a joint), or a summary of the data in the original set (using a built-in function). If the operation performed on the original table(s) is a valid relational operator then the resulting table should be a valid relational table. This means that the resulting table can be used, once again, in another relational operation to produce yet another valid relational table. Figure 4 shows how this closure property allows increasing more sophisticate analysis of relational data as it is summarized, expanded, summarized and so on through the use of valid relational operators. This closure property of the relational model provides a virtually unlimited extended analysis of relational data.

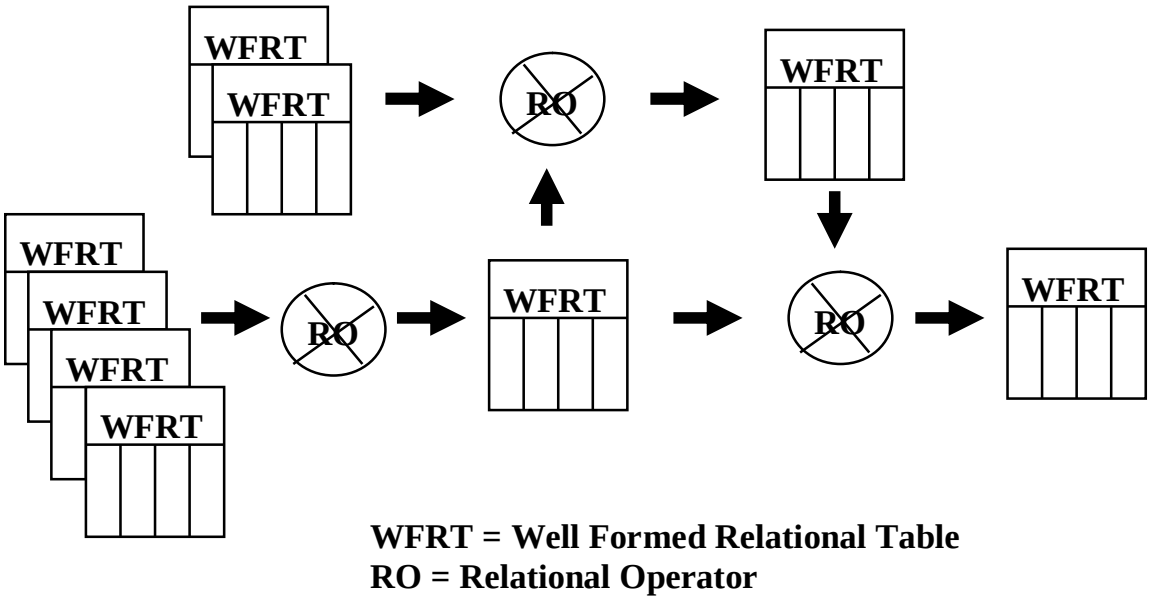

Figure 4: The Closure Property in Data Analysis

Next, consider Figure 5 which assumes a similar closure property for well formed data cubes. We do not currently have a definition for a well formed data cube but that will likely come along as we develop more theory about dimensional modeling. Neither do we, for that matter have a definition of a dimensional algebra operator because that will

have to come along later as the development of theory progresses. Nonetheless, assuming this closure property we can see the power of a dimensional algebra for extending the analytical capabilities for dimensional data. Without pushing one's imagination too far you can see how this development could move the exploitation of dimensional data from query languages to mathematical specifications. And when that happens we will see some amazing advances in what we can do with data.

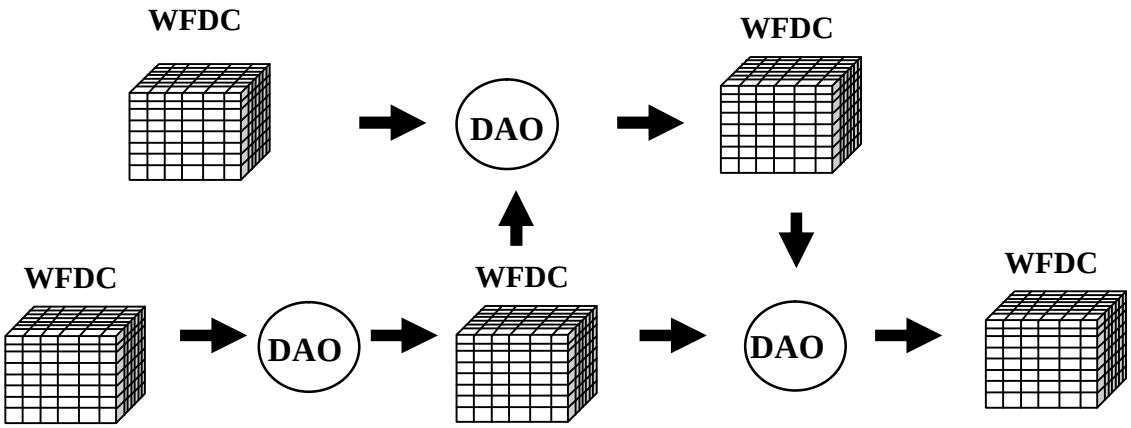

Figure 5: The Closure Property in Multidimensional Data

Summary

This chapter has shown the evolution of OLAP technology from spreadsheets to relational databases revealing how OLAP tools do not allow you to do anything that you could not do before they merely make it easier to achieve a specific kind of data analysis necessary for exploiting multidimensional data.

Fundamentals of Data Warehouse Design

Chapter 11: OLAP vs. Data Mining

Once a data warehouse is created, it can be exploited using OLAP technology or, in some cases, data mining. Ostensibly, one is descriptive and the other is explanatory. The subtleties of this distinction will be explored. Data mining can be predefined or serendipitous. Both approaches will be explained. In addition, there are a wide variety of tools and products on the market today for data warehousing but none of them is either complete or theoretically solid.

First, a Word About Metaphors

Most people know what a metaphor is from a long ago English class. However, Lakoff and Johnson revealed the true extent of metaphors in our daily conversation. According to them, "the essence of metaphor is understanding and experiencing one kind of thing in terms of another." [pg. 5] And this is really no different from what we already knew. However, they also say, "Our ordinary conceptual system, in terms of which we both think and act, is fundamentally metaphorical in nature." [pg 3] This is to say that the concepts that we use to understand the world around is primarily metaphorical in nature and this is an important insight because, if it is true, then understanding how we think about the world becomes a job of understanding our metaphors. They go on to say,

> "The concepts that govern our thoughts are not just matters of the intellect. They also govern our everyday functioning down to the most mundane details. Our concepts structure what we perceive, how we get around in the world, and how we relate to other people. Our conceptual system thus plays a central role in defining our everyday realities. If we are right in suggesting that our conceptual system is largely metaphorical, then the way we think, what we experience, and what we do everyday is very much a matter of metaphor." [pg 3]

A frequently used metaphor that Lakoff and Johnson use to support this assertion is the Time is Money metaphor. Since we do not fully grasp the phenomenon of time we often use this metaphor to express ideas about time. For example, we spend time and save time just like we spend and save money. So in our everyday conversation we use the Time is Money metaphor when talking about our utilization of time. For example, one might 'save time' or 'waste time'. One might 'invest their time' or 'budget their time'. These uses reveal the underlying metaphor and give us some insight into our understanding of the phenomenon of time. However, saved time cannot be put in a bank for later use and we certainly do not receive any interest on saved time. So at some point the metaphor breaks down and the fact that we are using a metaphor rather than a concept that models an underlying reality becomes a little more obvious.

One could easily stretch this notion to say that when we see metaphors used, especially very obvious metaphors, it is a sign that the person using the metaphor does not fully understand the concept about which they are speaking. When a project manager is "riding herd" on a project it is pretty obvious that he or she is using the project as cattle drive metaphor. They may wish to "round up the strays" or "start hoofing it to the next milestone". What is less obvious is that the use of this metaphor also reveals an inadequate understanding of project management because there are also many ways in which a project is very different from a cattle drive. First, people do no like being treated like cattle. They don't have the same herd instinct. And, when the cattle finally arrive at their destination they are slaughtered. They do not turn around and get ready for the next cattle drive. The point here is that the use of metaphors not only reveals our underlying conceptual structure but they also reveal an inadequate or erroneous underlying conceptual structure.

So what does all this have to do with data warehousing and data mining? Well, if you have not made the connection already, both data warehousing and data mining are metaphors. A warehouse is a large, out of the way, structure where things are stored. When one thinks of a warehouse one generally thinks of a place where volume is more important than accessibility. This may have been a well chosen metaphor in the early days of data warehousing when historical data from operational systems was warehoused for future usage. The fact that a metaphor was used reveals the fact that little was understood about this repository for historical data. As we saw in Chapter 3 data warehousing has evolved from a data push approach for which the warehouse metaphor was appropriate to a metric pull approach for which the metaphor is no longer appropriate. We should probably use a more appropriate term than data warehousing such as temporal databases, metric databases or multidimensional databases. However, once established, conventions are difficult to override. An example of one of the most enduring conventions in the history of computers is the word computer itself. The average desktop does far more processing than computing, yet the average person still refers to their desktop as a computer rather than a processor and probably always will. So the chances of establishing a more appropriate name for data warehousing are not promising.

Similarly, data mining is also a metaphor. In a gold mine, for example, precious metal is extracted from the mine by having people go into the mine and attempt to find valuable nuggets of ore. So, using this metaphor, a data miner goes into a data set and extracts useful nuggets of information. But the metaphor breaks down very quickly. First, of all there are a wide variety of data mining techniques that will yield different kinds of information. So if the mining metaphor were completely faithful it would be like going into a mine and bringing out nuggets of gold one time, silver another and copper yet another. Second, the relationships derived from data mining are not facts about the domain. The are potential relationships that need to be verified through further analysis. If this were true of mining operations every gold nugget would have to be further tested to insure that it is indeed gold and not some sort of fool's gold. If data mining approaches were referred to by their names such as association analysis there would be a lot less confusion. But, again, data mining is a catchy term and it has indeed caught on.

This creates further problems when one asks – what is the difference between data warehousing and data mining? Or – how do data warehousing and data mining compliment each other. If you think of a data warehouse as a large repository of historical data extracted from operational systems that you use to learn more about the historical operations of the business then data warehousing and data mining are almost exactly the same thing. However, if you think of the data warehouse as model of measurable business processes and think of data mining as association analysis, then the two have very little in common. Further, yet, if one asks – What is data mining? – you cannot answer – it is association analysis because your answer would be incomplete. You would have to answer – it is association analysis, or decision trees, or cluster analysis, or neural networks, and so on. But what about regression analysis and time series analysis? Both of these techniques were around long before anyone ever heard of data mining. And yet they are both included in the SQL Server 2005 Business Intelligence Studio under data mining. They are both useful technique and it is better to include them on pragmatic grounds rather than to exclude them on semantic grounds. But they reflect the confusion around the question – What is data mining?

Metaphors are useful for selling ideas or for quickly explaining new ideas. But, along the way they create a lot of misunderstandings. So for the sake of more clearly understanding the differences between data warehousing and data mining, we will think of data warehousing as using multidimensional datasets to temporally model measurable business processes. We will think of data mining as an umbrella term used to describe a collection of techniques that are used to extract useful information (usually in the form of patterns of some kind) from a large dataset. Some of those techniques will be discussed in this chapter and compared with SQL and OLAP.

Second, Another Word About Distinctions

We have already seen a number of distinctions drawn between common terms in data warehousing. For example, in Chapter 1 we drew some important distinctions between relational databases and data warehouses. In Chapter 10 we explained the difference between ROLAP, MOLAP and HOLAP. In Chapter 4 there was some discussion about why distinctions are important. Distinctions, while sometimes confusing, ultimately help to clear up some of the confusion created by vague metaphors and the fact that data warehousing may mean different things to different people. In the same spirit we will make some important distinctions in this chapter between SQL, OLAP and data mining. We will then go on to draw further distinctions between different kinds of data mining. These distinctions will, hopefully, clarify the murky concept of data mining, reveal the true nature of data mining and shed some light on when a given approach may be appropriate. Along the way we will also attempt to explain how things got murky in the first place.

SQL vs. OLAP vs. Data Mining

When a phrase become popular, many authors will attempt to take advantage of its popularity by titling their books or articles in such a way as to attract the attention of an audience eager to learn more about a new idea or buzz word. An example of this is a recent book, mercifully left in anonymity, focusing on data mining using SQL. This particular book on data mining with SQL is actually quite a good book that shows how to use SQL to do some fairly sophisticated data analysis on a relational database. However, if it were called Statistical Analysis Using SQL it would be a little more honest. Why quibble about the title of a book? The problem is that it slightly misleads the potential audience. It draws attention away from the true strengths of a fine book. And it further clouds and confuses the meaning of data mining.

Let's go back to basics. An SQL table represents an entity class within a universe of discourse. Rows in the table represent instances of that entity class. Attributes represent facts about those entities. It is entirely possible, although perhaps unlikely, that a perfectly valid relational database contains no numeric attributes. That is to say that all attributes are categorical. So the database itself does not contain measures or counts of anything. It is merely sets of things with certain characteristics.

An SQL query against such a database would identify entities or sets of entities for which certain facts were true. So basic SQL queries would provide instances of certain kinds of data or characterize the underlying data in some way.

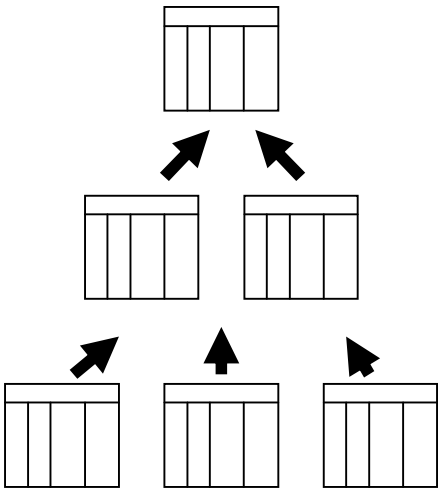

Figure 1: Pictorial Representation of SQL Queries

As was mentioned in Chapter 10, the result of an SQL query is a valid relational table providing facts about a collection of entities. Due to the closure property any table derived from an existing well formed relational tables using valid relational operators is again a well formed relational table. So the results of one query can feed into another query producing higher levels of summary or different combinations of data for analysis.

So even though SQL can be pushed to accomplish a wide variety of data manipulation feats in statistical analysis, it is best to think of it as a tool for analyzing a snapshot model of categorical information. Figure 1 provides a pictorial representation of the essence of SQL.

The focus of OLAP is quite different. Instead of modeling categories we model a key business process with the hope of improving it. The data is temporal and multidimensional. We have multiple dimensions because there are a number of factors that influence the performance of a process. The information is summarized in a data cube which facilitates multidimensional analysis. In the theatre example discussed in Chapter 3 a typical SQL kind of question might be - how many showings do we have per week or what is the average number of showing per theater. A typical OLAP inquiry would be how did a given promotion affect attendance and did it vary by movie? Figure 2 provides a pictorial representation of OLAP with its representative process improve and the associated data cube.

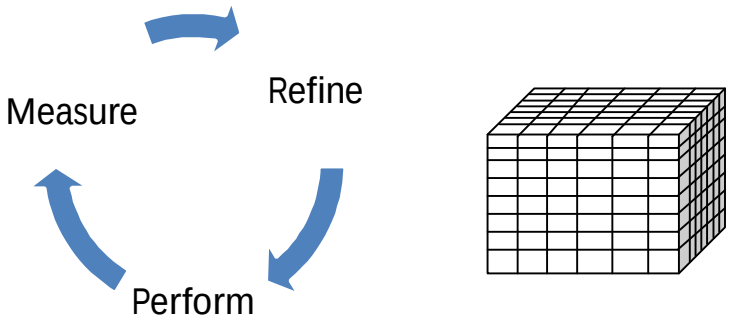

Figure 2: Pictorial Representation of OLAP

Data mining does not specifically address the analysis of categorical information or the improvement of key business processes. Instead, it attempts to reveal relationships between data items that were previously hidden from the people who use the data. Since there are a wide variety of ways in which data items can be related to each other there are a wide variety of data mining techniques. Some of these data mining techniques can operate on data extracted from a relational database or a data warehouse. Other techniques require a fair amount of data manipulation to put the data into the proper structure for data mining. Either way the input to a data mining model is a specially constructed data set and the output is the revelation of new relationships in that data. Whereas a person doing OLAP analysis should have some idea what to expect, the person doing data mining might be completely surprised by the results. Figure 3 provides a pictorial representation that summarizes the role of data mining.

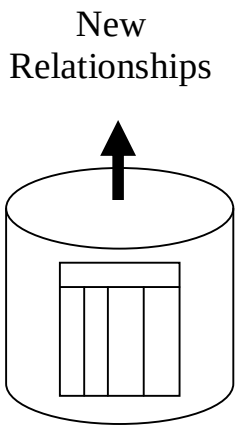

Figure 3: Pictorial Representation of Data Mining

→

Knowledge versus Serendipity

Before diving into the variety of data mining techniques, a point is in order regarding the status of the discoveries from a data mining exercise. Is this knowledge as some may claim? Or is this merely the possibility of knowledge? Data Mining is sometimes referred to as Knowledge Discovery in Databases (KDD). This designation is somewhat unfortunate because data mining does not discover knowledge and claims such as this casts doubt on the credibility of data mining. The essence of knowledge is not in the discovery but in the validation. Earlier treatments of data mining emphasized discovery whereas more recent treatments emphasize the validation processes. So when we discover a pattern in a dataset through data mining that is only the first step. Further analysis must be done to insure that this pattern is not just an artifact of the data set we worked on.

The notion that knowledge can be discovered algorithmically is faulty from both a statistical and philosophical perspective.

According to Plato, in order to be knowledge a thing must be true, you must believe it is true and you must be able to explain why it is true.

Knowledge versus Saving Appearances

Ultimately, any relationship that we discover must also be explained. So, if we discover, for example, that people who buy cabbage and potatoes at the grocery store also buy horseradish sauce, then we must be able to explain why that relationship exists before we can legitimately refer to it as knowledge.

Data Mining Techniques

Since this book is on data warehouse design not data mining only a quick cursory glace will be provided to explain some of the most common data mining techniques. Something to keep in mind while reading over the following quick survey is the diversity of data mining techniques. Someone who is an expect in a given technique may not know a great deal about other techniques and someone who is familiar with most of these techniques is probably not an expert in any of them.

Data Visualization

Data Visualization has been around long before data mining was ever heard of. Early spreadsheets provided rudimentary graphing capabilities and we have known for a long time that data presented in a graphical form is much easier to grasp than numerical data. The challenge in data visualization today and the reason that it is frequently considered a data mining technique is that complex data can some times reveal hidden patterns when the complex data is presented in a visual form.

Association Analysis

The example mentioned above regarding cabbage, potatoes and horseradish is an example of association analysis. This is also frequently referred to as market basket analysis. The question asked in association analysis is – do people who have feature A also have feature B. This feature can be a quality, an attribute or a behavior. The purpose of this analysis is to use the existence of one feature to identify the potential existence of another feature. If we know that shoppers who purchase cabbage are more likely to purchase potatoes we may stock up on potatoes when we are having a sale on cabbage. If we know that people who purchase cabbage also purchase horseradish we may put an end of aisle promotion for a new horseradish spread at the end of the aisle where the cabbage is sold.

Decision Trees

When you go to a new car lot to purchase a new car which factor is the most important to you in making a decision. Should the salesman ask you if you prefer luxury or economy? Or sportiness verses reliability? Is the color important? Is cost a factor? Clearly if the salesman asks you what color you like, he can spend a lot of time showing you red cars that you would never purchase because they are too expensive or not sporty enough. Similarly a politician attempting to win over a constituency would probably do better to say he is opposed to raising taxes than to say he is opposed to raising library fines. And people will care where he stands on crime but not care what his favorite food is. When we make decisions there are certain factors that we consider much more important while

other factors are of much less important. Say that I like black cars that are reliable. If I find a black car that is not reliable, I am not likely to purchase it. If I find a reliable car that doesn't come in black, I may still purchase it in another color. Data mining using decision trees attempts to determine which factors in a decision or outcome have the most influence and order these factors in a tree in order of influence.

Time Series and Regression

Both of these techniques were standard statistical techniques long before data mining came along but have been added to the arsenal of data mining technique because of their ability to reveal important patterns in data.

Clustering

Thinking back to our movie theatre example, try to imagine a single fact that would be true for all movie goers. It is difficult to imagine anything that they all have in common. Some like popcorn and some don't. Some will stand in line for the latest blockbuster movie some will wait until later when it is less crowded. Some will prefer the afternoon matinee some will not. Some will come to watch old movies, art movies, or foreign films. Others will not. It is difficult to predict behavior with so diverse an audience. However, if we could group movie goers into meaningful categories then we might be able to predict behaviors of the viewers by category. For example, on group may stand in line for a blockbuster movie, buy popcorn and only watch evening shows. Another group may prefer to watch the movie in a less crowded theater several weeks later in the afternoon. The first group may respond to flashy advertisements promoting the latest blockbuster movie while the second group may respond to discounts or matinee pricing. The purpose of cluster analysis is find natural groupings within a set of data and use those natural groupings to predict features or behaviors.

The Data Mining Process

We can think of the data mining process as having five steps.

1) Mining Technique Selection
2) Data Acquisition
3) Model Construction
4) Model Validation
5) Model Explanation

First we decide what we are trying to figure out. Each data mining technique has the potential to tell us something different about the target of our investigation. The techniques discussed above, which are a subset of all available techniques, provide different results, sometimes very different. So the first step is to identify what we would like to learn from the data.

Second we have to find suitable data and massage it into the proper form for the data mining algorithm. This may be the most difficult step in the process. Data does not occur naturally in the proper form for data mining algorithms even if the data resides in a relational database. If the data does reside in a relational database it may be possible to extract it in the right form using standard SQL. However, it is more likely that some procedural code will have to be written to manipulate the data from its existing form into the form needed for a mining algorithm.

Third, we run the data through a mining algorithm and construct of model of the data. The nature of this model will vary based up the algorithm. For example, a regression analysis will produce a regression line whereas an association analysis or decision tree will produce association rules or decision rules.

Fourth, we need to validate the model that we just constructed by testing it against additional data. In step three we probably used a subset of the data to construct the model. Now we extract another subset and see if our model makes reasonable predictions. That is – how close are the predicted outcomes produced by our model to the actual outcomes in the test dataset. If the outcomes are reasonable close then our model will appear to be valid.

Fifth, now that we have identified patterns in the data we have to explain why those patterns exist. Going back to the movie theatre example we might explain the behavior of people who stand in line for a blockbuster movie by observing that they are not just going to the movie for entertainment. It is also a social experience. In order to maximize this social experience they have to see the movie when it first comes out and they stand in line so they can see the movie with other who are equally as excited to see the movie as they are thus increasing the effect of the audience in the social experience. They buy popcorn not because they like popcorn but because popcorn is an integral part of the social experience.

OLAP verses Data Mining

Data Mining provides a rich set of technique to discover patterns in data. OLAP provides a means of analyzing multidimensional data for the purpose of improving a key business process. Even though the terms data warehousing and data mining are batted around interchangeable. Both contribute in significantly different ways to understanding our business through the data it produces.

Lakoff, G. and Johnson, M. (1980) Metaphors We Live By. University of Chicago Press.

Fundamentals of Data Warehouse Design

Chapter 12: Future Trends in Data Warehousing

It is a unique opportunity to reflect upon an emerging technology of potentially profound significant and attempt to predict just how profound that significance may be. There is also a high degree of risk. After a half a century of artificial intelligence we still do not have an R2D2. Despite extensive research in voice recognition and natural language database interfaces we still cannot do as Captain James T. Kirk did on the Starship Enterprise where he says "Computer," and then provides a question requiring not only data but insight, interpretation and speculation to answer. On the other hand, I can recalling telling my telecom class in the early 1990's that in the future nobody would understand the notion of 'waiting by the telephone' for an important call. "People, will carry their telephones with them," I would explain, "and the network will just find you." As impressive as that sounded in 1992, it is quaintly amusing today. I also speculated, around that time, that shopping behavior would change. "You will be watching your favorite TV show," I offered as a setup, "and you see somebody wearing something you like. You will just click on the actor and a page will pop up showing that outfit on your frame. If you like it you can order it." Well, ordering things online certainly came about. But the integration between TV and the web did not. Nor did the virtual dressing room. Perhaps they will. Perhaps they won't. The point is that there is always some risk about predicting the future directions of technologies.

In order for one of these technological marvels to come about it first has to be technologically and economically feasible. Then there has to be somebody who wants to develop it. Frequently, the success of a technology requires that other technologies be at a certain stage of development. For example, if web technologies were developed when everybody was still using 300 BAUD dial up modems and 56 kbps was the standard for corporate leased lines, then, goodbye web technologies. Finally, the new technology, if it is going to be significant, must satisfy some basic human need. That is, people must want to use the technology and use it in a very serious way even if it means changing the way they do things. There are a lot of places in which this causal chain can go wrong. So there is a great deal of risk in predicting where a given technology is likely to go.

Having said all that, there is still value in speculation. Speculation gives us things to look for and allows us to think about potential impacts of emerging technologies. Sometimes speculations can actually help bring about technological change by providing ideas to developers. This chapter will wrap up the discussion of metric driven data warehouse design by exploring one major potential outcome and its implications.

Initially, data push and metric pull may appear to be legitimate competing paradigms for data warehousing. The epistemological flaw is a little difficult to grasp and the distinction - that information derived from a data push model is information about the dataset while information derived from a metric pull model is information about the organization – may also be a bit elusive. However, the implications are enormous. The data push model has little future, in that it is founded on a model of data exploitation rather than a model

of data. The metric pull model, on the other hand, is likely have some major impacts and implications. The most impressive of these is the application of Taylor's Scientific Management to white color work.

The Impact On White Collar Work

The data driven view of data warehousing limits the future of data warehousing to the possibilities inherent in summarizing large collections of old data without a specific purpose in mind. The metric pull view of data warehousing opens up vast new possibilities for improving the efficiency and productivity of an organization by tracking the performance of key business processes. The introduction of quality management procedures in manufacturing a several decades ago dramatically improved the efficiency and productivity of manufacturing processes, but such improvements have not occurred in white-collar work.

The reason that we have not seen such an improvement in white-collar work is that we have not had metrics to track the productivity of white-collar workers. And even if we did have the metrics we did not have a reasonable way to collect them and track them over time. The identification of measurable key business processes and the modeling of those processes in a data warehouse provides the opportunity to perform quality management and process improvement on white-collar work.

Subjecting white-collar work to the same rigorous definition as blue-collar work may seem daunting, and indeed that level of definition and specification will not come easily. So what would motivate a business to do this? The answer is simple - businesses will have to do this when the competitors in their industry do it. Whoever does this first will achieve such productivity gains that competitors will have to follow suit in order to compete. In the early 1970's corporations were not revamping their internal procedures because computerized accounting systems were fun. They were revamping their internal procedures because they could not protect themselves from their competitors with out the information for decision making and organizational control provided by their accounting information systems. A similar phenomenon is likely to drive data warehousing.

The potential impacts of this restructuring cannot be overstated and are of such magnitude that they are difficult to grasp. Yet an apocryphal story about Eli Whitney can give us some insight. Eli Whitney, one of the fathers of mass production, when into the White House one day, as the story goes, with six muskets. He put these six muskets on a table in the presence of the present John Adams. While the president was watching he took the six muskets apart mixed up the parts on the table and reassembled six muskets from those parts. The president was so impressed, the story goes, that get gave Whitney a major contact to provide muskets for the U.S. Army. Why was this demonstration such a big deal and what does it have to do with data warehousing?

First, we will consider why this demonstration was a big deal. Prior to this incident musket were custom made one at a time by individual craftsmen. If a soldier's musket

broke, it would have to be repaired by a specially trained craftsman who accompanied the troops. Otherwise it would have to be saved for later repair or thrown away. If one musket had a damaged barrel and another a broken stock, you would not take the good stock from one musket and the good barrel from the other and make a single working musket. So when Eli Whitney made six muskets from interchangeable parts, it was a break through in musket production. We could stop here with this story and observe that most databases today are custom made by individual craftsmen and that the parts (information) are not interchangeable. If a database encounters a problem it takes a specially trained craftsman to repair that specific database and you cannot take parts from one database (tables, views, stored procedures, data) and use them to repair another database. But, while that observation is true and perhaps useful, it is not the point of this story.

The point of the story is that Whitney's demonstration introduced a concept that is central to mass production manufacturing and that is the idea of a reusable part. Making reusable parts implies standardization, refinement, and economies of scale. When the same part is made over and over again the process by which the part is made can be refined and improved. It can be made cheap, faster and better. And to a large extent the wide variety of technology that we have available today is due to the manufacturing processes that make the technology cheap enough and reliable enough for consumer use. This is not to diminish the importance of creativity, science and design in the advancement of technology. It is only to say that if the computer were designed and developed but there were no manufacturing processes to produce a computer cheaply and reliably we would not be in the computer age.

Imagine all of the things we take for granted today that rely on effective mass production and the refinement of reusable parts. In fact think of all the things that would disappear if craftsman based production were not replaced by mass production. The computer I am using to write this book just disappeared. So did the network that I use to get to my email. My printer is gone. If I go into another room to find something to do, I might find that my TV set, and DVD player have disappeared too. I can't call anybody to find out what is going on because my phone is gone. I can't hop in my car to go somewhere because my car is gone as is the garage door opener that lifts that garage door out of the way. Maybe I'll just make something to eat. But my refrigerator, toaster oven and microwave are good too. This all happened so quickly. At least it seems as though it happened quickly. I don't really know because my watch and all of my clocks are gone. A gust of wind blows open my front door because the door knob (not usually considered as a high tech device) was also mass produced and disappears when mass production goes away. I can't even sit down and read a book because the book was not copied by a scribe. It was mass produced as was the light bulb. Things are looking pretty glum. Indeed it is very hard to imagine what life would be like had the concepts of reusable parts and the refinement of manufacturing processes never been introduced.

And by the same token, it would have been very difficult for Eli Whitney and the President to imagine all of the implications of the ideas that the demonstration with the muskets revealed. So lets go back to the moment for some further consideration. Whitney

has just reassembled the muskets and the president is duly impressed. He offers Whitney a huge contract and then pauses to reflect. He stokes his chin and contemplates the implications of what he has just seen.

"You know," he say to Whitney, "there is more to this demonstration than muskets. There is an emerging idea – the idea of interchangeable parts. This is a profoundly important idea. By making interchangeable parts you can refine manufacturing processes to produce products cheaper, faster and better. Eventually, somebody will invent a car and use this idea to make the car affordable to the average person. And then some day the computer will be invented. It will be made cheaply enough for everyone to have to have one and it will be made faster and faster, more and more powerful. It will bring about a new age – the age of information. But the age of information could also be called the age of technology because this simple idea will make it possible to produce technological implementations of ideas that would have exists only as ideas without it. The will be big screen TV sets, and DVD player, Ipods and video games. The possibilities are endless!!"

Of course this is absurd, and it is intended to be. As Whitney and the president stood there ruminating over the implications of Whitney's demonstration there is no way they could have see the vast implications of the idea of reusable parts. No only could they have imagined it in their wildest dreams, but if someone did know the implications it would not be possible for them to explain those implications. There would be nothing in the experience of these two men that would allow them to grasp the idea of, say cable television or a DVD player or a computer, for example.

In this way, it is difficult for us to imagine just how much things could change if we were to add a similar level of discipline to white collar work. But we can still try. If you were to ask most white collar worker what they do, they would, more than likely, respond with a vague answer such as – "I work for the government", or "I work for a trade association", or "I work for a paper supplier". In other word, most white collar workers define their jobs as 'working' for a company of some kind. But what does 'working' mean? If you were to press the point, and ask what they actually do in terms of specific activities they may, once again vaguely, mention going to meetings, making and taking phone calls, reading and writing memos, coordinating, scheduling, networking and so on. If you push the point even further, and ask how they know on a given day, week or month, if they did better than the preceding day, week or month they will probably just look at you as though you are some intellectual curiosity out of touch with today's world of work. Compare this with somebody whose primary responsibility is sales. If you ask this person what they do he or she will respond with something very concrete such as – "I sell paper products to government clients." When you ask this person how he or she knows on a given day, week or month if they did better than the preceding day week or month they will be able to tell you with some precision. "My goal was to increase sales of product X to client Y by 5% and I achieved that so I did better."

"But that is sales," you might say, "most white collar work is not like that." But is this true? What about help desks, customer service, call centers, and claims processing, These aren't sales. But they all have clear goals. Their performance can be measured and hence

improved. People who work in these areas know, when they come into work each day, what they are trying to achieve. Compare this with "going to meetings, making and taking phone calls, reading and writing memos, coordinating, scheduling, networking and so on." Why isn't all white collar work as clearly defined as these business functions? There are three reasons. First, even if all white collar work were more clearly defined, goal driven and measurable until recently the technology to management this data effectively was not available. Second, white collar work has managed to get along successfully without being terribly efficient. Third, many people feel that white collar work is simply not amenable to precise definition. Let's consider each of these in turn.

The first issue is easy to address. OLAP technology has only been available for a relatively short amount of time. That does not mean that it would have been impossible to develop a dimensional model of a measurable business process twenty years ago. It means that implementing that model and exploiting the data would have been difficult. In the past decade we have seen advances in OLAP technology as well as advances in our understanding of dimensional modeling. So modeling measurable business processes, collecting data to measure those processes, and using that data to improve those processes is become much better understood. With all of those factors coming together it is simply more practical now to begin thinking of a business as a collection of measurable and improvable processes. The second point is also as easily overcome. White collar work has gotten away with being very inefficient for a long time, just as manufacturing had gotten away with being inefficient for a long time before job redesign and quality improvements brought manufacturing into the modern age. The problem is not – how do you get white collar worker to be more productive? The problem is – how do you stay in business once your competitors have become more productive? The 1970's saw widespread adoption of computer technology and millions of dollars poured into the development of information systems. Organizations were gutted, disrupted and restructured to address the introduction of information technology. Nobody did this because it was inherently fun. They did it because everybody else was doing it. Once one bank had automated account management all competitor banks would have to do the same thing or loose their ability to compete. Same thing with order processing, inventory management, cash flow and other business functions in other firms. So, while white collar work has stayed out of the spotlight of efficiency improvements, it is unlikely that it will be able to stay out of the spotlight.

Finally, on the last point – many people feel that white collar work is simply not amenable to precise definition – it bears pointing out that in Frederick Taylor's day craftsmen were equally as wary of scientific management as white collar workers are today. Craftsmen pointed to the complexity of their tasks; the need for background knowledge; the ability to address exceptions; and the like. These are the protests of someone performing an unstructured job. Once the job is structured and idealized many of these problems go away. Of course, this changes everything and most protests are not really protests regarding the difficulty of idealizing white collar work. They are protests against the possibility of change.

So, if we grant a willing suspension of disbelief and agree for the sake of argument that white collar work can be described in terms of measurable business process what changes can we expect to see? The first result of this would be a tremendous increase in productivity. Most people in white collar positions cannot easily distinguish which of their activities truly contribute to productivity and which contribute to the social environment of the office. If one wanted to work harder would the go to more meetings, make more phone calls, write more documents, or what? Which well defined measurable business processes it become more clear what one needs to do in order to improve the process and be more productive. Measurement of outcomes is a tried and true method of improving performance and the measurement of business processes is likely to provide dramatic improvements in white collar work.

If we are able to measure productivity then we are likely to find that some employees are more productive than others. Since this productivity contributes to the bottom line of the firm, it should be rewarded. So more productive employees should earn more than lesser productive employees. In today's white collar environment it is not unusual to have one employee be many times more productive than another employee. But it is very rare to have one employee make many times more money than another employee. This is to say that productivity is not adequately recognized nor rewarded in today's white collar environment. Clearly, paying one employee five time more than another employee would lead to a host of other problems. But sales people often make different amounts based on their performance and executives often receive compensation based on performance. So it does not seem unreasonable to have all employees whose performance is measurable receive differential compensation based on performance.

Let's think about this idea of each employee being compensated based on measurable performance and see where it may go. The industrial age saw major shifts in the distribution of population as people left their homes to work in cities and factories. Today we still the impact of that trend as populations center around cities where work is available and most people spend some of their valuable time commuting to an office where they can work in proximity to others. Much time and energy is spent so that people can work in close proximity to others. Why is this? One reason might be that people need to communicate. However, with modern information technology from email and instant messaging to voice and video technologies, it is hard to argue that people need to be anywhere near each other in order to communicate. In fact, it would be hard to argue that they need to be on the same continent in order communicate so that cannot be the reason.

For years people have been talking about telecommuting and working from home. And while there are many instances of this occurring, the overwhelming bulk of employees still hop in a car and go to an office. One reason for this may be that if your employer is going to pay you for eight hours of work they would like to get eight hours of work out of you. If you work at home they cannot be sure you are working. If you work in an office, at least they can see that you are there. Whether or not you are working is another matter. But you are there. And, indeed, you are being compensated for your time rather than your productivity. However, when employees are working on measurable business processes we know exactly how productive they are whether they are working in the office next

door or on the moon. So, instead of compensating employees for their time, we can compensate them for their work. And the need to be in an office to demonstrate that you are putting in your time begins to diminish.

This notion can be carried yet a little further. If employees to do not need to show up at an office in order to work, do they need to be employees of the firm that owns the office? We are really going to push this idea to its limits here, but why would corporations need employees if they can get all or most of their work done through independent contractors. If employees work on measurable business processes and get compensated for their performance on those processes then why can they be independent contractors doing work for hire? With a little imagination, one can imagine a corporation of the future in which all work is outsourced to independent contractors. These contractors work from where ever they choose to work and work as much or as little as they choose based on their productivity and their need for remuneration. So a marginally productive individual may work a full week for a moderately satisfactory level of income whereas a highly productive individual could choose to work fewer hours or to earn more money.

When a corporation needed some work done they could put a proposal on the Internet and individual contractors could bid on that proposal. The historical productivity of these contractors would be a matter of public record so the corporation would not be taking a major risk in hiring individuals with which they did not have previous experience. In fact, in the extreme the hiring process could be handled entirely by intelligent agents for the most efficient allocation of workers.

Now, this idea of intelligent agents bargaining for work, and individuals all working as independent agents pushes one to the limits of their imagination. But this is not because it is so unlikely. It is merely because it is so different from today that it is difficult to imagine. But the point of this mental exercise was to stretch our imagination to see just how much the world could change as a result of data warehousing technology and the application of Scientific Management to White Collar work. Certainly, other scenarios are possible. But hopefully the point has been made that the impacts of this technology can be enormous. And if these changes start coming about there will be a proportional amount of interest in research related to data warehousing. So next we turn to new directions that we might expect in data warehousing research.

→

The Implications For Research

The implications for research in data warehousing are rather profound. Current research focuses on issues such as data extraction and integration, data aggregation and summary sets, query optimization and update propagation. All of these problems are applied problems in software development and do not advance our understanding of the theory of data.

But a metric driven approach to data warehouse design introduces some problems, whose resolution can make a lasting contribution to the theory of data. Research problems in metric pull data warehousing include: 1) how do we identify key business processes; 2) how do we construct appropriate measures for these processes; 3) how do we know those measures are valid; 4) how do we know that a dimensional model has accurately captured the independent variables; 5) can we develop an abstract theory of aggregation so that the data aggregation problem can be understood and advanced theoretically; and, finally, 6) can we develop an abstract data language so that aggregations can be expressed mathematically by the user and realized by the machine?

The relational model introduced Structured Query Language, an entirely new data language that allowed non-technical people to access data in a database. SQL also provided a means of thinking about record selection and limited aggregation.

Research in data warehousing will likely yield some sort of a dimensional algebra that will provide, at the same time, a mathematical means of describing data aggregation and correlation, and a set of concepts for thinking about aggregation and correlation. To see how this could happen, think about how the relational model led us to think about the organization as a collection of entity types, or how statistical software made the concepts of correlation and regression much more concrete.

In the organization today, the database administrator and the statistician seem worlds apart. Of course, the statistician may have to extract some data from a relational database in order to do his or her analysis. And the statistician may engage in limited data modeling in designing a data set for analysis using a statistical tool. The database administrator on the other hand will spend most of his or her time in designing, populating and maintain a database. A limited amount of time may be devoted to statistical thinking when counts, sums or averages are derived from the database. But largely these two individuals will view themselves as participating in greatly differing disciplines.

But with dimensional modeling the gap between database theory and statistics begins to close. In dimensional modeling we have to begin thinking in terms of construct validity and temporal data. We need to think about correlations between dependent and independent variables. We begin to realize that the choice of data types (e.g. interval or ratio) will affect the types of analysis we can do on the data and hence potentially limit the queries. So the database designer has to address concerns that have traditionally been the domain of the statistician. Similarly, the statistician cannot afford the luxury of constructing a data set for a single purpose or a single type of analysis. The data set must be rich enough to allow the statistician to find relationships that may not have been considered when the data set was being constructed. Variables must be included that may potentially have impact, or may have impact at some times but not others, or may have impact in conjunction with other variables. So the statistician has to address concerns that have traditionally been the domain of the database designer.

What this points to is the fact that database design and statistical exploitation are just different ends of the same problem. Once these two ends have been connected by data warehouse technology, a single theory of data must be developed to address the entire problem. This unified theory of data would include entity theory and measurement theory at one end, and statistical exploitation at the other. The middle ground of this theory will show how decisions made in database design will affect the potential exploitations so that intelligent design decisions can be made that will allow full exploitation of the data to serve the organization's needs to model itself in data.

Synergies

According to Information Navigators, there are approximately 72 million hosts on the web. Many of these hosts do not support web sites, but many others (such as those owned by ISPs) have multiple web sites. If only 10 million of these web sites attract as few as 100 visitors a minute, then 1 billion records are generated every minute. This becomes 60 billion records per hour, or 1.4 trillion records per day. This is an enormous amount of information providing great insight into the behavior of consumers and information seekers, among others. This huge volume of temporal data cannot be exploited effectively without data warehouse technology. Hence, the growth of the web may well push the growth of data warehousing.

At the same time, data warehousing concepts continue to evolve. What started out as a dumping ground for used data is rapidly becoming a rigorously defined set of dimensional models to track key business processes. The technology is improving also. OLAP tools are making it easier to analyze dimensional data through visual interfaces and data mining tools are making it easier to find useful patterns in large volumes of data. In today's tools there are gaps between SQL and OLAP, and again between OLAP and data mining. Data mining approaches are segmented between visualization, statistical and artificial intelligence. These disparate approaches will eventually become integrated in seamless tool sets that provide a variety of analytical techniques along with some guidance on which approaches to use.

These OLAP and data mining tools will become more web friendly as demand to access data warehouses from disparate locations increases. It is conceivable that at some point in the very near future that web applications become the primary source for data in most corporate data warehouses, while web technologies become the primary vehicle for accessing that data.

Summary

This chapter looked at the potential impact on white collar work of data warehousing technology. It was in many ways an exercise in technology forecasting with a very speculative flavor. The point was not to predict where this technology might go with a high degree of accuracy. Instead it was speculate about where it might go with a reasonable degree of plausibility. The impacts on white collar work can be profound possibly bringing about major changes in how we view work. If impacts of the magnitude

suggested here do begin to come about then more attention will be paid to research in data warehousing. So some speculations on the future of data warehousing research were also provided. Finally, to come back to reality after a journey into a possible future, some practical synergies between data warehousing and web technologies were addressed. The purpose of this chapter was to be speculative and provide a vision of a possible world in which data warehousing technology has made a major impact. As for what will really happen, we will just have to wait and see.

Jarke, M. et.al. (2000) Fundamentals of Data Warehouses. Springer.