

Building Efficient Data Warehouses: Understanding the Issues of Data Summarization and Partitioning

Susan Tideman, SAS Institute Inc.

Robert Chu, SAS Institute Inc.

ABSTRACT

An efficient data warehouse enables better and faster decisions. To build an efficient data warehouse, you need to know how best to summarize data and how best to partition the summarized results.

By mastering the fundamental issues discussed in this paper, you will increase the return on your organization's investment in data warehouses, such as ones that you can build with the about-to-be-released SAS data warehouse software.

INTRODUCTION

First, this paper provides some background on:

- ❑ data warehouse subjects
- ❑ what a data warehouse contains about subjects
- ❑ how a data warehouse is different from an operational database
- ❑ the steps for building and using a data warehouse.

Next, this paper discusses the following steps in more detail:

- ❑ data summarization (logical aspects of data organization)
- ❑ data partitioning (physical aspects of data organization).

Finally, this paper gives you some pointers on:

- ❑ other issues related to data warehouses.

BACKGROUND

What is A Data Warehouse Subject?

A data warehouse typically contains multiple *subjects*.

A *subject* is a logical grouping of information that serves at least one business purpose.

- ❑ For instance, the following could be the subjects in a finance data warehouse: invoices, customer profiles, vendor profiles, employees, benefits, pensions, short-term debt, long-term debt, and so on.
- ❑ And the following could be the subjects in a sales and marketing data warehouse: competitors, customers, products, staff, new leads, sales, return on product development investments, and so on.
- ❑ Or a single, organization-wide warehouse could contain all of these subjects.

Subjects do not necessarily have a one-to-one correspondence to data sources.

- ❑ One data source can provide data to multiple subjects. For instance, the on-line transaction processing (OLTP) data source on product purchases could contribute to subjects such as sales, customers, return on product development investment, and so on.
- ❑ One subject can receive data from multiple data sources. The sales subject could receive data from multiple OLTP data

sources such as product purchases, training purchases, consulting purchases, and so on.

What Does A Data Warehouse Contain About A Subject?

For a given subject, a data warehouse may contain

- ❑ detail data (for example, transaction-oriented data)
- ❑ summary statistics (sometimes called summary data)
- ❑ data for subsetting (selecting), mapping (looking up one value to find another value), ordering (sorting) the detail and summary data, and data for weighting the summary data. For an example of subsetting, mapping, and ordering data, see Figure 1 and the notes associated with Figure 1.
- ❑ metadata (such as: data on data sources; data type, format, and length of detail variables; the summary period for summary statistics; the criteria to use for subsetting, mapping, ordering, and weighting; the age limits for keeping data in the data warehouse; data on data archives; and so on)
- ❑ lookup data (a special kind of detail data, often dated, that provides descriptive data). For example, customer information could contain customer id, date, company name, industry, parent corporation, subsidiaries, contact people, phone numbers, names, addresses, and so on).

Let's take a simple example. Suppose each detail record from the sales subject contains data on the sale date, sales person, customer id, product, price, and quantity.

- ❑ A sales analyst could need to see the total sales (sum of prices) by product, and within product, by month, and within month, by region. In this case, the summary statistics are sums.
- ❑ A marketing analyst could need to see the count of sales by product, and within product by industry, and within industry by sales person, and within sales person by month. In this case, the summary statistics are counts.
- ❑ The analysts are likely to
 - ❑ first, look at the summary statistics
 - ❑ second, decide what, if anything, needs to be further researched
 - ❑ third, drill down as far as the detail data or even as far as the OLTP data source, if necessary.

Typically the data are available as follows:

- ❑ The records or observations from data sources typically provide the detail data and the data for subsetting, mapping, ordering, and weighting.

Unless the data are old enough to have been aged out of the data warehouse into archives, the data are already available when needed. If the data have been aged out into archives, the data may be available with a delay of some sort.

- ❑ The data warehouse typically provides summary statistics. *Summary statistics* are statistics such as mean, median, mode, sum, count, and so on. The process of calculating summary statistics is called *data summarization*.

Summary statistics that have been requested in advance are automatically calculated by the data warehouse and thus are

already available when needed. Summary statistics that have not been requested in advance are calculated on an ad hoc basis and thus available with a delay of some sort.

How is A Data Warehouse Different from an Operational Database?

In case the term *data warehouse* is not entirely familiar to you, let's pause a moment and compare it with the term *operational database*.

A data warehouse is a special type of database. The following characteristics help distinguish a typical data warehouse (DW) from a typical operational database (ODB):

- ❑ Purpose
 - ❑ DW: Decision making
 - ❑ ODB: Routine business operations
- ❑ Number
 - ❑ DW: Typically one main DW and one test DW per organization, division, or department
 - ❑ ODB: Typically one ODB per application, thus many ODBs per organization, division, or department
- ❑ Typical User
 - ❑ DW: Business analyst
 - ❑ ODB: Clerical worker
- ❑ Structure
 - ❑ DW: Usually organized by subject and time period; usually pre-summarized, pre-indexed, and so on, in ways that correspond to analysis purposes
 - ❑ ODB: Usually organized for use with a specific transaction application; usually pre-indexed in ways that correspond to operational purposes
- ❑ Number of Data Sources
 - ❑ DW: Many
 - ❑ ODB: Few
- ❑ Data Encoding
 - ❑ DW: Does recoding if necessary for data consistency
 - ❑ ODB: Uses original encoding
- ❑ Volatility
 - ❑ DW: predictable updates at predictable intervals; typically, weekly, daily, or perhaps hourly
 - ❑ ODB: unpredictable updates at unpredictable intervals; typically, minutes or seconds or less
- ❑ Data Access
 - ❑ DW: Typically read-only access for analysts
 - ❑ ODB: Typically read-only or update access for operational users
- ❑ Summary Statistics (Data Summarization)
 - ❑ DW: Typically already calculated but sometimes calculated on ad hoc basis; typically statistics cover current and long-term data (five to ten years)
 - ❑ ODB: Typically calculated on an ad hoc basis; typically statistics cover current data only (sixty to ninety days)
- ❑ Fate of old data
 - ❑ DW: Typically archived

- ❑ ODB: Typically deleted.

What Are the Steps for Building and Using a Data Warehouse?

Let's briefly walk through the steps for building and using a data warehouse. They are listed here in a typical order, but the order is not strict. Some steps can be done in parallel or in another order or iteratively.

The steps are as follows:

1. *Decide on the subjects that are appropriate for your data warehouse.*

The analysts who will use the data warehouse can tell you what subjects they need and the type of data they need about each subject.

This step may take a considerable amount of time if you need to look at the entire organization.

2. *Find out how the analysts need to have the data summarized.*

Each analyst may need different summary statistics on the same or different subjects and may need the data sorted by different criteria in different orders. Similarly, each analyst may need to subset or map the data on one or more subjects by different criteria.

For example, you may find that a purchasing analyst needs vendor data summarized by month, and within month by vendor name. And a sales manager needs customer data summarized by week, and within week by region. And a performance analyst needs inquiry data summarized by hour, and within hour by system. And the CEO and board need vendor and customer data summarized by country, and within country by fiscal year.

3. *Locate data sources for the subjects and for subsetting, mapping, ordering, and so on.*

A *data source* is a file, library, database management system (DBMS), and so on that contains some or all of the data about a subject.

For instance, your vendor database could provide data needed for the vendor subject. And a leased database on national corporations could provide data needed for the invoices subject and the new-leads subject.

4. *Choose a common encoding scheme and write programs that extract data from the data sources, transform (recode) the data if necessary, and load the data as detail data in the data warehouse.*

It is often the case that different data sources use different encoding schemes. A data warehouse uses a common encoding scheme.

For example, you may find that one data source uses sex codes of M and F, and another data source uses a sex codes of 0 and 1. Similarly, you may find that one data source abbreviates company names and another data source contains complete company names.

You can use products such as SAS/ACCESS (R), the SAS SQL procedure in base SAS software, and SAS/CONNECT (R) to extract, transform, and transport data.

5. *Plan how to partition (physically separate and store) the data.*

Although the data warehouse is one entity logically, the amount of data in it probably is much too much to store in a

single DBMS or on a single disk. It may even be too much to store on a single system.

For example, you may need the detail data on sales in an ORACLE (R) DBMS on a Unix machine, the detail data on finance in a SAS (R) data library on a mainframe, the summary data on sales on a server that you can access with a Macintosh, the summary data on finance in a multi-dimensional database (MDDb) that you can access with PCs, customer lookup data on central Unix server, and vendor lookup data on another central Unix server.

Even if you could store the data together, you would still want to partition the data for performance efficiency.

For example, you are likely to want to separate recent data from older data. And you are likely to want to separate often-requested data from infrequently-requested data.

8. *Create an empty data warehouse and define the required metadata.*

You will have metadata on the detail data, summary data, lookup data, and criteria for subsetting, mapping, ordering, and weighting. Similarly you will have metadata on partitioning.

For example, you will have metadata on data sources, data variables, data attributes, patterns of encoding, one analyst's criteria for a subject, another analyst's criteria for that subject, criteria for aging out detail data and aging out summary data, data archives, and so on.

7. *Create and schedule one or more periodic (weekly, daily, or in some cases perhaps hourly) jobs.*

Periodically, you need to load new data, incorporate the new data into the summary statistics, and backup and report on the data warehouse. You could have one long sequential job. Or more typically you would have a first set of concurrent jobs followed by a second set of concurrent jobs followed by a third set of concurrent jobs.

For example, you could have:

- ☐ a number of concurrent jobs that run your programs that extract, transform, and load the data as detail data in your data warehouse(s).
- ☐ a number of concurrent jobs that run the data warehouse programs that use the metadata to calculate or update the requested summary statistics.
- ☐ a number of concurrent jobs that back up the data in the data warehouse, generate reports on the data in the data warehouse, and, optionally, create one or more datamarts.

A datamart is a temporary collection (for example, for a week) of read-only data that is usually a subset of the summary statistics and a subset of data for subsetting, mapping, ordering, and weighting. The physical implementation of a datamart can vary. For instance, a datamart can be a library of SAS data sets or a multi-dimensional database (MDDb) such as the type created by the SAS MDDb procedure.

Remember to plan for recovery in case a job fails. For instance, a job may fail due to a power outage, lack of disk space, and so on. To be practical, make recovery procedures granular. For instance, recover by subject not by the entire data warehouse.

8. *Use existing interfaces or design new interfaces to view and/or generate reports on the data.*

The interfaces can range from simple data browsers to

executive information systems. You can use one interface for everything or different interfaces for different purposes.

For instance, you can use SAS reporting tools like SAS/ASSIST (R) and programs developed with SAS/EIS (R), ORACLE reporting tools like programs developed with PowerObject (R), and so on.

9. *Plan how to archive the data in a manner that conveniently allows you to access the archived data when necessary, and write the programs that archive and retrieve data.*

Typically, older data are migrated from on-line disks to off-line disks or to tapes. If possible, use a method that enables you to retrieve the archived data and to integrate the archived data and current data transparently.

For example, you may want daily tapes that contain the data aged out that day. You may want one tape per subject or one tape per some other meaningful subset of that day's data. Or you may want daily increments (or subsets of the data) on weekly or monthly tapes.

MORE ABOUT DATA SUMMARIZATION AND PARTITIONING

Data Summarization

You can write or find programs or procedures or tools (like the SAS SUMMARY procedure, the SAS SQL procedure, or the ORACLE SQL*Plus tool) that calculate summary statistics. However, the more data that you have in your warehouse, the longer it takes to run the programs and the more space they require for intermediate files. So the more detail data you have, the more carefully you need to plan

- ☐ what statistics the data warehouse will calculate automatically
- ☐ more importantly, how the data on which the statistics are calculated should be subset, mapped, ordered, and weighted
- ☐ even more importantly, how to calculate as many statistics as possible without reading through all of the detail data (especially the archived detail data) every time you do the calculations.

If appropriate provisions are made ahead of time, the analysts using the data warehouse can get answers quickly even when large quantities of detail data and summarized data are involved.

Here are examples of the types of summary statistics that analysts are likely to ask for:

- ☐ mean, median, and mode
- ☐ count and number missing
- ☐ sum and weighted sum
- ☐ standard deviation and variance
- ☐ maximum, minimum, and range
- ☐ percentile.

Note that the calculation of some of these depends on context (for examples, see the appendix of this paper).

Analysts often need to see the statistics only for a specified subset or mapping or order of the data. Here are examples of the ways that analysts are likely to need to see the data:

- ☐ by day, by week, by biweek, by month, by quarter, and/or by year
- ☐ by person, by department, by division, by company, by parent company, and/or by industry
- ☐ by city, by state, by region, and/or by country

- and combinations of the above.

Here is a checklist to get you started planning the calculation of summary statistics:

- The data warehouse must carry the data on which the statistics are calculated. Remember to have it also carry the data on which the statistics can be subset, mapped, ordered, and weighted.
 - Allow analysts to request variables whose values are calculated from or looked up from the data in the data warehouse. For example, suppose an analyst is looking into the need for more fax equipment and wants statistics on incoming and outgoing fax traffic per hour. You can calculate hour from the timestamp on the fax records. Similarly, suppose an analyst wants to analyze customers by industry and the data sources do not carry a variable for industry. You can have lookup data that allows you to retrieve the industries for customers and use the industries for subsetting, mapping, and so on.
 - Allow for variations on the definition of a subset or mapping. For instance, one analyst may need to see data by calendar month and another may need to see data by biweekly pay period. Similarly, one may need calendar years and another may need fiscal years. And one may need to see customer data by parent company while another needs to see customer data by industry.
 - Base your choice of statistics on what is most meaningful and/or what has been requested. For example, suppose a group handles telephone calls and is looking at the issue of staffing. An analyst working on the issue would probably be interested in the pattern of calls per hour rather than in the total count of calls. Similarly, suppose a group handles marketing and is looking at the issue of advertising. An analyst working on the issue would probably not be as interested in the monthly sales figures per state (say, \$1M in Vermont and \$2M in California) as in the monthly sales figures weighted by (in this case, divided by) state population.
 - Avoid overdoing the statistics you calculate. Offer to subset and map and order and weight, but avoid automatically calculating every combination and permutation. There are far too many to calculate statistics automatically for them all. The only ones worth calculating automatically are the ones that have been requested.
 - Avoid putting artificial limits on the subsetting, mapping, ordering, and weighting that people request. If two people need to use similar but not identical weighting, subsetting, etc., it is much better to give each analyst the variation requested than to get analysts to agree to share statistics that are not quite what they need to see. The time and cost to calculate another set of summary statistics is much less than the time and cost to try to interpret statistics that are not quite what is needed.
 - Look for a way to reuse the same detail data for different analysts. Avoid duplicating detail data just because different analysts need to calculate different summary statistics or need to subset, map, order or weight them differently.
 - Whenever possible use incremental calculations for summary statistics. For instance, instead of calculating the mean by running against all of the data, update a cumulative sum with the most recent day's (or hour's or week's) data and update a cumulative count with the most recent day's (or hour's or week's) data, and calculate the mean by dividing the cumulative sum by the cumulative count.
- unlikely to need the data frequently (or right away). Another way to say that is that incremental calculations enable you to have up-to-date summary statistics that include data that are no longer on-line.
- Look for a way to keep the summary statistics for different periods independent of each other. For instance, if you need the mean over a week and the mean over a month, keep a cumulative sum and count for the week and a cumulative sum and count for the month. By keeping the periods independent of each other, you allow more flexibility. For example, you can avoid
 - keeping daily statistics that no one has requested just because weekly statistics and monthly statistics have been requested, and so on.
 - keeping daily statistics for 31 days when analysts only need them for the last few days, just because you need to calculate weekly statistics and monthly statistics, and so on.
 - calculating daily statistics on one pass, and then using another pass to calculate weekly and monthly statistics, and then another pass to calculate biweekly and quarterly statistics, and so on.
 - Remember that you can backload previously-collected data. Suppose your periodic jobs load a day's worth of data each day. At that rate, it may be a long time before there is enough data in the data warehouse to analyze trends. You can backload earlier days, weeks, months, and even years of data into the data warehouse (and include that detail data in the summary statistics) if analysts need to do trend analysis now.
 - Consider some way to keep data on the usage of the detail data, summary data, and subsetting, mapping, ordering, and weighting data. Analysts' needs will change over time. For instance, analysts may be subsetting, mapping, ordering, and weighting data in ways not anticipated earlier, and thus submitting ad hoc requests (which return answers more slowly than requests for statistics that are already calculated). If you can detect such a change, you can make the corresponding changes to the metadata so that those statistics, subsets, etc., are included in the ones that are provided automatically.
- Or if you cannot keep data on usage, at least suggest to analysts that they review their usage patterns from time to time and let you know of changes. That way you can add and delete metadata to "tune" their response times to their actual usage.
- Remember to plan how you will migrate changes in your test data warehouse to your main data warehouse.

The items in this checklist are not exhaustive. They will get you off to a good start and are food for thought.

Data Partitioning

In principle, any kind of logical data (detail, summary, lookup, etc.) in the data warehouse can be stored as one or more partitions. (The term *table* is sometimes used to refer to a partition.)

Partitioning is usually done to decrease response time (by decreasing access time). You can partition data in a number of ways, even within the same subject. For instance:

- Platform partitions (platform tables) separate the data likely to be accessed from one platform from the data likely to be accessed from another platform.
- Column partitions (column tables) separate frequently-requested variables from infrequently-requested variables and/or frequently-requested statistics from

Avoiding the reading of old data not only saves time. It also means that the detail data can be archived (or deleted if the amount is too massive to archive) as soon as the analysts are

infrequently-requested statistics.

- ❑ Row partitions (row tables) separate data based on a range or, more generally, on an expression. For instance, you can separate the recent data from the older data, or the data for one state from the data from another state, or a combination of these.
- ❑ Format partitions (format tables) separate data based on the format used for storage. For example, you can have detail data within a subject stored in an ORACLE DBMS and summary data within that subject stored in a SAS data library or SAS MDDB.

Ideally, allow for all types of partitions.

Here is a checklist to get you started planning the data partitioning.

- ❑ Consider the locations of the analysts who are likely to use the data.

For instance, suppose you have Los Angeles (LA) data and New York (NY) data and the LA-based analysts need to use the LA data frequently and the NY data infrequently, and vice versa. Store the LA data in a partition in LA and the NY data in a partition in NY.

- ❑ Make the partitioning as transparent as possible to the analysts.

For instance, suppose an analyst requests data and some of the data are on-line and the rest have been archived. You could write a message to the analyst explaining that there will be a delay because such and such data are not on-line and then ask if the analyst wants to wait or to cancel. If the analyst chooses to wait, you need to have enough metadata about archived data that you can access the requested data without asking the analyst how and where old data are stored.

- ❑ Match the storage format to the analysts' needs as closely as possible.

For instance, if analysts have requested multi-dimensional analysis or you anticipate frequent ad hoc multi-dimensional analyses on certain data, consider storing that data in an MDDB. You can even separate data within a subject (for instance, to put some in a SAS library and some in a SAS MDDB). To be practical, physically keep together the data that analysts need to use together (for subsetting, mapping, ordering, and weighting).

- ❑ Consider the summary period (day, week, and so on).

To correspond to the flexibility in calculating daily, weekly, etc. statistics, maintain flexibility in storing those periods. The easiest way to do that is to store each period's statistics (and associated data for ordering and weighting) in a different partition.

- ❑ Consider some way to keep data on the usage of the partitions. Analysts may change how often they need certain data, what data they subset, what data they map together, and what periods they need summary statistics on. Some of the changes can be reflected by changing the metadata in the data warehouse. For example, if analysts often request certain recently archived data, you can change the metadata to keep those data on-line longer before archiving.

Other changes will require repartitioning. For example, if two tables on the same drive are popular, you can move one to another drive. And if analysts often request multi-dimensional analyses on data that are not in an MDDB, you may want to move those data to an MDDB.

- ❑ Consider how to make repartitioning easy.

Plan ahead of time how to merge and split data based on platform partitions, column partitions, row partitions, and format partitions. For instance, you yourself may want to repartition data after you have some experience with your initial partitions. And over time analysts may need to group together data that are currently in two different MDDBs, or may start doing frequent multidimensional analyses on data not currently in an MDDB, or start using a period for summary statistics that was not used before, and so on.

- ❑ Consider replicating (duplicating) popular tables. For instance, if fifty analysts use a certain table, you can decrease average response time by having two copies of the table.
- ❑ Remember to plan how you will migrate changes in your test data warehouse to your main data warehouse.

The items in this checklist are not exhaustive. They will get you off to a good start and are food for thought.

What is the optimal data partitioning in your data warehouse? Is there an optimal data partitioning scheme for your data warehouse? How do you measure whether you have the optimized data partitioning scheme deployed in your data warehouse?

All these questions have no definite answers. Tradeoffs are usually involved. There are really no simple guidelines except that business aspects should take precedence over technical aspects as much as possible.

POINTERS ON RELATED ISSUES

In-depth discussion of these issues is beyond the scope of this paper, but the issues are worth mentioning briefly.

Metadata and Associated Versioning Issues

Over time, analysts' requirements for the data will change, and the data in the data sources and the partitioning of the data will change. For example, records from a data source may carry a new variable. Or summary data may be relocated.

Remember to plan how the metadata will reflect the changes, so that summary calculations and ad hoc requests (especially calculations and requests that include data both before and after the change) do not cause any errors. For instance, you may want to have one set of metadata that applies before the change date and another set of metadata that applies after the change date.

To prevent inconsistencies between the data and metadata, also remember to plan how you will block access to the metadata during changes to the metadata. Or plan to use a product like SAS/SHARE (R) that enables updates to take place to one part of the data and metadata while analysts are querying another part of the data and metadata.

Lookup Data and Associated Versioning Issues

Over time, customers or vendors or staff or equipment and so on may move from one category to another. For example, a customer may move a plant from one region to another, so that data before the change should be reflected in the statistics for one region and data after the change should be reflected in the statistics for the other region.

Remember to plan how the lookup data will reflect the changes, so that summary calculations and ad hoc requests (especially calculations and requests that include data both before and after the change) do not cause any errors. For instance, you may want to have 'implicit' metadata in the sense of enabling the detection

of multiple lookup records with the same key but different dates.

To prevent inconsistencies between the data and lookup data, also remember to plan how you will block access to the lookup data during changes to the lookup data. Or plan to use a product like SAS/SHARE that enables updates to take place to one part of the data and metadata while analysts are querying another part of the data and metadata.

Data Compression

Usually data compression reduces the size of a partition (table), but not always. The direction and size of the effect depends on the nature of the data.

Usually data compression slows response time (which includes the time for decompression), but not always. Decreased I/O time may outweigh increased CPU time. Even if data compression slows response time, that may not matter if the analysts at your site tend to generate reports in batch rather than to make inquiries interactively.

You may want to experiment to see the effect of data compression and then consider the tradeoffs. Regardless of the results of the experiment, you may want to repeat the experiment from time to time in case the effect is different due to changes in the data, changes in the use of batch mode or interactive queries, and so on. Because of possible changes, it is best to have one or more switches that turn data compression on or off.

Indexes on the Data

Almost always a data warehouse has indexes. Typically the indexes on detail level are re-created after the periodic jobs run that load the detail data. Typically the indexes on summary level are re-created after the periodic jobs run that calculate summary statistics. Re-creating indexes takes production time and indexes take storage space, but indexes can save analysts' time (especially if analysts use the data warehouse in interactive mode).

Evaluate the tradeoffs and carefully select what to index. Later, you may want to use the usage data to verify that you chose the appropriate indexes.

Exception Reporting

An analysts may not need to check on certain statistics regularly but to be notified if the statistics change significantly. Similarly, an analyst may need to know when a statistic exceeds the value predicted based on previous growth, or is fluctuating rapidly, or behaving in an atypical way. Similarly, you may need to know when the response time starts to get long because of an increased analyst base or degraded access to some partition.

Consider exception reporting that sends e-mail, runs programs, and/or generates reports if the data warehouse detects data that are behaving in ways that you and/or the analysts can describe with rules. That should save everyone much time spent discovering that some of this day's, week's, and so on, data are quite similar to the previous ones.

Data Security

You may want to have security access based on whole data warehouses and datamarts, or whole subjects, or partial subjects. For example, you may need to limit access to employee salary data while allowing free access to employee telephone extension data. Similarly, you may need to limit access to privileged information from vendors or customers or marketing partners.

Data Mining

Data mining is a process that discovers information (usually from data warehouses or operational databases) that traditional query and reporting tools cannot effectively reveal. As data mining becomes more popular, anticipate more use of your data warehouse(s) and datamart(s) for data mining, because so much data are already gathered there, encoded consistently, and in an easily accessible form.

Note that data mining often involves statistical analyses, so the SAS System with all its statistical procedures can be quite useful.

CONCLUSION

Mastering the issues of data summarization and data partitioning is necessary to build efficient data warehouses.

You can use existing SAS procedures to build data warehouses. But the about-to-be-released SAS data warehouse software can help you automate many of the steps. Some of the topics discussed here are addressed in the initial release of the SAS data warehouse software. Most, if not all, of the remaining topics will be addressed in upcoming releases of the SAS data warehouse software.

APPENDIX ON SUMMARY STATISTICS CALCULATIONS

Suppose we have three observations:

date	stock_price_at_closing	rate_changed
02SEP95	\$30	
03SEP95	\$60	100%
04SEP95	\$30	-50%

Note that rate_changed doubles and then halves.

Without the Concept of Variable Interpretation

If an analyst requests averages, the averages would be

- for stock_price_at_closing, $(\$30 + \$60 + \$30)/3 = \40
- for rate_changed, $(100\% + (-50\%))/2 = 25\%$.

Note that the simple average (arithmetic average) is assumed for both numeric averages here. But the average of rate_changed (25%) is incorrect.

With the Concept of Variable Interpretation

If an analyst requests averages, the averages would be

- for stock_price_at_closing, interpreted as a 'regular' numeric, $(\$30 + \$60 + \$30)/3 = \40
- for rate_changed, interpreted as a 'delta rate' numeric, $\text{SQRT}((1+100\%)(1-50\%)) - 1 = 0\%$.

Note that the arithmetic average is assumed for the 'regular' numeric and the geometric average is assumed for the 'delta rate' numeric. The average of rate_changed (0%) is correct.

ACKNOWLEDGMENTS

We thank Terry Lewis and Tony Fisher for reviewing this paper.

SAS (R), SAS/ACCESS (R), SAS/ASSIST (R), SAS/CONNECT (R), SAS/EIS (R) and SAS/SHARE (R) are registered trademarks or trademarks of SAS Institute Inc in the USA and other countries. ORACLE (R), PowerObject (R), and SQL*Plus (R) are registered trademarks of the Oracle Corporation. (R) indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Robert Chu
SAS Institute Inc.
SAS Campus Drive, Cary, NC 27513
Phone (919) 677-8000
E-mail sasrcc@unx.sas.com

Susan Tideman
SAS Institute Inc.
SAS Campus Drive, Cary, NC 27513
Phone (919) 677-8000
E-mail sassct@unx.sas.com

NOTES FOR FIGURE 1

There are four hierarchical levels in this figure:

- ❑ Level 1 - Data Warehouse
 - ❑ Logically, a data warehouse is a container for subjects.
 - ❑ Physically, a data warehouse 'owns' metadata for level 1 (for example, the name of the data warehouse support person, the data warehouse description, the data warehouse creation date, and so on).
- ❑ Level 2 - Subjects
 - ❑ Logically, a subject is a container for summary groups.
 - ❑ Physically, a subject 'owns' metadata for level 2 (for example, the name and description of the subject, the names and descriptions and properties of the variables, the data sources and data archives for detail data, and so on) and 'owns' the detail data for the subject.
- ❑ Level 3 - Summary Groups
 - ❑ Logically, a summary group is a container for summary levels. Summary groups can have zero or more criteria for subsetting and zero or more criteria for mapping.
 - ❑ Physically, a summary group 'owns' metadata for level 3 (for example, the list of which periods are in use, the subsetting and mapping criteria, if any, and so on).
- ❑ Level 4 - Summary Levels
 - ❑ Logically, a summary level is a container for statistics and the associated data for ordering and weighting. (Ordering variables are often called class variables.)
 - ❑ Physically, a summary level 'owns' metadata for level 4 (for example, the list of ordering variables, the period, the list of statistics requested, weights to be used for those statistics, and so on) and 'owns' the statistics and copies of the associated data for ordering and weighting.

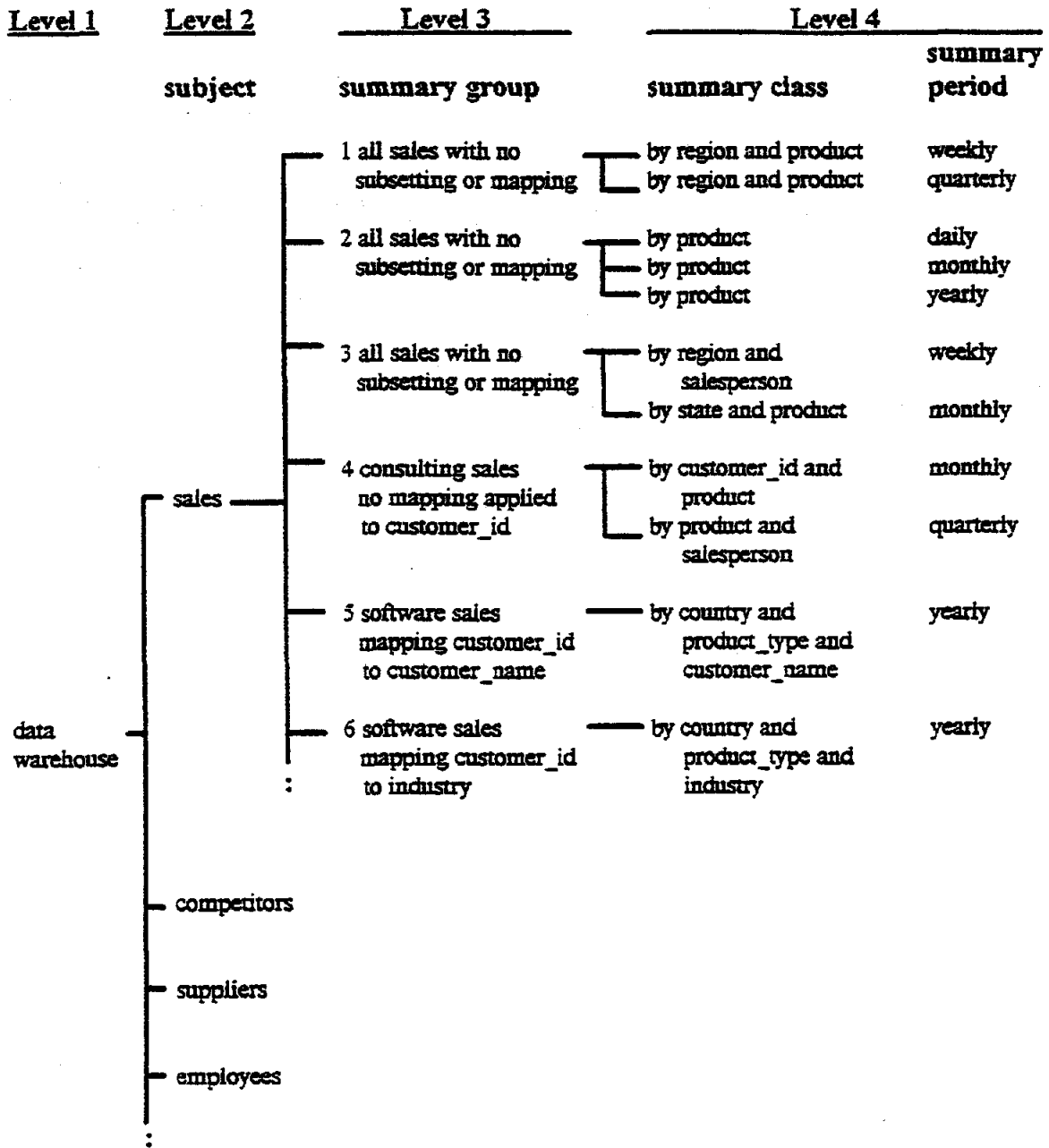


Figure 1: Example of a Logical Data Warehouse Structure