DSS Sizing and Tuning of Oracle8 for Windows NT on Compaq Servers

White Paper

Prepared By

Database Engineering

Compaq Computer Corporation

March 1998

COMPAQ

NOTICE

The information in this publication is subject to change without notice.

COMPAQ COMPUTER CORPORATION SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN, NOR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL.

This publication does not constitute an endorsement of the product or products that were tested. The configuration or configurations tested or described may or may not be the only available solution. This test is not a determination of product quality or correctness, nor does it ensure compliance with any federal, state or local requirements. Compaq does not warrant products other than its own strictly as stated in Compaq product warranties.

DSS Sizing and Tuning of Oracle8 for Windows NT on Compaq Servers

First Edition (March 1998)

Document Number **ECG156/0398**

Compaq Computer Corporation

Table of Contents

Introduction

The purpose of this document is to share the knowledge acquired by Compaq Systems Engineers in the area of configuration and performance tuning of Decision Support Systems using the Oracle8 Database and Microsoft Windows NT on Compaq servers. The system tested by Compaq represents a single query stream TPC Benchmark D (TPC-D) on the Compaq ProLiant family of Servers. General information and query data for the TPC-D benchmark are included in this paper. It is our desire to deliver the best technical information possible on a specific topic in a timely manner and in a highly useable format. Any comments, suggestions and feedback are always appreciated.

TPC Benchmark D (TPC-D)

The TPC Benchmark D is a decision support benchmark, which consists of business oriented ad-hoc queries and concurrent updates. The benchmark illustrates decision support systems that examine large volumes of data; execute queries with a high degree of complexity; and gives answers to critical business questions. The TPC-D benchmark evaluates performance of various decision support systems by the execution of sets of seventeen queries against a standard database. Appendix A of this document contains the schema of the TPC-D database, appendix B contains descriptions of each of the seventeen queries, and appendix C contains the SQL code for each of the seventeen queries used in the TPC-D benchmark examples. A full description of the TPC-D specification is available from the Transaction Processing Performance Council (Phone: 408-295-8894, Web site: www.tpc.org/dspec.html).

DSS vs. OLTP

Decision Support Systems (DSS) is a term used to describe the capability of a system to support the formulation of business decisions through complex queries against a database. It can also specifically refer to a database which is intended for this purpose, as opposed to one which primarily supports on-line transaction processing (OLTP) operations. Decision Support is different from OLTP. OLTP applications are update-intensive and generally consist of shorter transactions that access a small portion of a database, often through a primary key or index. Decision support applications typically consist of long and often complex read-only queries that access large portions of the database. Decision support databases are updated relatively infrequently, either by periodic batch runs, or by background "trickle" update streams. The database need not contain real-time or up-to-the-minute information, as decision support applications tend to process large amounts of data which usually would not be affected significantly by individual transactions.

Oracle Overview

The information in this document contains information related to Oracle8 Server version 8.0.4 for Windows NT version 4.0 or later. Because there is information available concerning generic tuning of Oracle8 Server, this document focuses on specific tuning suggestions for DSS on Compaq servers and Windows NT. A section on general Oracle8 architecture is included as a part of the Oracle Overview. Tuning sections follow the Oracle Overview. Wherever possible, this White Paper references other useful tuning documentation.

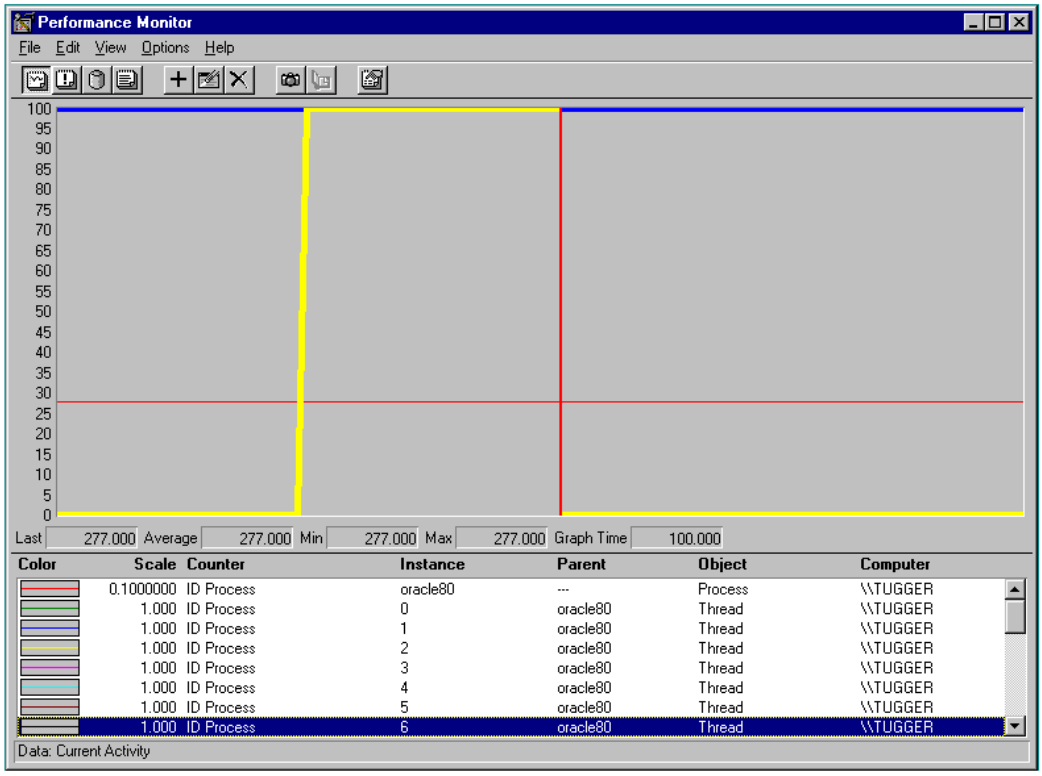
Architecture

Oracle8 Server for Windows NT is a 32-bit Windows NT application. Oracle8 Server is implemented on Windows NT as a single process, multi-threaded architecture. Each Oracle8 Server instance consists of a single process with multiple threads. The number of threads associated with the Oracle8 Server process varies depending upon options selected (background threads) and user connections (shadow threads). Note that Compaq disk subsystems provide asynchronous I/O, therefore no DBWR and LGWR I/O slave threads are required. The maximum number of shadow threads, and thus the maximum number of user connections, is

1024. The Windows NT Performance Monitor displays the process (ORACLE80) and threads (Threads 0,1 represent the Oracle service threads). If ARCHIVELOG mode is enabled, threads 2,3,4,5,6,7,8

represent the background threads [PMON, DBWR, ARCH, LGWR, CKPT, SMON and RECO respectively]. If ARCHIVELOG mode is not enabled, threads 2,3,4,5,6,7 represent the background threads [PMON, DBWR, LGWR, CKPT, SMON and RECO respectively] as shown in Figure 1. The process ID for each thread is listed in the "Last", "Average", "Min" and "Max" boxes in Performance Monitor. Descriptions of each background thread can be found in Figure 3.

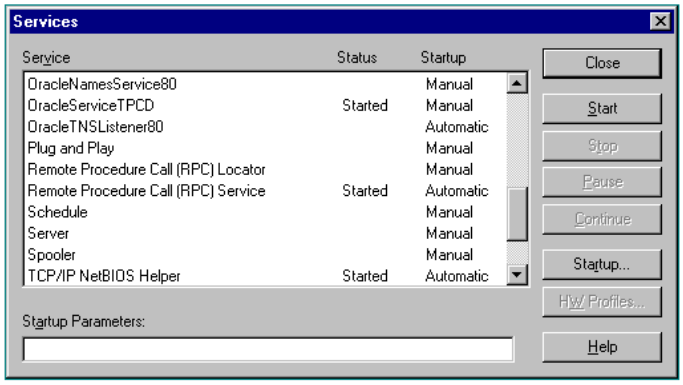
Figure 1: Performance Monitor – Oracle8 Server process and threads



The Oracle8 Server process is represented under Windows NT as a service (OracleServiceSSSS [SSSS= System ID]) associated with the executable ORACLE80.EXE. The Oracle Service consists of two threads, 0 and 1. Thread 0 is the Oracle process main thread acting as a service dispatcher and creating thread 1 to handle the service.

Each Oracle instance is associated with a specific Windows NT service and consists of a single process and multiple threads. Use the Windows NT Control Panel/Services to display the Oracle Services. OracleServiceTPCD represents the Oracle Instance TPCD as shown in Figure 2.

Figure 2: Control Panel/Services – Oracle8 Server instances



NOTE: Performance Monitor does separate the instances on the chart, log, and report. However the instances are all identified as "oracle80", making it very difficult to monitor multiple instances using Performance Monitor.

An Oracle instance consists of the System Global Area (SGA) and the Oracle service/ background/ shadow threads. Via an Oracle instance, an Oracle database can be created and accessed.

The SGA is a sharable memory construct that contains the following:

- Database Buffers Contains most recently used database blocks
- Shared Pool Contains shared SQL areas and data dictionary cache
- Redo Log Buffers Logs changes made to the database

Figure 3:

The Oracle8 Server background threads consist of the following:

Thread	Abbreviation	Description
Process Monitor	PMON	Responsible for the cleanup of abnormally terminated connections.
Database Writer	DBWR	Writes database blocks to datafiles.
Log Writer	LGWR	Writes the redo log entries to logfile.
Checkpoint Process	CKPT	Signals DBWR to perform updates on all data and control files of the database. If not present, LGWR does this.
System Monitor	SMON	Performs instance recovery and cleanup.
Recovery Process	RECO	Resolves failures with the distributed option.
Archival Process	ARCH	Copies full online redo log files to the archive device (if in ARCHIVELOG mode).

A dedicated shadow thread is a separate dedicated server thread, which acts on behalf of a particular user. One thread is created for each user who connects to the database. Any request that a user has for the database is performed through the shadow thread for that user. Oracle8 Server version 8.0.4 supports a multithreaded server (MTS) environment. An MTS server thread can service requests from any client.

When a shadow thread must read from the database, it checks to see if the data exists in the SGA. If the data exists in the SGA, the shadow thread reads it from the memory. If the data is not found in memory, the shadow thread goes directly to the datafiles and reads the data into the

SGA. When a shadow thread must write to the database, it writes into the SGA only. At a later time the DBWR writes this "dirty" data out to disk.

Parallel execution begins with a shadow thread executing a SQL statement that contains operations, some of which may be performed in parallel. This becomes the shadow parallel coordinator. It dispatches the execution of a statement to several parallel server threads and coordinates the results from the parallel server threads to send the results back to the user. The number of parallel server threads assigned to a single operation is called the "degree of parallelism".

The redo log contains a history of all committed transactions for the database to perform an instance recovery. A minimum of two redo logs is required and more may be used. When a redo log fills, a log switch occurs. At log switch time all new redo information goes to the next redo log file in line. If the system is running in ARCHIVELOG mode, which is recommended, the previous log file is copied out to an archive log file.

With a recent backup, the redo log files, and the redo log archive files, the database can be recovered if needed. The control and configuration files are used to store information of the state and layout of the database as well as system tunables. A more in-depth discussion of the Oracle8 Server architecture can be found in the *Oracle8 Server Concepts Manual.*

New Features in Oracle8 Version 8.0

Oracle8 version 8.0 is a major upgrade from Oracle7 version 7.3 and contains many new features. The features covered in this paper are briefly outlined below. For more details on Oracle8 new features, refer to the Appendix A of the *Oracle8 Server Migration Manual*.

New Feature	Description
Partitioned Tables and Indexes	Divides large tables and indexes into smaller and more manageable pieces (partitions). Oracle8 includes features to create, manage, backup and recover, and utilize partitions for better query and DML performance.
ROWID Enhancement	Oracle8 ROWIDs have been modified to handle partitioned tables and tablespace-relative data block addresses (DBAs). The new 10-byte ROWID contains the data object number, data block address and row number of the row.
Data Dictionary	Data Dictionary has more views providing extended information on partitions, parallel server and latches/locks. Also some views have additional columns defined to include new features and functionality.
Reverse-key indexes	Reverses the bytes of each column indexed, except the ROWID, but keeps the column order. Enables insertions to become distributed across all leaf keys in the index.
SQL*Loader Partitioned Object Support	Supports loading partitioned objects into the database. The direct path has been changed to accommodate mapping rows to partitions of tables and to support local and global indexes. Parallel direct-path includes concurrent loading of an individual partition as well as support for concurrent loading of a partitioned table.

Basic Tuning

This section contains basic tuning information as it pertains to installation and configuration of the hardware, operating system and Oracle8 server. The Tuning sections of this paper which follow contain information for tuning your DSS system.

Installation Issues

Flashable ROM

Apply the latest Compaq System ROMPaq and Option ROMPaq to the ProLiant Server.

Compaq System Configuration

During configuration of the ProLiant Server using the latest Compaq System Configuration utility, several settings are recommended.

System

Primary Operating System = Windows NT version 4.0 or later

Specifies the Primary Operating System for initial configuration options.

Compaq Memory

Base Memory = 640 Kbytes, Linear

Contiguously maps all memory so that Compaq Built-In Memory is not available at FA0000, but is instead added into extended memory. The previously reserved memory area at FE0000 is now available for use.

Array Configuration Utility

Controller Characteristics- Operating System = Windows NT 3.5x or later

This sets the drive geometry for the controller to be 32 sectors per track and the striping factor to be 256 for RAID-0 and RAID-1. The drive geometry for the controller is 32 sectors per track and the striping factor is 32 for RAID-5.

Log Drives: Array Accelerator Status - Logical Drive # = Disabled

Non-Log Drives: Array Accelerator Status - Logical Drive # = Enabled

This increases performance of the controller by writing data to RAM on the controller instead of directly to disk. The RAM on the controller is mirrored and backed up by batteries for increased reliability. The memory for the accelerator is either 4MB or 16MB, depending upon the version of the controller board. The accelerator cache can be configured at 100/0 percent, 75/25 percent, 50/50 percent, 25/75 percent or 0/100 percent for the amounts of memory that will be reserved for the read-ahead/write-posting functionality of the accelerator.

Windows NT

Prior to or immediately following installation of Oracle8 Server for Windows NT, there are several recommended changes to the Windows NT configuration.

Compaq Software Support for Microsoft Windows NT 4.0

Install the latest version of the Compaq Software Support Diskette to apply the latest drivers specific for Windows NT on Compaq hardware. The available drivers are listed below.

- 10/100 UTP Netflex3
- SMART Controller CPQARRAY
- 32-bit SCSI-2/Integrated PCI SCSI-2 adapters
- ProLiant Storage System
- SCSI Compression adapter
- System Management/Health driver
- UPS support
- Multiprocessor HAL support

The ECI driver for the SMART Controller CPQARRAY can be obtained from the Compaq web site at /www.compaq.com/support/files/server/winnt/index.html.

Microsoft Windows NT 4.0 Workstation and Server Service Pack

Install the latest version of the Service Pack to apply the latest fixes from Microsoft.

Memory

The Windows NT Enterprise Edition increases the per-process address limit from 2GB to 3 GB. This feature benefits applications that run on Compaq ProLiant systems with more than 2 GB of physical RAM and that can take advantage of a larger address space. If you have greater than 2GB RAM, you will need to install the Windows NT Enterprise edition and modify the boot.ini file by adding the /3GB parameter to the startup line. For example,

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(2)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(2)\WINNT="Windows NT Server Version 4.00" /3GB
multi(0)disk(0)rdisk(0)partition(2)\WINNT="Windows NT Server Version 4.00 [VGA
mode]" /basevideo /sos/basevideo /sos
```

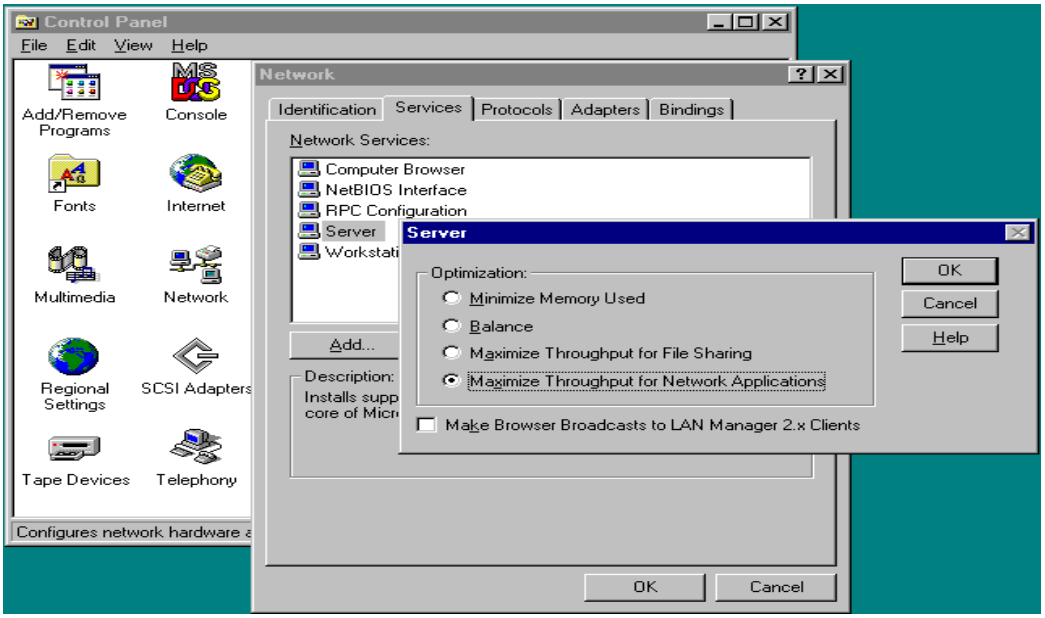
Virtual memory is your real memory (RAM) plus your swap file size. The virtual memory should be approximately one to one and a half times that of your physical memory (RAM). The virtual memory should not exceed twice your physical memory as the paging rate may go up during peak times, resulting in decreased performance. Virtual memory is set in the Windows NT Control Panel – System – Performance – Virtual Memory. Note: if you using larger amounts of RAM, it may not be practical to set your virtual memory to one to one and a half times that of your physical memory. When setting your virtual memory and swap file size, keep in mind that you want to reduce your amount of paging and swapping.

Server Configuration

Use Control Panel/Network to choose *Server* from the Installed Network Software list and Configure. Change the relationship of memory allocated to the network connections and memory allocated to applications running on the server by choosing *Maximize Throughput for Network Applications* (the default is *Maximize Throughput for File Sharing*). This optimizes the server

memory for the Oracle8 Server because it performs its own memory management for caching file and network I/O. See Figure 4.

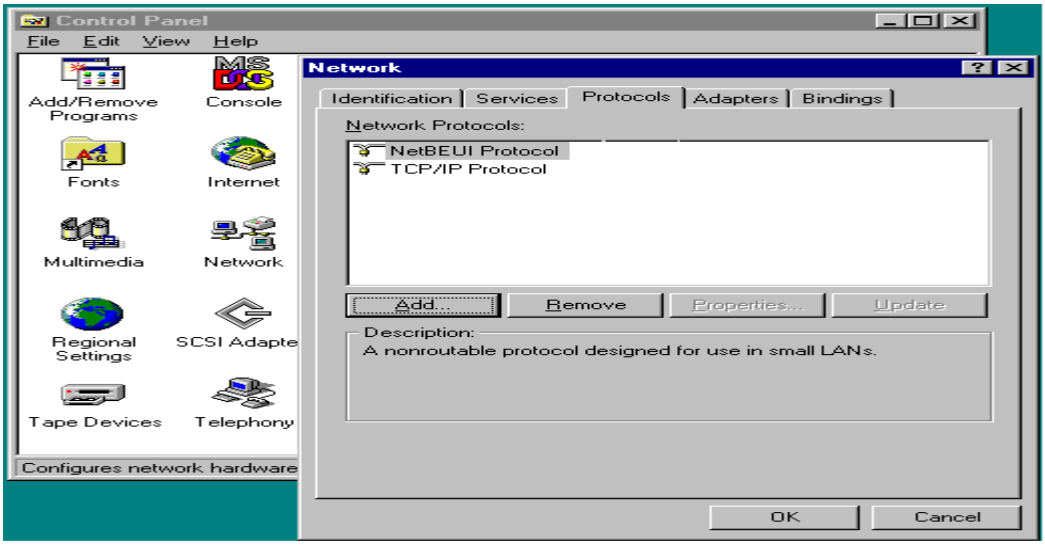
Figure 4: Control Panel/Network/Services/Server - memory configuration



Network Protocols

Use Control Panel/Network to choose any protocol not required for the activities on the server from the Installed Network Software List and Remove. See Figure 5.

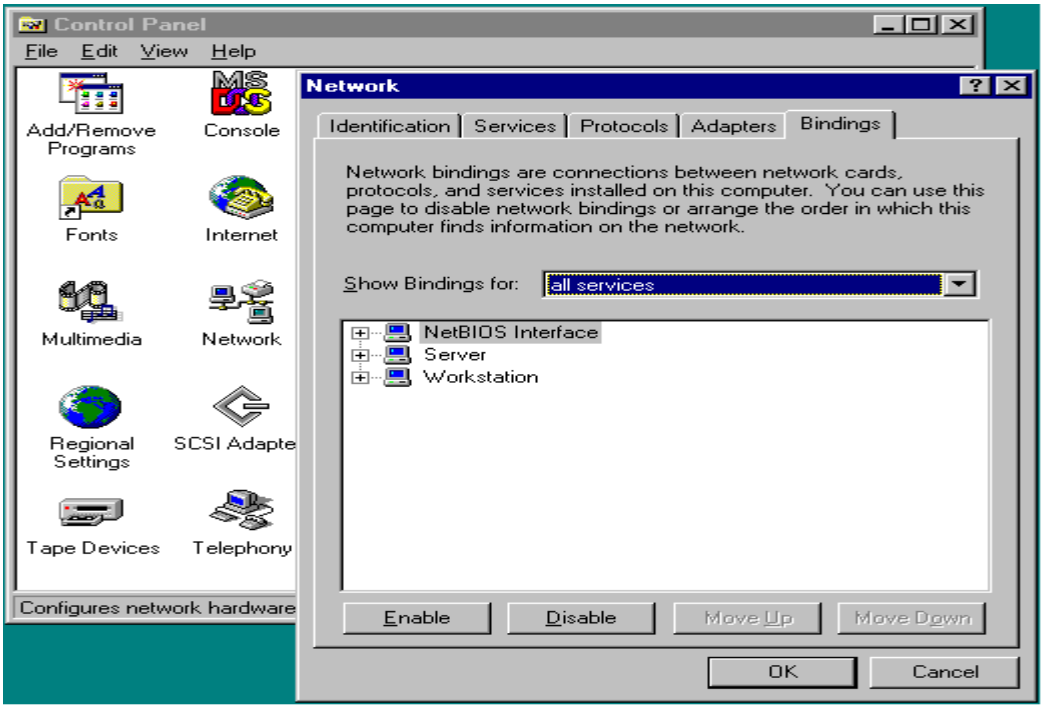
Figure 5: Control Panel/Network - network protocols



Server and Workstation Network Bindings

Use Control Panel/Network to choose *Bindings* to configure the Server and Workstation network bindings path. Move the protocol of primary use to the top followed by the protocol of second highest usage and so forth on both the Server and Workstation to reduce the average connection time. See Figure 6.

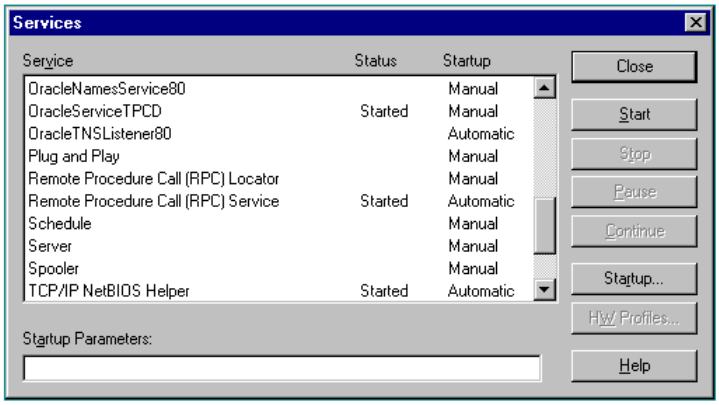
Figure 6: **Control Panel/Network/Bindings - protocol binding order**



Services

If the Oracle8 Server is dedicated to being a database server only, various services may be stopped to reduce overhead on the processor(s). The minimum service needed is OracleServiceSSSS [SSSS is the System ID], but Compaq also recommends the Server, EventLog and Workstation services. One or more services, depending upon the versions of Net8 and protocols supported, will be required to support user connections through Net8 (example, OracleTNSListener80). Turn off services using the Control Panel/Services (you may also want to change the service's Startup option to Manual). See Figure 7.

Figure 7: Control Panel/Services



Windows NT File System

Compaq testing indicated a 4%-7% performance gain when using raw devices over FAT or NTFS file systems in Windows NT. Compaq recommends using FAT for the initial boot partition to allow booting from a DOS diskette in emergencies to do repair work. Windows NT and Oracle executables are placed on the FAT partition as well. The system, log and data files are normally placed on NTFS or RAW file partitions. If you are I/O bound with your system, you may switch the log and data files to RAW partitions to gain I/O performance. You will, however, be limited by the restrictions of using RAW devices (i.e. backup, copy, etc.). Using Oracle8's backup utility, OCOPY80, and MKS Toolkit's dd functionality, you can minimize the impact of these restrictions. Each RAW partition equates to a single file and can be represented by a single drive letter (i.e., \\.\E:), or logical physical drive number (i.e., \\.\PhysicalDrive0) or, if you have many partitions per disk, as symbolic links (i.e., \\.\ORA_PARTITION_1 where ORA_PARTITION_1 is defined using Oracle's setlinks utility).

Oracle8 Server

Install the Oracle partition option. This option is needed for Oracle parallel query and partition functionality.

Use ORADIM80 to create each Oracle8 instance. ORADIM80 (Oracle Database Instance Manager) creates each database instance and assigns the administrator user ID and password. If you are recreating an Oracle8 instance, you will want to delete the old instance using:

```
oradim80 -delete -sid SSSS
```

before creating the new instance using:

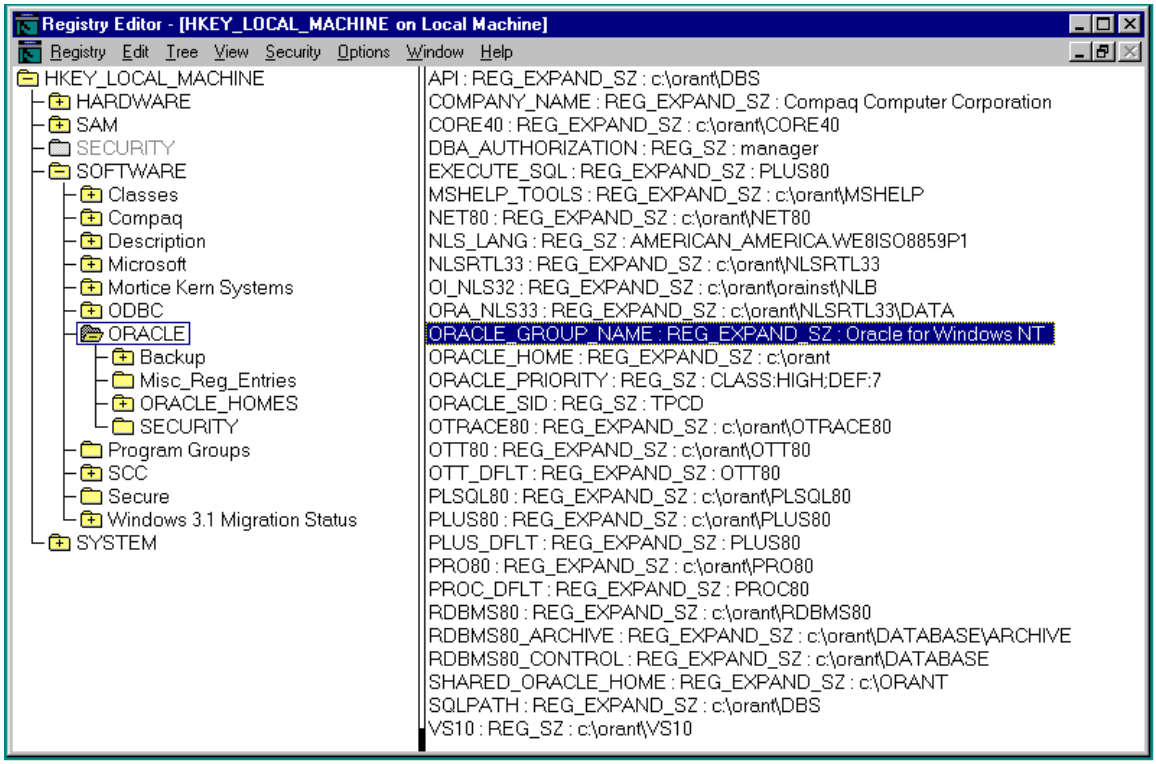
```
oradim80 -new -sid SSSS  -intpwd PWD -maxusers U -startmode manual
```

where SSSS is your instance ID, PWD is your internal password and U is the maximum number of internal users.

Use the Windows NT Registry Editor to make any necessary changes to the Oracle configuration parameters (for example, ORACLE_SID, LOCAL, ORACLE_HOME, and so on). The default setting for ORACLE_PRIORITY is CLASS:normal;DEF:normal. This can be set to CLASS:HIGH;DEF:7. The console will seem sluggish but can be fixed by increasing the foreground window priority under Control Panel – System - Performance.

See Figure 8 for Windows NT Registry settings for ORACLE.

Figure 8: Registry Editor



Tuning Goals

There are several factors involved in achieving the best performance from your system. This paper focuses on the hardware, Oracle8 Server, and the operating system. It is important to tune the application software to take advantage of the system resources. Due to the diversity of applications, tuning application software is beyond the scope of this paper. For more information on the tuning of applications, refer to the *Oracle8 Server Application Developer's Guide.*

When tuning a DSS system, the primary goal is to shorten the time it takes to run each query as much as possible. Since each query may differ significantly, it is likely that tuning the system to shorten one query may lengthen another. It is, then, important to determine which queries need to be reduced for acceptable response times across the board. Your most frequently run queries should be tuned first. Once these are tuned, or if you are unsure which of the queries should be tuned, you should tune your longest and your shortest running queries. Tuning your longest running queries can significantly reduce your query response time, while tuning your short running queries will free up high contention resources and ease the contention in many cases as well. This paper is the end result of tuning and evaluating three main areas of a DSS system: I/O, CPU and memory.

I/O Tuning

Due to the sequential nature of DSS systems, tuning I/O on DSS is very different from that of OLTP. Adding disk spindles will often reduce I/O contention in OLTP systems, due to their random I/O access. In a DSS system, you will need to tune I/O beyond the simple addition of disk spindles. This section will discuss I/O tuning for a DSS system in regards to disk layout, partitioning, parallelism, availability, performance and I/O limits.

Disk Layout

Disk Layout and partitioning are two of the most important areas of I/O tuning and DSS performance. It is helpful to know the partitioning requirements of your system prior to determining your disk layout. Within the area of disk layout, the following recommendations have been tested on Compaq ProLiant systems and offer optimal DSS performance.

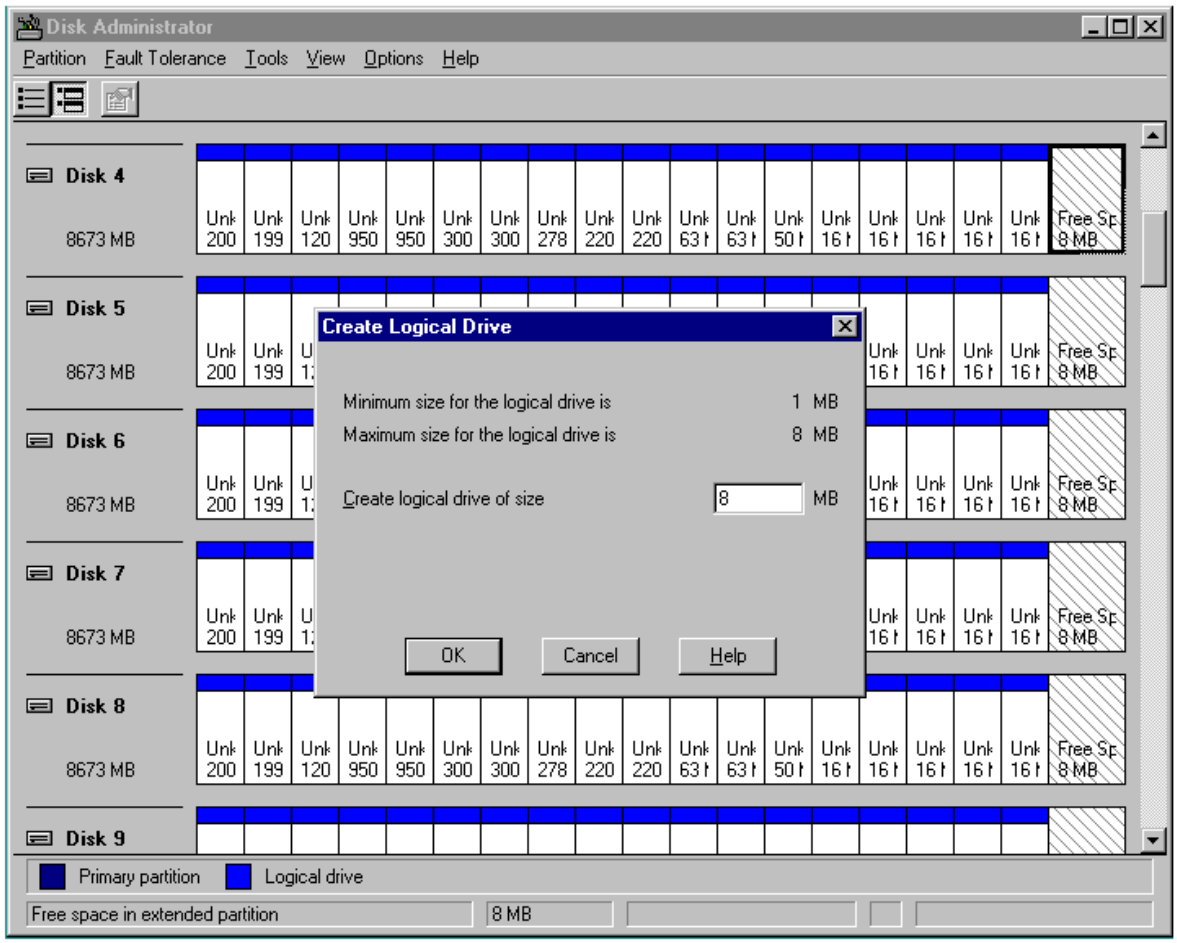
- Since DSS is extremely read intensive, you will want to balance your largest tables (i.e. DSS fact tables) across as many drives as possible. For example, in the TPC-D, we balanced the part, partsupp, lineitem, orders, supplier and customer tables across 40 drives, putting $1/40^{th}$ of each table and its index on one drive. Smaller objects, including the system tablespace, rollback segments, and nation and region tables, are combined on a single drive. Each partition can be assigned to its own tablespace, which allows the user to assign partitions to physical drives. In the TPC-D example, each $1/40^{th}$ of the tables is on its own partition, which is tied to a specific drive (i.e. partsupp is on tablespaces PS_01, PS_02…PS_40). With partitioning, this will reduce your read time by splitting it evenly among your drives.
- There should be one logical drive per physical disk, which will allow for one partition per physical disk for each database object. (i.e. If you have a table that has 40 partitions, you will want to set up 40 logical drives, as separate disks or group of disks, to contain that table) This will reduce the disk contention when reading from or writing to disk and increase the amount of sequential access, which can be much faster than random I/O. Depending upon the query, indexes may still provide better performance (i.e. index-only retrieval).

Once the disk layout has been determined, the Compaq Array Configuration Utility and Windows NT Disk administrator are used to configure the disks and create the partitions. In the Compaq Array Configuration Utility, one array needs to be created for each logical disk. (i.e. If

you have 14 disks attached to one controller, you will create 14 arrays of one disk each for RAID-0 or 7 arrays of two disks each for RAID-1.) The RAID level is specified when creating the logical disk. The logical disk should take up all the space available on each array. When all logical drives have been created, you will need to

split the logical drives into partitions. This is done through the Windows NT Disk Administrator. Figure 9 displays the Windows NT Disk Administrator for logical disks 4-8 in the TPC-D example. In this example, there are forty logical disks (one per physical disk), each having eighteen partitions. Each partition is created by selecting the disk and Partition – Create Logical.

Figure 9: Disk Administrator



Partitioning

The ability to partition tables and indexes in Oracle 8 allows for a more granular table/index layout, better management of table and disk storage and increased DSS and I/O performance. In a DSS system, performance can be improved by reducing full table and index scans to partition scans.

As stated above, partitioning and disk layout go hand in hand when developing and tuning a DSS system. When setting up partitions in Oracle8, you will need to consider the following.

- Table Partitioning – Oracle8 allows range partitioning, which uses a range of column values (partitioning key) to map rows or index entries to partitions. The partitioning key consists of an ordered list of up to sixteen columns and is based on the partitioning column values. You will need to determine which data values you wish to partition. The most common range partitioning is by date. A table cannot be partitioned if it is a cluster; contains LOBs, LONG or LONG RAW datatypes or objects; or is an index-organized table.

- Index Partitioning – Is similar to table partitioning. There are two types of index partitioning: Local and Global. In a local partitioned index, all keys in a particular index partition refer only to rows stored in a single underlying table partition (i.e. All keys in a particular index partition map to all rows in a single underlying table partition). In a global partitioned index, the keys in a particular index partition may refer to rows stored in more than one underlying table partition (i.e. All indexes in one partition may map to rows in three different partitions.) An index cannot be partitioned if it is a cluster index, defined on a clustered table, or a bitmap index on a partitioned table. Oracle8 supports prefixed and non-prefixed local indexes and prefixed global indexes. A prefixed index is partitioned on a left prefix of the index columns, while a non-prefixed index is partitioned on something other than a left prefix of the index columns. For DSS systems, local non-prefixed indexes can improve performance because many index partitions can be scanned in parallel by range queries on the index key. While these indexes are easier to manager, there is a potential of having to search every partitioned index to find a single element.
- Partitions – Each partition should be assigned to its own tablespace, which allows for a single partition to be assigned to a single physical disk. This feature enables the developer to place data on specific drives and, ultimately, balance the I/O across all disks. Since DSS databases are read-intensive, partitioning a table will allow for a faster read time. The number of partitions you have will, in part, determine the number of disks you will need. For example: in figure 9 above, each of the partitions created is assigned to its own tablespace (i.e. Partition 1 on all disks is reserved for partsupp. Partition 2 on all disks is reserved for orders. Thus partition 1 on disks 1-40 are assigned to the PS_01, PS_02, ... , PS_40 tablespaces. Partition 2 on disks 1-40 are assigned to ORD_01, ORD_02, ... , ORD_04 tablespaces)
- Storage parameters – All partitions of a table or index have the same logical attributes, but their physical attributes can be different. Assigned to its own tablespace, each partition can utilize different physical attributes (PCTFREE, PCTUSED, INITRANS, MAXTRANS) and storage parameters (INITIAL, NEXT, MINEXTENTS, MAXEXTENTS, FREELIST, FREELIST GROUPS). PCTFREE should be as close to zero as possible if no/little changes are done to data (i.e. updates). Inserts are okay with small PCTFREE value.
- Physical Device Selection – The frequency of partition used also impacts the selection of drives. More frequently used partitions may be moved to a faster device (i.e. moving from a Compaq 4.3-Gigabyte Ultra to a Compaq 9.1-Gigabyte drive), allowing for a decrease in the response time.

For more information on partitioning, please refer to the *Oracle8 Server Concepts Manual* or the *Oracle8 Server Application Developer's Guide.*

Parallelism

Parallelism offers performance improvement when full table scans, large joins, partitioning and sorting and other aggregate functions are used. Parallelism depends primarily on the number of processes, the number of CPU's, memory and the I/O layout.

Parallel Execution

When Oracle is not parallelizing the execution of SQL statements, each SQL statement is done sequentially by a single process. With parallel execution, multiple processes work together simultaneously to execute a single SQL statement. These processes will be referred to as parallel query processes or parallel server processes, but in the Windows NT architecture they are really threads of the Oracle process. The Oracle8 server can use parallel executions for operations such as table scans, joins, group by, order by, union, select distinct, aggregation, PL/SQL functions, create table as select, create/rebuild index, move/split partition, update, delete, insert…select, enable constraint (the table scan is parallelized) and star transformations. Oracle8 parallelizes SQL statements in the following ways.

- **Parallelize by block ranges for scan operations** – Parallel scans by block range break the table or index into pieces delimited by high and low ROWID values. The table or index can be non-partitioned or partitioned. For partitioned objects, the ROWID range cannot span a partition. Oracle sends the partition numbers with the ROWID ranges to avoid a partition map lookup. Predicates on the partitioning columns can be used to restrict the ROWID ranges to relevant partitions only (known as partition pruning).
- **Parallelize by partitions for operations on partitioned tables and indexes** – Partitions are already a logical division of the tables and indexes. These can be used to break up operations into smaller operations executed in parallel. This is done by assigning a different parallel server process to different partitions. Each partition will be accessed by a single parallel server process, but a parallel server process may need to access multiple partitions.
- **Parallelize by parallel server processes for inserts into nonpartitioned tables only** – Oracle8 can parallelize the work of insert operations by dividing the work among the parallel server processes. Since new rows do not have ROWIDs, the rows are distributed among the server processes to insert them into the free space.

Degree of Parallelism

When Oracle8 utilizes parallel execution, two or more of the instance's parallel server processes may be used to process a SQL statement. The number of parallel server processes associated with a single operation is known as the degree of parallelism. The degree of parallelism is specified at the statement level, using hints or the PARALLEL clause; at the table or index level, in the table or index definition; or by default based on the number of disks or CPUs. The following are factors involved in determining the degree of parallelism.

- The query uses the maximum degree of parallelism taken from all of the tables involved in the query and all of the potential indexes that are candidates to satisfy the query. The table or index that has the highest degree of parallelism determines the query's degree of parallelism. Keep in mind, when setting the degree of parallelism, too high of a degree of parallelism on one table may exhaust some other resource of your system like memory or CPU bandwidth for example.
- If a table has both a PARALLEL hint specification in the query and a parallel declaration in its table definition, the hint specification takes precedence over the parallel declaration of the table.
- If the maximum number of parallel server processes, determined by the PARALLEL_MAX_SERVERS, is already performing parallel tasks, the query must continue the remainder of its operations sequentially. PARALLEL_MAX_SERVERS is the total number of sequential streams accessing data plus the number of parallel engines for sorting, average, sum, count and other aggregate functions. PARALLEL_MAX_SERVERS should be set to two times the maximum degree of parallelism for each table times the number of users.

The degree of parallelism may vary between objects. The degree of parallelism for each table should be a factor of the number of partitions in the table. Optimizing TPC-D benchmarks on Compaq ProLiant machines has shown that the average degree of parallelism per object should be 2-3 times the number of CPUs. For small objects or objects that are not accessed with an index or partitioned scans, the degree of parallelism can be set to one.

Memory

Memory available for parallel server processes comes from the process memory, which in turn comes from virtual memory. You will need to make sure that you have enough memory for all of your processes. The PROCESSES parameter specifies the maximum number of operating system user processes that connect to the Oracle instance and should be large enough to contain the user processes, background processes and PARALLEL_MAX_SERVERS. Your SGA should be much less than your physical memory to maximize PGA memory. You will also want to make sure that your HASH_AREA_SIZE * PARALLEL_MAX_SERVERS is much less than your available real memory (physical memory minus SGA).

Availability vs. Performance

Once disk space for your DSS system has been calculated, it important to determine the level of protection for your drives. Database availability, performance and cost are considerations in determining the Compaq SMART Array Controller RAID level that is used. This RAID level will directly impact the physical disk space required for your DSS system. Compaq SMART Array Controller RAID levels 0, 1 and 5 are discussed from a performance aspect in this paper. Please refer to the Compaq Database Engineering White Paper *Configuring Compaq RAID Technology for Database Servers* for more information regarding the Compaq SMART Array Controller RAID selection and implementation

RAID-0 offers no fault tolerance for your drives. It has a 1:1 write ratio (1 physical write:1logical write) and a 1:1 read ratio (1 physical read:1 logical read). All RAID-0 data is stored in one physical location. It is the lowest cost option, as your total drives is equal to the number of logical drives required.

RAID-1 offers fault tolerance via data mirroring. It has a 2:1 write ratio (2 physical writes:1logical write) and a 1:1 read ratio. Due to mirroring, all RAID-1 data is stored in two physical locations, which provides the benefit of split reads. Split reads offers a performance boost, especially when there are not many writes, because the database may read from either copy of the data. RAID-1 is the highest cost, as twice the storage is required, but also the most fault tolerant.

RAID-5 offers fault tolerance via distributed data guarding. It has a 4:1 write ratio (2 physical reads + 2 physical writes: 1 logical write) and a 1:1 read ratio. All RAID-5 data is stored in one physical location, so there is no benefit from split reads. The amount of storage required is one drive (parity) for every x drives (data). Under RAID-5, there is a substantial performance penalty during writes of 20-35%, depending upon your application.

Compaq SMART Array Controllers have the flexibility to use either one or a combination of the above RAID levels. Depending upon your database availability, performance and cost requirements, it may be beneficial to use RAID-1 on critical drives and RAID-0 or RAID-5 on

other drives. Log drives should be placed on a RAID-1 drive, so that no log files are lost in the case of a single disk failure.

Array Controllers

The Compaq SMART Array Controller is preferable to standard SCSI controllers. The following are benefits achieved when using the Compaq SMART Array Controllers. This information is specific to the Compaq SMART-2/P and SMART-2DH controllers.

- Allows for up to 14 disks to be configured in the system as one or more logical drives. When multiple disks are collected into a single array, any logical drives created on that array will have their data striped across all of the disks. This will more evenly distribute the data across the disk spindles so that your I/O load will be more balanced on them. This works really well

  for random I/O loads or for sequential loads where you do not have more sequential streams accessing the data than you have defined logical drives on which the data coexists. For example, if you spread your table across four logical drives, to maintain sequential streams, you should keep your degree of parallelism on that table to four or less. Each logical drive will then be sequentially scanned, which results in a sequential scan of the physical disks.

- Multiple logical drives can be configured on each disk or array of disks. The maximum number of logical drives per single controller is 32. In a DSS system, you will want only one logical drive per disk.

- The accelerator cache can be configured as either read-ahead cache or write-posting cache or varying degrees of both. The memory for the accelerator is either 4MB or 16MB, depending upon the version of the controller board. The memory can be configured at 100/0 percent, 75/25 percent, 50/50 percent, 25/75 percent or 0/100 percent for the amounts of memory that will be reserved for the read-ahead/write-posting functionality of the accelerator. The configuration you choose will primarily depend upon the amount of writes to your database.

- Controller provided RAID levels – Defining RAID levels at the controller versus software RAID on Windows NT relieves system processor overhead for maintaining RAID.

I/O Limits

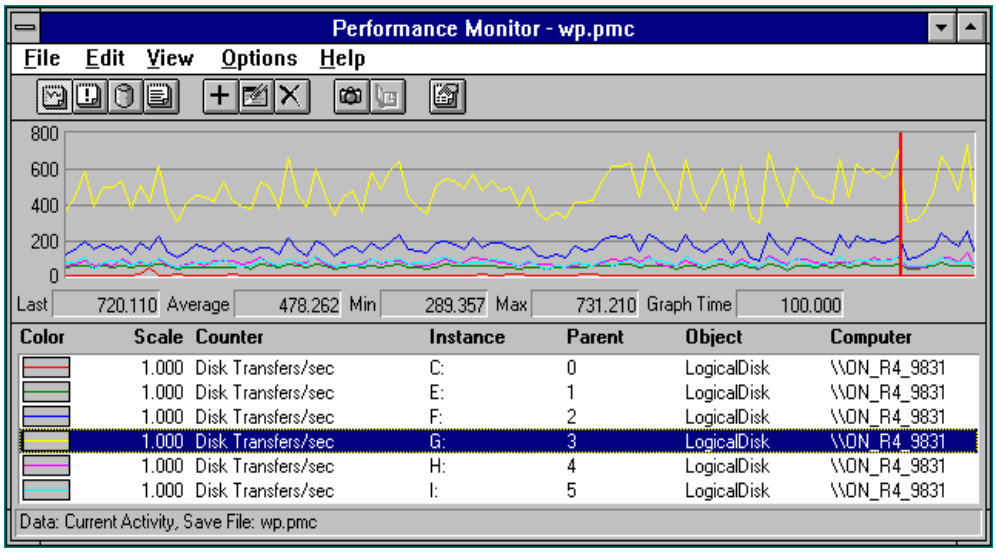
In a DSS system, it is optimal to tune for most data access to be sequential for better data throughput. When multiple processes are accessing a disk concurrently, it is important not to overload each individual disk with random I/O's. The following are recommended I/O limits for Compaq 4.3-Gigabyte Ultra and 1-inch and 10K RPM 9.1-Gigabyte drives. For 16K block size, Compaq recommends that random I/O's not exceed 60 I/O's per second per drive for the 4.3-Gigabyte Ultra drives, and 75 I/O's per second per drive for 9.1-Gigabyte drives.

While the system is under load, determine the number of I/O's per second to each logical volume using the Performance Monitor. See Figure 10. Then, using the formulas in the Compaq Database Engineering White Paper *Configuring Compaq RAID Technology for Database Servers*, you can calculate the number of I/O's per physical disk based on the fault tolerance level of that particular logical volume.

In the example displayed in Figure 10, there is no fault tolerance on logical NTFS volume G. The average I/O rate is 478.262 I/Os per second. If the drive set consists of seven 4.3-Gigabyte Ultra drives, then the I/O rate per second per drive is 478.262 / 7 = 68.32 I/O's per second per drive. The test purposely flooded the logical drive with I/O's, giving a scenario above the recommended 60 I/O's per second per disk and requiring the possibility of adding more disk drives or rearranging files. If the drive set consists of seven 9.1-Gigabyte drives, then the 68.32

I/O's per second per drive would be within the 75 I/O's per second per drive limit. Hence, you would not have to add more disk drives or rearrange files if you were using 9.1-Gigabyte drives.

Figure 10: Performance Monitor - Disk I/O



System Processor Scalability

Oracle8 Server for Windows NT provides good system processor scalability. By adding system processors, performance can be improved, provided that your system is processor-bound. While no operating system or hardware provides 100-percent scalability, Windows NT and Oracle8 provide an above-average level of scalability. When well-tuned Oracle instances are system processor bound, most applications see significant improvement from the addition of more powerful processors and more processors. Additional cache may provide some improvement, depending upon the number of processors used. If the system is I/O bound, adding system processors will not offer significant improvement. In DSS systems, it is important to determine the threshold for acceptable processor performance. This threshold will depend upon the number of users and the processor time that each will require.

The following charts represent system processor scalability tested on the Compaq ProLiant 3000, 5500 and 6500 servers using the exact queries defined in the TPC-D benchmark. Across all servers, the best improvement came with additional and faster processors. Adding cache provided minimal improvement, depending upon the number of processors used. Figure 11 displays results from tests on the Compaq ProLiant 5500 and 6500 servers (both Pentium Pro processors), using 3G RAM and 512K cache. They display processor scalability for 1, 2 and 4 processors.

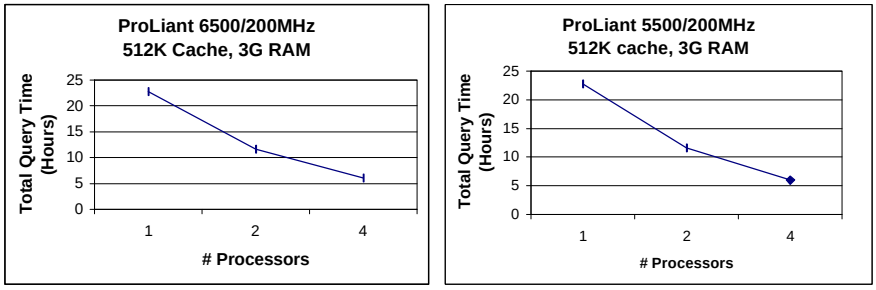


Figure 11: System Processor Scalability – # of Processors

Figure 12 displays the comparison between query time on the Compaq ProLiant 5500 (Pentium Pro processor) and ProLiant 3000 (Pentium II processor), using 512M RAM and 512K cache. The Pentium Pro processor clock speed is 200MHz, and the Pentium II processor clock speed is 300MHz. The maximum RAM for the Pentium II processors is 512M. Each set of queries was run with 2 processors.

Figure 12: Pentium Pro (200MHz) vs. Pentium II (300MHz) – TPCD Query Time (seconds)

Query #	Pentium Pro	Pentium II
1	15977.13	9817.73
4	2419.27	1748.42
15	417.69	275.9
10	2598.83	2023.72
11	454.73	381.98
6	287.21	203.26
2	540.37	456.47
16	1450.92	1101.16
14	579.38	428.99
8	3495.63	2552.86
12	898.03	670.11
17	320.23	272.1
3	4468.03	3472.14
5	6378.34	5143.45
13	87.46	82.43
7	7249.83	5837.44
9	18376.33	18672.73
# Seconds (total)	65999.41	53140.89
# Hours (total)	18.33	14.76

Memory Tuning

System Global Area (SGA) Size

Because the Oracle8 Server SGA resides in server memory, this is allocated at the startup of the instance. The SGA is made up of the following components:

Fixed Size + Variable Size + DB Block Buffers + Redo Buffers = Size of SGA

Fixed Size is determined by the Oracle products that you have installed. It is approximately 43K bytes.

Variable Size is determined by *init.ora* parameters such as SHARED_POOL_SIZE and DB_BLOCK_LRU_LATCHES.

DB Block Buffer is determined by the **DB_BLOCK_BUFFERS*DB_BLOCK_SIZE** parameters.

NOTE: If you are using asynchronous I/O and Oracle can lock down memory for asynchronous I/O, the DB Block Buffers will not be swappable.

Redo Buffer size is determined by the **LOG_BUFFERS** parameter plus approximately 80K bytes of overhead space. **LOG_BUFFERS** is specified in bytes.

Program Global Area (PGA) Size

The PGA is allocated by Oracle when a user thread connects and a session is created or for each Parallel Query thread. The PGA is a region of memory that contains data and control information for a single connection. This memory must be available at the connect time for a particular user, therefore the amount of free server memory is a limitation to the number of concurrent connections. The PGA's size is affected by the following parameters:

- **OPEN_LINKS**
- **DB_FILES**
- **LOG_FILES**

- **HASH_AREA_SIZE**
- **SORT_AREA_SIZE**

For a large data warehouse, HASH_AREA_SIZE may range from 8MB to 32MB or more for hash joins.

Processes

The init.ora parameter **PROCESSES** should be adjusted. This parameter specifies the number of operating system user processes that connect to the Oracle instance. This number must also include the Oracle service, background, and parallel server threads initiated at the Oracle instance startup. There are two Oracle service threads. The background threads include the seven processes defined in Figure 3. Therefore, the number must be at least the maximum concurrent connections (concurrent connections cannot exceed 1024) plus two service threads and the number of background threads. The V$PROCESS and V$BGPROCESS contain information about the Oracle threads being used.

Windows NT

In addition to memory required for Oracle8 and its processes, memory must also be available for Windows NT. Memory usage can be monitored via the Windows NT Task Manager – Performance window. Two primary goals in tuning memory for Windows NT are to reduce the amount of paging and swapping and to fit the SGA and PGAs into main memory.

Paging or swapping is the process in which the operating system moves information from one storage location (i.e. real memory, virtual memory, disk) to another. Excessive paging or swapping can reduce performance. To monitor paging, use the Windows NT Performance Monitor – Memory – Pages Input/sec. Pages Input/sec is the number of pages read in from disk to resolve memory references to pages that weren't in memory at the time of reference. This number should be extremely low. If excessive paging occurs, either increase the total memory on your system (RAM) or decrease the amount of memory you have allocated in the SGA.

Since the purpose of the SGA is to store data in memory for fast access, the SGA should always be stored in main memory. You can cause Oracle to read the entire SGA into memory when you start your instance by setting the value of PRE_PAGE_SGA to YES. This may increase the amount of instance startup time, but it is likely to decrease the amount of time necessary for Oracle to reach its full performance capacity after startup. Setting PRE_PAGE_SGA to YES does not prevent Windows NT from paging or swapping the SGA after it is initially read into memory. DSS systems, as tested in the TPCD, do not typically have a high cache hit ratio. Note that memory can be better utilized in places such as hash area and sort area than in the SGA.

Total Memory

The total amount of memory (RAM) for a DSS system is dependent upon your SGA, PGA, number of processes and Windows NT. Thus, the impact of additional RAM is dependent upon your system and database design, as well as your query types. Figure 13 displays the different Total Query Times for a TPC-D on a Compaq ProLiant 6500, using 1M cache and 3-Gigabyte, 2-Gigabyte, 1-Gigabyte and 512-Megabyte RAM. Figure 14 displays underlying data for queries significantly impacted by the addition of RAM.

NOTE: There is a constraint in Oracle8 memory usage which prohibits the Oracle process from using all 3GB of virtual address space.

Figure 13: Compaq ProLiant 6500, 1M cache (3G, 2G, 1G, 512M RAM) – TPCD Query Times

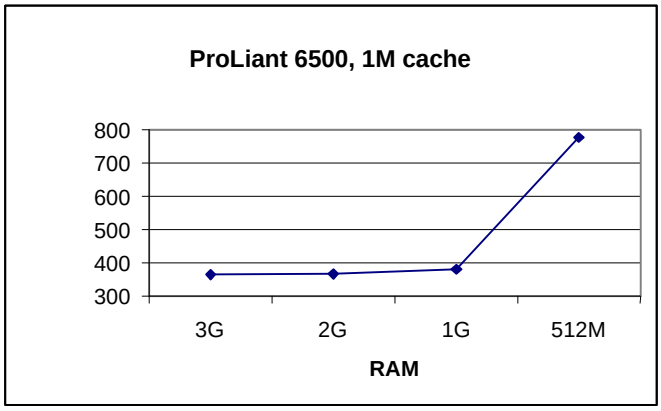


Figure 14: Compaq ProLiant 6500, 1M cache (3G, 2G, 1G, 512M RAM) – Selected TPCD query data. Query time in seconds.

Query #	3G	2G	1G	512M
5	2152.47	2158.85	2291.47	4599.31
7	1196.1	1217.14	1272.26	5244.27
9	3492.87	3456.51	4024.89	17112.15
11a	171.31	169.17	168.71	403.74
13	3.59	3.67	16.63	91.74
15b	225.93	225.51	225.09	224.94
17	16.2	16.42	16.46	225.4

In figure 11, assuming that you are cost bound, configuring the Compaq ProLiant 6500 with 1G RAM seems to be optimal. However, if you look at figure 12, you can see that the addition of RAM does not help all queries. Thus, it is also important to look at query definitions and functionality in determining the amount of RAM you need. The following gives explanation of memory usage for TPC-D queries identified in figure 12. The TPC-D schema and SQL code for these queries are included in Appendix A (schema) and Appendix C (SQL code).

Both queries 5 and 13 have a small return set (less than or equal to seven rows), but require a join of the largest tables in the database. Utilizing 512M and 1G RAM, queries 5 and 13 do not have enough memory to perform optimal joins. Looking at figure 12, you can see that 2G RAM provides enough memory for both joins.

Query 7's performance improves with the addition of RAM up to 3G, so 3G RAM is a viable configuration. This indicates that the additional memory helps in the 6-table join and the storage

of 2/7 of the Lineitem rows and 2/5 of the Customers and Suppliers rows. In this query, our generalization based on the configuration of 1G RAM would not be optimal.

Query 9 is optimized using 2G RAM. Like Query 7, Query 9 does a 6-table join, but has selection criteria applied to only one small table. 512M and 1G RAM do not offer enough memory for optimally joining the six tables. Note that additional RAM (i.e. 3G in query 9, or 2G or 3G in query 11) may hurt performance, which is most likely due to overhead of loading memory which may not be used efficiently.

Query 11 contains a three-table join of moderate and small size tables. However, this query returns millions of rows. Hence, 512M is too small for the sort size of this query. Looking at the data, 1G RAM should be sufficient.

Query 15 has a one-second change in query time from 512M to 3G. It does a simple aggregate function, which can be processed within 512M RAM. Thus, the addition of RAM beyond 512M does not help.

In Query 17, correlated subquery functionality is used to perform what-if analysis. It joins the largest table, Lineitem, to Part in both inner and outer queries. 1/1000 of the rows in the Part table qualifies based on the selection criteria. 512M RAM is not enough memory to store the qualifying rows. An additional 512M RAM provides enough memory to store all the qualifying rows, prior to doing the AVERAGE aggregate function.

Other INIT.ORA Parameters

This white paper has covered I/O, CPU and Memory tuning. Where applicable, INIT.ORA parameters (in all CAPITAL letters) have been discussed. This section contains additional INIT.ORA parameters, or a further explanation of previously discussed parameters, that should be set for a DSS system. When

tuning a DSS system, it is important to realize that optimization of parameters for one application do not necessarily transfer to other applications. There is some degree of tweeking that must be done to optimize your DSS system.

- DB_BLOCK_SIZE – Must be defined prior to database creation, as a change in DB_BLOCK_SIZE requires the database to be rebuilt. Should be set to the largest value possible. In a DSS system on Oracle 8.0.4 for Windows NT, DB_BLOCK_SIZE should be set to 8K or 16K.
- DB_BLOCK_BUFFERS – Will be lower in a DSS environment than in an OLTP environment. (DB_BLOCK_BUFFERS*DB_BLOCK_SIZE) Dimension tables and amount of updates will dictate the size of this. Full table scans, full index scans and full partition scans will bypass the main buffer cache.
- DB_FILE_MULTIBLOCK_READ_COUNT – Determines how many database blocks are read with a single operating system read. The maximum I/O size of the Compaq SMART-2 controller is 64K bytes. Thus, the DB_FILE_MULTIBLOCK_READ_COUNT should be set to 64K/DB_BLOCK_SIZE or a multiple thereof. For a DB_BLOCK_SIZE of 16K, the recommended DB_FILE_MULTIBLOCK_READ_COUNT is 4.
- SHARED_POOL_SIZE – Should be large enough to hold all data dictionary, shared SQL and compiled objects.

- HASH_AREA_SIZE – Memory area allocated for each process to do hash join work. The hash area does not cache blocks in the buffer cache. For DSS systems, this value may range from 8MB to 32MB. Larger tables require a larger HASH_AREA_SIZE. Setting the HASH_AREA_SIZE is system dependent and should be tuned over a period of time until an acceptable hash area size is identified.
- HASH_MULTIBLOCK_IO_COUNT – Determines the number of hash buckets for I/O transfer. For larger tables (greater than 100gig in size), this parameter should be increased. During our TPC-D test, HASH_MULTIBLOCK_IO_COUNT is set to 16.
- HASH_JOIN_ENABLED – Should be set to TRUE to allow for hash joins.
- SORT_AREA_SIZE – Specifies the maximum amount of PGA memory, which any Oracle thread can use for sorting.
- SORT_WRITE_BUFFERS – If SORT_DIRECT_WRITES is set to true, each user thread will have SORT_WRITE_BUFFERS * SORT_WRITE_BUFFER_SIZE * number of parallel query processes of memory allocated to it. These buffers are then used for sorting instead of the main buffer cache.
- ORDERED_NESTED_LOOP – Helps cost based optimizer to determine order of execution for each loop within a nested loop when ORDERED_NESTED_LOOP is set to TRUE.
- OPTIMIZER_MODE – Use CHOOSE or COST. Do not use RULE. The default value is CHOOSE.
- OPTIMIZER_PERCENT_PARALLEL – Specifies the amount of parallelism that the optimizer uses in its cost functions. The default of 0 means that the optimizer chooses the best serial plan. A value of 100 means that the optimizer uses each object's degree of parallelism in computing the cost of a full table scan operation. Low values favor indexes, and high values favor table scans. The recommended value is 100 divided by your number of concurrent users.
- PARALLEL_MIN_SERVERS – Specifies the minimum of parallel query processes for each instance. Oracle8 Server creates these processes at the instance startup.
- PARALLEL_MAX_SERVERS – Specifies the maximum of parallel query processes for each instance. As the number of SQL statements processed increases, Oracle8 Server will automatically create a new parallel process as needed. It will not exceed the value specified in PARALLEL_MAX_SERVERS.
- PARALLEL_SERVER_IDLE_TIME – The time which parallel servers will remain active. If parallel process has been idle for the period of time specified by PARALLEL_SERVER_ IDLE_TIME, then Oracle8 Server will terminate that process. The number of parallel processes will never drop below the value of PARALLEL_MIN_SERVERS.

Please refer to *Oracle8 Server Tuning* for more information regarding Oracle parallel execution parameters.

Conclusion

Tuning a DSS system is a highly system dependent and reiterative job. One configuration may work well for one DSS system and poorly for another DSS system. When initially setting up and tuning your system, you may have to go through numerous trials before getting an acceptable level of performance. You will also want to evaluate the DSS performance after your system is set up and determine if any of the above tuning suggestions need to be reapplied. Don't worry – your users will usually inform you if this is necessary. Good luck!

We would welcome feedback from your configurations and experiences to improve our information products in the future. Please send us any comments or suggestions on the attached form, attach addition sheets if necessary. This will help us tailor the future information products to your needs, and will enable us to make future revisions of this document and related new information products available to you.

Appendix A: TPC-D Schema

Note: SF = Scale Factor of TPC-D database (i.e. For a 100G database, SF=100)

PART Table Layout

Column Name	Datatype Requirements	Comment
P_PARTKEY	identifier	SF*200,000 are populated
P_NAME	variable text, size 55	
P_MFGR	fixed text, size 25	
P_BRAND	fixed text, size 10	
P_TYPE	variable text, size 25	
P_SIZE	integer	
P_CONTAINER	fixed text, size 10	
P_RETAILPRICE	decimal	
P_COMMENT	variable text, size 23	

Primary Key: P_PARTKEY

SUPPLIER Table Layout

Column Name	Datatype Requirements	Comment
S_SUPPKEY	identifier	SF*10,000 are populated
S_NAME	fixed text, size 25	
S_ADDRESS	variable text, size 40	
S_NATIONKEY	identifier	Foreign key reference to N_NATIONKEY
S_PHONE	fixed text, size 15	
S_ACCTBAL	decimal	
S_COMMENT	variable text, size 101	

Primary Key: S_SUPPKEY

PARTSUPP Table Layout

Column Name	Datatype Requirements	Comment
PS_PARTKEY	identifier	Foreign key reference to P_PARTKEY
PS_SUPPKEY	identifier	Foreign key reference to S_SUPPKEY
PS_AVAILQTY	integer	
PS_SUPPLYCOST	decimal	
PS_COMMENT	variable text, size 199	

Compound Primary Key: PS_PARTKEY, PS_SUPPKEY

CUSTOMER Table Layout

Column Name	Datatype Requirements	Comment
C_CUSTKEY	identifier	SF*150,000 are populated
C_NAME	variable text, size 25	
C_ADDRESS	variable text, size 40	
C_NATIONKEY	identifier	Foreign key reference to N_NATIONKEY
C_PHONE	fixed text, size 15	
C_ACCTBAL	decimal	
C_MKTSEGMENT	fixed text, size 10	
C_COMMENT	variable text, size 117	

Primary Key: C_CUSTKEY

ORDER Table Layout

Column Name	Datatype Requirements	Comment
O_ORDERKEY	identifier	SF*1,500,000 are sparsely populated
O_CUSTKEY	identifier	Foreign key reference to C_CUSTKEY
O_ORDERSTATUS	fixed text, size 1	
O_TOTALPRICE	decimal	
O_ORDERDATE	date	
O_ORDERPRIORITY	fixed text, size 15	
O_CLERK	fixed text, size 15	
O_SHIPPRIORITY	integer	
O_COMMENT	variable text, size 79	

Primary Key: O_ORDERKEY

LINEITEM Table Layout

Column Name	Datatype Requirements	Comment
L_ORDERKEY	identifier	Foreign key reference to O_ORDERKEY
L_PARTKEY	identifier	Foreign key reference to PS_PARTKEY
L_SUPPKEY	identifier	Foreign key reference to PS_SUPPKEY
L_LINENUMBER	decimal	
L_QUANTITY	decimal	
L_EXTENDEDPRICE	decimal	
L_DISCOUNT	decimal	
L_TAX	fixed text, size 1	
L_RETURNFLAG	fixed text, size 1	
L_LINESTATUS	fixed text, size 1	
L_SHIPDATE	date	
L_COMMITDATE	date	
L_RECEIPTDATE	date	
L_SHIPINSTRUCT	fixed text, size 25	
L_SHIPMODE	fixed text, size 10	
L_COMMENT	variable text, size 44	

Compound Primary Key: L_ORDERKEY, L_LINENUMBER

NATION Table Layout

Column Name	Datatype Requirements	Comment
N_NATIONKEY	identifier	25 nations are populated
N_NAME	fixed text, size 25	
N_REGIONKEY	identifier	Foreign key reference to R_REGIONKEY
N_COMMENT	variable text, size 152	

Compound Primary Key: L_ORDERKEY, L_LINENUMBER

REGION Table Layout

Column Name	Datatype Requirements	Comment
R_REGIONKEY	identifier	5 regions are populated
R_NAME	fixed text, size 25	
R_COMMENT	variable text, size 152	

Primary Key: R_REGIONKEY

TIME Table Layout

Column Name	Datatype Requirements	Comment
T_TIMEKEY	date	Number of days since 1-1-1970
T_ALPHA	fixed text, size 10	Date expressed as YYYY-MM-DD
T_YEAR	integer	Year expressed as YYYY
T_MONTH	integer	Number of months since 1-1-1970
T_WEEK	integer	Week of the year from 1 to 53
T_DAY	integer	Day of the month

Primary Key: T_TIMEKEY

Appendix B: TPC-D Query Definitions

Query 1 - Pricing Summary Report

TPC-D Business Question: Provides a summary pricing report for all Lineitems shipped as of a given date, a date always within 180 days of the greatest shipdate contained in the database.

Functionality: Query 1 performs multiple aggregations and summaries by reading and processing over 95% of the rows of the database's largest table. Only this single table is scanned with a very low number of rows being returned.

Query 2 - Minimum Cost Supplier

TPC-D Business Question: Find, in a given Region, for each Part of a certain type and size, the Supplier who can supply it at minimum cost. If several suppliers in that region offer the desired part type and size at the same (minimum) cost, the query lists the Parts from Suppliers with the 100 highest account balances.

Functionality: Query 2 is a correlated subquery based on a 5-table join in both outer query and inner query. Close to 5% of the Supplier rows result from the selection criteria and query processing, but only the 100 Suppliers with the highest account balances are returned.

Query 3 - Shipping Priority

TPC-D Business Question: Retrieves the shipping priority and potential revenue of Orders having the largest revenue among those that had not been shipped as of a given date. Orders are listed in decreasing order of revenue. If more than 10 unshipped Orders exist, only the 10 Orders with the largest revenue are listed.

Functionality: Query 3 performs a 3-table join on three of the larger tables in the database. Anywhere from 1/5 to _ of the rows of each of the 3 tables participate in the joins, with a final aggregation that produces a very high number of rows, in the millions for most volume points. Only the 10 Orders with the highest revenue are returned.

Query 4 - Order Priority Checking

TPC-D Business Question: Counts the number of Orders ordered in a given quarter of a given year in which at least one Lineitem was received by the Customer later than its committed date. The query lists the count of such Orders for each order priority sorted in ascending priority order.

Functionality: Query 4 is a correlated subquery in which about 1/28 of the Lineitem rows are selected for evaluation based on order date. An aggregation that produces a count by priority produces 5 rows in the answer set.

Query 5 - Local Supplier Volume

TPC-D Business Question: Lists for each Nation in a Region the revenue volume that resulted from Lineitem transactions in which the Customer ordering parts and the Supplier filling them were both within that Nation. The query considers only Parts ordered in a given year, displaying the Nations and revenue volume in descending order by revenue.

Functionality: This is a 5-table join of large and small tables, where the data aggregated is reduced down to 1/5 of the Customers and Suppliers (representing one Region out of five) and 1/7 of the Lineitems (one

year out of seven). The largest detail table has no direct selection applied to it. Five rows are returned, constituting the revenue for each nation in the selected region.

Query 6 - Forecasting Revenue Change

Business Question: Considers all the Lineitems shipped in a given year with discounts between DISCOUNT - 0.01 and DISCOUNT + 0.01. The query lists the amount by which the total revenue would have increased if these discounts had been eliminated for Lineitems with L_QUANTITY less than an inputted quantity.

Functionality: This query accesses the large detail table only (Lineitem) selecting about 12% of the rows, and returning a single column answer.

Query 7 - Volume Shipping

TPC-D Business Question: Finds, for two given Nations, the gross discounted revenues derived from Lineitems in which parts were shipped from a Supplier in either Nation to a Customer in the other Nation during 1995 and 1996. Two nations are given as input parameters.

Functionality: Query 7 is a 6-table join that requires a small 25-row table, NATION, to be aliased and processed as though it were two distinct look-up tables. A date constraint selects 2/7 of the Lineitem rows, while a join to the lookup tables result in 2/5 of the Customers and Suppliers being selected. Four rows are returned.

Query 8 - National Market Share

TPC-D Business Question: The market share for a given Nation within a given Region is defined as the fraction of the revenue from the products of a specified type in that Region that was supplied by Suppliers from the given Nation. The query determines this for the years 1995 and 1996.

Functionality: Query 8 is an 8-table join including the NATION table twice. CASE (if/else logic) is used to determine and return a percent value based on two different years. Selection constraints qualify 1/50 of the Part rows, and 1/25 of the Nations.

Query 9 - Product Type Profit Measure

TPC-D Business Question: Finds, for each nation and each year, the profit for all Parts ordered in that year which contain a specified substring in their part-names and which were filled by a Supplier in that nation.

Functionality: Query 9 is a 6-table join with selection criteria applied to only one small table, Part. A text string search is done against the PART table, resulting in 5% of the Part rows being selected. Revenue is aggregated for each combination of Year and Nation, returning 175 rows.

Query 10 - Returned Item Reporting

TPC-D Business Question: Finds the top 20 Customers, in terms of their effect on lost revenue for a given quarter, who have returned Parts. The query considers only Parts that were ordered in the specified quarter. Customers are listed in descending order of lost revenue**.**

Functionality: Query 10 is a 4-table join of three large tables and one look-up table. Customer detail is returned by this query, alongside of only one column of summarized data. 1/28 of the Order rows qualify based on date, while 1 in 4 Lineitems match the criteria of having a return flag of 'R'. Millions of rows are returned from this query, but the final sort determines which 20 rows are to be displayed in the answer set.

Query 11 – Important Stock Identification

TPC-D Business Question: Finds, from scanning the available stock of Suppliers in a given Nation, all the Parts that represent a significant percentage of the total value of all available Parts. The query displays the part number and the value of those Parts in descending order of value.

Functionality: Query 11 is a subquery with both the inner and outer query composed of a 3-table join of the same 3 moderate and small tables. A HAVING clause associates the two sub-queries. 1/25 of these tables' rows are selected, with several million rows being returned in the final answer set.

Query 12 - Shipping Modes and Order Priority

TPC-D Business Question: Counts, by shipping mode, for Lineitems actually received by Customers in a given year, the number of Lineitems belonging to Orders for which the Receiptdate exceeds the Commitdate for 2 different specified shipping modes. Only Lineitems that were actually shipped before the Commitdate are considered. Late Lineitems are partitioned into two groups, those with priority Urgent or High, and those with a priority other than Urgent or High.

Functionality: Query 12 is a two-table join of the two largest tables, with 1/7 of the Lineitems being selected based on date, and all Orders participating in the join. CASE functionality is used to allow a single processing of both tables in order to count the qualifying rows for two different sets of priority criteria. Two rows are returned from the aggregation.

Query 13 - Sales Clerk Performance

TPC-D Business Question: Computes the total loss of revenue on Orders placed by a given Order clerk due to Parts being returned by Customers. The query groups and orders the results by the year in which the Parts were ordered.

Functionality: Query 13 is a short-running query, as each clerk has only 1500 Orders no matter what the volume of data in the database. The join between the two largest tables involves a comparatively small amount of data, as only the Lineitems that have been returned for those 1500 Orders are processed. As there are seven years in the database, seven or fewer rows will be returned, one per year**.**

Query 14 - Promotion Effect

TPC-D Business Question: Determines what percentage of the revenue in a given year and month was derived from promotional Parts. The query considers only Parts actually shipped in that month and gives the percentage.

Functionality: Query 14 calculates the numerator and the denominator of a fraction at the same time by using the CASE syntax to scan and process the data in one single sweep. A two-table join is required, with only 1/84th (one month) of the largest table qualifying for the join based on a date.

Query 15 – Top Supplier

TPC-D Business Question: Finds the Supplier who contributed the most to the overall revenue for Parts shipped during a given quarter of a given year. In case of a tie, the query lists all Suppliers whose contribution was equal to the maximum, presented in Supplier number order.

Functionality: Query 15 requires either a view with aggregates or a temporary table. In the view, 1/28th of the Lineitems (3 months) are selected based on date and aggregated on Supplier. Close to all the Suppliers in the database will participate as output from this first step aggregation. The query itself returns details from the Supplier row which represents the maximum revenue. One row is returned from this query.

Query 16 - Parts/Supplier Relationship

TPC-D Business Question: Counts the number of Suppliers who can supply Parts that satisfy a particular customer's requirements. The Customer is interested in Parts of eight different sizes as long as they are not a given type, not of a given brand, and not from a Supplier who has had complaints registered at the Better Business Bureau. Results must be presented in descending count and ascending brand, type and size.

Functionality: Query 16 uses a subquery with text compares using wild cards to select eligible Suppliers. It uses a two table join for Part and Part Supplier. Inequality and IN/NOT IN functions, with comparison to parameters passed into the query, are used to pare down the answer set.

Query 17 - Small Quantity Order Review

TPC-D Business Question: Considers parts of a given brand and with a given container type and determines the average lineitem quantity of such parts ordered for all orders (past and pending) in the 7-year database. What would be the average yearly gross (undiscounted) loss in revenue if orders for these parts with a quantity of less than 20% of this average were no longer taken?

Functionality: Query 17 uses correlated subquery functionality to perform what-if analysis. A join of Lineitem and Part take place in both inner and outer queries. 1/1000th of the rows in the Part table qualify based on the selection criteria. The item quantity that represents 20% of the average item quantity is calculated in the subquery with the AVERAGE aggregation.

Appendix C: TPC-D Queries (SQL Code)

Query 1 – Original

```
SELECT
 L_RETURNFLAG,
 L_LINESTATUS,
 SUM(L_QUANTITY) AS SUM_QTY,
 SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
 SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS SUM_DISC_PRICE,
 SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT) * (1 + L_TAX)) AS SUM_CHARGE,
 AVG(L_QUANTITY) AS AVG_QTY,
 AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
 AVG(L_DISCOUNT) AS AVG_DISC,
 COUNT(*) AS COUNT_ORDER
FROM LINEITEM
WHERE
 L_SHIPDATE <= TO_DATE('1998-12-01','YYYY-MM-DD') - 119
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG, L_LINESTATUS;
```

Query 2 – Variant A

```
set_fetchrows=100;
SELECT
 S_ACCTBAL,
 S_NAME,
 N_NAME,
 P_PARTKEY,
 P_MFGR,
 S_ADDRESS,
 S_PHONE,
 S_COMMENT
FROM PARTS, SUPPLIER, PARTSUPP, NATION, REGION
WHERE P_PARTKEY = PS_PARTKEY
 AND S_SUPPKEY = PS_SUPPKEY
 AND P_SIZE = 49
 AND P_TYPE LIKE '%STEEL'
 AND S_NATIONKEY = N_NATIONKEY
 AND N_REGIONKEY = R_REGIONKEY
 AND R_NAME = 'AFRICA'
 AND (P_PARTKEY ,PS_SUPPLYCOST) IN
  (SELECT
    PS_PARTKEY,
    MIN(PS_SUPPLYCOST)
   FROM PARTSUPP, SUPPLIER, NATION, REGION
   WHERE
       S_SUPPKEY = PS_SUPPKEY
       AND S_NATIONKEY = N_NATIONKEY
       AND N_REGIONKEY = R_REGIONKEY
       AND R_NAME = 'AFRICA'
      GROUP BY PS_PARTKEY
  )
ORDER BY S_ACCTBAL DESC, N_NAME, S_NAME, P_PARTKEY;
```


Query 3 – Original

```
set_fetchrows=10;
SELECT
 L_ORDERKEY,
 SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS REVENUE,
 TO_CHAR(O_ORDERDATE, 'YYYY-MM-DD'),
 O_SHIPPRIORITY
FROM CUSTOMER, ORDERS, LINEITEM
WHERE
 C_MKTSEGMENT = 'MACHINERY'
 AND C_CUSTKEY = O_CUSTKEY
 AND L_ORDERKEY = O_ORDERKEY
 AND O_ORDERDATE < TO_DATE('1995-03-11','YYYY-MM-DD')
 AND L_SHIPDATE > TO_DATE('1995-03-11','YYYY-MM-DD')
GROUP BY L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY
ORDER BY REVENUE DESC, O_ORDERDATE;
```

Query 4 – Original

```
SELECT
 O_ORDERPRIORITY,
 COUNT(*) AS ORDER_COUNT
FROM ORDERS
WHERE
 O_ORDERDATE >= TO_DATE('1997-10-01','YYYY-MM-DD')
 AND O_ORDERDATE < ADD_MONTHS(TO_DATE('1997-10-01','YYYY-MM-DD'),3)
 AND EXISTS
  (SELECT
    *
   FROM LINEITEM
   WHERE
    L_ORDERKEY = O_ORDERKEY
    AND L_COMMITDATE < L_RECEIPTDATE
  )
GROUP BY O_ORDERPRIORITY
ORDER BY O_ORDERPRIORITY;
```

Query 5 – Original

```
SELECT
 N_NAME,
 SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS REVENUE
FROM CUSTOMER, ORDERS, LINEITEM, SUPPLIER, NATION, REGION
WHERE
 C_CUSTKEY = O_CUSTKEY
 AND O_ORDERKEY = L_ORDERKEY
 AND L_SUPPKEY = S_SUPPKEY
 AND C_NATIONKEY = S_NATIONKEY
 AND S_NATIONKEY = N_NATIONKEY
 AND N_REGIONKEY = R_REGIONKEY
 AND R_NAME = 'MIDDLE EAST'
 AND O_ORDERDATE >= TO_DATE('1994-01-01','YYYY-MM-DD')
 AND O_ORDERDATE < ADD_MONTHS(TO_DATE('1994-01-01','YYYY-MM-DD'),12)
GROUP BY N_NAME
ORDER BY REVENUE DESC;
```


Query 6 – Original

```
SELECT
 SUM(L_EXTENDEDPRICE * L_DISCOUNT) AS REVENUE
FROM LINEITEM
WHERE
 L_SHIPDATE >= TO_DATE('1997-01-01','YYYY-MM-DD')
 AND L_SHIPDATE < ADD_MONTHS(TO_DATE('1997-01-01','YYYY-MM-DD'),12)
 AND L_DISCOUNT BETWEEN 0.04 - 0.01 AND 0.04 + 0.01
 AND L_QUANTITY < 24;
```

Query 7 – Original

```
SELECT
 SUPP_NATION,
 CUST_NATION,
 YEAR,
 SUM(VOLUME) AS REVENUE
FROM
 (SELECT
   N1.N_NAME AS SUPP_NATION,
   N2.N_NAME AS CUST_NATION,
   TO_CHAR(L_SHIPDATE,'YYYY') AS YEAR,
   L_EXTENDEDPRICE * (1-L_DISCOUNT) AS VOLUME
  FROM SUPPLIER, LINEITEM, ORDERS, CUSTOMER, NATION N1, NATION N2
  WHERE
   S_SUPPKEY = L_SUPPKEY
   AND O_ORDERKEY = L_ORDERKEY
   AND C_CUSTKEY = O_CUSTKEY
   AND S_NATIONKEY = N1.N_NATIONKEY
   AND C_NATIONKEY = N2.N_NATIONKEY
   AND ((N1.N_NAME = 'UNITED STATES' AND N2.N_NAME = 'INDIA') OR
    (N1.N_NAME = 'INDIA' AND N2.N_NAME = 'UNITED STATES'))
   AND L_SHIPDATE BETWEEN TO_DATE('1995-01-01','YYYY-MM-DD') AND
    TO_DATE('1996-12-31','YYYY-MM-DD')
 ) SHIPPING
GROUP BY SUPP_NATION, CUST_NATION, YEAR
ORDER BY SUPP_NATION, CUST_NATION, YEAR;
```

Query 8 – Variant B

```
SELECT
 YEAR,
 SUM(DECODE(NATION, 'UNITED STATES', VOLUME, 0)) / SUM(VOLUME) AS
 MKT_SHARE
FROM
 (SELECT
   TO_CHAR(O_ORDERDATE,'YYYY') AS YEAR,
   L_EXTENDEDPRICE*(1-L_DISCOUNT) AS VOLUME,
   N2.N_NAME AS NATION
  FROM PARTS, SUPPLIER, LINEITEM, ORDERS, CUSTOMER, NATION N1, NATION
   N2, REGION
  WHERE
   P_PARTKEY = L_PARTKEY
   AND S_SUPPKEY = L_SUPPKEY
```



```
   AND L_ORDERKEY = O_ORDERKEY
   AND O_CUSTKEY = C_CUSTKEY
   AND C_NATIONKEY = N1.N_NATIONKEY
   AND N1.N_REGIONKEY = R_REGIONKEY
```

Query 8 – Variant B (Con't.)

```
   AND R_NAME = 'AMERICA'
   AND S_NATIONKEY = N2.N_NATIONKEY
   AND O_ORDERDATE BETWEEN TO_DATE ('1995-01-01','YYYY-MM-DD')
   AND TO_DATE('1996-12-31','YYYY-MM-DD')
   AND P_TYPE = 'SMALL BRUSHED STEEL'
  ) ALL_NATIONS
GROUP BY YEAR
ORDER BY YEAR;
```

Query 9 – Original

```
SELECT
 NATION,
 YEAR,
 SUM(AMOUNT) AS SUM_PROFIT
FROM
 (SELECT
  N_NAME AS NATION,
  TO_CHAR(O_ORDERDATE,'YYYY') AS YEAR,
  L_EXTENDEDPRICE * (1-L_DISCOUNT) - PS_SUPPLYCOST * L_QUANTITY AS
   AMOUNT
  FROM PARTS, SUPPLIER, LINEITEM, PARTSUPP, ORDERS, NATION
 WHERE
   S_SUPPKEY  = L_SUPPKEY
   AND PS_SUPPKEY = L_SUPPKEY
   AND PS_PARTKEY = L_PARTKEY
   AND P_PARTKEY  = L_PARTKEY
   AND O_ORDERKEY = L_ORDERKEY
   AND S_NATIONKEY = N_NATIONKEY
   AND P_NAME LIKE '%wheat%'
 ) PROFIT
 GROUP BY NATION, YEAR
 ORDER BY NATION, YEAR DESC;
```

Query 10 – Original

```
set_fetchrows=20;
SELECT
 C_CUSTKEY,
 C_NAME,
 SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT)) AS REVENUE,
 C_ACCTBAL,
 N_NAME,
 C_ADDRESS,
 C_PHONE,
 C_COMMENT
FROM CUSTOMER, ORDERS, LINEITEM, NATION
WHERE
 C_CUSTKEY = O_CUSTKEY
 AND L_ORDERKEY = O_ORDERKEY
 AND O_ORDERDATE >= TO_DATE('1995-01-01' ,'YYYY-MM-DD')
```



```
 AND O_ORDERDATE < ADD_MONTHS(TO_DATE('1995-01-01' ,'YYYY-MM-DD'),3)
 AND L_RETURNFLAG = 'R'
 AND C_NATIONKEY = N_NATIONKEY
GROUP BY C_CUSTKEY, C_NAME, C_ACCTBAL, C_PHONE, N_NAME, C_ADDRESS,
         C_COMMENT
ORDER BY REVENUE DESC;
```

Query 11 – Variant A

```
CREATE TABLE PART_VALUE0 (PARTNO INTEGER, VALUE NUMBER(20, 2) )
tablespace TS_S
;

INSERT INTO PART_VALUE0
SELECT
 PS_PARTKEY,
 SUM(PS_SUPPLYCOST*PS_AVAILQTY)
FROM PARTSUPP, SUPPLIER, NATION
WHERE
 PS_SUPPKEY = S_SUPPKEY
 AND S_NATIONKEY = N_NATIONKEY
 AND N_NAME = 'UNITED STATES'
GROUP BY PS_PARTKEY;

CREATE TABLE SUM_PART_VALUE0 (TOTAL_VALUE NUMBER(20, 2))
tablespace TS_S
;

INSERT INTO SUM_PART_VALUE0
SELECT
 SUM(PS_SUPPLYCOST*PS_AVAILQTY) * 0.0000010000
FROM PARTSUPP, SUPPLIER, NATION
WHERE
 PS_SUPPKEY = S_SUPPKEY
 AND S_NATIONKEY = N_NATIONKEY
 AND N_NAME = 'UNITED STATES';

SELECT
 PARTNO AS P_PARTKEY,
 VALUE
FROM PART_VALUE0
WHERE
 VALUE >
  (SELECT
    SUM(TOTAL_VALUE)
   FROM SUM_PART_VALUE0
  )
ORDER BY VALUE DESC;

DROP TABLE PART_VALUE0;

DROP TABLE SUM_PART_VALUE0;
```

Query 12 – Variant B

```
SELECT
 L_SHIPMODE,
 SUM(DECODE(O_ORDERPRIORITY,'1-URGENT',1,'2-HIGH',1,0)) AS
 HIGH_LINE_COUNT,
 SUM(DECODE(O_ORDERPRIORITY,'1-URGENT',0,'2-HIGH',0,1)) AS
```



```
 LOW_LINE_COUNT
FROM ORDERS, LINEITEM
WHERE
 O_ORDERKEY = L_ORDERKEY
 AND L_SHIPMODE IN ('SHIP','RAIL')
 AND L_COMMITDATE < L_RECEIPTDATE
 AND L_SHIPDATE < L_COMMITDATE
```

Query 12 – Variant B (Con't.)

```
 AND L_RECEIPTDATE >= TO_DATE('1993-01-01','YYYY-MM-DD')
 AND L_RECEIPTDATE < ADD_MONTHS(TO_DATE('1993-01-01',
   'YYYY-MM-DD'),12)
GROUP BY L_SHIPMODE
ORDER BY L_SHIPMODE;
```

Query 13 – Original

```
SELECT
 YEAR,
 SUM(REVENUE) AS REVENUE
FROM
 (SELECT
   TO_CHAR(O_ORDERDATE,'YYYY') AS YEAR,
   L_EXTENDEDPRICE * (1-L_DISCOUNT) AS REVENUE
  FROM LINEITEM, ORDERS
  WHERE
   O_ORDERKEY = L_ORDERKEY
   AND O_CLERK = 'Clerk#000000977'
   AND L_RETURNFLAG = 'R'
 ) PERFORMANCE
GROUP BY YEAR
ORDER BY YEAR;
```

Query 14 – Variant C

```
CREATE TABLE ALL_SALES0 (TYPE VARCHAR2(25), AMOUNT NUMBER(20, 2))
tablespace TS_S
;

CREATE TABLE SUM_PROMO_SALES0 (PROMO_AMOUNT NUMBER(20, 2))
tablespace TS_S
;

CREATE TABLE SUM_ALL_SALES0 (ALL_AMOUNT NUMBER(20, 2))
tablespace TS_S
;

INSERT INTO ALL_SALES0
SELECT
 P_TYPE,
 SUM(L_EXTENDEDPRICE *(1-L_DISCOUNT))
FROM  LINEITEM, PARTS
WHERE
 L_PARTKEY = P_PARTKEY
 AND L_SHIPDATE >= TO_DATE('1997-12-01','YYYY-MM-DD')
 AND L_SHIPDATE < ADD_MONTHS(TO_DATE('1997-12-01','YYYY-MM-DD'), 1)
GROUP BY P_TYPE;
```



```
INSERT INTO SUM_PROMO_SALES0
SELECT
 SUM(AMOUNT)
FROM  ALL_SALES0
WHERE
 TYPE LIKE 'PROMO%';
```

Query 14 – Variant C (Con't.)

```
INSERT INTO SUM_ALL_SALES0
SELECT
 SUM(AMOUNT)
FROM ALL_SALES0;

SELECT
 100.00*PROMO_AMOUNT/ALL_AMOUNT AS PROMO_REVENUE
FROM SUM_PROMO_SALES0, SUM_ALL_SALES0;

DROP TABLE ALL_SALES0;

DROP TABLE SUM_PROMO_SALES0;

DROP TABLE SUM_ALL_SALES0;
```

Query 15 – Variant B

```
CREATE TABLE REVENUE0 (SUPPLIER_NO INTEGER, TOTAL_REVENUE NUMBER(20,
   2))
tablespace TS_S;

INSERT INTO REVENUE0
SELECT
 L_SUPPKEY,
 SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT))
FROM  LINEITEM
WHERE
 L_SHIPDATE >= TO_DATE('1997-10-01','YYYY-MM-DD')
 AND L_SHIPDATE <  ADD_MONTHS(TO_DATE('1997-10-01','YYYY-MM-DD'),3)
GROUP BY L_SUPPKEY;

SELECT
 S_SUPPKEY,
 S_NAME,
 S_ADDRESS,
 S_PHONE,
 TOTAL_REVENUE
FROM  SUPPLIER, REVENUE0
WHERE
 S_SUPPKEY = SUPPLIER_NO
 AND TOTAL_REVENUE =
  (SELECT
    MAX(TOTAL_REVENUE)
   FROM REVENUE0
  )
ORDER BY S_SUPPKEY;

DROP TABLE REVENUE0;
```


Query 16 – Original

```
SELECT
 P_BRAND,
 P_TYPE,
 P_SIZE,
 COUNT(DISTINCT PS_SUPPKEY) AS SUPPLIER_CNT
FROM PARTSUPP,PARTS
```

Query 16 – Original (Con't.)

```
WHERE
 P_PARTKEY = PS_PARTKEY
 AND P_BRAND <> 'Brand#25'
 AND P_TYPE NOT LIKE 'STANDARD BRUSHED%'
 AND P_SIZE IN (35, 15 ,43 ,27 ,30, 8, 42, 17)
 AND PS_SUPPKEY NOT IN
  (SELECT
    S_SUPPKEY
   FROM SUPPLIER
   WHERE
    S_COMMENT LIKE '%Better Business Bureau%Complaints%'
  )
GROUP BY P_BRAND, P_TYPE, P_SIZE
ORDER BY SUPPLIER_CNT DESC, P_BRAND, P_TYPE, P_SIZE;
```

Query 17 – Original

```
SELECT
 SUM(L_EXTENDEDPRICE)/7.0 AS AVG_YEARLY
FROM LINEITEM, PARTS
WHERE
 P_PARTKEY = L_PARTKEY
 AND P_BRAND = 'Brand#25'
 AND P_CONTAINER = 'SM CAN'
 AND L_QUANTITY <
  (SELECT
    0.2* AVG(L_QUANTITY)
   FROM LINEITEM
   WHERE
    L_PARTKEY = P_PARTKEY);
```


User Registration/Evaluation Form

Please fill out and return to us this registration/evaluation form to help us keep you up to date with future revisions of this document and related new information products. Your effort will help us improve the quality of the future information products.

Name: _____
Title: _____
Company: _____
Address: _____
_____
_____
Phone Number: _____

Please evaluate the quality of this document:

	Excellent				Poor
Technical Accuracy	1	2	3	4	5
Organization	1	2	3	4	5
Clarity	1	2	3	4	5
Completeness	1	2	3	4	5
Timeliness	1	2	3	4	5

Please indicate the type of environment you have at your site:

Operating Systems	RDBMS	Processing Type
❑ SCO Unix	❑ Microsoft SQL Server	❑ On-Line Transaction Processing
❑ Microsoft Windows NT	❑ Sybase System 10	❑ Decision Support
❑ IBM OS/2	❑ Oracle8	❑ Batch Processing
❑ Novell Windows NT	❑ Other:	❑ Other:
❑ UnixWare		
❑ Other:		

Please indicate the type of information you would like us to provide in the future:

Topic	Operating Systems	RDBMS
❑ Configuration and Tuning	❑ Microsoft Windows NT	❑ Microsoft SQL Server
❑ Capacity Planning	❑ Novell Windows NT	❑ Sybase System 10
❑ Integration Information	❑ IBM OS/2	❑ Oracle8
❑ Competitive Analysis	❑ Novell UNIXWare	❑ Other:
❑ Systems Management	❑ SCO Unix	
❑ Other:	❑ Other:	

Additional Comments:

Return to:
Database Performance Engineering
Compaq Computer Corporation
MailCode 090803
20555 SH 249
Houston, Texas 77070

