Paper 143-27

Data Warehousing for the Enterprise

Gary Mehler, SAS Institute Inc., Cary North Carolina

ABSTRACT

Data warehousing is performed by many organizations to provide the information to support their query and reporting as well as more advanced analytic requirements. While many small to mid-size implementations can accomplish this using standard data manipulation environments like base products of the SAS® System, larger organizations may find that long-term support and maintenance of complex projects, multi-user design and development requirements, and a wide breadth of information sources make this difficult to manage. This paper discusses issues that arise in large enterprises and ways to avoid general classes of problems in the area of scalability.

INTRODUCTION

Data warehousing has been around for many years and has been adopted by many organizations to help manage the complex data flows and processes involved in normal business operations. To some, it is thought of as a collection of ETL (an acronym for the main functions of **E**xtract, **T**ransform, and **L**oad) tools. To others, it also includes the infrastructure required to regularly run business processes involving large data volumes and the generation of deliverables such as reports. However, for those involved in the largest enterprises, there is an additional set of requirements as well.

PROJECT REQUIREMENTS FOR ENTERPRISE DATA WAREHOUSES

Some basic differences between basic warehousing projects and enterprise-level projects exist and need to be considered to ensure that projects can continue to grow and scale as needed to meet their high-level objectives.

While a basic data warehouse (shown in Figure 1) is concerned mainly with the extraction and transformation of a limited number of information sources into data organizations that can service a set of pre-defined functional needs, enterprises add the requirement that the warehouse be a dynamically evolving entity able to service current as well as future anticipated needs.

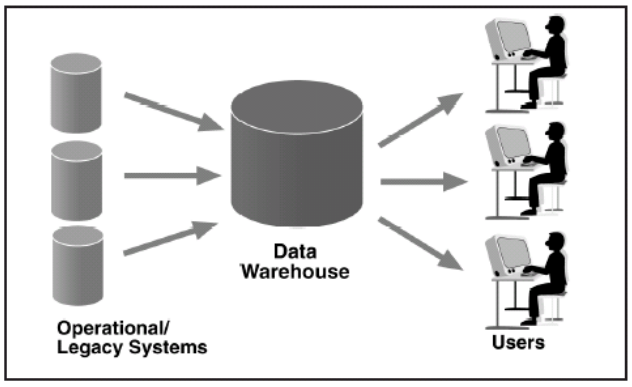

Figure 1. A basic data warehouse; one version of the truth for many users.

The need to continually evolve and meet a large number of end-user requirements usually means that a large team of people are responsible for its design, implementation, and support. Moreover, the complexity of the system requires that different levels of a warehouse exist to support initial development, integration and testing of the design, and finally actual use in the real world.

MULTI-USER PROJECTS

Larger design, implementation, and operational projects involve teams that have different types of members:

- a systems expert who helps architect the overall structure of the project. This person will be responsible for the production version of the project and understands the internal data structure and modeling requirements. This person is also knowledgeable about data available in operational and legacy systems.
- a small number of project leaders who have responsibility for the daily development of the project implementation. The people coordinate aspects of delivering a running system that is able to be rolled out by interpreting high-level descriptions into detailed work specifications.
- a team of ETL developers that develop actual process steps as part of a team. Due to the large scope of a typical project, this team works concurrently on different aspects of the same overall project. Individual pieces are typically developed in limited development areas and later merged into integration testing and then deployment areas.

In order for the entire project team to work successfully together, the tools they use need to be aware of and support both multi-user development projects as well as the ability to migrate completed portions to areas in which progressively larger integrations and tests can be performed.

SUPPORT AND MAINTENANCE OF LARGE PROJECTS

A group of developers can run into a lot of difficulty and overhead if they have to worry about synchronization problems and partitioning requirements on their work. A tool that can assist with this issue, from merging individual components to integrating larger pieces into testing environments is key for long-term success.

The issue boils down to being able to think of individual project components as inter-related and interdependent nodes in a collection of objects that comprise the overall project space. In this way, object-level dependencies and structures can be combined in ever-larger pieces. The alternative to this approach is to combine actual implementation code with the controlling algorithm, which can result in a relatively unstructured body that more approaches a linear code program. The key difference is that the structure can be placed above the implementation in the object-structured method, or interspersed with low-level implementation code.

Without advance planning and tool support, a project can easily end up too unstructured to effectively manage once it gets too

large. Tools that make it easy to specify the high-level architecture and keep filling in lower-level structures and implementations facilitates the most successful projects at large enterprises.

DIVERSE OPERATIONAL SOURCES

Enterprise-level projects can involve a number of sources that might not otherwise need to be considered together. In the example of a corporate scorecard application, data requirements for the four quadrants of finance, customer, internal process, and learning/growth might be fed from Oracle Financials, Siebel Call Center, SAP Materials Management, and PeopleSoft Human Resources. Bringing all these pieces together can be challenging because of the different underlying data structure assumptions made by each. A successful project with these data requirements needs to use an internal, common data model.

DESIGN ENVIRONMENT REQUIREMENTS FOR ENTERPRISE DATA WAREHOUSES

In addition to meeting the specific project requirements, a set of needs is projected onto the design environment and tools being used to implement and manage the project itself. Since the enterprise data warehouse will have a wide range of uses and therefore higher complexity, an underlying object orientation of design is key. These objects are considered metadata, or information about the data, its processing, and use in the enterprise. Figure 2 shows an enterprise-class warehouse. Note that the underlying connection that ties all the components and activities together is metadata.

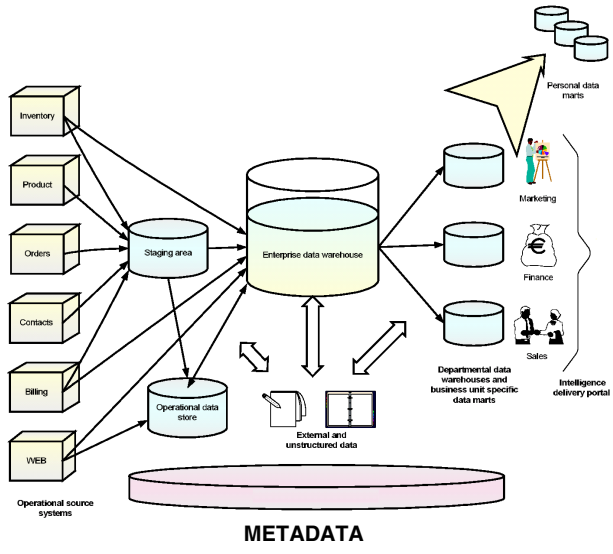

Figure 2. A Typical large data warehouse with metadata underlying the entire structure.

Functionally, we can think about an enterprise warehouse as having design and operational needs. From a design standpoint, the warehouse development team needs to bring sources together, transform and merge them as needed, and ultimately arrive at a target or endpoint configuration. A design environment is used to perform these tasks, and depends on components to implement each type of operation related to sources, transformations, and targets.

We can use the following terminology for these areas:

- source designers. These components support connection to source systems, whether they be databases, operational systems, web information, or any other regularly-used type of data. Wizard-based panels guide the user through the exploration and discovery process to locate the needed information.
- transformation components. These make it easy to perform common operations like joining multiple data sources in various ways, sorting or subsetting data based on some criteria, or other data-based operations.
- target designers. These components support the population of a specific endpoint for the data flow. This can refer to a data organization such as a star schema, or a particular destination requirement like writing analytic results back into a specific location in an operational system.

Pieces of the design environment are used to build larger structures we'll refer to as recipes, or reusable scripts for repeatable processes.

Beyond the design environment, there are administrative requirements for the repeatable operation of regular processes as well as security and other management tasks that will be covered later.

ACCESSING SOURCE DATA

In larger organizations, a broader range of information sources – including operational systems – need to be fully supported. This is in addition to the traditional data sources like database systems like Oracle, DB2, and Teradata, as well as legacy sources. While past concern about Y2K helped speed conversion from older legacy systems into newer operational systems, these conversions resulted in more implementations of operational ERP (Enterprise Resource Planning) systems like SAP R/3, PeopleSoft, and Oracle Applications. Although traditional databases underlie these operation systems, direct access is very difficult due to the vast number of tables involved (tens of thousands in large systems) and the internal inter-relationships and other complexities in them.

Another type of data source has become available and important in recent years: data related to website utilization. While companies that are solely dot-coms have not fared so well recently, most large enterprises do have a successful web-based portion of their operations. Managing an enterprise-wide website – either for internal or external use – is a separate operational area, but it still has valuable information in it related to customer activity and the success of marketing programs or information distribution, as shown in Figure 3. Information about how registered users are using the system, source address information for web users, page effectiveness, and link use patterns can be gleaned from web logs and other information sources maintained by web servers.

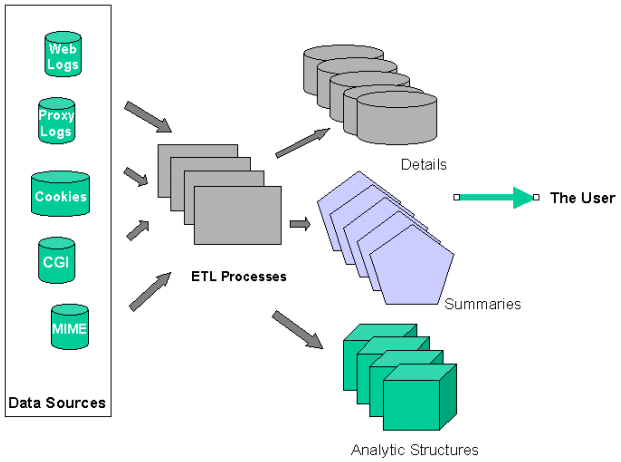

Figure 3. Website information seen as different types of data sources.

While operational systems helped consolidate various sources into single-point operational environments, they also brought new levels of internal complexity into the enterprise as well. The internal complexity resulted in more closed systems than before as each operational vendor sought to extend their product suites into further operational areas, from materials and inventory management into general financials and human resources. As these systems became more broadly usable, their internal complexity evolved as well, making it more difficult for traditional warehousing systems to penetrate into the operational data environments. Moreover, many of these vendors are moving into front-office customer relationship management (CRM) systems as well, further compounding this isolation effect.

In the largest enterprises, several operation systems may be present. Often they remain separate due to divisional level management issues, but for a complete vision of the enterprise, these sources need to integrated with each other as well. Besides the complexity of being able to access the data in the first place, data quality issues are amplified as more sources are added to the requirements. While data handling in a single operational system may be relatively straightforward, issues of matching and other consistency checks multiply in importance as more operational systems need to be integrated. This is shown in Figure 4.

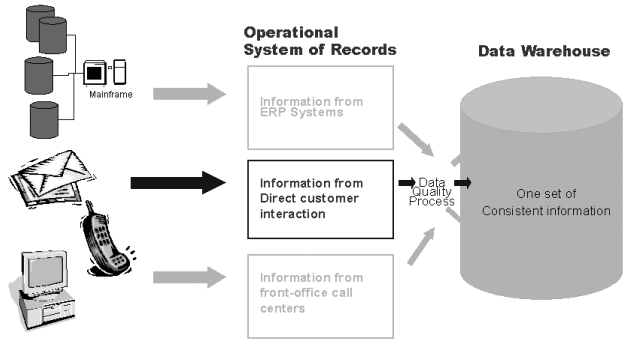

Figure 4. The centrality of data quality in multi-source systems.

Some operational vendors are responding to the warehouse integration issue by developing their own data warehouse software to support traditional warehousing requirements like consolidation and summarization. An example of this is SAP's Business Information Warehouse (BW), which is now an integrated part of larger SAP installations.

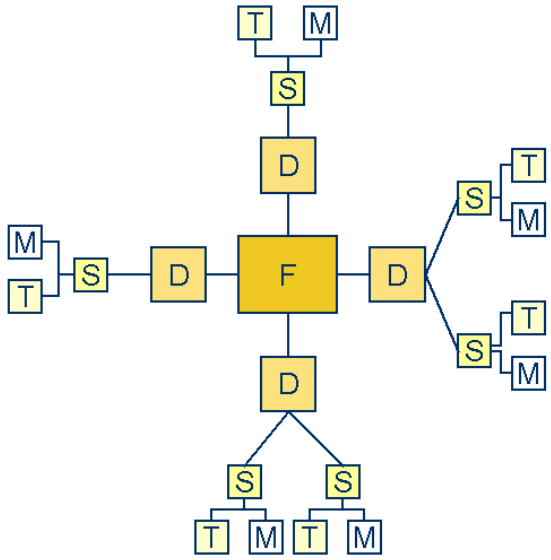

Figure 5. The layout of an SAP Business Warehouse snowflake schema. Using this information more generally can require an understanding of its internal layout within SAP BW.

However, these ERP vendor products remain largely focused on the needs of operational reporting to meet the core customer needs in that area.

For example, SAP's BW uses internal structures some have referred to as a snowflake schema of fact, dimension, SID, text, and master tables (see Figure 5 for an sample rendering)

While the snowflake meets the internal needs of SAP BW, it doesn't necessarily lend itself to extended analysis like the more common star schemas common in the OLAP world. Figure 6 shows a more normalized form of this information. Understanding and translating between the forms is a warehousing transformation issue.

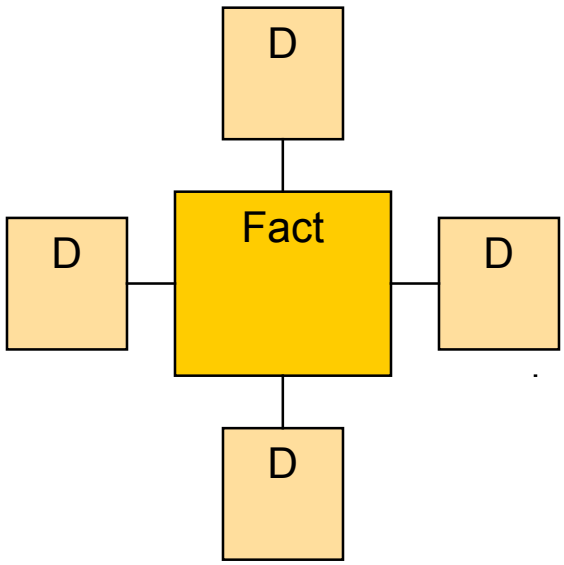

Figure 6. A more analysis-standard star schema comprised of just fact and dimension tables

In order to extend an external warehouse to support other operational sources, one must resort to additional third-party software products. More importantly, to extend them to the differing needs for analytic reports and deeper analysis like data mining, the structures set up for operational reporting become less useful. Users must resort to intricate changes or parallel warehouses to meet the higher-level analytic requirements. If SAP BW is present in the enterprise and needs to participate in enterprise-level analytics and reporting, the ability to extract data from it for general consumption is an important consideration.

Another important consideration for operation system integration into an enterprise data warehouse is management of the metadata from the operational system. While this information is vital during the discovery process of finding what is needed in the operational system, it is often extremely bulky in its entirety, and change management across metadata environments can quickly become an unwieldy problem. A reasonable approach is to not necessarily incorporate external metadata universes in their entirety into the warehouse metadata repository, but instead to bring in only what is specifically needed to address particular data access requirements defined for the warehousing processes.

Finally, in addition to access of specific operational entities like tables and cubes, the ability to interpret business-level components and other higher-level concepts are important in these systems as well. The enterprise warehouse needs to facilitate the understanding and reuse of this information as more complex objects are understood and exploited in the warehouse. The warehousing tools need to help the user understand how the business component organization works and how it can be re-exploited for further gain in the future.

DISCOVERY AND EXPLOITATION OF OPERATIONAL SOURCES

While the warehousing environment strives to make operational sources appear as straightforward to use as simple tables, the way in which they differ from traditional RDBMS sources is how the user finds the content in which they are interested. It isn't nearly as straightforward as already knowing what table holds the source data of interest. In most cases, a discovery phase needs to occur. This discovery process involves using powerful tools to navigate and search the internal metadata topology of the operational system to help find the right nugget of source data.

These tools help the user navigate common structures such as the internal data model of the operational system. The internal data model helps with understanding the interrelationships of entities that together need to be considered. Another approach is to help by subgrouping the internal tables by their type of content, either by functional product area or by table type, such as data dictionary, public interface table, or publicly useful tables.

Perhaps the most useful way to discover what is available is through powerful search algorithms. These searches expose verbose or "friendly" descriptions of internal entities to help the user understand how the metadata is really organized. These descriptions can make a large difference when one considers the alternative of using internal, cryptic database table names directly.

Once the expected source is located, other validation steps can be performed, such as viewing descriptions of the metadata to ensure that the right types of information are being found, such as numerics, character strings, or date values as expected. In addition, sometimes actual data needs to be previewed to ensure that test values appear as expected, or that a table isn't empty or deprecated but still present. The data discovery tools need to support these validation types so the user doesn't have to change to another environment to separately perform these types of validation.

CLOSED-LOOP PROCESSES FOR FRONT-OFFICE SYSTEMS

Another class of operational systems, for front-office customer relationship management (CRM), has similar requirements to back-office ERP systems for accessing data for extraction, but can add a new requirement for write-back into the operational system in certain cases. Figure 7 shows this visually, and again reminds how vital metadata is across the span of this type of system to help integrate the various data and process requirements.

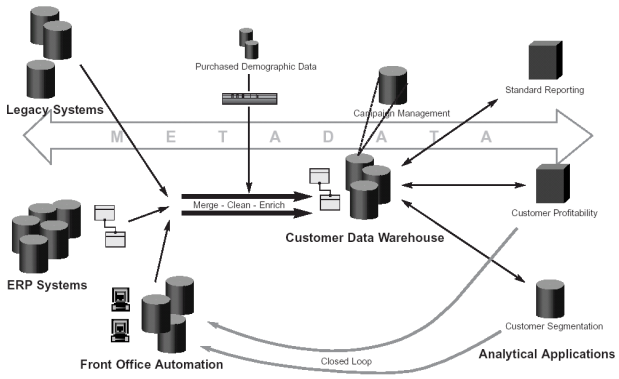

Figure 7. A Closed loop CRM architecture.

When applying advanced analytics to determine customer segmentation for marketing campaigns or other needs, the ability to deliver analytic results such as statistical or mining model scores back into the system becomes important. In this scenario, the ability to design processes to close the loop from extracting, analyzing, and inserting back into the CRM system needs to be considered as part of the overall warehousing process data and process flow.

DATA DESIGN ENVIRONMENT

Once the business goals are defined and roughly understood from a data source and process flow environment, a graphical design environment for specifying, testing, and supporting it is vital to the ability to rapidly design and validate the flows required to implement the goals. Figure 8 shows a typical entity-relationship diagram. Whether an external entity relationship diagramming tool is used for complex data model design or simpler target designs are needed, the ability to directly import or implement them in the data design environment is key to getting underway. After all, these are the goals or endpoints of the process being undertaken.

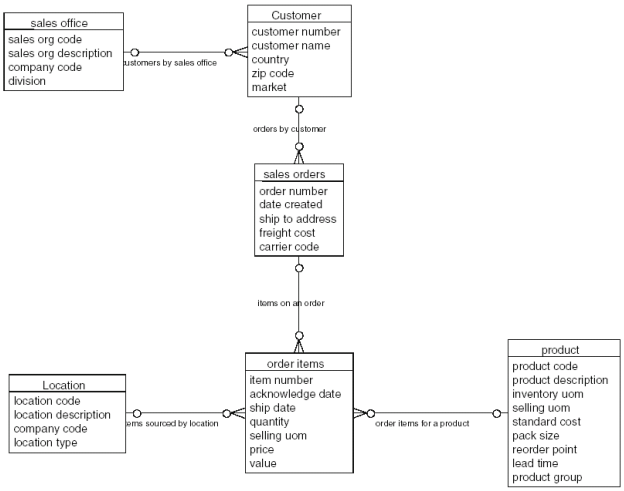

Figure 8. A simplified logical model to support a sales/marketing analysis application.

Working towards populating the endpoints or targets is how data flows and processes are planned and designed. An environment that supports as many common operations as possible graphically is a tremendous time saver over manually writing code to implement and manage the overall process.

To take a simple case, for example, to populate a specific target table definition, we need to understand the source information requirements and manipulations or transformations required to get the information into the right format. Source definitions can be from known table sources or from external, operational system as needed. Transformations involve typical data manipulations such as joining collections of tables, sorting, subsetting, or otherwise modifying the table structure. This is really the ETL process of defining what to Extract, how to Transform it, and into what it will ultimately be Loaded.

A graphical environment that supports visually selecting data elements and building processes around them should be how most of the process is performed. However, when intricate operations are required, the user should be easily able to extend through custom code in a standard, easy-to-use language such as a 4GL designed for data manipulation. Moreover, when a data/process flow has been defined, it needs to remain transparently accessible for future use and modification. An open repository for storage and access of previously-defined flows is key to being able to easily extend and re-use previously-done work with a minimum of overhead.

The ability to save and re-use data/process flows can be thought of as supporting cookbook development in which previously-defined processes can be located, studied, modified slightly if needed, and re-used. Using the cookbook analogy, such a flow could be thought of as a recipe to create a particular result. Figure 9 shows a sample recipe that captures some basic flow information as data migrates from source to its exploitation destination.

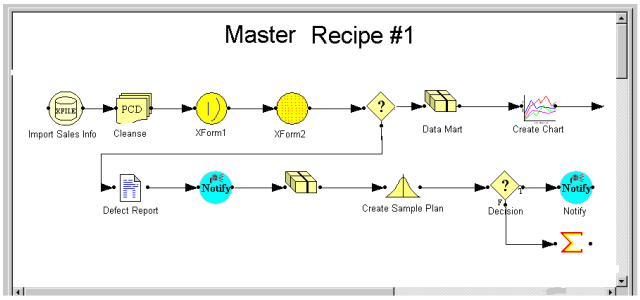

Figure 9. A example data recipe.

Collections of recipes are built up into larger, interdependent processes that comprise the implementation of segments of operation of the data warehouse. In this way, intermediate targets are created and built upon until the goal of an ultimate target is reached. Similarly, endpoints or targets of any particular process or recipe can be inputs or sources for others further downstream. So, as more and more process recipes are combined, more complex problems can be addressed. However, there will likely remain some ultimate targets (the final table or data model being populated), and original sources (external or operational tables for pure source information to the entire process).

In addition to constructing and re-using process recipes, certain commonly-used operations can be supported by process templates to speed development times. These templates specify common operations like sorts or joins and allow the user to drag data elements onto slots to more quickly achieve the desired results.

While the items like process recipes and templates discussed here aren't needed by every ETL designer, they greatly facilitate the development of larger, enterprise-class solutions. Other enterprise-level requirements are that the design environment support multiple concurrent developers and isolated development levels.

MEETING MULTI-USER AND MULTI-LEVEL REQUIREMENTS

Larger, enterprise projects will likely require that more than a single developer be involved in any given project phase. In such scenarios, a project manager designs the process while others implement various portions of it. When multiple users need to share access and update responsibilities to the body of processes being defined. This requires that the repository into which results are stored fully support multi-user access. Without this capability, complex merge processed are required, which can add substantially to the time and risk involved in a project.

In addition to multiple people working on the same project phase, large enterprises require that development efforts work in a isolated environment that protects operational data from interim, untested processes and results. Once development is largely complete, the processes are validated with snapshot data in a test environment until confidence is built that the results are correct. At this point, the processes can be declared production and moved into the real world to run against live data. Often this scenario is called a three-tier dev, test, prod environment.

The data design environment needs to fully support migration and merging of new pieces from earlier levels into later ones, just as pieces from multiple developers need to be integrated. Built-in support for this movement greatly simplifies the process as well

as adds reliability to it. Since this is a common operation in large enterprises, automation of this movement is vital.

To respond to future requirements, the design environment needs to support the ability to add function modules later on an as-needed basis. The ability to plug in new components for access to new types of data sources, or to perform specific types of repeatable operations makes it easy to extend the package without requiring frequent updates of the entire system.

DATA PROCESSING ENVIRONMENT

The design environment can be effectively performed on desktop computers to facilitate ease of use of the graphical tools since designers typically work with very little actual data and usually are mainly concerned with metadata only. Since descriptions of the data are generally much smaller than the immense volumes of actual data in operational systems, desktop computers can meet the data throughput requirements adequately.

In the real data world, though, data volumes can be extremely large. To facilitate timely processing of vast data volumes, server-based number crunching is still generally required. The ability of designed processed to be deployed and executed on larger, server-class systems is key. This means that even if the process was defined on a desktop PC, it might need to be deployed on a processing server that is a large UNIX box. This can maximize the benefit of data/network data locality and minimize excessive data movement to lower-power desktops on slower networks.

In addition, some types of activities may be able to be specified and executed within the operational server requirement itself. For example, when retrieving a subset of data, instead of extracting a very large table to an data processing server for processing, it is sometimes much more beneficial to ask the operational database or application server to perform the subsetting itself, only passing the needed information for processing. The design environment should support this execution selection whenever it is possible and advantageous on the operational system. For example, if reporting on the sales from yesterday stored in an SAP R/3 system, asking the SAP system to itself only return yesterday's results can make a lot more sense that asking for all sales results from the R/3 system, to perform a date-based query on another application-processing server.

Beyond tuning data-oriented operations, a data processing environment that can also handle other operations like analytic calculations is valuable as well. Without this capability, the data processing environment needs to integrate with another software package to do the various types of analytic exploitations required. The ability of the data processing environment to perform analytic tasks from basic queries to advanced forecasting and data mining makes if easier to design complex enterprise-level processes that have these components as important activities. Particularly when vast data volumes are required to solve enterprise-level problems, the ability to easily design these analytics in the same design environment greatly aids process design. The same is true of the ability to centrally monitor execution of data-oriented and analytic-oriented operations in the same way.

As more processing complexity is added, the need to schedule job execution becomes a requirement, since manual operation leads to increased overhead costs and decreased reliability. Further, the need to monitor currently-running jobs and review status logs for previously-executed jobs becomes a regular requirement for an enterprise-level administrator. Tools that support these requirements need to be used to fully deploy complex jobs.

Finally, the data processing server environment should be tuned for taking full advantage of the capabilities of the hardware on which it runs. Specifically, the ability to multi-thread common operations to speed execution of specific steps, or to process multiple requests simultaneously is required in large enterprises in which concurrent requests and requirements are the norm.

INTEGRATED MANAGEMENT ENVIRONMENT

As more complexity is added to operational and warehousing requirements, the number of components that need to be administered, maintained, and monitored can quickly become unmanageable. Administrators can spend a lot of time using too many different and separate management utilities to remain effective. An enterprise warehouse administrator needs a single management console into which management components can be added and from which many operations can be controlled.

Standard management console layouts use navigational panels as the basic layout shows in Figure 10. This is used to organize management functions and components, as well as a common display work area for the task currently being performed. Easy-to-use menus and toolbars simplify navigation to frequently-used tasks.

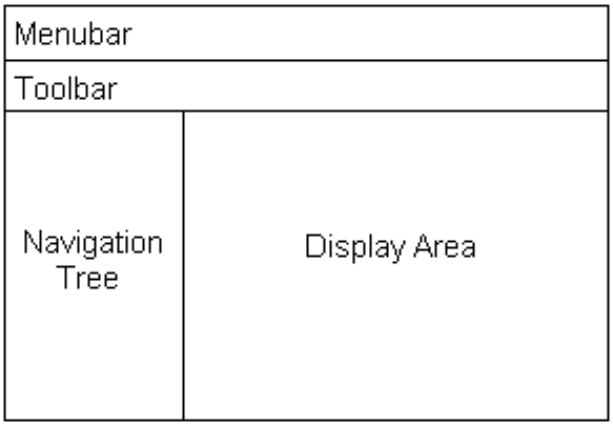

Figure 10. The basic layout of a management console. All management components are accessible from it and operate in a similar fashion.

Common activities to unify include task-oriented procedures like application maintenance and configuration, remote server connection parameters and security information, job scheduling and monitoring, and metadata repository management. From this single console, all of these areas can be easily controlled, making it possible to oversee more activities as they occur across the enterprise.

Complex operations need to be supported by wizards, which as step-by-step screen panels that walk the user through each step of the process. For example, the process of defining a connection to a remote database or operational server is a common activity, but one with various steps that cover locating and then connecting to the remote resource. A consistent way of specifying these steps becomes extremely helpful as the scale of the enterprise grows. In the warehousing world, the

administrative console should support setting up connections to each type of supported remote database, operational system, or other resource in a consistent and straightforward manner.

Once this information has been entered, the management console also needs to support maintenance of this information. So, if the information is being stored in a common information repository, the console has to be able to find this information again and make it easy for the administrator to modify settings as needed, perhaps to add new user permissions or when responding to changes such as updates to a database system.

The management console needs to be easily extended to support new products or functional requirements. An example would be an enterprise that adds a new front-office system like Siebel to the data warehouse requirements. When software is licensed to support data access, components to setup, manage, and utilize this functionality are automatically added into the set of management tools accessible via the management console as well.

METADATA REPOSITORY

A common, open metadata repository is an underlying requirement for success of an enterprise-class data warehouse. In addition to serving as a central repository for structured information describing systems, processes, data flows, and deliverables, it also serves as a single integration point for all applications requiring access to this information. If a core benefit of a data warehouse is "one version of the truth" in terms of data values, then the metadata repository is a central source for all aspects of warehousing processes and flows, configurations and parameters, security and user rights. We can then think of the metadata repository as the "one place to find everything else" other than actual data values themselves. Figure 11 shows a visual representation of this.

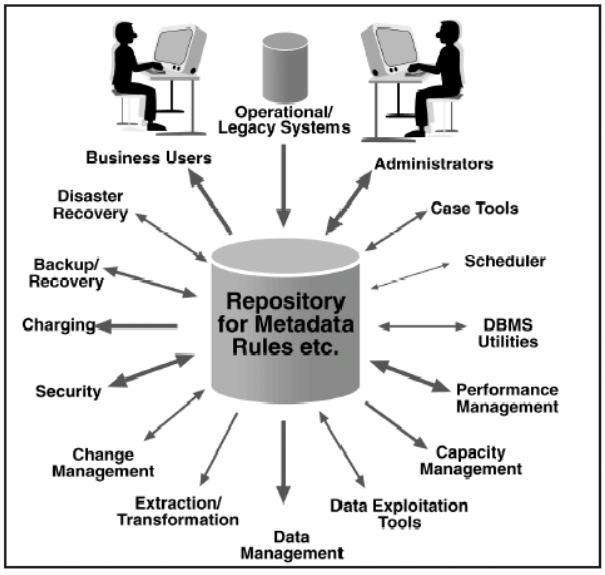

Figure 11. The centrality of metadata and metadata management.

The "everything else" storage support eliminates the need for private, proprietary information that is difficult to surface to other applications or users than that for which it was originally intended. For example, configuration and connection parameters for an external server can be used by any process needing data from that server, and for which the user has access rights. Any type of application can share and access information with other applications through the metadata repository as long as both know how to store or locate the information of interest to them.

This shareability requires that there be an open, standard interface to the repository so that various types of applications can access the information they need. A standard method like the use of XML-based request and result strings simplifies this, and is expected in enterprise-class repositories. XML-based access also means that simple web-based applications can participate as well through Java applets or server-side components that can utilize XML-based communication. Figure 12 shows how the metadata server provides this interface capability to easily access information in metadata repositories.

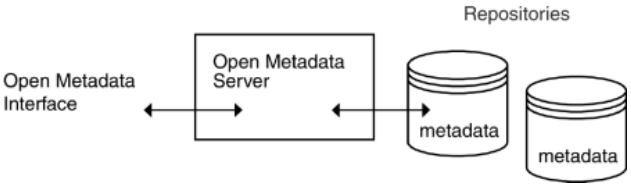

Figure 12. Open, standard access to metadata facilitates sharing across applications.

An example of what might be stored in the repository is a full description of the process of accessing a particular piece of data from an external system, perhaps a Siebel system. This description includes access to any process flow designed previously to construct data in the required form as well as possibly including connection parameters to allow direct access to the external Siebel system. Because of the structuring of this information, another separate application can retrieve and execute the process from the repository and not have to necessarily care about the underlying connectivity issues or other complexities.

In large organizations, heavy use of the metadata repository requires that it function efficiently, and support an in-memory database form of execution as well if needed. In this way, data that is needed by multiple applications can always be quickly accessed.

METADATA SECURITY REQUIREMENTS

This power of course requires that the repository be a secure storage mechanism. Access to sensitive data and important business processes requires that the repository be able to utilize a strong security methodology based on operating-system-level user validation and authentication. Only if the correct credentials are provided will access be granted to potentially sensitive information. This is particularly valuable for external sources such as financial or human resource operational systems. Once sources are designed, the warehouse staff can rest assured that only authorized users will have access to particular pieces of information as defined by their access credentials. For example, personnel information from PeopleSoft might only be available to staff in Human Resources management for certain types of government-required reports.

Looking more closely at security issues, the needs start with the need to authenticate users to validate they are who they say they are. This can be done by requiring a user/password combination that is validated against a secure, operating-system-level service. Therefore, access to a metadata repository has this as an initial

requirement. Once it is known who the user is, authorization facilities need to be used to ensure that any individual user has the appropriate right (to read or write, for example) any particular resource (such as a data table, a report, or other information). These facilities depend on an administrator already having assigned permission schemes that are the reference information source for access.

Closely tied to the authentication/authorization process are those administrative facilities that need to know about all of the types of objects being protected and easily allow the security administrator to create and maintain these settings. In addition to direct user-based functions, the ability to establish and maintain groups of users with similar access rights simplifies the overall security management process for a large enterprise.

Access templates are another important aspect of security. They greatly facilitate the process of defining access right patterns for specific or groups of resources based on users or group permissions. They can be thought of as allowing an administrator to define access policies for various types of resources.

CONCLUSION

Enterprises increasing need greater access to their information resources for a 360-degree view of their suppliers, customers, and internal resources. What has been described in this paper is the set of requirements for enterprise-class ETL and warehousing projects to help achieve this. Enterprise-class infrastructure, including scalable metadata management and data processing abilities are required, as well as a development environment that can scale to support teams of warehouse developers.

TRADEMARKS

REFERENCES

Curran, Thoman and Kenner, Gerhard (1998), “SAP R/3 Business Blueprint”, Prentice-Hall.

Hashmi, Naaem (2000), “Business Information Warehouse for SAP”, Premier Press.

Ryals, Michelle (2002), “Empowering Business Decisions with the use of Metadata”, *Proceedings of the Twenty-seventh Annual SAS Users Group International Conference*.

Todman, Chris (2001), “Designing a Data Warehouse Supporting Customer Relationship Management”, HP Press.

CONTACT INFORMATION

The author may be contacted at:

Gary Mehler
SAS Institute, Inc.
100 SAS Campus Drive
Cary, North Carolina 27513
Gary.Mehler@sas.com