Microsoft

Harnessing the Power of SQL Server™ 7.0

Turning data into information at Microsoft

White Paper

Microsoft® IT Showcase

Best practices, experiences and global solutions from Microsoft IT

CONTENTS

EXECUTIVE SUMMARY

Information in the workplace is of value only insofar as it supports the day-to-day decision-making activities of executives and other employees. For all companies, increasing the value of information helps ensure that decisions made will enhance business practices and increase profitability. Yet this task is compounded by the nature of the information itself. At Microsoft, for example, the volume of business-critical information has grown dramatically, and so has its distributed nature. Just as Microsoft is a worldwide enterprise, the information on which its decision-makers rely is generated and stored across the globe.

For these reasons, Microsoft, just as any other large company, is constantly making efforts to increase the value of information that drives corporate decisions and to make it more easily accessible. One such effort focuses on Microsoft® SQL Server version 7.0, the newest version of that database management system. The Microsoft Information Technology Group (ITG) relies on SQL Server 7.0 for storing, filing, sorting, sifting, and reporting a large volume of information—with very favorable results. So far, applications that ITG has based on SQL Server 7.0 have proven to be faster, more scalable, more available, and more easily maintained than applications based on prior versions of the product.

Two such applications are MS Sales and the three-tier SAP R/3 Enterprise Resource Planning (ERP) system. By deploying MS Sales, ITG has enabled Microsoft to remove 18 nonintegrated legacy computer systems that stored information in widely different formats. Now, for the first time, this information is stored and accessed in a consistent, uniform, and truly efficient manner. This means, to give just one example, that decision-makers can easily track sales every step of the way, from distributors to resellers to customers.

As for the SAP R/3 system, it too, has enabled Microsoft to remove legacy systems—more than 50 of them—which had been costly and troublesome to maintain. As a result, each year the company is saving more than $5 million in hardware-maintenance fees and $15 million in procurement costs. This ERP system also enables Microsoft to close its books within four or five days instead of four weeks, as had sometimes been required in the past.

In this paper, you will learn more about the benefits of these, SQL Server 7.0–based solutions, and about how ITG created them and is now supporting them. You also will learn about ITG's strategy for designing, deploying, managing, and maintaining SQL Server 7.0–based solutions in general, for present and future needs alike. The goal of this paper is to share the lessons ITG has learned along the way developing deploying and supporting SQL Server. 7.0

The Microsoft Environment

The Microsoft Information Technology Group (ITG) works hard to ensure that data is highly available and presented in a form that meets the stringent business requirements of employees so they can use it to make decisions quickly and intelligently. For example, Microsoft employees in all lines of business routinely rely on information stored in SQL Server–based databases. This information is used across the company from product development to finance. Such functions range from recording beta software defects using an online transaction processing (OLTP) system to data-mining one of the company's many online analytical processing (OLAP) systems.

By deploying SQL Server 7.0, the newest version of Microsoft SQL Server, ITG has enabled Microsoft employees to use information in an increasingly efficient manner. In one instance, ITG has noticed a significant reduction in the time required to run database consistency checks (DBCCs). DBCCs that had required up to five days to complete on a 165-GB SQL Server 6.5–based database take less than a day now that the database has been upgraded to SQL Server 7.0. ITG also noticed that SQL Server 7.0 replication significantly outperforms SQL Server 6.5 replication in a production environment.

Today, Microsoft has more than 6,000 SQL Server–based databases on hundreds of computers running the Windows NT® Server 4.0 operating system, as well as a few running on Windows® 2000 Advanced Server beta. Microsoft employees have access to several terabytes of SQL Server–based information. In addition, Microsoft developers are using Microsoft SQL Server to create a wide variety of applications from the MSN™ network of Internet services and microsoft.com to line-of-business (LOB) applications such as the MS Sales financial decision support system (discussed later in this paper).

Most LOB applications Microsoft uses today were developed internally using off-the-shelf Microsoft technology. Business units develop applications based upon SQL Server 7.0 to support a wide array of business functions such as finance, procurement, human resources, manufacturing and distribution, sales and support. In developing applications for internal use, business units determine all of the application requirements such as reporting, backup, availability and performance.

ITG participates in the process of deploying and supporting LOB applications by configuring hardware and software, networking servers, performing backups, monitoring servers and upgrading software. Together both ITG and the business units consider how employees will access applications and how those applications can best serve the information needs of employees. Below are a few of the scenarios, which are considered before setting out to create a new application:

Power users. Microsoft has thousands of power users, who expect extremely fast response times. To meet the needs of these users while keeping support costs down, ITG looks for solutions to avoid such problems as slow response, unexpected disconnection, or applications that "hang" on client computers. ITG developers must strive for acceptable performance even over relatively slow, 256 Kbps WAN connections.

Remote users. Microsoft employees frequently access corporate resources from their homes and while traveling. For this reason, ITG provides a broad range of remote

applications, including direct-dial and point-to-point tunneling. ITG also must work to ensure that applications can recover from errors in the event that remote users abruptly terminate their connection. Engineers within ITG continually evaluate and deploy the latest advances in remote connectivity since such solutions are key to enhancing future remote connectivity.

High availability. A large part of such work is ensuring that the applications these users depend on are running in a SQL Server 7.0 environment and on high-end hardware.

Intranet lifestyle. Employees at Microsoft use the corporate intranet as a vital link to the information they need to get their jobs done. By providing a consistent application user interface on the company intranet, ITG reduces support costs while enhancing the user experience. By integrating SQL Server 7.0 and Microsoft Internet Information Server (Windows NT Server's built-in Web server), ITG takes advantage of the consistent user interface already provided by the intranet while enhancing the information made available to employees.

Large number of international subsidiaries. Early in the design phase ITG must give careful consideration not only to the potential users of an application but also to the location of those users. ITG develops most applications locally but deploys them globally, often to as many as 180 subsidiaries, worldwide. Some employees at Microsoft work in the evenings and often on weekends and holidays, so there's rarely a time that key applications can be brought down for upgrading or maintenance without affecting someone. This means applications must be developed so that they can be upgraded and maintained quickly and easily. ITG has found SQL Server 7.0 provides the high availability that is critical.

Operations Framework for Deployment

ITG's operations framework for deployment is the methodology used to plan, and deploy SQL Server 7.0–based applications, in production. Long term plans for backing up data, and monitoring are formulated as part of the framework. A strong operations framework is imperative if ITG is to meet its charter to serve the information needs of Microsoft employees. Part of such a framework is 24 hour per day, seven days per week availability and a strong discipline of rules governing the operational support of SQL Server 7.0. Having a strong operations framework, including a tightly integrated support structure, helps minimize the time required to deploy new applications, resolve problems, and formulate effective strategies for disaster avoidance. In addition, the right operations framework helps keep costs low and user satisfaction high.

The ITG operations team supporting SQL Server 7.0 today has a strong system of support links and clear escalation paths. A strong organizational model helps ITG employees work more productively and reduces problem-resolution time as well. Problems are escalated to an appropriate level of technical expertise for resolution. ITG staff routinely divide complex projects into manageable components so that teams can work in parallel in order to more easily complete their projects on time.

The operations framework for deployment is organized into three primary areas:

- Hardware configuration
- Software installation
- Supporting the SQL Server-based application

Hardware Configuration

ITG seeks always to minimize the types and sizes of servers that it must support. With this in mind, the group develops hardware standards designed to enable technicians to easily configure, maintain, and install hardware components and to provide users with the consistent high performance they have come to rely upon.

Improving Standards

Developing hardware standards involves benchmarking a wide variety of hardware. Fault tolerance, hot-swappable components, OEM-driver support, remote diagnostics, and performance are all taken into consideration. Chassis size is also an issue, since floor space is at a premium in Microsoft's data centers.

By anticipating the needs of the various business units within Microsoft, ITG provides not only support but also additional value by continually benchmarking the latest hardware in its labs. Lab testing helps ITG to predict how well a chosen hardware configuration will perform once it is used in production. Such testing also helps to ensure that business units can scale their OLTP and OLAP systems with the help of more powerful hardware.

ITG is constantly evaluating new hardware for the purpose of adding such hardware to the standard platform. When a change is made, it's based upon a number of considerations: testing results, anticipated benefit to business units, hardware cost, and associated support costs for the new configuration, and advancements in technology. Adding more memory or storage, installing an additional CPU, or replacing a CPU with a faster model are not considered changes to ITG's standard hardware configuration. In contrast, turning to a new model or brand of hardware is considered a standards change.

One example of a standard hardware configuration calls for servers to be based on multiple CPUs. Such a configuration is especially important for servers running applications based on SQL Server 7.0. By enabling such applications to divide processes among many CPUs, multiprocessor hardware provides the performance and scalability required by various business units at Microsoft. Having the option of adding more processors means business units can purchase affordable hardware today while remaining open to scaling up as needed in the future.

Developing standards for hardware configurations is the responsibility of ITG's Global Networking and Systems Engineering team. This group of engineers is responsible not only for developing the company's internal standards for hardware but also for developing standards for the Windows NT Server 4.0 and Windows 2000 Advanced Server network operating systems (NOS) and for SQL Server 7.0. Work includes evaluation of beta products, escalation of implementation defects to product development for resolution, identifying needed product features, and engineering new

hardware standards for data center use. (For more on the standards developed by this team see “Standard Configuration and Performance Tuning.”)

Configuring Servers

Configuring servers involves acquiring the hardware, evaluating the requirements of the server, and then building the hardware to specification. A standard hardware configuration makes the process of configuring servers easy, because technicians are expert at those set standards. A standard hardware configuration also simplifies hardware selection for the business units. These business units can purchase a small, medium, or large hardware configuration designed to meet the demands of the application that will put the server to use.

The hardware configurations selected by the business units, for their particular LOB, are delivered to ITG for them to build within the corporate data center. As part of the process, ITG configures each machine with the requested number of processors, RAM, network cards, and hard drives. ITG’s server-build team configures servers in a staging area for hardware diagnostics. This team provides the ITG-required quality assurance by ensuring that hardware is built according to one of the group’s standard configurations. Having servers built uniformly helps to eliminate any surprises after the hardware goes into production use.

Staging Servers for Data-Center Use

In order to make the best use of its operational areas, ITG rack-mounts all of its servers. This approach enables the group to stack servers on top of one another, resulting in tall rows of servers that use floor space efficiently. After new equipment has been permanently secured, ITG provides network access by cabling the server to a network switch (Cisco Catalyst). Once ITG has enabled network access, the group begins installing and configuring the Windows NT Server 4.0 network operating system and SQL Server 7.0.

The Network Engineering team is responsible for routing suitable network protocols on each switch and for assuring that production computers running SQL Server 7.0 are cabled to the correct switch.

Software Installation

To make troubleshooting as easy as possible, ITG works to ensure that computers are configured for optimal performance and reliability, with monitoring and management tools installed. For example, to reduce problem-resolution time and simplify software deployment, ITG configures Windows NT Server 4.0 according to a group standard across all servers.

The primary benefit of a standard operating-system configuration is that it enables easier performance comparisons among servers. An additional benefit is that ITG does not need to troubleshoot the same problem multiple times on multiple servers. ITG can resolve a problem once, incorporate what is necessary to ensure the problem will be corrected into the standard NOS configuration, and then deploy the fix to all of the group’s servers.

Installing Windows NT Server 4.0

Installing Windows NT Server 4.0 also includes installing the latest Windows NT Server Service Pack and a number of Windows NT updates, often referred to as “hotfixes”. For ITG’s installation the process also involves installing third-party software for remote hardware diagnostics and Microsoft System Management Server for remote management.

To make installing Windows NT Server 4.0 as easy as possible, ITG has largely automated the process. Rather than relying upon every technician to configure Windows NT Server 4.0 the same, ITG automates the process using the Windows Scripting Host and Visual Basic® Script on servers running Windows 2000 Advanced Server, and batch files on servers running Windows NT Server 4.0. This approach ensures that once changes have become part of the standard, each subsequent change is put into production, automatically. The automated process invokes installation routines, and requires no intervention from technicians.

One benefit of this approach is that it ensures software is installed in the proper order. Most software-installation programs use “file versioning” for installing the latest software. However, for those installation routines that do not, ITG calls the installation routines in the appropriate order in its scripts. ITG thoroughly verifies new changes made to its installation scripts on test machines, prior to using the new changes in production.

The Site Operations team is responsible for installing the Windows NT Server 4.0 operating system. This team is also responsible for automating the installation process and for communicating changes to the standard operating-system configuration to technicians in subsidiaries throughout Microsoft. Site Operation’s installation process is developed locally in Redmond, Washington, but is used globally.

Installing SQL Server 7.0

Installing SQL Server 7.0 is the job of an ITG database administrator (DBA). These DBAs are experts when it comes to supporting SQL Server 7.0. The Data Base Operations (also known as DBOps) team is responsible for SQL Server 7.0 database administration. This team performs all the functions of a DBA, including SQL Server installation, configuration, backups, logon management, security management, monitoring.

Supporting the SQL Server-based Application

Having a properly installed and configured platform on which to run applications is key to reducing support costs, enhancing user experience, and simplifying any needed troubleshooting. ITG’s approach toward loading SQL Server–based applications involves developing strategies for backing up and monitoring the server. The approach also involves populating all databases and putting the application to use by making it available to a wide internal audience.

Early on, ITG provided the same level of operations support for all SQL Server–based applications. Support consisted of 30 DBAs divided among three eight-hour shifts. Each shift of DBAs was responsible for supporting SQL Server and more than 3,000 SQL Server–based

databases. From this experience, ITG found that problem resolution was slow for a number of reasons:

- The need for IT business units to re-explain issues to multiple DBAs
- Some DBAs not being familiar with all supported SQL Server–based applications
- DBAs not fully understanding the support requirements of each IT business unit

ITG overcame these obstacles by restructuring the DBOps team to reflect the diverse requirements of the various IT business units. For example, the business units require vastly different models of support depending on their working hours, the complexity of their applications, the criticality of their business systems, and so on. Today, SQL Server support provides 45 DBAs who are assigned in groups to each of six IT business units.

Formulating a Backup and Monitoring Strategy

Formulating a backup strategy begins when ITG staff ask individuals in the various business units about their tolerance for data loss. Applications that change only infrequently require only infrequent backup; accordingly, applications that change frequently must be backed up frequently. To minimize costs, ITG asks these individuals how frequently their data must be backed up and then schedules backups accordingly. Through the same question and answer process, ITG determines how long data should be retained.

The backup strategy also takes into consideration the varying levels of data-loss tolerance among the various business units. Some business units can tolerate the loss of 15 minutes' worth of transactions while others can tolerate no more than a five-minute loss. ITG schedules database and transaction log dumps to disk based upon each unit's reported tolerance to data loss.

To simplify the creation of a backup strategy, ITG provides the business units with several options. These options are made available based upon ITG's capability to back up and archive data discretely. ITG's approach helps to reduce the costs of purchasing backup tapes and/or having data archived offsite.

Formulating an effective monitoring strategy involves determining the availability needs of each particular business unit. Some business units require servers to be monitored continuously, while others require that systems only be monitored during certain hours. Understanding the needs of the various business units helps ITG to formulate an effective strategy for monitoring without requiring a large increase in headcount.

Populating the Databases

Populating SQL Server–based databases involves importing data from one or more existing SQL Server–based databases or data feeds. This step is not necessary for new databases, such as a product development bug-reporting database that does not require any original data.

SQL Server 7.0 directly supports database migration from Intel-based platforms to Alpha-based platforms. ITG takes advantage of this feature when it backs up databases

on Intel-based servers and then loads the same database to an Alpha-based computer running SQL Server 7.0. This feature enables ITG to further scale several large OLAP systems using Compaq Alpha computers running Windows NT Server 4.0 and SQL Server 7.0.

Often, data obtained from external data sources is saved within SQL Server databases using stored procedures. ITG automates the execution of many stored procedures by invoking them from within batch files. Once the batch files have been written and thoroughly tested, ITG submits them to the Windows NT Scheduler Service for execution.

The production support team is responsible for the aggregate three-tier SQL Server–based system running in production. Team members work closely with members of the ITG application development and test teams as well as with the DBOps DBA who manages the OLAP or OLTP databases.

Putting the Application to Use

Putting a new SQL Server–based application to use involves a good deal of testing. For OLAP systems, ITG tests the results of queries in datamarts to catch any last-minute performance or data issues before making information available to Microsoft employees at large.

The simulation of an application's production experience involves using batch jobs to replicate the production database to a development environment. Periodically, ITG refreshes the development database with current production data. Running both systems in parallel helps to ensure against any last-minute issues when the new application is in production use.

When ITG releases new versions of client/server–based LOB applications, the group also upgrades the application's SQL Server–based database to the most recent version available. Doing this allows ITG to enhance that application by taking advantage of the latest features of SQL Server. The upgrade process involves the following:

- Soliciting feedback for new application features from corporate end-users
- Planning the new application release
- Building a development environment using the latest version of SQL Server
- Replicating the production database to the development environment in order to simulate the production experience
- Developing the new application features
- Testing the code-complete application
- Upgrading SQL Server at use in production
- Replicating the refined application database back into production

Once ITG has thoroughly tested the new application and has gained first-hand experience understanding what the user experience will be, the group then makes the application available to a wide internal audience.

Backup and Restore

Once an application has been deployed, long-term support such as backing up data gets under way. Without data backups, recovery from human error would be extremely time-consuming if not impossible. Planning for the possibility that technicians and analysts might make mistakes helps ITG to safeguard information that is essential to Microsoft.

ITG has found SQL Server 7.0–based databases to be highly reliable and require far less maintenance. SQL Server 6.5 required DBAs to run DBCCs frequently. This process was time consuming and required frequent monitoring to see when the DBCC had completed. ITG continues to run DBCCs on SQL Server 7.0–based databases and is finding DBCCs complete five times faster, and report no database errors. As a result, this helps reduce downtime requirements for running DBCCs on SQL Server 7.0–based databases.

Only in extremely rare cases does ITG need to restore SQL Server–based data at Microsoft. Typically such a need arises as a result of human errors, such as an analyst making an erroneous configuration change or by accidentally deleting a database table. In the unlikely event that ITG must recover a SQL Server–based database, the group reloads the database and then, as appropriate for the application, rolls forward the transaction logs. According to ITG records, data center wide, this process occurs on average less than once yearly.

For SQL Server 7.0 the process of backing up involves saving SQL Server transaction logs and databases to a local logical drive on each server. ITG configured its servers for this purpose in mind, by creating a logical drive solely for storing transaction log backup files. A backup process later scans each logical drive daily and backs up the data to tape. This approach allows ITG to have a central backup department and virtually eliminates the need to have tape devices on each server.

The corporate backup team is responsible for archiving corporate SQL Server databases to tape, tape retention, and restoring data and log files to disk so that a DBA can restore any necessary data.

Monitoring and Maintenance

An effective strategy for monitoring is imperative if simple problems are to be resolved before they turn into something more complex. Resolving small problems before they become big ones helps ITG to lower support costs, and increases user satisfaction.

ITG uses Windows NT Server's built-in Performance Monitor to generate administrative alerts for both servers and services. Tier-one monitoring receives administrative alerts when the available threads on a SQL Server are less than one, for example. ITG uses the alert-forwarding feature of SQL Server 7.0 to forward SQL Server 7.0 alerts to a central location. Alerts received at a central location help to simplify monitoring required by ITG.

ITG relies on several monitoring techniques to identify problems when they first occur. Using a number of monitoring tools concurrently helps ITG to ensure that monitoring tools are working properly while providing more diverse information to those doing the monitoring. For example ITG has monitoring tools that do simple things, such as “PING servers”, and advanced tools that collect performance data. Simple tools are used to answer simple questions, such as “Is the server on the network?” Advanced tools are used to answer more advanced questions, such as “How well is the server performing?”

The Windows NT Scheduler service included with Windows NT 4.0, allows commands to be submitted to servers for later execution. Execution time is determined as one of the parameters when the “batch job” is submitted. The commands are typically written into a batch file, saved as an ASCII file, and then loaded onto each server. This approach allows each server to easily execute hundreds of commands at regular intervals. ITG uses the Windows NT Scheduler services extensively to support SQL Server 7.0. Such use includes, synchronizing the computer clocks, and backing up the system registry using RegBack – included in the Windows NT Server 4.0 resource kit.

As a best practice, ITG strives to limit information provided by monitoring tools only to the degree that indicates action is required. Monitoring tools that report information mixed with error messages is less desirable. Mixing informational messages with error messages requires sifting to detect the error messages. Such practices increase monitoring complexity and, consequently, as a result, problem-resolution time as well. Generally, ITG monitors SQL Server for the following:

Performance bottlenecks — Symptoms of a performance bottleneck appear in everything from SQL dumps and replication to logons from the system console. ITG uses Windows NT Performance Monitor to diagnose such symptoms. For example, by applying Windows NT Performance Monitor over an entire week, ITG can more easily determine if performance bottlenecks occur at regular intervals. Performance bottlenecks that occur at regular intervals are sometimes attributed to the start of a batch process, for example. By logging performance-monitor data to disk for an entire week, ITG has data from both light and heavy server usage to help determine the problem.

In general when performance monitor data is captured to a log file, all available “counters” are added except the network segment. This approach helps to ensure that ITG will have sufficient data to later analyze. ITG avoids adding the network segment counter to performance monitor, because they have found that it has caused some network cards to switch into a busy mode that creates an unusual amount of network and CPU overhead.

ITG recently developed an application it refers to as PerfCal. PerfCal was developed using the Visual C++® development system . The tool is used internally to continually collect performance statistics across hundreds of servers at once. Programmatically PerfCal invokes Windows NT Server Application Programming Interfaces (APIs) needed to read performance counters. PerfCal processes the performance data by storing it within a SQL Server database, so that it may be analyzed at any time. ITG staff use this

information to study performance characteristics, which then aid staff to tune server configurations to improve availability and performance.

SQL dumps — SQL dumps must complete in order for SQL Server–based database and transaction logs to be backed up. Internally developed batch process are used to perform SQL Server–based dumps automatically.

If an error in the dump process has been detected, a console utility e-mails the error message to the appropriate group. The addition of e-mail notification greatly helps to automate the monitoring processes. In general, ITG strives continually to improve its monitoring techniques through automation, for instance, by using the Windows NT Scheduler Service.

SQL Server blocking processes — These are a result of possible coding or flow issues in transact-SQL code. The term “blocking” refers to the inability for one transact-SQL process to complete because it is waiting for some resource to become available. Blocking is the result of one or more transact-SQL processes waiting for another transact-SQL process to “unlock” data.

Having an effective method of reporting blocking processes helps ITG to refine its transact-SQL code so as to make it as efficient as possible. Blocking processes are monitored by regularly recording and programmatically analyzing usage and blocking user levels from SQL Server system tables. (To see the transact-SQL code that ITG developed to monitor for blocking processes, refer to the Appendix.)

ITG relies on a number of investigative methods in order to resolve problems as soon as they are found. Such methods include the following:

Looking in the application-event logs for error events — As a best practice, ITG always looks in the application-event log, SQL error log, and system-event log for obvious error messages before doing any other analysis. When error messages are found in either of these logs, ITG searches the Microsoft Knowledge Base, available at http://www.microsoft.com, for articles explaining the source of the error message. As a best practice, ITG always corrects reported errors first, because the error may affect other areas of the system.

Contacting Microsoft Technical Support for assistance — ITG staff routinely engage Microsoft Technical Support to help resolve problems they have been unable to work out. ITG staff provides Microsoft Technical Support with all information they have been able to glean from their analysis. This positive hand-off helps Microsoft Technical Support begin working the issue immediately, which in turn helps minimize ITG’s overall problem-resolution time.

Moving a process to another server to eliminate a performance bottleneck — If ITG discovers that a process is causing a system bottleneck, it has several options available. The first option is to move the process to a server that’s not currently in heavy use. The second option is to reschedule the process so that it runs at a time when the server it is

already running on is not in heavy use. The third option is to rewrite the process to be more efficient.

Disaster-Avoidance Planning

In any large enterprise, an effective disaster-recovery strategy is a must. Small problems, which often are preventable, can lead potentially to disaster if appropriate measures are not taken within a reasonable amount of time. ITG's approach toward disaster recovery is built-in safety measures that allow the group to prevent small problems from leading to disaster. ITG's best practices for disaster avoidance include use of the following:

Fault-tolerant hardware — This includes, for example, hardware RAID 1 or RAID 5. ITG uses hardware RAID (redundant array of inexpensive disks) extensively on production servers. "Hot swappable" hard drives are also used to enhance the benefits RAID already provides. Hot-swappable hard drives enable ITG to replace hard drives as soon as they become degraded without requiring that their servers be shut down. By replacing hardware as soon as it becomes degraded, problems associated with having multiple components fail at the same time are minimized.

LOG replication of SQL Server–based databases — ITG performs such replication on all mission-critical systems. It does this by replicating transaction logs of each mission-critical server to standby servers located elsewhere, using the log shipping utility included in the resource kit for the BackOffice family. Log replication is a secondary precaution used in production. This has been done to ensure that mission-critical SQL Servers-based servers restart quickly in the event of a catastrophic hardware failure.

Effective database backup strategy for every SQL Server — This allows recovery from human errors, or hardware failures. ITG's backup strategy is to dump SQL Server transaction logs and databases to disk so that they will be backed up later to tape. Recovery from the erroneous removal of a database table, for example, involves restoring the database followed by the transaction logs.

Windows NT registry backup strategy of every Windows NT-based server — This provides an effective method of recovering quickly from erroneous configuration changes. Before ITG makes any configuration changes to its servers, it always backs up the system registry to disk. ITG has automated this process using an automated batch process and the registry tools included in the Windows NT-based server resource kit. The Windows NT Scheduler service runs a batch process, which backs up the SAM, SYSTEM, and SECURITY registry hives of each Windows NT-based server every night. Should ITG forget to use the "last known good" option built-in to Windows NT, the group can recover using its own process.

Performance Tuning and Standard Configuration

ITG's approach toward performance tuning strives to find the Windows NT-based server and SQL Server 7.0 configuration that will give the greatest performance advantage. Deploying newfound optimal configuration settings to all servers also ensures that every server can benefit from the increased performance. Such changes to configurations then

become part of the standard configuration. ITG has found that standardizing its configuration reduces time spent troubleshooting problems, while increasing the overall performance and reliability of every server.

For SQL Server 7.0, ITG takes advantage of the product's ability to automatically determine which configuration changes are required to increase performance. This minimizes the need for ITG to manually tune SQL Server 7.0 at all. In contrast, SQL Server 6.5 required allocating 80 percent of system resources to SQL Server 6.5 and 20 percent of system resources to Windows NT Server. This approach was effective but not always optimal for all servers running SQL Servers under various loads and various operating conditions. The fact that SQL Server 7.0 is now much easier to configure simplifies installation and helps ensure that performance will be much higher.

Windows NT Performance Monitor is used extensively as a measurement and estimation tool to gauge overall performance of SQL-based servers. Performance Monitor determines if a server running SQL Server could benefit from an additional CPU or memory. For example if Performance Monitor indicates that "processor utilization" is consistently above 80 percent, ITG will consider adding an additional CPU. If Performance Monitor reports that "available memory" dips below 4 megabytes, ITG will increase the physical memory of the server.

ITG enables disk performance counters using "Diskperf" (included in Windows NT Server), in order to analyze performance characteristics of a Windows NT-based server's disk sub-system. This enables Performance Monitor to report statistics for physical disk. If Performance Monitor indicates that average disk queue length remains above two, ITG will consider adding a second disk controller to help remove the performance bottleneck. As a rule ITG strives to use separate disk controllers for system data, SQL Server–based data, SQL Server–based transaction-logs and database-dumps. Multiple disk controllers help to ensure that "average disk queue length" will always be within ITG's guidelines.

Developing Configuration Standards

Over the past few years, ITG has become increasingly focused on setting configuration standards for Windows NT Server and recording the performance changes. Setting standards for the configuration of Windows NT Server 4.0 and SQL Server 7.0 has simplified ITG's SQL Server support and has improved SQL Server performance.

ITG must consider the performance of Windows NT Server 4.0, because SQL Server 7.0 relies upon the high performance preemptive multi-tasking provided by Windows NT Server 4.0. Preemptive multi-tasking allows multi-threaded execution of processes to occur across multiple CPUs. SQL Server 7.0 is especially designed to take advantage of these capabilities.

ITG can better and more easily support thousands of servers when set standards are in place. Standards allow for quick determination of operating system version, service packs, and hotfixes. Also, standards for the hardware, NOS, and service packs helps ITG to quickly determine how well each server is performing by enabling the comparison of the respective performance levels of several servers. The ability to compare and

contrast server performance against multiple similarly configured servers helps to determine quickly whether a server is performing optimally.

ITG has developed a formal process to change standards in order to deploy new changes effectively. This process is referred to as “change control”. Changes that will be put to use in production are included within a custom build process so that needed changes will be deployed to all servers. ITG staff use a combination of batch files, service pack installation routines, and hotfix installation utilities included in the Windows NT Server 4.0 resource kit, to customize the build process. Changes included within the build process include the following:

- Registry settings
- Hotfixes
- Service packs
- Network protocols
- Hardware
- ROM Packs
- OEM drivers

Before ITG staff put new configuration changes to use in production, a formal process of pilot testing has been performed. Testing consists of evaluating the need for such change, testing each change in a lab, then putting the changes to use in production on a limited number of servers, followed by system-performance monitoring using Windows NT Performance Monitor. As a rule ITG will deploy new configuration changes to a number of servers running Exchange Server, SQL Server, Internet Information Server, Remote Access Server, Proxy Server, Print Server, DNS Server, WINS Server, as well as several others. Having a diverse group of servers participating in the pilot helps to ensure that proposed changes to system configuration will work well on all servers.

Configuration changes are incorporated into the standard configuration based upon benefits such as higher system availability, scalability, or enhanced feature set. Quarterly changes to ITG’s standard configuration are intended to improve server performance as a long-term average. Approved changes that reduce server performance, once in production will be recalled from ITG’s standard configuration.

The approach taken by ITG is intended to provide fast servers while keeping support and operating costs as low as possible. The servers chosen by ITG tend to have components that are easily interchangeable and upgradeable. These standards help to simplify hardware support and ensure that maintenance occurs frequently. Limiting the types of equipment in use also helps to reduce the variety of replacement components that must be keep on hand.

Standard Hardware Configuration

New servers are built according to the specifications of each business unit and use standard interchangeable hardware components. ITG has made the server-selection

process easy for the business units through the use of standard equipment. Business units are free to select a small, medium, or large hardware platform for their application needs. Specific hardware configurations have been designed for each application platform. For example, ITG maintains 12 different SQL standards based on business unit capacity requirements for their specific application. Each standard is a balanced design based on overall costs: performance, scalability, density, serviceability, and redundancy.

Having several options available helps each business unit purchase equipment it can afford while keeping options open for adding more components later. For SQL Server 7.0, ITG commonly uses the most cost effective hardware available. Table 1 is a list of hardware standards ITG has established for use within Microsoft.

		Compaq		Dell
Server Model	**Proliant 1850R**	**Proliant 5500R**	**ALPHA 4100**	**PowerEdge 6300**
Number of processors	Up to 2	Up to 4	Up to 4	Up to 4
CPU Type	Pentium III	Pentium III/Xeon	EV5	Pentium III/Xeon
CPU Clock speed	500Mhz	500Mhz	533Mhz	500Mhz
CPU L2 cache size	256Kb	512 Kb or 1Mb	4Mb	512Kb or 1Mb
Networking	100 Mbps Fast Ethernet	100 Mbps Fast Ethernet	100 Mbps Fast Ethernet	100 Mbps Fast Ethernet
Integrated SCSI Wide-Ultra	Single Channel	Single Channel	Single Channel	Single Channel
Disk Array Controller				
Controller	SMART-3200	SMART-3200	HSZ70	PERC 2
RAID levels	1,5,10	1,5,10	1,5,10	1,5,10
Controller Cache	64 Mb	64 Mb	64 Mb to 256Mb	64Mb - 128 Mb
Battery backed cache	Yes	Yes	Yes	Yes
Maximum Configurations				
Max memory	2Gb	4Gb	8Gb	4Gb
*Maximum disk	370Gb RAID 5	810Gb RAID 5	2,452Gb RAID 5	1,224Gb RAID 5
** Max Windows NT Server volume size	200Gb	200Gb	200Gb	280Gb
I/O Support	PCI – 3 slots	PCI – 7 slots	Peer – PCI 8 slots	PCI – 8 slots (5x64bit)

* Using 9 GB SCSI drives

** Using 9 GB SCSI drives configured with RAID 5

Table 1: ITG Standards for servers running SQL Server 7.0

Installing, configuring, and supporting SQL Server 7.0 is easier when drive letter assignments are the same on every SQL Server. When installing and configuring Windows NT Server 4.0, ITG partitions drive arrays and assigns drive letters, according to their standards. Writing batch files intended to automate copying files, or dumping SQL Server transaction logs is easier to do when drive letter assignment is already known. The use of consistent drive letters ensures that batch files used on one SQL Server may be used on other SQL Servers without modification. SQL Server data back ups are easier to perform, monitor and record when drive letters are consistent across

servers, as well. Table 2 illustrates ITG's standard for SQL Server 7.0 drive partitions and directory structure.

DRIVE	DESCRIPTION	RAID
C	System (Windows NT Server only)	1
D	SQL Server and System Databases D:\SQL7	1 or 5
E	User Database Dumps E:\SQL\BAK	5
F	Logs F:\SQL\TRAN	5
G	Batch Processing	5
H	Data Device H:\SQL\DATA	1
I	Data Device I:\SQL\DATA	1
J	User Database Dumps J:\SQLBAK\	5
O	Log Files O:\SQL\DATA	1
T	Tempdb Files T:\SQL\DATA	1

Table 2: SQL Server 7.0 disk drive configuration

SQL Server performance can be improved when an appropriate "tempdb" size is chosen. ITG developed standards, which help it plan for capacity, while insuring that SQL Server performance will be optimal. ITG uses the SQL Server 7.0 "Autogrow" feature as an emergency growth measure only.

ITG calculates server storage requirements as a function of how large the SQL Server database is expected to become using the following formula:

> Total Storage Requirement = (Data Base Size) + (.25)(Data Base Size) + (.10)(Data Base Size) + (1.25)(Data Base Size)

ITG adds 25% storage for logs, 10% storage for tempdb, and 125% additional storage for backups.

Standard Windows NT Server 4.0 Configuration

SQL Server 7.0 stands to benefit from a properly configured operating system, capable of scaling to meet the demands of large OLAP systems. As part of its standard configuration for SQL Server 7.0, ITG makes several modifications to how Windows NT Server 4.0 is configured. Table 3 illustrates changes ITG makes to servers running Windows NT Server 4.0.

Configuration Change	Benefit of Change
SNMP service is installed	Service is used by remote hardware diagnostic tools
Network Monitoring Agent is installed	Enhances detail of network troubleshooting.
IPX is not installed	Not needed to access servers
Remove “group everyone” from CMD prompt registry key	Enhances security, by preventing “anyone” from access system registry from a CMD prompt
Remove “group everyone” from perfmon registry key	Enhances security, by preventing “anyone” from running Performance Monitor on production servers.
Change boot.ini to 10 seconds	Enables Windows NT to start 20 seconds faster upon boot
Add CMD prompt link to all users default profile	Enables technicians working at the server console to access a command prompt quickly, adds consistency
Add “net time” “at jobs” to Scheduler Service	Ensures that all servers are synchronized to the same “clock”
Add “registry backup script” to Scheduler Service	Ensure that ITG will have daily full backups of registry hives
Install Windows NT Server 4.0 Service Pack 4	Ensures ITG has most recent service pack software in use
Disable CD Auto run in system registry	Enhances level of control technicians have over running a CD ROM from the system console
Set event log sizes to 20megabytes	Ensures capturing events will be more feasible, used to enhance troubleshooting
Set the system and application event logs to overwrite as needed	Reduces “event log is full” pop-ups from occurring at the system console
Set tasking to “best foreground response time” or “none” for servers running SAP R/3	Improves performance from the system console.
Set static page file size to memory + 50 MB or 10%, whichever is greater, and located page file on system drive	Improves system performance on ITG’s servers
Set “optimization” for network access	Improves OLTP performance on ITG’s servers.
Install System Management Server	Enables DBAs to remotely administer SQL Server 7.0
Set “autoshare” to 1	Enables logical drives to have a “hidden share” for administrative purposes
Set Spooler Service to start manually	Technicians do not print from the console of SQL Servers.

Table 3: ITG’s Standard Configuration for Windows NT Server 4.0

Standard configurations such as those mentioned previously help ITG to scale its SQL Server-based servers easily in order to meet the demands of each application. The high

performance ensures that Microsoft employees can obtain the information they need quickly. Standards, and looking at performance as a long-term average, help ITG to simplify building servers while reducing problem-resolution time. By having servers built uniformly, ITG can compare and contrast the performance of servers that process similar information. This ability to quantify normal system behavior helps to make monitoring simpler and easier to automate.

Since ITG has begun to use SQL Server 7.0, many configuration changes of the past are no longer required. SQL Server 7.0's "self tuning capabilities" has simplified the support of SQL Server by eliminating guesswork. Eliminating the need to tune SQL Server also has enabled ITG to spend more time providing the applications that employees need and less time troubleshooting.

DATA WAREHOUSING CASE STUDY: MS SALES

The bigger the company, the greater its volume of data and the more dispersed that data tends to be. At the same time, the more crucial it is that corporate decision-makers be able to access company data quickly and easily, and receive it in a manner that's truly informative. Only then does the data gain real value as a way to help streamline day-to-day operations, reduce expenditures, and increase profitability. For many companies, the best way to achieve this state is through implementation of a data warehouse—a large data repository designed to efficiently store, process, and organize data for the purpose of reporting information and helping decision-makers better understand that information.

At Microsoft, data warehousing is achieved with the help of the Microsoft SQL Server 7.0 database management system and its Integrated Online Analytical Processing (OLAP) services. One data-warehouse solution that Microsoft has implemented using this technology is MS Sales, which tracks the distribution of Microsoft products from distributors to resellers to consumers. With MS Sales, decision-makers can determine the purchasing habits of resellers and consumers on a regional and global basis and use this information to tune business practices and improve profitability.

Prior to MS Sales, Microsoft had an abundance of data about its customers, distributors, and sales, but the data resided on different systems, in different formats, and with varying degrees of accuracy. Decision-makers had to interpret the context of data before they could use it to make a decision, compromising both efficiency and consistency. What Microsoft needed was a solution that would consolidate, validate, and standardize customer and sales data from around the world. Only with such a solution could decision-makers build strategic plans based on a common, accurate, and up-to-date body of information.

In developing MS Sales, ITG called upon the scalability, performance, and reliability provided by SQL Server to handle the complex and challenging nature of the solution. For example, developers determined that MS Sales would need to gather customer, sales, and inventory data from more than 800 distributors and resellers around the world. Electronic Data Interchange (EDI) would provide information on the products being sold directly to major distributors, the products moving between distributors and their resellers (including value-added resellers, mail-order firms, and retail chains), and the products sold by those resellers to end customers. MS Sales also would need to gather data on Microsoft's direct sales to major corporations and others that purchase Microsoft's products directly.

Developers implemented MS Sales on a SQL Server platform, and the benefits of the new solution became apparent almost from the beginning. For starters, MS Sales has enabled Microsoft to remove 18 nonintegrated legacy computer systems that had gathered data and stored it in widely different formats for different purposes. Now, decision-makers throughout Microsoft get their information from a single source: MS Sales. For the first time, the information available to decision-makers in Tokyo, Japan is the same as the information available to their counterparts in Paris, France or Redmond, Washington.

Moreover, plans built by analysts and strategists at the highest levels of the corporation are more accurate and reliable than before. This is because those plans are built on a uniform and consistent base of information, despite the fact that the sources of that information are more geographically and organizationally diverse than ever.

The process of replacing the Microsoft legacy systems with a single SQL Server–based data warehouse required time, effort, and executive sponsorship. Much has been learned from the experience of planning, designing and deploying MS Sales. The following pages discuss challenges faced, challenges overcome, and lessons learned along the way.

Requirements

There are two fundamental approaches to determining the requirements of a data-warehouse solution such as MS Sales. You can begin with the data, or you can begin with the decision-maker. Beginning with the data involves looking at all the information you might capture as part of your solution and then determining what reports you might produce from it. Beginning with the decision-maker involves understanding what reports are needed to support the business and then determining the information required to produce such reports.

Each approach has its strengths and weaknesses. Beginning with the data, you might develop a very large data warehouse producing many reports, but none that can properly support the business. Beginning with the decision-maker, you might define reports so carefully that you end up gaining a better view of where your company is going, but you may also find out that the information needed to produce such reports may not be available. Weighing both alternatives, the development team charged with developing MS Sales decided that the potential strengths of the second approach—beginning with the decision-makers—outweighed its potential weaknesses.

Consequently, team members directed their efforts toward creating a solution that would be able to find and process the information needed to produce the reports that decision-makers needed. They went about their task first by determining information and system requirements, and then by implementing each of four internal system components and a global front end.

Determining requirements also meant understanding what kind of team could make the implementation flow as smoothly and easily as possible. For example, project leaders decided that both business analysts and developers should work together on the team. As it turned out, this was a smart move. The business people became more interested in technology issues and the technical people became more interested in business issues. As a result, information requirements were easier to obtain and system requirements were easier to implement.

Determining Information Requirements

Understanding the information that’s required to support business practices, streamline day-to-day operations, increase revenue, and decrease expenditures is essential in implementing a successful data-warehouse solution. Accordingly, identifying the

information required to support a company's well-being is key to developing a data-warehouse solution that can make essential information available fast.

For MS Sales, information requirements were determined primarily through meetings of selected business analysts and decision-makers. MS Sales developers developed a sample data mart (database optimized for fast querying) and decision-support tool (front end) that could be used for demonstration purposes. In the beginning, the developers populated the data mart with information they felt would best demonstrate the sample system. Later, they populated it with information selected by decision-makers so they could see whether the system would support their needs. This sample system proved to be very helpful in determining the information requirements for MS Sales.

Determining System Requirements

Determining the system requirements needed to implement the production of a data-warehouse solution was key to creating a scalable, reliable solution capable of processing transactions and queries quickly.

To begin the process, the team determined the basic architectural requirements of each needed component. They decided that a single data-warehouse solution should replace the 18 legacy data systems, necessitating in turn, a highly scalable, three-tier system architecture. Under three-tier architecture, a thin client accesses servers responsible for business rules and transaction logic. In a data-warehouse solution, a three-tier architecture can help maximize scalability and performance by distributing functionality across multiple servers.

For reasons of reducing processing time and maximizing availability of new information, the analysts also determined that MS Sales should consist of four internal components, each running on its own dedicated hardware:

Data warehouse — While MS Sales is often referred to in its entirety as "a data warehouse," from a technical perspective a data warehouse is just one component—albeit the primary one—of the overall MS Sales solution. In MS Sales, the data warehouse collects, stores, and processes a large volume of data using a normalized database structure for storage efficiency.

Data pump — The data used in MS Sales comes from distributors, resellers, and other sources and tends to appear in a variety of formats. Before it can be sent to the data warehouse, it must be transformed into a consistent format. The data pump performs this transformation by validating, "scrubbing," and "cleansing" the data.

Data marts — Data marts are denormalized database structures optimized for fast queries. They are designed to simplify analysis, summarizing, reporting and "drilling down" for decision-makers and other data users. For MS Sales data marts were required to file information in a format that decision-makers would find more meaningful.

Factory — The factory addresses the conflict between the need for efficient storage and the need for fast querying. It does this by transferring data from the MS Sales data warehouse, which uses a normalized database structure for maximum storage efficiency, to the MS Sales data marts, which use a denormalized database structure for maximum

query performance. During the transfer the factory also applies business rules to the data.

Analysts determined that the four MS Sales components should run in parallel for the maximum efficiency and scalability, they also established a consistent direction of process flow from one component to another:

Data Pump → Warehouse → Factory → Data Marts

More detail on the process flow within MS Sales can be seen in Figure 1 .

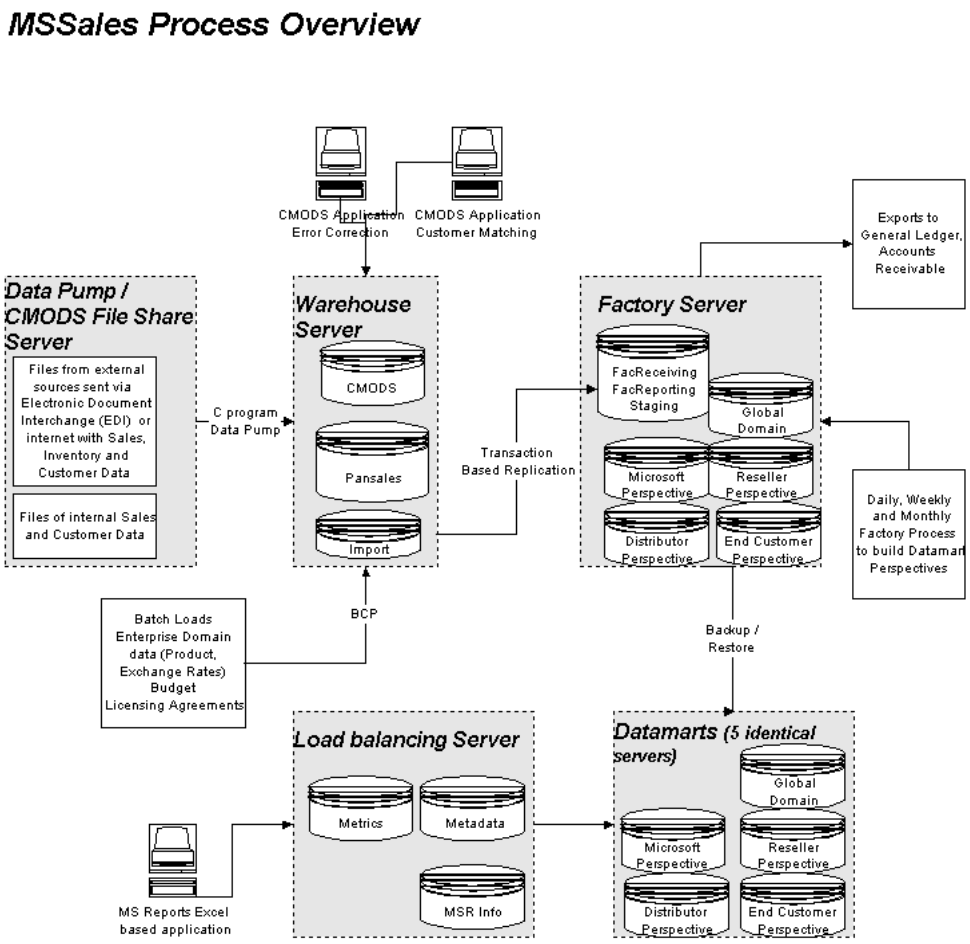

Figure 1 Process Flow of MS Sales

In addition to the four internal components, MS Sales features a front end known as a decision-support tool. This tool takes data from the data marts and provides it to users in an intuitive and highly interactive graphical user interface.

Implementation

Developers of MS Sales implemented the data warehouse over a period of two years. Implementation began after the information requirements and system requirements had

been determined. The next steps involved incrementally developing the data warehouse, data pump, data mart, and factory components. The project was structured to provide new internal releases to the business approximately once every quarter. This release strategy allowed business-users to provide timely feed back on the features as they were added. The following pages discuss the approach taken to implement each.

Data Warehouse

Developers began data-warehouse work by determining what common data was already being captured or could be captured across Microsoft's legacy operational data stores. They then compared this data against the data they considered necessary to produce the reports that would support the business. Next, they began copying data from the legacy systems into a prototype data warehouse. To do this, they created batch files to execute SQL stored procedures for exporting the data from the legacy systems into an ASCII file. Each ASCII file was required to have fields separated by a tab, to later serve as a common warehouse input file. By developing a common input file format, they enabled the data taken from the legacy systems to be put cleanly into the warehouse database.

For a short period of time information was stored in both the legacy systems and MS Sales. Developers of MS Sales would use the legacy systems as operational data stores. Once MS Sales would be fully implemented the legacy systems would be removed.

The developers also formulated rules and translation tables. The rules were designed to ensure referential integrity of information before it became permanently stored in the warehouse. The translation tables took information such as country codes, which had been stored inconsistently in the legacy systems, and translated it into a common index to be used by the warehouse.

The developers designed the warehouse to store data in a normalized structure for granular detail, to perform data-quality checks for referential integrity, and to replace character key values with integer key values. The warehouse holds 400 data-storage tables organized according to the following subject areas:

- Cost of Goods Sold
- Fiscal Periods
- Inventory Snapshots
- Organization (Distributors, Resellers, End Customers)
- Product
- Sales & Marketing Geography
- Sales Transactions
- Third-Party Royalties

The warehouse is a fully normalized (third normal form) database that maintains full referential integrity and enforces business rules through the use of triggers. The tables

dealing with sales, inventory, cost of goods sold, and third-party royalties are physically partitioned into separate “sub-tables” by fiscal month. All insert processes for sales and inventory data are processed in batches known as transmissions. A transmission will contain data for only one fiscal month. Likewise, all select processes for sales and inventory operate only on an entire fiscal month. Creating separate tables for sales and inventory according to fiscal month reduces contention during parallel processes for either inserting or selecting. Roughly 200 tables in the warehouse are used for the physical partitioning of the sales, inventory snapshot, cost of goods sold, and third-party royalties sub-tables.

All primary keys in the warehouse are implemented as integer data types. In cases where the true primary key is a character data type, an integer primary key is created using the SQL Server Identity property. The character primary key is stored in the table as an alternate primary key. The use of integer primary keys results in significantly more efficient joins during factory processing or query processing in the data marts.

As mentioned previously, the factory transports data from the warehouse to the data marts and, in so doing, transforms it from a normalized structure, which is more efficient for storage, to a denormalized structure, which is more efficient for queries. SQL Server replication provides a method of data transport that limits processing to the small incremental set of data that is actually inserted or changed. Since the log records are created as a normal part of SQL Server processing, there is little additional overhead in supporting a replication environment. In the current MS Sales production environment, 496,000 transactions between the warehouse database and factory database are replicated daily and 14 million are replicated monthly.

The data-transport method between the factory copy of the data marts and the actual data-mart servers is the SQL Server Dump-Copy-Load (DCL) utility. The databases are partitioned to include tables for specific perspectives and ranges of fiscal months. If a factory job is run to process all data for fiscal year 1997, then only the databases with fiscal year 1997 data are DCL transported into the data-mart servers. Since there are multiple databases for a given fiscal year, MS Sales creates a SQL Server task to DCL the databases in parallel. Figure 2 illustrates the normalized database structure used in the warehouse.

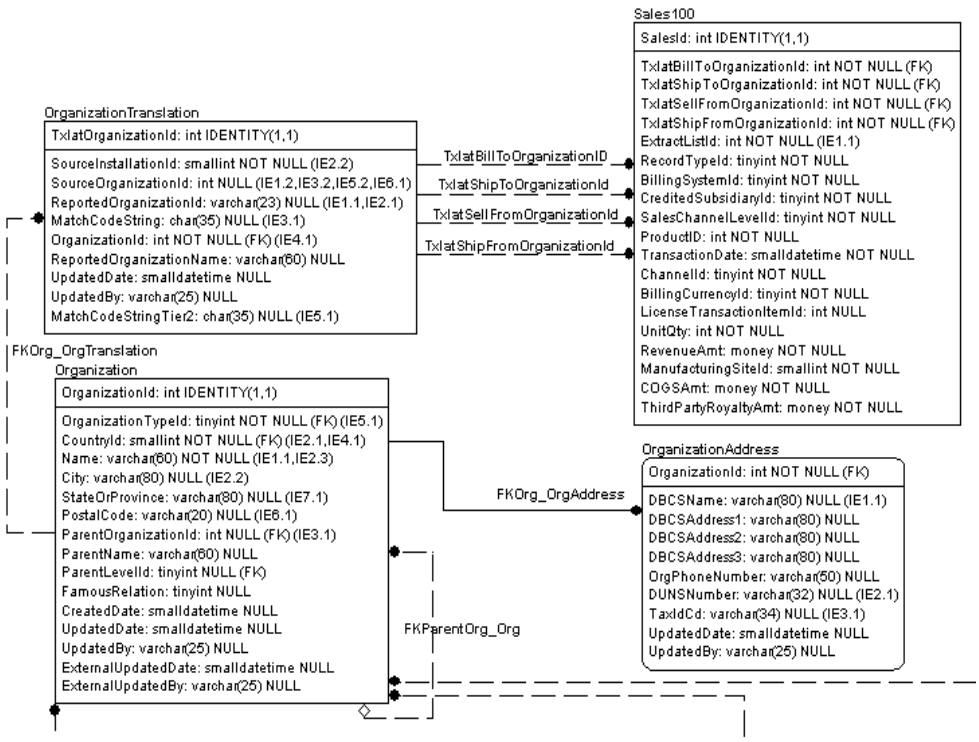

Figure 2 Normalized Database Structure

Data Pump

In order to automate the flow of data into the MS Sales warehouse, developers created an application that concurrently detects and loads received data files into the warehouse. As mentioned previously, the application takes data from the 18 legacy systems and routes it directly into MS Sales. (As detailed in this section, through this process Microsoft would eventually remove the legacy systems altogether.)

To handle the high volume of data arriving daily in these legacy systems, developers selected Microsoft Visual C++ to create a Windows NT Server-based service that they call the data pump. The data pump processes data originating from 800 distributors and resellers who supply information in a wide variety of formats, such as electronic data interchange (EDI), Microsoft Excel, and tab-delimited files. The data pump scrubs, cleanses, processes, matches, and stores the data in preparation for storage in the warehouse.

Note also that the data pump is a multithreaded application that runs under the control of the Windows NT Server operating system. Being multithreaded, the data pump takes advantage of multiprocessor architecture, which allows several processes to run concurrently.

After observing the operations of the data pump during its development, members of the MS Sales development team began the process of removing the legacy operational data systems. For each of the systems, team members rerouted data from distributors and resellers through the data pump and, once they were satisfied with the data pump

performance, they were able to remove the legacy systems that had previously handled the data.

Data Marts

The MS Sales data marts—SQL Server 7.0–based databases optimized for fast queries—are designed first and foremost to report information quickly. Fast ad hoc queries require tables, views, and indexes to be highly optimized. To achieve this, developers of MS Sales implemented its data marts in what's known as a "star schema" structure.

To design the tables, developers used report prototypes from business users throughout Microsoft to ensure the system would capture every data point that decision-makers might want to observe. For example, tables might be named Revenue Amount, Units Sold, Product, Subsidiary, Customer Name, Postal Code, TransactionDate. Developers also designed hierarchical structures for areas such as product lines, geography, and time (see more in "Relational Design"), and an organization taxonomy so that classification is clear and consistent. Such taxonomy is useful in a system that classifies entities in this case, organizations, in many different ways.

To improve Microsoft SQL Server performance, developers partitioned MS Sales data into small data sets. For example, the Fact tables are broken into separate tables that contain data for only one fiscal month. Partitioning each perspective by time also prevents the Fact tables from becoming too large.

The transaction table contains only IDs. To avoid having to change all partitions and perspectives when domain values change, developers designed the database containing the dimension tables to be shared across all partitions and all perspectives. Also known as the global-domains database, it needs to be built only once, and refreshed only once, when changes are made to domain values.

The MS Sales developers partitioned the data marts for other reasons as well. First, partitioning enables the data marts to provide a relatively small and focused set of tables rather than a much larger database system. Decision-makers can view the data from one of five perspectives depending on their role at Microsoft. Each perspective contains the Fact tables (Sales, Inventory, or Partitioned Domain tables), Product, and Organization, that best support that perspective.

Second, partitioning helps boost query performance. Because most queries are run on the most recent three months of detailed data, data that is older than this is summarized to monthly granularity. For more detail, see Figure 3, which provides a simplified view of the Sales Fact table and some related dimensions in the Reseller perspective.

For performance reasons, MS Sales developers decided to eliminate joins wherever possible. They accomplished this by denormalizing some hierarchical relationships and placing the primary keys on the Fact table and the primary dimension tables (Seller Organization, Buyer Organization, Product and Subsidiary).

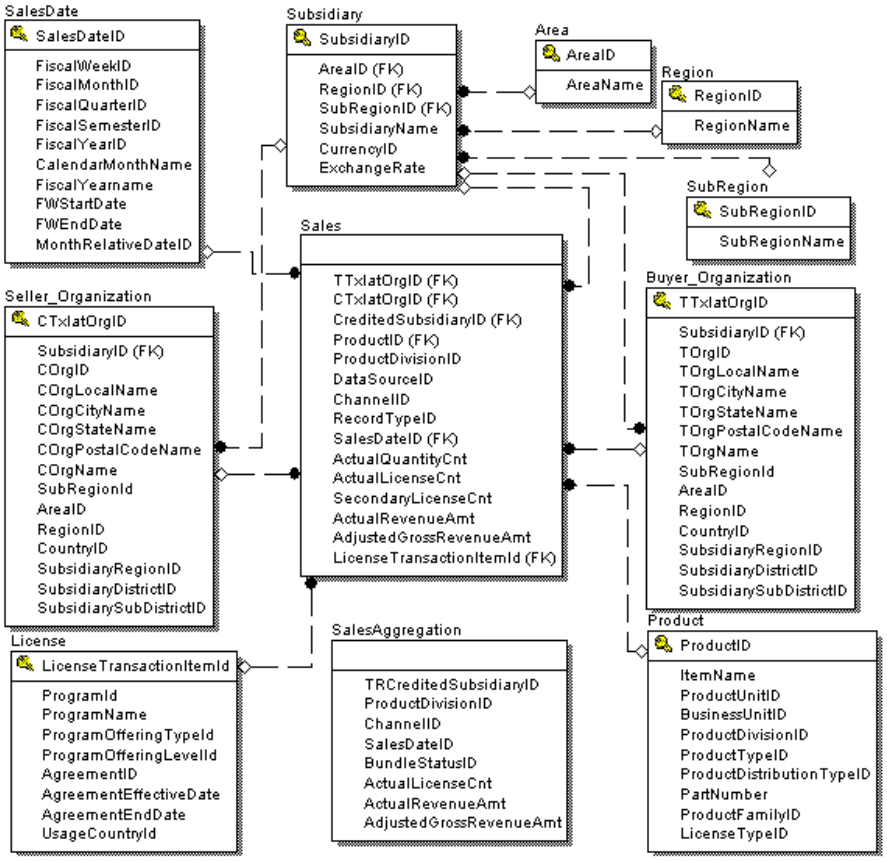

Figure 3 Data-Mart Fact Table

Because MS Sales is a financial system, the fiscal month is the primary time dimension of interest. Almost all data-mart queries focus on analyzing transactions for a specific fiscal month or series of fiscal months. To support this common retrieval grouping, data for each fiscal month is stored in a separate table. When a user queries the data mart for several fiscal months, the MS Reports user interface creates and processes a separate query for each fiscal month. The results of each data-mart query are then stored in a dynamically created worktable that is sent to the MS Reports client. (Note that MS Reports, a reporting system originally part of one of Microsoft's legacy operational data systems, is the foundation of the MS Sales front end. For more detail on how MS Reports works and the role it plays in MS Sales, see the Decision-Support tool section.)

As for indexes, they play an important role in providing efficient data access in SQL Server. To support flexible index creation, MS Sales developers created a table to store information determining how tables and columns should be indexed. When the Fact and dimension tables are being rebuilt during factory processing, the indexes are dropped and recreated based upon information stored in the table.

Factory

The factory is responsible for transforming data stored in the warehouse into denormalized structures optimized for fast ad hoc queries in the data marts. The factory

follows an extremely complex set of rules. This situation stems from the legacy operational data systems at Microsoft. Because each of these systems had its own way of processing, developers had to take this into account when they were building the factory. Their solution was to rationalize and consolidate all the rules of the various legacy systems into the factory so that accurate information could be presented in the data marts. The developers also added new rules to the factory to support new business requirements.

The partitioning and perspectives in the data marts help the factory perform many tasks in parallel that would be harder to perform in any other way.

The factory has two basic phases:

Initialization — Building the global-domain tables and dimensions without first partitioning them.

Leveling — Obtaining data for the Fact table in order to tie all of the dimensions together.

In the early days of MS Sales (long before its current SQL Server 7.0 implementation) factory processing occurred serially. While this was fine initially the solution did not scale well. As more information was added to the warehouse, factory-processing time began to take too long. To solve the problem, developers modified the factory so that parallel processing would permit leveling multiple months at a time. An initialization phase that feeds several other phases was added so as to build tables in the data marts in parallel.

The factory extracts data from the warehouse on a scheduled basis and prepares data for the data marts. During processing, the factory derives several attributes based on business rules. Because these rules change, typically at the beginning of the fiscal year, the attributes must be derived rather than stored permanently in the warehouse. This approach allows the warehouse to avoid storing and reprocessing data in order to “re-derive” values. Once a transaction is loaded into the warehouse, there is no normal situation where it would be reprocessed. The factory assumes responsibility for reprocessing data based on changes in business rules.

The factory also is responsible for transforming the normalized warehouse data into a “star schema” that’s better suited for decision-support access. At the center of the star is a table with either sales transactions or inventory snapshots for a given fiscal month. Each of these Fact tables has a single Dimension table for each of the following subject areas:

- Product
- License Item
- Selling Organization
- Buying Organization
- Time

The Dimension tables contain all the necessary attributes in a single denormalized structure. For example, the Product Dimension table has roughly 50 columns that are taken from 15 different tables of the Product subject area in the warehouse. This technique of denormalization means that Fact data can be joined with a substantial volume of Dimension data through a single join on an integer key (ProductId).

To further improve performance, the Dimension tables are partitioned along the same Time dimension as the Fact tables. For each occurrence of a Fact table (i.e., by fiscal month), there is a set of Dimension tables that contain only data for that specific Fact table. This means that a user interested in data for a particular fiscal month can work with a relatively small set of tables, rather than having to scan a few very large tables.

The complete set of factory processes are implemented in roughly 150 SQL Server stored procedures written in transact-SQL. When the factory retrieves data from the warehouse, it processes an entire fiscal month through all the stored procedures before returning to the beginning to process another month. To improve performance of this batch process, the factory processes four threads of work in parallel. The current configuration allows for three months of sales data and one month of inventory data to be processed in parallel.

The factory is implemented on a separate server from the warehouse. This architecture helps prevent resource contention when factory and warehouse processes are running simultaneously. To support factory processing, a complete copy of the warehouse database is replicated to the factory server. Using SQL Server replication, the publisher (the warehouse) sends all insert, update, and delete transactions to the subscriber (the factory). Although the warehouse is relatively large, the number of rows that are being changed or inserted is relatively small. Through SQL Server replication it is possible to move only the data that has changed from the warehouse server to the factory server.

Factory processing can take three hours or three days depending on how many months of sales are involved. The initial design of MS Sales (the pre–SQL Server 7.0 version) relied entirely on a single-threaded batch process. Since then, however, growth in data volume and complexity has required a reduction in total run time so that reports could be published on time.

Developers recently upgraded the factory server from a Compaq 5500 to a Compaq Alpha, because testing showed that such an upgrade would reduce factory run times by about 30 percent. This change was made possible by move to SQL Server 7.0, thanks to its ability to back up and restore databases between the Intel and Alpha platforms. In fact, this is how data-mart databases are moved from the factory server to the data-mart server.

Factory Process Flow

The factory consists of SQL Server stored procedures that execute the following processing steps:

Synchronization and Audit. With the help of SQL Server replication, the factory transfers data from the warehouse to a mirror image of the warehouse that resides on the factory

server. The Synchronization and Audit step ensures that replication has completed successfully and that all data is ready for processing.

Initialization — The Initialization step prepares several master tables and working tables that are used by the factory. Some warehouse tables are reformatted and indexed to support factory processing. Other temporary work tables are created to eliminate the need for the factory to process multiple times against large tables. These temporary work tables contain the minimum required subset of rows and columns to reduce the physical I/O required to access the data.

Sales and Inventory Processing — When the Initialization step is complete, Sales and Inventory Processing begins. The core of factory processing is the application of business rules and transformation of data into the star schema for the data marts.

Partitioned Domain Processing — Once the processing is complete for both Sales and Inventory Fact tables of a particular fiscal month, separate processing threads are activated to begin building the partitioned domain tables. This activity can occur in parallel with additional Sales and Inventory processing for different months.

Transport to Data-Mart Servers — When the previous two steps are complete, an automated process uses the SQL Server Dump-Copy-Load (DCL) utility to move the databases to the data-mart processing servers.

Note also that the factory maintains automated restart ability for each and every one of its processes. As each process completes, the factory records an entry in its Restart table. At the successful completion of a factory job, the Restart table is truncated. If the factory starts and finds entries on this table, then it knows it is processing in a Restart mode and will skip the processes that already have been completed. In addition to being an essential feature of the production environment, Automated Restart-ability is very helpful while the system is in development and errors are encountered frequently.

As the factory processes data, it records statistics after each SQL statement or major processing event (e.g., the end of a stored procedure). These statistics are recorded in a SQL Server table and include attributes such as:

- Event – Insert, Update, Delete, Select Into
- Object – Table, View or Index
- RowCount – Rows affected
- Status – Success or Error
- Timestamp

Several queries and views have been written to provide a quick insight into the current status of the factory and also a historical view of the processes that have occurred. Through such information MS Sales developers can obtain an up-to-the-second view of what processing is occurring. MS Sales developers also can monitor data volumes and processing times and identify trends that may identify queries that need optimization. The factory system includes a number of audits designed to detect “out of balance”

conditions and issue appropriate warnings. In some cases the factory shuts down if key check totals cannot be reconciled.

Decision-Support Tool

Developing a decision-support tool (including an intuitive user interface) while abstracting the physical database design is key to lowering support costs and increasing user satisfaction. Prior to MS Sales, Microsoft decision-makers had to rely on a number of different decision-support tools. To view information residing in the various legacy data systems, decision-makers typically had to use various tools.

Out of those legacy systems, a decision-support tool known as MS Reports was considered a favorite among decision-makers. For this reason, decision makers working on MS Sales have used MS Reports as a decision support tool for MS Sales. To better understand how this tool is used, consider first the workings of the decision support tool.

MS Reports was developed as a Visual Basic-based application, which interacts with Microsoft Excel. Launching MS Reports from within Microsoft Excel enabled decision-makers to take advantage of Microsoft Excel while performing ad hoc queries against the MS Sales data warehouse.

An important aspect of the MS Sales architecture is the level of physical abstraction made possible through the MS Reports front-end. MS Reports uses a metadata layer—a description of the data-mart schema and data dictionary, to map fields within the user interface to tables and columns in the data marts. When a database structure changes in the MS Sales data marts, a production-support team updates the metadata. Having this metadata layer between MS Reports and the data marts helps to ensure that decision-makers will not be affected adversely when necessary changes are made to database structures of the data marts.

MS Reports connects to the metadata database residing on a middle tier via ODBC and downloads metadata for its own use. Metadata is retrieved by SQL Server stored procedures invoked from within the MS Reports client. MS Reports uses the metadata to display information requested by the user.

The MS Reports user interface consists of multiple forms and controls, which allow users to perform ad hoc reporting using an intuitive user interface. Data is added to ad hoc queries as the user drags and drops fields onto a form.

MS Reports associates a transact-SQL LIST statement with each field and then executes the command to retrieve a list of domain values that can be used to populate a graphical list box. Fields the user would like to appear on the report are later associated with a transact-SQL WHERE clause.

MS Reports displays results by generating transact-SQL statements based on user interactions. MS Reports uses ODBC to send those transact-SQL statements to SQL Server 7.0. In turn, SQL Server, running on a data-mart server, receives the transact-SQL statements, executes each of them, and uses ODBC to return the results back to MS Reports. Transact-SQL statements are generated by MS Reports with information

maintained within the metadata. This ensures that simple modifications to transact-SQL statements are possible without requiring that MS Reports be completely rebuilt.

MS Reports uses a component called the “query resolution engine” to generate transact-SQL. This component supports aggregate navigation, merging of heterogeneous sub cubes, horizontal partitioning, as well as other query techniques commonly used to traverse a star schema. This component resides on a “middle tier” to minimize processing on the client computer.

MS Reports stores query results in a temporary SQL Server–based table by invoking several transact-SQL INSERT statements. MS Reports then writes results to the temporary table quickly by establishing several ODBC sessions with SQL Server and using each of them for concurrent execution. MS Reports displays query results to the user through either Microsoft Excel or Microsoft Access. After the user has made his or her selection, MS Reports issues the transact-SQL statement SELECT to retrieve the query results from the temporary SQL Server table.

A project is currently underway to base MS Reports upon the integrated OLAP services included in SQL Server 7.0. See the appendix for information on how ITG is putting OLAP services to use.

Scalability Planning

Planning for growth in any data warehouse is key to providing the up-to-date information decision-makers need now and into the future. As mentioned earlier, SQL Server 7.0 has simplified at least one aspect of such planning for the MS Sales developers: the migration from Intel architecture to Alpha architecture. SQL Server 7.0 running on an Alpha can load SQL Server 7.0 databases saved using an Intel-based system.

The Microsoft IT group also has found that SQL Server 7.0 and Compaq Alpha equipment together can provide a highly scalable and reliable platform on which to build data warehouses. What they’ve learned is that Alpha-based storage-area networking and SQL Server 7.0 working together can decrease network traffic and the time needed to perform SQL Server DCLs. Moreover, network access between servers running SQL Server is no longer required to move data between them. Exported SQL Server data can be imported directly from one SQL Server-based database into another that shares the same storage-area network bus. Transferring data across the system bus is much faster than transferring it over a network. With all this in mind, the Microsoft IT group is using SQL Server 7.0 and storage-area networking in creating other large production data warehouses.

Another advantage of SQL Server 7.0 in terms of data-warehouse scalability is replication. Replication makes it far easier to support a data warehouse based upon three-tier architecture. In MS Sales, for example, SQL Server 7.0 regularly replicates data from the warehouse database to the factory database. SQL Server 7.0 replication handles data more economically by moving only modified data. Moving the smallest data sets possible reduces unnecessary network traffic, frees up the data bus, and takes a load off the CPUs.

In future versions of MS Sales, developers will use SQL Server 7.0 replication to support multiple-factory parallel processing. Having several options available for dividing SQL Server–based processing among many servers helps to ensure that MS Sales will scale to meet the increasing demands of the future.

Partitioning and perspectives are other areas in which developers created MS Sales with the future in mind. For example, they partitioned data in MS Sales based upon the way SQL Server handles data sets. SQL Server handles smaller data sets concurrently more efficiently than millions of rows in tables at once. A separate machine for the factory ensures additional scalability for MS Sales because the processes of loading data and building reporting tables do not reside on the same box.

In the near future, MS Sales perspectives will be moved to the Microsoft OLAP services included in SQL Server 7.0. Data Transformation Services (DTS), also included in SQL Server 7.0, will be used to process cubes on a daily basis, as often as the factory runs. This approach will allow a parallel path to refresh the cubes.

Monitoring and Maintenance

Developers designed MS Sales so that its factory component would monitor and verify the accuracy of revenue figures during the transformation of information from warehouse to data mart. If revenue information is missing at any point along the way, the factory stops processing or repeats its last process. This approach is designed into MS Sales because stopping and restarting the factory is considered less problematic than delivering erroneous data to the data marts.

Also part of the design of MS Sales is the computing and storage of procedure history. According to this design, a transact-SQL procedure registers information about itself in a table, once when the procedure starts and once when it ends. At any time, developers can use the data in this table to create statistics. This approach ensures that information is always available for the purposes of “tuning” procedures, which tend to consume a large percentage of run time.

Conclusion

SQL Server made the removal of 18 legacy systems possible, by bringing the information together into one highly scalable solution, MS Sales. The move of MS Sales into a SQL Server 7.0 environment is part of an ongoing process. At the heart of this process is the search for ever-better ways to provide information to Microsoft decision-makers. Because of SQL Server 7.0 MS Sales is a powerful tool for finding patterns, discovering opportunities, and creating an accurate and highly detailed picture of customer buying patterns and preferences. With such information, Microsoft is better positioned to capitalize on opportunities and respond to customer needs.

INTEGRATION CASE STUDY: SAP R/3

When a growing business is forced to rely upon dozens of diverse computer systems, each with a complex custom interface, that business is bound to encounter increased operating costs and reduced ability to access valuable information. Prior to implementing a new ERP system based on Microsoft SQL Server, Windows NT Server and SAP R/3, Microsoft found itself in just such a position. The company depended on more than 50 legacy computer systems based on technologies ranging from Intel Pentium to IBM AS/400 and Digital Equipment Corporation VAX. Each system had its own method of processing, sorting, filing, and recording information about the general ledger, capital assets, finance, procurement, human resources, and order management/operations.

In such an environment, consolidating information in a way that would provide a unified "snapshot" of the company at a given point in time was a tedious and costly process. Moreover, although Microsoft's legacy Intel, AS/400, and VAX systems could communicate through a complex set of custom interfaces, the aggregate system was far from tightly integrated. Indeed, Microsoft analysts estimate that more than 90 percent of the 20,000+ batch jobs needed to retrieve disparate bits of information in that environment increased the complexity of the systems without providing an easy or efficient way to obtain the information that company managers needed.

Correcting this problem was the number one priority when Microsoft set out to overhaul its ERP system. Microsoft executives recognized that the corporation needed an altogether new solution to support the core business processes. As they envisioned it, such a solution would be tightly integrated, highly scalable, and capable of eliminating redundancies and the multiplicity of high-maintenance low-value systems.

After evaluating a number of available ERP systems, Microsoft executives chose not to rely on the company's existing VAX and AS/400 technologies. Instead, they chose an integrated solution based on SQL Server (as the database engine) and SAP R/3, running on Windows NT Server.

Their choice of SAP R/3 was based on a number of factors. First, SAP R/3 was a scalable, multinational system supporting international language and currency as well as broad and integrated business functions, including electronic commerce. Second, SAP R/3 offered a true, open, and highly scalable and very network friendly three-tier client/server infrastructure. Third, SAP R/3 was a known entity. For example, SAP had accumulated considerably more experience integrating SQL Server and Windows NT Server than most any other ERP system provider.

As for hardware, the executives chose another entity that was well-known within Microsoft: an Intel architecture–based platform from Compaq Computer Corporation. Finally, for implementation consulting, they chose Deloitte & Touche Consulting Group/ICS.

The project to implement the new SAP R/3–based ERP system took about 36 months, and when it was done, the benefits were apparent almost immediately. With a highly scalable, three-tier solution based upon SQL Server , Windows NT Server, and SAP R/3, Microsoft now closes its books within four or five days, whereas that task sometimes required four weeks previously. Moreover, by moving its financial, human-resources and

order-management systems off the legacy platforms, and implementing several web-based bolt-on front ends, Microsoft is saving more than $5 million in annual hardware maintenance fees and $15 million per year in reduced procurement costs.

Implementing such a solution—a tightly integrated, three-tier system at the heart of a business—requires careful planning as well as a thorough understanding of both businesses and technology requirements. In the following pages we discuss the approach taken by the Microsoft IT Group (ITG), the steps taken, and the lessons learned in implementing this new solution.

Goals for the ERP System

Prior to selecting, configuring, and implementing the new ERP system, ITG defined goals based primarily upon requirements for information as well as ease of implementation, management, and support. In particular, the executives wanted an ERP system capable of doing the following:

- Accommodating a diverse set of international language and currency requirements
- Making possible a single taxonomy
- Enabling the removal of legacy systems through consolidation and centralization

Accommodating Diverse Languages and Currencies

Before, the Microsoft ERP system relied on more than 50 legacy systems as a result of the diverse language and currency requirements of the company's various subsidiaries. In seeking to accommodate such diverse requirements, ITG wanted comprehensive integration so that the aggregate system would be easy to support, maintain, and upgrade later.

Their motivation for such a system was based on their experience with the prior ERP system. Under that solution, programmers had to customize AS/400 MAC-PAC code to meet the statutory requirements of each subsidiary. While this approach did accommodate a diverse set of requirements, it also resulted in each subsidiary having slightly different MAC-PAC implementations.

Moreover, managing, maintaining, enhancing, and upgrading legacy systems grew increasingly challenging as needed changes were performed regionally. To implement new MAC-PAC versions, for example, programmers had to be sent out to each subsidiary for a minimum of six or eight weeks at a time – sometimes much longer. And once programmers had completed their project they would have what basically amounted to a new MAC-PAC to work with.

As new versions of the MAC-PAC needed to be deployed, ITG staff had to track and merge the diverse set of subsidiary specific requirements into each new MAC-PAC release. While ITG had little trouble in merging in the changes, the process of upgrading MAC-PACs might take as many as 18 months to complete.

In order to meet the project objectives, any new ERP system would need to accommodate all subsidiary requirements within a single central system - no matter how

diverse the requirements. A few subsidiaries had to change business practices in cases where implementing such practices might add too much complexity to the new ERP system. The new unified ERP system would benefit both ITG and subsidiaries because it would be managed centrally and accessed globally at the same time.

Making Possible a Single Taxonomy

As ITG planners and analysts compared information stored in diverse systems implemented by disparate programmers over a long period of time, they discovered inconsistencies in the way information was defined. The fact is, the various legacy systems had evolved to use vastly different taxonomies, which in turn made intersystem information-sharing all the more difficult.

For example, many subsidiary had their own data definitions to store table names, field names, and record names. This made it difficult to compare information between subsidiaries, and it was particularly complex, time-consuming, and costly to obtain a corporate picture by rolling up and consolidating information from all subsidiaries.

To address such problems, ITG decided to standardize the corporate taxonomy used in financials, human resources, and order management. This meant the new ERP system would have to handle the currency, language, tax, and statutory requirements of dozens of countries. In order to effectively make use of a third party application, the new ERP solution also would have to support regional needs with very little customization.

Enabling the Removal of Legacy Systems

One of ITG's primary goals for the general ledger in the new ERP system was the ability to file information within a single up-to-date system so that worldwide consolidated information could be presented consistently and quickly. The ability to provide a unified view of how the corporation was doing as a whole was key in determining which ERP system ITG would implement.

Here, the challenge was that each subsidiary defined financial codes differently. This resulted in complexity when it became time to roll up reports into a unified general-ledger report. This also made it very difficult and time-consuming to compare information between subsidiaries. Further, once the information was no longer timely, it lost some of its value.

ITG's goal was to implement the corporate general ledger on a single centralized ERP system, which would simplify the process of providing necessary information and make that information available much faster than before. Toward this end, the finance group wanted to consolidate various systems ranging from ROSS financials on a VAX to 12 different versions of MAC-PAC AS/400. At the corporate level, executives wanted a financials system that was highly scalable, centrally managed, and globally accessible, in contrast to the then-current distributed system.

ITG also determined that consolidating diverse systems would benefit individual departments. Human resources (HR), for example, needed a solution that would enable the department to obtain an accurate count of all employees worldwide quickly, to account for employee location and cost center, and to increase user productivity and

efficiency. As ITG saw it, this meant the new ERP system would need to support intranet-based decision-support tools.

Strategy and Planning

For ITG, formulating a comprehensive implementation strategy was key to completing the ERP implementation on schedule. Because of the global nature of the solution, ITG enlisted individuals from corporate headquarters in Redmond, Washington; from Europe; and from the Far East to help with the planning. Among them were individuals who understood both the business requirements and the tactical details of managing such a project.

Creating the Implementation Team

ITG decided to form two teams to create the implementation parameters: a core team and an extended team. The core team focused on the detailed implementation and configuration within SAP R/3, designing business processes around the system and planning for any needed organizational changes. The extended team focused on analyzing the business side, determining gaps between the solution and the business requirements, and change management.

Sixty-five individuals, including business users, system experts, and external consultants, joined forces at corporate headquarters in Redmond, WA to work together. As ITG learned, it was a good decision to involve both business-savvy and technology-savvy people on the same team.

While the core team focused on eliminating obstacles, and communicating to upper management, the extended team provided support by acting as content experts playing “devil’s advocate” for all business decisions. The extended team also was responsible for testing new solution components and training users.

Planning the Implementation

Planning the implementation meant creating a phased approach toward implementing various SAP R/3 modules, which correspond to different but related business areas.

Also as part of the planning, ITG staff would conduct a test conversion. Through this, planners could know in advance what essential reports would look like so that modifications to business practices could be kept to a minimum. Deloitte & Touche Consulting Group/ICS staff helped ITG to plan the financial business processes and accounts-payable business and procurement processes and model those inside the SAP R/3 context.

ITG planned to implement its new ERP system in three primary phases: first, finance accounts-payable and procurement; second, human resources; and third, sales and distribution. Each phase, which would require up to 18 months to implement, represented the implementation of a set of modules. For each of these modules, ITG created a configuration plan and an implementation plan in order to keep the project on schedule.

Configuration plan — This plan supported the strategy for transitioning an SAP R/3 module from a development environment into a quality-assurance and test environment and then finally into production. ITG decided to use two additional environments in this way so as to minimize user impact when changes to the system were performed. The ITG-SAP team only allows changes to production on a quarterly basis in order to keep the system stable. While this approach was more costly from a capital-acquisition perspective, ITG determined it was necessary to ensure that quality and testing were built into the implementation strategy.

As part of the configuration planning, ITG learned a lot about how SAP R/3 and SQL Server were integrated. For starters, ITG found that the integrated system handled many of the functions that traditionally were performed by database administrators, thereby simplifying operational support.

Implementation plan — This plan supported the creation of the interface architecture, which describe what was needed so that diverse systems could transfer information to the ERP system. As part of this plan, ITG determined how they intended to use ABAP/4 (the language used by SAP R/3) to implement the interfaces. ITG determined that some ABAP/4 coding would be required, so they planned for specifications that would later be needed to implement the programs. The implementation plan also sought to answer the following questions:

- Would ITG seek to improve business processes through SAP R/3 directly or by customizing SAP R/3 to accommodate regional specific requirements?
- How would SAP R/3 business processes work, and what would its various reports look like?

The implementation plan called on ITG to keep the solution as simple as possible by limiting the use of ABAP/4 for customizing the ERP system. Whenever possible, ITG would use out-of-the-box configurations, and ABAP/4 coding would be performed only when necessary to closely model existing business processes inside SAP R/3.

Implementation Process

After putting a solid plan into place, ITG began implementing Microsoft's new ERP system. First, ITG configured the hardware and software installations. Second, ITG would configure the SAP R/3 modules to support the required business processes within the new ERP system. Third, ITG converted data already stored in the legacy systems into a unified and consistent format. This process consisted of developing interfaces to extract the information along with programs to perform the conversion. Fourth, ITG entered that data into the new solution, leveraging the test system for many practice runs of the conversion programs. Later ITG would upgrade the back-end database to the latest version of SQL Server, so that the system could accommodate the mission-critical performance and reliability requirements considering the significant database growth expected.

Configuring Hardware

Configuring hardware for the SAP R/3 solution meant that ITG had to deal with 39 separate servers. This was largely because Microsoft required a sizable number of servers to support several large projects prototyping new functionality in parallel, dedicated training environments, worldwide language requirements, significant levels of batch processing and web connections through DCOM. The project originally began with six ProLiant 4500 (4x133 MHz) test application servers, each with 512 megabytes of RAM and 12 gigabytes of hard disk space.

Production environment — The production environment started with two Compaq ProLiant 5000 (4x200 MHz) computers with 2 gigabytes of RAM and 508 gigabytes of hard disk space. One served as the production SQL Server 7.0 database server, and the other served as the hot-backup database server. ITG used the log shipping software included in the BackOffice® Resource Kit to synchronize transaction logs between the production server and a hot standby. These computers were connected to 16 application servers: six Compaq ProLiant 4500s (4x133 MHz), five ProLiant 5000s (4x200), and five ProLiant 5500s (4x200). Each of these computers had between 512 megabytes and 1 gigabyte of RAM.

Development environment — The development database server was a ProLiant 4500 (4x166 MHz) computer with 1 gigabyte of RAM and 88 gigabytes of hard disk space. That computer was connected to two ProLiant 4500 (4x133 MHz) development application servers, each with 340 megabyte of RAM and 16 gigabytes of hard disk space. Also part of the development environment were two additional ProLiant 4500s (4x133 MHz) acting as central instances for project prototyping and repetitive upgrade testing and each of which has 512 megabytes of RAM and 250 gigabytes of hard-disk space.

Testing environment — The testing environment consisted of two Compaq ProLiant 5000 (4x200 MHz) test database servers, each with 1 gigabyte of RAM and 152 gigabytes of hard disk space connected to eight Compaq ProLiant 5000 (4x200MHz) test application servers, with between 512 MB and 1 GB of RAM..

Configuring Software

ITG configured the software in the new SAP R/3 system with the primary goal of making the new system easy to administer. For this reason, members of the implementation team configured SAP R/3 so that it would handle many of the functions that are traditionally the responsibility of a database administrators:

- SAP R/3 is directly integrated with SQL Server using OLEDB and a database software layer. This ensures that SAP R/3 understands the SQL Server 7.0 database schema and tables and always “knows” where to obtain information.
- They implemented SAP R/3 in such a way that it would be able to manage the process of database-schema modification.

- They configured SAP R/3 so that it would manage security at the application level. This involved requiring only one secured SQL Server system account through which all SAP R/3 database tables and records are accessed.
- They configured access to SQL Server data at the SAP R/3 application layer so as to provide consistency to users through a requirement that all database transactions be performed in this way.

In addition to making the new SAP R/3 solution easy to administer, members of the ITG implementation team wanted to ensure the highest system performance possible. Toward that end, they did the following:

- They ensured that SAP R/3 code would be executed on application servers dedicated to a single functional area such as finance or human resources.
- They distributed the workload in a way, which would make the most efficient use of SAP R/3 data and executable cache. This was accomplished by grouping SAP R/3 users who would be likely to perform similar transactions on the same application server.

Converting Information Stored in Legacy Systems

Members of the ITG implementation team decided that in order to perform a successful data conversion, they would need to move the basis of their financial structure from departments to cost centers. Toward that end, ITG created conversion programs to eliminate data duplication in legacy systems supporting finance, accounts payable, and procurement. In particular, ITG expected to effect a 15 percent reduction in the volume of data stored in vendor accounts, which were distributed across a number of disparate AS/400 systems. To do this, ITG transferred vendor information to a SQL Server database and churned through it using transact-SQL.

Entering Information into SAP R/3

Members of the ITG implementation team found that once they had the conversion process in place, it was relatively easy to enter information into SAP R/3. For each AS/400 system, they exported data into temporary ASCII text files and then converted it into the new SAP R/3 solution. (To prevent future inconsistencies, the team also stopped maintaining vendor data in the legacy systems as soon as the data was transferred into SAP R/3.)

Throughout the information-entry process, the team relied extensively on the SAP R/3 software development kit. For example, with the SAP R/3 DCOM connector, the team automated the entry of information that originated in Microsoft's intranet-based procurement system. The kit also enabled ITG to develop console-based utilities to "batch up" information so that it could be entered at regular intervals. By entering data at regular intervals instead of through a continual stream, ITG was able to avoid disrupting scheduled maintenance on the SAP R/3 servers.

ITG performed almost all its information-entry coding with the SAP language ABAP/4, instead of using transact-SQL statements. This is because SAP R/3 integrates with SQL

Server directly (using OLEDB) and generates its own transact-SQL statements for entering information into the SQL Server databases. This is also because SAP R/3 takes responsibility for a variety of information-entry tasks. For example, it keeps track of the SQL Server database from which it obtains data and of the table relationships, it "understands" the SQL Server database schema, and it ensures referential integrity across tables.

Configuring SAP R/3 Modules

The next step in the implementation of the new SAP R/3 solution was to configure SAP R/3 modules. To do this, ITG modeled business practices and created interfaces that could be used to transfer information from legacy systems into SAP R/3. For this task, ITG relied on the Deloitte & Touche FastTrack for SAP methodology. This was a basic framework for scoping and planning the implementation; identifying targets; redesigning existing processes as necessary; and configuring, testing, and delivering each new process in the SAP R/3 environment. As part of this work ITG defined business processes, the scenarios in which the processes take place, and the scripts that create the business rules that manage the processes within the systems.

Within seven months, the ITG implementation team designed, configured, tested, and went live with the procurement and payables aspects of the SAP R/3 materials-management and financial modules necessary to the operations of the finance groups in March 1996. Three months later the consolidated worldwide General Ledger was converted onto R/3, and rollouts to worldwide subsidiaries and the human resources and order management projects began.

Upgrading to SQL Server 7.0

ITG began the SAP R/3 implementation with SQL Server 6.5 as the database back end, and converted to SQL Server 7.0 in July 1998. As database volumes exceeded 130 gigabytes, database consistency checks (DBCCs) under SQL Server 6.5 began to require nearly 100 hours to complete which was an operational risk. There also were problems with concurrency, deadlocking, and online-backup performance as the database grew so large. Throughout the original implementation, ITG worked very closely with SQL Server developers so that such problems would be addressed in the next version of the product. As soon as that version, SQL Server 7.0, reached a final beta release, ITG upgraded their production SAP R/3 databases to it. SQL Server 7.0 is designed to fully support databases more than 10 times the size of SQL Server 6.5.

To manage the upgrade process the ITG team relied heavily on a conversion kit and documentation provided by SAP. Along with this kit, the team ran the SQL Server Conversion Wizard to convert the SQL Server–based databases to version 7.0 of that product and then installed a number of newer SAP R/3 components that took advantage of the new SQL Server features.

Monitoring and Maintenance

After completing the first implementation of the SAP R/3 ERP system in 1996, the ITG support team focused on improving system monitoring and maintenance. Monitoring

involves ensuring that users will be able to efficiently and effectively use the system, and maintenance involves activities ranging from SQL Server database backups to performance tuning.

For example, every 10 minutes, the SQL Server transaction logs are backed up to a hot-standby server using the log shipping utility included in the BackOffice resource kit. If there's ever a failure in the primary server, ITG can switch to the secondary server. (Fortunately, ITG has never had to perform this operation, but it keeps the option open so to avert any potential disaster.)

For SAP R/3–specific maintenance, ITG negotiated with the business one hour per week of down time. Other systems that exchange information with SAP R/3 are configured specifically so they do not attempt to perform those operations during scheduled down times. During most weeks this is not necessary, and the R/3 servers are left running during this window – the system currently averages one reboot per month.

For database backups, ITG performs backups directly to striped tapes. There are no performance impacts from backing up SQL Server 7.0 databases while in use. Tape backups are generated each night and stored off site, in another facility. As an added precaution a computer facility 30 miles away from the production SAP R/3 servers houses the hot-backup system replicated over SONET lines. In the event of a major system problem at the Redmond facility, the remote facility can provide uninterrupted SAP R/3 support to Microsoft personnel.

As for the monitoring of system utilization, ITG does this with the help of CCMS and other tools included with SAP R/3. During peak utilization of the production instance (which occurs when 500-600 users access the ERP system and thus establish roughly 1,000 connections to SQL Server 7.0 using OLEDB), ITG uses CCMS to measure dialog-response times and to update response times over a period of time. ITG uses the information derived from these measurements for modeling future system requirements. ITG also uses Windows NT Performance Monitor, which is included with Windows NT Server 4.0, to monitor SQL Server 7.0 and the network operating system.

ITG has modeled the production system of the SAP R/3 ERP system in a stress test environment so as to more easily determine performance levels several months in advance. This approach is especially important when ITG needs to add functionality to the system. For example, using information obtained from CCMS, ITG can identify the performance impact of configuration changes to user transactions. ITG can then benchmark those transactions under a heavy load using a third-party testing tool along with the SAP DCOM connector and the SAP R/3 software- development kit.

Conclusion

With its ERP system based on Windows NT Server 4.0, SQL Server 7.0, and SAP R/3, Microsoft has a highly scalable and unified system around which all its mission-critical business processes can rally. With the help of this solution, Microsoft has been able to provide consistent information to 180 subsidiaries, reduce its dependence on highly specialized and costly database administrators, eliminate its need for VAX and AS/400 experts, and to cut administrative and management costs.

For example, the new SAP R/3 solution has enabled Microsoft to eliminate the data and account duplication and inconsistencies that were common when the business relied on diverse legacy systems. Consequently, the interfacing burden is much smaller, and most interfacing today is done right within SAP R/3. This means that when accounts payable are posted, the financial transaction is instantly reflected in the general ledger without anyone having to run an interface program.

Moreover, data definitions are stored in a central SAP repository. Information is published in one place, in one consistent format, and extracted only once from SAP R/3 and published to a SQL Server data warehouse. From this warehouse, systems that need a particular type of data (such as general-ledger accounts) can obtain it at regular intervals in a single, consistent format.

For Microsoft's HR and finance departments, the new SAP R/3 solution has vastly simplified head-count tracking and reporting. For example, more than 100 HR professionals routinely access the system from subsidiaries around the world and more than 25,000 Microsoft employees now have access to their HR data by way of the corporate intranet.

Other benefits are a direct result of the new SAP R/3 solution being based on version 7.0 of SQL Server. Under SQL Server 7.0, the new SAP R/3 solution provides higher performance from the point of view of the users. Average screen-update time is now less than 0.5 seconds, down from 1.5 seconds for the original version of the SAP R/3 solution running under SQL Server 6.5 . This improvement comes largely because SQL Server 7.0 accesses data more efficiently and as a result of row-level locking, which reduces the lock footprint on the database and user contention for data. Yet another benefit comes from the move away from single-threaded SAP R/3 updates, which were used to help minimize concurrency problems under SQL Server 6.5. Now, updates are performed from 10 independent processes, time needed to perform batch processing has decreased by 45 percent.

Further, with the new SAP R/3 solution based on SQL Server 7.0, cost-center definition and internal order definition for general-ledger accounts are being maintained centrally. This enables Microsoft's many subsidiaries to conform to a single taxonomy, which in turn makes it easier for ITG to code interfaces and publish information about interface structures.

While these benefits are impressive, it's important to remember that most of them appeared even after just a few months of implementation of the new SAP R/3 solution. Both ITG and executives throughout Microsoft look forward to the kind of long-term benefits that such a centralized system is ideally suited to provide.

CONCLUSION

Lessons Learned

As part of planning, developing, deploying, and supporting SQL Server 7.0–based applications such as MS Sales and the SAP R/3 ERP system, ITG gained a good deal of real-world experience. In this paper, ITG is sharing the lessons learned from that experience so that readers can apply the lessons, where appropriate, to their own environments. For the ITG implementation team, below are a few of the most essential lessons that were learned:

Lesson 1: Build a Clear Support Structure

To increase availability and simplify troubleshooting, a strong operational support structure and clear escalation paths are needed.

For ITG, one of the best ways of increasing availability was to clearly define the roles and responsibilities of various members of the SQL Server support staff and to maintain a clear escalation path for problem-solving. For example, at lower levels, members of the operations staff perform monitoring and maintenance, while at upper levels, engineers benchmark future solutions. Any issue that needs resolution follows a vertical escalation path to an individual at the most suitable level to resolve the issue. This approach ensures that necessary maintenance can be performed quickly and that future solutions can scale to meet the increasing demands of a growing information infrastructure.

Lesson 2: Teamwork Means Working *Together*

To build a successful application, keep the members of the team—both the business and the technical people—collaborating as closely as possible.

Members of the ITG implementation team have found, on this project as on many others, that attempting to build systems without thoroughly gathering essential business requirements is like working in a vacuum. A related problem is attempting to build systems with a team that's distributed far and wide.

On the MS Sales project, for example, teamwork was hampered because the staff members responsible for implementing the project were geographically divided and reported to different management. When problems arose, they had to be escalated to a very high level before they could be addressed. In response, ITG management relocated members of the MS Sales team to the same building. The result was a dramatic increase in collaboration and productivity.

Lesson 3: A Single Taxonomy is Stronger

To successfully share and manage information across multiple servers, a single, unified, and centrally managed taxonomy is ideal.

In both the MS Sales and ERP environments prior to the move to SQL Server 7.0, the existence of multiple taxonomies across diverse systems required that ITG maintain a number of complex interfaces across the globe. For example, ITG had to implement each system slightly differently to account for diverse language and currency requirements. By taking advantage of the scalability of SQL Server 7.0, ITG consolidated

information from a wide variety of sources, created a standard taxonomy, and simplified the sharing of information across its global enterprise.

Lesson 4: Use the Power of Transact-SQL

To optimize transact-SQL code, take advantage of transact-SQL stored procedures.

Transact-SQL code can play a major role in the development of large data warehouses such as that required by MS Sales. For this reason, the members of the ITG implementation team routinely audit the performance of transact-SQL stored procedures. By recording the time needed to execute each transact-SQL stored procedure, ITG can optimize their performance. Having such a process in place can help mitigate the risk associated with implementing changes to the system. (See the appendix for information on one of the monitoring tools that ITG uses to optimize transact-SQL performance.)

Lesson 5: Maintain a System of Checks and Balances

To minimize the risk of propagating erroneous information through complex time-consuming transformations, establish a strong system of checks and balances.

To help ensure that MS Sales users are working with accurate information, ITG developed a method of tracking data as it's transformed from the MS Sales warehouse server to the MS Sales data marts. If an error occurs during this transformation, the tracking mechanism ensures that MS Sales factory stops processing the data. Note that errors are often the result of designing or transact-SQL coding a defect into the system and then putting the erroneous functionality to use in production.

Lesson 6: Catch Small Problems Before They Become Big Ones

In most cases, a small problem can be stopped from growing into a big one through a few, highly directed troubleshooting tricks.

Throughout the SQL Server 7.0 implementation and as a matter of everyday best practices, ITG relies heavily on the information available in the Microsoft Knowledge Base located at http://www.microsoft.com particularly for resolving errors appearing in the Windows NT Server 4.0 System Event Logs. Generally, ITG tries to use the Knowledge Base for resolving such events before escalating issues to more technical engineering staff.

Future Direction

As the members of Microsoft ITG look to the future, they face the challenge of increasing the value of information even as that information is increasing in volume. They also face the challenge of serving an ever-increasing population of users. In contrast to the traditional IT role of serving only a select few decision-makers, the members of today's ITG must provide information to thousands of Microsoft employees. This means that ITG must pay special attention to issues of scalability and manageability in all the solutions it builds.

Hardware Challenges: Scalability and Manageability

As of mid-1999, the MS Sales solution encompassed 75 disks having over 1 terabyte of available storage, yet even at that size the solution is predicted to grow at a steady pace. With this in mind, hardware experts, engineers, and planners throughout ITG are preparing for the simultaneous challenges of scalability and manageability. For example, to enhance manageability ITG consolidated dozens of servers in late 1998 and early 1999, with the result of increasing per-server storage needs by 50 percent and concentrating ever-large portions of the business on a smaller number of servers. Such a move makes it all the more crucial that ITG pay careful attention to the scalability of its hardware, particularly any weak points in design that might result in server downtime.

To ensure that purchased hardware will scale to meet future demands, ITG standardizes on all hardware while simultaneously certifying that particular hardware will be able to support particular business solutions for a number of years. The idea here is to keep the burden of hardware-scalability issues on top system engineers instead of on the solution designers.

Redundancy is another high priority for the ITG server hardware. To meet the high-availability demands required of heavily consolidated servers, ITG is preparing to increase its redundancy requirements to cover “everything” so that a single component failure cannot spread system-wide. Two primary examples of highly redundant systems that ITG continues to evaluate include hot-swappable power supplies and hard drives.

ITG also is preparing for an ever increasing volume of data, to ensure it can be shared among many different computers. For example, if several hundred gigabytes of data need to be transferred between servers, ITG must have a fast method of doing this. This is an issue particularly germane to the MS Sales solution. In it, the data warehouse is divided into components, each of them running on dedicated hardware and requiring access to a large data set. This is so that the data can be transformed from the MS Sales warehouse into the MS Sales data marts. In the future, the solution will need an extremely fast way to share such information.

To support the critical data-sharing needs of the hardware supporting SQL Server–based solutions in the future, top system engineers within ITG are preparing for what is on the horizon. Hardware features currently being evaluated include:

Split mirror sets enable a server to logically own a local RAID 1 set of disks at one point in time and then later allow the assignment of half the disks to one computer and half the disks to another.

Remote data mirroring is similar to split mirror sets except that it is suitable for disk sets located remotely—500 meters apart or more. One drawback is that remote data mirroring can result in expensive propagation delays. This raises the question of whether a system using remote data mirroring can support sustained writes at 100 megabits per second. This means that ITG (just as any other large corporate IT department) must weigh the risks against the benefits of being able to manage disks in one area of the data center and computers in another.

LUN masking enables disk controllers to filter server access to volumes that are assigned to a specific server. For example, when the Windows NT Server 4.0 operating system boots, it mounts every disk that it detects. While this behavior is desirable when storage is local, it's undesirable across shared Storage Area Networks (SANs), which are becoming common at Microsoft. LUN masking makes SANs more effective by concealing a SAN volume from systems that should not have ownership of the drive set.

Software Challenge: Efficient Analysis

Supporting the hardware requirements of an ever-increasing volume of information is just one part of the challenge ITG faces. The other is making that information serve its purpose: to help executives and other Microsoft employees make the best business decisions they can. At ITG, one key to addressing this challenge is the Microsoft OLAP Services that are included in SQL Server 7.0.

For example, ITG uses OLAP Services to help users of Microsoft Excel to quickly and easily analyze large volumes of information in the company's Management Reporting System (MARS). MARS is a data-warehouse solution used by corporate decision-makers to generate profit-and-loss statements, balance sheets, and other internal statutory and tax reports.

In the past, MARS generated reports through MS Reports, which returns all information within an underlying pivot. This caused the MARS data marts to become less and less efficient because as the volume of information grew, users had to download an ever-larger volume of data in order to perform ad hoc queries. By replacing MS Reports as the report generator with OLAP Services, ITG has dramatically streamlined operation of the solution. This is because with the help of OLAP Services, MARS returns only the information that decision-makers need. Based upon how beneficial ITG has found OLAP Services to be, plans are underway to put the technology to use in many other facets of the company. (For more information on how ITG is implementing OLAP Services on MARS, see the Appendix.)

Software Challenge: MS Sales as a critical test bed for SQL Server

As early adopters of Microsoft beta products ITG provides valuable feedback on issues with the latest product releases. SQL Server 7.0 beta 3 was stable enough to be installed onto the MS Sales production system approximately six months before SQL Server 7.0 was released. The MS Sales team had installed early beta versions of SQL Server and then performed real world functional and performance testing. Bug reports and performance test results were given to the SQL Server development team, so that issues found internally would be resolved before the product was released. As new versions of products become available, the MS Sales team is committed to continually evaluating those products thus providing the real world feed back needed to produce enterprise ready products.

Closing

Larger market shares, larger customer bases and a rapidly evolving global economy are challenges common among most large businesses today. For all companies increasing the value of information makes lowering operating costs, and increasing profitability easier to do.

Microsoft's IT organization has developed many solutions using SQL Server 7.0 that have helped to lower operating costs, increase profitability while helping the company to become more nimble for change. ITG is sharing its experiences so that where applicable others may benefit from similar solutions in their own environments.

APPENDIX

ITG DBA Transact-SQL Monitoring Tool

Through the following transact-SQL procedure ITG devised a way to catalog SQL Server–based blocking information that is characterized by a high level of user contention. By storing information about the lead blocker into a table, ITG can later troubleshoot the cause of contention problems and tune the application to increase performance.

The procedure begins by identifying all the processes being blocked from **sysprocesses.** Then it inserts the process information into the table **sysproc**.

All the blocking locks (v65) are inserted into the table **blk_lock_info**. Information about the spid itself is also captured, such as the input buffer, wait type, spid number, number of blocked users, and so on. This information is inserted into the table **blk_inputbuffer**.

The script creates several tables as well as the procedure. (It is more efficient to make use of a permanent table than to continually queue off **sysprocesses**.) This is necessary because the procedure needs to use the tables. By running the script it will create everything needed to create the procedure and all its dependent objects.

In order to routinely check for blocking, this procedure can be called at regular intervals from a SQL exec task (v65) or a SQL agent job (v70). ITG database administrators run this tool every minute so they do not miss any significant blocking events. The data is later used to improve the performance of the application by eliminating the SQL blocking.

```
if exists (select * from sysobjects where id =
object_id('dbo.blk_inputbuffer') and sysstat & 0xf = 3)
drop table dbo.blk_inputbuffer
GO
if exists (select * from sysobjects where id = object_id('dbo.blk_lock_info')
and sysstat & 0xf = 3)
drop table dbo.blk_lock_info
if exists (select * from sysobjects where id = object_id('dbo.lock_output')
and sysstat & 0xf = 3)
drop table dbo.lock_output
GO
if exists (select * from sysobjects where id = object_id('dbo.sysproc') and
sysstat & 0xf = 3)
drop table dbo.sysproc
GO
CREATE TABLE dbo.blk_inputbuffer (
date datetime NULL ,
suid smallint NULL ,
sysprocesses_cnt int NULL ,
active_count int NULL ,
spid smallint NULL ,
Total_Blocked int NULL ,
RunTime_Sec int NULL ,
waittype binary (2) NULL ,
hostname varchar (15) NULL ,
program_name varchar (30) NULL ,
cmd varchar (16) NULL ,
InputBuff varchar (255) NULL
)
GO
CREATE TABLE dbo.blk_lock_info (
date datetime NULL ,
spid smallint NULL ,
table_id int NULL ,
page int NULL ,
```

```
locktype varchar (255) NULL
)
GO
CREATE TABLE dbo.lock_output (
spid smallint NULL ,
locktype varchar (255) NULL ,
table_id int NULL ,
page int NULL ,
dbname varchar (36) NULL
)
GO
CREATE TABLE dbo.sysproc (
spid smallint NOT NULL ,
kpid smallint NOT NULL ,
status char (10) NOT NULL ,
suid smallint NOT NULL ,
hostname char (30) NOT NULL ,
program_name char (30) NOT NULL ,
hostprocess char (8) NOT NULL ,
cmd char (16) NOT NULL ,
cpu int NOT NULL ,
physical_io int NOT NULL ,
memusage int NOT NULL ,
blocked smallint NOT NULL ,
waittype binary (2) NOT NULL ,
dbid smallint NOT NULL ,
uid smallint NOT NULL ,
login_time datetime NOT NULL ,
last_batch datetime NOT NULL ,
nt_domain char (30) NOT NULL ,
nt_username char (30) NOT NULL ,
net_address char (12) NOT NULL ,
net_library char (12) NOT NULL
)
GO
CREATE PROCEDURE block_analyze
WITH RECOMPILE
AS
SET NOCOUNT ON
DECLARE @cmd VARCHAR(255),
@spid SMALLINT,
@suid SMALLINT,
@sysprocesses_cnt INT,
@active_count INT,
@info VARCHAR(255),
@total_blocked INT,
@last_batch DATETIME,
@date DATETIME,
@command varchar(16),
@hostname varchar(15),
@program_name varchar(30),
@waittype binary(2)
SELECT @date = GETDATE()
IF (SELECT COUNT(*) FROM sysproc) > 0
TRUNCATE TABLE sysproc
ELSE
INSERT sysproc
SELECT spid ,kpid ,status ,suid ,hostname , program_name ,
hostprocess ,cmd , cpu , physical_io , memusage ,blocked ,
waittype ,dbid ,uid ,login_time ,last_batch ,
nt_domain ,nt_username ,net_address ,
net_library
FROM master..sysprocesses WHERE blocked > 0
OR (blocked = 0 and spid in (select blocked from master..sysprocesses))
IF (SELECT COUNT(blocked) FROM sysproc WHERE blocked > 0) > 0
BEGIN
SELECT @sysprocesses_cnt = COUNT(*) FROM sysproc
SELECT @active_count = COUNT(*) FROM master..sysprocesses
where spid > 9
```

```
and cmd != 'AWAITING COMMAND'
/* Identify lead blocker(s) */
DECLARE workman CURSOR FOR
SELECT spid FROM sysproc
WHERE spid IN (SELECT blocked FROM sysproc)
AND blocked = 0
OPEN workman
FETCH NEXT FROM workman INTO @spid
WHILE (@@fetch_status = 0)
BEGIN
SELECT @total_blocked = COUNT(*) FROM sysproc WHERE blocked > 0
SELECT @cmd = "sp_lock " + CONVERT(VARCHAR(5),@spid)
INSERT INTO lock_output EXEC(@cmd)
INSERT INTO blk_lock_info
SELECT @date,spid,table_id,page,locktype FROM lock_output
WHERE locktype LIKE "%-blk%"
TRUNCATE TABLE lock_output
SELECT @cmd = "DBCC INPUTBUFFER(" + CONVERT(VARCHAR(5),@spid) + ")"
INSERT INTO blk_inputbuffer (Inputbuff) EXEC(@cmd)
SELECT @suid = suid,
@command = cmd,
@hostname = hostname,
@program_name = program_name,
@waittype = waittype,
@last_batch = last_batch
FROM sysproc WHERE spid = @spid
UPDATE blk_inputbuffer
SET date = @date,
suid = @suid,
spid = @spid,
total_blocked = @total_blocked,
RunTime_Sec = DATEDIFF(ss, @last_batch, GETDATE()),
waittype = @waittype,
cmd = @command,
hostname = @hostname,
program_name = @program_name,
sysprocesses_cnt = @sysprocesses_cnt,
active_count = @active_count
WHERE date IS NULL
FETCH NEXT FROM workman INTO @spid
END
CLOSE workman
DEALLOCATE workman
TRUNCATE TABLE blk_info
TRUNCATE TABLE sysproc
END
ELSE
BEGIN
PRINT "No Blocking currently." RETURN
END
PRINT "Blocking Found"
RETURN
GO
```

Implementing OLAP Services in the MARS Application

Here are the steps taken by ITG to migrate some of the report-generation component of the MARS application from MS Reports to OLAP Services:

Execute proof of concept.

1. Research available literature to determine OLAP Service capabilities.
2. Build a simple cube with MARS data.
3. Build an intranet-based user interface using Office 2000 components.

Work with business partners to examine ad hoc queries.

1. Create a more robust cube.
2. Run ad hoc queries in order to benchmark performance.
3. Compare benchmark results against MS Reports.

Create a standard profit-and-loss (P&L) report.

1. Have business users create P&Ls.
2. Pilot-test OLAP Services worldwide and provide bench marks
3. Convene teams regularly to tackle issues (for example, technology versus policy) and to determine workarounds and tradeoffs suitable for large internal usage.

Since implementing OLAP Services in MARS, ITG has benchmark ad hoc queries running much faster than comparable queries using transact-SQL via MS Reports. The benchmark is significant considering that ITG had previously coded MS Reports for efficiency in the MARS application. For example, MS Reports performs aggregate navigation and merging of heterogeneous subcubes horizontal partitioning, as well as other query techniques commonly used to traverse a star schema. The fact that out-of-the-box technology can perform similar operations much faster is a big win for designers, planners, and users of the data warehouses that ITG builds.

Table 4 illustrates the results of the ITG benchmarks:

MARS Running Under:	SQL Server 7.0 and MS Reports	SQL Server 7.0 with OLAP Services
Average query	3.5 minutes	2 seconds
Long query	30 minutes	5 seconds
Worst query	Did not finish	11 seconds
Management of P&L	10–30 megabytes	200–500 kilobytes
Open Management of P&L over WAN	45 minutes	45 seconds

Table 4: Ad hoc queries faster under OLAP Services

It is worth noting that part of the MARS upgrade from MS Reports to OLAP Services involved some of the Web components of Office 2000. With these components (which were made available to MARS users by way of the Microsoft MARS intranet), client computers can access the OLAP Services cube directly. Figure 4 illustrates a query result obtained using OLAP Services then displayed on a Web page using the Web components of Office 2000.

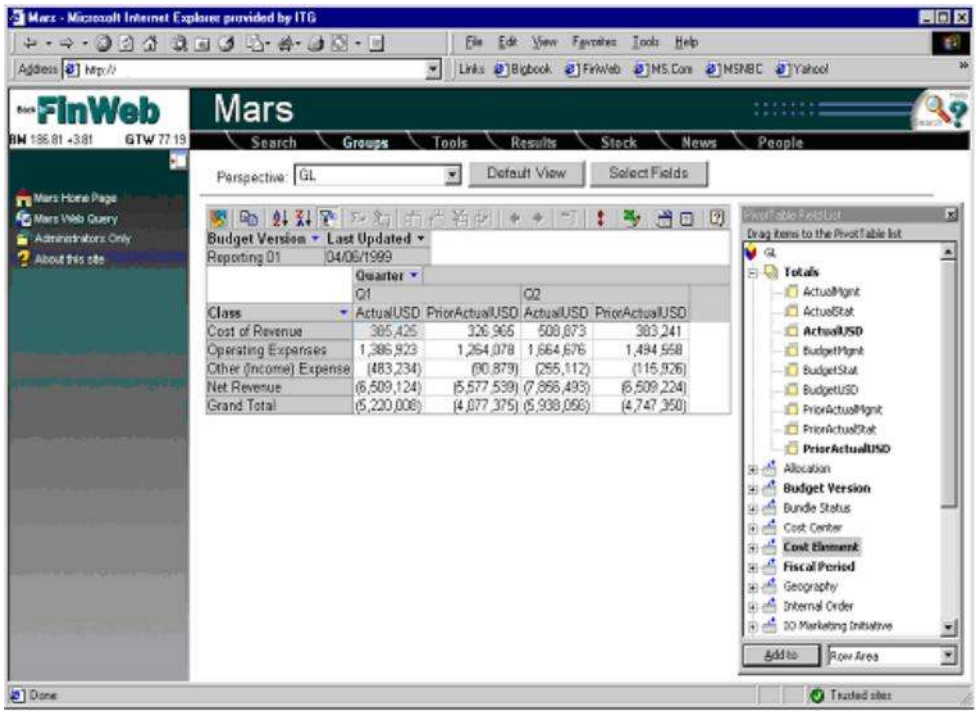

Figure 4 Query Results from OLAP Services

ITG analysts and planners began pilot-testing Office 2000 Web components with OLAP Services and in one day had a working Web site that supported ad hoc queries. This means that in the future ITG can avoid having to build more SQL Server–based (aggregated) tables by letting OLAP Services build the aggregations automatically. In the case of the MARS upgrade, ITG configured OLAP Services to point to the MARS data marts, and OLAP Services built multidimensional cubes for them. The MARS data marts are required to remain in place, but custom aggregations built by designers and planners may likely no longer be needed. In the future, ITG will use OLAP Services in a multitude of applications so as to present information to decision-makers faster and in a more consistent format.

FOR MORE INFORMATION

You can find the latest information on Microsoft SQL Server at:

http://www.microsoft.com/sql/

To view additional IT Showcase material, please visit http://www.microsoft.com/technet/itshowcase

For any questions, comments or suggestions on this document, or to obtain additional information about Microsoft IT Showcase, please send e-mail to showcase@microsoft.com