



# HP configuration and load guide for Microsoft SQL Server 2008 Fast Track Data Warehouse

## Table of contents

Executive summary.....	2
Fast Track architecture overview .....	2
Important points and caveats.....	2
Building hardware components .....	3
Memory requirements.....	3
Host Bus Adapters (HBA) .....	4
HBA queue depth.....	4
MSA storage array settings .....	4
MPIO Settings.....	5
Allocating storage for Fast Track.....	7
Expanding TempDB .....	10
Creating databases.....	10
Creating partitions .....	13
Create tables .....	14
Creating indexed tables.....	14
Creating non-indexed tables.....	15
Creating table indexes.....	16
Loading data to staging tables .....	17
Loading data to final database.....	17
Loading indexed tables.....	17
Loading non-indexed tables.....	19
Summary .....	20
Appendix A .....	21
For more information.....	23

## Executive summary

Microsoft® SQL Server (SQL) Fast Track Data Warehouse for Hewlett-Packard (HP) servers, storage and networking products provides a prescriptive approach for balancing server, storage, network and software configurations for architecting Microsoft SQL Server 2008 data warehouse solutions. This SQL Fast Track configuration and load guide provides detailed steps to aid in the building of a SQL Fast Track solution while also providing the required steps to load new or existing data into the fast track environment. These guidelines are prescribed order to ensure that data is loaded sequentially optimizing performance, which is a key factor in providing the lower cost, high performance capability that fast track provides. HP has developed several ProLiant based Fast Track reference architecture which provide the base hardware needed to build out a solution up to 24TB in size. The examples used in this document are based on tests using the HP ProLiant DL385 based solution. All Fast Track reference architectures can be found [here](#) at HP ActiveAnswers.

**Target audience:** The target audience for this document consists of IT planners, architects, DBAs, CIOs, CTOs, and business intelligence (BI) users who are looking to learn how to implement a SQL 2008 Fast Track solution.

This white paper describes testing performed by HP in August, 2009.

## Fast Track architecture overview

This document is a reference configuration companion document to the Microsoft SQL Server Technical Article, "Implementing a SQL Server Fast Track Data Warehouse", which describes a repeatable architectural approach for implementing a scalable model for a symmetric multiprocessor (SMP)-based Microsoft SQL Server 2008 data warehouse. The end result of the process described in this companion guide represents a recommended minimal SQL Server 2008 configuration, inclusive of all the software and hardware, required to achieve and maintain a baseline level of "out of box" scalable performance when deploying SQL Server 2008 data warehousing (SSDW) sequential data access workload scenarios versus traditional random I/O methods.

This document provides specific details about the configuration and bill of materials for one such reference architecture, based on the HP ProLiant DL385 G6 and the HP StorageWorks 2000fc Modular Smart Array G2 (MSA2000fc). This configuration is targeted at a data mart environment of roughly 40 users. It is optimized at 6TB of compressed data and scalable to 12TB of user data capacity.

## Important points and caveats

- The configuration described here and the approach detailed in this configuration and load guide is exclusively designed for, and is only applicable to, sequential data workloads. Use of this approach on other workload types is not appropriate and may yield configurations that are inefficient.
- **ALL** procedures and best practices defined in this guide must be implemented in their entirety in order to preserve and maintain the sequential order of the data and sequential I/O against the data.
- The recommendations detailed in this paper have been reached through recent lab-based testing performed by HP on HP hardware.

**Figure 1.** Fast Track 6TB reference architecture with the ProLiant DL385 G6 and (3) StorageWorks MSA2312fc



A complete SQL Server DBMS (Database Management System) configuration, or "stack," is a collection of all the components that are configured to work together to support the database application. This includes the physical server hardware (with its BIOS settings and appropriate firmware releases), memory, CPU (number, type, clock and bus speed, cache and core count), operating system settings, the storage arrays and interconnects, disk (capacity, form factor and spindle speeds), database, DBMS settings and configuration, and even table layouts, indexing strategy, and physical data structures.

The primary goal of Fast Track, which is also a common goal when designing most data center infrastructures, is a balanced configuration where all components can be utilized to their maximum capability. Architecting and maintaining a balance prevents over subscribing certain components within the stack to a point where the expected performance is not realized; understanding the performance limits of your configuration can help prevent wasted cost for components that will never realize their potential due to other constraints within the stack.

## Building hardware components

Before installing and configuring Microsoft SQL Server 2008, several hardware and software configuration steps must be performed. This section lists some of the settings that can be made to ensure optimum performance for all Fast Track components.

### Memory requirements

Minimum memory requirements for Fast Track configurations are based on the number of CPU cores residing in the system. In general, the system should have 4GB of memory per CPU core. For the DL385 G6 configuration used in this document, we have 12 cores with a total memory size of 48 GB. These memory requirements ensure that adequate performance is achieved during intensive load operations as well as general query processing.

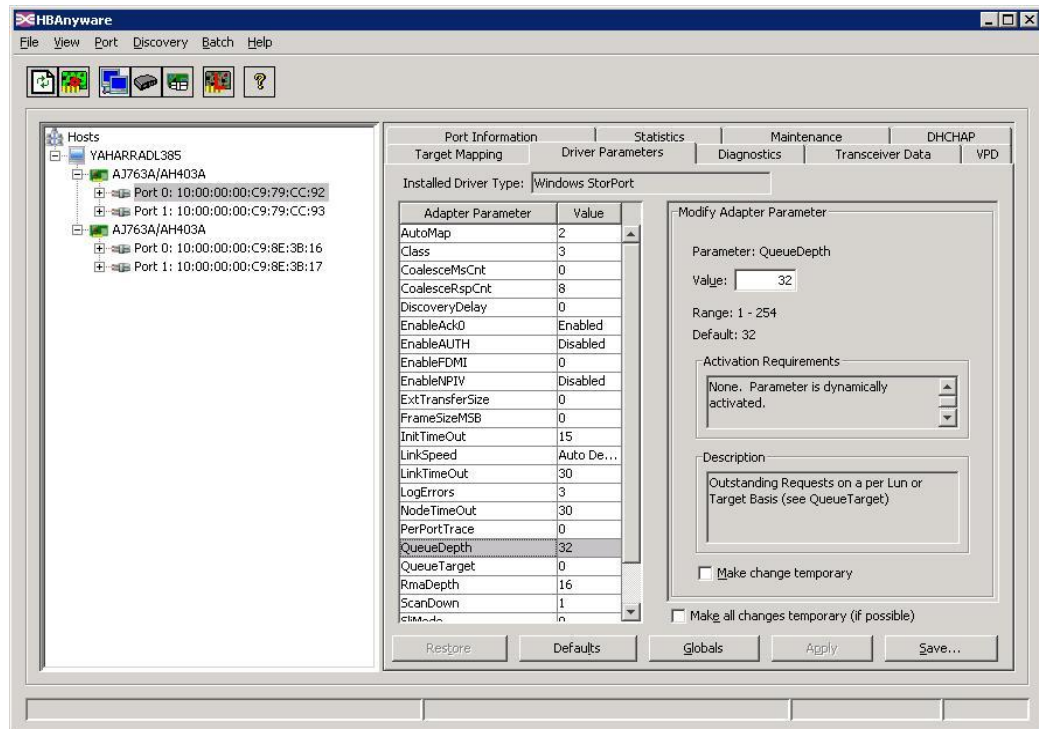
## Host Bus Adapters (HBA)

In our example configuration utilizing the DL385 G6, 2 HP StorageWorks 82E 8GB Fiber Channel HBAs are specified. These HBA cards are x8 PCI Express 2.0 cards. In standard form, the DL385 G6 server has 1 available x4 and 1 available x8 PCI Express slot, so the HBAs will be inserted into these 2 remaining slots. In the case of the DL585 G6, there are 3 available x8 PCI-E slots, so these slots should be utilized for 3 out of 4 HBAs specified (with the final card residing in a x4 slot), and finally the specification for the DL785G6 is for 8 HBAs, 3 of which should reside in the x8 slots with the remaining 5 cards in x4 slots.

## HBA queue depth

Queue depth for all testing was set at the default of 32. During the testing, raising the queue depth beyond 32 didn't show any significant increase in performance, though other query and data types might benefit from a change. Shown below in figure 2, is the setting to change the queue depth on the Emulex based HBAs. The HBAAnywhere program can be downloaded from the HP website under the "drivers" section for the 82E HBAs.

**Figure 2.** Queue depth settings for the HP 82E HBA

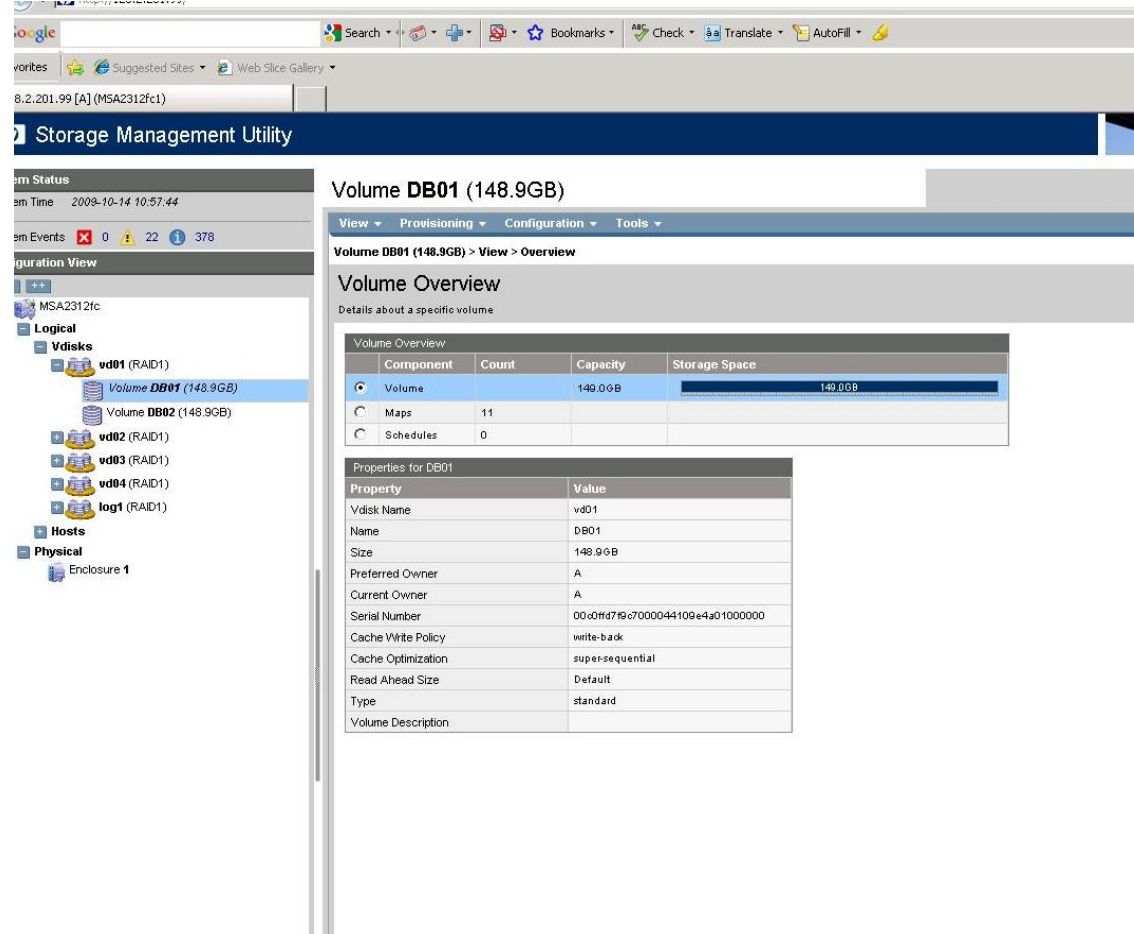


## MSA storage array settings

Creating LUNs within the HP StorageWorks Modular Smart Array (MSA) interface first involves creating "Vdisks". A Vdisk is simply a RAID set in which you can create LUNs. For each MSA2000fc array, 5 Vdisks will be created. Each Vdisk will be a 2 disk RAID1 array. When creating the Vdisk, all default settings should be used. Chunk size defaults to 64K, and the controller setting default to Auto which tries to allocate Vdisks by alternating preferred and owned controllers.

Once Vdisks have been created, two same size “Volumes” (MSA equivalent of a LUN) must be created. When creating the Volumes, ensure that the Cache Optimization setting is set to “Super Sequential”. This optimizes the controller’s cache for sequential reads, rather than a mix of random and sequential. Figure 3 below shows an example of a Fast Track volume in the MSA2000fc.

**Figure 3.** Volume settings



## MPIO Settings

When using the preferred operating system, Microsoft Windows® Server 2008, the Multipath I/O (MPIO) feature should be added through the Server Manager program. Installing this feature enables the Microsoft Device Specific Module (DSM). After installing MPIO, the MPIO method should be checked in the “Disk Management” section of Server Manager to ensure that it’s set to “Round-Robin with Subset” and only 1 path for each disk or LUN shows up in Disk Management. Round Robin with subset marks a set of paths as “Active” and a set of paths as “Passive”, and performs Round Robin operations on the “Active” paths only, failing over to the “Passive” paths when the active paths fail. In addition, the MSA2000fc storage controllers require that the following MPCLAIM command be executed after configuring MPIO:

Open a command prompt window, and run the following command:

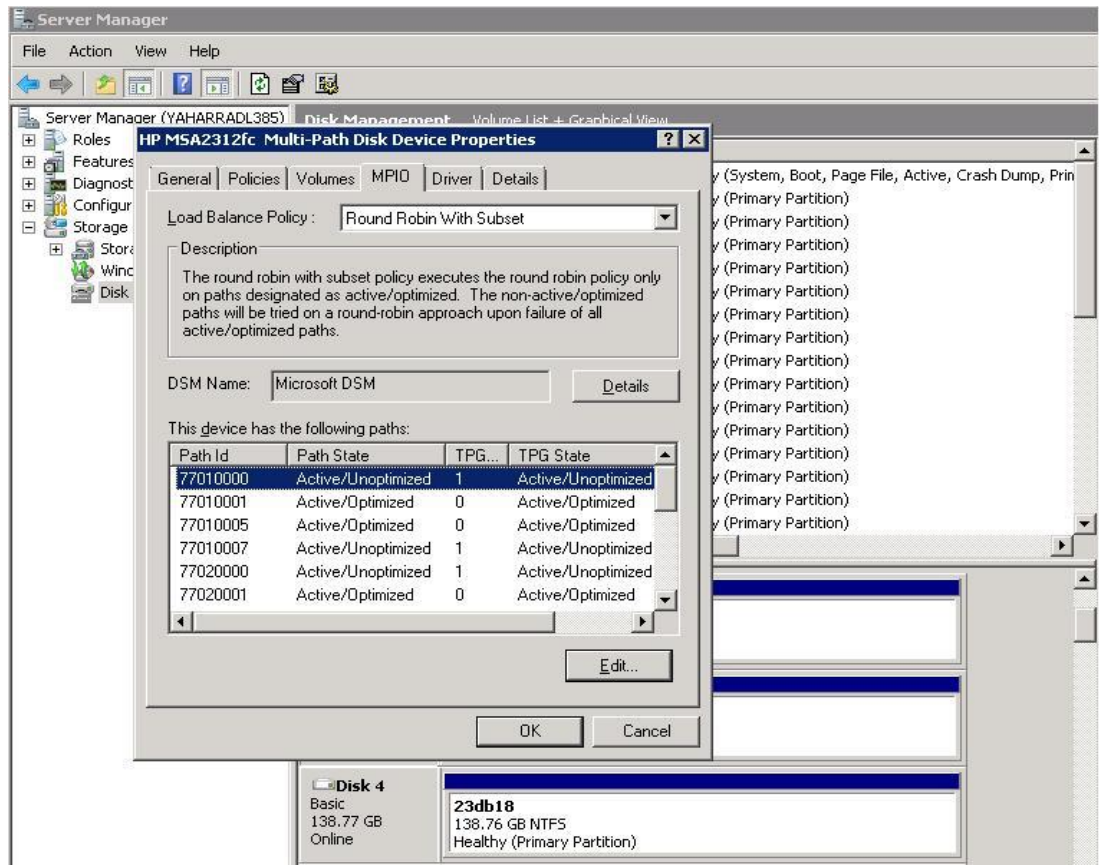
```
mpclaim -r -l -d "HP MSA2"
```

**NOTE:** There must be exactly six spaces between **HP** and **MSA2**.

**NOTE:** As soon as the **mpclaim** command completes, the host will be rebooted.

Refer to figure 4 below for the proper MPIO setting for all Fast Track LUNs:

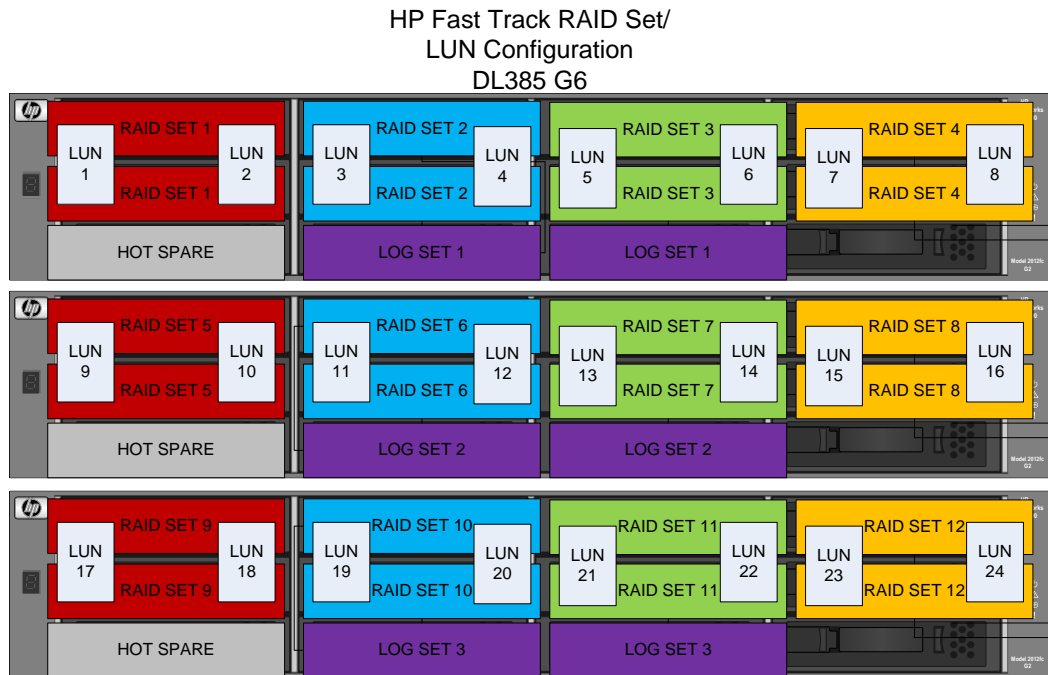
**Figure 4.** MPIO setting



# Allocating storage for Fast Track

All current Fast Track configurations leverage the HP StorageWorks MSA2000fc G2 array, which allows for dual reads when drives are mirrored. For sequential data reads for data warehouse queries, this capability enables tremendous throughput per storage volume—up to 240 MB/s. The Fast Track approach and supporting storage array architecture are optimized for sequential reads. To support a non-optimized random I/O data warehousing workload, up to 2 to 3 times the number of drives would be required to achieve the same throughput. For Fast Track, in order to achieve a “core balanced configuration, the general rule is to have one 2-disk RAID1 set for every core to store the database files. The example configuration used here, the DL385 G6 configuration utilizes a total of 12 cores, so we allocated 24 disks in 12 RAID1 configurations. Since the MSA2000fc array features “dual-read” capability whereby 2 simultaneous queries can perform read operations on the same RAID set (1 query reads from 1 of the 2 drives in the RAID set), we also allocate 2 LUNs per RAID set, for a total of 24 LUNs to store database files. One additional RAID1 pair is allocated in each MSA2000fc for transaction logs. Setting aside 1 drive in each array for hot spare purposes, we have a total of 33 drives allocated for Fast Track within the 3 MSA2000fcs as shown below in figure 5.

Figure 5. LUN Layout with (3) MSA2312fc controllers



**Table 1.** LUN summary

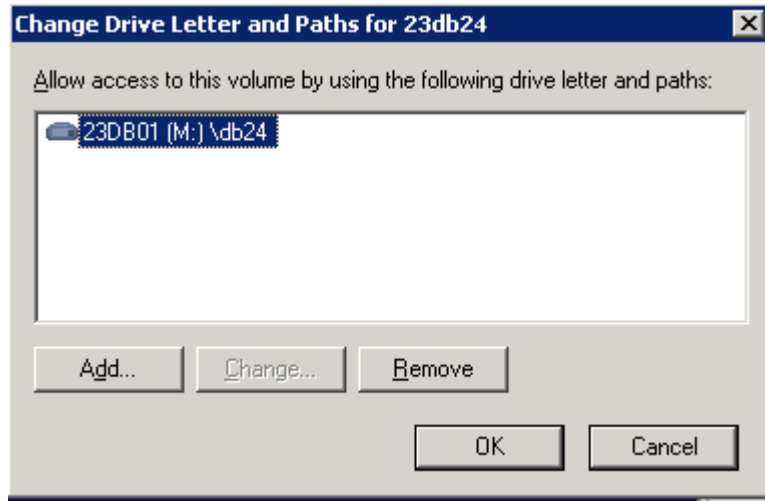
The table below summarizes the LUN configuration using 33 300GB drives:

LUN	Function	Formatted Size
1	Database Storage	139GB
2	Database Storage	139GB
3	Database Storage	139GB
4	Database Storage	139GB
5	Database Storage	139GB
6	Database Storage	139GB
7	Database Storage	139GB
8	Database Storage	139GB
9	Database Storage	139GB
10	Database Storage	139GB
11	Database Storage	139GB
12	Database Storage	139GB
13	Database Storage	139GB
14	Database Storage	139GB
15	Database Storage	139GB
16	Database Storage	139GB
17	Database Storage	139GB
18	Database Storage	139GB
19	Database Storage	139GB
20	Database Storage	139GB
21	Database Storage	139GB
22	Database Storage	139GB
23	Database Storage	139GB
24	Database Storage	139GB
LOG1	Transaction logs	279GB
LOG2	Transaction logs	279GB
LOG3	Transaction logs	279GB
<b>TOTAL Database Storage</b>		<b>3.34TB</b>
<b>TOTAL log Storage</b>		<b>837GB</b>

Since we have a total of 27 LUNs, we cannot simply assign a drive letter to each LUN, so the best method is to use Mount Points. We can take the first database LUN, assign it a drive letter such as M:, then create 26 additional empty directories within the "M:" drive. These directories will correspond to each of the database and transaction log LUNS created on the storage arrays. The example used here creates 24 directories for the database LUNs (db01 to db24) and 3 log LUNs (log01 to log03). Once the empty directories are created, we can go into Windows Disk Management and mount each LUN to the corresponding directory under M.

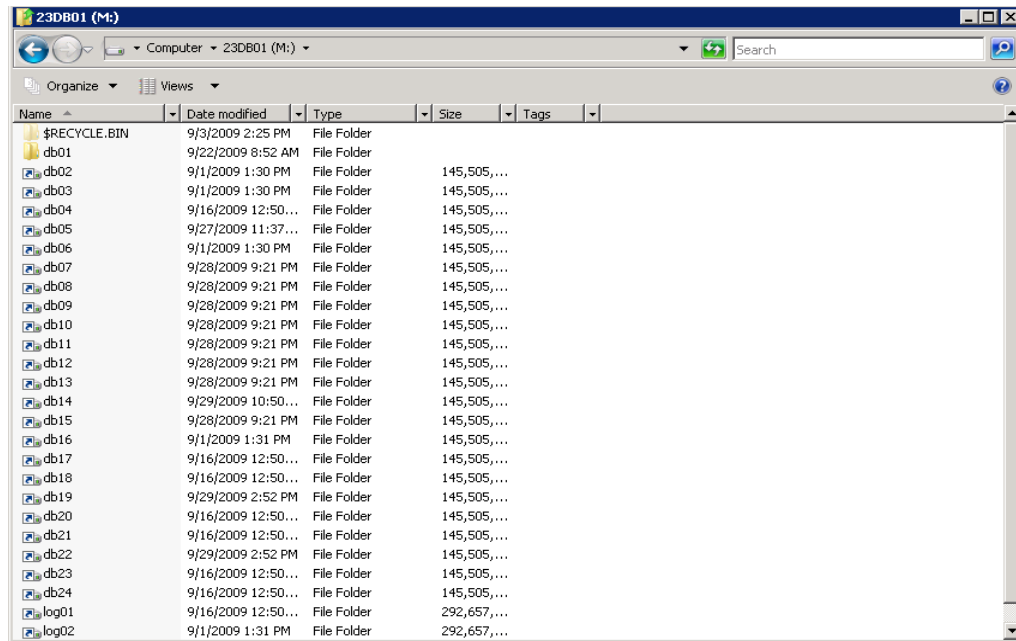


Figure 6. Setting up mount points



Once complete, all LUNs will now be contained within a sub-directory of the M: drive and M:\db02 will refer to LUN #2, etc.

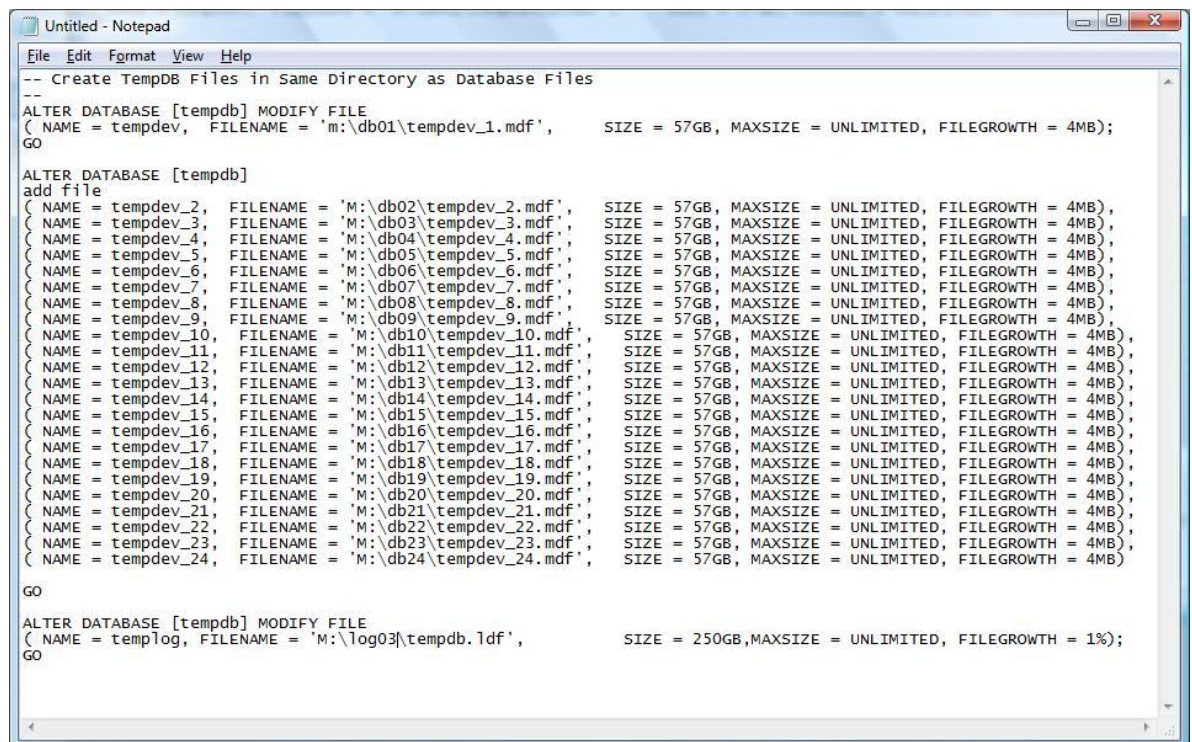
Figure 7. View of M: drive with all mount points



## Expanding TempDB

When working with partitioned databases, extensive use of TempDB is employed to sort and order data. To ensure peak performance from operations utilizing TempDB, adequate space must be allocated. In general space equal to the size of the largest table in the production database should be used as minimum guidelines for the size of TempDB storage. With Fast Track, the TempDB files can reside on the same LUNs as the production database files. In figure 8 below, the SQL script creates 24 files for TempDB with a size of 57GB each. If more TempDB performance is desired, than TempDB files can be created on a separate set of disk spindles, but testing has shown that adequate performance is achieved with files residing on the same LUNs as the production DB files.

**Figure 8.** TempDB expansion script



```
Untitled - Notepad
File Edit Format View Help
-- Create TempDB Files in Same Directory as Database Files
--
ALTER DATABASE [tempdb] MODIFY FILE
(NAME = tempdev, FILENAME = 'm:\db01\tempdev_1.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB);
GO

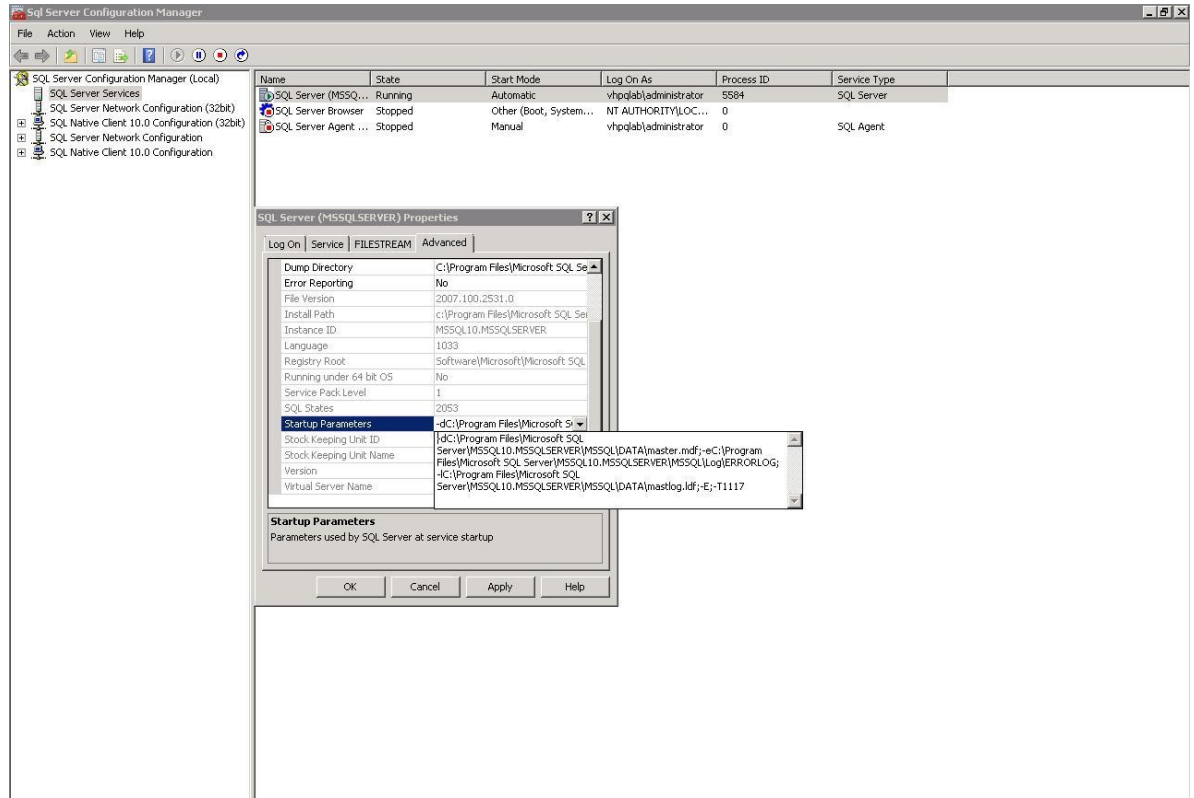
ALTER DATABASE [tempdb]
add file
(NAME = tempdev_2, FILENAME = 'M:\db02\tempdev_2.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_3, FILENAME = 'M:\db03\tempdev_3.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_4, FILENAME = 'M:\db04\tempdev_4.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_5, FILENAME = 'M:\db05\tempdev_5.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_6, FILENAME = 'M:\db06\tempdev_6.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_7, FILENAME = 'M:\db07\tempdev_7.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_8, FILENAME = 'M:\db08\tempdev_8.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_9, FILENAME = 'M:\db09\tempdev_9.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_10, FILENAME = 'M:\db10\tempdev_10.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_11, FILENAME = 'M:\db11\tempdev_11.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_12, FILENAME = 'M:\db12\tempdev_12.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_13, FILENAME = 'M:\db13\tempdev_13.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_14, FILENAME = 'M:\db14\tempdev_14.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_15, FILENAME = 'M:\db15\tempdev_15.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_16, FILENAME = 'M:\db16\tempdev_16.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_17, FILENAME = 'M:\db17\tempdev_17.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_18, FILENAME = 'M:\db18\tempdev_18.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_19, FILENAME = 'M:\db19\tempdev_19.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_20, FILENAME = 'M:\db20\tempdev_20.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_21, FILENAME = 'M:\db21\tempdev_21.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_22, FILENAME = 'M:\db22\tempdev_22.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_23, FILENAME = 'M:\db23\tempdev_23.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB),
(NAME = tempdev_24, FILENAME = 'M:\db24\tempdev_24.mdf', SIZE = 57GB, MAXSIZE = UNLIMITED, FILEGROWTH = 4MB);
GO

ALTER DATABASE [tempdb] MODIFY FILE
(NAME = templog, FILENAME = 'M:\log03\tempdb.ldf', SIZE = 250GB, MAXSIZE = UNLIMITED, FILEGROWTH = 1%);
GO
```

## Creating databases

Once the hardware is configured and the LUNs have been created, the next step is to build the staging and production Fast Track databases. When installing Microsoft SQL Server to support Fast Track, SQL Server 2008 Enterprise edition should be installed and configured to use the “-E” option and also to enable the 1117 trace flag by using the -T1117 parameter upon startup. The “-E” option increases the number of extents that are used for each file in a filegroup, while the 1117 trace flag ensures even expansion of all files within a filegroup. This startup option can be configured using the SQL Server Configuration Manager program, and editing the “Startup Parameters” field. Simply add a semi-colon to the end, then the -E switch, another semicolon and then the -T1117 option.

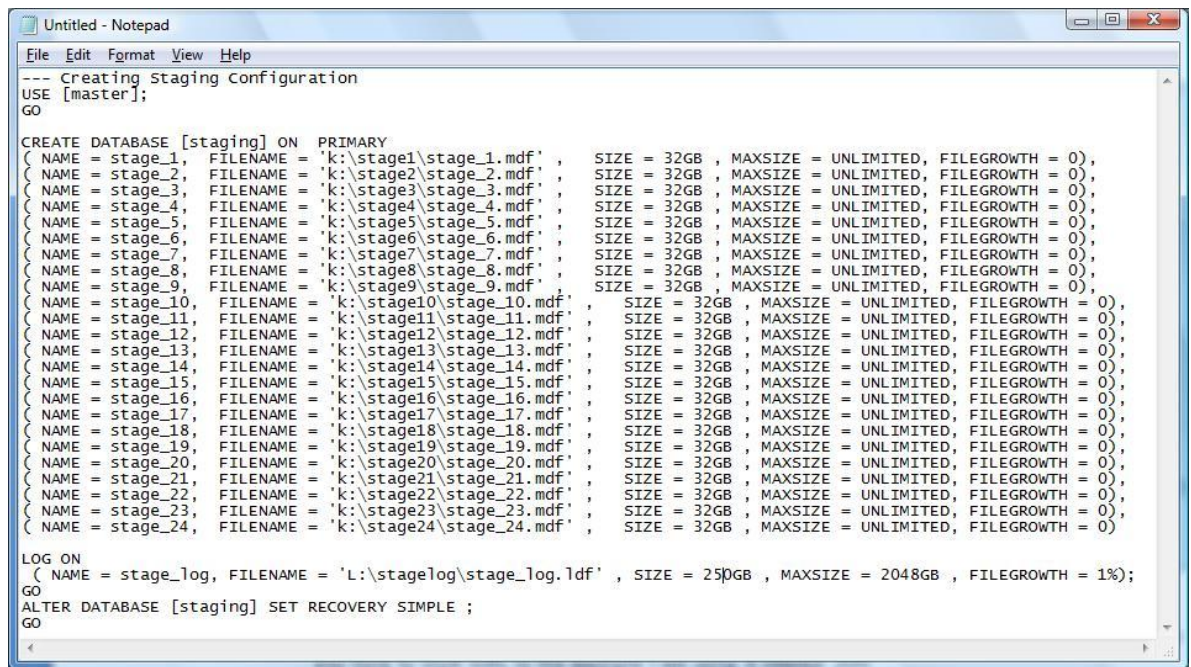
Figure 9. Setting SQL 2008 startup options



After installing SQL Server, the next step is to create the staging and production databases. The Staging Database is created on a separate set of disk spindles (internal server disks) and is used to hold the tables that will be ordered and indexed before we sequentially load the data into the final production database. In order to ensure that data can be loaded quickly and sequentially into the final production database, both the staging and production are partitioned based on the index keys. In the example depicted in this whitepaper we used a standard 1 TB TPC-H<sup>1</sup> database configuration. This database was used for example purposes only, and no performance tests were run against this database. For TPC-H, both the “Lineitem” and “Orders” tables are indexed and partitioned based on the “Shipdate” parameter. For database storage, two same size database files are allocated within the LUN. The MSA2000fc G2 storage array has a “dual-read” feature that allows a program such as SQL Server to read data from both drives in a RAID1 array simultaneously. So if we create multiple database files on LUNs created on a RAID1 set, we can increase read throughput by using these simultaneous reads. The SQL scripts in figures 10 and 11 are example of what was used to create both the staging and production databases and the associated files.

<sup>1</sup> TPC-H is a trademark of the Transaction Performance Processing Council and a complete list of all benchmarks can be found at [www.tpc.org](http://www.tpc.org)

Figure 10. Creating staging database



```
Untitled - Notepad
File Edit Format View Help
--- Creating staging Configuration
USE [master];
GO

CREATE DATABASE [staging] ON PRIMARY
( NAME = stage_1, FILENAME = 'k:\stage1\stage_1.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_2, FILENAME = 'k:\stage2\stage_2.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_3, FILENAME = 'k:\stage3\stage_3.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_4, FILENAME = 'k:\stage4\stage_4.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_5, FILENAME = 'k:\stage5\stage_5.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_6, FILENAME = 'k:\stage6\stage_6.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_7, FILENAME = 'k:\stage7\stage_7.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_8, FILENAME = 'k:\stage8\stage_8.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_9, FILENAME = 'k:\stage9\stage_9.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_10, FILENAME = 'k:\stage10\stage_10.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_11, FILENAME = 'k:\stage11\stage_11.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_12, FILENAME = 'k:\stage12\stage_12.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_13, FILENAME = 'k:\stage13\stage_13.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_14, FILENAME = 'k:\stage14\stage_14.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_15, FILENAME = 'k:\stage15\stage_15.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_16, FILENAME = 'k:\stage16\stage_16.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_17, FILENAME = 'k:\stage17\stage_17.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_18, FILENAME = 'k:\stage18\stage_18.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_19, FILENAME = 'k:\stage19\stage_19.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_20, FILENAME = 'k:\stage20\stage_20.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_21, FILENAME = 'k:\stage21\stage_21.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_22, FILENAME = 'k:\stage22\stage_22.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_23, FILENAME = 'k:\stage23\stage_23.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = stage_24, FILENAME = 'k:\stage24\stage_24.mdf' , SIZE = 32GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0)

LOG ON
( NAME = stage_log, FILENAME = 'L:\stage1og\stage_log.ldf' , SIZE = 250GB , MAXSIZE = 2048GB , FILEGROWTH = 1%);
GO
ALTER DATABASE [staging] SET RECOVERY SIMPLE ;
GO
```

In this example, 24 database files are created to be consistent with the production database. Each database file is set to 32GB, but can be adjusted based on the total amount of storage that you have to work with. In this example we are using 6 internal 300GB SAS drives. Four drives are used for the staging DB and created using a RAID5 configuration, while two drives are configured in a RAID1 array for transaction logs. If you have a need to load a larger amount of data, additional external storage may be needed to accommodate the staging database. In the next example, the script to create the final production database (TPC-H in this example) is shown. Two database files are created for each LUN and 3 transaction log files are created, one for each of the MSA2000fc storage arrays.

Figure 11. Creating production database

```
File Edit Format View Help
-- Creating ITB Configuration
-- M: Drive uses Mount Points for DB02-DB24 and LOG01-LOG03
USE [master];
GO

CREATE DATABASE [tpch] ON PRIMARY
( NAME = tpch_1a, FILENAME = 'M:\db01\tpch_1a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_2a, FILENAME = 'M:\db02\tpch_2a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_3a, FILENAME = 'M:\db03\tpch_3a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_4a, FILENAME = 'M:\db04\tpch_4a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_5a, FILENAME = 'M:\db05\tpch_5a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_6a, FILENAME = 'M:\db06\tpch_6a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_7a, FILENAME = 'M:\db07\tpch_7a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_8a, FILENAME = 'M:\db08\tpch_8a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_9a, FILENAME = 'M:\db09\tpch_9a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_10a, FILENAME = 'M:\db10\tpch_10a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_11a, FILENAME = 'M:\db11\tpch_11a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_12a, FILENAME = 'M:\db12\tpch_12a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_13a, FILENAME = 'M:\db13\tpch_13a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_14a, FILENAME = 'M:\db14\tpch_14a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_15a, FILENAME = 'M:\db15\tpch_15a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_16a, FILENAME = 'M:\db16\tpch_16a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_17a, FILENAME = 'M:\db17\tpch_17a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_18a, FILENAME = 'M:\db18\tpch_18a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_19a, FILENAME = 'M:\db19\tpch_19a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_20a, FILENAME = 'M:\db20\tpch_20a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_21a, FILENAME = 'M:\db21\tpch_21a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_22a, FILENAME = 'M:\db22\tpch_22a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_23a, FILENAME = 'M:\db23\tpch_23a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_24a, FILENAME = 'M:\db24\tpch_24a.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0)

LOG ON
( NAME = tpch_log, FILENAME = 'M:\log01\tpch_log.ldf' , SIZE = 250GB , MAXSIZE = 2048GB , FILEGROWTH = 1%) ,
( NAME = tpch_log2, FILENAME = 'M:\log02\tpch_log2.ldf' , SIZE = 250GB , MAXSIZE = 2048GB , FILEGROWTH = 1%) ,
( NAME = tpch_log3, FILENAME = 'M:\log03\tpch_log3.ldf' , SIZE = 250GB , MAXSIZE = 2048GB , FILEGROWTH = 1%)

GO

ALTER DATABASE [tpch] SET COMPATIBILITY_LEVEL = 100
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [tpch].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO

ALTER DATABASE [tpch] SET DISABLE_BROKER
ALTER DATABASE [tpch] SET ALLOW_SNAPSHOT_ISOLATION OFF
ALTER DATABASE [tpch] SET PARAMETERIZATION SIMPLE
ALTER DATABASE [tpch] SET READ_COMMITTED_SNAPSHOT OFF
ALTER DATABASE [tpch] SET RECOVERY SIMPLE
GO

ALTER DATABASE [tpch]
ADD FILE
( NAME = tpch_1b, FILENAME = 'M:\db01\tpch_1b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_2b, FILENAME = 'M:\db02\tpch_2b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_3b, FILENAME = 'M:\db03\tpch_3b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_4b, FILENAME = 'M:\db04\tpch_4b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_5b, FILENAME = 'M:\db05\tpch_5b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_6b, FILENAME = 'M:\db06\tpch_6b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_7b, FILENAME = 'M:\db07\tpch_7b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_8b, FILENAME = 'M:\db08\tpch_8b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_9b, FILENAME = 'M:\db09\tpch_9b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_10b, FILENAME = 'M:\db10\tpch_10b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_11b, FILENAME = 'M:\db11\tpch_11b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_12b, FILENAME = 'M:\db12\tpch_12b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_13b, FILENAME = 'M:\db13\tpch_13b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_14b, FILENAME = 'M:\db14\tpch_14b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_15b, FILENAME = 'M:\db15\tpch_15b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_16b, FILENAME = 'M:\db16\tpch_16b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_17b, FILENAME = 'M:\db17\tpch_17b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_18b, FILENAME = 'M:\db18\tpch_18b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_19b, FILENAME = 'M:\db19\tpch_19b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_20b, FILENAME = 'M:\db20\tpch_20b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_21b, FILENAME = 'M:\db21\tpch_21b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_22b, FILENAME = 'M:\db22\tpch_22b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_23b, FILENAME = 'M:\db23\tpch_23b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0) ,
( NAME = tpch_24b, FILENAME = 'M:\db24\tpch_24b.mdf' , SIZE = 40GB , MAXSIZE = UNLIMITED, FILEGROWTH = 0)

TO FILEGROUP [primary];
GO
```

**Note:** Fast Track does not support “Full” recovery mode. As shown in this example, recovery mode is set to “Simple”. This does not provide for transaction log based database recovery as log files are deleted after they are written into the database. For performance and capacity reasons, this is the only supported recovery mode for Fast Track.

## Creating partitions

Creating database partitions in the staging database helps to order the data before inserting into the final production database and provides faster access to data with a minimal set of indexes. The following example uses a partitions scheme that creates 169 partitions based on a range of ship dates. Matching partitions are created in the final production database as well. The data from each partition will then be inserted into the corresponding partition in the final database using a SELECT/INSERT statement for all indexed tables.

Figure 12. Partition script

```
File Edit Format View Help
USE staging;

CREATE PARTITION FUNCTION pfmonthly_168 (DATETIME)
AS RANGE RIGHT FOR VALUES (
'19920101', '19920115', '19920201', '19920215', '19920301', '19920315', '19920401', '19920415', '19920501', '19920515', '19920601', '19920615',
'19920701', '19920715', '19920801', '19920815', '19920901', '19920915', '19921001', '19921015', '19921101', '19921115', '19921201', '19921215',
'19930101', '19930115', '19930201', '19930215', '19930301', '19930315', '19930401', '19930415', '19930501', '19930515', '19930601', '19930615',
'19930701', '19930715', '19930801', '19930815', '19930901', '19930915', '19931001', '19931015', '19931101', '19931115', '19931201', '19931215',
'19940101', '19940115', '19940201', '19940215', '19940301', '19940315', '19940401', '19940415', '19940501', '19940515', '19940601', '19940615',
'19940701', '19940715', '19940801', '19940815', '19940901', '19940915', '19941001', '19941015', '19941101', '19941115', '19941201', '19941215',
'19950101', '19950115', '19950201', '19950215', '19950301', '19950315', '19950401', '19950415', '19950501', '19950515', '19950601', '19950615',
'19950701', '19950715', '19950801', '19950815', '19950901', '19950915', '19951001', '19951015', '19951101', '19951115', '19951201', '19951215',
'19960101', '19960115', '19960201', '19960215', '19960301', '19960315', '19960401', '19960415', '19960501', '19960515', '19960601', '19960615',
'19960701', '19960715', '19960801', '19960815', '19960901', '19960915', '19961001', '19961015', '19961101', '19961115', '19961201', '19961215',
'19970101', '19970115', '19970201', '19970215', '19970301', '19970315', '19970401', '19970415', '19970501', '19970515', '19970601', '19970615',
'19970701', '19970715', '19970801', '19970815', '19970901', '19970915', '19971001', '19971015', '19971101', '19971115', '19971201', '19971215',
'19980101', '19980115', '19980201', '19980215', '19980301', '19980315', '19980401', '19980415', '19980501', '19980515', '19980601', '19980615',
'19980701', '19980715', '19980801', '19980815', '19980901', '19980915', '19981001', '19981015', '19981101', '19981115', '19981201', '19981215'
);

use tpch;

CREATE PARTITION FUNCTION pfmonthly_168 (DATETIME)
AS RANGE RIGHT FOR VALUES (
'19920101', '19920115', '19920201', '19920215', '19920301', '19920315', '19920401', '19920415', '19920501', '19920515', '19920601', '19920615',
'19920701', '19920715', '19920801', '19920815', '19920901', '19920915', '19921001', '19921015', '19921101', '19921115', '19921201', '19921215',
'19930101', '19930115', '19930201', '19930215', '19930301', '19930315', '19930401', '19930415', '19930501', '19930515', '19930601', '19930615',
'19930701', '19930715', '19930801', '19930815', '19930901', '19930915', '19931001', '19931015', '19931101', '19931115', '19931201', '19931215',
'19940101', '19940115', '19940201', '19940215', '19940301', '19940315', '19940401', '19940415', '19940501', '19940515', '19940601', '19940615',
'19940701', '19940715', '19940801', '19940815', '19940901', '19940915', '19941001', '19941015', '19941101', '19941115', '19941201', '19941215',
'19950101', '19950115', '19950201', '19950215', '19950301', '19950315', '19950401', '19950415', '19950501', '19950515', '19950601', '19950615',
'19950701', '19950715', '19950801', '19950815', '19950901', '19950915', '19951001', '19951015', '19951101', '19951115', '19951201', '19951215',
'19960101', '19960115', '19960201', '19960215', '19960301', '19960315', '19960401', '19960415', '19960501', '19960515', '19960601', '19960615',
'19960701', '19960715', '19960801', '19960815', '19960901', '19960915', '19961001', '19961015', '19961101', '19961115', '19961201', '19961215',
'19970101', '19970115', '19970201', '19970215', '19970301', '19970315', '19970401', '19970415', '19970501', '19970515', '19970601', '19970615',
'19970701', '19970715', '19970801', '19970815', '19970901', '19970915', '19971001', '19971015', '19971101', '19971115', '19971201', '19971215',
'19980101', '19980115', '19980201', '19980215', '19980301', '19980315', '19980401', '19980415', '19980501', '19980515', '19980601', '19980615',
'19980701', '19980715', '19980801', '19980815', '19980901', '19980915', '19981001', '19981015', '19981101', '19981115', '19981201', '19981215'
);

use staging;

CREATE PARTITION SCHEME psmonthly_168_PRIMARY
AS PARTITION [pfmonthly_168]
ALL TO ([PRIMARY]);

use tpch;

CREATE PARTITION SCHEME psmonthly_168_PRIMARY
AS PARTITION [pfmonthly_168]
ALL TO ([PRIMARY]);
```

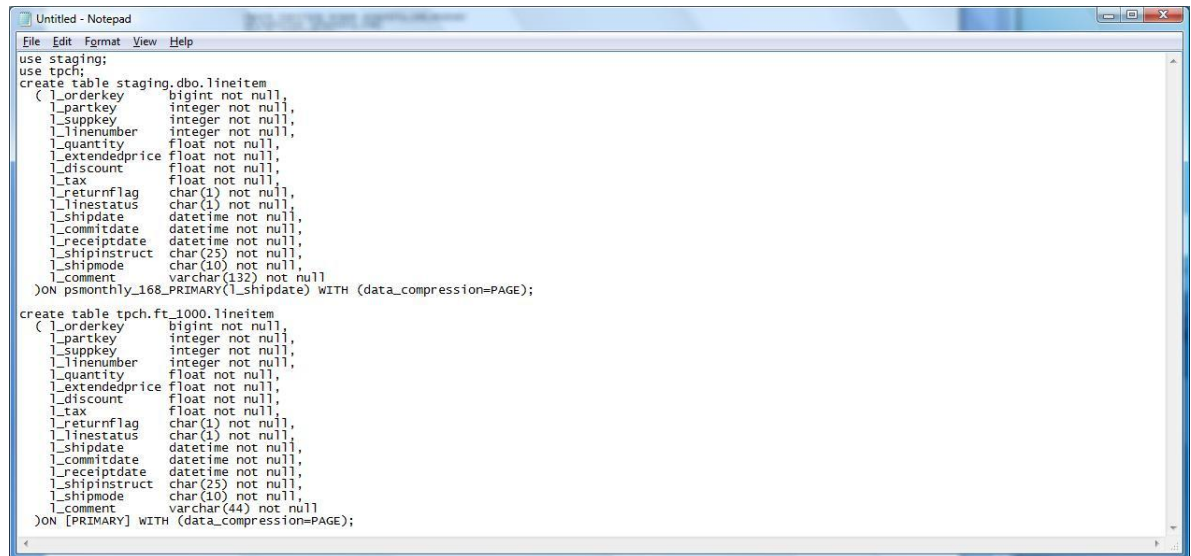
## Create tables

Two types of tables are created for Fast Track, indexed and non-indexed. The following sections detail the creation of these 2 types of tables.

### Creating indexed tables

The tables that will be indexed must be created in both the staging and production databases. The remaining tables in the database need only be created in the final production database. In our example using the TPC-H database, we have two tables, Lineitem and Orders which will be indexed. The script example in figure 13 shows the creation of the lineitem table on both the staging and production database.

Figure 13. Creating indexed tables



```
Untitled - Notepad
File Edit Format View Help
use staging;
use tpch;
create table staging.dbo.lineitem
(
  l_orderkey      bigint not null,
  l_partkey       integer not null,
  l_suppkey       integer not null,
  l_linenumber    integer not null,
  l_quantity      float not null,
  l_extendedprice float not null,
  l_discount      float not null,
  l_tax           float not null,
  l_returnflag    char(1) not null,
  l_linestatus    char(1) not null,
  l_shipdate      datetime not null,
  l_commitdate    datetime not null,
  l_receiptdate   datetime not null,
  l_shipinstruct  char(25) not null,
  l_shipmode      char(10) not null,
  l_comment       varchar(132) not null
)ON psmonthly_168_PRIMARY(l_shipdate) WITH (data_compression=PAGE);

create table tpch.ft_1000.lineitem
(
  l_orderkey      bigint not null,
  l_partkey       integer not null,
  l_suppkey       integer not null,
  l_linenumber    integer not null,
  l_quantity      float not null,
  l_extendedprice float not null,
  l_discount      float not null,
  l_tax           float not null,
  l_returnflag    char(1) not null,
  l_linestatus    char(1) not null,
  l_shipdate      datetime not null,
  l_commitdate    datetime not null,
  l_receiptdate   datetime not null,
  l_shipinstruct  char(25) not null,
  l_shipmode      char(10) not null,
  l_comment       varchar(44) not null
)ON [PRIMARY] WITH (data_compression=PAGE);
```

**Note:** Page compression is used on all tables in the production database. Testing has shown that typical data compresses up to 2.5 times normal storage size.

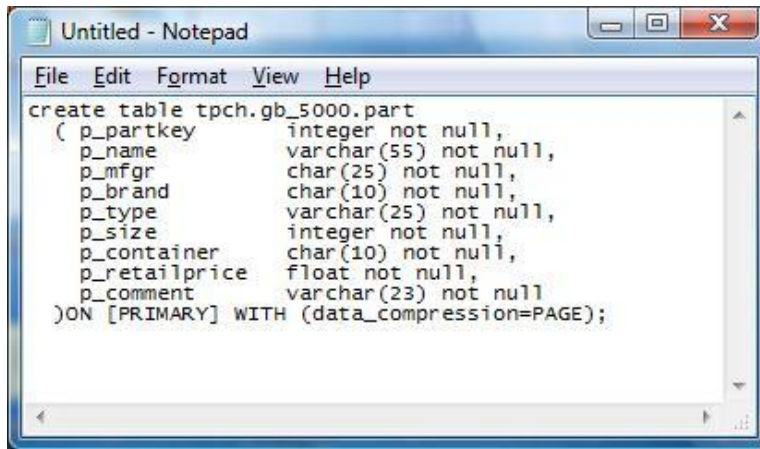
All remaining indexed tables should be created using a similar fashion as shown in the example above.

## Creating non-indexed tables

Non-indexed tables are created on the final production database only. Data is simply inserted from the load files into the final production tables using bulk insert commands detailed later in this guide.

The example shown below in figure 14 creates the TPC-H “part” table.

**Figure 14.** Non-indexed table creation

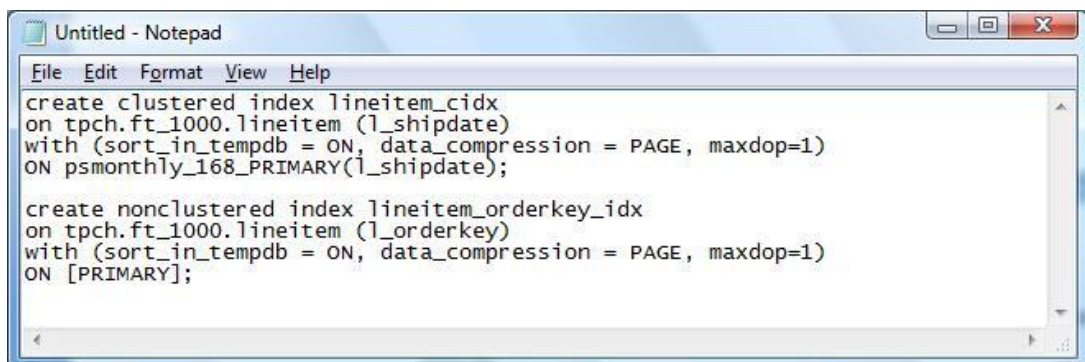


```
create table tpch.gb_5000.part
( p_partkey      integer not null,
  p_name         varchar(55) not null,
  p_mfgr        char(25) not null,
  p_brand       char(10) not null,
  p_type        varchar(25) not null,
  p_size        integer not null,
  p_container   char(10) not null,
  p_retailprice float not null,
  p_comment     varchar(23) not null
)ON [PRIMARY] WITH (data_compression=PAGE);
```

## Creating table indexes

Once the tables are created, the indexes can be created as well. For Fast Track, two indexes are created, one clustered index and one un-clustered index. In order to ensure that data is stored sequential throughout the indexing process and spillage does not occur, the MAXDOP parameter is utilized. By setting MAXDOP=1, the index process will be slower, but by specifying a single process, we can ensure that partitions are read one by one, and data is always read and written to the same partitions. The examples in figure 15 show the SQL queries used to create clustered and un-clustered indexes for one of our indexed tables, lineitem. Indexes should also be created for remaining indexed tables:

**Figure 15.** Index creation



```
create clustered index lineitem_cidx
on tpch.ft_1000.lineitem (l_shipdate)
with (sort_in_tempdb = ON, data_compression = PAGE, maxdop=1)
ON psmmonthly_168_PRIMARY(l_shipdate);

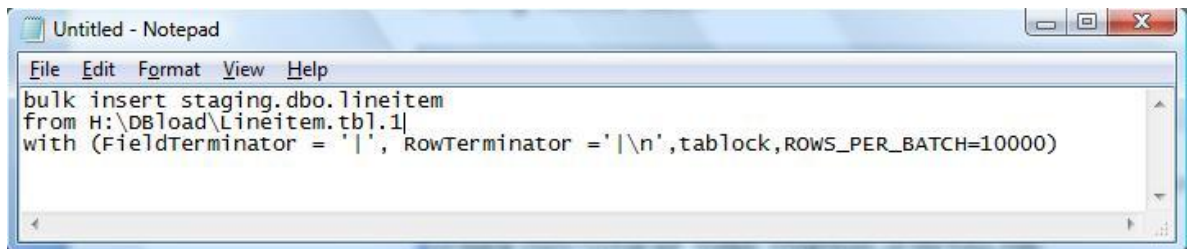
create nonclustered index lineitem_orderkey_idx
on tpch.ft_1000.lineitem (l_orderkey)
with (sort_in_tempdb = ON, data_compression = PAGE, maxdop=1)
ON [PRIMARY];
```



## Loading data to staging tables

Bulk insert operations are used to load data into the indexed tables in the staging database. Data is typically loaded from flat files that are broken into several pieces to assist in the bulk insert process using parallel loads. A general guideline is to use 1 parallel process for each CPU core available in the server. So in this example, using a 12 core server, the flat file data was broken into 12 pieces. Twelve simultaneous queries are run and the data inserted into the proper partition based on the ship date range. The SQL query example in figure 16 below one of the 12 queries that is run to load our example "lineitem" table in the staging database:

**Figure 16.** Bulk insert to staging database

A screenshot of a Notepad window titled "Untitled - Notepad". The window contains a SQL query for a bulk insert operation. The query is as follows:

```
bulk insert staging.dbo.lineitem
from H:\DBload\Lineitem.tbl.1
with (FieldTerminator = '|', RowTerminator = '\n', tablock, ROWS_PER_BATCH=10000)
```

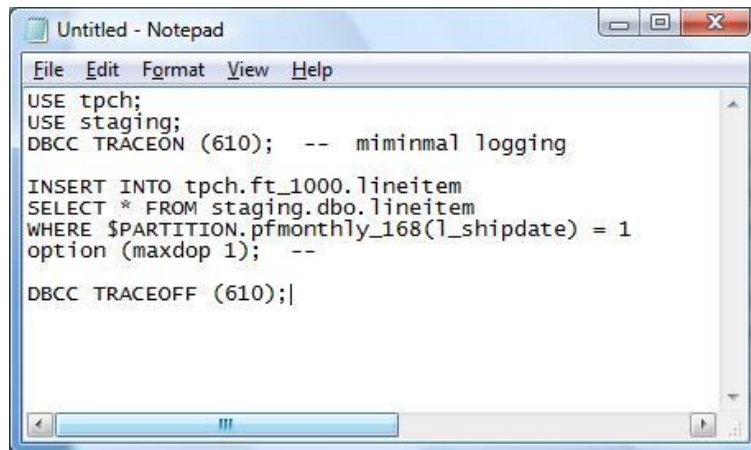
## Loading data to final database

The final step of the Fast Track load process is to insert the data into the final production tables. These processes are different for indexed and non-indexed tables. This section details each load process.

### Loading indexed tables

During heavy system utilization, using parallel load queries, the possibility exists that the batches of data will not fit into memory, which may cause multiple sort operations leading to interleaving with data being loaded in a non-sequential fashion. For this reason, the final load process utilizes the Insert/Select process using the MAXDOP=1 parameter. While this dramatically slows down the load process, it ensures that partitions are loaded one at a time, and no possibility of interleaving exists. Figure 17 shows an example of a load using partition 1. A SQL "While" loop can be constructed to loop through all of the partitions. Another example of this is shown in figure 18.

Figure 17. Insert data into production database

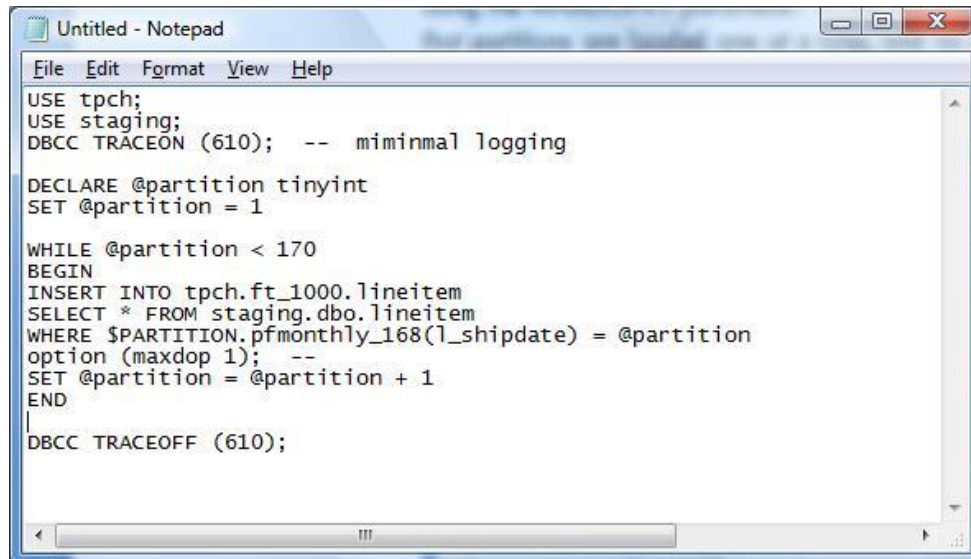


```
USE tpch;
USE staging;
DBCC TRACEON (610); -- minimal logging

INSERT INTO tpch.ft_1000.lineitem
SELECT * FROM staging.dbo.lineitem
WHERE $PARTITION.pfmonthly_168(l_shipdate) = 1
option (maxdop 1); --

DBCC TRACEOFF (610);|
```

Figure 18. Using a loop to walk through partitions



```
USE tpch;
USE staging;
DBCC TRACEON (610); -- minimal logging

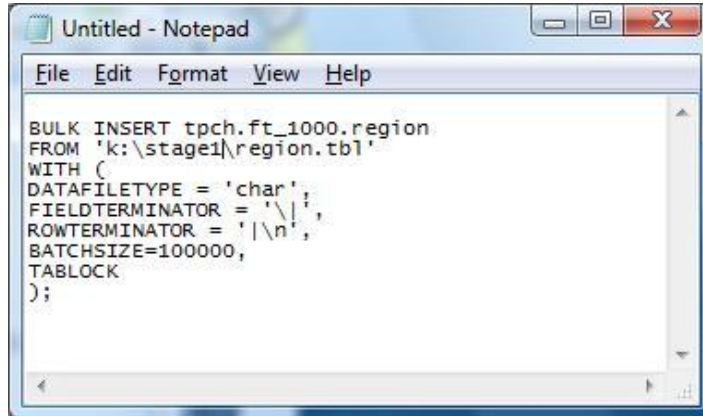
DECLARE @partition tinyint
SET @partition = 1

WHILE @partition < 170
BEGIN
INSERT INTO tpch.ft_1000.lineitem
SELECT * FROM staging.dbo.lineitem
WHERE $PARTITION.pfmonthly_168(l_shipdate) = @partition
option (maxdop 1); --
SET @partition = @partition + 1
END
|
DBCC TRACEOFF (610);
```

## Loading non-indexed tables

For the non-indexed data, a simple Bulk Insert process is used to move the data from the load files into the production tables. The example shown below in figure 19 depicts the bulk insert process using the “region” table as an example.

**Figure 19.** Non-indexed load



```
BULK INSERT tpch.ft_1000.region
FROM 'k:\stage1\region.tbl'
WITH (
DATAFILETYPE = 'char',
FIELDTERMINATOR = '\\',
ROWTERMINATOR = '\\n',
BATCHSIZE=100000,
TABLOCK
);
```

This process should be repeated for all remaining non-indexed tables.

## Summary

Building a Fast Track database requires that some extra attention be paid to detail. The performance benefits gained by using Fast Track are only realized when data is loaded sequentially into the database. The processes outlined in this document are used to ensure that data can be loaded sequentially each time data is inserted into the database.

The best practices prescribed in this document are the results of rigorous lab testing and should be used when applicable to attain the best performance during the building and loading of a Fast Track database.

# Appendix A

## Bill of Materials

Table 2 includes the bill of materials for the server, storage, switching and rack components of the reference configuration.

**Table 2.** Bill of materials

Qty	Part Number	Description
<b>Server Configuration</b>		
1	570102-001	HP ProLiant DL385 G6 2435 HPM
2	384842-B21	HP 72GB 3G SAS 10K SFF DP ENT HDD
6	492620-B21	HP 300GB 3G SAS 10K SFF DP ENT HDD
2	AJ763A	HP 82E PCIe FC HBA Dual Port (8Gb)
6	497767-B21	8 GB REG PC2-6400 2 x 4 GB Dual Rank Kit
<b>Storage Configuration</b>		
3	AJ795A	HP 2312fc DC Enh Modular Smart Array
33	AJ736A	HP MSA2 300GB 15k rpm 3.5 SAS HDD
<b>Switch Configuration</b>		
1	AM868A	HP 8/24 24-Ports / 16-Active Enabled SAN Switch
1	T3677A	HP 4/32 8-Port Upgrade LTU
16	221692-B21	2m LC-LC Cable
1	263474-B22	HP Cat 5e Cables – 8 Pack
12	AJ715A	HP 4Gb Shortwave B-series FC SFP+ 1 Pack
4	AJ716A	HP 8Gb Shortwave B-series FC SFP+ 1 Pack
1	J9279A	HP ProCurve Switch 2510G-24
<b>Rack Configuration</b>		
1	AF002A	HP Universal Rack 10642 G2 Shock Rack
1	AF002A#001	Factory Express Base Racking
1	AF062A	HP 10K G2 600W Stabilizer Kit
1	AF054A	HP 10642 G2 Sidepanel Kit

<b>Qty</b>	<b>Part Number</b>	<b>Description</b>
1	AF054A#0D1	Factory integrated
2	252663-B24	HP 16A High Voltage Modular PDU
1	252663-B24#0D1	Factory integrated
2	AF593A	HP 3.6m C19 Nema L6-20P NA/JP Pwr Crd
1	AF593A#0D1	Factory integrated
1	120672-B21	HP 9000 Series Ballast Option Kit
1	120672-B21#0D1	Factory integrated

## For more information

Implementing a SQL Server Fast Track Data Warehouse, <http://msdn.microsoft.com/en-us/library/dd459178.aspx>

HP ActiveAnswers, [www.hp.com/solutions/activeanswers](http://www.hp.com/solutions/activeanswers)

SQL Server solutions on ActiveAnswers, [www.hp.com/solutions/activeanswers/microsoft/sql](http://www.hp.com/solutions/activeanswers/microsoft/sql)

HP and Microsoft SQL Server Business Intelligence solutions, [www.hp.com/solutions/microsoft/sqlbi](http://www.hp.com/solutions/microsoft/sqlbi)

HP and Microsoft SQL Server Business Intelligence Configurations and SQL Server Fast Track Data Warehouse reference architectures, [www.hp.com/solutions/microsoft/sqlbiconfigs](http://www.hp.com/solutions/microsoft/sqlbiconfigs)

HP and Microsoft SQL Server solutions, [www.hp.com/solutions/microsoft/sql](http://www.hp.com/solutions/microsoft/sql)

HP and Microsoft, [www.hp.com/go/microsoft](http://www.hp.com/go/microsoft)

HP ProLiant DL385 G6 server, <http://www.hp.com/servers/dl385-g6>

HP StorageWorks MSA2000, [www.hp.com/go/msa](http://www.hp.com/go/msa)

HP ProCurve Networking, [www.procurve.com](http://www.procurve.com)

How to buy, [www.hp.com/buy](http://www.hp.com/buy)

To help us improve our documents, please provide feedback at [http://h20219.www2.hp.com/ActiveAnswers/us/en/solutions/technical\\_tools\\_feedback.html](http://h20219.www2.hp.com/ActiveAnswers/us/en/solutions/technical_tools_feedback.html).

## Technology for better business outcomes

© Copyright 2009 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation.

4AA3-0349ENW, November 2009



Get connected

[www.hp.com/go/getconnected](http://www.hp.com/go/getconnected)

Current HP drivers, support & security alerts  
delivered directly to your desktop

