CHAPTER 1

What Is a Data Warehouse?

Congratulations—you've joined a team either building or about to build a data warehouse. Do you really know what you've gotten yourself into? This may seem like a stupid question, but I've found that what people call a data warehouse varies significantly. In fact so much so, that I treat the term "data warehouse" with deep suspicion. I apologize for being so skeptical, but I've found that over 90% of what people call a data warehouse is open for debate! How do you tell someone his or her data warehouse is not really one without starting a fight?

A few years ago, there was no such thing as data warehousing. Now we hear about data warehouses everywhere and everyone seems to be building them. Success stories abound in technical and business journals. Many database conferences now have a data warehousing track or special interest group (SIG). Moreover, businesspeople have bought into them "hook, line, and sinker." They all want data warehouses and data marts. Now, they even want them via the Web! These are most often referred to as Web houses. That's the good news—there's plenty of demand.

But, demand for something by itself is not sufficient justification. For example, I would like to retire from the workforce right now. But as my wife kindly reminds me, it does not make sense given our financial reserves. Far too often, I've seen data warehouses being built for all the wrong reasons:

- Businesspeople ask for one since it's in vogue to have one.
- The chief information officer (CIO) decides to sponsor a data warehousing project initiative.
- Information Systems (IS) management submits a data warehousing proposal for funding.
- IS management combines several reporting systems into a warehouse.
- IS management renames an existing reporting system a data warehouse.

The point is that a true data warehouse should solve a genuine business need and thus be sponsored by the businesspeople who will benefit from it. Moreover, a true data warehouse follows some very specific design guidelines we'll be discussing in this book. Something is not a data warehouse simply because someone wants it to be or says it is.

Why am I making such a fuss over this? It's actually quite simple. The techniques espoused in this book will only work for genuine data warehouses. These exact same techniques will either not work or actually make things worse for entities that are not data warehouses. As such, this chapter is actually quite critical in terms of your data warehouse's success.

THE NATURE OF THE BEAST

So just how do you decide if you're working on a true data warehouse? First, examine the intended nature of your database and the application it supports. For each subject area in your data warehouse, simply ask your sponsoring business user to provide the following eight items:

- Mission statement
- Number of ad-hoc query users
- Number ad-hoc queries per day per ad-hoc user
- Number of pre-canned report users
- Number of pre-canned reports per day per pre-canned user
- Number of pre-canned reports

- Amount of history to keep in months, quarters, or years
- Typical daily, weekly, or monthly volume of data to record

These answers should help you categorize your database application into one of the following choices:

- Online transaction processing (OLTP)
- Operational data store (ODS)
- Online analytical processing (OLAP)
- Data mart/data warehouse (DM/DW)

Use the criteria outlined in Table 1–1 to make your distinction.

Table 1–1 General Database Application Categorizations

	OLTP	*ODS*	*OLAP*	*DM / DW*
Business Focus	Operational	Operational / Tactical	Tactical	Tactical / Strategic
End User Tools	Client/Server or Web	Client/Server or Web	Client/Server	Client/Server or Web
DB Technology	Relational	Relational	Cubic	Relational
Transaction Count	Large	Medium	Small	Small
Transaction Size	Small	Medium	Medium	Large
Transaction Time	Short	Medium	Medium	Long
DB Size in GB	10–400	100–800	100–800	800—80,000
Data Modeling	Traditional ERD	Traditional ERD	N/A	Dimensional
Normalization	3–5 NF[1]	3 NF	N/A	0 NF

1.Normal Form

For example, suppose your answers are as follows:

- "The point of sale (POS) subject area of the data warehouse should enable executives and senior sales managers to perform predictive, "what-if" sales analysis and historical analysis of:
  - A sales campaign's effectiveness
  - Geographic sales patterns

  - Calendar sales patterns
  - The effects of weather on sales
- 20 ad-hoc query users
- 10–20 ad-hoc queries a day per ad-hoc user
- 40 pre-canned report users
- 1–4 pre-canned reports a day per pre-canned user
- 60 months of history
- 40 million sales transactions per day

From this example, we can discern that we genuinely have a candidate for a data mart or data warehouse. First, the mission statement clearly indicates that our users' requirements are of a more tactical or strategic nature. Second, the majority of our report executions will clearly be ad-hoc (200–400 ad-hoc versus a maximum of 160 pre-canned). Third, we have significant historical data requirements and large amounts of raw data—and thus a potentially very large database (especially once we consider aggregates as well).

While it may seem like I've painted an example tailored to the conclusion, I've actually found the process to be this straightforward and easy in most cases. Unfortunately, these days, people tend to call any reporting database a data warehouse. It's okay for people to call their projects whatever they like, but as I pointed out, the techniques in this book only apply to the DM/DW column of Table 1–1.

DATA WAREHOUSE VS. BIG DATABASE

One of the key mistakes people make is labeling their database as a data warehouse solely based on its size. Over the past decade, three phenomena have occurred resulting in major increases in average database size:

- The cost of space versus the value of the data has decreased.
- Companies now value the data as a critical business asset.
- Companies have merged into large multi-national entities.

In other words, the cost of keeping data online is cheap, the perceived value of that data is now very high, and the size of companies

and their data needs have grown. As such, many of today's OLTP and ODS databases routinely grow into the 100–800 gigabyte (GB) range. But that does not make them data warehouses. For example, SAP and PeopleSoft enterprise resource planning (ERP) databases of 400 GB or more are not uncommon, yet they are not data warehouses, even at these extremely large sizes. Remember, size alone does not a data warehouse make.

The simplest way to avoid labeling a large database as a data warehouse is to add some DBA-centric questions and answers to the description of the nature of that database. For each subject area in your data warehouse, simply ask the physical DBA to provide estimates for the following seven items:

- The number of tables
- Average big table row count
- Average big table size in GB
- Largest table's row count
- Largest table's size in GB
- Largest transaction rollback needed in GB
- Largest temporary segment needed in GB

Data warehouses generally have fewer, larger tables, whereas non-data warehouse databases usually possess more, smaller tables. Of additional interest are the temporary and rollback segment needs of the database. Data warehouses tend to need them as large as the largest object (for rebuilds), whereas non-data warehouse databases only need them large enough for the largest transaction.

Use the criteria outlined in Table 1–2 for your evaluation.

Table 1–2 General Database Application Characteristics

	OLTP	*ODS*	*OLAP*	*DM / DW*
Number of Tables	100–1000's	100–1000's	10–100's	10–100's
Average Table's Row Count	10's of Thousands	10's of Thousands	10–100's of Millions	100–1000's of Millions
Average Table's Size in GB	10's of MB	10's of MB	10's of GB	10–100's of GB

Table 1–2 General Database Application Characteristics (Continued)

	OLTP	*ODS*	*OLAP*	*DM / DW*
Largest Table's Row Count	10–100's of Millions	10–100's of Millions	10–100's of Millions	100–10,000's of Millions
Largest Table's Size in GB	10's of GB	10's of GB	10's of GB	10–100's of GB
Rollback Segment's Size in GB	100's of MB	100's of MB	N/A	10–100's of GB
Temp Segment's Size in GB	100's of MB	100's of MB	N/A	10–100's of GB

Continuing with our previous example, suppose your requirements are as follows:

- 8 tables
- 500 million rows per big table
- 50 GB per big table
- 2 billion rows for largest table
- 160 GB for largest table
- 160 GB to rebuild largest table
- 60 GB to rebuild largest index

From this example, we can again discern that we have a data mart or data warehouse. First, we have very few tables. A typical OLTP or ERP database would have hundreds or even thousands of tables. Second, the row counts of our smallest big table and largest table have the right order of magnitude. Row counts expressed with lots of zeros or in powers of ten greater than ten (e.g., 10^{10}) are more likely to be in data warehouses. Finally, look at our rollback and temporary segments' needs. They're as big as some entire databases!

While it may seem like I've once again painted an example tailored to the conclusion, I've actually found the process to be this straightforward and easy in most cases as well. Unfortunately, these days, people tend to call any very large database a data warehouse. Once again, it's okay for people to call their projects whatever they like. But as pointed out, the techniques in this book only apply to the DM/DW column of Table 1–2.

OPERATIONAL DATA STORES DON'T COUNT

Frequently people don't understand why an ODS is not a data warehouse. Since many ODS projects are referred to as data warehousing initiatives, people often mistakenly assume that an ODS is therefore a data warehouse. That assumption is false as an ODS is merely a stepping-stone to a true data warehouse. An ODS is simply a means to an end, and not the end itself. Let's see where the ODS fits into the data warehousing equation.

Companies generally have numerous legacy application systems that were developed with varying technologies over a long period of time. For example, an insurance company may have different policy and commission applications across its different business units (e.g., life, health, property, casualty, and investments). It also would not be uncommon to have several such applications for the various product families within each different business unit (e.g., for investments, IRAs vs. annuities vs. 401Ks vs. 403Bs). Moreover, there could even be different applications by product nature (e.g., individual vs. group policies). So, an insurance company could have dozens of policy and commission applications across many different hardware and software platforms. Furthermore, these applications were very likely developed in total seclusion from the others. Thus, each application is really like an island unto itself (often referred to as stovepipe applications).

Now, imagine that you need to generate reports for a specific customer or agent, John Smith. Since John Smith the customer or agent might exist in one or more of those different applications, the insurance company needs a common staging area to merge this eclectic data into one centralized source. Such a centralized collection of disparate but interrelated data sources is known as an ODS. Figure 1–1 demonstrates a typical OD.

An ODS contains the centralized, single-source location for OLTP data. It is very often referred to as the system of record. Moreover, an ODS typically keeps a window of history on that data (usually by merely adding date and timestamp columns to the OLTP data). So, an ODS can be quite large, often into the 400+ GB range. But, ODS data is in its most raw form, sometimes nothing more than

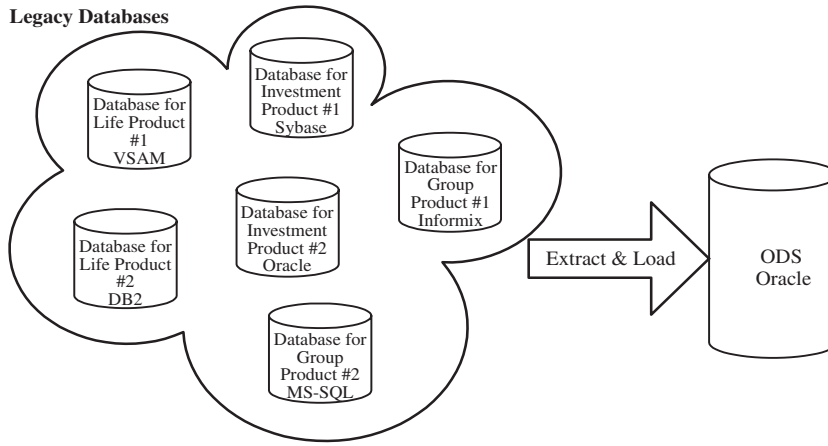

Figure 1–1 Typical ODS Source and Target Architecture

a copy of OLTP data with dates and timestamps. No useful transformations or aggregations have been performed to translate that transactional data into the tactical or strategic format necessary for executive management reporting needs. Therefore, to repetitively report off that ODS data in its unprocessed form would be very expensive. Thus, ODS data needs to be transformed into a format suitable for effective and efficient reporting. This pathway for loading a data warehouse via an ODS is shown in the highlighted portion of Figure 1–2.

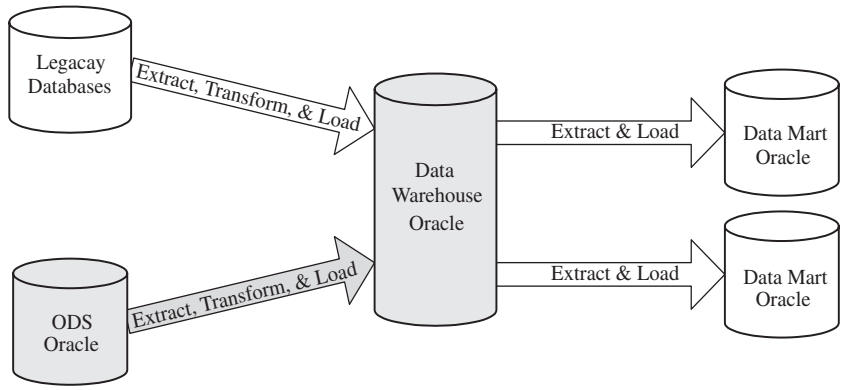

Figure 1–2 Typical Data Warehouse Data Loading Options

Also note that Figure 1–2 shows that you can just as easily bypass the ODS and directly transform legacy database data into the data warehouse. The point is that an ODS is not mandatory. For example, let's assume that we have a number of legacy application databases that were all developed in Oracle. Furthermore, let's assume that we have an accurate data dictionary for all business attributes such that all like tables and columns across those different Oracle databases have exactly the same type and size. In this case, building an ODS would merely serve to remove duplicate rows. In such a case, we might reasonably forgo building an ODS.

Figure 1–2 also shows that the data warehouse resides separately from the data marts. The point is that a data warehouse and a data mart are not quite the same thing. The primary difference between a data mart and a data warehouse is simply a question of scope. A data warehouse is a single, large store for the transformation of all legacy databases or ODS data. So, everyone would report off an enterprise data warehouse. A data mart is a smaller, specialized store for the transformation of all related legacy databases and ODS data, generally referred to as a subject area. For example, a consumer retail company might keep a data mart of cash register or POS data. A typical company might then have several to several dozen such data marts.

EXECUTIVE INFORMATION SYSTEMS DON'T COUNT

Our original question was: What is a data warehouse? As we've discovered, it's a large, centralized, specialized database for doing data management and executive reporting. In the old days, we just called such databases executive information systems (EISs). A logical question is then: How is a data warehouse different from an EIS? While it may not be readily apparent, there are some key differences.

The primary difference is the intended audience. EISs were built just to support making tactical decisions, meaning they were used by mid-level management. But, an effective data warehouse will support both mid-level and true executive management for both tactical and strategic decisions. A data warehouse contains the data necessary to make decisions such as "Should we even be in this business?" and "Is the return on investment (ROI) of the current business the best we can

do, or is there another business whose opportunity cost makes it worth considering?"

Another key difference is the method used to obtain that information. EISs generally provided mostly canned reports, with limited user-driven query capabilities. As such, tuning an EIS database was generally very straightforward. A data warehouse, on the other hand, possesses fewer canned reports—reports are mostly used for tactical decision-making. These strategic decisions require much more business-savvy user interaction. The user typically poses what-if scenarios to drill down to a conclusion. As such, tuning a data warehouse is a monumental challenge. The DBA must find a structure conducive to any number of unknown and often nightmarish queries.

By far, the biggest difference is the sheer magnitude in size difference between an EIS and data warehouse. The EIS databases preceded today's cheap hardware, so they tended to be on the same size scale as the OTLP systems from which they were derived. This indeed is quite important, because tuning a billion-row, multi-gigabyte table is a big challenge, even with today's super-fast hardware. In fact, yesteryears' database systems could not handle databases of this magnitude, let alone optimize queries against them.

So, data warehousing has genuinely become a market niche for any DBA. But, there is a price to be paid by DBAs making this switch, as they will find their OLTP skills and instincts will quickly erode. More importantly, other DBAs will find the data warehousing DBA to appear arrogant at times. Because, after dealing with billions of rows and hundreds of gigabytes to terabytes, how does one get excited about typical OLTP sizes? It's actually quite fun to sound like Carl Sagan and state: "My average table has billions and billions of rows…"

Warehouses Evolve without Phases

The typical database application development life cycle is something like the following:

- Deliver a version.
- Begin work on the next version.

- Perform maintenance on the current version.
- Promote changes or deltas to the current version.
- Incorporate changes or deltas into the new version.
- Repeat the process.

For EDS and 7-Eleven, we had to have a customer signature and scheduled downtime to promote a database application change for any OLTP system. This practice makes good business sense. When OLTP systems run the customer's business, you don't want to make unapproved or unscheduled changes that could result in customer OLTP application downtime, because such downtime could cost the customer real money.

In data warehousing, things are very different. There really is no database application as the database itself is the object of desire from the customer's viewpoint. The data warehouse may be queried by end-user tools and have batch programs for loading, but the database itself is really the heart and soul of the data warehouse. Customers see its information at their disposal as the real deliverable. Or, as I sometimes like to say, "It's the database, Stupid."

As users mine the data warehouse to answer new and more involved business questions, they quite often and regularly find something lacking. The most common requests are often to add a new column to a table or create a new summarization or aggregate table that does not exist. The first solves a missing data problem and the second reduces report runtimes. In addition, users often ask for columns to be displayed differently or contain additional data. The point is that change requests come in daily, from mid-level managers to true executives.

So, the data warehousing application development lifecycle looks more like:

- Deliver the first version.
- Promote changes or deltas to the current version.
- Repeat the process.

This evolutionary method actually requires a much more cautious approach to promoting changes. The batch load programmers, the DBA, and the project manager must all be 100% in sync with each other at all times because there is no real version control of the code

or database data definition language (DDL) to fall back on. Data warehouse changes occur with too much frequency and urgency to follow a strict development methodology. From the OLTP perspective, the data warehouse team appears to fly by the seat of their pants. So, a great project manager, a detail-oriented project lead, and a very experienced DBA are needed to make this process work.

THE WAREHOUSE ROLLER COASTER

Finally, I want to remind the reader of the enormous challenges for any data warehousing DBA. I often reminisce about the past decade and feel that Dickens' "It was the best of times, it was the worst of times" best describes my data warehousing experiences. Be prepared as a data warehousing DBA to experience little joy from few wins and a lot of agony from numerous defeats. With so few Golden Rules and fellow data warehousing DBAs in existence, expect more of the latter. But remember that if you don't succeed at first, try, try again. It's taken me nearly 20 years of working with Oracle and 10 years of data warehousing experience to feel like anything more than a base novice. There is no shame in making mistakes in data warehousing. In fact, it's the only proven method to finding the best solutions. Or, as Babe Ruth once said, "Every strike brings me closer to the next home run."