



## ***Implementing a data warehouse for success And deploying to the web***

### **Index**

<b>Your Data Warehouse: A Business Success or Science Project?</b>	<b>3</b>
<b>Phase 1: Business Exploration</b>	<b>3</b>
Business Requirements Analysis: The Missing Link of Data Warehousing	3
<b>Phase 2: Project Definition</b>	<b>3</b>
<b>Phase 3: Product Validation</b>	<b>4</b>
Manageability Issues	5
Ease of Use Issues	5
Enterprise Scalability Issues	5
Making the process of evaluating data warehousing software more effective	6
Get references	6
Multiple vendor demos	6
Stock analyst reports	6
Check how well the software handles maintenance	6
Understand the trade-offs the software makes	6
Vendor road shows	6
Check the financial stability of the vendor	7
Have a representative team perform the evaluation	7
Evaluating an end user tool	7
<b>Phase 4: Implementation</b>	<b>7</b>
Getting Started with Data Warehousing	7
Look into the publications on the Technical Evaluations on the WWW	8
Go to a data warehousing conference	8
Read up on some fundamental technical topics	8
Bringing Performance to Your Data Warehouse	8
User Requirements	8
The Flow of Data	8
Performance Issues	9
Addressing Bottlenecks	9
The Hardware Approach	10
Parallel Processing	10
Multidimensional Servers	10
Distributed Data Warehouse	11
The Software Approach	11
OLAP	11
Data Marts	11
Partitioning	11
Query Regulation	11
Data Warehousing Gotchas	11
<b>Phase 5: Production</b>	<b>13</b>
The Issues	14
<b>Phase 6: Before Web-Enabling Your Warehouse</b>	<b>15</b>
<b>Phase 7: Deploying to the Web</b>	<b>16</b>
Guide for Enterprise Web Applications	16
Application Server Architecture	16
Data Driven	17
Enterprise Class Integration	17
Rapid Application Development (RAD) Environment	17
Open, Standards-based	18



## **Your Data Warehouse: A Business Success or Science Project?**

The decision to try and reap the benefits of a data warehouse has been made. Now what? The instinctive reaction of many IT Groups is to jump right into “speeds and feeds” mode and begin the vendor interrogation process. How big can we make it? How long to load umpteen terabytes? How do we index/partition/distribute/manage this thing? How do we sift through the vendor hype? The successful decision support solutions typically start with another line of questioning entirely. A business based methodology is needed to plan, implement, and support a data warehouse solution that will meet the specific goals of the users.

The following process can be used to ensure that the exhaustive efforts involved in implementing a data warehouse result in measurable benefits in a short time-frame, at a reasonable and justifiable cost while preserving the sanity of those chartered with the implementation.

### **Phase 1: Business Exploration**

The purpose of the business exploration phase is to clearly answer the questions “What?” and “Why?” The activities in this phase draw out end-user information requirements that translate into a need for decision support data. At the end of Phase 1, the outcome should be a prioritised list of business initiatives that can be supported by the development of decision support applications, requirements statements that specify end-user information needs and associated benefits and a profile of DSS users. Additional outcomes might be a list of data elements that satisfy end-user information requirements and a business case that demonstrates the value of initial data warehouse implementation.

### **Business Requirements Analysis: The Missing Link of Data Warehousing**

The primary goal of any data warehouse project is to turn data into information. “Classic” data warehouse design techniques emphasise the essential role of an enterprise data model (EDM) as the foundation for building the data warehouse. While this is true, the EDM only represents half of what is required to turn data into information. A well-constructed data model based on production systems and other known sources of data allows for an understanding of what data is available to the organisation. But what information do the intended users of decision support and other informational processing capabilities require?

Imagine a pyramid with data from production sources as the base and the end users at the pinnacle. The EDM portion of a data warehouse design would be described as “bottom-up,” while determining the organisation’s requirements for information from key end users would be the “top-down” portion.

The top-down portion of data warehouse design is accomplished through a Business Requirements Analysis, which consists of:

1. understanding what the organisation’s business goals and objectives are;
2. determining what metrics are used to monitor those goals and objectives; and
3. decomposing those metrics into natural language descriptions and their component parts.

These resulting component parts can then be mapped to data elements in production systems. This mapping of the component parts of decomposed business metrics to data elements found in production sources of data is the merging of the understanding of what data is available to the organisation from the bottom-up with the understanding of what information is required by (or at least requested by) the organisation from the top-down.

The results of a Business Requirements Analysis are essential business meta data and the ability to focus the scope and goals of the data warehouse to meet the objective of turning data into information.

### **Phase 2: Project Definition**

Once information requirements are established, we help answer the question “How?” Project definition is analogous to the design phase in common methodologies in that its purpose is to develop a clear picture of the implemented solution. A basic technical architecture is

established that includes software components, performance constraints, access volumes, and platform selections.

It is worth bearing in mind during this phase that in excess of 50 percent of the data warehouse projects undertaken are considered to be failures or, at the very least, unsuccessful after 18 months. The definition of failure is based on one or more of the following measures:

1. The project was not delivered or completed.
2. The project was delivered and the data warehouse turned out to be a data basement, i.e., no one could use it.
3. The project was improperly sized and architected and misaligned with the business.
4. The project was abandoned mid-stream and restarted on a new development path.

Why do data warehouse projects end in failure? Here is a summary of the most common reasons:

1. Paralysis by analysis coupled with an elongated evaluation and tender period. A preoccupation with technology versus the real business needs pollutes the decision-making process.
2. Political issues (or an all out war) centred on control between IT and the business users: who funds and who owns the data warehouse; when it will be delivered to the business; can the existing operational world (MVS/mainframe) (a) run the data warehouse and (b) be cost effective if it can run it.
3. Unclear and inconsistent or non-existent short-term goals (deliverables). Too much long-term strategic focus inhibits quick hit deliveries.
4. Depending on leading-edge hardware technology and software just out of the lab to deliver a production data warehouse this year.
5. A mistaken belief that bench-marking and the resulting price/performance evaluation of each competing vendor will lead to the right decision.
6. Picking up "how to" or "do it yourself" books (or education) for your staff and thinking that that's enough to ensure success. Data warehouse projects use off-the-shelf and open systems components; they do not use off-the-shelf integration experience or project management.
7. Having no defined query tool methodology or understanding of what the real users need to have, to be effective.
8. Using summary and sample data as the basis for the data warehouse. The secret of data warehouse success is in detailed transaction data.

### **Phase 3: Product Validation**

The purpose of this phase is to prove that the selected vendors and products can meet expectations in implementation. Tools vendors claim to be infinitely flexible and database vendors claim to be infinitely scaleable. While common sense will tell us that multi-terabyte demonstrations on small commodity servers are unrealistic, we have to dig deeper to determine the overall fit of the products to the project. The outcome of this phase will predict if a tool can perform the needed analysis, if the servers can deliver the desired performance, or if the data has the desired business value. This phase can significantly reduce the project risk by performing one of the following activities:

- Technical proof of concept
- Concept demonstration
- On-site pilot
- Benchmark
- Reference visits
- Formal proposals

In spite of the many benefits of data warehousing, first-generation data warehouses have exposed many issues and challenges in effectively delivering the significant advantages that they promise. Some of these issues are present in all data warehouses, although many are aggravated as the data warehouse grows in size and complexity. These issues are:

- manageability,
- ease-of-use,
- and enterprise scalability.

## Manageability Issues

Key among these issues are manageability and operational issues. New tools have appeared in the past couple of years to deal with the difficulties in building a data warehouse and building both technical and business metadata. But no tools have been available for managing the data warehouse once it is built.

The following are some key manageability issues:

- How do you tune the data warehouse for performance without impacting saved queries and decision support applications?
- How do you build the large summary tables necessary for data warehouse performance without impacting the availability of your warehouse?
- How do you effectively manage security in the warehouse environment?
- How do you manage the partitioning of the data warehouse, especially in a heterogeneous RDBMS environment?

## Ease of Use Issues

Other key issues in the data warehouse involve ease-of-use for end users. For example:

- How do you make warehouses easy to use for non-technical business users? Research has shown that only 10 percent of warehouse users are knowledge users. Since most knowledge users cannot run ad hoc queries, they rely on pre-canned reports. In this type of static environment, they do not take full advantage of the warehouse's potential because they cannot ask the questions that they should be asking.
- How do you provide ease of use while allowing the flexibility of ad hoc queries and avoid heavy IT resource expenditures in application development?
- How do users quickly and easily determine what information is available in one or more large data warehouses?
- How do users determine the precise meaning of data?

## Enterprise Scalability Issues

A final set of issues are directly related to scaling a data warehouse into a large warehouse. Some of these issues directly relate to huge size, although many of them relate to support for very different subject areas, data sources, and user **Groups**. For example,

- How do you query across multiple subject areas? For instance, how do you relate both orders and shipments in the same query?
- How do you integrate and query internal corporate data with external (or syndicated) data?
- How do you integrate multiple warehouses in decentralised companies?
- How do you physically manage extremely large data warehouses, ranging up to the terabytes in size?

No Company may face all of these issues, but many of these issues do or will apply as the company's warehouses are established and grow.

## **Making the process of evaluating data warehousing software more effective**

Here are some ideas that may make the process of evaluating data warehousing software more effective. This is not a comprehensive list of tasks to follow in a technology evaluation. Rather, these are points that seem to be rarely discussed or followed in this wave of interest in data warehousing.

### **Get references**

Talking to reference sites is one of the most effective means of getting practical information. You would be surprised how important operational issues surface while doing evaluations. Some hints on reference gathering practices that have worked for me are:

- Ask the software vendor for a complete list of referenceable sites - Try to have options as to which organisations you will call.
- If this is a major decision for your company, call 5-6 sites - You need a minimum number of sites to help you detect patterns.
- Make a telephone appointment to talk with the reference - The reference will appreciate this.
- Plan on 20 minutes with the reference - Again the reference will appreciate this.
- Send your questions to the reference in advance - Some of the references will be more comfortable if they know what you'll be asking.
- Send a thank you note to your references asking if it would be okay to make a quick follow-up call if necessary - This will lay the groundwork if you have to call about another issue.

### **Multiple vendor demos**

If you are going to see multiple vendor demos, build a test case that each vendor will follow. This will allow you to compare apples to apples and peaches to peaches. Leave some open time at the end of the demo so the vendors can show features that were not covered well in the test case. One more point. Because departing from the standard vendor takes time and effort on part of the vendor, many will be unwilling to do this unless you are talking about a major purchase.

### **Stock analyst reports**

Read stock analyst reports on publicly held vendors and the industry outlook. Many times these reports can be an excellent source of information.

### **Check how well the software handles maintenance**

Most of the time spent with a software tool will be with maintenance. See how well the tool handles changes. For instance, most tools work with something like a data dictionary. See what are the consequences of changing the name of a field in the data dictionary. See how the dictionary helps you locate and change queries, reports, forms, macros, etc. that may be affected by the name change.

### **Understand the trade-offs the software makes**

Designers of tools trade off speed, capacity, computer resource consumption, ease of development, ease of use, and ease of maintenance. For example, several report and query tools can be made quite accessible to end users if you are willing to maintain extensive data dictionaries. Several OLAP tools attain quick retrieval times by requiring the storage of huge amounts of pre-calculated numbers. To prevent some nasty surprises once the tool has been purchased, make sure the persons making the buying decision understand these trade-offs.

### **Vendor road shows**

Go to the vendor road shows to talk with other attendees. Sometimes the audience at the vendor road shows is the best source of information. If you'll make a point of talking with

several other attendees, chances are you will come across a person who is in at the same stage in evaluating warehousing tools. You will find that you and that person can exchange information that is mutually beneficial.

### **Check the financial stability of the vendor**

If you work for an organisation with an accounts receivable department, the people in that department can help you with this. A simple check could save you some major potential grief.

### **Have a representative team perform the evaluation**

Often technology acquisitions fail or go awry because a **Group** within an organisation felt it did not get its views heard during the evaluation. One of the first steps in a technology evaluation is to identify all 'interested parties' in the acquisition. Make sure these parties are asked how they want to be represented in the evaluation. If parties that are in conflict with each other will actively participate, if you do not have the skills and/or patience to be a mediator, seek the services of an outside facilitator. Facilitation skills can be especially helpful if you have sessions dedicated to setting criteria, making your short list, and making the final decision.

### **Evaluating an end user tool**

If you're evaluating an end user tool, let an end user lead the evaluation effort. It seems odd but some organisations buy end user tools with little input from the end users of these tools.

## **Phase 4: Implementation**

Each implementation phase must be a customised project supporting the activities of infrastructure development, requirements analysis, design, development, and organisational change. These activities should include:

- Logical database design
- Open database access & installation
- Tuning
- Application construction
- Systems initialisation
- Physical database design
- Project management
- Operations analysis, design development & support

## **Getting Started with Data Warehousing**

If you are new to this field and the way you like to get into a new field is by getting an overview, we suggest that you read the books "Building the Data Warehouse" by W. H. Inmon, "The Data Warehouse Toolkit" by Ralph Kimball, "Data Warehouse from Architecture to Implementation" by Barry Devlin, and "Data Warehousing in the Real World" by Sam Anahory and Dennis Murray.

With due respect to all the other fine books on data warehousing and decision support, when read in combination we believe these four books provide a great introduction and overview of the strategic and tactical issues system developers face. Especially valuable are Inmon's overall overview, Kimball's description of data modeling and query/report tools, Devlin's descriptions of data extraction, cleaning, and loading issues and metadata, and Anahory/Murray's description of the tasks that must be done so a system can run efficiently and their description of the main tasks in a data warehouse project.

Visit a couple of organisations that have had warehousing systems in production for over a year. You will get an excellent education if you can ask an organisation who 'has done it' what are the biggest issues it faced in developing systems and what are the biggest issues it faces in maintaining systems. Also, ask what the organisation felt it did right and what it felt it could have done differently. We believe that if you do this you will better see a side of data warehousing that does not come out clearly in the data warehousing literature.

### **Look into the publications on the Technical Evaluations on the WWW**

Some excellent, in-depth studies of different parts of data warehousing technical architecture are available. These studies are expensive and their descriptions of product features get out of date in a hurry. But they are invaluable for the context they provide for deeper understanding of data warehousing and decision support technical and managerial issues.

### **Go to a data warehousing conference**

None of these conferences is inexpensive. Their worth comes from, again, the exposure you get to issues that are not well covered in the vendor material and books and magazine articles. You will learn as much from talking with the conference attendees as you will from attending the conference sessions.

### **Read up on some fundamental technical topics**

You may find you will be greatly helped by reading up on SQL queries (especially multi-table and summary queries and subqueries), database indexing, join processing, and how query optimisation works. Also helpful would be some knowledge about how applications can be partitioned in a "traditional" client/server environment and how partitioning can work in an Internet/Intranet environment.

## **Bringing Performance to Your Data Warehouse**

### **User Requirements**

Today, users want and need on-line access to these large amounts of historical data to do various analyses. And remember, they want that data now! Understanding past trends can help an organisation plan for the future. Evaluating the buying trends of customers can assist a marketing department's sales strategy. Assessing a product's historical performance can provide valuable insight into the maintenance of inventory, the introduction of new products, or the retirement of older products. These are just a few of the ways a data warehouse can contribute to an organisation's decision making process.

To provide a system to meet the organisation's decision support needs, Information Systems must work closely with the end users to understand their data and reporting needs. Even after building the data warehouse with the data required, IS must continue to work with the users to address frequently changing needs. This is because the process of analysing data and turning it into information requires a dynamically changing environment. Analysis begets more analysis, which begets even more analysis... and on and on! As the user learns more from the data, the information gleaned will prompt more questions and conclusions, as well as new requirements. IS must design a system that will lend itself to this ever-changing environment.

Performance is one area that will need to be continually addressed. An organisation's growing history of data must be readily available, on-line. The flow of data from the data warehouse to the user must not be hindered by a system that cannot easily and quickly perform queries to the data.

### **The Flow of Data**

If we look at the data warehousing environment, we see that information flows as illustrated by the following examples:

Information flows out of the legacy systems into the data warehouse. The data from legacy systems may be merely loaded into the data warehouse, or the data may actually be massaged or scrubbed before it enters the data warehouse. Once the data warehouse is built, the users can begin to access the information for analysis and the retrieval of data.

Granted, there are other components of the data warehouse (such as metadata, security and warehouse management), but for our purpose, this paper will deal with the basic four components:

- the legacy systems,
- extraction,
- the data warehouse,

- and the front-end access.

We must now consider how freely information flows through the data warehouse environment. Freely refers to the users' access to data, not from a security standpoint, but more from the ability to find and use data for decision support. Is the data truly accessible and available, or are there bottlenecks?

### **Performance Issues**

Bottlenecks between the legacy systems and the data warehouse may be a result of volume or co-ordination of the data. As the data flows out of the legacy systems, there may be issues of synchronisation. As we build our subject-oriented data warehouse, one system may lag behind another in delivering related data. Once the data is loaded into the data warehouse, we are then faced with a possible retrieval bottleneck. "Data warehouse", by virtue of the definition, implies BIG databases. Each knowledge worker must read through all this data to extract the information needed. All this processing of the data in the data warehouse will certainly impact performance and cause bottlenecks for the decision support application.

Let's consider these retrieval bottlenecks. What could be contributing to them? There is really only one possibility: too much data is being processed! Data warehouses adversely impact I/O processing. The access of these large amounts of data can slow down any transfer of data from the disk drive to the processor. This is because the processor is forced to read through all the data in order to process the users' queries. Networks will also experience the effects of large volumes of data being moved to users' PC's. In this case, users may unnecessarily transfer large amounts of data down to their PC's for analysis.

Even if users require less data than expected, they can still affect the system performance by entering an erroneous request. That request may again cause too much data to be processed! Even worse, if the user gets the wrong information, the user may actually make the request again. In an attempt to correct the problem, the user may think that the next request will be the right one. This is magnified when applications are too complicated for the average user to understand. Needless to say, valuable processing time is being used to retrieve, or sometimes just read through, volumes of wrong information.

Ideally, we should allow the users to consider their requests, whether by taking various slices of the data, or by starting with a large view of the data and then whittling it down to smaller amounts of data. Users should not have to worry that the requests they make may adversely impact performance! Also, we should not waste the users' time. Knowledge workers should not have to wait hours for the results, or work late when the system is least busy. It becomes imperative that we address the performance bottlenecks before the data warehouse is built!

### **Addressing Bottlenecks**

There are various options being proposed in the data warehouse environment to address performance bottlenecks. Some are new technologies. Others have been around for some time. They are:

- massive parallel processing (MPP)
- multidimensional servers
- distributed warehouse
- OLAP (or multidimensional queries)
- data marts
- partitioning
- query regulation/police
- Advanced Indexing: inverted indexes, bit-map indexes

We can categorise these technologies into two areas: those that impact the hardware environment and those that impact the software environment. In hardware, there is massively parallel processing (MPP), dedicated multidimensional servers and distributing the warehouse across processors. In software, there is OLAP, partitioning the data warehouse, distributing portions of the data warehouse to application specific needs (data marts), query regulation, and indexing.

As we examine these options, two parties must always be considered: the user **Group** and the Information Systems **Group**. Any solution must benefit both **Groups**. Too often, a solution benefits one **Group** while the other **Group** suffers. The knowledge workers may not get the system they need to do their jobs. The IS **Group** may become burdened with a solution that is cumbersome and expensive to maintain. This in turn can cause requests for additional solutions, and the users to suffer. As we discuss each option, we will keep this in mind and note whenever an option could present a “win-win” solution for both **Groups**.

## **The Hardware Approach**

If we first consider the hardware approaches, we see technology moving towards faster processing and bigger data storage capabilities. The computer industry has always been able to build a better chip that will process data faster. The industry has also been able to build on-line storage devices that can hold massive amounts of information. These advances allow us to process more data faster. If the system starts to run slow, buy a faster processor. If we need to store more data, purchase a bigger storage drive. This may appear to be a simple solution to our performance problem. The drawbacks are that this may be a very expensive solution and it may provide a quick, but only temporary fix. This is partially because the connection between the fast processor and the big drive has not made proportionally the same advances.

What seems to be lagging is the ability to build a “pipeline” that effectively moves all that data between the fast processor and the huge data storage device. Whether the “pipe” is direct to the CPU or over a network, the movement of data from storage to CPU has continued to constrain the system from performing at its peak. We continue to hear warnings in the data warehouse arena to make sure we “manage” the network now that data warehousing is introducing such volumes of data movement across systems. So how will a faster processor address the management of massive amounts of data if there are still issues about getting the data to and from the processor? It cannot, without help.

## **Parallel Processing**

One proposal is to give the processor some more processors to help do the work. Parallel processing divides the work among multiple processors. This requires that the relational database management system be able to divide its work among these processors. Therefore one processor could scan the database, while another could sort data, while another could perform a table join. If this is not a feature of the relational database, parallel processing becomes a task of IS, where IS must programmatically “funnel” each task to a processor. If you have been in the industry for any length of time, you may recognise this as “throwing more hardware at the problem.” It aids in relieving performance bottlenecks, but does not provide a permanent solution to the I/O bottleneck.

A way of thinking about parallel processing is to compare it to handling rush hour traffic. If we have a million cars coming into the city every morning, would it be better to add more roads (or lanes on the highway), or would it be better to lessen the number of cars coming into the city? If more people car-pooled or took public transportation, there would be less cars on the roads. This is the concept behind parallel processing (more lanes) versus indexing (less cars traveling the same roads). With parallel processing, users may see an immediate improvement in performance, but the tendency is for the performance to flatten, then deteriorate. As users experience a performance degradation, this is an indication that a processor needs to be added, putting IS in a constant reactive mode.

## **Multidimensional Servers**

Another option is to create a specialised database that will maintain the data in a format geared toward the type of retrievals users will be making. These multidimensional databases require investment in proprietary databases and are often placed on a dedicated server. Multidimensional servers require new design approaches, consequently impacting the IS maintenance load and costs. Using relational technology and standard RDBMS's eases maintenance issues -- both resource maintenance and administration.

## **Distributed Data Warehouse**

Distributing the data warehouse takes a similar approach to parallel processing. Divide the data warehouse into data subsets and place each on a separate processor unit. We again distribute the workload. Like parallel processing, this can become a maintenance nightmare for IS.

## **The Software Approach**

With the software approach we address the performance dilemma from the standpoint of the user. What if we could control the types of queries the user will do?

### **OLAP**

OLAP (on-line analytical processing), or multidimensional querying, tunes the system for analysis, much like the OLTP approach which tunes performance around the transaction. OLAP or multidimensional querying takes the approach of defining the "dimensions" that the knowledge worker will require. OLAP claims to support complex queries, but appears to lend itself more to a "predictive" analysis of the data warehouse.

### **Data Marts**

Another form of distributing the data creates "mini" data warehouses (data marts). Data marts also address specific decision support application needs. These subsets of the data warehouse can be on one or multiple hardware platforms. Distributing the data can become a burden for IS. If you consider the dynamic nature of decision support systems - an application that changes often - then any environment that requires a continual assessment of design, location and access of the data can become a major undertaking to maintain. Similar to the "islands of automation," data marts can lead to "islands of data warehouses," where IS will need to build bridges between data marts.

### **Partitioning**

Partitioning improves the retrieval performance by segmenting the data into logical areas. A good example of this is if you were to automate all the phone books in the country. Very likely, the letter "S" may be about 25 - 35% of all the names. This is based on the fact that "Smith" is the most common English surname. You could partition your data so that all surnames starting with "S" are on their own partition. Therefore, when you need to look up a name beginning with "S", you would only look at the one set of data. Also, when you look up any surnames beginning with letters other than "S", you would not have to read through the "S" data.

The downside to partitioning is that it requires constant tuning in a dynamic environment where unpredictable queries are made. A query this week may look at one partition, but next week the same user's queries may span two or more partitions. This could conceivably become a warehouse management nightmare for IS, as well as delay users needs for ad-hoc queries.

### **Query Regulation**

With query regulation, we now "manage" the user's side of the decision support application. This method encompasses "trapping" the user's query, assessing the query and then determining if, when and how the query will be performed. This option is not totally transparent to the user (especially if it is determined that the query will not be performed, or will be performed at a later time). This could not only be a full-time support issue for IS, but query regulation could also adversely affect the knowledge worker's ability to freely access the data warehouse.

## **Data Warehousing Gotchas**

Here are some points for the warehouse builder we rarely see discussed or we do not see discussed enough in the barrage of articles about data warehousing. Forewarned is forearmed!

**You are going to spend much time extracting, cleaning, and loading data**

W. H. Inmon, in his data warehousing books, estimates that, on average, 80% of the time building a data warehouse will be spent on this type of work.

**Despite best efforts at project management, data warehousing project scope will increase**

To paraphrase Inmon, traditional projects start with requirements and end with data. Data warehousing projects start with data and end with requirements. Once warehouse users see what they can do with this technology, they will want much more. (Which is fine!) One piece of advice for the warehouse builder is never to ask the warehouse user what information he wants. Rather, ask what information he wants next.

**You are going to find problems with systems feeding the data warehouse**

Problems that have gone undetected for years will pop up. You are going to have to make a decision on whether to fix the problem in what you thought was the 'read-only' data warehouse or fix the transaction processing system.

**You will find the need to store data not being captured by any existing system**

A very common problem is to find the need to store data that are not kept in any transaction processing system. For example, when building sales reporting data warehouses, there is often a need to include information on off-invoice adjustments not recorded in an order entry system. In this case the data warehouse developer faces the possibility of modifying the transaction processing system or building a system dedicated to capturing the missing information.

**You will need to validate data not being validated by transaction processing systems**

Typically once data is in the warehouse many inconsistencies are found with fields containing 'descriptive' information. For example, many times no controls are put on customer names. Therefore, you could have 'DEC', 'Digital' and, 'Digital Equipment' in your database. This is going to cause problems for a warehouse user who expects to perform an ad hoc query selecting on customer name. The warehouse developer, again, may have to modify the transaction processing systems or develop (or buy) some data scrubbing technology.

**Some transaction processing systems feeding the warehousing system will not contain detail**

This problem is often encountered in customer or product oriented warehousing systems. Often it is found that a system which contains information that the designer would like to feed into the warehousing system does not contain information down to the product or customer level. By the way, this is what some people label a 'granularity' problem.

**Many warehouse end users will be trained and never or seldom apply their training**

I once read a study that claimed that only one quarter of the people who get training in a query tool actually become heavy users of the tool.

**After end users receive query and report tools, requests for IS written reports may increase**

This phenomenon was seen with many of the information centres of the 1980s. It comes about because the query and report tools allow the users to gain a much better appreciation of what technology could do. However, for many reasons, the users are unable to use the new tools themselves to realise the potential. By the way, if this happens do some honest research on why. Granted there are many reports that are so complex that IS expertise is going to be required no matter what tool the end user has. However, many times this phenomenon points to training needs.

**Your warehouse users will develop conflicting business rules**

Many warehouse tools allow users to perform calculations. But, the tools will allow users to perform the same calculation differently. For instance, suppose you are summarising beverage sales by flavour category. Also suppose that the flavour category includes cherry and cola. If you have a cherry cola brand there is a chance that two users will classify the brand in different categories. You will find that there are means to incorporate some of the

business rules in your warehouse. However, the number of possible business rules is so large that you will not be able to incorporate all rules.

### **Large scale data warehousing can become an exercise in data homogenising**

Data has quirks! Sometimes when we developers combine detailed data for different subjects, in our efforts to make everything 'fit' we can take the life out of the data. For instance, if your company sells food and drinks, you want to be careful if you are building a sales data warehouse for both lines of business. You have to make a judgment call as to whether these businesses fit the same logical and/or physical model.

### **'Overhead' can eat up great amounts of disk space**

A popular way to design a decision support relational databases is with star or snowflake schemas. Persons taking this approach usually also build aggregate fact tables. If there are many dimensions to the data, be aware that the combination of the aggregate tables and indexes to the fact tables and aggregate fact tables can eat up many times more space than the raw data. If you are using multidimensional databases, be aware that certain products pre-calculate and store summarised data. As with star/snowflake schemas, storage of this calculated data can eat up far more storage than the raw data.

### **The time it takes to load the warehouse will expand to the amount of the time in the available window... and then some**

You'll need to understand the different approach to updating the warehouse. Before you decide that you can do complete refreshes, be aware that "There's all day Sunday to load the database!" have been famous last words of more than a handful of warehouse developers.

### **Assigning security cannot be done with a transaction processing system mind set**

At a recent conference we attended, a speaker discussed how data warehousing requires a philosophical shift from the "need to know" to the "right to know". So, if you are building a sales information system, if you are letting the Manchester manager for dog food see only his sales and are excluding him from seeing the Northern region sales for crusty rolls, think twice! There may be some information useful for dog food sales in that crusty roll information. By the way, we are not necessarily advocating wide open access. We are warning you that if you deal with security (and many organisations are, to great danger, avoiding it), you will have both a technical and philosophical challenge.

### **You are building a HIGH maintenance system**

Reorganisations, product introductions, new pricing schemes, new customers, changes in production systems, etc. are going to affect the warehouse. If the warehouse is going to stay 'current' (and being current will be a big selling point of the warehouse), changes to the warehouse have to be made fast.

### **You will fail if you concentrate on resource optimisation to the neglect of project, data, and customer management issues and an understanding of what adds value to the customer**

If you provide a system that is fast and technically elegant but adds little value or has suspect data, you will probably lose your customer from day one and will have a tough time getting him back. For the most part, use of data warehousing systems is optional. The customer has to want to use the system.

## **Phase 5: Production**

If you've done it right so far, your ecstatic end users will be demanding even more types of data and access to even more detail. The implementation cycle becomes iterative. As the system grows, you will be revisiting the ongoing management, administration, and support issues. Activities to help ensure continued growth and success include:

- Systems operation and administration
- Database administration
- End-user support
- Performance management

The companies that have followed this process have achieved a business solution that works efficiently, meets the end-users' requirements, and proves its worth in a short period of time.

## **The Issues**

Here we present some of the issues that you may face when your systems are "in production", as if these systems ever achieve the stability implied by that term. How you will deal with the issues will depend on your environment. This list is presented because, just as mentioned in my gotchas page, forewarned is forearmed!

### **You will be challenged to learn about business and feeder system changes that will affect the Data Warehousing and Decision Support Systems (DW/DSS) systems**

You, as the system developer would like to know of developments that will affect the DW/DSS systems in time to allow adequate time to assess what is impacted, make changes, test changes, etc. Of course this is no new concern to anyone doing systems maintenance. If you are responsible for a system being fed from, say, 10 sources, you may have much more exposure than you have with the typical transaction processing system. And though intelligent use of the data extraction, cleaning, and loading tools and the information catalogues can greatly ease the burden here, many changes will require a fair amount of effort. By the way, keeping informed and assessing the impact of technically driven changes to the feeder systems may be more difficult than keeping track of the business driven changes. If your IS organisation has change control meetings, it is a major mistake for a DW/DSS developer not to attend those meetings regularly.

### **You will have to figure out if, when, and how to purge data**

There comes a point when it does not make business sense to hold certain data in the warehousing system. This usually comes sooner than you expect. Either you are at some type of capacity limit or more likely, you are restructuring data and it is not worth the effort to restructure certain data. Before you get into a discussion about purging data, one piece of advice is to learn about less expensive, alternative means of storage.

### **You will have to determine which queries and reports should be IS written and which should be user written**

Probably when you got started into this area you had an idea about who would be doing what. And if you are like most DW/DSS developers, after you have been in production a while you have seen how reality has differed from your expectations. A very common IS expectation is that the end users will take over the overwhelming majority of query and report writing duties. And an all too common reality is that IS ends up taking over almost all the query and report writing or IS writes some semi-canned queries and the potential of the system for answering ad hoc questions never gets fully realised. You may have a challenge on two fronts. You may have to push the end users into "deep water". You may also have to convince your IS staff that the report and query building tools are not "toys".

### **You will find endless opportunities to tune DW/DSS system databases**

I once saw a quote from the director of IS of a well-known retailing business who said that the biggest data warehousing lesson he learned is "there aren't many data warehousing experts out there". If you are allowing a fair degree of end user developed access to systems and your systems are large and complex, you will discover that there are myriad ways to drag the systems down to a crawl. It is unlikely than an "expert" can foresee all the problems. And many of the problems are so crazy that the only way you are going to solve them is on a trial-and-error basis. By the way, you may have sold the DW concept as a way that "killer queries" will not drag down your "production" systems. Now that you've put in a data warehousing system, you will find out that the users are just as dependent on the data warehousing systems for recurring needs as they are on the so-called production systems and killer queries hurt wherever they occur.

### **You will have to balance the need for building aggregate structures for processing efficiency with the desire not to build a maintenance nightmare**

Many DW/DSS systems involve building structures to contain aggregated information. These "structures" can be many things - separate tables in relational systems, dimensions in the

OLAP world, etc. Anyway, after a while you will see countless ways to add or refine these aggregate structures usually in the name of reducing end user retrieval time. The issue you face is balancing your desire to speed things up with the need to be careful with how much a maintenance burden you want to take on. There are two aspects of this burden. First, you have to consider developer time. Secondly, you have to consider the amount of time it takes to update your systems on a recurring basis.

**You will be pressured to:**

- 1. Use your DW/DSS system as a source for transaction processing control reports**
- 2. Provide a means to interactively “correct” DW/DSS system data**

This happens once it is found that it is easier to do these things in the DW/DSS system.

**You will be uncertain as to what tools should be used for an application**

DW/DSS systems present IS with yet another set of tools with overlapping uses. You will find that it is not clear what is the best tool for many applications. For instance, if you have invested in relational and multidimensional database technology, you will find that for many applications, at a technical level, it is a toss-up as to which database technology will do the job better. Many organisations also have a heavy duty tool and a more lightweight tool that have similar ends. You will come across many situations where it is not clear whether to go heavy duty or lightweight.

**You will have to figure out how to test the effect of structure changes on end user written queries and reports**

After a while you are going to make some database structure changes that may affect the reports and queries that your end users have written. In order that the need to re-test their work does not come as too bad a surprise to your end users, may we suggest that you get them into good housekeeping habits early on. This means, for example, not keeping their work in 10 different directories and storing descriptions of their work.

**You will have to determine how problems with feeder system update processing affect DW/DSS system update processing**

Again, if you have 10 systems feeding your data warehouse, you are going to have to develop an appreciation of what to do when there is a processing problem with one or several of those feeder systems. At the simplest level, this means determining if and when you will process updates to the data warehousing system. At a more difficult level, this means determining if and how to process partial updates to the warehousing system. The dependencies in DW/DSS update processing can get quite complex. Do take the time to understand these dependencies especially if you do not have the most well-behaved feeder systems.

**You will have to keep reconciling feeder systems with the DW/DSS systems**

After things are going smoothly for a while, sometimes there is a tendency to be slack in whatever process you have implemented to reconcile systems. Also, if you have end users reconcile information, you may find that it is an ongoing discussion as to how to handle responsibility for regular reconciliation.

**You will have to perform euthanasia on some DW/DSS systems**

DW/DSS systems tend to be changed frequently. They experience entropy much more quickly than, say, general ledger systems. If your firm is used to keeping and patching a system for as long as you keep a refrigerator (and these days there are firms like that dipping their feet in DW/DSS for the first time), you may be in for a surprise.

## **Phase 6: Before Web-Enabling Your Warehouse**

Assess the success of your existing warehouse effort.

Are people actually using the warehouse? More or less than anticipated? If less, what seems to be the problems? (Ask the users.) If more popular than expected, how are you handling growth?

Define your target users. Who are they? Are they local or remote? Fixed or mobile? What types of hardware and software do they use? Do you have a waiting list of would-be users, or are the “target” users dragging their feet—and not their mice—to use the system?

Explore what other kinds of users—both internal and external—could benefit from access. (Note: This may not be obvious.)

Divide current and potential users into “classes,” according to the level of functionality they will require from access tools. Are current users making full use of the client/server access tools provided? Is your IT staff still getting numerous requests to run reports or do ad-hoc querying? Is your budget strained by managing the costs of installing access software on so many client machines?

Begin pondering the security ramifications of opening up access. Examine whether the data not only is sensitive as it stands but could also be combined with information available elsewhere—internally or externally—to derive proprietary knowledge about your operations.

Analyse the organisational implications of providing universal access. Do you have a corporate culture that shies away from openness? Management that protects its business processes from scrutiny? Departments or teams of workers who would object to “outsiders” peeking into their operations?

Check to see whether your vendors have Web-enabled features available or under development. Does your vendor have enough experience to help you through a pilot project? Can your vendor provide beta or even alpha code so you can test the waters? If not, are there third-party tools that can help you quickly generate Web functionality, given your architecture and configuration?

Assess additional investments needed. If you already have your warehouse up and running and a fairly reliable network in place, your main costs will come by way of increased server capacity and the programmer time needed to build the necessary Web pages through which users can query the warehouse. These costs can be offset by possible savings in client software, depending on user needs. And the improvements in customer service or reduced paper printing and distribution activities can also be used to offset any additional server or development costs.

## **Phase 7: Deploying to the Web**

### **Guide for Enterprise Web Applications**

The World Wide Web is a compelling platform for the delivery and dissemination of data-driven, interactive enterprise applications. Application development for the web has consistently been a challenge, in both the development of applications and the delivery of these solutions to the Enterprise. Corporations looking to leverage the Web as a strategic platform for implementing innovative business solutions - in effect becoming Web-centric enterprises - need to examine the issues surrounding the development of Web applications. This document lists and describes some of the major features that a web application development environment should comprise. This information is obtained through the use of many standard analyst organisations (Gartner **Group**, Meta **Group**, Forrester Research , Hurwitz **Group**, and Giga Information **Group**). This information is ideal when comprising requirements definitions for software that will enable your Corporation to web-enable its Enterprise.

- Application Server Architecture
- Data Driven
- Enterprise Class Integration
- Rapid Application Development Environment
- Open, Standards-driven
- Company Requirements

### **Application Server Architecture**

The broad reach and browser-based simplicity of the Web eliminates the time and cost associated with application deployment. The client browser performs the display of

information while all processing intensive business logic is handled on the server. Users of these web applications are connected to the data source for only the duration of each transaction rather than for the entire session. Optimizations in the architecture makes database connections readily available to users without incurring the overhead of a persistent and dedicated data source connection. The Application Server provides a high speed and scaleable three-tier architecture for web applications. Application servers with distributed architectures are critical for enterprise applications.

Features to look for in an Application Server Architecture

- N-tier Architecture
- Distributed across CPU's
- Automatic Load Balancing
- Broker Requests vs. Dedicated Processes Persistence Engine Security Support (SSL, Database, etc.)

## **Data Driven**

There is a significant difference between a simple application for the Web and a commercial business application. The difference lies in the data driven nature of commercial business applications. Web applications for the enterprise are data driven and the application development environment should be data driven, as well, not page driven. The delivery of Web applications should be optimised for a Web environment by caching persistent database connections, multiplexing users across shared database connections, and providing the fastest, full-featured connectivity available for databases. Data driven applications also need transaction management features to control transactions beyond the standard SQL actions.

Features to look for in a Data Driven development environment:

- Native Database Drivers
- Persistent Database Connections
- Multiplexed Database Connections
- Automatic Load Balancing
- Transaction Management
- Stored Procedures
- Multiple Data Sources support
- DB/2 Certified

## **Enterprise Class Integration**

The Web-centric enterprise is one that takes full advantage of the Web and leverages existing core applications at the same time. Web application development environments must integrate with core enterprise applications and services to truly extend the reach of these applications and servers to the enterprise. This includes leveraging the existing application logic in these applications and services, creating a new interface - a web-centric interface - to a familiar application.

Features to look for in Enterprise Class Integration:

- PeopleSoft Applications API
- SAP R/3 API
- Web Site Management (Wallop)
- Enterprise Security, DCE (Gradient)
- Entity Relationship Modeling (ERwin)
- System Environment Manager
- Convergence Applications
- CORBA Integration

## **Rapid Application Development (RAD) Environment**

In the early phase of the Web, applications were developed from scratch in either C/C++ or Perl due to the lack of mature tools for building commercial applications. Rapid Application

Development features cut time in development and subsequent code maintenance. This is an important feature in the rapid pace of application development for the Web.

Features to look for in a Rapid Application Development Environment:

- Generates code (Java)
- Generates HTML
- Generates SQL
- Allows Editing of code
- Wizards
- Automated Session/State

### **Open, Standards-based**

An open architecture is crucial for the success of web applications. The Web is based on standard technologies, such as HTML and Java, that allow for universal access across all platforms. The client/server tools used before were limited by proprietary languages, platform limitations, and closed environments. The architecture for developing web applications should allow developers to integrate any database, HTML editor, web server, browser display features (HTML, Java, JavaScript, ActiveX, VRML, etc.), and any web browser across multiple platforms.

Features to look for in an Open, Standards-based environment:

- Java Programming
- Thin (HTML) client
- Netscape ONE
- Any Client Operating System
- Any Web Server
- Heterogeneous Servers (NT, Unix)
- Client-side Processing Integration

### **Company Requirements**

The company behind any leading software must be solid, proven, and have a business model that creates a successful solution for the customer. The company should believe in its partner program and fully support it to provide the best end-to-end solution for the customer. A single company that attempts to provide all product and services often lacks direction and focus on its core values. A proven customer base of live applications and success stories gives further confidence in the ability to deliver the Total Solution for Web-Enabling the Enterprise.

Features to look for in a First Class software company:

- Strong Channel Model
- Independent Software Company
- Deployed Customer Base
- Complete Solution
- Experienced/Complete Management Team