Veritas Storage Foundation™ for DB2 Administrator's Guide

Solaris

5.0

N18527F

symantec™

Veritas Storage Foundation™ for DB2 Administrator's Guide

Documentation version 5.0

PN: N18527F

Legal Notice

The Licensed Software and Documentation are deemed to be "commercial computer software" and "commercial computer software documentation" as defined in FAR Sections 12.212 and DFARS Section 227.7202.

Symantec Corporation 20330 Stevens Creek Blvd. Cupertino, CA 95014 USA

http://www.symantec.com

Technical Support

For technical assistance, visit http://support.veritas.com and select phone or email support. Use the Knowledge Base search feature to access resources such as TechNotes, product alerts, software downloads, hardware compatibility lists, and our customer email notification service.

Contents

Chapter 1 Introducing Veritas Storage Foundation for DB2

Appendix A Veritas Storage Foundation for DB2 command line interface

Chapter 1

Introducing Veritas Storage Foundation for DB2

This chapter includes the following topics:

About Veritas Storage Foundation for DB2

There are two versions of this product:

- Veritas Storage Foundation for DB2 Standard Edition
- Veritas Storage Foundation for DB2 Enterprise Edition
  The Enterprise Edition contains everything in the Standard Edition plus Storage Mapping, Database FlashSnap, Storage Checkpoints, and Storage Rollback.

Note: Veritas Storage Foundation/High Availability (HA) for DB2 is available only with the Enterprise Edition.

Unless otherwise noted, features pertain to both the Standard and Enterprise Edition products.

Components of Veritas Storage Foundation for DB2

Veritas Storage Foundation for DB2 combines the strengths of the core technology products with database-specific enhancements to offer performance, availability, and manageability for DB2 database servers.

Veritas Storage Foundation for DB2 includes the following products:

- Veritas Volume Manager (VxVM)
  A disk management subsystem that supports disk striping, disk mirroring, and simplified disk management for improved data availability and performance.
- Veritas File System (VxFS)
  A high-performance, fast-recovery file system that is optimized for business-critical database applications and data-intensive workloads. VxFS offers online administration, letting you perform most frequently scheduled maintenance tasks (including online backup, resizing, and file system changes) without interrupting data or system availability. VxFS also provides support for large file systems (of more than 8 exabytes in a 64-bit environment) and large files (in the exabyte range in a 64-bit environment).
  Veritas File System offers the following performance-enhancing features that are of particular interest in a database environment:
  - Veritas Quick I/O is a VxFS feature that improves the throughput for DB2 databases built on VxFS file systems. Quick I/O delivers raw device performance to databases run on VxFS, providing the administrative advantages of using file systems without the performance penalties.
  - Veritas Cached Quick I/O further enhances database performance by leveraging large system memory to selectively buffer the frequently accessed data.
  - Veritas Concurrent I/O improves the performance of regular files on a VxFS file system without the need for extending namespaces and presenting the files as devices. This simplifies administrative tasks and allows relational databases (such as DB2), which do not have a sequential read/write requirement, to access files concurrently.

- A feature of the Enterprise Edition, VxFS Storage Checkpoint technology lets you create a point-in-time image of a file system. Storage Checkpoints are treated like any other VxFS file system and can be created, mounted, unmounted, and removed with VxFS and Veritas Storage Foundation administrative utilities.

- Veritas Storage Mapping
  Storage Mapping, a feature of the Enterprise Edition, lets you map DB2 tablespace containers to physical devices and display storage object I/O statistics. Both storage object I/O statistics and the storage structure can be displayed using either the `vxstorage_stats`command or the Veritas Storage Foundation GUI.

- Veritas Database FlashSnap
  Database FlashSnap, a feature of the Enterprise Edition, lets you create, resynchronize, and reverse resynchronize volume snapshots for databases. The snapshots can be used on a second host. Also, database administrators can perform these tasks without `root` privileges. Database FlashSnap tasks may be performed through the Veritas Storage Foundation GUI or the command line interface.

- Database Dynamic Storage Tiering
  Database Dynamic Storage Tiering, a feature of the Enterprise Edition, enables you to manage your data so that less-frequently used data can be moved to slower, less expensive disks, allowing frequently-accessed data to be stored on the faster disks for quicker retrieval.

- Veritas Enterprise Administrator
  Veritas Enterprise Administrator (VEA) is the infrastructure that allows you to access Veritas Storage Foundation for DB2, Veritas Volume Manager, and Veritas File System information and features through the GUI.

An optional High Availability (HA) version of Veritas Storage Foundation for DB2 Enterprise Edition, which includes Veritas Cluster Server, is available for customers who have high system-availability requirements.

How Veritas Volume Manager works

Databases require their storage media to be robust and resilient to failure. It is vital to protect against hardware and disk failures and to maximize performance using all the available hardware resources. Using a volume manager provides this necessary resilience and eases the task of management. A volume manager can help you manage hundreds of disk devices and makes spanning, striping, and mirroring easy.

Veritas Volume Manager (VxVM) builds virtual devices called volumes on top of physical disks. Volumes are accessed by a file system, a database, or other applications in the same way physical disk partitions would be accessed. Using volumes, VxVM provides the following administrative benefits for databases:

Table 1-1 Veritas Volume Manager features

Feature	Benefit
Spanning of multiple disks	Eliminates media size limitations.
Striping	Increases throughput and bandwidth.
Mirroring or RAID-5	Increases data availability.
Online relayout	Allows volume layout changes without application or database downtime. Online relayout can be used to change performance or reliability characteristics of unerlying storage.
Volume resynchronization	Ensures that all mirrors contain exactly the same data and that the data and parity in RAID-5 volumes agree.
Dirty Region Logging (DRL)	Speeds the recovery of mirrored volumes after a system crash.
Volume snapshots	Allows backup of volumes based on disk mirroring. VxVM provides full-sized and space-optimized instant snapshots, which are online and off-host point-in-time copy solutions.
FastResync	Separately licensed, optional feature that performs quick and efficient resynchronization of stale mirrors. FastResync is included with the Enterprise Edition and is also included as part of the Veritas FlashSnap option with the Standard Edition.
Disk group split and join	Separately licensed, optional feature that supports general disk group reorganization and allows you to move volume snapshots to another host for off-host backup. Disk group split and join is included with the Enterprise Edition and is also included as part of the Veritas FlashSnap option with the Standard Edition.
Hot-relocation	Automatically restores data redundancy in mirrored and RAID-5 volumes when a disk fails.

Table 1-1 Veritas Volume Manager features *(continued)*

Feature	Benefit
Dynamic multipathing (DMP)	Allows for transparent failover, load sharing, and hot plugging of physical disks.
Volume sets	Allows several volumes to be represented by a single logical mount device.
Dynamic LUN Expansion	Allows you to resize a disk after it has been initialized while preserving the existing data on the disk.
Storage Expert	Helps diagnose configuration problems with VxVM.
Cluster Volume Manager (CVM)	Separately licensed, optional feature that allows you to use VxVM in a cluster environment.
Veritas Volume Replicator (VVR)	Separately licensed, optional feature that provides data replication for disaster recovery solutions.
Free space pool management	Simplifies administration and provides flexible use of available hardware.
Online administration	Allows configuration changes without system or database down time.

For a more detailed description of VxVM and its features, refer to the *Veritas Volume Manager Administrator's Guide.*

About volumes

A volume is a virtual disk device that appears to applications, databases, and file systems like a physical disk partition without the physical limitations of a disk partition. A volume consists of one or more plexes, each holding a copy of the selected data in the volume. Due to its virtual nature, a volume is not restricted to a particular disk or a specific area of a disk. For example, a volume can span multiple disks and can be used to create a large file system.

Volumes consist of other virtual objects that can be manipulated to change the volume's configuration. Volumes and their virtual components are referred to as Volume Manager objects. You can manipulate Veritas Volume Manager objects in a variety of ways to optimize performance, provide redundancy of data, and perform backups or other administrative tasks on one or more physical disks without interrupting applications. As a result, data availability and disk subsystem throughput are improved.

You can change the configuration of a volume without causing disruption to databases or file systems that are using the volume. For example, you can mirror a volume on separate disks or move the volume to use different disk storage.

About disk groups

A disk group is a collection of disks that share a common configuration (for example, configuration objects that belong to a single database). We recommend creating one disk group for each database.

You can move a disk group and its components as a unit from one host to another host. For example, you can move volumes and file systems that belong to the same database and are created within one disk group as a unit. You must configure a given volume from disks belonging to one disk group.

In releases before Veritas Storage Foundation 4.0 for DB2, the default disk group was `rootdg`. For VxVM to function, the rootdg disk group had to exist and it had to contain at least one disk. This requirement no longer exists, and VxVM can work without any disk groups configured (although you must set up at least one disk group before you can create any volumes of other VxVM objects).

About volume layouts

A Redundant Array of Independent Disks (RAID) is a disk array in which a group of disks appears to the system as a single virtual disk or a single volume. VxVM supports several RAID implementations, as well as spanning.

The following volume layouts are available to satisfy different database configuration requirements:

- Spanning and concatenation
- Striping (RAID-0)
- Mirroring (RAID-1)
- Mirrored-Stripe Volumes (RAID-0+1)
- Striped-Mirror Volumes (RAID-1+0)
- RAID-5

Caution: Spanning or striping a volume across multiple disks increases the chance that a disk failure will result in failure of that volume. Use mirroring or RAID-5 to substantially reduce the chance of a single volume failure caused by a single disk failure.

How spanning and concatenation work

Concatenation maps data in a linear manner onto one or more subdisks in a plex. To access all of the data in a concatenated plex sequentially, data is first accessed in the first subdisk from beginning to end. Data is then accessed in the remaining subdisks sequentially from beginning to end, until the end of the last subdisk.

You can use concatenation with multiple subdisks when there is insufficient contiguous space for the plex on any one disk. This form of concatenation can be used for load balancing between disks, and for head movement optimization on a particular disk.

Concatenation using subdisks that reside on more than one VxVM disk is called spanning.

Warning: Spanning a plex across multiple disks increases the chance that a disk failure results in failure of the assigned volume. Use mirroring (RAID-1) or striping with parity (RAID-5) to reduce the risk that a single disk failure results in a volume failure.

How striping (RAID-0) works

Striping is a technique of mapping data so that the data is interleaved among multiple physical disks. Data is allocated in equal-sized units (called stripe units) that are interleaved between the disks. Each stripe unit is a set of contiguous blocks on a disk. A stripe consists of the set of stripe units at the same position across all columns. A column is a set of one or more subdisks within a striped plex.

Striping is useful if you need large amounts of data written to or read from physical disks, and performance is important. Striping is also helpful in balancing the I/O load from multi-user applications across multiple disks. By using parallel data transfer to and from multiple disks, striping significantly improves data-access performance.

When striping across multiple disks, failure of any one disk will make the entire volume unusable.

How mirroring (RAID-1) works

Mirroring is a technique of using multiple copies of the data, or mirrors, to duplicate the information contained in a volume. In the event of a physical disk failure, the mirror on the failed disk becomes unavailable, but the system continues to operate using the unaffected mirrors. For this reason, mirroring increases system reliability and availability. A volume requires at least two mirrors to provide redundancy of data. A volume can consist of up to 32 mirrors. Each of

these mirrors must contain disk space from different disks for the redundancy to be effective.

How striping plus mirroring (mirrored-stripe or RAID-0+1) works

VxVM supports the combination of mirroring above striping. The combined layout is called a mirrored-stripe layout. A mirrored-stripe layout offers the dual benefits of striping to spread data across multiple disks, while mirroring provides redundancy of data. For mirroring above striping to be effective, the striped plex and its mirrors must be allocated from separate disks.

The layout type of the data plexes in a mirror can be concatenated or striped. Even if only one is striped, the volume is still termed a mirrored-stripe volume. If they are all concatenated, the volume is termed a mirrored-concatenated volume.

How mirroring plus striping (striped-mirror, RAID-1+0 or RAID-10) works

VxVM supports the combination of striping above mirroring. This combined layout is called a striped-mirror layout and mirrors each column of the stripe. If there are multiple subdisks per column, each subdisk can be mirrored individually instead of each column. A striped-mirror volume is an example of a layered volume.

Compared to a mirrored-stripe volume, a striped-mirror volume offers the dual benefits of striping to spread data across multiple disks, while mirroring provides redundancy of data. A striped-mirror volume enhances redundancy, which makes it more tolerant of disk failure, and reduces recovery time after disk failure.

For databases that support online transaction processing (OLTP) workloads, we recommend either mirrored-stripe or striped-mirror volumes to improve database performance and reliability. For highest availability, we recommend striped-mirror volumes (RAID 1+0).

How striping with parity (RAID-5) works

RAID-5 provides data redundancy through the use of parity (a calculated value that the system uses to reconstruct data after a failure). While data is written to a RAID-5 volume, parity is also calculated by performing an exclusive OR (XOR) procedure on data. The resulting parity is then written to another part of the volume. If a portion of a RAID-5 volume fails, the data that was on that portion of the failed volume can be recreated from the remaining data and the parity.

RAID-5 offers data redundancy similar to mirroring, while requiring less disk space. RAID-5 read performance is similar to that of striping but with relatively slow write performance. RAID-5 is useful if the database workload is read-intensive

(as in many data warehousing applications). You can snapshot a RAID-5 volume and move a RAID-5 subdisk without losing redundancy.

About online relayout

As databases grow and usage patterns change, online relayout lets you change volumes to a different layout, with uninterrupted data access. Relayout is accomplished online and in place. Use online relayout to change the redundancy or performance characteristics of the storage, such as data organization (RAID levels), the number of columns for RAID-5 and striped volumes, and stripe unit size.

About volume resynchronization

When storing data redundantly, using mirrored or RAID-5 volumes, Veritas Volume Manager ensures that all copies of the data match exactly. However, if the system crashes, small amounts of the redundant data on a volume can become inconsistent or unsynchronized. For mirrored volumes, unsynchronized data can cause two reads from the same region of the volume to return different results if different mirrors are used to satisfy the read request. In the case of RAID-5 volumes, unsynchronized data can lead to parity corruption and incorrect data reconstruction.

In the event of a system crash, Veritas Volume Manager ensures that all mirrors contain exactly the same data and that the data and parity in RAID-5 volumes agree. This process is called volume resynchronization. Not all volumes require resynchronization after a system failure. VxVM notices when a volume is first written and marks it as dirty. Only volumes that are marked dirty when the system reboots require resynchronization.

The process of resynchronization can impact system and database performance. However, it does not affect the availability of the database after system reboot. You can immediately access the database after database recovery although the performance may suffer due to resynchronization. For very large volumes or for a very large number of volumes, the resynchronization process can take a long time. You can significantly reduce resynchronization time by using Dirty Region Logging (DRL) for mirrored volumes or by making sure that RAID-5 volumes have valid RAID-5 logs. However, using logs can slightly reduce the database write performance.

For most database configurations, we recommend using dirty region logs or the RAID-5 logs when mirrored or RAID-5 volumes are used. It is also advisable to evaluate the database performance requirements to determine the optimal volume configurations for the databases.

About dirty region logging

Dirty region logging (DRL), if enabled, speeds the recovery of mirrored volumes after a system crash. DRL keeps track of the regions that have changed due to I/O writes to a mirrored volume. DRL uses this information to recover only those portions of the volume that need to be recovered.

Note: If a version 20 data change object (DCO) volume is associated with a volume, a portion of the DCO volume can be used to store the DRL log. There is no need to create a separate DRL log for a volume that has a version 20 DCO volume.

For more information on DCOs and DCO volumes, see the *Veritas Volume Manager Administrator's Guide*.

About volume sets

Volume sets are an enhancement to VxVM that allow several volumes to be represented by a single logical mount device. All I/O from and to the underlying volumes is directed via the I/O interfaces of the volume set. The volume set feature supports the multi-device enhancement to Veritas File System (VxFS). This feature allows file systems to make best use of the different performance and availability characteristics of the underlying volumes. For example, file system metadata could be stored on volumes with higher redundancy, and user data on volumes with better performance.

About volume snapshots

A volume snapshot is a point-in-time image of a volume. Veritas Volume Manager provides three volume snapshot features based on disk mirroring:

- Full-sized instant snapshots
- Space-optimized instant snapshots
- Emulation of third-mirror snapshots

About Veritas FastResync

Veritas FastResync (previously called Fast Mirror Resynchronization or FMR) is included with the Enterprise Edition. It is also included as part of the Veritas FlashSnap option with the Standard Edition.

Veritas FastResync performs quick and efficient resynchronization of stale mirrors (mirrors that are not synchronized). This increases the efficiency of the VxVM snapshot mechanism, and improves the performance of operations such as backup

and decision support. Typically, these operations require that the volume is quiescent, and that they are not impeded by updates to the volume by other activities on the system. To achieve these goals, the snapshot mechanism in VxVM creates an exact copy of a primary volume at an instant in time. After a snapshot is taken, it can be accessed independently of the volume from which it was taken.

Veritas Storage Foundation for DB2 Enterprise Edition includes a feature called Database FlashSnap, which takes advantage of the FastResync and disk group split and join features. Database FlashSnap provides a quicker and easier way for database administrators to use volume snapshots.

See "How Veritas Volume Manager works" on page 17.

See "How Veritas Database FlashSnap works" on page 35.

How non-persistent FastResync works

Non-persistent FastResync allocates its change maps in memory. If non-persistent FastResync is enabled, a separate FastResync map is kept for the original volume and for each snapshot volume. Unlike a dirty region log (DRL), these maps do not reside on disk nor in persistent store. The advantage is that updates to the FastResync map have little impact on I/O performance, as no disk updates need to be performed. However, if a system is rebooted, the information in the map is lost, so a full resynchronization is required when performing a snapback operation. This limitation can be overcome for volumes in cluster-shareable disk groups, provided that at least one of the nodes in the cluster remains running to preserve the FastResync map in its memory.

How persistent FastResync works

Non-persistent FastResync has been augmented by the introduction of persistent FastResync. Unlike non-persistent FastResync, Persistent FastResync keeps the FastResync maps on disk so that they can survive system reboots and system crashes. When the disk groups are rejoined, this allows the snapshot plexes to be quickly resynchronized. This ability is not supported by non-persistent FastResync.

If persistent FastResync is enabled on a volume or on a snapshot volume, a DCO and a DCO log volume are associated with the volume.

The DCO object is used not only to manage FastResync maps, but also to manage DRL recovery maps and special maps called copy maps that allow instant snapshot operations to be resume following a system crash.

Persistent FastResync can also track the association between volumes and their snapshot volumes after they are moved into different disk groups. When the disk groups are rejoined, this allows the snapshot plexes to be quickly resynchronized. This ability is not supported by non-persistent FastResync.

About disk group split and join

Disk group split and join is included with the Enterprise Edition. It is also included as part of the Veritas FlashSnap option with the Standard Edition.

VxVM provides a disk group content reorganization feature that supports general disk group reorganization and allows you to move volume snapshots to another host for off-host backup. Additional options to the `vxdg` command enable you to take advantage of the ability to remove all VxVM objects from an imported disk group and move them to a newly created target disk group (split), and to remove all VxVM objects from an imported disk group and move them to an imported target disk group (join). The move operation enables you to move a self-contained set of VxVM objects between the imported disk groups.

About hot-relocation

In addition to providing volume layouts that help improve database performance and availability, VxVM offers features that you can use to further improve system availability in the event of a disk failure. Hot-relocation is a feature that allows a system to react automatically to I/O failures on mirrored or RAID-5 volumes and restore redundancy and access to those volumes.

VxVM detects I/O failures on volumes and relocates the affected portions to disks designated as spare disks or free space within the disk group. VxVM then reconstructs the volumes that existed before the failure and makes them redundant and accessible again.

The hot-relocation feature is enabled by default and is recommended for most database configurations. After hot-relocation occurs, we recommend verifying the volume configuration for any possible performance impact. It is also a good idea to designate additional disks as spares to augment the spare pool.

While a disk is designated as a spare, you cannot use the space on that disk for the creation of VxVM objects within its disk group. VxVM also lets you free a spare disk for general use by removing it from the pool of hot-relocation disks.

About DMP-supported disk arrays

VxVM provides administrative utilities and driver support for disk arrays that can take advantage of its Dynamic Multipathing (DMP) feature. Some disk arrays provide multiple ports to access their disk devices. These ports, coupled with the host bus adaptor (HBA) controller and any data bus or I/O processor local to the array, make up multiple hardware paths to access the disk devices. Such disk arrays are called multipathed disk arrays. This type of disk array can be connected to host systems in many different configurations, (such as multiple ports connected to different controllers on a single host, chaining of the ports through a single

controller on a host, or ports connected to different hosts simultaneously). DMP is available for multiported disk arrays from various vendors and provides improved reliability and performance by using path failover and load balancing.

See the *Veritas Volume Manager Administrator's Guide*.

See the *Veritas Volume Manager Hardware Notes*.

About dynamic LUN expansion

Dynamic LUN expansion allows you to resize a disk after it has been initialized while preserving the existing data on the disk.

See the *Veritas Volume Manager Administrator's Guide*.

About Storage Expert

Storage Expert consists of a set of simple commands that collect VxVM configuration data and compare it with “best practice.” Storage Expert then produces a summary report that shows which objects do not meet these criteria and makes recommendations for VxVM configuration improvements.

These user-configurable tools help you as an administrator to verify and validate systems and non-optimal configurations in both small and large VxVM installations.

Storage Expert components include a set of rule scripts and a rules engine. The rules engine runs the scripts and produces ASCII output, which is organized and archived by Storage Expert's report generator. This output contains information about areas of VxVM configuration that do not meet the set criteria. By default, output is sent to the screen, but you can redirect it to a file using standard UNIX redirection.

See the *Veritas Volume Manager Administrator's Guide*.

About cluster functionality (optional)

VxVM includes an optional, separately licensable clustering feature, known as Cluster Volume Manager, that enables VxVM to be used in a cluster environment. With the clustering option, VxVM supports up to 16 nodes per cluster.

See the *Veritas Volume Manager Administrator's Guide*.

About Veritas Volume Replicator (optional)

Veritas Volume Replicator (VVR) is an optional, separately licensable feature of VxVM. VVR is a data replication tool designed to maintain a consistent copy of

application data at a remote site. It is built to contribute to an effective disaster recovery plan. If the data center is destroyed, the application data is immediately available at the remote site, and the application can be restarted at the remote site.

VVR works as a fully integrated component of VxVM. VVR benefits from the robustness, ease of use, and high performance of VxVM and, at the same time, adds replication capability to VxVM. VVR can use existing VxVM configurations with some restrictions. Any application, even with existing data, can be configured to use VVR transparently.

See the Veritas Volume Replicator documentation.

How Veritas File System works

Veritas File System (referred to as VxFS) is an extent-based, intent logging file system intended for use in UNIX environments that deal with large volumes of data and that require high file system performance, availability, and manageability. VxFS also provides enhancements that make file systems more viable in database environments.

The following sections provide a brief overview of VxFS concepts and features that are relevant to database administration.

See the *Veritas File System Administrator's Guide*.

About Veritas Quick I/O

Databases can run on either file systems or raw devices. Database administrators often create their databases on file systems because it makes common administrative tasks (such as moving, copying, and backing up) easier. However, running databases on most file systems significantly reduces database performance.

When performance is an issue, database administrators create their databases on raw devices. VxFS with Quick I/O presents regular, preallocated files as raw character devices to the application. Using Quick I/O, you can enjoy the management advantages of databases created on file systems and achieve the same performance as databases created on raw devices.

See “About Quick I/O” on page 71.

About Veritas Cached Quick I/O

Cached Quick I/O allows databases to make more efficient use of large system memory while still maintaining the performance benefits of Quick I/O. Cached

Quick I/O provides an efficient, selective buffering mechanism to back asynchronous I/O. Using Cached Quick I/O, you can enjoy all the benefits of Quick I/O and achieve even better performance.

Cached Quick I/O is first enabled for the file system and then enabled on a per file basis.

See "About Cached Quick I/O" on page 93.

About Veritas Concurrent I/O

Veritas Concurrent I/O improves the performance of regular files on a VxFS file system without the need for extending namespaces and presenting the files as devices. This simplifies administrative tasks and allows databases, which do not have a sequential read/write requirement, to access files concurrently.

Veritas Concurrent I/O allows for concurrency between a single writer and multiple readers or between multiple writers. It minimizes serialization for extending writes and sends I/O requests directly to file systems.

See "About Concurrent I/O" on page 107.

About extent-based allocation

The UFS file system supplied with Solaris uses block-based allocation schemes that provide good random access to files and acceptable latency on small files. For larger files, such as database files, this block-based architecture limits throughput. This limitation makes the UFS file system a less than optimal choice for database environments.

When storage is allocated to a file on a VxFS file system, it is grouped in extents, as opposed to being allocated a block at a time as with the UFS file system.

By allocating disk space to files in extents, disk I/O to and from a file can be done in units of multiple blocks. This type of I/O can occur if storage is allocated in units of consecutive blocks. For sequential I/O, multiple block operations are considerably faster than block-at-a-time operations. Almost all disk drives accept I/O operations of multiple blocks.

The VxFS file system allocates disk space to files in groups of one or more extents. VxFS also allows applications to control some aspects of the extent allocation for a given file. Extent attributes are the extent allocation policies associated with a file.

See "Preallocating space for Quick I/O files using the setext command" on page 77.

About fast file system and database recovery

Veritas File System begins recovery procedures within seconds after a system failure by using a tracking feature called intent logging. This feature records pending changes to the file system structure in a circular intent log. The intent log recovery feature is not readily apparent to users or a system administrator except during a system failure. During system failure recovery, the VxFS `fsck` utility performs an intent log replay, which scans the intent log and nullifies or completes file system operations that were active when the system failed. The file system can then be mounted without completing a full structural check of the entire file system. Replaying the intent log may not completely recover the damaged file system structure if there was a disk hardware failure; hardware problems may require a complete system check using the `fsck` utility provided with Veritas File System.

About online system administration

The VxFS file system provides online system administration utilities to help resolve certain problems that impact database performance. You can defragment and resize a VxFS file system while it remains online and accessible to users.

How the defragmentation utility works

Free resources are originally aligned in the most efficient order possible and are allocated to files in a way that is considered by the system to provide optimal performance. When a file system is active for extended periods of time, new files are created, old files are removed, and existing files grow and shrink. Over time, the original ordering of free resources is lost and the file system tends to spread across the disk, leaving unused gaps or fragments between areas that are in use. This process, known as fragmentation, leads to degraded performance because the file system has fewer choices when selecting an extent (a group of contiguous data blocks) to assign to a file. You should analyze the degree of fragmentation before creating new database files.

VxFS provides the online administration utility `fsadm` to resolve fragmentation problems. The utility can be run on demand and should be scheduled regularly as a `cron` job.

How the resizing utility works

Changes in database size can result in file systems that are too large or too small for the current database. Without special utilities, expanding or shrinking a file system becomes a a matter of stopping applications, offloading the contents of

the file system, rebuilding the file system to a new size, and then restoring the data. Data is unavailable to users while these administrative tasks are performed.

The VxFS file system utility `fsadm` provides a mechanism to resize file systems without unmounting them or interrupting users' productivity. Because the VxFS file system can only be mounted on one device, expanding a file system means that the underlying device must also be expandable while the file system is mounted. Working with VxVM, VxFS provides online expansion capability.

About cross-platform data sharing

Veritas Cross-Platform Data Sharing allows data to be serially shared among heterogeneous systems where each system has direct access to the physical devices that hold the data. This feature can be used only in conjunction with Veritas Volume Manager. Shared or parallel access is possible for read-only data.

See the *Veritas Storage Foundation Cross-Platform Data Sharing Administrator's Guide*.

Support for multi-volume file systems

The multi-volume file system (MVS) feature allows several volumes to be represented by a single logical object. All I/O to and from an underlying logical volume is directed by way of volume sets. A volume set is a container for multiple different volumes. This feature can be used only in conjunction with Veritas Volume Manager.

About Quality of Storage Service (optional)

The Quality of Storage Service (QoSS) feature is included with the Enterprise Edition.

The QoSS option is built on the multi-volume support technology introduced in this release. Using QoSS, you can map more than one device to a single file system. You can then configure policies that automatically relocate files from one device to another, or relocate files by running file relocation commands. Having multiple devices lets you determine where files are located, which can improve performance for applications that access specific types of files and reduce storage-related costs.

Database Dynamic Storage Tiering is built on top of QoSS and automates the migration process for DB2 database objects.

Support for large file systems and large files (optional)

Support for large file systems is included with the Enterprise Edition.

In conjunction with VxVM, VxFS can support file systems up to 8 exabytes in size.

You have the option of creating a file system using:

- Version 4 disk layout, which supports file systems up to one terabyte. The Version 4 disk layout encompasses all file system structural information in files, rather than at fixed locations on disk, allowing for greater scalability.
- Version 5, which supports file systems up to 32 terabytes. Files can be a maximum of two terabytes. File systems larger than one terabyte must be created on a Veritas Volume Manager volume.
- Version 6, which supports file systems up to 8 exabytes. The Version 6 disk layout enables features such as multi-device support, Cross-Platform Data Sharing, named data streams, file change log. File systems created on VxFS 4.1 will by default use the Version 6 disk layout. An online conversion utility, `vxupgrade`, is provided to upgrade existing disk layouts to Version 6 on mounted file systems.

For large database configurations, this eliminates the need to use multiple file systems because of the size limitations of the underlying physical devices.

Changes implemented with the VxFS Version 4 disk layout have greatly expanded file system scalability, including support for large files.

You can create or mount file systems with or without large files by specifying either the `largefiles` or `nolargefiles` option in `mkfs` or `mount` commands.

See "Creating a VxFS file system " on page 51.

About Storage Checkpoints and Storage Rollback

The Storage Checkpoint and Storage Rollback features are included with the Enterprise Edition. With the Standard Edition, they can be purchased as part of the Veritas FlashSnap option.

Veritas File System provides a Storage Checkpoint facility that allows you to create a persistent, point-in-time image of all user files in a file system—the Storage Checkpoint remains even after the file system is unmounted or the system is rebooted. Storage Checkpoints present a view of a file system at a point in time, and subsequently identifies and maintains copies of the original file system blocks. Instead of using a disk-based mirroring method, Storage Checkpoints save disk space and significantly reduce I/O overhead by using the free space pool available to a file system.

The time required to create a Storage Checkpoint is typically only a couple of seconds. After a Storage Checkpoint is created, a consistent database backup image is made and the database can then resume its normal operation.

The Storage Rollback facility can then be used for rolling back the file system image to the point in time when the Storage Checkpoints were taken. In addition, Storage Checkpoints also keep track of the block change information that enables incremental database backup at the block level.

Storage Checkpoints are writable, and can be created, mounted, and removed. Performance enhancements in maintaining Data Storage Checkpoints (Storage Checkpoints that are complete images of the file system) makes using the Storage Rollback feature easier and more efficient, therefore more viable for backing up large databases.

Multi-Volume File System (MVS) Storage Checkpoint creation allows database backups without having to shut down the database.

MVSs provide the ability to create and administer Storage Checkpoint allocation policies. Storage Checkpoint allocation policies specify a list of volumes and the order in which Storage Checkpoint data is allocated to them. These allocation policies can be used to control where a Storage Checkpoint is created, allowing for separating Storage Checkpoint metadata and data onto different volumes. They can also be used to isolate data allocated to a Storage Checkpoint from the primary file system, which can help prevent the Storage Checkpoint from fragmenting space in the primary file system.

See "About Storage Checkpoints and Storage Rollback" on page 129.

About restoring file systems using Storage Checkpoints

Storage Checkpoints can be used by backup and restore applications to restore either individual files or an entire file system. Restoring from Storage Checkpoints can recover data from incorrectly modified files, but typically cannot be used to recover from hardware damage or other file system integrity problems. File restoration can be done using the `fsckpt_restore`(1M) command.

See the *Veritas File System Administrator's Guide*.

About quotas

VxFS supports quotas, which allocate per-user and per-group quotas and limit the use of two principal resources: files and data blocks. You can assign quotas for each of these resources.

Each quota consists of two limits for each resource:

- The hard limit represents an absolute limit on data blocks or files. A user can never exceed the hard limit under any circumstances.

- The soft limit is lower than the hard limit and can be exceeded for a limited amount of time. This allows users to temporarily exceed limits as long as they fall under those limits before the allotted time expires.

You can use quotas to limit the amount of file system space used by Storage Checkpoints.

With Veritas Storage Foundation for DB2, you can enable, disable, set, and display quota values for a single file system, for multiple file systems, or for all file systems in a database using the `db2ed_ckptquota`command.

See "Administering Storage Checkpoint quotas using db2ed_ckptquota" on page 293.

See the *Veritas File System Administrator's Guide*.

About cluster functionality (optional)

File system clustering is an optional, separately licensed feature of VxFS, where one system is configured as a primary server for the file system, and the other members of a cluster are configured as secondaries. All servers access shared disks for file data operations. If the primary server fails, one of the secondary servers takes over the file system operations.

See the *Veritas File System Administrator's Guide*.

How Veritas Storage Mapping works

Veritas Storage Mapping is a feature included with Veritas Storage Foundation for DB2 Enterprise Edition.

Veritas has defined and implemented a library called Veritas Mapping Service (VxMS) that provides a mapping interface to VxFS file systems, VxVM volumes, and physical disks. With Veritas Storage Foundation for DB2 , you can take full advantage of this feature to map datafiles or containers, depending on your database, to physical devices and display storage object I/O statistics. With the `vxstorage_stats` command, you can view the complete I/O topology mapping of datafiles or containers through intermediate layers like logical volumes down to actual physical devices. You can also use `vxstorage_stats` to view statistics for VxFS file systems, VxVM volumes, and physical devices. This information can be used to determine the exact location of a data block on a disk and to help identify hot spots.

This command can help you avoid I/O contention. For example, you can use the information to avoid backing up two tablespaces that share the same physical disk.

Both storage object statistics and the storage structure are displayed in the Veritas Storage Foundation for DB2 GUI.

See "About Storage Mapping " on page 111.

How Veritas Database FlashSnap works

Veritas Database FlashSnap is a feature included with Veritas Storage Foundation Enterprise Edition. It is also a separately licensed option available with Veritas Storage Foundation Standard Edition.

Veritas Database FlashSnap offers a flexible and efficient means of managing business-critical data. Database FlashSnap lets you capture an online image of an actively changing database at a given instant, called a point-in-time copy. You can perform system backup, upgrade, or perform other maintenance tasks on point-in-time copies while providing continuous availability of your critical data. If required, you can offload processing of the point-in-time copies onto another host to avoid contention for system resources on your production server.

Database FlashSnap takes advantage of the Persistent FastResync and Disk Group Content Reorganization features of VxVM. Database FlashSnap also streamlines database operations. Once configured, the database administrator can create snapshots, resynchronize data, and reverse resynchronize data without involving the system administrator.

Veritas Storage Foundation for DB2 provides three commands that can be executed by the database administrator and do not require root privileges:

- `db2ed_vmchecksnap`
- `db2ed_vmsnap`
- `db2ed_vmclonedb`

These commands let database administrators take advantage of the VxVM snapshot functionality without having to deal with storage operations in day-to-day database uses. To use Database FlashSnap, you must configure the volumes used by the database.

See "Hosts and storage for Database FlashSnap " on page 192.

How Database Dynamic Storage Tiering works

Today, more and more data needs to be retained. Eventually, some of the data is no longer needed as frequently, but it still takes up a large amount of disk space. Database Dynamic Storage Tiering matches data storage with the data's usage requirements so that data is relocated based on requirements determined by the

database administrator (DBA). The feature enables you to manage your data so that less-frequently used data can be moved to slower, less expensive disks, allowing frequently-accessed data to be stored on the faster disks for quicker retrieval.

The DBA can create a file allocation policy based on filename extension before new files are created, which will create the datafiles on the appropriate tier during database creation.

The DBA can also create a file relocation policy for database files, which would relocate files based on how frequently a file is used.

See "About Database Dynamic Storage Tiering" on page 157.

About the vxdbd daemon

The `vxdbd` daemon handles communication to and from the Veritas Storage Foundation for DB2 software. By default, `vxdbd` communicates with Veritas Storage Foundation for DB2 over port number 3233. If there are conflicts with this port or other port-related problems, you can change the port by changing the `VXDBD_SOCKET` setting located in the `/etc/vx/vxdbed/admin.properties` file.

Normally the `vxdbd` daemon starts automatically when the host boots up. However, if the daemon reports errors or if the daemon process dies, you may have to manually start or stop it. There is also a `status` command that reports the current state of the daemon to confirm that it is currently running.

Only the root user can stop `vxdbd`. Any user can start `vxdbd` or display its status.

Note: You must have a valid HOME directory for `vxdbd` to work correctly with several Veritas Storage Foundation for DB2 features. If you receive the following error message, but you confirm that vxdbd is running (using `ps -ef | grep vxdbd` rather than `vxdbdctrl status`), you may not have a valid HOME directory or it may not be available to `vxdbd`:

```
VXDBA_PRODUCT exec_remote ERROR V-81-7700 Can not connect to the vxdbd.
```

To see the status of the vxdbd daemon

- Use the `vxdbdctrl status` command:

  ```
  /opt/VRTSdbcom/bin/vxdbdctrl status
  ```

  If the daemon is running you see the following output:

  ```
  vxdbd is alive
  ```

To start the vxdbd daemon

- Use the `vxdbdctrl start` command:

  ```
  /opt/VRTSdbcom/bin/vxdbdctrl start
  ```

To stop the vxdbd daemon

- As root, use the `vxdbdctrl stop` command:

  ```
  /opt/VRTSdbcom/bin/vxdbdctrl stop
  ```

To change the communications port used by the vxdbd daemon

1. As the root user, stop the `vxdbd` daemon:

   ```
   /opt/VRTSdbcom/bin/vxdbdctrl stop
   ```

2. In the `/etc/vx/vxdbed/admin.properties` file, change the value of the VXDBD_SOCKET variable to a new port number:

   ```
   VXDBD_SOCKET=3233
   ```

3. Restart the vxdbd daemon:

   ```
   /opt/VRTSdbcom/bin/vxdbdctrl start
   ```

4. If the system is part of a multi-host configuration, change the port on all hosts involved by repeating this procedure on each host.
5. Restart the DBED agent `vxpal`.

   See the section "Starting the DBED agent" in the *Veritas Storage Foundation for DB2 Graphical User Interface Guide.*

About Veritas Storage Foundation for DB2 graphical user interface

An alternative to the command line interface, the Veritas Storage Foundation for DB2 GUI allows you to manage the storage used by databases.

In this release, Veritas Storage Foundation for DB2 offers both a Java-based GUI and a Web-based GUI. The Web-based GUI does not contain the Database Dynamic Storage Tiering functionality or the scheduler functionality.

You can use the GUI to:

- Display database, tablespace, container, and file system information and manage the database state.
- Manage volume and file movement with Database Dynamic Storage Tiering. With this feature, you can also get reports on file usage. (Java-based GUI only.)
- Create, display, mount, unmount, and remove Storage Checkpoints.
- Automate tasks, such as creating or cloning a database using a Storage Checkpoint, with the scheduler. (Java-based GUI only.)
- Roll back databases to Storage Checkpoints.
- Collect and display statistics on file system and DB2 space usage.
- Collect and display storage object I/O statistics and the storage structure.
- Monitor file system and DB2 tablespace and container space usage and automatically grow the file system as needed.
- Examine volumes used by the file systems and overall system configuration.
- Start or stop instances or duplicate databases. You can duplicate a database using Storage Checkpoints or Database FlashSnap.
- Create or shut down a clone database.
- Create, modify, validate, or remove snapplans.
- Create, resynchronize, or reverse resynchronize a snapshot database.

For more information, see the appropriate online help and the *Veritas Storage Foundation for DB2 Graphic User Interface Guide*.

About Veritas NetBackup (optional)

Veritas NetBackup provides backup, archive, and restore capabilities for database files and directories contained on client systems in a client-server network. NetBackup server software resides on platforms that manage physical backup

storage devices. The NetBackup server provides robotic control, media management, error handling, scheduling, and a repository of all client backup images.

Administrators can set up schedules for automatic, unattended full and incremental backups. These backups are managed entirely by the NetBackup server. The administrator can also manually back up clients. Client users can perform backups, archives, and restores from their client system, and once started, these operations also run under the control of the NetBackup server.

Veritas NetBackup can be configured for DB2 in an Extended Edition (EE) or Extended-Enterprise Edition (EEE) environment. For detailed information and instructions on configuring DB2 for EEE, see "Configuring for a DB2 EEE (DPF) Environment" in the *Veritas NetBackup for DB2 System Administrator's Guide for UNIX*.

Veritas NetBackup, while not a shipped component of Veritas Storage Foundation for DB2, can be purchased separately.

About Veritas Storage Foundation/High Availability for DB2 (optional)

Veritas Storage Foundation/High Availability (HA) (VCS) lets database administrators integrate multiple servers into high availability database configurations that can significantly reduce the down time of DB2 databases caused by a system hardware or software failure.

In addition to the Veritas products included in the base Veritas Storage Foundation for DB2, Veritas Storage Foundation/HA for DB2 incorporates the following products:

- Veritas Cluster Server ™ (VCS) for DB2
- Veritas Cluster Server™ (VCS) Enterprise Agent for DB2

Note: Veritas Storage Foundation/HA (VCS) for DB2 is available only for the Enterprise Edition.

Chapter 2

Setting up databases

This chapter includes the following topics:

Tasks for setting up new databases

If you are using Veritas Storage Foundation for DB2 to set up a new database, complete these tasks in the order listed below:

Determine the number and sizes of file systems you need for the database you want to create.	See the *Veritas File System Administrator's Guide*.
Create volumes to meet your file system needs. You can use disk mirroring as a safeguard against disk failures and striping for better performance.	
If you plan to create volume snapshots for the database and use them on either the same host or a secondary one, ensure that your volume layout is consistent with Database FlashSnap requirements.	
Create the VxFS file systems you need on the volumes.	See "File system creation guidelines" on page 50. See "Creating a VxFS file system " on page 51.
Install and configure your database. You must create Quick I/O files before creating the tablespaces.	See "About Quick I/O" on page 71.
If you would like the ability to view detailed storage stack topology information to ensure your storage stack configuration is optimized for the database, configure and use Storage Mapping.	See "About Storage Mapping " on page 111.
If you want to use Database FlashSnap for off-host processing after converting your database files to use Quick I/O or ODM and your volume layout is inconsistent with Database FlashSnap requirements, you will need to "relayout" your volume manager configuration after your database files have been converted.	See the *Veritas Volume Manager Administrator's Guide*.
If you are not currently running on VxVM and VxFS, make sure Veritas Storage Foundation for DB2 is installed and convert your existing database configuration.	See the *Veritas Storage Foundation for DB2 Installation Guide*.

For backup and recovery on the same host, you can use the Storage Checkpoint facility to create file system snapshots of the database. A Storage Checkpoint creates an exact image of a database instantly and provides a consistent image of the database from the point in time the Storage Checkpoint was created.

See "About Storage Checkpoints and Storage Rollback" on page 129.

About setting up a disk group

Before creating file systems for a database, set up a disk group for each database.

A disk group lets you group disks, volumes, file systems, and files that are relevant to a single database into a logical collection for easy administration. Because you can move a disk group and its components as a unit from one machine to another, you can move an entire database when all the configuration objects of the database are in one disk group. This capability is useful in a failover situation.

Disk group configuration guidelines

Follow these guidelines when setting up disk groups:

- Only disks that are online and do not already belong to a disk group can be used to create a new disk group.
- Create one disk group for each database.
- The disk group name must be unique. Name each disk group using the DB2 database name specified by the environment variable `$DB2DATABASE` and a `dg` suffix. The `dg` suffix helps identify the object as a disk group. Also, each disk name must be unique within the disk group.

  Note: Users should not share a disk group between different DB2 instances. Although it is not recommended, sharing a disk group among all databases in the same instance may make sense if the instance contains several small databases. In this case, name the disk group using the DB2 instance name specified by the environment variable `$DB2INSTANCE` and a `dg` suffix.

- Never create container files using file systems or volumes that are not in the same disk group.

In earlier releases of Veritas Volume Manager, a system installed with VxVM was configured with a default disk group, `rootdg`, that had to contain at least one disk.

VxVM can now function without any disk group having been configured. Only when the first disk is placed under VxVM control must a disk group be configured.

Note: Most VxVM commands require superuser or equivalent privileges.

See "About tuning VxVM " on page 251.

Creating a disk group

You can use the `vxdg` command to create a new disk group. A disk group must contain at least one disk at the time it is created. You also have the option to create a shared disk group for use in a cluster environment.

Disks must be placed in disk groups before they can be used by VxVM. You can create disk groups to organize your disks into logical sets of disks.

Before creating a disk group, make sure the following conditions have been met:

Prerequisites

- Only disks that are online and do not belong to a disk group can be used to create a disk group.
- The disk group name must be unique in the host or cluster.
- Creating a disk group requires at least one disk.

Usage notes

- Veritas Storage Foundation for DB2 only supports single disk groups.
- New disks must be placed under VxVM control and then added to a dynamic disk group before they can be used for volumes.
- When you place a disk under VxVM control, the disk is either encapsulated or initialized. Encapsulation preserves any existing data on the disk in volumes. Initialization destroys any existing data on the disk.
- If you place the root disk under VxVM control, you must encapsulate the disk. If you want to create an alternate boot disk, you can mirror the encapsulated root disk.
- For information on the `vxdg` command, see the `vxdg`(1M) manual page.

To create a new disk group

- Use the `vxdg` command as follows:

```
# /opt/VRTS/bin/vxdg init disk_group [disk_name=disk_device]
```

The following is an example of creating a disk group using the `vxdg` command:

To create a disk group named PRODdg on a raw disk partition c1t1d0s2, where the disk name PRODdg01 references the disk within the disk group:

```
# /opt/VRTS/bin/vxdg init PRODdg PRODdg01=c1t1d0s2
```

Adding disks to a disk group

When a disk group is first created, it can contain only a single disk. You may need to add more disks to the disk group. Before adding disks, make sure the following conditions have been met:

Usage notes

- When you place a disk under VxVM control, the disk is either encapsulated or initialized. Encapsulation preserves any existing data on the disk in volumes. Initialization destroys any existing data on the disk.
- If you place the boot disk under VxVM control, you must encapsulate it. If you want to create an alternate boot disk, you can mirror the encapsulated boot disk.
- Boot disk encapsulation requires a system reboot.
- Disks cannot be added to deported disk groups.
- Disks must be under VxVM control and in a disk group before they can be used to create volumes.
- Disks must be online before they can be added to a disk group.
- Disks that already belong to a disk group cannot be added to another disk group.

To add disks to a disk group

- Use the vxdg command as follows:

```
# /opt/VRTS/bin/vxdg -g disk_group adddisk \
[disk_name=disk_device]
```

The following is an example of adding disks to a disk group using the vxdg command:

To add disks named PRODdg02, PRODdg03, and PRODdg04 to the disk group PRODdg:

```
# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg02=c1t2d0s2

# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg03=c1t3d0s2

# /opt/VRTS/bin/vxdg -g PRODdg adddisk PRODdg04=c1t4d0s2
```

About selecting a volume layout

Veritas Volume Manager offers a variety of layouts that allow you to configure your database to meet performance and availability requirements. The proper selection of volume layouts provides optimal performance for the database workload.

An important factor in database performance is the tablespace placement on the disks.

Disk I/O is one of the most important determining factors of your database's performance. Having a balanced I/O load usually means optimal performance. Designing a disk layout for the database objects to achieve balanced I/O is a crucial step in configuring a database.

When deciding where to place tablespaces, it is often difficult to anticipate future usage patterns. VxVM provides flexibility in configuring storage for the initial database set up and for continual database performance improvement as needs change. VxVM can split volumes across multiple drives to provide a finer level of granularity in data placement. By using striped volumes, I/O can be balanced across multiple disk drives. For most databases, ensuring that different containers or tablespaces, depending on which database you are using, are distributed across the available disks may be sufficient.

Striping also helps sequential table scan performance. When a table is striped across multiple physical devices, a high transfer bandwidth can be achieved by closely matching several DB2 parameters to ensure that database extents match the stripe size for the device. Another very important consideration when using the DB2 database, which by default performs striping at the tablespace container level, is setting the DB2_STRIPED_CONTAINERS variable.

See “About tuning VxVM ” on page 251.

How to choose appropriate stripe unit sizes

When creating a striped volume, you need to decide the number of columns to form a striped volume and the stripe unit size. You also need to decide how to stripe the volume. You may stripe a volume across multiple disk drives on the same controller or across multiple disks on multiple controllers. By striping across multiple controllers, disk I/O can be balanced across multiple I/O channels. The decision is based on the disk and controller bandwidth and the database workload. In general, for most OLTP databases, use the default stripe unit size of 64 K or smaller for striped volumes and 16 K for RAID-5 volumes.

How to choose between mirroring and RAID-5

VxVM provides two volume configuration strategies for data redundancy: mirroring and RAID-5. Both strategies allow continuous access to data in the event of disk failure. For most database configurations, we recommend using mirrored, striped volumes. If hardware cost is a significant concern, but having higher data availability is still important, use RAID-5 volumes.

RAID-5 configurations have certain performance implications you must consider. Writes to RAID-5 volumes require parity-bit recalculation, which adds significant I/O and CPU overhead. This overhead can cause considerable performance penalties in online transaction processing (OLTP) workloads. If the database has a high read ratio, however, RAID-5 performance is similar to that of a striped volume.

Volume configuration guidelines

Follow these guidelines when selecting volume layouts:

- Put the database log files on a file system created on a striped and mirrored (RAID-0+1) volume separate from the index or data tablespaces. Stripe multiple devices to create larger volumes if needed. Use mirroring to improve reliability. Do not use VxVM RAID-5 for redo logs.
- When normal system availability is acceptable, put the tablespaces on file systems created on striped volumes for most OLTP workloads.
- Create striped volumes across at least four disks. Try to stripe across disk controllers. For sequential scans, ensure that the `NUM_IOSERVERS` and `DB2_PARALLEL_IO` settings are tuned to match the number of disk devices used in the stripe.
- For most workloads, use the default 64 K stripe-unit size for striped volumes and 16 K for RAID-5 volumes.
- When system availability is critical, use mirroring for most write-intensive OLTP workloads. Turn on Dirty Region Logging (DRL) to allow fast volume resynchronization in the event of a system crash.
- When system availability is critical, use RAID-5 for read-intensive OLTP workloads to improve database performance and availability. Use RAID-5 logs to allow fast volume resynchronization in the event of a system crash.
- For most decision support system (DSS) workloads, where sequential scans are common, experiment with different striping strategies and stripe-unit sizes. Put the most frequently accessed tables or tables that are accessed together on separate striped volumes to improve the bandwidth of data transfer.

See "About tuning VxVM " on page 251.

Creating a volume

Veritas Volume Manager uses logical volumes to organize and manage disk space. A volume is made up of portions of one or more physical disks, so it does not have the limitations of a physical disk.

For databases where the data storage needs to be resilient and the data layout needs to be optimized for maximum performance, we recommend using VxVM. The striping and mirroring capabilities offered by a volume manager will help you achieve your manageability, availability, and performance goals.

After you decide on a volume layout, you can use the `vxassist` command to create the volume.

Before creating a volume, make sure the following conditions are met:

Usage notes

- Creating a volume requires a disk group name, volume name, volume size, and volume layout. You will also need to know subdisk names if you are creating a striped volume.
- Striped or mirrored volumes require at least two disks.
- Striped pro and concatenated pro volumes are mirrored by default, so a striped pro volume requires more disks than an unmirrored striped volume and a concatenated pro volume requires more disks than an unmirrored concatenated volume.
- You cannot use a striped pro or concatenated pro volume for a `root` or `swap` volume.
- A RAID-5 volume requires at least three disks. If RAID-5 logging is enabled, a RAID-5 volume requires at least four disks. RAID-5 mirroring is not supported.

To create a volume

- Use the `vxassist` command as follows:

```
# /opt/VRTS/bin/vxassist -g disk_group make volume_name \
size disk_name
```

The following is an example of creating a volume using the `vxassist` command:

To create a 1 GB mirrored volume called `db01` on the `PRODdg` disk group:

```
# /opt/VRTS/bin/vxassist -g PRODdg make db01 1g PRODdg01
```

Creating a volume set

Volume Sets enable the use of the Multi-Volume Support feature with Veritas File System (VxFS). It is also possible to use the Veritas Enterprise Administrator (VEA) to create and administer volumes sets. For more information, see the VEA online help.

Before creating a volume set, make sure the following conditions are met:

Usage notes

- Before creating a volume set, you must have at least one volume created.
  See "Creating a volume " on page 48.
- A maximum of 256 volumes may be configured in a volume set.
- Only Veritas File System is supported on a volume set.
- The first volume in a volume set must be larger than 20MB.
- Raw I/O from and to a volume set is not supported.
- Volume sets can be used instead of volumes with the following `vxsnap` operations on instant snapshots: `addmir`, `dis`, `make`, `prepare`, `reattach`, `refresh`, `restore`, `rmmir`, `split`, `syncpause`, `syncresume`, `syncstart`, `syncstop`, `syncwait`, and `unprepare`. The third-mirror break-off usage model for full-sized instant snapshots is supported for volume sets provided that sufficient plexes exist for each volume in the volume set.
  See the *Veritas Volume Manager Administrator's Guide*.
- Most VxVM commands require superuser or equivalent privileges.
- For details regarding usage of the `vxvset` command, see the `vxvset(1M)` manual page.

To create a volume set for use by Veritas file system (VxFS)

- Use the following command:

  ```
  # /opt/VRTS/bin/vxvset [-g diskgroup] -t vxfs make volset volume
  ```

  where:

  - `volset` is the name of the volume set
  - `volume` is the name of the first volume in the volume set
  - `-t` defines the content handler subdirectory for the application that is to be used with the volume. This subdirectory contains utilities that an application uses to operate on the volume set. The operation of these utilities is determined by the requirements of the application and not by VxVM.

For example, to create a volume set named `db01vset` that contains the volume `db01`, in the disk group `PRODdg`, you would use the following command:

```
# /opt/VRTS/bin/vxvset -g PRODdg -t vxfs make db01vset db01
```

Adding a volume to a volume set

After creating a volume set, you can add additional volumes.

To add a volume to a volume set

- Use the `vxvset` command as follows:

  ```
  # /opt/VRTS/bin/vxvset [-g diskgroup] [-f] addvol volset \
  volume
  ```

  Warning: The `-f` (force) option must be specified if the volume being added, or any volume in the volume set, is either a snapshot or the parent of a snapshot. Using this option can potentially cause inconsistencies in a snapshot hierarchy if any of the volumes involved in the operation is already in a snapshot chain.

  See the *Veritas Volume Manager Administrator's Guide*.

For example, to add the volume `db02`, to the volume set `db01vset`, use the following command:

```
# /opt/VRTS/bin/vxvset -g PRODdg addvol db01vset db02
```

File system creation guidelines

Follow these guidelines when creating VxFS file systems:

- Specify the maximum block size and log size when creating file systems for databases.
- Except for specifying the maximum log size and support for large files as required, use the VxFS defaults when creating file systems for databases.
- Never disable the intent logging feature of the file system.
- For database logs, create a single file system using a simple (and mirrored, if necessary) volume. Put the other tablespaces and database files on separate file systems created on striped, striped and mirrored, or RAID-5 volumes.

- When using the command line, use the mount points to name the underlying volumes. For example, if a file system named `/db01` is to be created on a mirrored volume, name the volume `db01` and the mirrors `db01-01` and `db01-02` to relate to the configuration objects. If you are using the `vxassist` command or the GUI, this is transparent.
- If Quick I/O is supported, select `vxfs` as the file system type to take advantage of the Quick I/O feature, online administration, fast recovery of the VxFS file system, and superior reliability features.

Choose a file system block size that matches or is a multiple of the `PAGESIZE` parameter in the `create database` or `create tablespace` statement for your DB2 database or tablespace. The PAGESIZE parameter is defined in the `create database` or `create tablespace` statement.

It is possible to have a file system block size that is smaller than the database page size because the database page-size limit can be bigger than the file system block size. It is fine if the file system block size is smaller than the database page size because VxFS will not perform multiple I/O operations for each database I/O operation. VxFS is capable of performing I/Os with multiple blocks. For example, if your database page size is 8k and your file system block size is 4K, VxFS can put two 4k blocks together to perform one 8k database I/O operation. The DB2 instance will also need data in `EXTENTSIZE`, which is a multiple of `PAGESIZE`. These page size rules also apply for extent size.

When creating the file system, set the number of file system blocks in the intent log so that the log size is 16MB. For example, if the file system block size is 8K (that is, 8192), it will take 2000 blocks to make a 16MB log (2000 x 8192 = ~16MB). If the file system block size is 4K (that is, 4096), then twice as many blocks as in the 8K case would need to be allocated (4000 in this example).

Creating a VxFS file system

Always specify `vxfs` as the file system type to take advantage of Quick I/O, Storage Rollback, online administration, fast recovery of the VxFS file system, and superior reliability features.

Before creating a file system, see the following notes:

Usage notes

- See the `mkfs`(1M) and `mkfs_vxfs`(1M) manual pages for more information about the options and variables available for use with the `mkfs` command.
- See the `mount`(1M) and `mount_vxfs`(1M) manual pages for more information about mount settings.

To create a VxFS file system on an existing volume

- Use the `mkfs` command as follows:

```
# /usr/sbin/mkfs -F vxfs [generic_options] \
[-o specific_options] special [size]
```

where:

- `vxfs` is the file system type
- `generic_options` are the options common to most file systems
- `specific_options` are options specific to the VxFS file system
- `special` is the full path name of the raw character device or VxVM volume on which to create the file system (for example, `/dev/vx/rdsk/PRODdg/db01`)
- `size` is the size of the new file system (optional)

If you do not specify `size`, the file system will be as large as the underlying volume or device partition.

For example, to create a VxFS file system that has an 8 KB block size and supports files larger than 2 GB on the newly created `db01` volume:

```
# /usr/sbin/mkfs -F vxfs -o largefiles,bsize=8192,\
logsize=2000 /dev/vx/rdsk/PRODdg/db01
```

The `-o largefiles` specific option allows you to create files larger than 2 GB.

Note: Because `size` is not specified in this example, the size of the file system will be calculated automatically to be the same size as the volume on which the file system is created.

The `mkfs` command displays output similar to the following:

```
version 6 layout

20480 sectors, 10240 blocks of size 1024, log size 1024 blocks
```

You can now mount the newly created file system.

See "Mounting a file system " on page 54.

Large file system and large file support

In conjunction with VxVM, VxFS can support file systems up to 8 exabytes in size. For large database configurations, this eliminates the need to use multiple file systems because of the size limitations of the underlying physical devices.

Changes implemented with the VxFS Version 6 disk layout have greatly expanded file system scalability, including support for large files. You can create or mount file systems with or without large files by specifying either the `largefiles` or `nolargefiles` option in `mkfs` or `mount` commands. If you specify the `nolargefiles` option, a file system cannot contain files 2 GB or larger.

Before creating a VxFS file system, make sure the following conditions are met:

Usage notes

- See the `mount_vxfs` (1M) and `mkfs_vxfs` (1M) manual pages for detailed information on mounting and creating file systems.
- See the `fsadm_vxfs` (1M) manual pages for detailed information about large files.

To enable large files on a file system that was created without the largefiles option

- Use the `fsadm` command as follows:

```
# /opt/VRTS/bin/fsadm -F vxfs -o largefiles \
/mount_point
```

Note: Make sure the applications and tools you use can handle large files before enabling the large file capability. Applications and system administration utilities can experience problems if they are not large file aware.

Multi-volume support

The multi-volume support feature enabled by VxFS Version 6 disk layout allows several volumes to be represented by a single logical object, known as a volume set. The `vxvset` command can be used to create and administer volume sets in Veritas Volume Manager.

VxFS's multi-volume support feature can be used with volume sets. There are two VxFS commands associated with multi-volume support:

- `fsapadm` - VxFS allocation policy administration utility
- `fsvoladm` - VxFS device administration utility

See the *Veritas File System Administrator's Guide*.

Mounting a file system

After creating a VxFS file system, mount the file system using the `mount` command.

By default, the command tries to enable Quick I/O. If Quick I/O is not installed or licensed, no error messages are displayed unless you explicitly specify the mount option. If necessary, you can turn the Quick I/O option off at mount time or you can remount the file system with the option.

Before mounting a file system, make sure the following conditions are met:

Prerequisites
- You must be logged in as `root` to mount a file system.

Usage notes
- The mount point must be an absolute path name (that is, it must begin with `/`).
- See the `mount_vxfs` (1M) manual page for more information about mount settings.
- See the `mount` (1M) manual page for more information about generic mount options.

To mount a file system

- Use the `mount` command as follows:

```
# /usr/sbin/mount -F vxfs [generic_options] [-r] \
[-o specific_options] special /mount_point
```

where:

- `generic_options` are the options common to most file systems
- `-r` mounts the file system as read only
- `specific_options` are options specific to the VxFS file system
- `special` is a block special device
- `/mount_point` is the directory where the file system will be mounted

For example, to mount a file system named `/db01` that supports large files on volume `/dev/vx/dsk/PRODdg/db01`:

```
# mkdir /db01

# /usr/sbin/mount -F vxfs -o largefiles /dev/vx/dsk/PRODdg/db01 \
/db01
```

If you would like `/db01` to be mounted automatically after rebooting, add an entry for it in `/etc/vfstab` as follows:

```
/dev/vx/dsk/PRODdg/db01 /db01 vxfs  largefiles,qio          0   2
```

If you do not need to use Quick I/O files, set `noqio` instead of `qio` as one of the options.

Unmounting a file system

If you no longer need to access the data in a file system, you can unmount the file system using the `umount` command.

Before unmounting a file system, make sure the following conditions have been met:

Prerequisites	■ A file system must exist and be mounted in order to be unmounted.
Usage notes	■ You cannot unmount a file system that is in use. See the `umount` (1M) manual page for more information on mounting file systems.

To unmount a file system

1 Use the `fuser` command to make sure that the file system is not being used:

```
# fuser -c /mount_point
```

where the `-c` option provides information on file system mount points and any files within mounted file systems.

If the file system is being used and you need to unmount it, use the `fuser -ck` command. See the `fuser`(1M) man page for more information.

2 Unmount the file system using one of the `umount` command options:

- `umount special`
- `umount /mount_point`
- `umount -f /mount_point`

where:

- `special` is a block special device
- `/mount_point` is the location where the file system is mounted
- `-f` forcibly unmounts the mount point

The following is an example of unmounting a file system:

To verify that the file system `/db01` is not in use and then unmount the file system:

```
# fuser -c /db01
/db01:
# umount /db01
```

About fragmentation

When free resources are initially allocated to files in a Veritas file system, they are aligned in the most efficient order possible to provide optimal performance. On an active file system, the original order is lost over time as files are created, removed, or resized. As space is allocated and deallocated from files, the available free space becomes broken into fragments. This means that space must be assigned to files in smaller and smaller extents. This process is known as fragmentation. Fragmentation leads to degraded performance and availability. The degree of fragmentation depends on file system usage and activity.

How to control fragmentation

Allocation units in VxFS are designed to help minimize and control fragmentation. Over time, however, file systems eventually become fragmented.

VxFS provides online reporting and optimization utilities to enable you to monitor and defragment a mounted file system. These utilities are accessible through the file system administration command, `fsadm`. Using the `fsadm` command, you can track and eliminate fragmentation without interrupting user access to the file system.

Types of fragmentation

VxFS addresses two types of fragmentation:

- Directory Fragmentation
  As files are created and removed, gaps are left in directory inodes. This is known as directory fragmentation. Directory fragmentation causes directory lookups to become slower.
- Extent Fragmentation
  As files are created and removed, the free extent map for an allocation unit changes from having one large free area to having many smaller free areas. Extent fragmentation occurs when files cannot be allocated in contiguous chunks and more extents must be referenced to access a file. In a case of extreme fragmentation, a file system may have free space that cannot be allocated.

How to monitor fragmentation

You can monitor fragmentation in VxFS by running reports that describe fragmentation levels. Use the `fsadm` command to run reports on directory fragmentation and extent fragmentation. The `df` command, which reports on file system free space, also provides information useful in monitoring fragmentation.

Use the following commands to report fragmentation information:

- `fsadm -D`, which reports on directory fragmentation.
- `fsadm -E`, which reports on extent fragmentation.
- `/opt/VRTS/bin/fsadm [-F vxfs] -o s`, which prints the number of free extents of each size.

Defragmenting a file system

You can use the online administration utility `fsadm` to defragment or reorganize file system directories and extents.

The fsadm utility defragments a file system mounted for read/write access by:

- Removing unused space from directories.
- Making all small files contiguous.
- Consolidating free blocks for file system.

The following options are for use with the `fsadm` utility:

`-d` Reorganizes directories. Directory entries are reordered to place subdirectory entries first, then all other entries in decreasing order of time of last access. The directory is also compacted to remove free space.

Note: If you specify `-d` and `-e`, directory reorganization is always completed first.

`-a` Use in conjunction with the `-d` option to consider files not accessed within the specified number of days as “aged” files. Aged files are moved to the end of the directory. The default is 14 days.

`-e` Reorganizes extents. Files are reorganized to have the minimum number of extents.

Note: If you specify `-d` and `-e`, directory reorganization is always completed first.

`-D -E`	Produces reports on directory and extent fragmentation, respectively. **Note:** If you use both `-D` and `-E` with the `-d` and `-e` options, the fragmentation reports are produced both before and after reorganization.
`-v`	Specifies verbose mode and reports reorganization activity.
`-l`	Specifies the size of a file that is considered large. The default is 64 blocks.
`-t`	Specifies a maximum length of time to run, in seconds. **Note:** The `-t` and `-p` options control the amount of work performed by fsadm, either in a specified time or by a number of passes. By default, `fsadm` runs five passes. If both `-t` and `-p` are specified, fsadm exits if either of the terminating conditions are reached.
`-p`	Specifies a maximum number of passes to run. The default is five. **Note:** The `-t` and `-p` options control the amount of work performed by fsadm, either in a specified time or by a number of passes. By default, `fsadm` runs five passes. If both `-t` and `-p` are specified, fsadm exits if either of the terminating conditions are reached.
`-s`	Prints a summary of activity at the end of each pass.
`-r`	Specifies the pathname of the raw device to read to determine file layout and fragmentation. This option is used when fsadm cannot determine the raw device.

Note: You must have superuser (`root`) privileges to reorganize a file system using the `fsadm` command.

To defragment a file system

- Run the `fsadm` command followed by the options specifying the type and amount of defragmentation. Complete the command by specifying the mount point or raw device to identify the file system.

  ```
  # /opt/VRTS/bin/fsadm [-d] [-D] [-e] [-E] [-s] [-v] \

  [-l largesize] [-a days] [-t time] [-p pass_number] \
  [-r rawdev_path] mount_point
  ```

  Refer to the *Veritas File System Administrator's Guide* for instructions and information on scheduling defragmentation. Veritas File System Administrator's Guide for instructions and information on scheduling defragmentation.

  For example, to defragment a file system:

  ```
  # /opt/VRTS/bin/fsadm -d -D /db2data_qiovm

  Directory Fragmentation Report
               Dirs        Total      Immed    Immeds   Dirs to   Blocks
               Searched    Blocks     Dirs     to Add   Reduce    Reduce
    total             5         1         4         0         0 0
    Directory Fragmentation Report
               Dirs        Total      Immed    Immeds   Dirs to   Blocks
               Searched    Blocks     Dirs     to Add   Reduce    Reduce
    total             5         1         4         0         0 0
  ```

Resizing a file system

If you need to extend or shrink a VxFS file system, you can use the `fsadm` command.

If a VxFS file system requires more space, you can use this procedure to extend the size of the file system. If a VxFS File System is too large and you need the space elsewhere, you can use this procedure to shrink the file system.

Remember to increase the size of the underlying device or volume before increasing the size of the file system.

See the *Veritas Volume Manager Administrator's Guide*.Veritas Volume Manager Administrator's Guide.

Before resizing a file system, the following conditions must be met:

Prerequisites	■ This task requires a mounted file system. You must know either the desired size or the amount of space to add to or subtract from the file system size.
Usage notes	■ See the `format`(1M) manual page. See the `fsadm_vxfs`(1M) manual page.

To resize a file system

◆ Use `fsadm` command as follows:

```
# /opt/VRTS/bin/fsadm -F vxfs [-b newsize] \
  [-r rawdev] /mount_point
```

where:

- newsize is the size (in sectors) to which the file system will increase or shrink
- rawdev specifies the name of the raw device if there is no entry in `/etc/vfstab` and `fsadm` cannot determine the raw device
- /mount_point is the location where the file system is mounted

For example, to extend the file system `/db01` to 2 GB:

```
# /opt/VRTS/bin/fsadm -F vxfs -b 2g /db01
```

Note: See the *Veritas File System Administrator's Guide* and `fsadm_vxfs` (1M) manual page for information on how to perform common file system tasks using `fsadm`.

Resizing a file system and the underlying volume

The `fsadm` command resizes the file system only. If you attempt to use `fsadm` to make the file system the same size or larger than the underlying volume, the `fsadm` command will fail. To resize the file system and its underlying volume, use the `vxresize` command instead.

Warning: Resizing a volume with a usage type other than FSGEN or RAID5 can result in data loss. If such an operation is required, use the `-f` option to forcibly resize such a volume.

Before resizing a file system and the underlying volume, the following conditions must be met:

Prerequisites

- You must know the new desired size of the file system.

Usage notes

- `vxresize` works with VxFS and UFS file systems only.
- If the file system is mounted and VxFS, you can grow or shrink the size. If a VxFS file system is unmounted, you cannot grow or shrink the size.
- If the file system is mounted and UFS, you can grow the size only. If the file system is unmounted and UFS, you can grow size only.
- When resizing large volumes, `vxresize` may take a long time to complete.
- Resizing a volume with a usage type other than FSGEN or RAID5 can result in data loss. If such an operation is required, use the `-f` option to forcibly resize such a volume.
- You cannot resize a volume that contains plexes with different layout types.
- See the `vxresize` (1M) manual page for more details.

To resize a file system and the underlying volume

◆ Use the `vxresize` command as follows:

```
# /etc/vx/bin/vxresize -g disk_group -b -F vxfs -t \
homevolresize homevol volume_size disk_name disk_name
```

For example, to extend a 1-gigabyte volume, `homevol`, that contains a VxFS file system, to 10 gigabytes using the spare disks `disk10` and `disk11`, enter:

```
# /etc/vx/bin/vxresize -b -F vxfs -t homevolresize homevol 10g \
disk10 disk11
```

The `-b` option specifies that this operation runs in the background. Its progress can be monitored by specifying the task tag `homevolresize` to the `vxtask` command.

Chapter 3

Managing the SFDB repository

This chapter includes the following topics:

About the SFDB repository

The SFDB repository stores metadata information required by SFDB software. This information includes data about user databases, snapshot databases, storage configuration, scheduled tasks, and storage statistics.

In prior releases of Veritas Storage Foundation for DB2, the repository was stored on the local host as plain text files, which made the repository vulnerable to corruption or other errors. Also, repository information could not be accessed from a secondary host.

In this release of Veritas Storage Foundation for DB2, the repository is stored in a relatinal database and is managed by a lightweight embedded relational DBMS-VxDBMS. VxDBMS is a special OEM version of a Sybase ASA (Adapative Server Anywhere) DBMS, which is delivered and supported by Symantec. The SFDB repository database consists of a database file, `dbed_db.db`, and transaction log files, *`yymmddxx`*`.log`. VxDBMS supports remote client access from any host in the network that has proper authorization and configuration.

SFDB requires only occasional interaction outside of the initial installation and configuration of Veritas Storage Foundation for DB2.

See the *Veritas Storage Foundation Installation Guide* for more information on configuring the SFDB repository.

Runtime management tasks for SFDB

Most interactions with the SFDB repository are handled automatically by SFDB code. Administrators only need to handle SFDB backup and restore activites and to monitor the hard disk space usage of the repository. A configuration file `/etc/vx/vxdbed/admin.properties` is created during installation to record the file system and volume used for the SFDB repository. The VxDBMS server starts automatically after a system reboot via `rc` files.

Starting, stopping, and checking the SFDB repository with sfua_db_config

See the *Veritas Storage Foundation Installation Guide* for more information on configuring the SFDB repository.

If for any reason the VxDBMS process is not running or is having problems, use the `sfua_db_config` command to start, stop, and check the status of the repository database and its server.

Backing up and restoring the SFDB repository with sfua_rept_adm

Use the `sfua_rept_adm` command to manage the backing up and restoring of the SFDB repository. With this command, you can:

- Create and configure a schedule for automatically backing up the SFDB repository, including specifying the type of backup (full or incremental), its start time and frequency, and the destination directory for the backed up files.
- Restore the repository from backup files.
- Disable or enable an existing backup schedule.
- Create and configure a free-space monitoring schedule that emails an alert when the free space on the file system containing the repository falls to a specified threshhold.
- Disable or enable an existing free-space monitoring schedule.

Note: You must be logged in as root to use the `sfua_rept_adm` command.

Table 3-1 sfua_rept_adm command options

Option	Description
`-h`	Displays help for the command.
`-o backup_sched`	Creates a full or incremental backup schedule. The backup type is specified by the accompanying `-t` option. Only one full and one incremental backup schedule can be created. Creating a second schedule overwrites any existing backup schedule of that same type. After a backup schedule is created it is automatically enabled.
`-o backup_enable`	Re-enables the existing full or incremental backup schedule, as specified by the `-t` option.
`-o backup_disable`	Disables the existing full or incremental backup schedule, as specified by the `-t` option.
`-o restore`	Restores the SFDB repository from the backup files in the directory specified by the `-m` option.
`-o space_monitor`	Creates and configures a free-space monitoring schedule that emails a warning when the free space on the file system containing the SFDB repository reaches or goes below a threshold specified by the `-w` option.
`-o space_monitor_enable`	Enables the free-space monitoring schedule, if one exists.
`-o space_monitor_disable`	Disables the free-space monitoring schedule, if one exists.
`-t full | incr`	Specifies the type of backup schedule being created, enabled , or disabled by the accompanying `-o` option. Specify `full` for a full backup or `incr` for an incremental backup. An incremental backup copies the SFDB repository database transaction logs to the directory specified by the `-m` option. A full backup copies these transaction log files and the database file. No more than two backup schedules can be enabled simultaneously, one incremental backup and one full backup. For example, you cannot have two different incremental backup schedules.

Table 3-1 sfua_rept_adm command options *(continued)*

Option	Description
`-f backup_freq`	Specifies the frequency of the scheduled backups as `h` (hours), `d` (days), or `w` (weeks) from the start time specified by the `-s` option. By default, the backup frequency value is assumed to be hours (`h`).
`-s start_time`	Specifies the time to start the backup process in *hh*`:`*mm*`:`*ss* format.
`-m backup_dest`	Specifies the directory where the backup files are stored.
`-w warn_threshold`	Specifies the warning threshold for the free-space monitoring schedule, as a number represetning the percentage of free space on the file system containing the backup files. If the percentage of free space falls to this value or lower, then a warning is emailed according to the other settings in this free-space monitoring schedule.
`-e notify_email`	Specifies the email address to send the warning message when the repository file system free space falls below the threshold specified by the `-w` option.
`-u smtp_sender`	Specifies the SMTP sender in whose name the warning email is emailed when the repository file system free space falls below the threshold specified by the `-w` option.
`-s smtp_server`	specifies the smtp email server to use when sending the warning message when the repository file system free space falls below the threshold specified by the `-w` option.

To create a backup schedule for the SFDB repository

- Use the `sfua_rept_adm -o backup_sched` command:

  ```
  sfua_rept_adm -o backup_sched -t full|incr -f backup_freq \
  -s start_time -m backup_dest
  ```

  After a backup schedule is created, it is automatically enabled. You can create only one of each type of backup schedule, incremental (`-t incr`) and full (`-t full`). If you create a a new backup schedule, it automatically replaces any currently-enabled backup schedule. You can disable the current backup schedule with the `-o backup_disable` command, and re-enable it with `-o backup_enable`.

  In an HA environment, use NFS to mount the backup destination directory on all nodes.

To restore the SFDB repository from a backup

- Use the `sfua_rept_adm -o restore` command:

  ```
  sfua_rept_adm -o restore -m backup_dest
  ```

  This command restores the repository using the full backup and all incremental backup files from the backup directory specified by `-m backup_dest`.

To determine if a repository backup has failed

- Use either of these methods:
  - Check the system console for error messages received at the time the backup was scheduled.
  - Verify the existence of the proper backup files in the backup directory (specified by `-m backup_dest`). The type of repository backup you schedule determines which files should be found in this directory. If an incremental backup, only repository transaction log files (`yymmddxx.log`) are created there. If a full backup, both transaction log files and a repository database file (`dbed_db.db`) are created.

Monitoring free space for the SFDB repository

To gaurd against the SFDB repository failing by filling its file system, use the `sfua_rept_adm -o space_monitor` command to create a monitoring schedule.

Table 3-1 shows all options for the `sfua_rept_adm` command, including those used for creating, disabling, and re-enabling a free space monitoring schedule.

This schedule monitors the available free space on the repository file system. If the free space falls below a specified threshold (a percentage value), a warning is emailed to a specified user.

Note: You must be logged in as root to use the `sfua_rept_adm` command.

To create a free-space monitoring schedule for the repository file system

- Use the `sfua_rept_adm -o space_monitor` command:

  ```
  sfua_rept_adm -o space_monitor -w warn_threshold -e notify_email \
  -u smtp_sender -s smtp_server
  ```

  After a free-space monitoring schedule is created, it is automatically enabled. When the free space on the file system containing the SFDB repository falls below the threshold specified by `-w warn_threshold`, a warning is sent to the email address specified by `-e notify_email`.

Adding a new system to an HA configuration

When adding a new system to an existing HA configuration, you must also add the system to the existing SFDB repository so that it can share the same repository data.

To add a new system to the SFDB repository

1. After installing Veritas Storage Foundation for DB2, add the new system to the cluster.

   See the *Veritas Cluster Server User's Guide*.

2. Make sure the system is running using the `hasys` command:

   ```
   # hasys -state
   ```

3 Add the system to the `Sfua_Base` group by running the following command sequence:

```
# haconf -makerw

# hagrp -modify Sfua_Base SystemList -add new_sys sys_id

# hares -modify sfua_ip Device new_sys_NIC -sys new_sys

# haconf -dump -makero
```

4 Copy the `/etc/vx/vxdbed/.odbc.ini` file from an existing node to the new system using a remote file copy utility such as `rcp`, `tcp`, or `scp`.

For example, to use `rcp`:

```
# rcp /etc/vx/vxdbed/.odbc.ini new_sys:/etc/vx/vxdbed
```

Accessing an off-host repository in a non-VCS environment

When creating an off-host clone database in a non-VCS environment, you must make sure that the secondary host has access to the SFDB repository located on the primary host. Because there is no cluster file system to ensure this shared access, you must establish access by copying the primary host's `.odbc.ini` file to the secondary host.

To share an off-host SFDB repository in a non-VCS environment

- Copy the DSN file `/etc/vx/vxdbed/.odbc.ini` from the primary host (where the repository exists and is mounted) to the corresponding location on the secondary host. Use a remote file copy utility such as `rcp`, `tcp`, or `scp` to make this copy.

  For example, to use `rcp` from the primary host:

  ```
  # rcp /etc/vx/vxdbed/.odbc.ini second_host:/etc/vx/vxdbed
  ```

Chapter 4

Using Veritas Quick I/O

This chapter includes the following topics:

About Quick I/O

Veritas Quick I/O is a VxFS feature included in Veritas Storage Foundation for DB2 that lets applications access preallocated VxFS files as raw character devices. Quick I/O provides the administrative benefits of running databases on file systems without the typically associated degradation in performance.

How Quick I/O works

Veritas Quick I/O supports direct I/O and kernel asynchronous I/O and allows databases to access regular files on a VxFS file system as raw character devices.

The benefits of using Quick I/O are:

- Improved performance and processing throughput by having Quick I/O files act as raw devices.
- Ability to manage Quick I/O files as regular files, which simplifies administrative tasks such as allocating, moving, copying, resizing, and backing up containers.

How Quick I/O improves database performance

Quick I/O's ability to access regular files as raw devices improves database performance by:

- Supporting kernel asynchronous I/O
- Supporting direct I/O
- Avoiding kernel write locks on database files
- Avoiding double buffering

Supporting kernel asynchronous I/O

Asynchronous I/O is a form of I/O that performs non-blocking system level reads and writes, allowing the system to handle multiple I/O requests simultaneously. Operating systems such as Solaris provide kernel support for asynchronous I/O on raw devices, but not on regular files. As a result, even if the database server is capable of using asynchronous I/O, it cannot issue asynchronous I/O requests when the database runs on file systems. Lack of asynchronous I/O significantly degrades performance. Quick I/O lets the database server take advantage of kernel-supported asynchronous I/O on file system files accessed using the Quick I/O interface.

Supporting direct I/O

I/O on files using `read`() and `write`() system calls typically results in data being copied twice: once between user and kernel space, and later between kernel space and disk. In contrast, I/O on raw devices is direct. That is, data is copied directly between user space and disk, saving one level of copying. As with I/O on raw devices, Quick I/O avoids extra copying.

Avoiding kernel write locks

When database I/O is performed using the `write`() system call, each system call acquires and releases a write lock inside the kernel. This lock prevents multiple simultaneous write operations on the same file. Because database systems usually implement their own locking to manage concurrent access to files, per file writer locks unnecessarily serialize I/O operations. Quick I/O bypasses file system per file locking and lets the database server control data access.

Avoiding double buffering

Most database servers maintain their own buffer cache and do not need the file system buffer cache. Database data cached in the file system buffer is therefore redundant and results in wasted memory and extra system CPU utilization to manage the buffer. By supporting direct I/O, Quick I/O eliminates double buffering. Data is copied directly between the relational database management system (RDBMS) cache and disk, which lowers CPU utilization and frees up memory that can then be used by the database server buffer cache to further improve transaction processing throughput.

About Quick I/O requirements

To use Quick I/O, you must:

- Preallocate files on a VxFS file system
- Use a special file naming convention to access the files

Preallocating files

Preallocating database files for Quick I/O allocates contiguous space for the files. The file system space reservation algorithms attempt to allocate space for an entire file as a single contiguous extent. When this is not possible due to lack of contiguous space on the file system, the file is created as a series of direct extents. Accessing a file using direct extents is inherently faster than accessing the same data using indirect extents. Internal tests have shown performance degradation in OLTP throughput when using indirect extent access. In addition, this type of preallocation causes no fragmentation of the file system.

You must preallocate Quick I/O files because they cannot be extended through writes using their Quick I/O interfaces. They are initially limited to the maximum size you specify at the time of creation.

See "Extending a Quick I/O file" on page 86.

About Quick I/O naming conventions

VxFS uses a special naming convention to recognize and access Quick I/O files as raw character devices. VxFS recognizes the file when you add the following extension to a file name:

```
::cdev:vxfs:
```

Whenever an application opens an existing VxFS file with the extension `::cdev:vxfs:` (`cdev` being an acronym for character device), the file is treated as if it were a raw device. For example, if the file `temp01` is a regular VxFS file, then an application can access `temp01` as a raw character device by opening it with the name:

```
.temp01::cdev:vxfs:
```

Note: We recommend reserving the `::cdev:vxfs:` extension only for Quick I/O files. If you are not using Quick I/O, you could technically create a regular file with this extension; however, doing so can cause problems if you later enable Quick I/O.

How to set up Quick I/O

Quick I/O is included in the VxFS package shipped with Veritas Storage Foundation for DB2. By default, Quick I/O is enabled when you mount a VxFS file system.

If Quick I/O is not available in the kernel, or the Veritas Storage Foundation for DB2 license is not installed, a file system mounts without Quick I/O by default, the Quick I/O file name is treated as a regular file, and no error message is displayed. If, however, you specify the `-o qio` option, the `mount` command prints the following error message and terminates without mounting the file system.

```
VxFDD: You don't have a license to run this program
vxfs mount: Quick I/O not available
```

Depending on whether you are creating a new database or are converting an existing database to use Quick I/O, you have the following options:

If you are creating a new database:

- You can use the `qiomkfile` command to preallocate space for database files and make them accessible to the Quick I/O interface.
  See "Creating database containers as Quick I/O files using qiomkfile" on page 75.
- You can use the `setext` command to preallocate space for database files and create the Quick I/O files.

See "Preallocating space for Quick I/O files using the setext command" on page 77.

If you are converting an existing database:

- You can create symbolic links for existing VxFS files, and use these symbolic links to access the files as Quick I/O files.
  See "Accessing regular VxFS files as Quick I/O files" on page 78.
- You can convert your existing DB2 database files to use the Quick I/O interface using the `qio_getdbfiles` and `qio_convertdbfiles` commands.
  See "Converting DB2 containers to Quick I/O files" on page 79.

Creating database containers as Quick I/O files using qiomkfile

You can create Database Managed Space (DMS) containers with the type 'DEVICE' using Quick I/O. The best way to preallocate space for tablespace containers and to make them accessible using the Quick I/O interface is to use the `qiomkfile`. You can use the `qiomkfile` to create the Quick I/O files for either temporary or permanent tablespaces.

Prerequisites

- You can create Quick I/O files only on VxFS file systems.
- If you are creating containers on an existing file system, run `fsadm` (or similar utility) to report and eliminate fragmentation.
- You must have read/write permissions on the directory in which you intend to create DB2 Quick I/Ofiles.

Usage notes

- The `qiomkfile` command creates two files: a regular file with preallocated, contiguous space, and a file that is a symbolic link pointing to the Quick I/O name extension.
- See the `qiomkfile`(1M) manual page for more information.

`-a`
: Creates a symbolic link with an absolute path name for a specified file. Use the `-a` option when absolute path names are required. However, the default is to create a symbolic link with a relative path name.

`-e`
: Extends a file by a specified amount to allow DB2 tablespace resizing.

  See "Extending a Quick I/O file" on page 86.

`-r`
: Increases the file to a specified size to allow DB2 tablespace resizing.

  See "Extending a Quick I/O file" on page 86.

`-s`	Specifies the space to preallocate for a file in bytes, kilobytes, megabytes, gigabytes, or sectors (512 bytes) by adding a `k`, `K`, `m`, `M`, `g`, `G`, `s`, or `S` suffix. The default is bytes—you do not need to attach a suffix to specify the value in bytes. The size of the file that is preallocated is the total size of the file (including the header) rounded to the nearest multiple of the file system block size.

Warning: Exercise caution when using absolute path names. Extra steps may be required during database backup and restore procedures to preserve symbolic links. If you restore files to directories different from the original paths, you must change the symbolic links that use absolute path names to point to the new path names before the database is restarted.

To create a container as a Quick I/O file using qiomkfile

1 Create a Quick I/O-capable file using the `qiomkfile` command:

```
$ /opt/VRTS/bin/qiomkfile  -s file_size /mount_point/filename
```

2 Create tablespace containers using this file with the following SQL statements:

```
$ db2 connect to database
$ db2 create tablespace tbsname managed by database using \
( DEVICE /mount_point/filename size )
$ db2 terminate
```

An example to show how to create a 100MB Quick I/O-capable file named `dbfile` on the VxFS file system `/db01` using a relative path name:

```
$ /opt/VRTS/bin/qiomkfile -s 100m /db01/dbfile
$ ls -al
-rw-r--r--   1 db2inst1  db2iadm1 104857600  Oct 2 13:42  .dbfile
lrwxrwxrwx   1 db2inst1  db2iadm1        19  Oct 2 13:42   dbfile -> \
.dbfile::cdev:vxfs:
```

In the example, `qiomkfile` creates a regular file named `/db01/.dbfile`, which has the real space allocated. Then, `qiomkfile` creates a symbolic link named `/db01/dbfile`. This symbolic link is a relative link to the Quick I/O interface for `/db01/.dbfile`, that is, to the `.dbfile::cdev:vxfs:` file. The symbolic link allows `.dbfile` to be accessed by any database or application using its Quick I/O interface.

We can then add the file to the DB2 database `PROD`:

```
$ db2 connect to PROD
$ db2 create tablespace NEWTBS managed by database using \
( DEVICE '/db01/dbfile' 100m )
$ db2 terminate
```

Preallocating space for Quick I/O files using the setext command

As an alternative to using the `qiomkfile` command, you can also use the VxFS `setext` command to preallocate space for database files.

Before preallocating space with `setext`, make sure the following conditions have been met:

Prerequisites
- The `setext` command requires superuser (`root`) privileges.

Usage notes
- You can use the `chown` command to change the owner and group permissions on the file after you create it.
  See the `setext` (1M) manual page for more information.

To create a Quick I/O database file using setext

1 Access the VxFS mount point and create a file:

```
# cd /mount_point

# touch .filename
```

2 Use the `setext` command to preallocate space for the file:

```
# /opt/VRTS/bin/setext -r size -f noreserve -f chgsize  \
.filename
```

3 Create a symbolic link to allow databases or applications access to the file using its Quick I/O interface:

```
# ln -s .filename::cdev:vxfs: filename
```

4 Change the owner and group permissions on the file:

```
# chown db2inst1:db2iadm1 .filename

# chmod 660 .filename
```

An example to show how to access the mount point `/db01`, create a container, preallocate the space, and change the permissions:

```
# cd /db01
# touch .dbfile
# /opt/VRTS/bin/setext -r 100M -f noreserve -f chgsize .dbfile
# ln -s .dbfile::cdev:vxfs: dbfile

# chown db2inst1:db2iadm1 .dbfile

# chmod 660 .dbfile
```

Accessing regular VxFS files as Quick I/O files

You can access regular VxFS files as Quick I/O files using the `::cdev:vxfs:` name extension.

While symbolic links are recommended because they provide easy file system management and location transparency of database files, the drawback of using symbolic links is that you must manage two sets of files (for instance, during database backup and restore).

Usage notes

- When possible, use relative path names instead of absolute path names when creating symbolic links to access regular files as Quick I/O files. Using relative path names prevents copies of the symbolic link from referring to the original file when the directory is copied. This is important if you are backing up or moving database files with a command that preserves the symbolic link.
  However, some applications require absolute path names. If a file is then relocated to another directory, you must change the symbolic link to use the new absolute path. Alternatively, you can put all the symbolic links in a directory separate from the data directories. For example, you can create a directory named `/database` and put all the symbolic links there, with the symbolic links pointing to absolute path names.

To access an existing regular file as a Quick I/O file on a VxFS file system

1 Access the VxFS file system mount point containing the regular files:

```
$ cd /mount_point
```

2 Create the symbolic link:

```
$ mv filename .filename
$ ln -s .filename::cdev:vxfs: filename
```

This example shows how to access the VxFS file `dbfile` as a Quick I/O file:

```
$ cd /db01
$ mv dbfile .dbfile
$ ln -s .dbfile::cdev:vxfs: dbfile
```

This example shows how to confirm the symbolic link was created:

```
$ ls -lo .dbfile dbfile

-rw-r--r--  1 db2inst1 104890368 Oct 2 13:42  .dbfile

lrwxrwxrwx  1 db2inst1    19     Oct 2 13:42   dbfile ->
.dbfile::cdev:vxfs:
```

Converting DB2 containers to Quick I/O files

Special commands, available in the `/opt/VRTSdb2ed/bin` directory, are provided to assist you in converting an existing database to use Quick I/O. You can use the `qio_getdbfiles` command to extract a list of file names from the database system

tables and the `qio_convertdbfiles` command to convert this list of database files to use Quick I/O.

Note: It is recommended that you create a Storage Checkpoint before converting to or from Quick I/O.

See "Creating Storage Checkpoints using db2ed_ckptcreate" on page 282.

Before converting database files to Ouick I/O files, the following conditions must be met:

Prerequisites

- Log in as the DB2 instance owner (typically, the user ID `db2inst1`) to run the `qio_getdbfiles` and `qio_convertdbfiles` commands.
- You must predefine the DB2 environment variable `$DB2DATABASE`.
- The SFDB repository must exist before you can convert to Quick I/O files. Run the `db2ed_update` command to update or create the repository.
- Files you want to convert must be regular files on VxFS file systems or links that point to regular VxFS files

Usage notes

- Converting existing database files to Quick I/O files may not be the best choice if the files are fragmented. Use the `-f` option to determine the fragmentation levels and choose one of two approaches: Either exclude files that are highly fragmented and do not have sufficient contiguous extents for Quick I/O use, or create new files with the `qiomkfile` command, rather than convert them with the `qio_convertdbfiles` command.
- If you choose to create new files, they will be contiguous. You must then move data from the old files to the new files using the DB2 database export/import facility `db2move` or `restore redirect`, and then define the new files to the database.
- `qio_getdbfiles` skips any tablespaces that have a type of system managed space (SMS), as these tablespaces are based on a directory format and not suitable for conversion.
  The `qio_getdbfiles` command retrieves a list of database files and saves them in a file named `mkqio.dat` in the current directory.
- Instead of using the `qio_getdbfiles` command, you can manually create the `mkqio.dat` file containing the DB2 instance filenames that you want to convert to Quick I/O files.
- The `qio_convertdbfiles` command exits and prints an error message if any of the database files are not on a VxFS file system. If this happens, you must remove any non-VxFS files from the `mkqio.dat` file before running the `qio_convertdbfiles` command.

The following options are available for the `qio_getdbfiles` command:

`-T` Lets you specify the type of database as `db2`. Specify this option only in environments where the type of database is ambiguous (for example, when multiple types of database environment variables, such as `$ORACLE_SID`, `SYBASE`, `DSQUERY`, and `$DB2INSTANCE`, are present on a server).

The following options are available for the `qio_convertdbfiles` command:

`-a` Changes regular files to Quick I/O files using absolute path names. Use this option when symbolic links need to point to absolute path names (for example, at a site that uses SAP).

`-f` Reports on the current fragmentation levels for database files listed in the mkqio.dat file. Fragmentation is reported as not fragmented, slightly fragmented, fragmented, highly fragmented.

`-h` Displays a help message.

-T	Lets you specify the type of database as db2. Specify this option only in environments where the type of database is ambiguous (for example, when multiple types of database environment variables, such as $ORACLE_SID, SYBASE, DSQUERY, and $DB2INSTANCE are present on a server).
-u	Changes Quick I/O files back to regular files. Use this option to undo changes made by a previous run of the qio_convertdbfiles script.

To extract a list of DB2 containers to convert

- With the database instance up and running, run the qio_getdbfiles command from a directory for which you have write permission:

```
$ cd /extract_directory

$ export DB2DATABASE=database_name

$ /opt/VRTSdb2ed/bin/qio_getdbfiles
```

The qio_getdbfiles command extracts the list file names from the database system tables and stores the file names and their size in bytes in a file called mkqio.dat under the current directory.

Note: Alternatively, you can manually create the mkqio.dat file containing the DB2 database container names that you want to convert to use Quick I/O. You can also manually edit the mkqio.dat file generated by qio_getdbfiles, and remove files that you do not want to convert to Quick I/O files.

Note: To run the qio_getdbfiles command, you must have permission to access the database and permission to write to the /extract_directory.

The mkqio.dat list file should look similar to the following:

```
file1 --> .file1::cdev:vxfs:
file2 --> .file2::cdev:vxfs:
file3 --> .file3::cdev:vxfs:
file4 --> .file4::cdev:vxfs:
file5 --> .file5::cdev:vxfs:
```

To convert the DB2 database files to Quick I/O files

1 Make the database inactive by either shutting down the instance or disabling user connections.

Warning: Running the `qio_convertdbfiles` command while the database is up and running can cause severe problems with your database, including loss of data and corruption.

2 Run the `qio_convertdbfiles` command from the directory containing the `mkqio.dat` file:

```
$ cd /extract_directory

$ export DB2DATABASE=database_name

$ /opt/VRTSdb2ed/bin/qio_convertdbfiles
```

The list of files in the `mkqio.dat` file is displayed. For example:

```
file1 --> .file1::cdev:vxfs:
file2 --> .file2::cdev:vxfs:
file3 --> .file3::cdev:vxfs:
file4 --> .file4::cdev:vxfs:
file5 --> .file5::cdev:vxfs:
```

Run the `qio_convertdbfiles` command (with no options specified) to rename the file *filename* to `.filename` and creates a symbolic link to `.filename` with the Quick I/O extension. By default, the symbolic link uses a relative path name.

The `qio_convertdbfiles` script exits and prints an error message if any of the database files are not on a VxFS file system. If this happens, you must remove any non-VxFS files from the `mkqio.dat` file before running the `qio_convertdbfiles` command again.

3 Make the database active again.

You can now access these database files using the Quick I/O interface.

To undo the previous run of qio_convertdbfiles and change Quick I/O files back to regular VxFS files

1. If the database is active, make it inactive by either shutting down the instance or disabling user connections.
2. Run the following command from the directory containing the `mkqio.dat` file:

```
$ cd /extract_directory

$ export DB2DATABASE=database_name

$ /opt/VRTSdb2ed/bin/qio_convertdbfiles -u
```

The list of Quick I/O files in the `mkqio.dat` file is displayed. For example:

```
.file1::cdev:vxfs: --> file1
.file2::cdev:vxfs: --> file2
.file3::cdev:vxfs: --> file3
.file4::cdev:vxfs: --> file4
.file5::cdev:vxfs: --> file5
```

The `qio_convertdbfiles` command with the undo option (`-u`) specified renames the files from <.filename> to <filename> and undoes the symbolic link to `.filename` that was created along with the Quick I/O files.

About sparse files

Support for sparse files lets applications store information (in inodes) to identify data blocks that have only zeroes, so that only blocks containing non-zero data have to be allocated on disk.

For example, if a file is 10KB, it typically means that there are blocks on disk covering the whole 10KB. Assume that you always want the first 9K to be zeroes. The application can go to an offset of 9KB and write 1KB worth of data. Only a block for the 1KB that was written is allocated, but the size of the file is still 10KB.

The file is now sparse. It has a hole from offset 0 to 9KB. If the application reads any part of the file within this range, it will see a string of zeroes.

If the application subsequently writes a 1KB block to the file from an offset of 4KB, for example, the file system will allocate another block.

The file then looks like:

- 0-4KB - hole
- 4-5KB - data block

- 5-9KB - hole
- 9-10KB - data block

So a 1TB file system can potentially store up to 2TB worth of files if there are sufficient blocks containing zeroes. Quick I/O files cannot be sparse and will always have all blocks specified allocated to them.

Displaying Quick I/O status and file attributes

You can obtain and display information about Quick I/O status and file attributes using various options of the `ls` command:

-al	Lists all files on a file system, including Quick I/O files and their links.
-1L	Shows if Quick I/O was successfully installed and enabled.
-a1L	Shows how a Quick I/O file name is resolved to that of a raw device.

To list all files on the current file system, including Quick I/O files and their links

- Use the `ls -al` command with the file names:

```
$ ls -al filename .filename
```

The following example shows how to use the `-a` option to display the absolute path name created using `qiomkfile`:

```
$ ls -al d* .d*

-rw-r--r--  1 db2inst1 db2iadm1 104890368  Oct 2 13:42  .dbfile

lrwxrwxrwx  1 db2inst1 db2iadm1   19       Oct 2 13:42  dbfile ->
 .dbfile::cdev:vxfs:
```

To show a Quick I/O file resolved to a raw device

- Use the `ls` command with the file names as follows:

```
$ ls -alL filename .filename
```

The following example shows how the Quick I/O file name `dbfile` is resolved to that of a raw device:

```
$ ls -alL d* .d*

crw-r--r--   1 db2inst1 db2iadm1 45,  1        Oct 2 13:42    dbfile

-rw-r--r--   1 db2inst1 db2iadm1 104890368     Oct 2 13:42    .dbfile
```

Extending a Quick I/O file

Although Quick I/O files must be preallocated, they are not limited to the preallocated sizes. You can grow or "extend" a Quick I/O file by a specific amount or to a specific size, using options to the `qiomkfile` command. Extending Quick I/O files is a fast, online operation and offers a significant advantage over using raw devices.

Before extending a Quick I/O file, make sure the following conditions have been met:

Prerequisites

- You must have sufficient space on the file system to extend the Quick I/O file.

Usage notes

- You can also grow VxFS file systems online (provided the underlying disk or volume can be extended) using the `fsadm` command. You can expand the underlying volume and the filesystem with the `vxresize` command.
- You must have superuser (`root`) privileges to resize VxFS file systems using the `fsadm` command.
- See the `fsadm_vxfs` (1M) and `qiomkfile` (1M) manual pages for more information.

The following options are available with the `qiomkfile` command:

`-e`	Extends the file by a specified amount to allow DB2 container resizing.
`-r`	Increases the file to a specified size to allow DB2 container resizing.

To extend a Quick I/O file

1 If required, ensure the underlying storage device is large enough to contain a larger VxFS file system (see the `vxassist`(1M) manual page for more information), and resize the VxFS file system using `fsadm` command:

```
# /opt/VRTS/bin/fsadm -b <newsize> /mount_point
```

where:

- `-b` is the option for changing size
- <newsize>is the new size of the file system in bytes, kilobytes, megabytes, blocks, or sectors
- <mount_point>is the file system's mount point

2 Extend the Quick I/O file using the `qiomkfile` command:

```
$ /opt/VRTS/bin/qiomkfile -e extend_amount /mount_point/filename
```

or

```
$ /opt/VRTS/bin/qiomkfile -r newsize /mount_point/filename
```

An example to show how to grow VxFS file system `/db01` to 500MB and extend the `tbs1_cont001` Quick I/O file by 20MB:

```
# /opt/VRTS/bin/fsadm -b 500M /db01

$ /opt/VRTS/bin/qiomkfile -e 20M /db01/tbs1_cont001
```

An example to show how to grow VxFS file system `/db01` to 500MB and resize the `tbs1_cont001` Quick I/O file to 300MB:

```
# /opt/VRTS/bin/fsadm -b 500M /db01

$ /opt/VRTS/bin/qiomkfile -r 300M /db01/tbs1_cont001
```

Monitoring tablespace free space with DB2 and extending tablespace containers

DB2 does not automatically make use of extended DMS files. When tablespace space needs to be extended, a number of DB2 commands must be run. Unlike raw devices, a Database Administrator can easily extend Quick I/O files online. Using this method, a Database Administrator can monitor the free space available in the DB2 tablespaces and use the `qiomkfile` command to grow the Quick I/O files

online as needed (typically when the file is about 80 to 90% full). This method does not require you to lock out unused disk space for Quick I/O files. The free space on the file system is available for use by other applications.

Before extending tablespaces, make sure the following conditions have been met:

Prerequisites
- Log on as the DB2 instance owner.

Usage notes
- Monitor the free space available in the Quick I/O file, and grow the file as necessary with the `qiomkfile` command.
- A Database Administrator can grow underlying VxFS file systems online (provided the underlying disk or volume can be extended) using the `fsadm` command. See the `fsadm` (1M) manual page for more information.
- It is strongly recommended that if a DB2 tablespace is running out of space, that all containers are grown by the same amount. It is possible to add extra containers to the tablespace, but this results in DB2 performing a relayout of the data to balance usage across all available containers in the tablespace.
  Also refer to the DB2 Administration Guide section on *Managing Storage and the DB2 SQL Reference Guide* for information on the `ALTER TABLESPACE` command.

To monitor the free space available in a DB2 tablespace

- Use the following DB2 commands:

```
$ db2 connect to database
$ db2 list tablespaces show detail
$ db2 terminate
```

To extend a Quick I/O file using qiomkfile

- Use the `qiomkfile` command to extend the Quick I/O file (if the container is running low on free blocks):

```
$ /opt/VRTS/bin/qiomkfile -e extend_amount filename
```

To extend a DB2 tablespace by a fixed amount

- Use the following DB2 commands:

```
$ db2 connect to database

$ db2 alter tablespace tablespace-name extend (ALL amount)

$ db2 terminate
```

This example shows how to monitor the free space on the tablespaces in database `PROD`:

```
$ db2 connect to PROD

$ db2 list tablespaces show detail

$ db2 terminate
```

This example shows how to extend the three DB2 containers owned by tablespace `EMP` by 500MB using the `qiomkfile` command:

```
$ /opt/VRTS/bin/qiomkfile -e 500M tbsEMP_cont001

$ /opt/VRTS/bin/qiomkfile -e 500M tbsEMP_cont002

$ /opt/VRTS/bin/qiomkfile -e 500M tbsEMP_cont003
```

This example shows how to notify DB2 that all containers in tablespace `EMP` have grown by 500MB:

```
$ db2 connect to PROD

$ db2 alter tablespace EMP extend (ALL 500M)

$ db2 terminate
```

This example shows how to verify the newly allocated space on the tablespace `EMP` in database `PROD`:

```
$ db2 connect to PROD

$ db2 list tablespaces show detail

$ db2 terminate
```

Recreating Quick I/O files after restoring a database

If you need to restore your database and were using Quick I/O files, you can use the `qio_recreate` command to automatically recreate the Quick I/O files after you have performed a full database recovery. The `qio_recreate` command uses the `mkqio.dat` file, which contains a list of the Quick I/O files used by the database and the file sizes.

For information on recovering your database, refer to the documentation that came with your database software.

Before recreating Quick I/O with the `qio_recreate` command, make sure the following conditions have been met:

Prerequisites

- Recover your database before attempting to recreate the Quick I/O files.
- Log in as the DB2 instance owner (typically, the user ID db2inst1) to run the `qio_recreate` command.
- In the directory from which you run the `qio_recreate` command, you must have an existing `mkqio.dat` file.
- The DB2DATABASE environment variable must be set.
  See "Converting DB2 containers to Quick I/O files" on page 79.

Usage notes

- The `qio_recreate` command supports only conventional Quick I/O files.
- Refer to the `qio_recreate`(1M) manual page for more information.

To recreate Quick I/O files after recovering a database

- Use the `qio_recreate` command as follows:

```
# /opt/VRTSdb2ed/bin/qio_recreate
```

You will not see any output if the command is successful.

When you run the `qio_recreate` command, the following actions occur:

If...	Then...
a Quick I/O file is missing	the Quick I/O file is recreated.
a symbolic link from a regular VxFS file to a Quick I/O file is missing	the symbolic link is recreated.
a symbolic link and its associated Quick I/O file are missing	both the link and the Quick I/O file are recreated.

If...	Then...
a Quick I/O file is missing and the regular VxFS file that it is symbolically linked to is not the original VxFS file	the Quick I/O file is not recreated and a warning message is displayed.
a Quick I/O file is smaller than the size listed in the `mkqio.dat` file	the Quick I/O file is not recreated and a warning message is displayed.

Disabling Quick I/O

If you need to disable the Quick I/O feature, you first need to convert any Quick I/O files back to regular VxFS files. Then, remount the VxFS file system using a special mount option.

Before disabling Quick I/O, make sure the following condition has been met:

Prerequisite — The file system you are planning to remount must be located in the `/etc/filesystems` file.

To disable Quick I/O

1. If the database is active, make it inactive by either shutting down the instance or disabling user connections.
2. To change Quick I/O files back to regular VxFS files, run the following command from the directory containing the `mkqio.dat` list:

   ```
   $ /opt/VRTSdb2ed/bin/qio_convertdbfiles -u
   ```

   The list of Quick I/O files in the `mkqio.dat` file is displayed. For example:

   ```
   .file1::cdev:vxfs: --> file1
   .file2::cdev:vxfs: --> file2
   .file3::cdev:vxfs: --> file3
   .file4::cdev:vxfs: --> file4
   .file5::cdev:vxfs: --> file5
   ```

   The `qio_convertdbfiles` command with the undo option (`-u`) renames the files from .filename to filename and removes the symbolic link to .filename that was created along with the Quick I/O files.
3. To remount the file system with Quick I/O disabled, use the `mount -o noqio` command as follows:

   ```
   # /opt/VRTS/bin/mount -F vxfs -o remount,noqio /mount_point
   ```

Chapter 5

Using Veritas Cached Quick I/O

This chapter includes the following topics:

About Cached Quick I/O

Veritas Cached Quick I/O maintains and extends the database performance benefits of Veritas Quick I/O by making more efficient use of large, unused system memory through a selective buffering mechanism. Cached Quick I/O also supports features that support buffering behavior, such as file system read-ahead.

How Cached Quick I/O works

Cached Quick I/O is a specialized external caching mechanism suitable for dynamically changing memory allocation available for caching DB2 tablespace data. Cached Quick I/O can be selectively applied to containers that are suffering an undesirable amount of physical disk I/O due to insufficient DB2 BufferPool reservation. Cached Quick I/O works by taking advantage of the available physical memory that is left over after the operating system reserves the amount it needs and the DB2 BufferPools have been sized to their normal operating capacity. This extra memory serves as a cache to store file data, effectively serving as a second-level cache backing the BufferPool(s).

For example, consider a system configured with 12GB of physical memory, an operating system using 1GB, and a total DB2 BufferPool size of 3.5GB. Unless you have other applications running on your system, the remaining 7.5GB of memory is unused. If you enable Cached Quick I/O, these remaining 7.5GB become available for caching database files.

Note: You cannot allocate specific amounts of the available memory to Cached Quick I/O. When enabled, Cached Quick I/O takes advantage of available memory.

Cached Quick I/O is not beneficial for all containers in a database. Turning on caching for all database containers can degrade performance due to extra memory management overhead (double buffer copying). You must use file I/O statistics to determine which individual database containers benefit from caching, and then enable or disable Cached Quick I/O for individual containers.

If you understand the applications that generate load on your database and how this load changes at different times during the day, you can use Cached Quick I/O to maximize performance. By enabling or disabling Cached Quick I/O on a per-container basis at different times during the day, you are using Cached Quick I/O to dynamically tune the performance of a database.

For example, files that store historical data are not generally used during normal business hours in a transaction processing environment. Reports that make use of this historical data are generally run during off-peak hours when interactive database use is at a minimum. During normal business hours, you can disable Cached Quick I/O for database files that store historical data in order to maximize memory available to other user applications. Then, during off-peak hours, you can enable Cached Quick I/O on the same files when they are used for report generation. This will provide extra memory resources to the database server without changing any database configuration parameters. Enabling file system read-ahead in this manner and buffering read data can provide great performance benefits, especially in large sequential scans.

You can automate the enabling and disabling of Cached Quick I/O on a per-container basis using scripts, allowing the same job that produces reports to tune the file system behavior and make the best use of system resources. You can specify different sets of containers for different jobs to maximize file system and database performance.

Note: Modifying Cached Quick I/O settings does not require any database level changes. So, unlike modifying existing database global memory or bufferpool settings, Cached Quick I/O does not require the database to be restarted for changes to take effect.

How Cached Quick I/O improves database performance

Enabling Cached Quick I/O on suitable Quick I/O files improves database performance by using the file system buffer cache to store data. This data storage speeds up system reads by accessing the system buffer cache and avoiding disk I/O when searching for information.

Having data at the cache level improves database performance in the following ways:

- For read operations, Cached Quick I/O caches database blocks in the system buffer cache, which can reduce the number of physical I/O operations and therefore improve read performance.
- For write operations, Cached Quick I/O uses a direct-write, copy-behind technique to preserve its buffer copy of the data. After the direct I/O is scheduled and while it is waiting for the completion of the I/O, the file system updates its buffer to reflect the changed data being written out. For online transaction processing, Cached Quick I/O achieves better than raw device performance in database throughput on large platforms with very large physical memories.
- For sequential table scans, Cached Quick I/O can significantly reduce the query response time because of the read-ahead algorithm used by Veritas File System. If a user needs to read the same range in the file while the data is still in cache, the system is likely to return an immediate cache hit rather than scan for data on the disk.

How to set up Cached Quick I/O

To set up and use Cached Quick I/O, you should do the following in the order in which they are listed:

- Enable Cached Quick I/O on the underlying file systems used for your database.
- Exercise the system in your production environment to generate file I/O statistics.
- Collect the file I/O statistics while the files are in use.
- Analyze the file I/O statistics to determine which files benefit from Cached Quick I/O.
- Disable Cached Quick I/O on files that do not benefit from caching.

Enabling Cached Quick I/O on a file system

Cached Quick I/O depends on Veritas Quick I/O running as an underlying system enhancement in order to function correctly. Follow the procedures listed here to ensure that you have the correct setup to use Cached Quick I/O successfully.

Prerequisites

- You must have permission to change file system behavior using the `vxtunefs` command to enable or disable Cached Quick I/O. By default, you need superuser (`root`) permissions to run the `vxtunefs` command, but other system users do not. Superuser (`root`) must specifically grant database administrators permission to use this command as follows:

  ```
  # chown root:db2iadm1 /opt/VRTS/bin/vxtunefs

  # chmod 4550 /opt/VRTS/bin/vxtunefs
  ```

  where users belonging to the `db2iadm1` group are granted permission to run the `vxtunefs` command. We recommend this selective, more secure approach for granting access to powerful commands.
- You must enable Quick I/O on the file system. Quick I/O is enabled automatically at file system mount time.

If you have correctly enabled Quick I/O on your system, you can proceed to enable Cached Quick I/O as follows:

- Set the file system Cached Quick I/O flag, which enables Cached Quick I/O for all files in the file system.
- Setting the file system Cached Quick I/O flag enables caching for all files in the file system. You must disable Cached Quick I/O on individual Quick I/O files that do not benefit from caching to avoid consuming memory unnecessarily. This final task occurs at the end of the enabling process.

Enabling and disabling the qio_cache_enable flag

As superuser (`root`), set the `qio_cache_enable` flag using the `vxtunefs` command after you mount the file system.

To enable the qio_cache_enable flag for a file system

- Use the `vxtunefs` command as follows:

  ```
  # /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=1 /mount_point
  ```

  For example:

  ```
  # /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=1 /db02
  ```

  where `/db02` is a VxFS file system containing the Quick I/O files and setting the `qio_cache_enable` flag to "1" enables Cached Quick I/O. This command enables caching for all the Quick I/O files on this file system.

To disable the flag on the same file system

- Use the `vxtunefs` command as follows:

  ```
  # /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=0 /mount_point
  ```

  For example:

  ```
  # /opt/VRTS/bin/vxtunefs -s -o qio_cache_enable=0 /db02
  ```

  where `/db02` is a VxFS file system containing the Quick I/O files and setting the `qio_cache_enable` flag to "0" disables Cached Quick I/O. This command disables caching for all the Quick I/O files on this file system.

Making Cached Quick I/O settings persistent across reboots and mounts

You can make the Cached Quick I/O system setting persistent across reboots and mounts by adding a file system entry in the `/etc/vx/tunefstab` file.

Note: The `tunefstab` file is a user-created file. For information on how to create the file and add tuning parameters, see the `tunefstab` (4) manual page.

To enable a file system after rebooting

- Put the file system in the `/etc/vx/tunefstab` file and set the flag entry:

  ```
  /dev/vx/dsk/dgname/volname qio_cache_enable=1
  ```

  where:

  - `/dev/vx/dsk/dgname/volname` is the name of a block device
  - *`dgname`* is the name of the disk group
  - *`volname`* is the name of the volume

For example:

```
/dev/vx/dsk/PRODdg/db01 qio_cache_enable=1
/dev/vx/dsk/PRODdg/db02 qio_cache_enable=1
```

where `/dev/vx/dsk/PRODdg/db01` is the block device on which the file system resides.

The `tunefstab` (4) manual pages contain information on how to add tuning parameters.

See the `tunefstab` (4) manual page.

Note: `vxtunefs` can specify a mount point or a block device; `tunefstab` must always specify a block device only.

Using vxtunefs to obtain tuning information

Check the setting of the `qio_cache_enable` flag for each file system using the `vxtunefs` command.

To obtain information on only the qio_cache_enable flag setting

- Use the `grep` command with `vxtunefs`:

  ```
  # /opt/VRTS/bin/vxtunefs /mount_point | grep qio_cache_enable
  ```

  For example:

  ```
  # /opt/VRTS/bin/vxtunefs /db01 | grep qio_cache_enable
  ```

  where `/db01` is the name of the file system. This command displays only the `qio_cache_enable` setting as follows:

  ```
  qio_cache_enable = 0
  ```

  You can also use the `vxtunefs` command to obtain a more complete list of I/O characteristics and tuning statistics.

  See the `vxtunefs` (1) manual page.

To obtain information on all vxtunefs system parameters

- Use the `vxtunefs` command without `grep`:

  ```
  # /opt/VRTS/bin/vxtunefs /mount_point
  ```

For example:

```
# /opt/VRTS/bin/vxtunefs /db01
```

The `vxtunefs` command displays output similar to the following:

```
Filesystem i/o parameters for /db01
read_pref_io = 65536
read_nstream = 1
read_unit_io = 65536
write_pref_io = 65536
write_nstream = 1
write_unit_io = 65536
pref_strength = 10
buf_breakup_size = 1048576
discovered_direct_iosz = 262144
max_direct_iosz = 1048576
default_indir_size = 8192
qio_cache_enable = 1
write_throttle = 0
max_diskq = 1048576
initial_extent_size = 8
max_seqio_extent_size = 2048
max_buf_data_size = 8192
hsm_write_prealloc = 0
read_ahead = 1
inode_aging_size = 0
inode_aging_count = 0
fcl_maxalloc = 130150400
fcl_keeptime = 0
fcl_winterval = 3600
```

The `vxtunefs`(1) manual pages contain a complete description of `vxtunefs` parameters and the tuning instructions.

See the `vxtunefs`(1) manual page.

Determining candidates for Cached Quick I/O

Determining which files can benefit from Cached Quick I/O is an iterative process that varies with each application. For this reason, you may need to complete the following steps more than once to determine the best possible candidates for Cached Quick I/O.

Before determining candidate files for Quick I/O, make sure the following conditions have been met:

Prerequisites	■ You must enable Cached Quick I/O for the file systems. See "Enabling Cached Quick I/O on a file system" on page 96.
Usage notes	■ See the `qiostat` (1M) manual page for more information.

Collecting I/O statistics

Once you have enabled Cached Quick I/O on a file system, you need to collect statistics to determine and designate the files that can best take advantage of its benefits.

To collect statistics needed to determine files that benefit from Cached Quick I/O

1. Reset the `qiostat` counters by entering:

   ```
   $ /opt/VRTS/bin/qiostat -r /mount_point/filenames
   ```

2. Run the database under full normal load and through a complete cycle (24 to 48 hours in most cases) to determine your system I/O patterns and database traffic in different usage categories (for example, OLTP, reports, and backups) at different times of the day.

3. While the database is running, run `qiostat -l` to report the caching statistics as follows:

   ```
   $ /opt/VRTS/bin/qiostat -l /mount_point/filenames
   ```

   or, use the `-i` option to see statistic reports at specified intervals:

   ```
   $ /opt/VRTS/bin/qiostat -i n /mount_point/filenames
   ```

   where n is time in seconds

   For example:

   To collect I/O statistics from all database containers on file system `/db01`:

   ```
   $ /opt/VRTS/bin/qiostat -l /db01/*
   ```

About I/O statistics

The output of the `qiostat` command is the primary source of information to use in deciding whether to enable or disable Cached Quick I/O on specific files. Statistics are printed in two lines per object.

The second line of information is defined as follows:

- CREAD is the number of reads from the VxFS cache (or total number of reads to Quick I/O files with cache advisory on)
- PREAD is the number of reads going to the disk for Quick I/O files with the cache advisory on
- HIT RATIO is displayed as a percentage and is the number of CREADS minus the number of PREADS times 100 divided by the total number of CREADS. The formula looks like this:

```
(CREADs - PREADs) * 100/ CREADs
```

The `qiostat -l` command output looks similar to the following:

```
/db01/tbs1_cont001  6       1    21     4     10.0  0.0
  6     6       0.0

/db01/tbs2_cont001 62552 38498 250213 153992  21.9  0.4
62567 49060    21.6

/db01/tbs2_cont002 62552 38498 250213 153992  21.9  0.4
62567 49060    21.6
```

Analyze the output to find out where the cache-hit ratio is above a given threshold. A cache-hit ratio above 20 percent on a file for a given application may be sufficient to justify caching on that file. For systems with larger loads, the acceptable ratio may be 30 percent or above. Cache-hit-ratio thresholds vary according to the database type and load.

Using the sample output above as an example, the file `/db01/tbs1_cont001` does not benefit from the caching because the cache-hit ratio is zero. In addition, the file receives very little I/O during the sampling duration.

However, the files `/db01/tbs2_*` have a cache-hit ratio of 21.6 percent. If you have determined that, for your system and load, this figure is above the acceptable threshold, it means the database can benefit from caching. Also, study the numbers reported for the read and write operations. When you compare the number of reads and writes for the `/db01/tbs2_*` files, you see that the number of reads is roughly twice the number of writes. You can achieve the greatest performance gains with Cached Quick I/O when using it for files that have higher read than write activity.

Based on these two factors, `/db01/tbs2_cont001` and `/db01/tbs2_cont002` are prime candidates for Cached Quick I/O.

See "Enabling and disabling Cached Quick I/O for individual files" on page 102.

Effects of read-aheads on I/O statistics

The number of CREADs in the qiostat output is the total number of reads performed, including Cached Quick I/O, and the number of PREADs is the number of physical reads. The difference between CREADs and PREADs (CREADS - PREADS) is the number of reads satisfied from the data in the file system cache. Thus, you expect that the number of PREADs would always be equal to or lower than the number of CREADs.

However, the PREADs counter also increases when the file system performs read-aheads. These read-aheads occur when the file system detects sequential reads. In isolated cases where cache hits are extremely low, the output from qiostat could show that the number of CREADs is lower than the number of PREADs. The cache-hit ratio calculated against these CREAD/PREAD values is misleading when used to determine whether Cached Quick I/O should be enabled or disabled.

Under these circumstances, you can make a more accurate decision based on a collective set of statistics by gathering multiple sets of data points. Consequently, you might want to enable Cached Quick I/O for all the container files that contain a particular table, across multiple tablespaces used by a given database, even if the containers in just one of the tablespaces exhibited a high cache hit ratio. In general, we expect all containers in a tablespace to have approximately the same cache hit ratio.

Other tools for analysis

While the output of the qiostat command is the primary source of information to use in deciding whether to enable Cached Quick I/O on specific files, we also recommend using other tools in conjunction with qiostat. For example, benchmarking software that measures database throughput is also helpful. If a benchmark test in which Cached Quick I/O was enabled for a certain set of data files resulted in improved performance, you can also use those results as the basis for enabling Cached Quick I/O.

Enabling and disabling Cached Quick I/O for individual files

After using qiostat or other analysis tools to determine the appropriate files for Cached Quick I/O, you need to disable Cached Quick I/O for those individual files that do not benefit from caching using the qioadmin command.

Prerequisites	■ Enable Cached Quick I/O for the file system before enabling or disabling Cached Quick I/O at the individual file level.
Usage notes	■ You can enable or disable Cached Quick I/O for individual files while the database is online. ■ You should monitor files regularly using `qiostat` to ensure that a file's cache-hit ratio has not changed enough to reconsider enabling or disabling Cached Quick I/O for the file. ■ Enabling or disabling Cached Quick I/O for an individual file is also referred to as setting the cache advisory on or off. ■ See the `qioadmin` (1) manual page.

Setting cache advisories for individual files

You can enable and disable Cached Quick I/O for individual files by changing the cache advisory settings for those files.

To disable Cached Quick I/O for an individual file

◆ Use the `qioadmin` command to set the cache advisory to `OFF` as follows:

```
$ /opt/VRTS/bin/qioadmin -S filename=OFF /mount_point
```

For example, to disable Cached Quick I/O for the file `/db01/dbfile`, set the cache advisory to `OFF`:

```
$ /opt/VRTS/bin/qioadmin -S dbfile=OFF /db01
```

To enable Cached Quick I/O for an individual file

◆ Use the `qioadmin` command to set the cache advisory to `ON` as follows:

```
$ /opt/VRTS/bin/qioadmin -S filename=ON /mount_point
```

For example, running `qiostat`shows the cache hit ratio for the file `/db01/dbfile`reaches a level that would benefit from caching. To enable Cached Quick I/O for the file `/db01/dbfile`, set the cache advisory to `ON`:

```
$ /opt/VRTS/bin/qioadmin -S dbfile=ON /db01
```

Making individual file settings for Cached Quick I/O persistent

You can make the enable or disable individual file settings for Cached Quick I/O persistent across reboots and mounts by adding cache advisory entries in the `/etc/vx/qioadmin` file.

Cache advisories set using the `qioadmin` command are stored as extended attributes of the file in the inode. These settings persist across file system remounts and system reboots, but these attributes are not backed up by the usual backup methods, so they cannot be restored. Therefore, always be sure to reset cache advisories after each file restore. This is not necessary if you maintain the cache advisories for Quick I/O files in the `/etc/vx/qioadmin` file.

To enable or disable individual file settings for Cached Quick I/O automatically after a reboot or mount

- Add cache advisory entries in the `/etc/vx/qioadmin` file as follows:

```
device=/dev/vx/dsk/<diskgroup>/<volume>

filename,OFF

filename,OFF

filename,OFF

filename,ON
```

For example, to make the Cached Quick I/O settings for individual files in the `/db01` file system persistent, edit the `/etc/vx/qioadmin` file similar to the following:

```
#
# List of files to cache in /db01 file system
#
device=/dev/vx/dsk/PRODdg/db01

dbfile01,OFF
dbfile02,OFF
dbfile03,ON
```

Determining individual file settings for Cached Quick I/O using qioadmin

You can determine whether Cached Quick I/O is enabled or disabled for individual files by displaying the file's cache advisory setting using the `qioadmin` command.

Note: To verify caching, always check the setting of the flag `qio_cache_enable` using `vxtunefs`, along with the individual cache advisories for each file.

To display the current cache advisory settings for a file

- Use the `qioadmin` command with the `-P` option as follows:

  ```
  $ /opt/VRTS/bin/qioadmin -P filename /mount_point
  ```

  For example, to display the current cache advisory setting for the file `dbfile` in the `/db01` file system:

  ```
  $ /opt/VRTS/bin/qioadmin -P dbfile /db01

  dbfile,OFF
  ```

Chapter 6

Using Veritas Concurrent I/O

This chapter includes the following topics:

About Concurrent I/O

Veritas Concurrent I/O improves the performance of regular files on a VxFS file system without the need for extending namespaces and presenting the files as devices. This simplifies administrative tasks and allows databases, which do not have a sequential read/write requirement, to access files concurrently. This chapter describes how to use the Concurrent I/O feature.

With DB2 8.2.2 or later, you can use the Veritas Concurrent I/O feature to improve data write performance for DMS tablespaces with FILE containers. Concurrent I/O can also improve data write performance for most SMS tablespaces.

Quick I/O is still an alternative solution for DMS tablespaces.

See "About Quick I/O" on page 71.

In some cases (for example, if the system has extra memory), Cached Quick I/O may further enhance performance.

See "About Cached Quick I/O" on page 93.

How Concurrent I/O works

Traditionally, UNIX semantics require that read and write operations on a file occur in a serialized order. Because of this, a file system must enforce strict

ordering of overlapping read and write operations. However, databases do not usually require this level of control and implement concurrency control internally, without using a file system for order enforcement.

The Veritas Concurrent I/O feature removes these semantics from the read and write operations for databases and other applications that do not require serialization.

The benefits of using Concurrent I/O are:

- Concurrency between a single writer and multiple readers
- Concurrency among multiple writers
- Minimalization of serialization for extending writes
- All I/Os are direct and do not use file system caching
- I/O requests are sent directly to file systems
- Inode locking is avoided
- **Note:** Concurrent I/O is not recommended for system temporary tablespaces or system catalog tablespaces.

Enabling and disabling Concurrent I/O

Concurrent I/O is not turned on by default and must be enabled manually. You will also have to manually disable Concurrent I/O if you choose not to use it in the future.

Enabling Concurrent I/O

Because you do not need to extend name spaces and present the files as devices, you can enable Concurrent I/O on regular files.

Warning: If you use the `-o cio` option with the `mount` command to mount your primary database file systems, the Concurrent I/O settings will not be preserved when using Database FlashSnap commands or the command.

Before enabling Concurrent I/O, make sure the following conditions have been met:

Prerequisites

- To use the Concurrent I/O feature, the file system must be a VxFS file system.
- Make sure the mount point on which you plan to mount the file system exists.
- Make sure the DBA can access the mount point.

Usage notes

- Files that are open and using Concurrent I/O cannot be opened simultaneously by a different user not using the Concurrent I/O feature.
- Veritas NetBackup cannot backup a database file if the file is open and using Concurrent I/O. However, you can still backup the database online using the utility.
- When a file system is mounted with the Concurrent I/O option, do not enable Quick I/O. DB2 will not be able to open the Quick I/O files and the instance start up will fail. (Quick I/O is not available on Linux.)
- If the Quick I/O feature is availabe, do not use any Quick I/O tools if the database is using Concurrent I/O.
- See the (1M) manual page for more information about mount settings.

To enable Concurrent I/O on a file system using mount with the -o cio option

- Mount the file system using the `mount`command as follows:

  ```
  # /usr/sbin/mount -F vxfs -o cio special /mount_point
  ```

  where:

  - *special* is a block special device
  - /*mount_point* is the directory where the file system will be mounted.

For example, to mount a file system named `/datavol` on a mount point named `/db2data`:

To enable Concurrent I/O on a DB2 tablespace when creating the tablespace

1. Use the `db2 -v "create regular tablespace..."` command with the `no file system caching` option.
2. Set all other parameters according to your system requirements.

To enable Concurrent I/O on an existing DB2 tablespace

- Use the DB2 `no file system caching` option as follows:

```
# db2 -v "alter tablespace tablespace_name no file system caching"
```

  where *`tablespace_name`* is the name of the tablespace for which you are enabling Concurrent I/O.

To verify that Concurrent I/O has been set for a particular DB2 tablespace

1. Use the DB2 `get snapshot` option to check for Concurrent I/O.

```
# db2 -v "get snapshot for tablespaces on dbname"
```

   where *`dbname`* is the database name.

2. Find the tablespace you want to check and look for the `File system caching` attribute. If you see `File system caching = No`, then Concurrent I/O is enabled.

Disabling Concurrent I/O

If you need to disable Concurrent I/O, unmount the VxFS file system and mount it again without the mount option.

To disable Concurrent I/O on a file system using the mount command

1. Shutdown the DB2 instance.
2. Unmount the file sytem using the `umount` command.
3. Mount the file system again using the `mount` command without using the `-o cio` option.

To disable Concurrent I/O on a DB2 tablespace

- Use the DB2 `file system caching` option as follows:

```
# db2 -v "alter tablespace tablespace_name file system caching"
```

  where *`tablespace_name`* is the name of the tablespace for which you are disabling Concurrent I/O.

Chapter 7

Using Storage Mapping

This chapter includes the following topics:

About Storage Mapping

Storage mapping is included with Veritas Storage Foundation for DB2 Enterprise Edition.

Storage mapping allows a DB2 database administrator to display detailed storage layer topology information for a DB2 container. The database administrator can view the layout of all the containers for a tablespace to identify potential trouble spots. You may obtain and view detailed storage topology information using the `vxstorage_stats` commands or the Veritas Storage Foundation for DB2 GUI.

Access to mapping information is important since it allows for a detailed understanding of the storage hierarchy in which files reside, information that is critical for effectively evaluating I/O performance.

Mapping files to their underlying device is straightforward when datafiles are created directly on a raw device. With the introduction of host-based volume managers and sophisticated storage subsystems that provide RAID features, however, mapping files to physical devices has become more difficult.

With the Veritas Storage Foundation for DB2 Storage Mapping option, you can map containers to physical devices. Veritas Storage Mapping relies on Veritas Mapping Service (VxMS), a library that assists in the development of distributed

SAN applications that must share information about the physical location of files and volumes on a disk.

The Veritas Storage Foundation for DB2 Storage Mapping option supports a wide range of storage devices and allows for “deep mapping” into EMC, Hitachi, and IBM Enterprise Storage Server (“Shark”) arrays. Deep mapping information identifies the physical disks that comprise each LUN and the hardware RAID information for the LUNs.

You can view storage mapping topology information and I/O statistics using:

- The `vxstorage_stats` command. This command displays the complete I/O topology mapping of specific containers through intermediate layers like logical volumes down to actual physical devices.
- Veritas Storage Foundation for DB2 GUI. The Veritas Storage Foundation for DB2 performs file mapping and displays both storage mapping topology information and I/O statistics.

Note: For Solaris users, if you use Veritas FlashSnap Agent for Symmetrix, you cannot use the mapping functionality for non-Symmetrix arrays.

Verifying Veritas Storage Mapping setup

Before using the Veritas Storage Mapping option, verify that the features are set up correctly.

To verify that your system is using the Veritas Storage Mapping option

1 Verify that you have a license key for the Storage Mapping option.

```
# /opt/VRTS/bin/vxlictest -n "Veritas Mapping Services" -f \
 "Found_Edi_map"

Found_Edi_map feature is licensed
```

2 Verify that the `VRTSvxmsa` package is installed.

```
# pkginfo -l VRTSvxmsa
```

Output similar to the following is displayed:

```
PKGINST:  VRTSvxmsa
NAME: VxMS - Veritas Mapping Service, Application Libraries.
CATEGORY: system,utilities
ARCH:  sparc
VERSION: 4.2.1-REV=build218_2004.10.29
BASEDIR:  /opt
PSTAMP:  oigpsol0920041029112628
INSTDATE: Nov 02 2004 15:22
STATUS:  completely installed
FILES: 33 installed pathnames  8 shared pathnames  14 directories
15 executables      2931 blocks used (approx)
```

Using vxstorage_stats

The `vxstorage_stats` command displays detailed storage mapping information and I/O statistics about an individual VxFS file. The mapping information and I/O statistics are recorded only for VxFS files and VxVM volumes.

In `vxstorage_stats` command output, I/O topology information appears first followed by summary statistics for each object.

The command syntax is as follows:

```
/opt/VRTSdb2ed/bin/vxstorage_stats [-m] [-s] [-i interval \
-c count] -f filename
```

Prerequisites

- You must log in as the instance owner or `root`.

Usage notes

- The `-s` option displays the file statistics for the specified file.
- The `-c` *`count`* option specifies the number of times to display statistics within the interval specified by `-i` *`interval`*.
- The `-i` *`interval`*option specifies the interval frequency for displaying updated I/O statistics.
- The `-f` *`filename`* option specifies the file to display I/O mapping and statistics for.
- For more information, see the `vxstorage_stats`(1m) online manual page.
- The `-m` option displays the I/O topology for the specified file.

Finding container information for DB2 UDB

Currently, DB2 UDB does not offer an interface to link a database page in a table or index to a particular container's offset. However, the `vxstorage_stats` command or the Veritas Storage Foundation for DB2 Graphical User Interface can be used to display the I/O topology. No special setup is needed to view the topology.

To locate a table in a tablespace

1 Connect to the database PROD as the instance owner.

```
$db2 connect to PROD
Database Connection Information

Database server        = DB2/SUN64 7.2.5
SQL authorization ID   = INST1
Local database alias   = PROD
```

2 Enter the following query to display all the tablespaces in the table TABLE01. In this example, TBSPACE is the primary tablespace for all the table data, INDEX_TBSPACE contains the index (if any), and LONG_TBSPACE is the tablespace to store LONG column.

```
$db2 "select tbspace, index_tbspace, long_tbspace from \
syscat.tables where tabname='TABLE01'"


TBSPACE            INDEX_TBSPACE      LONG_TBSPACE
------------------ ------------------ ------------------
USER1              -                  -
1 record(s) selected.
```

3 After locating the tablespace name, enter the following command to find the container path name:

```
$db2 list tablespaces

 ...
Tablespace ID     = 3
Name              = USER1
Type              = Database managed space
Contents          = Any data
State             = 0x0000
Detailed explanation:
Normal
...
```

The output indicates the Tablespace ID is 3 and the container path name is `USER1`:

```
$db2 list tablespace containers for 3

Tablespace Containers for Tablespace 3
Container ID = 0
Name = /data/system01.dbf
Type = Disk
```

Tablespace `USER1` contains only one container:

```
/data/system01.dbf
```

Displaying Storage Mapping information for DB2 containers

After the container information has been determined, display storage mapping information.

To display Storage Mapping information

- Use the `vxstorage_stats` command with the `-m` option:

  ```
  $/opt/VRTSdb2ed/bin/vxstorage_stats -m -f file_name
  ```

  For example:

  ```
  $/opt/VRTSdb2ed/bin/vxstorage_stats -m -f \
  /data/system01.dbf
  ```

  Output similar to the following is displayed:

```
TY NAME                           NSUB   DESCRIPTION   SIZE          OFFSET      PROPERTIES
                                                       (sectors)     (sectors)
TY NAME                           NSUB   DESCRIPTION   SIZE          OFFSET   PROPERTIES
v /dev/vx/rdsk/testdg/stripevol   1      MIRROR        204800        0
pl vxvm:testdg/stripevol-01       3      STRIPE        205056        0        Stripe_size:128
rd /dev/vx/rdmp/c1t3d0s2          1      PARTITION     68352         0
sd /dev/rdsk/c1t3d0s2             1      PARTITION     71127180      0
da c1t3d0                         0      DISK          71127180      0
rd /dev/vx/rdmp/c1t4d0s2          1      PARTITION     68352         0
sd /dev/rdsk/c1t4d0s2             1      PARTITION     71127180      0
da c1t4d0                         0      DISK          71127180      0
rd /dev/vx/rdmp/c1t5d0s2          1      PARTITION     68352         0
sd /dev/rdsk/c1t5d0s2             1      PARTITION     71127180      0
da c1t5d0                         0      DISK          71127180      0
```

Note: For file type (`fi`), the `SIZE` column is number of bytes, and for volume (`v`), plex (`pl`), sub-disk (`sd`), and physical disk (`da`), the `SIZE` column is in 512-byte blocks. Stripe sizes are given in sectors.

Displaying I/O statistics information

To display I/O statistics information

- Use the `vxstorage_stats` command with the `-s` option to display I/O statistics information:

  ```
  $ /opt/VRTSdb2ed/bin/vxstorage_stats -s -f file_name
  ```

  For example:

  ```
  $ /opt/VRTSdb2ed/bin/vxstorage_stats -s -f \
  /PRODqio/PRODqiotbs
  ```

  Output similar to the following is displayed:

  ```
  I/O OPERATIONS     I/O BLOCKS(512 byte)      AVG TIME(ms)

  OBJECT       READ  WRITE   B_READ  B_WRITE   AVG_RD  AVG_WR

  /data/system01.dbf       2  2479   8  5068810   0.00  53.28

  /dev/vx/rdsk/mydb/myindex       101  2497   1592  5069056   12.18  52.7

  vxvm:mydb/myindex-01       101  2497   1592  5069056   12.18  52.76

  /dev/rdsk/c3t1d3s3       131  1656   2096  1689696   14.43  39.09

  c3t1d3       131  1656   2096  1689696   14.43  39.09

  EMC000184502242:02:0c:02       8480  231019   275952  23296162   -  -

  EMC000184502242:31:0c:02       3244  232131   54808  23451325   -  -

  /dev/rdsk/c3t1d15s4       0  1652   0  1689606   0.00  46.47

  c3t1d15       0  1652   0  1689606   0.00  46.47

  EMC000184502242:01:0c:02       23824  1188997   1038336  32407727   -

  EMC000184502242:32:0c:02       5085  852384   135672  29956179   -  -

  /dev/rdsk/c3t1d2s4       14  1668   200  1689834   18.57  34.19

  c3t1d2       14  1668   200  1689834   18.57  34.19

  EMC000184502242:16:0c:02       4406  271155   121368  23463948   -  -

  EMC000184502242:17:0c:02       3290  269281   55432  23304619   -  -

  I/O OPERATIONS     I/O BLOCKS(512 byte)      AVG TIME(ms)

  OBJECT       READ  WRITE   B_READ  B_WRITE   AVG_RD  AVG_WR

  /data/system01.dbf       2  2479   8  5068810   0.00  53.28
  ```

```
/dev/vx/rdsk/mydb/myindex       101  2497   1592  5069056   12.18  52.78
vxvm:mydb/myindex-01       101  2497   1592  5069056   12.18  52.76
/dev/rdsk/c3t1d3s3       131  1656   2096  1689696   14.43  39.09
c3t1d3       131  1656   2096  1689696   14.43  39.09
EMC000184502242:02:0c:02       8480  231019   275952  23296162   -  -
EMC000184502242:31:0c:02       3244  232131   54808  23451325   -  -
/dev/rdsk/c3t1d15s4       0  1652   0  1689606   0.00  46.47
c3t1d15       0  1652   0  1689606   0.00  46.47
EMC000184502242:01:0c:02       23824  1188997   1038336  32407727   -  -
EMC000184502242:32:0c:02       5085  852384   135672  29956179   -  -
/dev/rdsk/c3t1d2s4       14  1668   200  1689834   18.57  34.19
c3t1d2       14  1668   200  1689834   18.57  34.19
EMC000184502242:16:0c:02       4406  271155   121368  23463948   -  -
EMC000184502242:17:0c:02       3290  269281   55432  23304619   -  -
```

To display Storage Mapping and I/O statistics information at repeated intervals

- Use the `vxstorage_stats` command with the `-i interval` and `-c count` options to display storage mapping and I/O statistics information at repeated intervals. The `-i interval` option specifies the interval frequency for displaying updated I/O statistics and the `-c count` option specifies the number of times to display statistics.

  ```
  $ /opt/VRTSdb2ed/bin/vxstorage_stats [-m] [-s] \
  [-i interval -c count ] -f file_name
  ```

  For example, to display statistics twice with a time interval of two seconds:

  ```
  $ /opt/VRTSdb2ed/bin/vxstorage_stats -s -i2 -c2 \
  -f /prodqio/PRODqiotbs
  ```

  Output similar to the following is displayed:

  ```
         OPERATIONS    FILE BLOCKS(512 byte)        AVG TIME(ms)

  OBJECT       READ  WRITE   B_READ   B_WRITE   AVG_RD   AVG_WR

  /data/system01.dbf        0  0  0   0   0.00   0.00

  /dev/vx/rdsk/mapdg/data_vol        0  1  0   2   0.00   0.00

  vxvm:mapdg/data_vol-01        0  1  0   2   0.00   0.00

  /dev/rdsk/c1t10d0s2        0  1  0   2   0.00   0.00

  c1t10d0        0  1  0   2   0.00   0.00

  vxvm:mapdg/data_vol-03        0  1  0   2   0.00   0.00

  /dev/rdsk/c1t13d0s2        0  1  0   2   0.00   0.00

  c1t13d0        0  1  0   2   0.00   0.00
  ```

About arrays for Storage Mapping and statistics

Veritas Storage Foundation for DB2 provides “deep” mapping information and performance statistics for supported storage arrays. Deep mapping information consists of identifying the physical disks that comprise each LUN and the hardware RAID information for the LUNs.

Veritas Array Integration Layer (VAIL) software interfaces third-party hardware storage arrays with Veritas storage software. VAIL providers are software modules

that enable Veritas applications to discover, query, and manage third-party storage arrays.

On Solaris, the following VAIL providers support these third-party storage arrays:

- The `vx_hicommand` provider manages Hitachi arrays.
- The `vx_emc_symmetrix` provider manages EMC Symmetrix arrays.

For the most up-to-date array support information, see the appropriate hardware compatibility list (HCL) on the Veritas Technical Support Web page at:

http://support.veritas.com

If you want to use storage array information accessible through the VAIL providers, install VAIL and perform any required configuration for the storage arrays and VAIL providers. To use deep mapping services and performance statistics for supported storage arrays, you must install both VAIL and Veritas Mapping Services (VxMS).

You will need to install required third-party array CLIs and APIs on the host where you are going to install VAIL before you install VAIL. If you install any required CLI or API after you install VAIL, rescan the arrays so that Veritas Storage Foundation for DB2 can discover them.

For detailed information about supported array models, see the *Veritas Array Integration Layer Array Configuration Guide*.

Chapter 8

Converting existing database configurations to VxFS

This chapter includes the following topics:

Converting native file systems to VxFS with Quick I/O

If you are currently using file systems native to your operating system, use the procedure to upgrade each file system used by the database to a VxFS file system with Quick I/O.

To convert a native file system to VxFS with Quick I/O

1. Make the DB2 database inactive by either shutting down the database or disabling all user connections.
2. Create a backup of the UFS file system.
3. Unmount the UFS file system.
4. Remove the entry for the UFS file system from the `/etc/vfstab` directory.
5. Create a VxFS file system of the same size or larger than the original UFS file system, using the mount point where the UFS file system was originally mounted.

   See "Creating a VxFS file system " on page 51.

6 Preallocate Quick I/O files using `qiomkfile`.

See "Creating database containers as Quick I/O files using qiomkfile" on page 75.

7 Restore the backup that you created earlier to the Quick I/O files in the new VxFS file system.

8 Reactivate the DB2 database.

Upgrading from earlier VxFS version layouts

Before starting the upgrade process, make sure the following conditions have been met:

Prerequisites
- Perform a full backup of the file system before upgrading to a new disk layout.

Usage notes
- The `vxupgrade` command lets you to upgrade the VxFS file system disk layout while the file system is mounted. See the `vxupgrade`(1M) manual page for more details.
- VxFS supports four file system disk layouts: Versions 4, 5, 6, and 7. New file systems are created with the Version 6 (for large file systems) disk layout by default when the current version of Veritas Storage Foundation for DB2 is installed on a system. You must minimally upgrade to Version 4 disk layout if you want to use the Storage Rollback or Veritas NetBackup BLI Backup features.

To upgrade an existing VxFS file system to a new file system disk layout version

◆ Use the `vxupgrade` command to upgrade to Version 4, 5, 6, or 7 disk layout:

```
# /opt/VRTS/bin/upgrade -n new_version new_version/mount_point
```

where:

- new_version is the version of the file system disk layout you want to upgrade to
- `/mount_point` is the location where the file system is mounted

This is an example of upgrading to disk layout Version 7:

```
# /opt/VRTS/bin/vxupgrade -n 7 /db01
```

To use Quick I/O after upgrading the file system disk layout to version 4, 5, 6, or 7

1. Make the DB2 database inactive by either shutting down the database or disabling all user connections.
2. Make each datafile accessible as a Quick I/O file.

   See "Accessing regular VxFS files as Quick I/O files" on page 78.
3. Reactivate the DB2 database.

Converting from raw devices

If the database is currently using raw disks or volumes, you can do the conversion using DB2 redirected restore. DB2 redirected restore allows you to redefine or redirect data to new tablespace containers during a restore. Use the following procedure to use VxFS with the Quick I/O feature. For more information on DB2 backup and restore, refer to the relevant chapters in the administration guide or command reference book.

See the *DB2 Administration Guide*.

See the *DB2 Command Reference*.

Warning: The procedure provided assumes that the database runs on a single file system after the upgrade.

To convert from raw devices to VxFS with Quick I/O

1. Create a VxFS file system using a size that is 10 percent larger than the original database or total raw device size. You can create more file systems based on your performance and availability requirements.

   See "Creating a VxFS file system " on page 51.
2. Verify that the database can perform rollforward recovery. In order to do this, the database parameters LOGRETAIN and/or USEREXIT must be enabled. For example:

   ```
   $ db2 get db cfg for DATABASE | egrep "LOGRET|USEREXIT"
   $ db2 update db cfg for DATABASE USING LOGRETAIN ON
   $ db2 update db cfg for DATABASE USING USEREXIT ON
   ```
3. Preallocate Quick I/O files using `qiomkfile`.

   See "Creating database containers as Quick I/O files using qiomkfile" on page 75.

4 Backup the existing database.

For example, use the DB2 `backup database` command to dump all data:

```
$ db2 backup database DATABASE to /dumpdir
```

5 Stop the DB2 instance and then restart.

For example:

```
$ db2stop force
$ db2 db2start
```

6 Restore the database to the newly defined Quick I/O files:

```
$ db2 restore db DATABASE from /dumpdir taken at TIME \

replace existing redirect
```

7 Redefine the storage that the database is now to use:

```
$ db2 set tablespace containers for TBS_ID using \

(device '/qio_file_path' TBS_SIZE)
```

8 Continue the restore process:

```
$ db2 restore db DATABASE continue
$ db2 rollforward db DATABASE to end of logs
$ db2 rollforward db DATABASE complete
```

This is an example of a redirected restore

- Create a VM volume on VM group PRODdg (as root)

```
# vxassist -g PRODdg make newdata1 500m
```

- Create a new VxFS file system and mount (as root)

```
# mkdir /newdata1
# mkfs -F vxfs /dev/vx/rdsk/PRODdg/newdata1
# mount -F vxfs /dev/vx/rdsk/PRODdg/newdata1 /newdata1
# chown -R db2inst1:db2iadm /newdata1
```

- Check on required settings and prepare new containers

```
# su - db2inst1
```

```
$ db2 connect to PROD
$ db2 list tablespaces
$ db2 update db cfg for PROD USING LOGRETAIN ON
$ db2 update db cfg for PROD USING USEREXIT ON
$ db2 terminate

$ cd /newdata1

$ /opt/VRTSvxfs/sbin/qiomkfile -s 200m data_container_001
```

- Backup the database and restore redirect into the new DMS devices (Quick I/O files)

```
$ db2 backup database PROD to /dump_device
  timestamp: 20010828121254

$ db2stop force
$ db2start
$ db2 restore db PROD from /dump_device taken at 20010828121254
  replace existing redirect
```

- Check for tablespaces state (restore pending, storage must/may be defined)

```
$ db2 connect to PROD
$ db2 list tablespaces show detail
```

- Issue a SET TABLESPACE CONTAINERS command for each tablespace whose containers must be redefined

  The `set tablespace containers` command converts one tablespace container. Repeat the `set tablespace containers` step as necessary for other tablespaces.

  For example, to redirect tablespace container 5 to DMS device data_container_001 under /newdata1 directory:

```
$ db2 set tablespace containers for 5 using (device '/newdata1/data_
$ db2 restore db PROD continue
$ db2 rollforward db PROD to end of logs
$ db2 rollforward db PROD complete
$ db2 connect to PROD
```

- Verify the conversion

```
$ db2 list tablespace containers for 5
```

An alternative migration strategy is to use the `db2move` command to export and import specific tables from a database. This command is used as follows:
For example, to migrate table cust using the db2move command:

```
$ db2move PROD export -tn cust
$ db2 connect to PROD
$ db2 drop table cust
$ db2 create table cust (cust_no char(6) not null, cust_name char(20) n
$ db2 terminate
$ db2move PROD load -l /data1
```

Chapter 9

Using Storage Checkpoints and Storage Rollback

This chapter includes the following topics:

About Storage Checkpoints and Storage Rollback

The Veritas Storage Checkpoint feature is available with the Enterprise Edition as part of the Veritas File System package and is used for the efficient backup and recovery of DB2 databases, including partitioned databases in an SMP environment. Storage Checkpoints can also be mounted, allowing regular file system operations to be performed. This chapter describes what Storage Checkpoints and storage rollback are and how to make use of these technologies through Veritas Storage Foundation.

The Storage Checkpoint facility is similar to the snapshot file system mechanism; however, a Storage Checkpoint persists after a system reboot. A Storage Checkpoint creates an exact image of a database instantly and provides a consistent image of the database from the point in time the Storage Checkpoint was created. The Storage Checkpoint image is managed and available through the GUI, or the Veritas Storage Foundation command line interface (CLI).

See the *Veritas Storage Foundation for DB2 Graphical User Interface Guide*.

A direct application of the Storage Checkpoint facility is Storage Rollback. Because each Storage Checkpoint is a consistent, point-in-time image of a file system, Storage Rollback is the restore facility for these on-disk backups. Storage Rollback rolls back changed blocks contained in a Storage Checkpoint into the primary file system for restoring the database faster.

For more information on Storage Checkpoints and Storage Rollback, see the *Veritas File System Administrator's Guide*.

How Storage Checkpoints and Storage Rollback work

A Storage Checkpoint is a disk and I/O efficient snapshot technology for creating a “clone” of a currently mounted file system (the primary file system). Like a snapshot file system, a Storage Checkpoint appears as an exact image of the snapped file system at the time the Storage Checkpoint was made. However, unlike a snapshot file system that uses separate disk space, all Storage Checkpoints share the same free space pool where the primary file system resides unless a Storage Checkpoint allocation policy is assigned. A Storage Checkpoint can be mounted as read-only or read-write, allowing access to the files as if it were a regular file system. A Storage Checkpoint is created using the `db2ed_ckptcreate` command or the GUI.

Initially, a Storage Checkpoint contains no data—it contains only the inode list and the block map of the primary fileset. This block map points to the actual data on the primary file system. Because only the inode list and block map are needed and no data is copied, creating a Storage Checkpoint takes only a few seconds and very little space.

A Storage Checkpoint initially satisfies read requests by finding the data on the primary file system, using its block map copy, and returning the data to the requesting process. When a write operation changes a data block "n" in the primary file system, the old data is first copied to the Storage Checkpoint, and then the primary file system is updated with the new data. The Storage Checkpoint maintains the exact view of the primary file system at the time the Storage Checkpoint was taken. Subsequent writes to block "n" on the primary file system do not result in additional copies to the Storage Checkpoint because the old data only needs to be saved once. As data blocks are changed on the primary file system,

the Storage Checkpoint gradually fills with the original data copied from the primary file system, and less and less of the block map in the Storage Checkpoint points back to blocks on the primary file system.

You can set a quota to limit how much space a file system will give to all storage checkpoints, to prevent the checkpoints from consuming all free space. See the command `db2ed_ckptquota` for more information.

Storage Rollback restores a database on the primary file systems to the point-in-time image created during a Storage Checkpoint. Storage Rollback is accomplished by copying the “before” images from the appropriate Storage Checkpoint back to the primary file system. As with Storage Checkpoints, Storage Rollback restores at the block level, rather than at the file level. Storage Rollback is executed using the `db2ed_ckptrollback` command.

Note: Whenever you change the structure of the database (for example, by adding or deleting containers), you must run `db2ed_update`. Also note if you delete a database, volume, or file system without updating the repository with `db2ed_update`, and then subsequently create a new database with the same name, the `db2ed_ckptdisplay_all`command displays the previous checkpoints.

Mountable Storage Checkpoints can be used for a wide range of application solutions, including backup, investigations into data integrity, staging upgrades or database modifications, and data replication solutions.

If you mount a Storage Checkpoint as read-write, the `db2ed_ckptrollback` command and GUI will not allow you to roll back to this Storage Checkpoint. Veritas Storage Foundation can no longer guarantee that the Storage Checkpoint data still represents the correct point-in-time image. A subsequent rollback from this Storage Checkpoint could corrupt the database. When a Storage Checkpoint is to be mounted as read-write, the `db2ed_ckptmount` command creates a “shadow” Storage Checkpoint and mounts it as read-write. This allows the database to still be rolled back to the original Storage Checkpoint.

Space requirements for Storage Checkpoints

To support storage rollback, the file systems need extra disk space to store the Storage Checkpoints. The extra space needed depends on how the Storage Checkpoints are used. Storage Checkpoints that are used to keep track of the block changes contain only file system block maps, and therefore require very little additional space (less than 1 percent of the file system size). Veritas Storage Foundation for DB2 provides a quota command (`db2ed_ckptquota`) to set space limits for all checkpoints of a given file system.

To support Storage Checkpoints and storage rollback, VxFS needs to keep track of the original block contents when the Storage Checkpoints were created. The additional space needed is proportional to the number of blocks that have been changed since a Storage Checkpoint was taken. The number of blocks changed may not be identical to the number of changes. For example, if a data block has been changed many times, only the first change requires a new block to be allocated to store the original block content. Subsequent changes to the same block require no overhead or block allocation.

If a file system that has Storage Checkpoints runs out of space or reaches its quota limit, by default VxFS removes the oldest Storage Checkpoint automatically instead of returning an `ENOSPC` error code (UNIX errno 28- No space left on device), which can cause the DB2 database to fail. Removing Storage Checkpoints automatically ensures the expected I/O semantics, but at the same time, eliminates a key recovery mechanism. You can set the Storage Checkpoint to be non-removable when you create it. In this case, an `ENOSPC` will return to DB2. The DBA will need to grow the VxFS file system size to work around the problem.

When restoring a file system that has data-full Storage Checkpoints from tape or other offline media, you need extra free space on the file system. The extra space is needed to accommodate the copy-on-write algorithm needed for preserving the consistent image of the Storage Checkpoints. The amount of free space required depends on the size of the restore and the number of Storage Checkpoints on the file system.

If you are restoring the entire file system, in most cases, you no longer need the existing Storage Checkpoint. You can simply re-make the file system using the `mkfs` command, and then restore the file system from tape or other offline media.

If you are restoring some of the files in the file system, you should first remove the data-full Storage Checkpoints that are no longer needed. If you have very limited free space on the file system, you may have to remove all data-full Storage Checkpoints in order for the restore to succeed.

Always reserve free disk space for growing volumes and file systems. You can also preallocate sufficient space for each file system when the file system is first created or manually grow the file system and logical volume where the file system resides.

See the `vxassist`(1) and `fsadm_vxfs`(1) manual pages for more information.

Performance of Storage Checkpoints

Veritas File System attempts to optimize the read and write access performance on both the Storage Checkpoint and the primary file system. Reads from a Storage Checkpoint typically perform at nearly the throughput of reads from a normal

VxFS file system, allowing backups to proceed at the full speed of the VxFS file system.

Writes to the primary file system are typically affected by the Storage Checkpoints because the initial write to a data block requires a read of the old data, a write of the data to the Storage Checkpoint, and finally, the write of the new data to the primary file system. Having multiple Storage Checkpoints on the same file system, however, will not make writes slower. Only the initial write to a block suffers this penalty, allowing operations like writes to the intent log or inode updates to proceed at normal speed after the initial write.

The performance impact of Storage Checkpoints on a database is less when the database files are Direct I/O files. A performance degradation of less than 5% in throughput has been observed in a typical OLTP workload when the Storage Checkpoints only keep track of changed information. For Storage Checkpoints that are used for Storage Rollback, higher performance degradation (approximately 10 to 20 percent) has been observed in an OLTP workload. The degradation should be lower in most decision-support or data warehousing environments.

Reads from the Storage Checkpoint are impacted if the primary file system is busy, because the reads on the Storage Checkpoint are slowed by all of the disk I/O associated with the primary file system. Therefore, performing database backup when the database is less active is recommended.

Storage Checkpoint allocation policies

The Veritas File System provides Multi-Volume File Systems (MVS) when used in conjunction with the Volumes Set feature in Veritas Volume Manager. A volume set is a container for multiple different volumes. MVS enables creation of a single file system over multiple volumes, each volume with properties of its own. This helps administrators specify which data goes on which volume types. For more details about MVS, see *Veritas Volume Manager Administrator's Guide*. Setting up a storage configuration for MVS operations is a system administrator's responsibility and requires superuser (`root`) privileges.

Multi-Volume File Systems provide a database administrator, through the checkpoint administration interface, the ability to create Storage Checkpoint Allocation Policies.

A Storage Checkpoint Allocation policy specifies a list of volumes and the order in which to attempt allocations. Once defined, a database administrator can use these policies to:

- Control where the storage checkpoint should be created, enabling separation of metadata and data of a storage checkpoint to different volumes.

- Separate storage checkpoints so that data allocated to a storage checkpoint is isolated from the primary file system. This helps control the space used by the checkpoint and prevents the checkpoint from fragmenting the space in the primary fileset.

When policies are assigned to a storage checkpoint, the database administrator must specify the mapping to both metadata and file data. If no policies are specified for the storage checkpoint, the data is placed randomly within the primary file system. Data and metadata of storage checkpoints can have different policies assigned to them or use the same policy to be applied to data and metadata. Multiple checkpoints can be assigned the same checkpoint allocation policy. A partial policy is also allowed; a partial policy means that the policy does not exist on all file systems used by the database.

Once the policy is assigned to checkpoints, the allocation mechanism attempts to satisfy the request from each device in the policy in the order the devices are defined. If the request cannot be satisfied from any of the devices in the policy, the request will fail, even if other devices exist in the file system which have space. Only those devices can provide allocation that are listed in the policy. This implementation is the mechanism for preventing allocation requests from using space in other devices which are not specified in the policy. It is recommended that you allocate sufficient space for the volumes defined in the Storage Checkpoint policy or update the policy to include additional volumes. This also helps in retaining the old Storage Checkpoints.

Once the assigned policy is deleted, the allocation for metadata and file data for subsequent requests of storage checkpoint will return to the no policy assigned state.

For VxFS file systems disk layout Version 7, the volumes on the VxFS Multi-Volume File System cvan be either one of these types: `dataonly` and `metadataok`. Only `metadataok` volumes can be used to store checkpoint metadata. By default, only the first volume that is being added to the VxVM volume set is a `metadataok` volume. This means only the first volume that is being added to the VxVM volume set can be specified in the `ckpt_metadata_policy` by default. Use the following file system command to change the default setting. To check the flags of each volume in a VxFS Multi-Volume File System, execute the following file system command as `root`:

```
/opt/VRTS/bin/fsvoladm queryflags mountpoint
```

To change a `dataonly` volume to a `metadataok` volume, execute the following file system command as `root`:

```
/opt/VRTS/bin/fsvoladm clearflags dataonly mountpoint vol-name
```

The following are usage notes for Storage Checkpoint allocation policies:

Usage notes

- Since the Storage Checkpoint allocation policies feature is associated with the MVS file system, it is available only on file systems using disk layout Version 6.
- Storage Checkpoint allocation policy requires VxVM Volume Set and VxFS Multi-Volume File Systems features to be enabled. These features are included in the Enterprise Edition of Storage Foundation.
  See the Multi-Volume File System chapter in the *Veritas File System Administrator's Guide* for creating Volume Sets and MVS file systems for the primary file systems used by the database datafiles.
- Data allocation is done by the volumes in the order that was assigned in the policy.
- The maximum length of a Storage Checkpoint allocation policy name is 64 characters.

Using Storage Checkpoint allocation policies

You can use the `db2ed_ckptpolicy` command to manage Storage Checkpoint allocation policies.

See "Creating and working with Storage Checkpoint allocation policies using db2ed_ckptpolicy" on page 290.

Note: You cannot administer Storage Checkpoint allocation policies through the the GUI.

The following are usage notes for the `db2ed_ckptpolicy` and `db2ed_ckptcreate` commands:

Usage notes

- See the `db2ed_ckptpolicy`(1M) and `db2ed_ckptcreate`(1M) manual pages for more information.
- The `db2ed_ckptpolicy` command needs to be executed by the DB2 database administrator.

In the following example, the file systems for database datafiles are set up as follows:

- Two MVS file systems `/mvsfs/v1` and `/mvsfs/v2` used for database datafiles.
- File system `/mvsfs/v1` is created on volume set `mvsvset1`.
- File system `/mvsfs/v2` is created on volume set `mvsvset2`.

- Volume set `mvsvset1` contains volumes `mvsv1`, `mvsv2`, and `mvsv3`.
- Volume set `mvsvset2` contains volumes `mvsv4` and `mvsv5`.

Use the `db2ed_ckptpolicy` command with the following options.

```
$ db2ed_ckptpolicy -D DB2DATABASE  [ -I DB2INSTANCE ] [-n] \
[-h] options
```

Where *`options`* could be any of the following parameters:

```
-o create|update|remove -p ckpt_sample
```

```
-o display [-c ckpt_name | -p ckpt_sample]
```

```
-o assign -c ckpt_name -p \
ckpt_data_policy[,ckpt_metadata_policy]
```

Creating a Storage Checkpoint allocation policy

Create a Storage Checkpoint allocation policy with the `-o create` option to `db2ed_ckptpolicy`.

To create a Storage Checkpoint allocation policy

- Use the `db2ed_ckptpolicy` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptpolicy -D DB2DATABASE \
 -o create -p ckpt_policy
```

Note: A partial policy indicates that the Storage Checkpoint allocation policy does not include all the file systems used by the database.

This example shows how to create a Storage Checkpoint allocation policy with the name `ckpt_sample` database for a database named `PROD`:

```
$ db2ed_ckptpolicy -D PROD -o create -p ckpt_sample
```

Output similar to the following is displayed.

```
File System: /mvsfs/v2 (MVS volumes: mvsv4,mvsv5)

Assigned Data Policy: NONE (MVS Volumes: N/A)

Assigned Meta Data Policy: NONE (MVS Volumes: N/A)

Please enter the volume name(s), separated by space, for the
 policy ckpt_sample [skip,quit]: mvsv4

File System: /mvsfs/v1 (MVS volumes: mvsv1,mvsv2,mvsv3)

Assigned Data Policy: NONE (MVS Volumes: N/A)

Assigned Meta Data Policy: NONE (MVS Volumes: N/A)

Please enter the volume name(s), separated by space, for the
policy ckpt_sample [skip,quit]: mvsv2

The following information will be used to create policy ckpt_sample

ckpt_sample        /mvsfs/v2        mvsv4

ckpt_sample        /mvsfs/v1        mvsv2
```

Assigning a Storage Checkpoint allocation policy

You can use either of the following methods to assign an allocation policy to a Storage Checkpoint:

- Use the `db2ed_ckptpolicy` command to assign allocation policies to an existing Storage Checkpoint.

- Use the -p *ckpt_data_policy*[,*ckpt_metadata_policy*] option to the db2ed_ckptcreate command to supply policies when executed.

Note: The db2ed_ckptcreate command automatically assigns the policies when the Storage Checkpoint is created.

The following procedure uses db2ed_ckptpolicy to assign an allocation policy to an existing Storage Checkpoint. This example uses PROD as the database name and Checkpoint_1096060202 as a sample Storage Checkpoint.

To assign an allocation policy to an existing Storage Checkpoint

1 Create an online Storage Checkpoint for database PROD.

```
$ db2ed_ckptcreate -D PROD -o online
```

As a result, Checkpoint_1096060202 is created.

2 Assign a Storage Checkpoint policy to the Checkpoint_1096060202.

```
$ db2ed_ckptpolicy -D PROD -n -o assign \
-c Checkpoint_1096060202 -p ckpt_data,ckpt_metadata
```

3 Display the details of the Storage Checkpoint allocation policy assigned to Checkpoint_1096060202.

```
$ db2ed_ckptpolicy -D PROD -n -o display \
-c Checkpoint_1096060202
```

Output similar to the following is displayed:

```
Storage Checkpoint    File System   Data Policy   Meta Data Policy
------------------    -----------   -----------   ----------------
Checkpoint_1096060202  /mvsfs/v2    ckpt_data     ckpt_metadata
Checkpoint_1096060202  /mvsfs/v1    ckpt_data     ckpt_metadata
```

To assign an allocation policy to a new Storage Checkpoint

1 Use `db2ed_ckpcreate` to assign an allocation policy to a new Storage Checkpoint.

```
$ db2ed_ckptcreate -D PROD -o online -p ckpt_data,ckpt_metadata
```

As a result, a Storage Checkpoint allocation policy is assigned to Checkpoint_1096060122, which is a new Storage Checkpoint.

2 Display the details of the Storage Checkpoint allocation policy assigned to checkpoint 1096060122.

```
$ db2ed_ckptpolicy -D PROD -n -o display \
-c Checkpoint_1096060122
```

Output similar to the following is displayed:

```
Storage Checkpoint    File System   Data Policy   Meta Data Policy
------------------    -----------   -----------   ----------------
Checkpoint_1096060122  /mvsfs/v2    ckpt_data     ckpt_metadata
Checkpoint_1096060122  /mvsfs/v1    ckpt_data     ckpt_metadata
```

Displaying a Storage Checkpoint allocation policy

To verify that the Storage Checkpoint allocation policy is correct, you can display the policy.

To display a Storage Checkpoint allocation policy

- Use the `-o display` option to list all the Storage Checkpoint allocation policies contained in the file systems used by the database.

```
$ db2ed_ckptpolicy -D DB2DATABASE -n -o display
```

Output similar to the following is displayed:

```
Policy Name           File System Coverage

------------------    --------------------

ckpt                  Complete

ckpt_data             Complete

ckpt_metadata         Complete

new_ckpt              Partial

ckpt_sample           Complete
```

Note: `Partial` in the File System Coverage column indicates that the Storage Checkpoint allocation policy does not include one or more of the file systems used by the database.

To display Storage Checkpoint allocation policy information

- Use the `-o display -p` *`checkpointpolicy_name`* option to display information related to a specific Storage Checkpoint allocation policy.

```
$ db2ed_ckptpolicy -D DB2DATABASE -n -o display \
 -p checkpointpolicy_name
```

Output similar to the following is displayed:

```
Policy Name        File System       MVS volumes

------------       -----------       -----------

ckpt_sample        /mvsfs/v2         mvsv4

ckpt_sample        /mvsfs/v1         mvsv2
```

To display the allocation policies assigned to a Storage Checkpoint

- Use the `-o display -c checkpoint_xxxxxxxxx` option to display the allocation policies assigned to the Storage Checkpoint.

  ```
  $ db2ed_ckptpolicy -D DB2DATABASE -n -o display \
  -c checkpoint_xxxxxxxxxx
  ```

  Output similar to the following is displayed:

  ```
  Storage Checkpoint      File System     Data Policy   Meta Data Policy

  ------------------      --------------  -----------   ---------------

  Checkpoint_1095125037   /mvsfs/v2       ckpt_data     ckpt_metadata

  Checkpoint_1095125037   /mvsfs/v1       ckpt_data     ckpt_metadata
  ```

Updating a Storage Checkpoint allocation policy

After creating a Storage Checkpoint allocation policy, you can make changes at a later time.

To update a Storage Checkpoint allocation policy

- Use the `-o update -p checkpoint_policy_name` option to update an allocation policy.

  This example shows how to update a Storage Checkpoint allocation policy named `ckpt_sample`:

  ```
  $ db2ed_ckptpolicy -D PROD DB2DATABASE -o update -p ckpt_sample
  ```

  Output similar to the following is displayed:

  ```
  File System: /mvsfs/v2 (MVS volumes: mvsv4,mvsv5)

  Policy: ckpt_sample (MVS volumes: mvsv4)

  Please enter the volume name(s), separated by space, for the
  policy ckpt_sample [skip,quit]: mvsv5

  File System: /mvsfs/v1 (MVS volumes: mvsv1,mvsv2,mvsv3)

  Policy: ckpt_sample (MVS volumes: mvsv2)

  Please enter the volume name(s), separated by space, for the
  policy ckpt_sample [skip,quit]: mvsv2,mvsv3

  The following information will be used to create policy ckpt_sample

  ckpt_sample       /mvsfs/v2      mvsv5

  ckpt_sample       /mvsfs/v1      mvsv2,mvsv3

  where mvsv4 is the volume currently using the allocation policy.
  ```

  The output displays the volumes that are currently assigned in the Storage Checkpoint allocation policy.

  You are prompted to enter any new volumes to which you would want to assign the Storage Checkpoint allocation policy.

Removing a Storage Checkpoint allocation policy

If a Storage Checkpoint allocation policy no longer meets your needs, you can remove the policy.

To remove a Storage Checkpoint allocation policy

- Use the following command to remove a Storage Checkpoint allocation policy:

```
$ db2ed_ckptpolicy -D DB2DATABASE -n -o remove \
-p checkpointpolicy_name
```

Converting from regular VxFS file system to MVS

The following procedure describes the conversion of a regular VxFS file system to MVS file system and optionally add new volume to it. Converting a regular VxFS to MVS requires superuser (`root`) privileges.

For further details on creating and administrating the VxVM Volume Sets and VxFS Multi-Volume Files System, refer to the *Veritas Volume Manager Administrator's Guide* and *Veritas File System Administrator's Guide*.

To convert from a regular VxFS file system to MVS

1. Select a non-MVS file system to convert to MVS and unmount it:

```
# umount /mnt1
```

2. Create the volume set:

```
# vxvset -g dgname make myvset old_vol
```

3. Mount the volume set:

```
# mount -F vxfs /dev/vx/dsk/dgname/myvset /mnt1
```

4. Upgrade the volume set's file system to Version 6 disk layout:

```
# vxupgrade -n 6 /mnt1
```

   See the *Veritas File System Installation Guide* and the `vxfsconvert(1M)` and `vxupgrade(1M)` manual pages for information on upgrading VxFS disk layouts:

5. Add the new volume to the volume set:

```
# vxvset -g dgname addvol myvset new_vol
```

6 Add the new volume to the file system. You must specify the size of the volume:

```
# fsvoladm add /mnt1 new_vol 2g
```

where `new_vol` is the name of the newly added volume and `2g` is the size of the volume:

7 Verify the new volume in the file system:

```
# fsvoladm list /mnt1
```

Partitioned databases and Version Checkpoints

Veritas Storage Foundation for DB2 provides commands for creating and using Storage Checkpoints in a DB2 Universal Database (UDB) partitioned database environment as well as an Enterprise Edition (EE) environment.

DB2 UDB Enterprise Server Edition (ESE) supports data partitioning across clusters of computers with a feature called Data Partition Feature (DPF). You can install DB2 ESE with multiple partitions on a single system or you can distribute it on multiple systems.

In a partitioned database environment, Storage Checkpoints that apply to multiple partitions in an SMP environment are referred to as Version Checkpoints. Veritas Storage Foundation for DB2 supports SMP environments. It does not support MPP environments.

Veritas Storage Foundation for DB2 commands that apply to multiple partitions end with the suffix `_all`. For example, `db2ed_ckptcreate_all` creates a Version Checkpoint for a partitioned DB2 database by calling `db2ed_ckptcreate` on every database partition.

Backing up and recovering the database using Storage Checkpoints

Storage Checkpoints can be created by specifying one of the following options: online or offline. To create a Storage Checkpoint with the online option, the instance should be online and you must enable the `LOGRETAIN`, `USEREXIT`, and/or `LOGARCHMETH1` database configuration parameters. For the offline option, the database should be offline.

During the creation of the Storage Checkpoint, the database is placed in suspended mode. You can roll back the entire database to an online or offline Storage

Checkpoint. After the rollback is complete, you may roll the database forward to restore the database if you have used an online Storage Checkpoint.

To allow the easiest recovery, always keep the `LOGRETAIN` and/or `USEREXIT` database configuration parameters enabled, regardless of whether the database is online or offline when you create Storage Checkpoints.

Verifying a Storage Checkpoint using the command line

After creating a Storage Checkpoint and before using it to back up or restore a database, you can verify that the Storage Checkpoint is free of errors.

The following are usage notes for verifying a Storage Checkpoint:

Usage notes

- See the `db2ed_ckptcreate`(1M) and `db2ed_ckptmount`(1M) manual pages for more information.
- For a database environment with multiple partitions in an SMP environment, see the `db2ed_ckptcreate_all`(1M) and `db2ed_ckptmount_all`(1M) manual pages for more information.

To verify that a Storage Checkpoint is error-free using the command line

1 Create and mount a Storage Checkpoint:

```
$ /opt/VRTS/bin/db2ed_ckptcreate -I db2inst -D PROD -o online

Creating online Storage Checkpoint of database PROD.

Storage Checkpoint Checkpoint_903937870 created.

$ mkdir /tmp/ckpt_ro

$ /opt/VRTS/bin/db2ed_ckptmount -I db2inst -D PROD \
-c Checkpoint_903937870 -m /tmp/ckpt_ro
```

If the specified mount point directory does not exist, then `db2ed_ckptmount` creates it before mounting the Storage Checkpoint, as long as the DB2 instance owner has permission to create it.

2 Examine the contents of the Storage Checkpoint:

```
$ ls -l /tmp/ckpt_ro/db2vol_82/db2inst1
drwxr-xr-x 3 db2inst1 dba 1024   Nov 11 2000 .
drwxr-xr-x 3 db2inst1 dba 512    Nov 16 11:00 ..
-rw-r--r-- 1 db2inst1 dba 209747968    Nov 16 10:58 .tstmp
-rw-r--r-- 1 db2inst1 dba 209747968    Nov 16 10:58 .tstab
lrwxrwxrwx 1 db2inst1 dba 18 Nov 11 2000 tstmp -> .tstmp::cdev:vxfs:
lrwxrwxrwx 1 db2inst1 dba 18 Nov 11 2000 tstab -> .tstab::cdev:vxfs:
```

Storage Checkpoints can only be used to restore from logical errors (for example, a human error). Because all the data blocks are on the same physical device, Storage Checkpoints cannot be used to restore files due to a media failure. A media failure requires a database restore from a tape backup or a copy of the database files kept on a separate medium. The combination of data redundancy (disk mirroring) and Storage Checkpoints is recommended for highly critical data to protect them from both physical media failure and logical errors.

Backing up using a Storage Checkpoint

You can back up a database by creating a Storage Checkpoint using the `db2ed_ckptcreate` command, mount the Storage Checkpoint as read-only using the `db2ed_ckptmount` command, and then back it up using tools such as `tar` or `cpio`.

In a DB2 ESE SMP environment, you can use `db2ed_ckptcreate_all` to create a checkpoint for a partitioned DB2 database. You can use `db2ed_ckptmount_all` to mount all Storage Checkpoints for a checkpoint for a partitioned DB2 database.

The following are usage notes for backing up a database with the `db2ed_ckptcreate` command:

Usage notes

- See the `db2ed_ckptcreate`(1M), `db2ed_ckptmount`(1M), `tar`(1), and `cpio`(1) manual pages for more information.
- For a database with multiple partitions in an SMP environment, see the `db2ed_ckptcreate_all`(1M) and `db2ed_ckptmount_all`(1M) manual pages for more information.

To back up a frozen database image using the command line

1 Assuming all the database datafiles in this example reside on one VxFS file system named `/db01`, create a Storage Checkpoint using the `db2ed_ckptcreate` command:

```
$ /opt/VRTS/bin/db2ed_ckptcreate -I db2inst -D PROD -o online

Creating online Storage Checkpoint of database PROD.
Storage Checkpoint Checkpoint_903937870 created.
```

2 Mount the Storage Checkpoint using the `db2ed_ckptmount` command:

```
$ /opt/VRTS/bin/db2ed_ckptmount -I db2inst -D PROD \
-c Checkpoint_903937870 -m /tmp/ckpt_ro
```

If the specified mount point directory does not exist, then `db2ed_ckptmount` creates it before mounting the Storage Checkpoint, as long as the DB2 instance owner has permission to create it.

3 Use `tar` to back up the Storage Checkpoint:

```
$ cd /tmp/ckpt_ro
$ ls
db01
$ tar cvf /tmp/PROD_db01_903937870.tar ./db01
```

Recovering a database using a Storage Checkpoint

Because Storage Checkpoints record the before images of blocks that have changed, you can use them to do a file-system-based storage rollback to the exact time when the Storage Checkpoint was taken. You can consider Storage Checkpoints as backups that are online, and you can use them to roll back an entire database. Rolling back to or restoring from any Storage Checkpoint is generally very fast because only the changed data blocks need to be restored.

Suppose a user deletes a table by mistake right after 4:00 p.m., and you want to recover the database to a state just before the mistake. You created a Storage Checkpoint (`Checkpoint_903937870`) while the database was running at 11:00 a.m., and you have the `LOGRETAIN` and/or `USEREXIT` database configuration parameters enabled.

To recover the database using a Storage Checkpoint

1. Ensure that the affected database is inactive, and use Storage Rollback to roll back the database from the Storage Checkpoint you created at 11:00 a.m.:

   ```
   $ /opt/VRTS/bin/db2ed_ckptrollback -I db2inst1 -D PROD \
   -c Checkpoint_903937870
   ```

   In a DB2 UDB EEE or ESE environment, `db2ed_ckptrollback_all` can be used to roll back a partitioned DB2 database to a checkpoint. The `db2ed_ckptrollback_all` command calls `db2ed_ckptrollback` on every partition.

2. Start up the instance if it is down:

   ```
   $ db2start
   ```

3. To re-apply archive logs to the point before the table was deleted to recover the database to 4:00 p.m, enter:

   ```
   $ db2 rollforward database PROD to isotime
   ```

   where `ISOTIME` is the point in time where all committed transactions are to be rolled forward. The time must be expressed in local time or Coordinated Universal Time (UTC). The UTC format is `yyyy-mm-dd-hh.mm.ss.nnnnnn` (year, month, day, hour, minutes, seconds, microseconds).

4. To complete the roll forward, enter:

   ```
   $ db2 rollforward database PROD complete
   ```

Cloning the DB2 database using db2ed_clonedb

You can use the `db2ed_clonedb` command to clone a DB2 database using a Storage Checkpoint.

Cloning an existing database using a Storage Checkpoint must be done on the same host.

You have the option to manually or automatically recover the database when using the `db2ed_clonedb` command:

- Manual (Interactive) recovery, which requires using the `-i` option, of the clone instance allows the user to control the degree of recovery by specifying which archive log files are to be replayed.
- Automatic (non-interactive) recovery, which is the default usage of the command, recovers the entire database and replays all of the archive logs. You will not be prompted for any archive log names.

Before cloning the DB2 database, the following conditions must be met:

Prerequisites

- You must first create a Storage Checkpoint.
  See "Creating Storage Checkpoints using db2ed_ckptcreate" on page 282.
- Before using `db2ed_clonedb` to clone a database either within the same instance or across instances, ensure that your locale is the same as the database locale and that the locale is the same across instances. If you do not know what locale to set, refer to the file `/opt/VRTSdb2ed/lib/DB2CODPAGE.tbl` for a mapping between the locale and the database Code page.
- Make sure you have enough space and system resources to create a clone instance on your system.
- A clone database takes up as much memory and machine resources as the primary database.

Usage notes

- The `db2ed_clonedb` command is used to create a copy of a database, cloning all existing database files to new locations.
- The `db2ed_clonedb` command only works when the instance is up.
- It is assumed that the user has a basic understanding of the database recovery process.
- When cloning a database using `db2ed_clonedb`, any database within the target instance that has the same name as the source database to be cloned will be temporarily uncataloged and therefore unavailable. If the source database is being cloned within the same instance, it will be temporarily uncataloged while `db2ed_clonedb` is running. The database will be recataloged on completion of `db2ed_clonedb`.
- See the `db2ed_clonedb`(1M) manual page for more information.

Options for the `db2ed_clonedb` command are:

Table 9-1 db2ed_clonedb command options

Option	Description
`-I SOURCE_INSTANCE`	Specifies the source DB2 database. If the instance is not specified here, it is set as the current user.
`-S SOURCE_DATABASE`	Specifies the name of the source DB2 database.
`-T TARGET_DATABASE`	Specifies the name of the new DB2 database that will be created.
`-c CKPT_NAME`	Indicates the name of the Storage Checkpoint to use for creating the new database. The Storage Checkpoint is mounted automatically during the cloning process.
`-m MOUNT_POINT`	Indicates the location of the database containers to be mounted.
`-l`	Requires the argument `TARGET_DATABASE_REDOLOG_DIRECTORY`. Specifies the redo log directory of the target database.
`-i`	Runs the command in interactive mode where you must respond to prompts by the system. The default mode is non-interactive. (Optional)
`-a`	Requires the argument `RECOVERY_LOG_LOCATION`. If this option is specified, a minimal database recovery will occur automatically after the clone is created. (Optional)
`-o umount`	Shuts down the clone database and unmounts the Storage Checkpoint file system.
`-o restartdb`	Mounts the Storage Checkpoint file system and starts the clone database. The `-o restartdb` option will not attempt to recover the clone database.
`-d`	Used with the `-o umount` option. If the `-d` option is specified, the Storage Checkpoint used to create the clone database will be removed along with the clone database.

To clone a DB2 database with manual DB2 recovery

- Use the `db2ed_clonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -S source_db -T target_db1 \
-c Checkpoint_1049927758 -m /db2clone/target_db1

Clone database succeeded.
```

To clone a DB2 database with automatic DB2 recovery

- Use the `db2ed_clonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -S source_db -T target_db2 \
-c Checkpoint_1049927758 -m /db2clone/target_db2 -a \
/db2clone/db2inst1/NODE0000/SQL00002/SQLOGDIR/

Clone database succeeded.

The database will be rolled forward to 2003-04-09-22.36.02.0000.

/db2clone/db2inst1/NODE0000/SQL00002/SQLOGDIR/

will be used to be searched for archived logs during recovery.

Rollforward Status

Input database alias          = TARGET_DB2

Number of nodes have returned status = 1

Node number          = 0

Rollforward status          = DB  working

Next log file to be read          = S0000002.LOG

Log files processed          =  -

Last committed transaction          = 2003-04-08-20.32.53.000000

DB20000I  The ROLLFORWARD command completed successfully.

Rollforward Status

Input database alias          = TARGET_DB2

Number of nodes have returned status = 1

Node number          = 0

Rollforward status          = not pending

Next log file to be read          =

Log files processed          = S0000002.LOG - S0000002.LOG

Last committed transaction          = 2003-04-08-20.32.53.000000

DB20000I  The ROLLFORWARD command completed successfully.
```

To clone a DB2 database with interactive DB2 recovery

- Use the `db2ed_clonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -S source_db -T target_db3 \
-c Checkpoint_1049927758 -m /db2clone/target_db3 -i
/db2clone/db2inst1/NODE0000/SQL00002/SQLOGDIR/ will be used to
be searched for archived logs during recovery.

Press <Return> to continue
Or <r> to retry
Or <q> to quit:

The database will be rolled forward to 2003-04-09-22.36.02.0000.
The archived logs will be retrieved from
/db2clone/db2inst1/NODE0000/SQL00002/SQLOGDIR/.
The estimated minimum logs are S0000002.LOG-S0000002.LOG.

You can retrieve all log files by executing
cp /db2clone/db2inst1/NODE0000/SQL00002/SQLOGDIR/*.LOG
/db2clone/target_db3/REDOLOG
from another session and then

Press q to continue

Or Press Enter to retrieve the minimum logs.

The recovery process will stop if more logs is required.

Press <Return> to continue ...
   or <q> to skip ...

Rollforward Status

Input database alias          = TARGET_DB3
Number of nodes have returned status = 1
Node number          = 0
Rollforward status          = DB  working
Next log file to be read          = S0000002.LOG
Log files processed          =  -
Last committed transaction          = 2003-04-08-20.32.53.000000

DB20000I  The ROLLFORWARD command completed successfully.

Rollforward Status
```

```
Input database alias          = TARGET_DB3
Number of nodes have returned status = 1
Node number          = 0
Rollforward status          = not pending
Next log file to be read          =
Log files processed          = S0000002.LOG - S0000002.LOG
Last committed transaction          = 2003-04-08-20.32.53.000000

DB20000I  The ROLLFORWARD command completed successfully.
```

To shut down the clone database and unmount the Storage Checkpoint

- Use the `db2ed_clonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -T target_db3 -o umount
```

To mount a Storage Checkpoint file system and start the clone database

- Use the `db2ed_clonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -T target_db3 -o restartdb
```

To delete a clone database and the Storage Checkpoint used to create it

- Use the `db2ed_clonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -T target_db3 -o umount -d
```

Guidelines for DB2 recovery

For optimal DB2 recovery, follow these guidelines:

- Ensure that the `LOGRETAIN` and/or `USEREXIT` database configuration parameters are enabled. Enabling these parameters makes the following roll-forward recovery techniques available for use:
  - Point in time recovery using the `ROLLFORWARD DATABASE` command
  - Online processing of `BACKUP` and `RESTORE DATABASE` commands

    When `LOGRETAIN` is enabled, log files are not overwritten when they become inactive. When `USEREXIT` is enabled, DB2 will call the user exit program when a log file is ready for archiving.
- You must have `SYSADM`, `SYSCTRL`, or `SYSMAINT` authority to use the `ROLLFORWARD DATABASE` command.

- A database must be restored successfully (using the `RESTORE DATABASE` command) before it can be rolled forward.
- After Storage Rollback, perform DB2 recovery, applying some or all of the archived redo logs.

To perform a complete recovery, use:

- `db2rollforward database database_name to end of logs`

To perform a point-in-time recovery, use:

- `db2rollforward database database_name to PIT`

  Where *PIT* is the point in time you are rolling forward to. The point in time must be specified in UTC or local time.

  An example of how to use the `db2rollforward` command using local time:

  ```
  db2rollforward database db to yyyy-mm-dd-hh.mm.ss.nnnnnn using local ti
  ```

To complete the recovery, use:

- `db2rollforward database database_name complete`

See your DB2 documentation for complete information on recovery.

Using the GUI to perform Storage Checkpoint-related operations

You can use the GUI to create Storage Checkpoints and then roll back an entire database using any of the previously created Storage Checkpoints.

See the *Veritas Storage Foundation for DB2 Graphical User Interface Guide*.

Chapter 10

Using Database Dynamic Storage Tiering

This chapter includes the following topics:

About Database Dynamic Storage Tiering

More and more data is being retained. Eventually, some of the data is no longer needed as frequently, but still takes up a large amount of disk space. Database Dynamic Storage Tiering matches data storage with the data's usage requirements so that data is relocated based on requirements determined by the database administrator (DBA). The feature enables you to manage your data so that less-frequently used data can be moved to slower, less expensive disks, allowing frequently-accessed data to be stored on the faster disks for quicker retrieval. Storage classes are used to designate which disks make up a particular tier.

There are two common ways of defining storage classes:

- Performance, or storage, cost class: The most-used class consists of fast, expensive disks. When data is no longer needed on a regular basis, the data

can be moved to a different class that is made up of slower, less expensive disks.

- Resilience class: Each class consists of non-mirrored volumes, mirrored volumes, and n-way mirrored volumes.
  For example, a database is usually made up of data, an index, and logs. The data could be set up with a three-way mirror because data is critical. The index could be set up with a two-way mirror because the index is important, but can be recreated. The logs are not required on a daily basis and could be set up with no mirroring.

Dynamic Storage Tiering (DST) policies control initial file location and the circumstances under which existing files are relocated. These policies cause the files to which they apply to be created and extended on specific subsets of a file systems's volume set, known as placement classes. The files are relocated to volumes in other placement classes when they meet specified naming, timing, access rate, and storage capacity-related conditions.

In addition to preset policies, you can manually move files to faster or slower storage, when necessary. You can run reports that list active policies, display file activity, display volume usage, or show file statistics.

Database Dynamic Storage Tiering building blocks

To use Database Dynamic Storage Tiering, your storage must be managed using the following features:

- VxFS multi-volume file system
- VxVM volume set
- Volume tags
- Dynamic Storage Tiering policies

About VxFS multi-volume file systems

Multi-volume file systems are file systems that occupy two or more virtual volumes. The collection of volumes is known as a volume set, and is made up of disks or disk array LUNs belonging to a single Veritas Volume Manager (VxVM) disk group. A multi-volume file system presents a single name space, making the existence of multiple volumes transparent to users and applications. Each volume retains a separate identity for administrative purposes, making it possible to control the locations to which individual files are directed. This feature is available only on file systems using disk layout Version 6 or later.

See "Converting a VxFS file system to a VxFS multi-volume file system" on page 164.

See the `fsvoladm`(1M) manual page.

About VxVM volume sets

Volume sets allow several volumes to be represented by a single logical object. Volume sets cannot be empty. All I/O from and to the underlying volumes is directed via the I/O interfaces of the volume set. The volume set feature supports the multi-volume enhancement to Veritas File System (VxFS). This feature allows file systems to make best use of the different performance and availability characteristics of the underlying volumes. For example, file system metadata could be stored on volumes with higher redundancy, and user data on volumes with better performance.

About volume tags

You make a VxVM volume part of a placement class by associating a volume tag with it. For file placement purposes, VxFS treats all of the volumes in a placement class as equivalent, and balances space allocation across them. A volume may have more than one tag associated with it. If a volume has multiple tags, the volume belongs to multiple placement classes and is subject to allocation and relocation policies that relate to any of the placement classes. Multiple tagging should be used carefully.

A placement class is a Dynamic Storage Tiering attribute of a given volume in a volume set of a multi-volume file system. This attribute is a character string, and is known as a volume tag.

About Dynamic Storage Tiering policies

Dynamic Storage Tiering allows administrators of multi-volume VxFS file systems to manage the placement of files on individual volumes in a volume set by defining placement policies that control both initial file location and the circumstances under which existing files are relocated. These placement policies cause the files to which they apply to be created and extended on specific subsets of a file system's volume set, known as placement classes. The files are relocated to volumes in other placement classes when they meet the specified naming, timing, access rate, and storage capacity-related conditions.

Database Dynamic Storage Tiering in a High Availability (HA) environment

Veritas Cluster Server does not provide a bundled agent for volume sets. If issues arise with volumes or volume sets, the issues can only be detected at the DiskGroup and Mount resource levels.

The DiskGroup agent brings online, takes offline, and monitors a Veritas Volume Manager (VxVM) disk group. This agent uses VxVM commands. When the value of the StartVolumes and StopVolumes attributes are both 1, the DiskGroup agent onlines and offlines the volumes during the import and deport operations of the disk group. When using volume sets, set StartVolumes and StopVolumes attributes of the DiskGroup resource that contains the volume set to 1. If a file system is created on the volume set, use a Mount resource to mount the volume set.

The Mount agent brings online, takes offline, and monitors a file system or NFS client mount point.

If you are using any of the Database Dynamic Storage Tiering commands in a high availability (HA) environment, the time on each system in the cluster must be synchronized. Otherwise, the scheduled task may not be executed at the expected time after a service group failover.

For more information, see the *Veritas Cluster Server Bundled Agents Reference Guide*.

Configuring Database Dynamic Storage Tiering

To use Database Dynamic Storage Tiering, you must:

- Review Database Dynamic Storage Tiering command requirements.
- Define database parameters.
- Set up storage classes.
- Convert an existing VxFS database file system to a VxFS multi-volume file system for use with Database Dynamic Storage Tiering.
- Classify, or tag, volumes so that the tags indicate the quality of the underlying disk.
- Display the free space on each class.
- Add or remove volumes as necessary.

Database Dynamic Storage Tiering command requirements

Before defining your database parameters, review the following command requirements:

- A DB2 database must be up and running.
- Only the DB2 instance owner can run Database Dynamic Storage Tiering commands.

- Run the `db2ed_update` command before running any of the Database Dynamic Storage Tiering commands. You should also run the `db2ed_update` command if any of the database files change.
  Because the Database Dynamic Storage Tiering commands retrieve database information from the repository, the repository must be up to date.
- Define the `LD_LIBRARY_PATH` environment variable as follows:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/VRTSdbms3/lib; \
export LD_LIBRARY_PATH
```

- To display Veritas Storage Foundation for DB2 messages with the correct tag, define the `SFUA_DB2` environment variable as follows:

```
SFUA_DB2=DB2; export SFUA_DB2
```

- If you are using any of the Database Dynamic Storage Tiering commands in a high availability (HA) environment, the time on each system in the cluster must be synchronized.
- Create the volumes that you want to add to the multi-volume file system in the same disk group as the file system volume. As root, use the following command to change the owner of each volume:

```
# /opt/VRTS/bin/vxedit -g disk_group \
set user=DB2INSTANCE volume
```

- Change the owner of the mount point on which you want to implement Database Dynamic Storage Tiering to the DB2 instance owner.

Defining database parameters

Running the `dbdst_admin` command defines parameters for the entire database. You must run this command at least once to define the database parameters for Database Dynamic Storage Tiering. Three pre-defined storage classes will be created (PRIMARY, SECONDARY, and BALANCE). Parameter values are stored in the SFDB repository.

Set at least one of the parameters in `maxclass`, `minclass`, `statinterval`, `sweeptime`, `sweepinterval`, `purgetime`, or `purgeinterval`, to enable default values. Add at least one class to enable the default classes.

Table 10-1 lists the options for the `dbdst_admin` command.

Table 10-1 dbdst_admin command options

Option	Description
`-D DB2DATABASE`	Specifies the name of the DB2 database.
`-I DB2INSTANCE`	Optional parameter that specifies the name of the DB2 instance.
`list`	Lists all the Database Dynamic Storage Tiering parameters of the database, including class name and description. This option should be used exclusively from the other options.
`maxclass=`	Maximum number of storage classes allowed in the database. Default value is 4.
`minclass=`	Minimum number of storage classes allowed in the database. Default value is 2.
`sweepinterval=`	Interval for file sweeping for file relocation. Default value is 1, which means one per day. If this value is set to 0, all scheduled sweep tasks will become unscheduled.
`sweeptime=`	Time per day for the file sweep to take place. Times are entered in 24-hour periods and should list hour: minute. For example, 8:30 AM is represented as 08:30 and 10:00 PM is represented as 22:00. Default value is 22:00.
`statinterval=`	Interval in minutes for gathering file statistics. Default value is 30, which represents every 30 minutes. If this value is set to 0, all scheduled tasks will become unscheduled.
`purgeinterval=`	Number of days after which the file statistics in the repository will be summarized and purged. Default value is 30. It is recommended that you set your purge interval sooner because you will not be able to view any statistics until the first 30-day interval is over, if you use the default.
`purgetime=`	Time per day for the file purge to take place. Times are entered in 24-hour periods and should list hour: minute. For example, 8:30 AM is represented as 08:30 and 8:00 PM is represented as 20:00. Default value is 20:00.
`addclass=`	Parameter that allows you to add a class to a database. The information should be entered as *`class:"description"`*, where *`class`* represents the class name and *`description`* is a string of up to 64 characters enclosed by double quotes used to describe the class.

Table 10-1 dbdst_admin command options *(continued)*

Option	Description
`rmclass=`	Parameter that allows you to remove a class from a database. Enter the class name as it appears in the database.

Note: If you do not want to change specific default values, you can omit those parameters when you run the `dbdst_admin` command. You only need to enter the parameters that need to be changed.

Before you define database parameters, review the following information:

To define database parameters

- Use the `dbdst_admin` command as follows:

```
$ /opt/VRTS/bin/dbdst_admin -D DB2DATABASE [-I DB2INSTANCE] -o list,\
maxclass=number,minclass=number,sweepinterval=interval,\
sweeptime=hh:mm,statinterval=interval,purgeinterval=interval,\
purgetime=hh:mm,addclass=class:"description",rmclass=class
```

For example, to add a class called tier1 for database PROD, and to set up a purge interval of one, meaning that the file statistics will be gathered for one day and then summarized and purged, use the `dbdst_admin` command as follows:

```
$ /opt/VRTS/bin/dbdst_admin -D PROD -o addclass=tier1:"Fast Storage",\
purgeinterval=1
```

Setting up storage classes

When you define your database parameters, three pre-defined storage classes are created. You will need to add or remove storage classes to fit your needs.

Adding storage classes

In addition to the default storage classes, you can add storage classes to better manage your data.

Before adding a storage class, review the following information:

To add a storage class

- Use the `dbdst_admin` command as follows:

```
$ /opt/VRTS/bin/dbdst_admin -D DB2DATABASE -o addclass=class:\
"description"
```

For example, to create a storage class named "FAST" for an EMC array, use the `dbdst_admin` command as follows:

```
$ /opt/VRTS/bin/dbdst_admin -D DB2DATABASE -o addclass=FAST:\
"fast EMC array"
```

Removing storage classes

If you no longer require a specific storage class, you can remove it. You cannot remove the pre-defined storage classes (PRIMARY, SECONDARY, and BALANCE).

Before removing a storage class, review the following information:

To remove a storage class

- Use the `dbdst_admin` command as follows:

```
$ /opt/VRTS/bin/dbdst_admin -D DB2DATABASE rmclass=class
```

For example, to remove a storage class called "SLOW," use the `dbdst_admin` command as follows:

```
$ /opt/VRTS/bin/dbdst_admin -D DB2DATABASE rmclass=SLOW
```

Displaying storage classes

You can display a list of Database Dynamic Storage Tiering properties and storage classes using the `dbdst_admin` command.

Before displaying your storage classes, review the following information:

To display existing storage classes and properties

- Use the `dbdst_admin` command as follows:

```
$ /opt/VRTS/bin/dbdst_admin -D DB2DATABASE -o list
```

Converting a VxFS file system to a VxFS multi-volume file system

To convert your existing VxFS file system to a VxFS multi-volume file system, you must convert a single volume to a volume set.

Converting a single volume to a volume set

When you convert to a volume set using the `dbdst_convert` command, the original volume will be renamed to a new volume name. The mount device name will become the new volume set name. Creating the new volume set name with the mount device name nullifies the need to rename the mount device in various locations.

Before converting to a volume set, make sure the following conditions have been met:

Prerequisites
- The DB2 database must not be active.
- Create at least one additional volume.

Usage notes
- You must convert the single-volume file system on which you plan to implement Database Dynamic Storage Tiering.
- The file system can be mounted or unmounted when you run the `dbdst_convert` command.
- If the file system has n volumes, volumes 1 through n–1 will be placed in the storage class "PRIMARY" and volume n will be placed in the storage class "SECONDARY."
- The volumes specified when running the conversion must be in the same disk group as the mount device.

To convert a mount device from a single volume device to a volume set

1 Use the `dbdst_convert` command as follows:

```
$ /opt/VRTS/bin/dbdst_convert -D DB2DATABASE [-I DB2INSTANCE] \
-M mount_device -v volume_name,volume_name
```

2 Bring database objects online.

For example, to convert a volume-based db2data file system to a Database Dynamic Storage Tiering-ready volume set file system on mount device /dev/vx/dsk/db2dg/db2data, use the `dbdst_convert` command as follows:

```
$ /opt/VRTS/bin/dbdst_convert -D DB2DATABASE -M /dev/vx/dsk/db2dg/db2data \
-v new_vol1,new_vol2
```

After conversion, you will have a volume set named db2data containing three volumes (db2data_b4vset, new_vol1, and new_vol2). The file system will have two storage classes defined as PRIMARY and SECONDARY. The volumes will be assigned as follows:

- PRIMARY storage class will contain volumes db2data_b4vset and new_vol1.
- SECONDARY storage class will contain volume new_vol2.

Classifying volumes into a storage class

Before creating a DST policy or manually moving data, assign classes to your volumes.

Before assigning classes to volumes, review the following information:

Usage notes

- You must convert your VxFS file system to a multi-volume file system first.
- Storage classes must be registered using the `dbdst_admin` command before assigning classes to volumes.
- The database can be online or offline.

To classify a volume

- Use the `dbdst_classify` command as follows:

```
$ /opt/VRTS/bin/dbdst_classify -D DB2DATABASE -M mount_device \
-v volume_name:class[,volume_name:class]
```

For example, to assign the class "FAST" to volume new_vol1, use the `dbdst_classify` command as follows:

```
$ /opt/VRTS/bin/dbdst_classify -D DB2DATABASE -M /dev/vx/dsk/db2dg/db2data \
-v new_vol1:FAST
```

Displaying free space on your storage classes

To see the free space, class information, and volume information on your storage classes, use the `dbdst_show_fs` command.

Table 10-2 on page 166. shows the `dbdst_show_fs` command options.

Table 10-2 dbdst_show_fs command options

Option	Description
`-D DB2DATABASE`	Specifies the name of the DB2 database.
`-I DB2INSTANCE`	Optional parameter that specifies the name of the DB2 instance.
`-o volume`	Displays the free space on volumes in each class.
`-m`	Specifies the mount point.

Before displaying the free space on a storage class, review the following information:

Prerequisites
- Make sure the file system is mounted.

Usage notes
- See the dbdst_show_fs(1M) manual page.

To display the free space on a storage class

- Use the dbdst_show_fs command as follows:

```
$ /opt/VRTS/bin/dbdst_show_fs -D DB2DATABASE [-I DB2INSTANCE] \
-o volume -m mount_point
```

Adding new volumes to a storage class

You can use the dbdst_addvol command to add volumes to a volume set.

Before adding a volume, review the following information:

Usage notes
- The database must be inactive when adding volumes to a storage class.

To add a volume to a volume set

- Use the dbdst_addvol command as follows:

```
$ /opt/VRTS/bin/dbdst_addvol -D DB2DATABASE [-I DB2INSTANCE] \
-M mount_device -v volume_name:class[,volume_name:class]
```

Removing volumes from a storage class

You may need to remove a volume from a volume set. To remove a volume, use the dbdst_rmvol command.

Before removing a volume, review the following information:

Usage notes
- The database must be inactive when removing volumes from a storage class.
- Only a volume that does not contain any file system data can be removed

To remove a volume from a volume set

- Use the dbdst_rmvol command as follows:

```
$ /opt/VRTS/bin/dbdst_rmvol -D DB2DATABASE [-I DB2INSTANCE] \
-M mount_device -v volume_name[,volume_name]
```

Dynamic Storage Tiering policy management

You can choose to manually relocate files or tablespaces, or you can use a preset Dynamic Storage Tiering (DST) policy.

Relocating files

You can relocate archive logs, datafiles, and external files if the files are no longer being used frequently. For example, if you maintain seasonal data, you may choose to move the files when you are in different seasons.

Table 10-3 on page 168. shows the `dbdst_file_move` command options.

Table 10-3 dbdst_file_move command options

Option	Description
`-o external | datafile`	Specifies whether to move external files or datafiles. Use this option with the `-f` option.
`-o archive1 | archive2`	Specifies which archive logs to move. Do not use this option with the `-f` option. This is supported by DB2 8.2 or later.
`-f` *`listfile`*	Specifies a listfile that contains a list of files or directories to be moved.
`-c` *`class`*`[:days]`	Specifies the storage class to which the files should be moved. If the *`days`* option is used, the files will be moved to the class specified if they have not been accessed in the number of days specified. Do not specify *`days`* if you are using the `-o datafile` option.
`-R`	Removes the policy for the specified object.

Before relocating a file, review the following information:

Prerequisites
- The database must be up when you run this command.

Usage notes
- Multiple partitions cannot reside on the same tablespace.

To relocate a file

- Use the `dbdst_file_move` command as follows:

```
/opt/VRTS/bin/dbdst_file_move -D DB2DATABASE -o datafile \
-f listfile -c storage_class:days [-c storage_class:days]
```

Relocating tablespaces

Use the `dbdst_tbs_move` command to move tablespaces to the desired storage class. The command queries the SFDB repository for the tablespace file names, then performs a one-time move based on your immediate requirements.

To relocate a tablespace

- Use the `dbdst_tbs_move` command as follows:

```
$ /opt/VRTS/bin/dbdst_tbs_move -D DB2DATABASE [-I DB2INSTANCE] \
-t tablespace -c class
```

  where

  - *`tablespace`* indicates which tablespace to move.
  - *`class`* indicates to which class the tablespace should be moved.

Using preset policies

Use the `dbdst_preset_policy` command to set a policy based on file name patterns before the files are created.

Table 10-4 on page 169. shows the dbdst_preset_policy command options.

Table 10-4 dbdst_preset_policy command options

Option	Description
`-d` *`directory`*	Indicates the directory on which the placement policy will be applied.
`-o sms | ats`	Specifies file placement for a SMS tablespace or DB2 automatic storage path.
`-s`	Displays the most commonly used file name patterns for DB2 SMS tablespaces or DB2 automatic storage paths. Use this option with the `-o sms | ats` option.
`-t` *`tablespace`*	Specifies the SMS tablespace when using the `-o sms` option.
`-e`	Enforces the file system of the specified directory. Use this option if there was an error in the previous enforcement that has been corrected and needs to be enforced again.
`-R`	Removes all pattern-based placement policies related to this directory.

Table 10-4 dbdst_preset_policy command options *(continued)*

Option	Description
`-l`	Lists the existing file placement that is set to the specified directory.
`-P pattern_spec`	Specifies file patterns and class assignment. This option will automatically place files in the desired class as soon as they are created. Existing files and newly created files will be moved immediately to the class specified.
`-f pattern_file`	Specifies a file that contains a particular class and pattern. New files with this pattern will be placed in the class immediately. Existing files will be moved as well.
`-E`	Specifies that existing files should be moved to the designated class in a one-time move to be scheduled at a later time, which is the sweeptime specified in the `dbdst_admin` command.

To create a preset policy

- Use the `dbdst_preset_policy` command as follows:

```
$ /opt/VRTS/bin/dbdst_preset_policy -D DB2DATABASE -d directory \
-P pattern_spec
```

To create a preset policy for a SMS tablespace or automatic storage path

- Use the `dbdst_preset_policy` command as follows:

```
$ /opt/VRTS/bin/dbdst_preset_policy -D DB2DATABASE -o sms|ats \
-t tablespace -P pattern_spec
```

To display the most commonly used file name patterns for SMS tablespaces or automatic storage paths

- Use the `dbdst_preset_policy` command as follows:

```
$ /opt/VRTS/bin/dbdst_preset_policy -D DB2DATABASE -s -o sms|ats
```

How to use file statistics

You can choose to collect statistics on one or more data files in your database. Collecting statistics allows you to make decisions as to when to move files to slower or faster storage. By default, statistics are collected every thirty minutes.

Setting up statistic collection

To set up statistic collection on files, use the `dbdst_fstat` command.

For SMS tablepsaces, you can only collect statistics on files with extensions *.DAT, *.INX, and *.LB if the `-t tablespace` option is used.

Table 10-5 on page 171. shows the `dbdst_fstat` command options.

Table 10-5 dbdst_fstat command options

Option	Description
`-o start | stop`	Starts or stops file statistics collection.
`-o stopall`	Stops all statistics collection.
`-o list`	Lists files for which statistics are to be collected.
`-f listfile`	Specifies a user-defined file that contains a list of files on which to gather statistics.
`filename`	Specifies the file on which to gather statistics.
`-t tablespace`	Specifies the tablespace on which to gather statistics.

To start collecting statistics on a file

- Use the `dbdst_fstat` command as follows:

```
$ /opt/VRTS/bin/dbdst_fstat -D DB2DATABASE -o start  \
-f listfile
```

Viewing file statistics

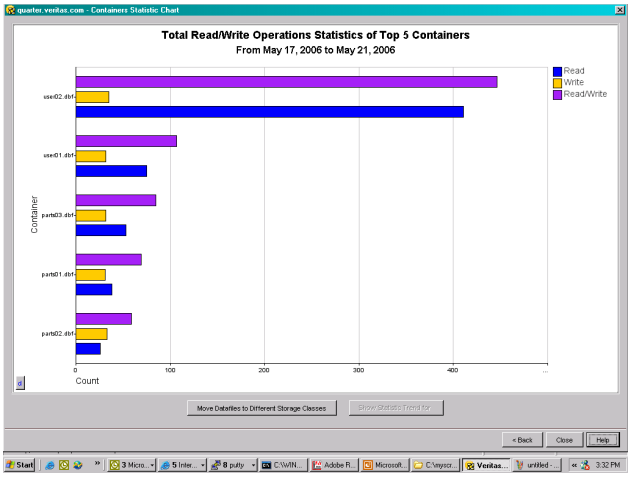
After you have set up statistic collection, you can use the GUI to view the statistics. When you defined your database parameters, the purge interval default value is 30, which means that statistics will be collected and summarized after the initial 30-day period, unless you used a different value. Set a smaller value if you want to view statistics earlier.

For more details, see the *Veritas Storage Foundation for DB2 Graphical User Interface Guide* or the Java GUI online help.

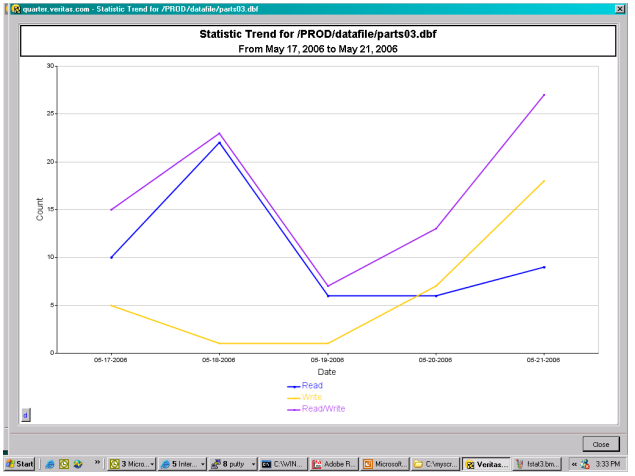
To view file statistics in the GUI

1. Select **DBEDAgent** > **DB2 Instance**, then click on a specific instance.
2. Right click on a specific database and choose **Database Dynamic Statistics Chart**.

3 Enter report parameters and click **Next**. You will see information similar to the following:



4 To display the statistical trend, click **Show Statistic Trend**. The Show Statistic Trend line chart displays.



5 To close the chart pages when finished, click **Close**.

Running Database Dynamic Storage Tiering reports

You can create a report that lists all updated allocation policies or you can view an audit report, which lists recent relocation changes for a specific date range resulting from your policies.

Viewing modified allocation policies

To create a list of modified allocation policies, use the `dbdst_report` command with the `policy` option.

To list allocation policies

- Use the `dbdst_report` command as follows:

```
$ /opt/VRTS/bin/dbdst_report -D DB2DATABASE -o policy
```

For example, to view a list of modified allocation policies, use the `dbdst_report` command as follows:

```
$ /opt/VRTS/bin/dbdst_report -D DB2DATABASE -o policy
```

Viewing audit reports

To view an audit report, which lists recent file relocation changes within a specific date range, use the `dbdst_report` command with the `audit` option.

To view an audit report

- Use the `dbdst_report` command as follows:

```
$ /opt/VRTS/bin/dbdst_report -D DB2DATABASE -o audit \
startdate=yyyy-mm-dd,enddate=yyyy-mm-dd
```

For example, to view an audit report of changes from January 1, 2006 through March 1, 2006, use the `dbdst_report` command as follows:

```
$ /opt/VRTS/bin/dbdst_report -D DB2DATABASE -o audit \
startdate=2006-01-01,enddate=2006-03-01
```

Load balancing in a database environment

To get better performance in a database environment, normally you would use a volume striped over several disks. As the amount of data stored in the file system increases over time, additional space in the form of new disks must be added.

To increase space, you could perform a volume relayout using the `vxrelayout` command. However, changing a large volume from a four-way striped volume to six-way striped volume involves moving old block information into temporary space and writing those blocks from the temporary space to a new volume, which takes a long time. To solve this problem, Veritas Storage Foundation for DB2 offers a new feature called a Load Balanced File System (LBFS).

An LBFS is created on a multi-volume file system where individual volumes are not striped over individual disks. For data-availability, these individual volumes can be mirrored. The file system on the LBFS has a special placement policy called a balance policy. When the balance policy is applied, all the files are divided into small "chunks" and the chunks are laid out on volumes so that adjacent chunks are on different volumes. The default chunk size is 1MB and can be modified. Since every file contains chunks on all available volumes, it is important that individual volumes that make up the LBFS and volume set be of same size and same access properties. Setting up the file system in this way provides the same benefit as striping your volumes. Use the `dbdst_makelbfs` command to create an LBFS file system. Note that you cannot convert an existing file system to an LBFS file system.

Load balancing file system

You can define allocation policies with a balance allocation order and "chunk" size to files or a file system, known as load balancing. The chunk size is the maximum size of any extent that files or a file system with this assigned policy can have. The chunk size can only be specified for allocation policies with a balance allocation order.

A load balancing policy specifies the balance allocation order and a non-zero chunk size. The balance allocation order distributes allocations randomly across the volumes specified in the policy and limits each allocation to a maximum size equal to the specified chunk size.

Load balancing extends the behavior of policy enforcement by rebalancing extent allocations such that each volume in the policy is as equally used as possible. Policy enforcement handles the following cases:

- New volumes are added to the policy, and the extents associated with a file need rebalancing across all volumes, including the new ones.
- Volumes are removed from the volume set or from the policy, and the extents for a file residing on a removed volume need to be moved to other volumes in the policy.
- A load balancing policy is assigned to a file and its extents have to be reorganized to meet the chunk size requirements defined in the policy.

The load balancing policy is intended for balancing data extents belonging to files across volumes defined in the policy. However, there is no restriction imposed in assigning load balancing policy for metadata.

Note: If the fixed extent size is less than the chunk size, then the extent size will be limited to the largest multiple of the fixed extent size that is less than the chunk size. If the fixed extent size is greater than the chunk size, then the extent size will be the fixed extent size.

Creating a Load Balancing File System

Use the `dbdst_makelbfs` command to create a Load Balancing File System (LBFS). In an LBFS, the file extents are distributed on every volume.

Before creating an LBFS, review the following information:

Prerequisites

- You must run the `db2ed_update` command before running the `dbdst_makelbfs` command to make sure the repository is updated.
- The mount point at which you want to mount the LBFS should exist and be owned by the user.

Usage notes

- The `dbdst_makelbfs` command can be executed while the database is online.
- You can not convert an existing file system into an LBFS.
- It is recommended that you mirror your volumes either within the disk array or with Veritas Volume Manager.
- For more information, see the `dbdst_makelbfs`(1M) manual page.

Table 10-6 on page 175. provides the `dbdst_makelbfs` command options.

Table 10-6 dbdst_makelbfs command options

Option	Description
`-D DB2DATABASE`	Specifies the name of the DB2 database for which information will be retrieved.
`[-I DB2INSTANCE]`	Specifies the name of the DB2 instance. This is optional.
`-m` *`mount_point`*	Specifies the mount point where the newly created LBFS needs to be mounted. This mount point should exist and be owned by the user.
`-g` *`disk_group`*	Specifies the disk group on which the volumes reside.
`-C` *`chunk_size`*	Indicates the size in megabytes (MB) of each file extent. The default size is 1MB.

Table 10-6 dbdst_makelbfs command options *(continued)*

Option	Description
`-v volume_name`	Specifies a volume or list of volumes to be used to create a volume set and LBFS. All volumes specified in the list must be owned by the user executing the command. The volumes should be separated with a comma and no spaces.

To create a Load Balancing File System

- Use the `dbdst_makelbfs` command as follows:

```
$ /opt/VRTS/bin/dbdst_makelbfs -D DB2DATABASE -m mount_point \
-g disk_group -C chunk_size -v volume_name,volume_name
```

For example, to create an LBFS with volumes vol1,vol2,vol3 and vol4 on the PROD database, use the `dbdst_makelbfs` command as follows:

```
$ /opt/VRTS/bin/dbdst_makelbfs -D PROD -g db2data -m  /db2_lbfs \
-C 2 -v vol1,vol2,vol3,vol4
```

Adding volumes to a Load Balancing File System

To add additional space to a load balancing file system (LBFS), use the `dbdst_addvol` command using the balance class name. When you add new volumes, all the file-chunks will be redistributed so that adjacent chunks are different volumes, and every volume has approximately same amount of free space.

Prerequisites
- You must have a Load Balancing File System.
  See "Creating a Load Balancing File System" on page 175.

Usage notes
- For more details, see the `dbdst_addvol`(1M) manual page.

To add a volume to a Load Balancing File System

- Use the `dbdst_addvol` command as follows:

```
$ /opt/VRTS/bin/dbdst_addvol -D DB2DATABASE -M mount_device \
-v volume_name:class_name,volume_name:class_name
```

For example, to add two new volumes, new_vol1 and new_vol2, to an existing LBFS, use the `dbdst_addvol` command as follows:

```
$ /opt/VRTS/bin/dbdst_addvol -D PROD -M  /dev/vx/dsk/db2data/vol1 \
-v new_vol1:BALANCE,new_vol2:BALANCE
```

Database Dynamic Storage Tiering use cases for DB2

Use cases were created in order to provide examples of how and why you would use Database Dynamic Storage Tiering.

Creating a preset placement policy for a DB2 SMS tablespace and DB2 automatic storage path

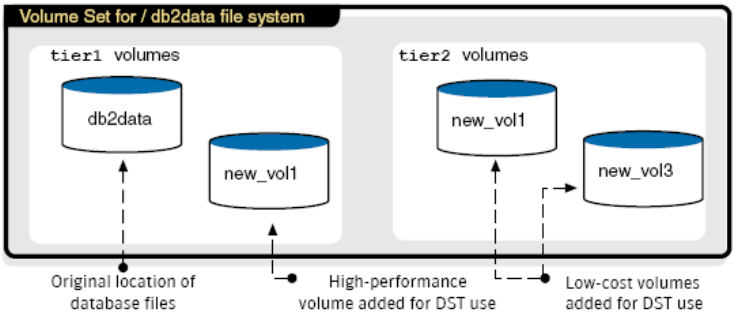
In many cases it is desirable to permanently segregate certain types of database files from other data managed by a system for performance or availability reasons. For example, a data center might choose to reserve its highest-quality storage for database datafiles and index files, and relegate other data to a second tier in the storage hierarchy.

Database Dynamic Storage Tiering preset file placement policies make such segregation possible. An administrator can create a preset placement policy and assign it to a database's file system as the active DST policy.

Database Dynamic Storage Tiering preset policies are useful with DB2's System Managed Storage (SMS) tablespace and automatic storage path features. Preset policies specify where files should be created before the files exist. For automatic storage paths, the database must be created before assigning the preset policies for tablespaces to be managed by automatic storage when they are created.

Figure 10-1 on page 178. shows an example of storage configuration for a file system that contains files for the DB2 database PROD. For SMS tablespaces, the database administrator wants to reserve tier1 volumes for DB2 data and index files, and allocate all other files to tier2 volumes. For automatic storage paths, the database administrator wants to reserve tier1 volumes for the System Catalog Tablespace and all other tablespaces on tier2 volumes.

Figure 10-1 Database storage configuration example



First, create a DB2 database, PROD, using automatic storage path. Then, run the db2ed_update command as follows:

```
$ /opt/VRTS/bin/db2ed_update -D PROD
```

Then you must set up your system to use Database Dynamic Storage Tiering.

To add the fast_storage and slow_storage storage classes

- Use the `dbdst_admin` command as follows:

```
$ /opt/VRTS/bin/dbdst_admin -D PROD -o addclass=\
tier1:"Fast Storage"

$ /opt/VRTS/bin/dbdst_admin -D PROD -o addclass=\
tier2:"Slow Storage"
```

To convert the database's file system and add volumes for use with Database Dynamic Storage Tiering

- Use the `dbdst_convert` command as follows:

```
$ /opt/VRTS/bin/dbdst_convert -D PROD \
-M /dev/vx/dsk/db2dg/db2data -v new_vol1,new_vol2,new_vol3
```

To classify volumes into storage classes

- Use the `dbdst_classify` command as follows:

```
$ /opt/VRTS/bin/dbdst_classify -D PROD \
-M /dev/vx/dsk/db2dg/db2data -v new_vol1:tier1
```

```
$ /opt/VRTS/bin/dbdst_classify -D PROD \
-M /dev/vx/dsk/db2dg/db2data -v new_vol2:tier2,\
new_vol3:tier2
```

Once the volumes are configured, an administrator can set policies for SMS tablespaces and automatic storage path.

To create a policy for the System Catalog Tablespace in automatic storage path

- Use the `dbdst_preset_policy` command as follows:

```
$ /opt/VRTS/bin/dbdst_preset_policy -D PROD -o ats -P \
"tier1=*.CAT;tier2=*.TMP,*.UTM,*.USR,*.LRG"
```

To create a SMS directory, and create and assign a file placement policy

1. Create a directory for your SMS tablespace.

```
$ mkdir /db2data/tbs01
```

2. Use the `dbdst_preset_policy` command as follows:

```
$ /opt/VRTS/bin/dbdst_preset_policy -D PROD -d /db2data/tbs01 \
-P "tier1=SQL*.DAT,SQL*.INX:tier2=SQL*.LB,SQL*.LBA,SQL*.LF,SQL*.BKM"
```

The command creates the SMS tablespace container directory /db2data/tbs01, as well as a file placement policy for the file system /db2data.

Once the preset policy has been created and assigned to the database's file system, a database administrator can use DB2 administrative commands to create the SMS tablespace tbs01 specifying /db2data/tbs01 as the tablespace container. Space for data and index files created by DB2 during the course of operations are allocated on fast_storage volumes; all other files are allocated on slow_storage volumes. Database Dynamic Storage Tiering also enforces the policy to relocate already-existing database files to the appropriate placement class.

It is important to run db2ed_update to record the tablespace information in the SFDB repository:

```
$ /opt/VRTS/bin/db2ed_update -D PROD
```

Migrating existing partitioned data in a DB2 database

Perhaps the simplest application of multi-tier storage to databases is relocation of partitioned data between different placement classes as usage requirements change. If exact relocation times are unpredictable, or if relocation is infrequent, administrators may wish to relocate table partitions as business requirements surface rather than defining strict periodic relocation schedules.

For example, a large retailer keeps the most recent year of sales information in the fast storage class and the rest of the data in the slow storage class. The database administrator creates one table per quarter, using four tables to represent a year of sales data. Each table resides in a separate tablespace. The database administrator then unions the four tables together, called year_sales, using the UNION ALL construct in DB2. Every quarter, the database administrator rolls out the oldest table to slower storage to make room for the new table in the fast storage class. Assume that the current tables are sales_0304, sales_0404, sales_0105, and sales_0205. The sales_0304 table represents the oldest quarter, and it resides in tablespace tbs_sales_0304.

First, you must set up your system to use Database Dynamic Storage Tiering.

To add the fast_storage and slow_storage storage classes

- Use the `dbdst_admin` command as follows:

```
$ /opt/VRTS/bin/dbdst_admin -D PROD -o addclass=\
fast_storage:"Fast Storage for Production DB"

$ /opt/VRTS/bin/dbdst_admin -D PROD -o addclass=\
slow_storage:"Slow Storage for Production DB"
```

To convert the database's file system and add volumes for use with Database Dynamic Storage Tiering

- Use the `dbdst_convert` command as follows:

```
$ /opt/VRTS/bin/dbdst_convert -D PROD \
-M /dev/vx/dsk/db2dg/db2data -v new_vol1,new_vol2,new_vol3
```

To classify volumes into storage classes

- Use the `dbdst_classify` command as follows:

```
$ /opt/VRTS/bin/dbdst_classify -D PROD -M /dev/vx/dsk/db2dg/db2data \
-v new_vol1:fast_storage,db2data_b4vset:fast_storage

$ /opt/VRTS/bin/dbdst_classify -D PROD \
-M /dev/vx/dsk/db2dg/db2data -v new_vol2:slow_storage,\
new_vol3:slow_storage
```

Once the volumes are configured, an administrator can roll out the oldest table, sales_0304, and add a new table

To move the oldest sales data into a slower class and add a new tablespace

1. Use the `dbdst_tbs_move` command as follows:

```
$ /opt/VRTS/bin/dbdst_tbs_move -D PROD -t tbs_sales_0304 \
-c slow_storage
```

2. Drop the year_sales view.
3. Create a new tablespace, tbs_sales_0305, and create the table sales_0305 in this tablespace.
4. Recreate the file year_sales by performing a UNION ALL, including sales_0305 and excluding sales_0304.
5. Move the tablespace to fast storage as follows:

```
$ /opt/VRTS/bin/dbdst_tbs_move -D PROD -t tbs_sales_0305 \
-c fast_storage
```

Scheduling the relocation of archive logs

Because they are the primary mechanism for recovering from data corruption, database logs are normally kept on premium storage, both for I/O performance and data reliability reasons. Even after they have been archived, logs are normally kept online for fast recovery, but the likelihood of referring to an archived log decreases significantly as its age increases. This suggests that archived database logs might be relocated to lower-cost volumes after a certain period of inactivity.

The rapidly decaying probability of use for archive logs suggests that regular enforcement of a placement policy that relocates them to lower-cost storage after a period of inactivity can reduce an enterprise's average cost of online storage.

For example, a customer could be using a large DB2 database with thousands of active sessions, which needs to be up and running 24 hours a day and seven days a week with uptime of over 99%, and the database generates a large number of archive logs per day. If the database goes down for any reason, there is business requirement to bring the database back online and functional with in 15 minutes. The archive logs need to be created in a fast EMC array. Archive logs older than a week can be moved to a mid-range Clarion array. Archive logs older than 15 days can be moved slow JBOD disks. The database administrator purges the archive logs after 30 days. To set up a policy like this, see the following example. Assume that archive logs are created on the file system, /proddb_archlog, using DB2 commands.

To add the NEW, MEDIUM, and OLD storage classes

- Use the `dbdst_admin` command as follows:

```
$ /opt/VRTS/bin/dbdst_admin -D PROD -o addclass=\
NEW:"EMC Storage for Production DB"

$ /opt/VRTS/bin/dbdst_admin -D PROD -o addclass=\
MEDIUM:"Clarion Storage for Production DB"

$ /opt/VRTS/bin/dbdst_admin -D PROD -o addclass=\
OLD:"JBOD Storage for Production DB"
```

To convert the database's file system and add volumes for use with Database Dynamic Storage Tiering

- Use the `dbdst_convert` command as follows:

```
$ /opt/VRTS/bin/dbdst_convert -D PROD -g db2dg \
-M /dev/vx/dsk/db2dg/db2log -v emc_v1,clarion_v1,jbod_v1
```

To classify volumes into storage classes

- Use the `dbdst_classify` command as follows:

```
$ /opt/VRTS/bin/dbdst_classify -D PROD -g db2dg \
-M /dev/vx/dsk/db2dg/db2log -v emc_v1:NEW

$ /opt/VRTS/bin/dbdst_classify -D PROD -g db2dg \
-M /dev/vx/dsk/db2dg/db2log -v clarion_v1:MEDIUM

$ /opt/VRTS/bin/dbdst_classify -D PROD -g db2dg \
-M /dev/vx/dsk/db2dg/db2log -v jbod_v1:OLD
```

Once the volumes are configured, an administrator can define file placement policy rules that specify access age-based relocation of selected files and assign them to the database's file system.

To define rules that periodically relocate archive logs

- Use the `dbdst_file_move` command as follows:

  ```
  $ /opt/VRTS/bin/dbdst_file_move -D PROD -o archive1 -c MEDIUM:7 -c OLD:
  ```

  This relocates files in the archive1 directory that have not been accessed for seven days to the MEDIUM volume, and files that have not been accessed for 15 days to the OLD volume.

Database Dynamic Storage Tiering translates these commands into DST access age-based policy rules, merges them with the file system's placement policy, and assigns the resulting policy to the file system. By default, Database Dynamic Storage Tiering enforces the active policy daily. During enforcement, the new rules relocate qualifying files to the destination storage tiers specified in the `dbdst_file_move` commands used to create the policies.

Chapter 11

Using Database FlashSnap for backup and off-host processing

This chapter includes the following topics:

About Veritas Database FlashSnap

Veritas Database FlashSnap is included with Veritas Database Edition Enterprise Edition.

Database FlashSnap lets you capture an online image of an actively changing database at a given instant, known as a snapshot. You can perform backups and off-host processing tasks on snapshots while providing continuous availability of your critical data, with minimal interruption to users. Database FlashSnap commands can be executed from either the command line or the GUI.

Veritas Database FlashSnap offers you a flexible way to efficiently manage multiple point-in-time copies of your data, and reduce resource contention on your business-critical servers. Database FlashSnap allows database administrators to create a consistent copy of a database without root privileges by creating a snapshot. A snapshot copy of the database is referred to as a database snapshot.

You can use a database snapshot on the same host as the production database or on a secondary host sharing the same storage. A database snapshot can be used for off-host processing applications, such as backup, data warehousing, and decision-support queries. When the snapshot is no longer needed, the database administrator can import the original snapshot back to the primary host and resynchronize the snapshot to the original database volumes.

Database FlashSnap also allows you to resynchronize your original database volumes from the data in the snapshot if the original volumes become corrupted. This is referred to as reverse resynchronization.

Database FlashSnap can significantly reduce the time it takes to back up your database, increase the availability of your production database, and still maintain your production database's performance.

Note: To use Database FlashSnap, you must have Veritas Storage Foundation Enterprise Edition on all systems on which you intend to use Database FlashSnap.

To use Database FlashSnap, you must first configure the volumes used by the database.

Typical database problems solved with Database FlashSnap

Database FlashSnap is designed to enable you to use database snapshots to overcome the following types of problems encountered in enterprise database environments:

- In many companies, there is a clear separation between the roles of system administrators and database administrators. Creating database snapshots

typically requires superuser (root) privileges, privileges that database administrators do not usually have.

- In some companies, database administrators are granted root privileges, but managing storage is typically not central to their job function or their core competency.
- Creating database snapshots is a complex process, especially in large configurations where thousands of volumes are used for the database. One mistake can render the snapshots useless.

Because it does not require root privileges, Database FlashSnap overcomes these obstacles by enabling database administrators to create consistent snapshots of the database more easily. The snapshots can be utilized for repetitive use.

About Database FlashSnap applications

The following are typical applications of Veritas Database FlashSnap:

- Database Backup and Restore
  Enterprises require 24/7 online data availability. They cannot afford the downtime involved in backing up critical data offline. By creating a clone database or a duplicate volume snapshot of data, and then using it to back up your data, your business-critical applications can continue to run without extended down time or impacted performance. After a clone database or snapshot volume is created, it can be used as a source to back up the original database.
- Decision-Support Analysis and Reporting
  Operations such as decision-support analysis and business reporting may not require access to real-time information. You can direct such operations to use a clone database that you have created from snapshots using Veritas Database FlashSnap, rather than allowing them to compete for access to the primary volume or database. When required, you can quickly resynchronize the clone database with the primary database to get up-to-date information.
- Application Development and Testing
  Development or service groups can use a clone database created with Database FlashSnap as a test database for new applications. A clone database provides developers, system testers, and quality assurance groups with a realistic basis for testing the robustness, integrity, and performance of new applications.
- Logical Error Recovery
  Logical errors caused by an administrator or an application program can compromise the integrity of a database. You can recover a database by restoring the database files from a volume snapshot from a clone database created from

volume snapshots using Database FlashSnap. These solutions are faster than restoring database files from tape or other backup media.

Database FlashSnap

The system administrator needs to configure storage according to the requirements specified in the snapplan.

Database FlashSnap allows you to check the storage setup against requirements set forth in the snapplan. Depending on the results, the database administrator may need to modify the snapplan or the system administrator may need to adjust the storage configuration. Properly configuring storage is the only aspect of using Database FlashSnap that requires the system administrator's participation.

To use Database FlashSnap, a database administrator must first define their snapshot requirements. For example, they need to determine whether off-host processing is required and, if it is, which host should be used for it. In addition, it is also important to consider how much database downtime can be tolerated. Database snapshot requirements are defined in a file called a snapplan. The snapplan specifies snapshot options that will be used when creating a snapshot image (such as whether the snapshot mode will be `online_snapshot`, `online_mirror`, or `offline`).

After creating the snapplan, the database administrator must validate it to ensure that it is correct. During validation the snapplan is copied to the repository before using it to create a snapshot. Depending on the validation results, the database administrator may need to modify the snapplan or the system administrator may need to adjust the storage configuration.

After storage is configured as specified in the snapplan and the snapplan has been validated, the database administrator can create snapshots of the database and create database clones based on the snapshots on either the same host or a secondary one.

A database clone can be used on a secondary host for off-host processing, including decision-support analysis and reporting, application development and testing, database backup, and logical error recovery. After a user has finished using the clone on a secondary host, the database administrator can shut down the clone and move the snapshot database back to the primary host. Regardless of whether a snapshot is used on the primary or secondary host, it can be resynchronized with the primary database using Database FlashSnap. Database FlashSnap utilizes Veritas Volume Manager FastResync to quickly resynchronize the changed section between the primary and snapshot.

Database FlashSnap can also be used to recover the primary copy of the database if it becomes corrupted by overwriting it with the snapshot. You can recover the

primary database with a snapshot using the reverse resynchronization functionality of Database FlashSnap.

See the *Veritas Volume Manager User's Guide*.

Database FlashSnap commands

The Database FlashSnap feature consists of three commands:

- `db2ed_vmchecksnap` (used on the primary host)
  Creates and validates the snapshot plan used to create a snapshot image of a DB2 database. You can also use `db2ed_vmchecksnap` to copy, list, or remove a snapplan or make sure the storage is configured properly for the task.
- `db2ed_vmsnap` (used on the primary host)
  Creates a snapshot image of a DB2 database by splitting the mirror volumes used by the database. You can also use `db2ed_vmsnap` to resynchronize snapshot volumes with their original volumes. The command also allows you to resynchronize the original volumes from the data in the snapshot volumes, which is useful if the original volumes become corrupted. Resynchronizing the original volumes from the snapshot volumes is known as reverse resynchronization.
- `db2ed_vmclonedb` (used on the primary or secondary host)
  Mounts and starts a clone database using snapshot volumes. It can also shut down a clone snapshot database and deport its volumes, as well as restart a clone snapshot database that has been shut down. The snapshot image can be brought up on the same host running the primary database or on a secondary host.

All of these commands can be executed by the DB2 database administrator and do not require superuser (`root`) privileges.

Note: Database FlashSnap operations can be executed from either the command line or the GUI.

Database FlashSnap options

Database FlashSnap offers three options for creating database snapshots. The option you choose is specified in the snapplan.

- `online_snapshot`
  Use this option to:
  - create a clone database for decision-support, reporting, and testing purposes.

- take over the primary database if it becomes corrupted
- back up the primary database at the operating system level
  With this option, the `db2ed_vmsnap` command will first put the database into `WRITE SUSPEND` mode. After `db2ed_vmsnap` finishes creating the snapshot, it will take the database out of `WRITE SUSPEND` mode. The online redo log must be included in the snapshot.

- `online_mirror`
  Use this option to create a "hot backup." The online_mirror option creates a mirror of the database without creating a clone database. Creating a clone database is not allowed with the `online_mirror` option. The mirror database can be used to restore the primary database if media failure occurs.
  With this option, the `db2ed_vmsnap` command will first put the database into `WRITE SUSPEND` mode. After `db2ed_vmsnap` finishes creating the snapshot, it will take the database out of `WRITE SUSPEND` mode. The active log must not be included in the snapshot.

- `offline`
  The `offline` option can be used to clone or back up a database. With this option, the database must be inactive when the snapshot is created. Because the database is inactive, it is not required to be placed into `WRITE SUSPEND` mode. This type of snapshot is a valid database backup. You must include all data and active transaction log space volumes used by its database in the snapshot.

Note: In this release of Veritas Storage Foundation for DB2, Database FlashSnap supports third mirror break-off snapshots only. Third mirror break-off snapshots are fully synchronized, full-sized snapshots. See the *Veritas Volume Manager Administrator's Guide*.

How to plan for Database FlashSnap

Before using Database FlashSnap, you must determine your intended application. You then need to make the following decisions:

- Which snapshot mode is appropriate: `online_snapshot`, `online_mirror`, or `offline`?
- Will you need one or two hosts?

Snapshot mode

Table 11-1 shows common uses for each snapshot mode.

Table 11-1 Snapshot modes and their purposes

Snapshot mode	Purpose
`online_snapshot`	Choose `online_snapshot` if you want to: ■ Clone a database. A clone database can be used for decision-support analysis, reporting, development, or testing. ■ Recover the primary database if it is corrupted by either taking over the primary role or performing a reverse resync. With `online_snapshot` you cannot roll forward any logs because the primary and secondary log sequences are different. ■ Back up the primary database on the same or a different host.
`online_mirror`	Choose `online_mirror` if you want to: ■ Create a “hot backup” image. ■ Back up the primary database on the same host at the operating system level using different set of storage. ■ Perform a point-in-time recovery of the primary database if it becomes corrupted. With `online_mirror`, it is possible to roll forward to recover changes to your database.
`offline`	Choose `offline` if you want to: ■ Clone your production database if it is offline. A clone database can be used for decision-support analysis, reporting, development, or testing. ■ Back up your production database if it is offline.

One or two snapshot hosts

If maintaining the performance of your primary database is critical, you can offload processing of the snapshots to a secondary host. For off-host processing, storage must be shared between the primary and secondary hosts.

If cost savings is most important, you can choose to do the processing on the same host as the primary database to save on hardware costs.

Hosts and storage for Database FlashSnap

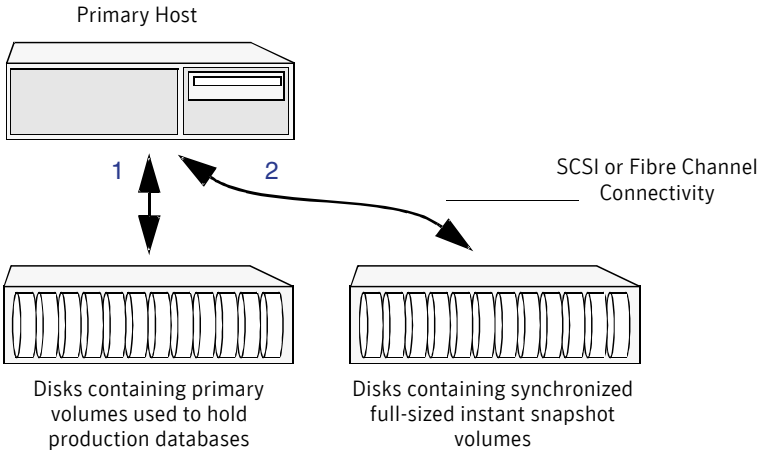
Database FlashSnap requires sufficient Veritas Volume Manager disk space, and can be used on the same host that the database resides on (the primary host) or on a secondary host.

Setting up a storage configuration for Database FlashSnap operations is a system administrator's responsibility and requires superuser (root) privileges. Database FlashSnap utilities do not address setting up an appropriate storage configuration.

Single-host configuration

Figure 11-1 shows the suggested arrangement for implementing Database FlashSnap solutions on the primary host to avoid disk contention.

Figure 11-1 Example of a Database FlashSnap solution on a primary host



Two-host configuration

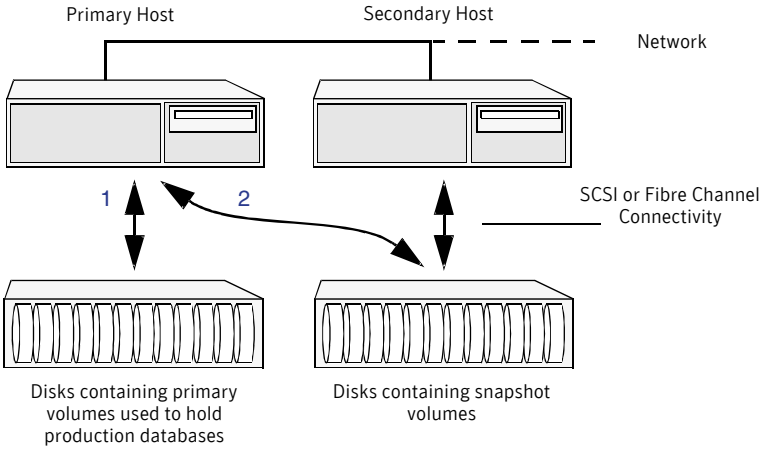
A Database FlashSnap configuration with two hosts allows CPU- and I/O-intensive operations to be performed for online backup and decision support without degrading the performance of the primary host running the production database. A two-host configuration also allows the snapshot database to avoid contending for I/O resources on the primary host.

Figure 11-2 shows a two-host configuration.

For off-host processing applications, both the primary and secondary hosts need to share the storage in which the snapshot database is created. Both the primary

and secondary hosts must be able to access the disks containing the snapshot volumes.

Figure 11-2 Example of an off-host Database FlashSnap solution



Host and storage requirements

Before using Database FlashSnap, ensure that:

- All files are on VxFS file systems over VxVM volumes. Raw devices are not supported.
- Symbolic links to containers are not supported.
- DB2 containers and active logs are in the same disk group.

In addition, before attempting to use Database FlashSnap with two hosts, ensure that:

- The versions of Veritas Storage Foundation for DB2 on the primary and secondary hosts are the same.
- The same version of DB2 is installed on both hosts.
- The DB2 binaries and datafiles are on different volumes and disks.
- The UNIX login for the database user and group must be the same on both hosts.
- You have a Veritas Storage Foundation for DB2 Enterprise Edition license on both hosts.

Creating a snapshot mirror of a volume or volume set used by the database

With Database FlashSnap, you can mirror the volumes used by the database to a separate set of disks, and those mirrors can be used to create a snapshot of the database. These snapshot volumes can be split and placed in a separate disk group. This snapshot disk group can be imported on a separate host, which shares the same storage with the primary host. The snapshot volumes can be resynchronized periodically with the primary volumes to get recent changes of the primary database. If the primary containers become corrupted, you can quickly restore them from the snapshot volumes. Snapshot volumes can be used for a variety of purposes, including backup and recovery, and creating a clone database.

You must create snapshot mirrors for all of the volumes used by the database containers before you can create a snapshot of the database.

Before creating a snapshot mirror, make sure the following conditions have been met:

Prerequisites

- You must be logged in as superuser (root).
- The disk group must be version 110 or later. For more information on disk group versions, see the `vxdg`(1M) online manual page.
- Be sure that a data change object (DCO) and a DCO log volume are associated with the volume for which you are creating the snapshot.
- Persistent FastResync must be enabled on the existing database volumes and disks must be assigned for the snapshot volumes. FastResync optimizes mirror resynchronization by tracking updates to stored data that have been missed by a mirror. When a snapshot mirror is reattached to its primary volumes, only the updates that were missed need to be re-applied to resynchronize it. FastResync increases the efficiency of the volume snapshot mechanism to better support operations such as backup and decision support. For detailed information about FastResync, see the *Veritas Volume Manager Administrator's Guide*.
- Snapshot mirrors and their associated DCO logs should be on different disks than the original mirror plexes, and should be configured correctly for creating snapshots by the system administrator.
- When creating a snapshot mirror, create the snapshot on a separate controller and separate disks from the primary volume.
- Snapplan validation (`db2ed_vmchecksnap`) fails if any volume in the disk group is not used for the database.

Usage notes

- Create a separate disk group for DB2 database-related files.
- Do not share volumes between DB2 database files and other software.
- `INSTANCE_HOME` cannot be included in the snapshot mirror.
- Resynchronization speed varies based on the amount of data changed in both the primary and snapshot volumes during the break-off time.
- Do not share any disks between the original mirror and the snapshot mirror.
- Snapshot mirrors for datafiles and archive logs should be created so that they do not share any disks with the data of the original volumes. If they are not created in this way, the VxVM disk group cannot be split and, as a result, Database FlashSnap will not work.

Note: Database FlashSnap commands support third-mirror break-off snapshots only. The snapshot mirror must be in the `SNAPDONE` state.

To create a snapshot mirror of a volume

1 Create a volume:

```
vxassist -g diskgroup make volume_name size disk_name
```

For example:

```
# vxassist -g PRODdg make snapvset_1 8g data01
```

2 Create a DCO and enable FastResync on the volume:

```
vxsnap -g diskgroup prepare volume_name [alloc=storage_attribute]
```

For example:

```
# vxsnap -g PRODdg prepare snapvset [alloc="snap01"]
```

3 Verify that a DCO and DCO log are attached to the volume:

```
vxprint -g diskgroup -F%fastresync volume_name
vxprint -g diskgroup -F%hasdcolog volume_name
```

This command returns `on` if a DCO and DCO log are attached to the specified volume.

For example:

```
# vxprint -g PRODdg -F%fastresync snapdata
on
# vxprint -g PRODdg -F%hasdcolog snapdata
on
```

4 Create a mirror of the volume:

```
vxsnap -g diskgroup addmir volume_name [alloc=storage_attribute]
```

For example:

```
# vxsnap -g PRODdg addmir snapdata [alloc="data02"]
```

5 List the available mirrored data plex for the volume:

```
vxprint -g diskgroup -F%name -e"pl_v_name in \"volume_name\""
```

For example:

```
# vxprint -g PRODdg -F%name -e"pl_v_name in\"snapdata\""
snapdata-01
snapdata-02
```

6 Set the `dbed_flashsnap` tag on the break-off data plex for the volume:

```
vxedit -g diskgroup set putil2=dbed_flashsnap plex_name
```

For example:

```
# vxedit -g PRODdg set putil2=dbed_flashsnap snapdata-02
```

7 Verify that the `dbed_flashsnap` tag has been set on the data plex for each volume in the volume set:

```
vxprint -g diskgroup -F%name -e"pl_v_name in \"volume_name\" \
&& p2 in \"dbed_flashsnap\""
```

For example:

```
# vxprint -g PRODdg -F%name -e"pl_v_name in \"snapdata\" \
&& p2 in \"dbed_flashsnap\""
snapdata-02
```

8 Verify the disk group configuration:

```
vxprint -g diskgroup
```

For example:

```
# vxprint -g PRODdg

TY NAME         ASSOC           KSTATE   LENGTH   PLOFFS STATE    TUTIL0 PUTIL0
dg PRODdg       PRODdg          -        -        -      -        -      -

dm data01       Disk_17         -        33483648 -      -        -      -
dm data02       Disk_24         -        33483648 -      -        -      -

v  snapdata     fsgen           ENABLED  16777216 -      ACTIVE   -      -
pl snapdata-01  snapdata        ENABLED  16777216 -      ACTIVE   -      -
sd data01-01    snapdata-01     ENABLED  16777216 0      -        -      -
pl snapdata-02  snapdata        ENABLED  16777216 -      SNAPDONE -      -
sd data02-01    snapdata-02     ENABLED  16777216 0      -        -      -
dc snapdata_dco snapdata        -        -        -      -        -      -
v  snapdata_dcl gen             ENABLED  1696     -      ACTIVE   -      -
pl snapdata_dcl-01 snapdata_dcl ENABLED  1696     -      ACTIVE   -      -
sd data01-02    snapdata_dcl-01 ENABLED  1696     0      -        -      -
pl snapdata_dcl-02 snapdata_dcl DISABLED 1696    -       DCOSNP   -      -
sd data02-02    snapdata_dcl-02 ENABLED  1696     0      -        -      -
```

To create a snapshot mirror of a volume set

1 Create a volume:

```
vxassist -g diskgroup make volume_name size disk_name
```

For example:

```
# vxassist -g PRODdg make snapvset_1 8g snap01
# vxassist -g PRODdg make snapvset_2 4g snap01
```

2 Create a volume set for use by VxFS:

```
vxvset -g diskgroup -t vxfs make vset_name volume_name
```

For example:

```
# vxvset -g PRODdg -t vxfs make snapvset snapvset_1
```

3 Add the volume to the volume set:

```
vxvset -g diskgroup addvol vset_name volume_name
```

For example:

```
# vxvset -g PRODdg addvol snapvset snapvset_2
```

4 Create a DCO and enable FastResync on the volume set:

```
vxsnap -g diskgroup prepare vset_name [alloc=storage_attribute]
```

For example:

```
# vxsnap -g PRODdg prepare snapvset [alloc="snap01"]
```

5 Verify that a DCO and DCO log are attached to each volume in the volume set:

```
vxprint -g diskgroup -F%fastresync volume_name
vxprint -g diskgroup -F%hasdcolog volume_name
```

This command returns `on` if a DCO and DCO log are attached to the specified volume.

For example:

```
# vxprint -g PRODdg -F%fastresync snapvset_1
on
# vxprint -g PRODdg -F%fastresync snapvset_2
on
# vxprint -g PRODdg -F%hasdcolog snapvset_1
on
# vxprint -g PRODdg -F%hasdcolog snapvset_2
on
```

6 Create a mirror of the volume set volumes:

```
vxsnap -g diskgroup addmir vset_name [alloc=storage_attribute]
```

For example:

```
# vxsnap -g PRODdg addmir snapvset [alloc="snap02"]
```

7 List the available mirrored data plex for each volume in the volume set:

```
vxprint -g diskgroup -F%name -e"pl_v_name in \"volume_name\""
```

For example:

```
# vxprint -g PRODdg -F%name -e"pl_v_name in\"snapvset_1\""
snapvset_1-01
snapvset_1-02

# vxprint -g PRODdg -F%name -e"pl_v_name in\"snapvset_2\""
snapvset_2-01
snapvset_2-02
```

8 Set the `dbed_flashsnap` tag on the break-off data plex for each volume in the volume set:

```
vxedit -g diskgroup set putil2=dbed_flashsnap plex_name
```

For example:

```
# vxedit -g PRODdg set putil2=dbed_flashsnap snapvset_1-02
# vxedit -g PRODdg set putil2=dbed_flashsnap snapvset_2-02
```

9 Verify that the `dbed_flashsnap` tag has been set on the data plex for each volume in the volume set:

```
vxprint -g diskgroup -F%name -e"pl_v_name in \"volume_name\" \
&& p2 in \"dbed_flashsnap\""
```

For example:

```
# vxprint -g PRODdg -F%name -e"pl_v_name in \"snapvset_1\" \
&& p2 in \"dbed_flashsnap\""
snapvset_1-02

# vxprint -g PRODdg -F%name -e"pl_v_name in \"snapvset_2\" \
&& p2 in \"dbed_flashsnap\""
snapvset_2-02
```

10 Verify the snapshot volume configuration:

```
vxprint -g diskgroup
```

For example:

```
# vxprint -g PRODdg
TY NAME         ASSOC               KSTATE   LENGTH   PLOFFS STATE    TUTIL0  PUTIL0
dg PRODdg       PRODdg              -        -        -      -        -       -

dm snap01       Disk_9              -        33483648 -      -        -       -
dm snap02       Disk_11             -        33483648 -      -        -       -

vt snapvset     -                   ENABLED  -        -      ACTIVE   -       -
v  snapvset_1   snapvset            ENABLED  16777216 -      ACTIVE   -       -
pl snapvset_1-01 snapvset_1         ENABLED  16777216 -      ACTIVE   -       -
sd snap01-01    snapvset_1-01       ENABLED  16777216 0      -        -       -
pl snapvset_1-02 snapvset_1         ENABLED  16777216 -      SNAPDONE -       -
sd snap02-01    snapvset_1-02       ENABLED  16777216 0      -        -       -
dc snapvset_1_dco snapvset_1        -        -        -      -        -       -
v  snapvset_1_dcl gen               ENABLED  1696     -      ACTIVE   -       -
pl snapvset_1_dcl-01 snapvset_1_dcl ENABLED  1696     -      ACTIVE   -       -
sd snap01-03    snapvset_1_dcl-01   ENABLED  1696     0      -        -       -
pl snapvset_1_dcl-02 snapvset_1_dcl DISABLED 1696     -      DCOSNP   -       -
sd snap02-02    snapvset_1_dcl-02   ENABLED  1696     0      -        -       -
v  snapvset_2   snapvset            ENABLED  8388608  -      ACTIVE   -       -
pl snapvset_2-01 snapvset_2         ENABLED  8388608  -      ACTIVE   -       -
sd snap01-02    snapvset_2-01       ENABLED  8388608  0      -        -       -
pl snapvset_2-02 snapvset_2         ENABLED  8388608  -      SNAPDONE -       -
sd snap02-03    snapvset_2-02       ENABLED  8388608  0      -        -       -
dc snapvset_2_dco snapvset_2        -        -        -      -        -       -
v  snapvset_2_dcl gen               ENABLED  1120     -      ACTIVE   -       -
pl snapvset_2_dcl-01 snapvset_2_dcl ENABLED  1120     -      ACTIVE   -       -
sd snap01-04    snapvset_2_dcl-01   ENABLED  1120     0      -        -       -
pl snapvset_2_dcl-02 snapvset_2_dcl DISABLED 1120     -      DCOSNP   -       -
sd snap02-04    snapvset_2_dcl-02   ENABLED  1120     0      -        -       -
```

This example shows the steps involved in creating a snapshot mirror for the volume `data_vol` belonging to the disk group `PRODdg`.

Prepare the volume `data_vol` for mirroring:

```
# vxsnap -g PRODdg prepare data_vol alloc=PRODdg01
```

Verify that FastResync is enabled:

```
# vxprint -g PRODdg -F%fastresync data_vol
on
```

Verify that a DCO and a DCO log are attached to the volume:

```
# vxprint -g PRODdg -F%hasdcolog data_vol
on
```

Create a snapshot mirror of `data_vol`:

```
# vxsnap -g PRODdg addmir data_vol alloc=PRODdg02
```

List the data plexes:

```
# vxprint -g PRODdg -F%name -e"pl_v_name in \"data_vol\""
data_vol-01
data_vol-02
```

Choose the plex that is in the `SNAPDONE` state. Use the `vxprint -g diskgroup` command to identify the plex that is in the `SNAPDONE` state.

Decide which data plex you want to use and set the *`tag name`* tag for it:

```
# vxedit -g PRODdg set putil2=PRODtag data_vol-02
```

Verify that the *`PRODtag`* tag has been set to the desired data plex, `data_vol-02`:

```
# vxprint -g PRODdg -F%name -e"pl_v_name in \"data_vol\" \
&& p2 in \"PRODtag\""
data_vol-02
```

To verify that the snapshot volume was created successfully, use the `vxprint -g` *`disk_group`* command as follows:

```
# vxprint -g PRODdg
```

```
v  data_vol      fsgen           ENABLED  4194304  -        ACTIVE   -       -
pl data_vol-01   data_vol        ENABLED  4194304  -        ACTIVE   -       -
sd PRODdg03-01   data_vol-01     ENABLED  4194304  0        -        -       -
pl data_vol-02   data_vol        ENABLED  4194304  -        SNAPDONE -       -
sd PRODdg02-01   data_vol-02     ENABLED  4194304  0        -        -       -
dc data_vol_dco  data_vol        -        -        -        -        -       -
v  data_vol_dcl  gen             ENABLED  560      -        ACTIVE   -       -
pl data_vol_dcl-01 data_vol_dcl  ENABLED  560      -        ACTIVE   -       -
sd PRODdg01-01   data_vol_dcl-01 ENABLED  5600     -        -        -       -
pl data_vol_dcl-02 data_vol_dcl  DISABLED 560      -        DCOSNP   -       -
sd PRODdg02-02   data_vol_dcl-02 ENABLED  5600     -        -        -       -
```

Identify that the specified plex is in the `SNAPDONE` state. In this example, it is `data_vol-02`.

The snapshot mirror is now ready to be used.

Upgrading existing volumes to use with Database FlashSnap for DB2

You may have to upgrade a volume created using a version older than VxVM 5.0 so that it can take advantage of Database FlashSnap.

Note: Snapshots that were created using a version older than VxVM 4.0 are not supported by Database FlashSnap.

The plexes of the DCO volume require persistent storage space on disk to be available. To make room for the DCO plexes, you may need to add extra disks to the disk group, or reconfigure existing volumes to free up space in the disk group. Another way to add disk space is to use the disk group move feature to bring in spare disks from a different disk group.

Existing snapshot volumes created by the `vxassist` command are not supported. A combination of snapshot volumes created by `vxassist` and `vxsnap` are not supported.

To upgrade an existing volume created with an earlier version of VxVM

1. Upgrade the disk group that contains the volume to a version 120 or higher before performing the remainder of this procedure. Use the following command to check the version of a disk group:

   ```
   # vxdg list diskgroup
   ```

   To upgrade a disk group to the latest version, use the following command:

   ```
   # vxdg upgrade diskgroup
   ```

2. If the volume to be upgraded has a DRL plex or subdisk from an earlier version of VxVM, use the following command to remove this:

   ```
   # vxassist [-g diskgroup] remove log volume [nlog=n]
   ```

   Use the optional attribute `nlog=n` to specify the number, *n*, of logs to be removed. By default, the `vxassist` command removes one log.

3. For a volume that has one or more associated snapshot volumes, use the following command to reattach and resynchronize each snapshot:

   ```
   # vxsnap [-g diskgroup] snapback snapvol
   ```

   If persistent FastResync was enabled on the volume before the snapshot was taken, the data in the snapshot plexes is quickly resynchronized from the original volume. If persistent FastResync was not enabled, a full resynchronization is performed.

4. Use the following command to turn off persistent FastResync for the volume:

   ```
   # vxvol [-g diskgroup] set fastresync=off volume
   ```

5. Use the following command to dissociate a DCO object from an earlier version of VxVM, DCO volume and snap objects from the volume:

   ```
   # vxassist [-g diskgroup] remove log volume logtype=dco
   ```

6 Use the following command on the volume to upgrade it:

```
# vxsnap [-g diskgroup] prepare volume \
alloc="disk_name1,disk_name2"
```

Provide two disk names to avoid overlapping the storage of the snapshot DCO plex with any other non-moving data or DCO plexes.

The `vxsnap prepare` command automatically enables persistent FastResync on the volume and on any snapshots that are generated from it. It also associates a DCO and DCO log volume with the volume to be snapshot.

7 To view the existing DCO plexes and see whether there are enough for the existing data plexes, enter:

```
# vxprint -g diskgroup
```

There needs to be one DCO plex for each existing data plex.

8 If there are not enough DCO plexes for the existing data plexes, create more DCO plexes:

```
# vxsnap [-g diskgroup] addmir dco_volume_name [alloc=disk_name]
```

where *dco_volume_name* is the name of the DCO volume you are creating.

9 If the plex is in a `SNAPDONE` state, convert it to an `ACTIVE` state:

```
# vxplex [-g diskgroup] convert state=ACTIVE data_plex
```

10 Convert the data plexes to a `SNAPDONE` state and associate a DCO plex with the data plex that will be used for snapshot operations:

```
# vxplex [-g diskgroup] -o dcoplex=dco_plex_name convert \
  state=SNAPDONE data_plex
```

where *dco_plex_name* is the name of the DCO plex you are creating.

In this example, the volume, `data_vol`, is upgraded to make use of VxVM 5.0 features.

Upgrade the disk group, `PRODdg`:

```
# vxdg upgrade PRODdg
```

Remove the DRL plexes or subdisks, belonging to an earlier version of VxVM, from the volume to be upgraded:

```
# vxassist -g PRODdg remove log data_vol logtype=drl
```

Reattach any snapshot volume back to the primary volume to be upgraded:

```
# vxsnap -g PRODdg snapback SNAP-data_vol
```

Turn off FastResync on the volume to be upgraded:

```
# vxvol -g PRODdg set fastresync=off data_vol
```

Disassociate and remove any older DCO object and DCO volumes:

```
# vxassist -g PRODdg remove log  data_vol logtype=dco
```

Upgrade the volume by associating a new DCO object and DCO volume:

```
# vxsnap -g PRODdg prepare data_vol alloc="PRODdg01 PRODdg02"
```

View the existing DCO plexes and plex state.

In this example, there are enough DCO plexes for the data plexes. Also, no data plex is associated with a DCO plex.

```
# vxprint -g PRODdg
```

```
v  data_vol     fsgen           ENABLED   4194304 -        ACTIVE   -       -
pl data_vol-01  data_vol        ENABLED   4194304 -        ACTIVE   -       -
sd PRODdg01-01  data_vol-01     ENABLED   4194304 0        -        -       -
pl data_vol-04  data_vol        ENABLED   4194304 -        SNAPDONE -       -
sd PRODdg02-03  data_vol-04     ENABLED   4194304 0        -        -       -
dc data_vol_dco data_vol                  ----    -        -
v  data_vol_dcl gen             ENABLED   560     -        ACTIVE   -       -
pl data_vol_dcl-01 data_vol_dcl ENABLED   560     -        ACTIVE   -       -
sd PRODdg01-02  data_vol_dcl-01 ENABLED   560     0        -        -       -
pl data_vol_dcl-02 data_vol_dcl ENABLED   560     -        ACTIVE   -       -
sd PRODdg02-02  data_vol_dcl-02 ENABLED   560     0        -        -       -
```

Convert the data plex state from `SNAPDONE` to `ACTIVE`:

```
# vxplex -g PRODdg convert state=ACTIVE data_vol-04
```

Associate the data plex with a new DCO plex and convert it back to a `SNAPDONE` state:

```
# vxplex -g PRODdg -o dcoplex=data_vol_dcl-02 convert \
state=SNAPDONE data_vol-04

# vxprint -g PRODdg
```

```
pl data_vol-03  -               DISABLED 4194304  -       -        -       -
sd PRODdg02-01  data_vol-03     ENABLED  4194304  0       -        -       -
v  data_vol     fsgen           ENABLED  4194304  -       ACTIVE   -       -
pl data_vol-01  data_vol        ENABLED  4194304  -       ACTIVE   -       -
sd PRODdg01-01  data_vol-01     ENABLED  4194304  0       -        -       -
pl data_vol-04  data_vol        ENABLED  4194304  -       SNAPDONE -       -
sd PRODdg02-03  data_vol-04     ENABLED  4194304  0       -        -       -
dc data_vol_dco data_vol        -        -        -       -        -       -
v  data_vol_dcl gen             ENABLED  560      -       ACTIVE   -       -
pl data_vol_dcl-01 data_vol_dcl ENABLED  560      -       ACTIVE   -       -
sd PRODdg01-02  data_vol_dcl-01 ENABLED  560      0       -        -       -
pl data_vol_dcl-02 data_vol_dcl DISABLED 560      -       DCOSNP   -       -
sd PRODdg02-02  data_vol_dcl-02 ENABLED  560      0       -        -       -
```

In this example, there are fewer DCO plexes than data plexes.

```
# vxprint -g PRODdg
```

```
pl data_vol-03  -               DISABLED    4194304     -   -        -       -
sd PRODdg02-01  data_vol-03     ENABLED     4194304     0   -        -       -
v  data_vol     fsgen           ENABLED     4194304     -   ACTIVE   -       -
pl data_vol-01  data_vol        ENABLED     4194304     -   ACTIVE   -       -
sd PRODdg01-01  data_vol-01     ENABLED     4194304     0   -        -       -
pl data_vol-04  data_vol        ENABLED     4194304     -   ACTIVE   -       -
sd PRODdg02-03  data_vol-04     ENABLED     4194304     0   -        -       -
dc data_vol_dco data_vol        ----        -           -
v  data_vol_dcl gen             ENABLED     560         -   ACTIVE   -       -
pl data_vol_dcl-01 data_vol_dcl ENABLED     560         -   ACTIVE   -       -
sd PRODdg01-02  data_vol_dcl-01 ENABLED     560         0   -        -       -
```

Add a DCO plex to the DCO volume using the `vxassist mirror` command:

```
# vxsnap -g PRODdg addmir data_vol_dcl alloc=PRODdg02
```

Associate the data plex with the new DCO plex and convert it to a `SNAPDONE` state:

```
# vxplex -g PRODdg -o dcoplex=data_vol_dcl-02 convert
state=SNAPDONE -V data_vol-04
```

```
# vxprint -g PRODdg
```

```
pl data_vol-03  -               DISABLED 4194304  -     -        -       -
v  data_vol     fsgen           ENABLED  4194304  -     ACTIVE   -       -
pl data_vol-01  data_vol        ENABLED  4194304  -     ACTIVE   -       -
sd PRODdg01-01  data_vol-01     ENABLED  4194304  0     -        -       -
pl data_vol-04  data_vol        ENABLED  4194304  -     SNAPDONE -       -
sd PRODdg02-03  data_vol-04     ENABLED  4194304  0     -        -       -
dc data_vol_dco data_vol        -        -        -     -        -       -
v  data_vol_dcl gen             ENABLED  60       -     ACTIVE   -       -
pl data_vol_dcl-01 data_vol_dcl ENABLED  560      -     ACTIVE   -       -
sd PRODdg01-02  data_vol_dcl-01 ENABLED  560      0     -        -       -
pl data_vol_dcl-02 data_vol_dcl DISABLED 560      -     DCOSNP   -       -
sd PRODdg02-02  data_vol_dcl-02 ENABLED  560      0     -        -       -
```

Summary of database snapshot steps

You can use Database FlashSnap commands to create a snapshot of your entire database on the same host or on a different one. Three types of snapshots can be created: `online_snapshot`, `online_mirror`, or `offline`.

If the `SNAPSHOT_MODE` specified in the snapplan is set to `online_snapshot`, `db2ed_vmsnap` first puts the database into `WRITESUSPEND` mode. After the database is in `WRITESUSPEND` mode, the snapshot volumes are created by breaking off the mirrors and splitting the snapshot volumes into a separate disk group. The split disk group will have the prefix specified in the snapplan. The database is taken out of `WRITESUSPEND` mode using the `write resume` command.

You can run the `db2ed_vmclonedb` command on the secondary host to import the disk group and create a clone database. You can run the `db2ed_vmsnap` command to synchronize the snapshot volumes with the primary database. The snapshot volumes can also be used to restore the primary database if it becomes corrupted. In this case, the `ALLOW_REVERSE_RESYNC` parameter must be set to `yes` in the snapplan.

If the `SNAPSHOT_MODE` is set to `online_mirror`, then `db2ed_vmsnap` puts the database into `WRITESUSPEND` mode. The snapshot volumes are created by breaking off the mirrors and splitting the snapshot volumes into a separate disk group. The split disk group has the prefix specified in the snapplan. The database is taken out of `WRITESUSPEND` mode using the `write resume` command. The database snapshot can be used to restore the primary database if it becomes corrupted.

If the `SNAPSHOT_MODE` is set to `offline`, the database must be inactive before the snapshot is created. The secondary host must be different than the primary host. Since the database is inactive, no active logs exist.

`online_snapshot`, `online_mirror`, and `offline` snapshots provide a valid backup image of the database. You can use the snapshot as a source for backing up the database or creating a clone database (only for `online_snapshot` or `offline`) for decision-support purposes.

By using Database FlashSnap commands, you can create snapshots of all volumes on a database using the snapplan. Optionally, you can use the VxVM command (`vxsnap`) to create volume snapshots. However, unlike the Database FlashSnap commands, the `vxsnap` command does not automate disk group content reorganization functions.

For more information about the `vxsnap` command, see *Veritas Volume Manager Administrator's Guide*.

Note: Make sure the volumes used by the database are configured properly before attempting to take a snapshot. This requires superuser (`root`) privileges.

Anytime you change the structure of the database (for example, by adding or deleting containers), you must run `db2ed_update`.

Database FlashSnap commands must be run by the DB2 instance owner.

To create a snapshot image of a database

1. Create a snapshot mirror of a volume or volume set.

   See "Creating a snapshot mirror of a volume or volume set used by the database" on page 194.

2. Use the `db2ed_vmchecksnap` command to create a snapplan template and check the volume configuration to ensure that it is valid for creating volume snapshots of the database.

   The snapplan contains detailed database and volume configuration information that is needed for snapshot creation and resynchronization. You can modify the snapplan template with a text editor.

   The `db2ed_vmchecksnap` command can also be used to:

   - List all snapplans associated with a specific DB2DATABASE (`db2ed_vmchecksnap -o list`).
   - Remove the snapplan from the repository (`db2ed_vmchecksnap -o remove -f SNAPPLAN`).
   - Copy a snapplan from the repository to your local directory (`db2ed_vmchecksnap -o copy -f SNAPPLAN`).

3. Use the command to create snapshot volumes for the database.

4 On the secondary host, use the `db2ed_vmclonedb` command to create a clone database using the disk group deported from the primary host.

   The `db2ed_vmclonedb` command imports the disk group that was deported from the primary host, recovers the snapshot volumes, mounts the file systems, recovers the database, and brings the database online. If the secondary host is different, the database name can be same. You can use the `-o recoverdb` option to let `db2ed_vmclonedb` perform an automatic database recovery, or you can use the `-o mount` option to perform your own point-in-time recovery and bring up the database manually. For a point-in-time recovery, the snapshot mode must be `online_snapshot`.

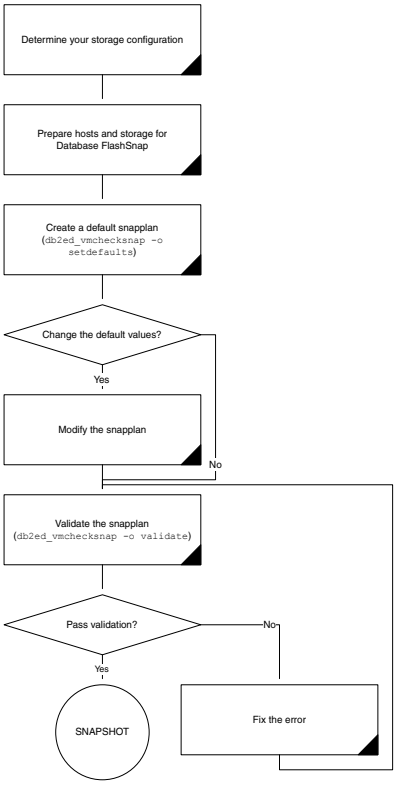
   You can also create a clone on the primary host. Your snapplan settings specify whether a clone should be created on the primary or secondary host.

5 You can now use the clone database to perform database backup and other off-host processing work.

6 The clone database can be used to reverse resynchronize the original volume from the data in the snapshot, or can be discarded by rejoining the snapshot volumes with the original volumes (that is, by resynchronizing the snapshot volumes) for future use.

Figure 11-3 depicts the sequence of steps leading up to taking a snapshot using Database FlashSnap.

Figure 11-3 Prerequisites for creating a snapshot of your database



There are many actions you can take after creating a snapshot of your database using Database FlashSnap. You can create a clone of the database for backup and off-host processing purposes. You can resynchronize the snapshot volumes with the primary database. In the event of primary database failure, you can recover it by reverse resynchronizing the snapshot volumes.

Figure 11-4 Actions you can perform if the SNAPSHOT_MODE is set to online_snapshot or offline

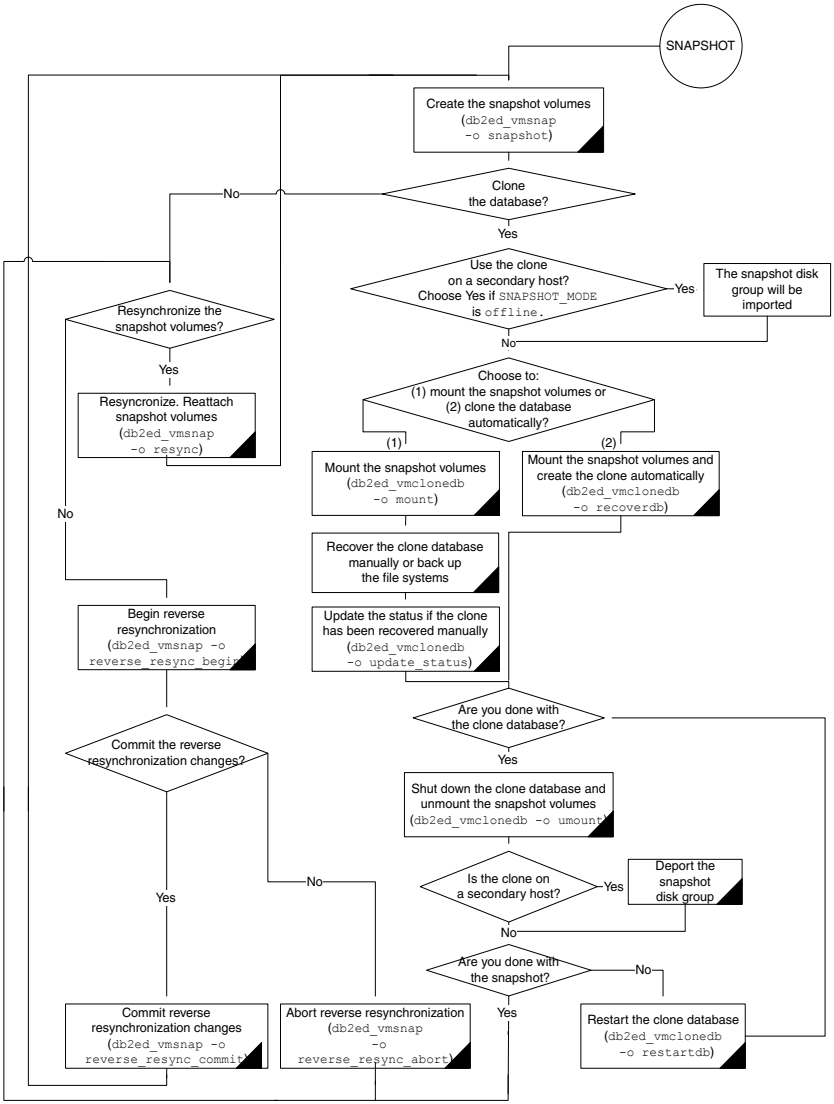
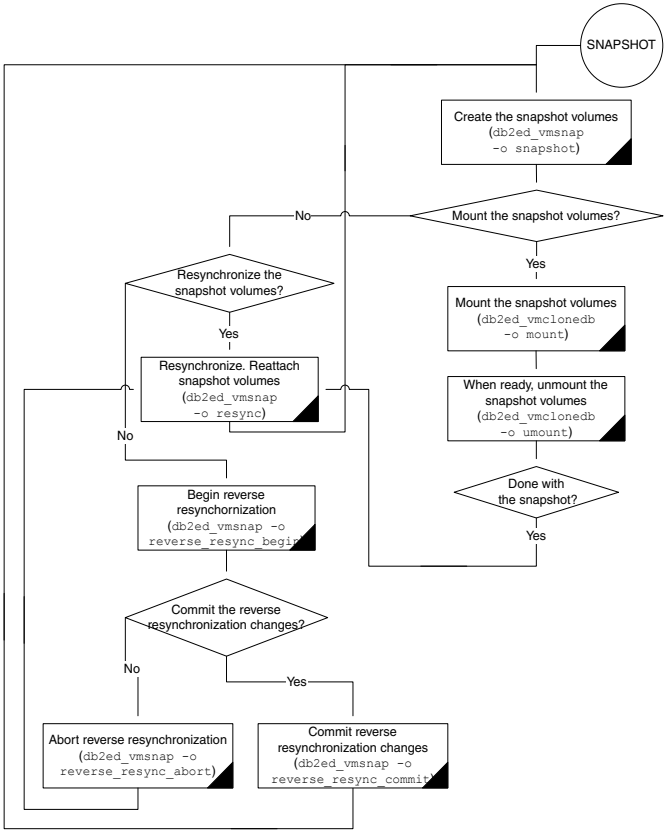


Figure 11-5 Actions you can perform if the SNAPSHOT_MODE is set to online_mirror



Creating a snapplan (db2ed_vmchecksnap)

The `db2ed_vmchecksnap` command creates a snapplan that `db2ed_vmsnap` uses to create a snapshot of a DB2 database. The snapplan specifies snapshot scenarios (such as `online_snapshot`, `online_mirror`, or `offline`).

You can name a snapplan file whatever you choose. Each entry in the snapplan file is a line in `parameter=argument` format.

Table 11-2 shows which parameters are set when using `db2ed_vmchecksnap` to create or validate a snapplan.

Table 11-2 Parameter values for db2ed_vmchecksnap

Parameter	Value
`SNAPSHOT_VERSION`	Specifies the snapshot version for this major release of Veritas Storage Foundation for DB2.
`PRIMARY_HOST`	The name of the host where the primary database resides.
`SECONDARY_HOST`	The name of the host where the database will be imported.
`PRIMARY_DG`	The name of the VxVM disk group used by the primary database.
`SNAPSHOT_DG`	The name of the disk group containing the snapshot volumes. The snapshot volumes will be put into this disk group on the primary host and deported. The secondary host can import this disk group to start a clone database.
`DB2DATABASE`	The name of the DB2 database.
`DB2HOME`	The home directory of the database.
`REDOLOG_DEST`	The full path of the redo logs.

Table 11-2 Parameter values for db2ed_vmchecksnap *(continued)*

Parameter	Value
SNAPSHOT_MODE	offline or online_snapshot or online_mirror Specifies whether the database should be offline, online_snapshot, or online_mirror when the snapshot is created. By default, the SNAPSHOT_MODE is online_snapshot. If the snapshot is created while the SNAPSHOT_MODE for the database is set to online_snapshot or online_mirror, the db2ed_vmsnap command will first put the database into WRITE SUSPEND mode. After db2ed_vmsnap finishes creating the snapshot, it will take the database out of WRITE SUSPEND mode. If the database is offline, it is not necessary to put the database into WRITE SUSPEND mode. If the SNAPSHOT_MODE is offline, the secondary host must be different than the primary host. If the SNAPSHOT_MODE is online_mirror, the primary and secondary host must be the same, and the ALLOW_REVERSE_RESYNC parameter must be set to yes in the snapplan. If the SNAPSHOT_MODE is online_snapshot, the clone can be created on either the primary or a secondary host.
SNAPSHOT_PLAN_FOR	The default value is database and cannot be changed.
SNAPSHOT_PLEX_TAG	Specifies the snapshot plex tag. Use this variable to specify a tag for the plexes to be snapshot. The maximum length of the plex_tag is 15 characters.
SNAPSHOT_MIRROR	Specifies the number of mirrors that will be part of the snapshot. The default value is 1.
SNAPSHOT_VOL_PREFIX	Specifies the snapshot volume prefix. Use this variable to specify a prefix for the snapshot volumes split from the primary disk group. A volume name cannot be more than 32 characters. You should consider the length of the volume name when assigning the prefix.

Table 11-2 Parameter values for db2ed_vmchecksnap *(continued)*

Parameter	Value
ALLOW_REVERSE_RESYNC	yes or no By default, reverse resynchronization is off (set equal to no) with the SNAPSHOT_MODE set to online_snapshot. If the SNAPSHOT_MODE is online_mirror, reverse resynchronization is set equal to yes. If ALLOW_REVERSE_RESYNC is set to yes, data from the snapshot volume can be used to overwrite the primary volume.

When you first run db2ed_vmchecksnap, use the -o setdefaults option to create a snapplan using default values for variables. You may then edit the file manually to set the variables for different snapshot scenarios.

Before creating a snapplan, make sure the following conditions have been met:

Prerequisites

- Storage must be configured properly.
  See "Hosts and storage for Database FlashSnap " on page 192.
- You must be the DB2 instance owner.
- The disk group must be version 110 or later. For more information on disk group versions, see the vxdg(1M) manual page.
- Be sure that a DCO and DCO volume are associated with the volume for which you are creating the snapshot.
- Snapshot plexes and their associated DCO logs should be on different disks than the original plexes, and should be configured correctly for creating snapshots by the system administrator.
- Persistent FastResync must be enabled on the existing database volumes and disks must be assigned for the snapshot volumes.
- The database must be running with LOGRETAIN mode on.

Usage notes

- The snapplan must be created on the primary host.
- After creating the snapplan using the db2ed_vmchecksnap command, you can use a text editor to review and update the file, if necessary.
- It is recommended that you create a local working directory to store your snapplans in.
- See the db2ed_vmchecksnap(1M) online manual page for more information.
- Database FlashSnap commands are not supported for multi-partitioned DB2 databases.

To create a snapplan

1 Change directories to the working directory you want to store your snapplan in.

```
$ cd /working_directory
```

2 Create a snapplan with default values using the `db2ed_vmchecksnap` command:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D DB2DATABASE \
-f SNAPPLAN -o setdefaults -t host_name -p PLEX_TAG
```

3 Open the snapplan file in a text editor and modify it as needed.

Note: The default setting for `SNAPSHOT_MODE` is `online_snapshot`.

In this example, a snapplan, `snap1`, is created for a snapshot image in a single-host configuration and default values are set. The host is named `host1` and the working directory is `/export/snap_dir`.

```
$ cd /export/snap_dir

$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap1 -o setdefaults \
-t host1 -p PRODtag

Snapplan snap1 for PROD.
=====================================================
SNAPSHOT_VERSION=5.0
PRIMARY_HOST=host1
SECONDARY_HOST=host1
PRIMARY_DG=PRODdg
SNAPSHOT_DG=SNAP_PRODdg
DB2DATABASE=PROD
DB2HOME=/PROD_HOME
REDOLOG_DEST=/PROD_HOME/inst1/NODE0000/SQL00001/SQLOGDIR/
SNAPSHOT_MODE=online_snapshot
SNAPSHOT_PLAN_FOR=database
SNAPSHOT_PLEX_TAG=PRODtag
SNAPSHOT_MIRROR=1
SNAPSHOT_VOL_PREFIX=SNAP_
ALLOW_REVERSE_RESYNC=no
```

In this example, a snapplan, `snap2`, is created for a snapshot image in a two-host configuration, and default values are set. The primary host is `host1`, the secondary host is `host2`, and the working directory is `/export/snap_dir`.

```
$ cd /export/snap_dir

$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap2 -o setdefaults \
-p PRODtag -t host2

Snapplan snap2 for PROD.
=====================================================
SNAPSHOT_VERSION=5.0
PRIMARY_HOST=host1
SECONDARY_HOST=host2
PRIMARY_DG=PRODdg
SNAPSHOT_DG=SNAP_PRODdg
DB2DATABASE=PROD
DB2HOME=/PROD_HOME
REDOLOG_DEST=/PROD_HOME/inst1/NODE0000/SQL00001/SQLOGDIR/
SNAPSHOT_MODE=online_snapshot
SNAPSHOT_PLAN_FOR=database
SNAPSHOT_PLEX_TAG=PRODtag
SNAPSHOT_MIRROR=1
SNAPSHOT_VOL_PREFIX=SNAP_
ALLOW_REVERSE_RESYNC=no
SNAPSHOT_MIRROR=1
```

Creating multi-mirror snapshots

To make Database Snapshots highly available, the snapped snapshot volume should contain more than one mirror. This makes the snapshot volumes available even if one of the mirrors gets disabled. Snapshot volumes can be mounted and the entire database snapshot is usable even if one of the mirror gets disabled. The multi-mirror snapshots are enabled with the `SNAPSHOT_MIRROR=`*n* keyword in the snapplan.

Note: Multi-mirror snapshots require no changes to the Command Line usage or arguments for the Flashsnap tools.

Before taking the snapshot, make sure all tagged snapshot mirrors are in the `SNAPDONE` state.

To create a multi-mirror snaphot

1 Add the second mirror and DCO log. When allocating storage for the second mirror and DCO logs, make sure the snap volumes are splittable. If snap volumes are not splittable, `db2ed_vmchecksnap` fails with appropriate errors.

```
# vxsnap -g dg_a addmir dg_a_vol1 alloc=dg_a03
```

2 Tag the newly added mirror with the same tag as that of the first mirror.

Assume that the volume has `fastresync = on`, has `dcolog = on`, and already has one `SNAPDONE` mirror and is tagged with `db2ed_flashsnap`.

```
# vxedit -g dg_a set putil2=db2ed_flashsnap dg_a_vol1-03
```

3 Add the `SNAPSHOT_MIRROR` keyword to the snapplan. Here is a sample snapplan.

```
SNAPSHOT_VERSION=5.0
PRIMARY_HOST=host1
SECONDARY_HOST=host1
PRIMARY_DG=PRODdg
SNAPSHOT_DG=SNAP_PRODdg
DB2DATABASE=PROD
REDOLOG_DEST=/prod_ar
SNAPSHOT_ARCHIVE_LOG=yes
SNAPSHOT_MODE=online
SNAPSHOT_PLAN_FOR=database
SNAPSHOT_PLEX_TAG=db2ed_flashsnap
SNAPSHOT_MIRROR=2
SNAPSHOT_VOL_PREFIX=SNAP_
ALLOW_REVERSE_RESYNC=no
```

Validating a snapplan (db2ed_vmchecksnap)

After creating a snapplan, the next steps are to validate the snapplan parameters and check whether the snapshot volumes have been configured correctly for creating snapshots. If validation is successful, the snapplan is copied to the repository. The snapplan is validated using the `db2ed_vmchecksnap` command with the `-o validate` option.

Consider the following prerequisites and notes before validating a snapplan:

Prerequisites

- The database must be up and running while executing the `db2ed_vmchecksnap` command.

Usage notes

- The `db2ed_vmchecksnap` command must be run as the DB2 instance owner.
- The default behavior is to force validation. Use the `-n` option if you want to skip validation.
- After validating the snapplan, you have the option of modifying the snapplan file to meet your storage configuration requirements.
- When using `db2ed_vmchecksnap` to validate the snapplan and storage, you can save the validation output. The system administrator can use this information to adjust the storage setup if the validation fails.
- If a snapplan is updated or modified, you must re-validate it. It is recommended that snapplans are revalidated when changes are made in the database disk group.
- The `db2ed_vmchecksnap` command must be used on the primary host.
- See the `db2ed_vmchecksnap`(1M) manual page for more information.

To validate a snapplan

1 Change directories to the working directory your snapplan is stored in:

```
$ cd /working_directory
```

2 Validate the snapplan using the `db2ed_vmchecksnap` command:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D DB2DATABASE \
-f SNAPPLAN -o validate
```

In an HA environment, you must modify the default snapplan, use the virtual host name defined for the resource group for the PRIMARY_HOST and/or SECONDARY_HOST, and run validation.

In the following example, a snapplan, `snap1`, is validated for a snapshot image in a single-host configuration. The primary host is `host1` and the working directory is `/export/snap_dir`.

```
$ cd /export/snap_dir

$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap1 -o validate$

SNAPSHOT_MODE is online_snapshot

PRIMARY_HOST is host1

SECONDARY_HOST is host1
```

```
The version of PRIMARY_DG-PRODdg is 140.

SNAPSHOT_DG is SNAP_PRODdg

SNAPSHOT_PLAN_FOR is database

Examining DB2 volume and disk layout for snapshot

Volume prodvol1 on PRODdg is ready for snapshot.
Original plex and DCO log for prodvol1 is on PRODdg1.

Snapshot plex and DCO log for prodvol2 is on PRODdg2.

Volume prodvol2 on PRODdg is ready for snapshot.
Original plex and DCO log for prodvol2 is on PRODdg1.

Snapshot plex and DCO log for prodvol2 is on PRODdg2.

SNAP_PRODdg for snapshot will include: PRODdg2

ALLOW_REVERSE_RESYNC is NO

The snapplan snap1 has been created.
```

In the following example, a snapplan, `snap2`, is validated for a snapshot image in a two-host configuration. The primary host is `host1`, the secondary host is `host2`, and the working directory is `/export/snap_dir`.

```
$ cd /export/snap_dir

$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap2 -o validate

SNAPSHOT_MODE is online_snapshot
PRIMARY_HOST is host1
SECONDARY_HOST is host2
The version of PRIMARY_DG-PRODdg is 140.
SNAPSHOT_DG is SNAP_PRODdg
SNAPSHOT_PLAN_FOR is database
Examining DB2 volume and disk layout for snapshot.
Volume prodvol1 on PRODdg is ready for snapshot.
Original plex and DCO log for arch is on PRODdg1.
Snapshot plex and DCO log for arch is on PRODdg2.
Volume prodvol2 on PRODdg is ready for snapshot.
Original plex and DCO log for prod_db is on PRODdg1.
Snapshot plex and DCO log for prod_db is on PRODdg2.
SNAP_PRODdg for snapshot will include: PRODdg2
ALLOW_REVERSE_RESYNC is NO
The snapplan snap2 has been created.
```

Displaying, copying, and removing a snapplan (db2ed_vmchecksnap)

Consider these notes before listing all snapplans for a specific DB2 database, displaying a snapplan file, or copying and removing snapplans.

Usage notes

- If the local snapplan is updated or modified, you must re-validate it.
- If the database schema or disk group is modified, you must revalidate it after running `db2ed_update`.

To list all available snapplans for a specific DB2 database

- Use the `db2ed_vmchecksnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D DB2DATABASE -o list
```

In the following example, all available snapplans are listed for the database `PROD`.

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -o list
 The following snapplan(s) are available for PROD:

 SNAP_PLAN     SNAP_STATUS      DB_STATUS     SNAP_READY
     snap1       init_full              -     yes
     snap2       init_full              -     yes
```

Note: The command output displays all available snapplans, their snapshot status (`SNAP_STATUS`), database status (`DB_STATUS`), and whether a snapshot may be taken (`SNAP_READY`).

See "Database FlashSnap snapshot status and database status" on page 335.

To display detailed information for a snapplan

- Use the `db2ed_vmchecksnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D DB2DATABASE \
  -f SNAPPLAN -o list
```

In the following example, the snapplan `snap1` is displayed.

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap1 -o list
SNAPSHOT_VERSION=4.0
PRIMARY_HOST=host1
SECONDARY_HOST=host1
PRIMARY_DG=PRODdg
SNAPSHOT_DG=SNAP_PRODdg
DB2DATABASE=PROD
DB2HOME=/PROD_HOME
REDOLOG_DEST=/PROD_HOME/inst1/NODE0000/SQL00001/SQLOGDIR/
SNAPSHOT_MODE=online_snapshot
SNAPSHOT_PLAN_FOR=database
SNAPSHOT_PLEX_TAG=PRODtag
SNAPSHOT_VOL_PREFIX=SNAP_
ALLOW_REVERSE_RESYNC=yes
SNAPSHOT_MIRROR=1
STORAGE_INFO
PRODdg02
SNAP_PLEX=prod_db-02

STATUS_INFO
SNAP_STATUS=init_full
DB_STATUS=init
LOCAL_SNAPPLAN=/export/snap_dir/snap1
```

To copy a snapplan from the repository to your current directory

- If you want to create a snapplan similar to an existing snapplan, you can simply create a copy of the existing snapplan and modify it. To copy a snapplan from the repository to your current directory, the snapplan must not already be present in the current directory.

  Use the `db2ed_vmchecksnap` command as follows:

  ```
  $ /opt/VRTS/bin/db2ed_vmchecksnap -D DB2DATABASE \
   -f SNAPPLAN -o copy
  ```

  This example shows the snapplan, `snap1`, being copied from the repository to the current directory.

  ```
  $ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap1 -o copy
  Copying 'snap1' to '/export/snap_dir'
  ```

To remove a snapplan from the repository

- A snapplan can be removed from a local directory or the repository if the snapplan is no longer needed.

  Use the `db2ed_vmchecksnap` command as follows:

  ```
  $ /opt/VRTS/bin/db2ed_vmchecksnap -D DB2DATABASE -f SNAPPLAN \
   -o remove
  ```

  This example shows the snapplan, `snap1`, being removed from the repository.

  ```
  $ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap1 -o remove
  The snapplan snap1 has been removed.
  ```

Creating a snapshot (db2ed_vmsnap)

The `db2ed_vmsnap` command creates a snapshot of a DB2 database by splitting the mirror volumes used by the database into a snapshot database. You can use the snapshot image on either the same host as the database or on a secondary host provided storage is shared by the two hosts.

The snapshot image created by `db2ed_vmsnap` is a frozen image of a DB2 database's containers.

Before creating a snapshot, make the sure the following conditions are met:

Prerequisites
- You must be logged in as the DB2 instance owner.
- You must create and validate a snapplan using `db2ed_vmchecksnap` before you can create a snapshot image with `db2ed_vmsnap`.

Usage notes
- The `db2ed_vmsnap` command can only be used on the primary host.
- When creating a snapshot volume, create the snapshot on a separate controller and on separate disks from the primary volume.
- The online redo log must be included in the snapshot if a clone database is to be started.
- Resynchronization speed varies based on the amount of data changed in both the primary and secondary volumes when the mirror is broken off.
- See the `db2ed_vmsnap`(1M) manual page for more information.

To create a snapshot

1. If `SNAPSHOT_MODE` is set to `offline` in the snapplan, terminate all connections to the database.
2. Create the snapshot image using the `db2ed_vmsnap` command:

```
$ /opt/VRTS/bin/db2ed_vmsnap -D DB2DATABASE -f SNAPPLAN \
 -o snapshot [-F]
```

Note: To force snapshot creation, use the `-F` option. The `-F` option can be used after a snapshot operation has failed and the problem was fixed without using Veritas Storage Foundation for DB2 commands. (That is, the volumes were synchronized without using Veritas Storage Foundation for DB2 commands.) In this situation, the status of the snapplan will appear as unavailable for creating a snapshot. The `-F` option ignores the unavailable status, checks for the availability of volumes, and creates the snapshot after the volumes pass the availability check.

After the snapshot is created, `db2ed_vmsnap` returns values you will need to run `db2ed_vmclonedb`. These values include the snapshot disk group, the snapplan name, and the primary database server name (*`server_name`*). Make a note of these values so you have them when running `db2ed_vmclonedb`. You can also use the command `db2ed_vmchecksnap -f` *`snapplan`* `-o list` to access the information regarding the snapshot disk group.

The snapshot volumes now represent a consistent backup copy of the database. You can back up the database by copying the snapshot volumes to tape or other backup media.

See "Backing up the database from snapshot volumes (db2ed_vmclonedb)" on page 228.

You can also create another DB2 database for decision-support purposes.

See "Cloning a database (db2ed_vmclonedb)" on page 232.

In this example, a snapshot image of the database, PROD, is created for a single-host configuration. In this case, the SECONDARY_HOST parameter is set the same as the PRIMARY_HOST parameter in the snapplan.

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap1 -o snapshot

db2ed_vmsnap started at 2006-03-10 13:10:54
DB20000I  The SET WRITE command completed successfully.
DB20000I  The SET WRITE command completed successfully.
A snapshot of DB2DATABASE PROD is in DG SNAP_PRODdg.
Snapplan snap1 is used for the snapshot.
DB2 server (server_name) is kagu

If -r <relocate_path> is used in db2ed_vmclonedb,
        make sure <relocate_path> is created and owned by
        DB2 Instance Owner.
        Otherwise, the following mount points need to be
        created and owned by DB2 Instance Owner:

        /PROD_HOME.
        /PROD_TBS.

db2ed_vmsnap ended at 2006-03-10 13:11:17
```

In this example, a snapshot image of the primary database, PROD, is created for a two-host configuration. In this case, the SECONDARY_HOST parameter specifies a different host name than the PRIMARY_HOST parameter in the snapplan.

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap2 -o snapshot

db2ed_vmsnap started at 2006-03-10 13:13:53
DB20000I  The SET WRITE command completed successfully.
DB20000I  The SET WRITE command completed successfully.
A snapshot of DB2DATABASE PROD is in DG SNAP_PRODdg.
Snapplan snap2 is used for the snapshot.
DB2 server (server_name) is kagu

If -r <relocate_path> is used in db2ed_vmclonedb,
        make sure <relocate_path> is created and owned by
```

```
                DB2 Instance Owner.
                Otherwise, the following mount points need to be
                created and owned by DB2 Instance Owner:

                /PROD_HOME.
                /PROD_TBS.

db2ed_vmsnap ended at 2006-03-10 13:14:16
```

Backing up the database from snapshot volumes (db2ed_vmclonedb)

Snapshots are most commonly used as a source for backing up a database. The advantage of using snapshot volumes is that the backup will not contest the I/O bandwidth of the physical devices. Making the snapshot volumes available on a secondary host will eliminate the extra loads put on processors and I/O adapters by the backup process on the primary host.

A clone database can also serve as a valid backup of the primary database. You can back up the primary datafiles to tape using snapshot volumes. You can also use the DB2 Backup utility to backup a snapshot database if all the tablespaces are DMS type.

Figure 11-6 shows a typical configuration when snapshot volumes are located on the primary host.

Figure 11-6 Example system configuration for database backup on the primary host

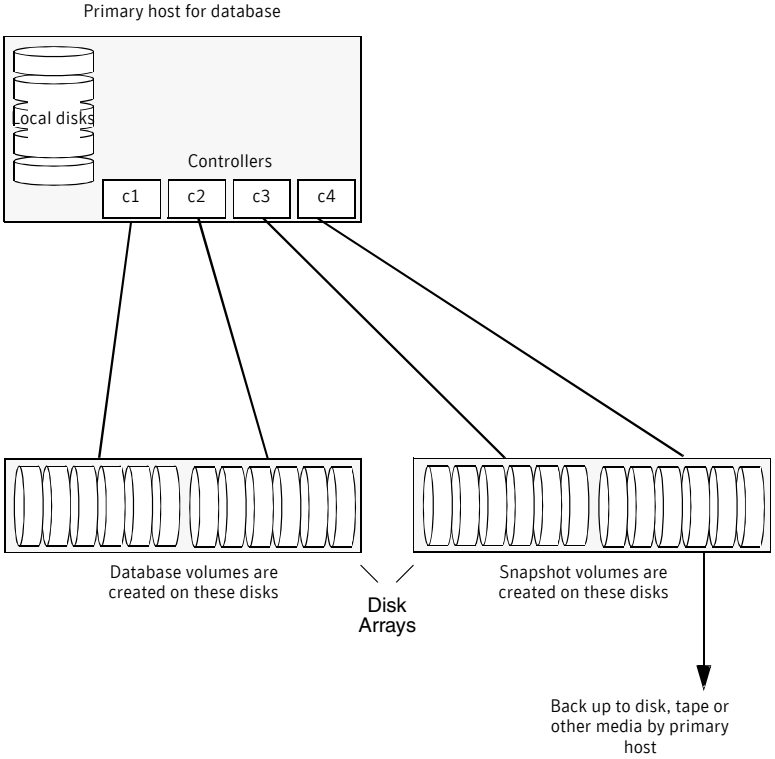
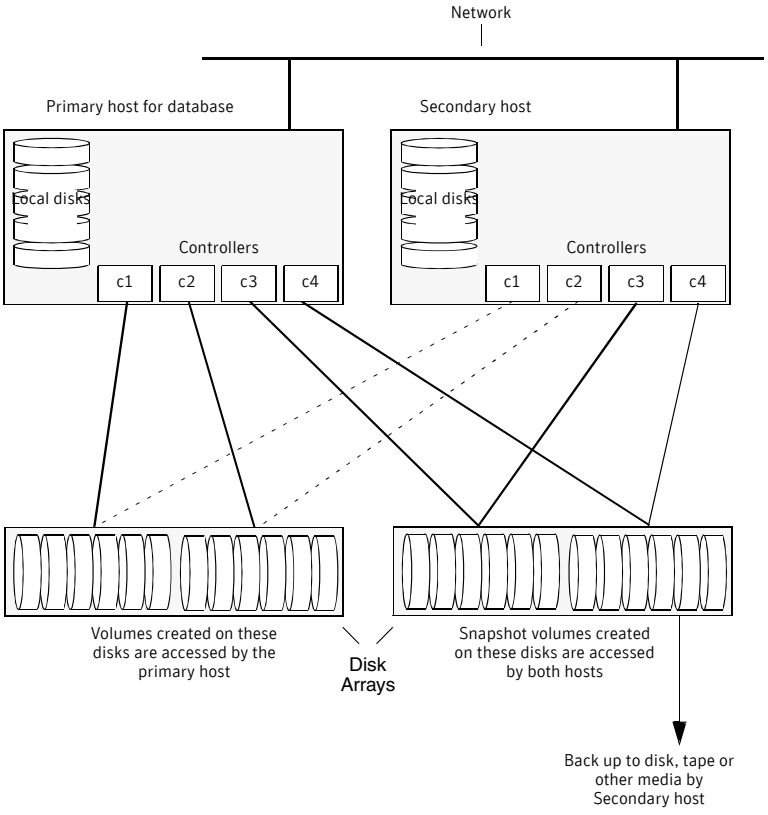


Figure 11-7 shows a typical configuration when snapshot volumes are used on a secondary host.

Figure 11-7 Example system configuration for database backup on a secondary host



Prerequisites

- You must be logged in as the DB2 instance owner to use `db2ed_vmclonedb` command.
- Before you can use the `db2ed_vmclonedb` command, you must create and validate a snapplan, and create a snapshot.
  See "Summary of database snapshot steps" on page 209.
  See "Validating a snapplan (db2ed_vmchecksnap)" on page 220.
  See "Creating a snapshot (db2ed_vmsnap)" on page 225.
- The volume snapshot must contain the entire database.
- Before you can use the `db2ed_vmclonedb` command with the `-r` *`relocate_path`* option (which specifies the initial mount point for the snapshot image), the system administrator must create the mount point and then change the owner to the DB2 instance owner.
- The `SNAPSHOT_MODE` must be `online_snapshot` or `offline`. If `SNAPSHOT_MODE` is set to `offline`, a two-host configuration is required.

Usage notes

- The `db2ed_vmclonedb` command can be used on the secondary host.
- In a single-host configuration, the primary and secondary hosts are the same.
- In a single-host configuration, `-r` *`relocate_path`* is required.
- See the `db2ed_vmclonedb`(1M) manual page for more information.

Mounting the snapshot volumes and backing up

Before using the snapshot volumes to do a backup, you must first mount them.

To mount the snapshot volumes

- Use the `db2ed_vmclonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D DB2DATABASE -g snap_dg \
-o mount,new_db=new_db,server_name=svr_name [-r relocate_path]
```

You can now back up an individual file or a group of files under a directory onto the backup media.

In this example, snapshot volumes are mounted.

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o mount,new_db=NEWPROD,server_name=kagu -f snap1 -r /clone

db2ed_vmclonedb started at 2006-03-10 13:21:32
Mounting /clone/PROD_HOME on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol1.
Mounting /clone/PROD_TBS on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol2.
```

```
If the database NEWPROD is recovered manually, you must run
db2ed_vmclonedb -o update_status to change the snapshot status.
db2ed_vmclonedb ended at 2006-03-10 13:22:08
```

Restoring from backup

Backup copies are used to restore volumes lost due to disk failure, or data destroyed due to human error. If a volume's data is corrupted and you know that you need to restore it from backup, you can use Database FlashSnap's reverse resynchronization function to restore the database.

Cloning a database (db2ed_vmclonedb)

Veritas Storage Foundation lets you create a clone database using snapshot volumes. You can use snapshots of a primary database to create a clone of the database at a given point in time. You can then implement decision-support analysis and report generation operations that take their data from the database clone rather than from the primary database to avoid introducing additional burdens on the production database.

A clone database can also serve as a valid back up of the primary database.

See "Backing up the database from snapshot volumes (db2ed_vmclonedb)" on page 228.

You can also back up the primary database to tape using snapshot volumes.

The resynchronization functionality of Database FlashSnap allows you to quickly refresh the clone database with up-to-date information from the primary database. Reducing the time taken to update decision-support data also lets you generate analysis reports more frequently.

Using Database FlashSnap to clone a database

In a single-host configuration, the `db2ed_vmclonedb` command creates a clone database on the same host. The command can also be used to shut down the clone database and unmount its file systems. When creating or unmounting the clone database in a single-host configuration, `-r relocate_path` is required so that the clone database's file systems use different mount points than those used by the primary database.

When used in a two-host configuration, the `db2ed_vmclonedb` command imports the snapshot disk group, mounts the file systems on the snapshot volumes, and starts a clone database. It can also reverse the process by shutting down the clone database, unmounting the file systems, and deporting the snapshot disk group.

Prerequisites

- You must be logged in as the DB2 instance owner.
- Before you can use the `db2ed_vmclonedb` command, review the procedure for taking a snapshot of a database, especially the steps required to validate a snapplan and to create the snapshot.
  See "Summary of database snapshot steps" on page 209.
  See "Validating a snapplan (db2ed_vmchecksnap)" on page 220.
  See "Creating a snapshot (db2ed_vmsnap)" on page 225.
- The volume snapshot must contain the entire database.
- The system administrator must provide the database administrator with access to the necessary volumes and mount points.
- Before you can use the `db2ed_vmclonedb` command with the `-r` *`relocate_path`* option (which specifies the initial mount point for the snapshot image), the system administrator must create the mount point and then change the owner to the DB2 instance owner.
- If `SNAPSHOT_MODE` is set to `offline`, a two-host configuration is required.

Usage notes

- The `db2ed_vmclonedb` command can be used on either the primary the secondary host as specified in the snapplan.
- In a single-host configuration, `-r` *`relocate_path`* is required. This command is also needed if the name of the clone database is different than the primary database.
- The `server_name=`*`svr_name`* option is required.
- See the `db2ed_vmclonedb`(1M) manual page for more information.

Note: When using the `-o recoverdb` option to clone a database, any database within the target instance that has the same name as the primary database to be cloned will be temporarily uncataloged and therefore unavailable.

Warning: If the primary database is being cloned on the same host and within the same instance, it will be temporarily uncataloged while `db2ed_vmclonedb` is running. The database will be recataloged on completion of `db2ed_vmclonedb`.

To mount a database and recover it manually

1 Start and mount the clone database to allow manual database recovery:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D DB2DATABASE -g snap_dg \
-o mount,new_db=db_name,server_name=svr_name -f SNAPPLAN \
[-r relocate_path]
```

2 Recover the database manually.

- Create a configuration file with content similar to the following:

```
DB_NAME=old_db,new_db
DB_PATH=old_dbpath,new_dbpath
INSTANCE=old_instance,new_instance
LOG_DIR=old_logdir,new_logdir
CONT_PATH=old_containerpath1,new_containerpath1
CONT_PATH=old_containerpath2,new_containerpath2
```

- Run the following command:

```
db2inidb newdb as snapshot relocate using configfile
```

  When this command is run, DB2 initiates a crash recovery and brings up the clone database.
  After this command is run, the primary database is uncataloged.

- Recatalog the primary database:

```
db2 catalog database DB2DATABASE on dpbath
```

For detailed information about manual database recovery, refer to your DB2 documentation.

3 Update the snapshot status information for the clone database in the SFDB repository:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -I inst01 \
-o update_status,new_db=newprod,server_name=kagu \
-f snap1 -r /clone
```

In this example, file systems are mounted without bringing up the clone database. The clone database must be manually created and recovered before it can be used. This example is for a clone created on the same host as the primary database.

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o mount,new_db=NEWPROD,server_name=kagu -f snap1 -r /clone

db2ed_vmclonedb started at 2006-04-01 13:30:12
Mounting /clone/testvol on /dev/vx/dsk/SNAP_PRODdg/SNAP_testvol.
Mounting /clone/udb_home on /dev/vx/dsk/SNAP_PRODdg/SNAP_udb_home.

If the database NEWPROD is recovered manually, you must run
db2ed_vmclonedb -o update_status to change the snapshot status.
db2ed_vmclonedb ended at 2006-04-01 13:30:24
```

The database is recovered manually using `db2inidb`.

The database status (`database_recovered`) needs to be updated for a clone database on the primary host after manual recovery has been completed.

```
$ /opt/VRTS/bin/db2ed_vmclonedb \
-o update_status,new_db=NEWPROD,server_name=kagu \
-f snap1 -r /clone

db2ed_vmclonedb started at 2006-04-01 13:37:09
The snapshot status has been updated.
db2ed_vmclonedb ended at 2006-04-01 13:37:23
```

In this example, file systems are mounted without recovering the clone database. The clone database must be manually recovered before it can be used. This example is for a clone created on a secondary host.

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o mount,new_db=NEWPROD,server_name=kagu -f snap2 -r /clone

db2ed_vmclonedb started at 2006-03-10 13:21:32
Mounting /clone/PROD_HOME on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol1.
Mounting /clone/PROD_TBS on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol2.

If the database NEWPROD is recovered manually, you must run
db2ed_vmclonedb -o update_status to change the snapshot status.
db2ed_vmclonedb ended at 2006-03-10 13:22:08
```

The database is recovered manually.

The snapshot status (`database_recovered`) is updated for a clone database on a secondary host after manual recovery has been completed.

```
$ /opt/VRTS/bin/db2ed_vmclonedb -o update_status,new_db=NEWPROD \
-f snap2

db2ed_vmclonedb started at 2006-03-10 13:39:49
The snapshot status has been updated.
db2ed_vmclonedb ended at 2006-03-10 13:40:01
```

To clone the database automatically

- Use the `db2ed_vmclonedb` command as follows:

  ```
  $ /opt/VRTS/bin/db2ed_vmclonedb -D DB2DATABASE -g snap_dg \
  -o recoverdb,new_db=db_name,server_name=svr_name -f SNAPPLAN \
  [-r relocate_path]
  ```

Where:

- *DB2DATABASE* is the name of the DB2 database used to create the snapshot.
- *snap_dg* is the name of the diskgroup that contains all the snapshot volumes.
-
- server_name=*svr_name* specifies the server on which the primary database resides.
- *SNAPPLAN* is the name of the snapplan file.
- *relocate_path* is the name of the initial mount point for the snapshot image.

Note: When cloning a database on a secondary host, ensure that PRIMARY_HOST and SECONDARY_HOST parameters in the snapplan file are different.

In the following example, a clone of the primary database is automatically created on the same host as the primary database.

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o recoverdb,new_db=NEWPROD,server_name=kagu -f snap1 -r /clone

db2ed_vmclonedb started at 2006-03-10 14:03:32
Mounting /clone/PROD_HOME on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol1.
Mounting /clone/PROD_TBS on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol2.
Files and control structures were changed successfully.
Database was catalogued successfully.
DBT1000I  The tool completed successfully.
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory
cache is refreshed.
DBT1000I  The tool completed successfully.
db2ed_vmclonedb ended at 2006-03-10 14:04:04
```

In the following example, a clone of the primary database is automatically created on a secondary host.

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o recoverdb,new_db=NEWPROD,server_name=kagu -f snap2 -r /clone

db2ed_vmclonedb started at 2006-03-10 14:03:32
Mounting /clone/PROD_HOME on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol1.
Mounting /clone/PROD_TBS on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol2.
```

```
Files and control structures were changed successfully.
Database was catalogued successfully.
DBT1000I  The tool completed successfully.
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory
cache is refreshed.
DBT1000I  The tool completed successfully.
db2ed_vmclonedb ended at 2006-03-10 14:04:04
```

Shutting down the clone database and unmounting file systems

When you are done using the clone database, you can shut it down and unmount all snapshot file systems with the `db2ed_vmclonedb -o umount` command.

Note: Any mounted Storage Checkpoints need to be unmounted before running `db2ed_vmclonedb -o umount`.

To shut down the clone database and unmount all snapshot file systems

- Use the `db2ed_vmclonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D DB2DATABASE \
-o umount,new_db=db_name,server_name=svr_name \
-f SNAPPLAN [-r relocate_path]
```

In this example, the clone database is shut down and file systems are unmounted for a clone on the same host as the primary database (a single-host configuration).

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -I inst01 \
-o umount,new_db=NEWPROD,server_name=kagu \
-f snap1 -r /clone

db2ed_vmclonedb started at 2006-03-10 14:44:42
DB20000I  The FORCE APPLICATION command completed successfully.
DB21024I  This command is asynchronous and may not be effective immediately

Umounting /clone/PROD_TBS.
Umounting /clone/PROD_HOME.
db2ed_vmclonedb ended at 2006-03-10 14:44:48
```

In this example, the clone database is shut down, file systems are unmounted, and the snapshot disk group is deported for a clone on a secondary host (a two-host configuration).

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -I inst01 \
-o umount,new_db=NEWPROD,server_name=kagu -f snap2 -r /clone

db2ed_vmclonedb started at 2006-03-10 15:02:33
DB20000I  The FORCE APPLICATION command completed successfully.
DB21024I  This command is asynchronous and may not be effective immediately.

Umounting /clone/PROD_TBS.
Umounting /clone/PROD_HOME.
db2ed_vmclonedb ended at 2006-03-10 15:02:39
```

Restarting a clone database

If the clone database is down as a result of using `db2ed_vmclonedb -o umount` or rebooting the system, you can restart it with the `-o restartdb` option.

Note: This option can only be used when a clone database is created successfully. If the clone database is recovered manually, `-o update_status` must be run to update the status before `-o restartdb` will work.

To start the clone database

- Use the `db2ed_vmclonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D DB2DATABASE -g snap_dg \
-o restartdb,new_db=db_name,server_name=svr_name -f SNAPPLAN \
[-r relocate_path]
```

In this example, the clone database is re-started on the same host as the primary database (a single-host configuration).

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o restartdb,new_db=NEWPROD,server_name=svr_name -f snap1 -r /clone

db2ed_vmclonedb started at 2006-03-10 14:47:36
Mounting /clone/PROD_HOME on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol1.
Mounting /clone/PROD_TBS on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol2.
Files and control structures were changed successfully.
Database was catalogued successfully.
DBT1000I  The tool completed successfully.
db2ed_vmclonedb ended at 2006-03-10 14:48:24
```

In this example, the clone database is re-started on the secondary host (a two-host configuration).

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o restartdb,new_db=NEWPROD,server_name=kagu -f snap1 -r /clone

db2ed_vmclonedb started at 2006-03-10 15:03:10
Mounting /clone/PROD_HOME on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol1.
Mounting /clone/PROD_TBS on /dev/vx/dsk/SNAP_PRODdg/SNAP_prodvol2.
Files and control structures were changed successfully.
Database was catalogued successfully.
DBT1000I  The tool completed successfully.
db2ed_vmclonedb ended at 2006-03-10 15:03:51
```

Resynchronizing the snapshot to your database

When you have finished using a clone database or want to refresh it, you can resynchronize it with the original database. This is also known as refreshing the snapshot volume or merging the split snapshot image back to the current database image. After resynchronizing, the snapshot can be retaken for backup or decision-support purposes.

There are two choices when resynchronizing the data in a volume:

- Resynchronizing the snapshot from the original volume.
- Resynchronizing the original volume from the snapshot. This choice is known as reverse resynchronization. Reverse resynchronization may be necessary to restore a corrupted database and is usually much quicker than using alternative approaches such as full restoration from backup media.

The recovery method is different depending on how the `SNAPSHOT_MODE` is set in the snapplan.

See "Snapshot mode" on page 191.

Before resynchroninzing the snapshot to your database, make sure the following conditions are met:

Prerequisites

- You must be logged in as the DB2 instance owner.
- Before you can resynchronize the snapshot image, you must validate a snapplan and create the snapshot.
  See "Summary of database snapshot steps" on page 209.
  See "Validating a snapplan (db2ed_vmchecksnap)" on page 220.
  See "Creating a snapshot (db2ed_vmsnap)" on page 225.
- If a clone database has been created, shut it down and unmount the file systems using the `db2ed_vmclonedb -o umount`command. This command also deports the disk group.
  See "Shutting down the clone database and unmounting file systems" on page 237.

Usage notes

- The `db2ed_vmsnap` command can only be executed on the primary host.
- In a two-host configuration, the `db2ed_vmsnap` command imports the disk group that was deported from the secondary host and joins the disk group back to the original disk group. The snapshot volumes again become plexes of the original volumes. The snapshot is then resynchronized.
- See the `db2ed_vmsnap`(1M) manual page for more information.

Note: If you plan to resynchronize the snapshot image immediately after taking a snapshot, wait for approximately a minute or two after taking the snapshot and before running `db2ed_vmsnap -o resync` to allow time for Veritas Enterprise Administrator to refresh the objects.

To resynchronize the snapshot image

- Use the `db2ed_vmsnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmsnap -D DB2DATABASE -f SNAPPLAN -o resync
```

In this example, the snapshot image is resynchronized with the primary database.

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap1 -o resync

db2ed_vmsnap started at 2006-03-10 14:57:36
The option resync has been completed.
db2ed_vmsnap started at 2006-03-10 14:58:05
```

Now you can again start creating snapshots.

Resynchronizing your database to the snapshot

If your database becomes corrupted, you can use reverse resynchronization to recover the database from a clone. The reverse resynchronization feature of Veritas Database FlashSnap enables you to resynchronize the primary database or volume with a clone database or snapshot volume.

Reverse resynchronization requires the primary database to be inactive so that it remains unchanged.

Warning: Upon completion of reverse resynchronization, the content of the original database is discarded. Storage Checkpoints taken on the original database before or after the snapshot was created are discarded. The `db2ed_vmsnap -o reverse_resync_commit` command cannot be undone and should be used with extreme caution.

Before resynchronizing a database to a snapshot, make sure the following conditions have been met:

Prerequisites

- You must be logged in as the DB2 instance owner.
  Before you can reverse resynchronize the snapshot image, review the database snapshot procedure and create the snapshot.
  See "Summary of database snapshot steps" on page 209.
  See "Creating a snapshot (db2ed_vmsnap)" on page 225.
- The mount point for the primary database must be owned by the DB2 instance owner.
- If a clone database has been created, you must shut it down and unmount the file systems using the command before you can reverse resynchronize the snapshot image. This command also deports the disk group.
  See "Shutting down the clone database and unmounting file systems" on page 237.
- The primary database must be inactive.

Usage notes

- The `db2ed_vmsnap` command can only be executed on the primary host.

To begin reverse resynchronization

◆ Use the `-o reverse_resync_begin` option to the `db2ed_vmsnap` command:

```
$ /opt/VRTS/bin/db2ed_vmsnap -D DB2DATABASE -f SNAPPLAN \
-o reverse_resync_begin
```

Any mounted storage checkpoints must be unmounted before running `db2ed_vmsnap -o reverse_resync`.

After executing `reverse_resync_commit`, checkpoints created on the original database will be deleted.

The `-o reverse_resync_begin` option imports the disk group that was deported from the secondary host and joins it back to the original disk group. The command unmounts the original volumes, mounts the snapshot volumes with the file systems that are configured for the primary database, and brings up the database snapshot image as the primary database. This operation requires the primary database to be offline so that its contents remain unchanged.

To abort reverse resynchronization

- Use the `-o reverse_resync_begin` option of the `db2ed_vmsnap` command:

  ```
  $ /opt/VRTS/bin/db2ed_vmsnap -D DB2DATABASE -f SNAPPLAN \
  -o reverse_resync_abort
  ```

  If your snapshot was created with `SNAPSHOT_MODE` set to `online_mirror`, you cannot run `db2ed_vmsnap -o reverse_resync_begin` after running `db2ed_vmsnap -o reverse_resync_abort`. You must take a new snapshot before performing reverse resynchronization again.

  The `-o reverse_resync_abort` option aborts `-o reverse_resync_begin`, unmounts the snapshot volumes, and mounts the original volumes back with the file systems that are configured to use the volume. This operation is only allowed after `-o reverse_resync_begin` has been executed and cannot be used after reverse resynchronization has been committed (`-o reverse_resync_commit`).

To commit reverse resynchronization changes

- Use the `-o reverse_resync_commit` option of the `db2ed_vmsnap` command:

  ```
  $ /opt/VRTS/bin/db2ed_vmsnap -D DB2DATABASE -f SNAPPLAN \
  -o reverse_resync_commit
  ```

  The `-o reverse_resync_commit` option commits the reverse resynchronization changes after you have verified that they are acceptable. The operation resynchronizes the original volume from the data in the snapshot.

This example is valid only for the `online_snapshot` mode.

Reverse resynchronization is started on the primary host:

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap1 \
-o reverse_resync_begin

db2ed_vmsnap started at 2006-05-26 13:37:11
Files and control structures were changed successfully.
Database was catalogued successfully.
DBT1000I  The tool completed successfully.
db2ed_vmsnap ended at 2006-05-26 13:37:48
```

Reverse resychronization is aborted on the primary host.

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap1 \
-o reverse_resync_abort
```

```
db2ed_vmsnap started at 2006-05-26 13:38:16
DB20000I  The FORCE APPLICATION command completed successfully.
DB21024I  This command is asynchronous and may not be effective immediately.

DB20000I  The UNCATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is r
Files and control structures were changed successfully.
Database was catalogued successfully.
DBT1000I  The tool completed successfully.
DB20000I  The UNCATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is r
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is r
The option reverse_resync_abort has been completed.
db2ed_vmsnap ended at 2006-05-26 13:39:31
```

Reverse resychronization changes are committed on the primary host.

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap1 \
-o reverse_resync_commit

db2ed_vmsnap started at 2006-05-26 13:40:32
DB20000I  The FORCE APPLICATION command completed successfully.
DB21024I  This command is asynchronous and may not be effective immediately.

DB20000I  The UNCATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is r
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is r
db2ed_vmsnap ended at 2006-05-26 13:41:35
```

Removing a snapshot volume

If a snapshot volume is no longer needed, you can remove it and free up the disk space for other uses by using the `vxedit rm` command.

Prerequisites

- You must be logged in as root.
- If the volume is on a mounted file system, you must unmount it before removing the volume.

To remove a snapplan and snapshot volume

1 To remove the snapshot and free up the storage used by it:

If the snapshot has been taken:

- Remove the snapshot as follows:

```
# vxsnap -g diskgroup dis snapshot_volume
# vxvol -g diskgroup stop snapshot_volume
# vxedit -g diskgroup -rf rm snapshot_volume
```

If the snapshot has not been taken and the snapshot plex (mirror) exists:

- Remove the snapshot as follows:

```
# vxsnap -g diskgroup rmmir volume
```

2 Remove the DCO and DCO volume:

```
# vxsnap -g diskgroup unprepare volume
```

3 Remove the snapplan:

```
# /opt/VRTS/bin/db2ed_vmchecksnap -D db -f snapplan -o remove
```

For example, the following commands will remove a snapshot volume from disk group `PRODdg`:

```
# vxsnap -g PRODdg dis snap_v1
# vxvol -g PRODdg stop snap_v1
# vxedit -g PRODdg -rf rm snap_v1
```

Using Database FlashSnap in an HA environment

Veritas Storage Foundation for DB2 supports Database FlashSnap in the HA environment.

When using Database FlashSnap in the HA environment, observe the following limitations:

- Modify the default snapplan to use the virtual host name defined for the database resource group for the `PRIMARY_HOST` and/or the `SECONDARY_HOST` parameters and validate the snapplan before creating a snapshot by running the following command:

```
db2ed_vmchecksnap -D DB2DATABASE -f SNAPPLAN \
-o validate
```

- The primary database must be down before you perform reverse resychronization (`db2ed_vmsnap -D DB2DATABASE -f SNAPPLAN -o reverse_resync_begin`).When Veritas Cluster Server (VCS) detects that the primary database is down, it starts the failover process and the repository is unmounted and the `db2ed_vmsnap` command is aborted.

To avoid the VCS failover process

1. As root, temporarily freeze the VCS resource group for the database:

   ```
   # hagrp -freeze ResourceGroup
   ```

2. Shut down the primary database.
3. Run reverse resynchronization:

   ```
   # db2ed_vmsnap -D DB2DATABASE -f SNAPPLAN -o \
      reverse_resync_begin
   ```

4. After reverse reynchronization changes are committed (`-o reverse_resync_commit`), verify that the database has been started in `ARCHIVELOG` mode. This information is provided in the status messages that appear after running committing reverse resynchronization changes.
5. Unfreeze the resource group:

   ```
   # hagrp -unfreeze ResourceGroup
   ```

Chapter 12

Using Veritas NetBackup for database backup

This chapter includes the following topics:

Components of Veritas NetBackup

Veritas NetBackup for DB2 has the following features:

- Media and device management
- Scheduling facilities
- Multiplexed backups and restores
- Transparent execution of both DB2 and regular file system backup and restore operations
- Shared devices and tapes used during other file backups
- Centralized and networked backup operations
- Parallel backup and restore operations
- Incremental backups of DB2 databases

About using Veritas NetBackup for backup and restore

With Veritas NetBackup, you can perform high performance, online (hot) backups of databases that must be available on a 24x7 basis. NetBackup supports the Extended Edition (EE) and the Enterprise Extended Edition (EEE) environments. NetBackup also supports DPF for DB2 8.1 and higher.

Veritas NetBackup lets you back up and restore database files and directories. You can set up schedules for automatic, unattended database backup, as well as full or incremental backup. These backups are managed entirely by the NetBackup server. You can also manually back up database files from any of the NetBackup clients. Client users can perform database backups and restores from their client systems on demand. The topics below summarize NetBackup's backup and restore capabilities for DB2. More information on backup and restore procedures and supported EE and EEE versions is available in the system administrator's guide.

See 'Using NetBackup for DB2 on UNIX' in the *Veritas NetBackup for DB2 System Administrator's Guide for UNIX*.

About performing a backup

There are two types of DB2 backups: database logs and archive logs. These two types of backups can be performed by NetBackup automatically, manually, or by using the `DB2 BACKUP DATABASE` command. More information on performing a backup is available in the system administrator's guide.

See 'Performing a Backup' in the *Veritas NetBackup for DB2 System Administrator's Guide for UNIX*.

About DB2 policy backups

You can back up a DB2 policy automatically or manually. The most convenient way to back up your database is to set up schedules for automatic backups.

About DB2 restore

The procedure for restoring a DB2 database depends on the database involved and the problems that you have on your system. You can browse the backups using the `db2 list history` command or using the NetBackup `bplist` command before restoring. More information and a complete description of how to recover a DB2 database is available in the system administrator's guide.

See the *DB2 UDB Administration Guide Data Recovery and High Availability Guide*.

About configuring Veritas NetBackup for a DB2 EEE (DPF) environment

Veritas NetBackup can be configured for DB2 in an Extended Edition (EE), Extended-Enterprise Edition (EEE), or Database Partitioning Feature (DPF) environment. Two types of DB2 backup policies are required. One is used to backup the catalog nodes and the other is used to backup all the nodes, including the catalog node. Detailed information and instructions on configuring DB2 for EEE is available in the system administrator's guide.

See 'Configuration for a DB2 EEE (DPF) Environment' in the *Veritas NetBackup for DB2 System Administrator's Guide for UNIX*.

About using Veritas NetBackup to backup and restore Quick I/O files

Veritas NetBackup does not follow symbolic links when backing up files. Typical backup management applications are designed this way to avoid backing up the same data twice. This would happen if both the link and the file it points to were included in the list of files to be backed up.

A Quick I/O file consists of two components: a hidden file with the space allocated for it, and a link that points to the Quick I/O interface of the hidden file. Because NetBackup does not follow symbolic links, you must specify both the Quick I/O link and its hidden file in the list of files to be backed up.

To view all files and their attributes in the db01 directory:

```
$ ls -la /db01

total 2192

drwxr-xr-x    2 root   root  96  Oct 20 17:39     .

drwxr-xr-x    9 root   root  8192  Oct 20 17:39     ..

-rw-r--r--    1 db2   dba  1048576   Oct 20 17:39    .dbfile

lrwxrwxrwx    1 db2   dba  22  Oct 20 17:39    dbfile ->\
.dbfile::cdev:vxfs:
```

In the example above, you must include both the symbolic link `dbfile` and the hidden file `.dbfile` in the file list of the backup class.

If you want to back up all Quick I/O files in a directory, you can simplify the process by just specifying the directory to be backed up. In this case, both components of

each Quick I/O file will be properly backed up. In general, you should specify directories to be backed up unless you only want to back up some, but not all files, in those directories.

Because NetBackup is tightly integrated with the Veritas Database Edition for DB2, NetBackup backs up extent attributes of a Quick I/O file and restores them accordingly. Quick I/O files can then be backed up and restored as regular files using NetBackup, while preserving the Quick I/O file's extent reservation. Without this feature, restoring the file could cause the loss of contiguous reservation, which can degrade performance.

When restoring a Quick I/O file, if both the symbolic link and the hidden file already exist, NetBackup will restore both components from the backup image. If either one of or both of the two components are missing, NetBackup creates or overwrites as needed.

Chapter 13

Tuning for performance

This chapter includes the following topics:

Additional documentation

Use the tuning tips and information provided in this chapter in conjunction with other more in-depth publications, such as:

- *IBM Configuration and Performance RedBooks* (IBM Corporation)
- *DB2 UDB V8.2 Performance Tuning Guide* (IBM Corporation)
- *DB2 High Performance Design and Tuning* (Prentice Hall)
- *Veritas Volume Manager Administrator's Guide*, chapter on "VxVM Performance Monitoring"

About tuning VxVM

Veritas Volume Manager (VxVM) is tuned for most configurations ranging from small systems to larger servers. On smaller systems with less than a hundred drives, tuning should not be necessary and Veritas Volume Manager should be capable of adopting reasonable defaults for all configuration parameters. On very large systems, however, there may be configurations that require additional tuning of these parameters, both for capacity and performance reasons.

For more information on tuning VxVM, see the *Veritas Volume Manager Administrator's Guide*.

About obtaining volume I/O statistics

If your database is created on a single file system that is on a single volume, there is typically no need to monitor the volume I/O statistics. If your database is created on multiple file systems on multiple volumes, or the volume configurations have changed over time, it may be necessary to monitor the volume I/O statistics for the databases.

Use the `vxstat` command to access information about activity on volumes, plexes, subdisks, and disks under VxVM control, and to print summary statistics to the standard output. These statistics represent VxVM activity from the time the system initially booted or from the last time the counters were reset to zero. If no VxVM object name is specified, statistics from all volumes in the configuration database are reported. Use the `-g` option to specify the database disk group to report statistics for objects in that database disk group.

VxVM records the following I/O statistics:

- count of operations
- number of blocks transferred (one operation can involve more than one block)
- average operation time (which reflects the total time through the VxVM interface and is not suitable for comparison against other statistics programs)

VxVM records the preceding three pieces of information for logical I/Os, including reads, writes, atomic copies, verified reads, verified writes, plex reads, and plex writes for each volume. VxVM also maintains other statistical data such as read failures, write failures, corrected read failures, corrected write failures, and so on. In addition to displaying volume statistics, the `vxstat` command is capable of displaying more detailed statistics on the components that form the volume. For detailed information on available options, refer to the `vxstat`(1M) manual page.

To reset the statistics information to zero, use the `-r` option. You can reset the statistics information for all objects or for only those objects that are specified. Resetting just prior to an operation makes it possible to measure the impact of that particular operation.

The following is an example of output produced using the `vxstat` command:

```
   OPERATIONS           BLOCKS           AVG TIME(ms)

TYP  NAME         READ     WRITE     READ      WRITE     READ     WRITE

vol  log2            0      6312        0      79836      .0       0.2
```

```
vol  db02     2892318   3399730  0283759    7852514    20.6       25.5
```

Additional information is available on how to use the `vxstat` output to identify volumes that have excessive activity and how to reorganize, change to a different layout, or move these volumes.

Additional volume statistics are available for RAID-5 configurations.

See the `vxstat`(1M) manual page.

See the "Performance Monitoring" section of the "Performance Monitoring and Tuning" chapter in the *Veritas Volume Manager Administrator's Guide*.

About tuning VxFS

Veritas File System provides a set of tuning options to optimize file system performance for different application workloads. VxFS provides a set of tunable I/O parameters that control some of its behavior. These I/O parameters help the file system adjust to striped or RAID-5 volumes that could yield performance far superior to a single disk. Typically, data streaming applications that access large files see the largest benefit from tuning the file system.

Most of these tuning options have little or no impact on database performance when using Quick I/O. However, you can gather file system performance data when using Quick I/O, and use this information to adjust the system configuration to make the most efficient use of system resources.

How monitoring free space works

In general, VxFS works best if the percentage of free space in the file system is greater than 10 percent. This is because file systems with 10 percent or more of free space have less fragmentation and better extent allocation. Regular use of the `df` command to monitor free space is desirable. Full file systems may have an adverse effect on file system performance. Full file systems should therefore have some files removed or should be expanded.

See the `fsadm_vxfs`(1M) manual page.

About monitoring fragmentation

Fragmentation reduces performance and availability. Regular use of `fsadm`'s fragmentation reporting and reorganization facilities is therefore advisable.

The easiest way to ensure that fragmentation does not become a problem is to schedule regular defragmentation runs using the `cron` command.

Defragmentation scheduling should range from weekly (for frequently used file systems) to monthly (for infrequently used file systems). Extent fragmentation should be monitored with `fsadm` or the `df -os` commands.

There are three factors that can be used to determine the degree of fragmentation:

- Percentage of free space in extents that are less than eight blocks in length
- Percentage of free space in extents that are less than 64 blocks in length
- Percentage of free space in extents that are 64 or more blocks in length

An unfragmented file system will have the following characteristics:

- Less than 1 percent of free space in extents that are less than eight blocks in length
- Less than 5 percent of free space in extents that are less than 64 blocks in length
- More than 5 percent of the total file system size available as free extents that are 64 or more blocks in length

A badly fragmented file system will have one or more of the following characteristics:

- More than 5 percent of free space in extents that are less than 8 blocks in length
- More than 50 percent of free space in extents that are less than 64 blocks in length
- Less than 5 percent of the total file system size available as free extents that are 64 or more blocks in length

The optimal period for scheduling extent reorganization runs can be determined by choosing a reasonable interval, scheduling `fsadm` runs at the initial interval, and running the extent fragmentation report feature of `fsadm` before and after the reorganization.

The “before” result is the degree of fragmentation prior to the reorganization. If the degree of fragmentation approaches the percentages for bad fragmentation, reduce the interval between `fsadm`. If the degree of fragmentation is low, increase the interval between `fsadm` runs.

How tuning VxFS I/O parameters works

VxFS provides a set of tunable I/O parameters that control some of its behavior. These I/O parameters are useful to help the file system adjust to striped or RAID-5 volumes that could yield performance far superior to a single disk. Typically, data

streaming applications that access large files see the biggest benefit from tuning the file system.

If VxFS is being used with Veritas Volume Manager, the file system queries VxVM to determine the geometry of the underlying volume and automatically sets the I/O parameters. VxVM is queried by `mkfs` when the file system is created to automatically align the file system to the volume geometry. If the default alignment from `mkfs` is not acceptable, the `-o align=n` option can be used to override alignment information obtained from VxVM. The `mount` command also queries VxVM when the file system is mounted and downloads the I/O parameters.

If the default parameters are not acceptable or the file system is being used without VxVM, then the `/etc/vx/tunefstab` file can be used to set values for I/O parameters. The `mount` command reads the `/etc/vx/tunefstab` file and downloads any parameters specified for a file system. The `tunefstab` file overrides any values obtained from VxVM. While the file system is mounted, any I/O parameters can be changed using the `vxtunefs` command, which can have tunables specified on the command line or can read them from the `/etc/vx/tunefstab` file.

The `vxtunefs` command can be used to print the current values of the I/O parameters.

See the `vxtunefs`(1M) and `tunefstab`(4) manual pages.

About tunable VxFS I/O parameters

The following are tunable VxFS I/O parameters:

`read_pref_io` — The preferred read request size. The file system uses this parameter in conjunction with the `read_nstream` value to determine how much data to read ahead. The default value is 64K.

`write_pref_io` — The preferred write request size. The file system uses this parameter in conjunction with the `write_nstream` value to determine how to do flush behind on writes. The default value is 64K.

`read_nstream` — The number of parallel read requests of size `read_pref_io` that you can have outstanding at one time. The file system uses the product of `read_nstream` multiplied by `read_pref_io` to determine its read ahead size. The default value for `read_nstream` is 1.

`write_nstream` The number of parallel write requests of size `write_pref_io` that you can have outstanding at one time. The file system uses the product of `write_nstream` multiplied by `write_pref_io` to determine when to do flush behind on writes. The default value for `write_nstream` is 1.

`default_indir_ size` On VxFS, files can have up to ten variably sized direct extents stored in the inode. After these extents are used, the file must use indirect extents that are a fixed size. The size is set when the file first uses indirect extents. These indirect extents are 8K by default. The file system does not use larger indirect extents because it must fail a write and return `ENOSPC` if there are no extents available that are the indirect extent size. For file systems with many large files, the 8K indirect extent size is too small. Large files that require indirect extents use many smaller extents instead of a few larger ones. By using this parameter, the default indirect extent size can be increased so that large files in indirects use fewer large extents.

Be careful using this tunable. If it is too large, then writes fail when they are unable to allocate extents of the indirect extent size to a file. In general, the fewer and the larger the files on a file system, the larger the `default_indir_size` parameter can be. The value of this parameter is generally a multiple of the `read_pref_io` parameter.

This tunable is not applicable on Version 4 disk layouts.

`discovered_direct_iosz` Any file I/O requests larger than the `discovered_direct_iosz` are handled as discovered direct I/O. A discovered direct I/O is unbuffered similar to direct I/O, but does not require a synchronous commit of the inode when the file is extended or blocks are allocated. For larger I/O requests, the CPU time for copying the data into the page cache and the cost of using memory to buffer the I/O data becomes more expensive than the cost of doing the disk I/O. For these I/O requests, using discovered direct I/O is more efficient than regular I/O. The default value of this parameter is 256K.

initial_extent_ size	Changes the default initial extent size. VxFS determines the size of the first extent to be allocated to the file based on the first write to a new file. Normally, the first extent is the smallest power of 2 that is larger than the size of the first write. If that power of 2 is less than 8K, the first extent allocated is 8K. After the initial extent, the file system increases the size of subsequent extents (see max_seqio_extent_size) with each allocation. Since most applications write to files using a buffer size of 8K or less, the increasing extents start doubling from a small initial extent. initial_extent_size can change the default initial extent size to be larger, so the doubling policy will start from a much larger initial size and the file system will not allocate a set of small extents at the start of file. Use this parameter only on file systems that will have a very large average file size. On these file systems, it will result in fewer extents per file and less fragmentation. initial_extent_size is measured in file system blocks.
max_direct_iosz	The maximum size of a direct I/O request that will be issued by the file system. If a larger I/O request comes in, then it is broken up into max_direct_iosz chunks. This parameter defines how much memory an I/O request can lock at once, so it should not be set to more than 20 percent of memory.
max_diskq	Limits the maximum disk queue generated by a single file. When the file system is flushing data for a file and the number of pages being flushed exceeds max_diskq, processes will block until the amount of data being flushed decreases. Although this doesn't limit the actual disk queue, it prevents flushing processes from making the system unresponsive. The default value is 1MB.

`max_seqio_extent_size` Increases or decreases the maximum size of an extent. When the file system is following its default allocation policy for sequential writes to a file, it allocates an initial extent that is large enough for the first write to the file. When additional extents are allocated, they are progressively larger (the algorithm tries to double the size of the file with each new extent) so each extent can hold several writes' worth of data. This is done to reduce the total number of extents in anticipation of continued sequential writes. When the file stops being written, any unused space is freed for other files to use. Normally, this allocation stops increasing the size of extents at 2048 blocks, which prevents one file from holding too much unused space. `max_seqio_extent_size` is measured in file system blocks.

Enables or disables caching on Quick I/O files. The default behavior is to disable caching. To enable caching, set `qio_cache_enable` to 1. On systems with large memories, the database cannot always use all of the memory as a cache. By enabling file system caching as a second level cache, performance may be improved. If the database is performing sequential scans of tables, the scans may run faster by enabling file system caching so the file system will perform aggressive read-ahead on the files.

write_throttle

Warning: The write_throttle parameter is useful in special situations where a computer system has a combination of a lot of memory and slow storage devices. In this configuration, sync operations (such as fsync()) may take so long to complete that the system appears to hang. This behavior occurs because the file system is creating dirty pages (in-memory updates) faster than they can be asynchronously flushed to disk without slowing system performance.

Lowering the value of write_throttle limits the number of dirty pages per file that a file system will generate before flushing the pages to disk. After the number of dirty pages for a file reaches the write_throttle threshold, the file system starts flushing pages to disk even if free memory is still available. The default value of write_throttle typically generates a lot of dirty pages, but maintains fast user writes. Depending on the speed of the storage device, if you lower write_throttle, user write performance may suffer, but the number of dirty pages is limited, so sync operations will complete much faster.

Because lowering write_throttle can delay write requests (for example, lowering write_throttle may increase the file disk queue to the max_diskq value, delaying user writes until the disk queue decreases), it is recommended that you avoid changing the value of write_throttle unless your system has a a large amount of physical memory and slow storage devices.

If the file system is being used with VxVM, it is recommended that you set the VxFS I/O parameters to default values based on the volume geometry.

If the file system is being used with a hardware disk array or volume manager other than VxVM, align the parameters to match the geometry of the logical disk. With striping or RAID-5, it is common to set read_pref_io to the stripe unit size and read_nstream to the number of columns in the stripe. For striping arrays, use the same values for write_pref_io and write_nstream, but for RAID-5 arrays, set write_pref_io to the full stripe size and write_nstream to 1.

For an application to do efficient disk I/O, it should issue read requests that are equal to the product of read_nstream multiplied by read_pref_io. Generally, any multiple or factor of read_nstream multiplied by read_pref_io should be a good size for performance. For writing, the same rule of thumb applies to the write_pref_io and write_nstream parameters. When tuning a file system, the best thing to do is try out the tuning parameters under a real-life workload.

If an application is doing sequential I/O to large files, it should issue requests larger than the `discovered_direct_iosz`. This causes the I/O requests to be performed as discovered direct I/O requests, which are unbuffered like direct I/O but do not require synchronous inode updates when extending the file. If the file is too large to fit in the cache, then using unbuffered I/O avoids throwing useful data out of the cache and lessons CPU overhead.

About obtaining file I/O statistics using the Quick I/O interface

The `qiostat`command provides access to activity information on Quick I/O files on VxFS file systems. The command reports statistics on the activity levels of files from the time the files are first opened using their Quick I/O interface. The accumulated `qiostat` statistics are reset once the last open reference to the Quick I/O file is closed.

The qiostat command displays the following I/O statistics:

- Number of read and write operations
- Number of data blocks (sectors) transferred
- Average time spent on read and write operations

When Cached Quick I/O is used, `qiostat` also displays the caching statistics when the `-l` (the long format) option is selected.

The following is an example of `qiostat` output:

```
                    OPERATIONS        FILE BLOCKS             AVG TIME(ms)

 FILENAME          READ      WRITE     READ      WRITE       READ      WRITE

 /db01/file1          0         00        0        0.0       0.0

 /db01/file2          0         00        0        0.0       0.0

 /db01/file3      73017     181735   718528    1114227      26.8       27.9

 /db01/file4      13197      20252   105569     162009      25.8      397.0

 /db01/file5          0         00        0        0.0       0.0
```

For detailed information on available options, see the `qiostat`(1M) manual page.

About I/O statistics data

Once you gather the file I/O performance data, you can use it to adjust the system configuration to make the most efficient use of system resources.

There are three primary statistics to consider:

- file I/O activity
- volume I/O activity
- raw disk I/O activity

If your database is using one file system on a striped volume, you may only need to pay attention to the file I/O activity statistics. If you have more than one file system, you may need to monitor volume I/O activity as well.

First, use the `qiostat -r` command to clear all existing statistics. After clearing the statistics, let the database run for a while during a typical database workload period. For example, if you are monitoring a database with many users, let the statistics accumulate for a few hours during prime working time before displaying the accumulated I/O statistics.

To display active file I/O statistics, use the `qiostat` command and specify an interval (using `-i`) for displaying the statistics for a period of time. This command displays a list of statistics such as:

```
                    OPERATIONS        FILE BLOCKS          AVG TIME(ms)

FILENAME          READ      WRITE     READ      WRITE      READ      WRITE

/db01/cust1        218         36      872        144      22.8       55.6

/db01/hist1          0         10        4        0.0      10.0

/db01/nord1         10         14       40         56      21.0       75.0

/db01/ord1          19         16       76         64      17.4       56.2

/db01/ordl1        189         41      756        164      21.1       50.0

/db01/roll1          0         50        0        200       0.0       49.0

/db01/stk1        1614        238     6456        952      19.3       46.5

/db01/sys1           0         00        0        0.0       0.0

/db01/temp1          0         00        0        0.0       0.0

/db01/ware1          3         14       12         56      23.3       44.3

/logs/log1           0         00        0        0.0       0.0

/logs/log2           0        217 0   2255        0.0       6.8
```

File I/O statistics help identify files with an unusually large number of operations or excessive read or write times. When this happens, try moving the “hot” files or busy file systems to different disks or changing the layout to balance the I/O load.

```
Mon May 11 16:21:20 2015

/db/dbfile01               813     0      813       0      0.3     0.0

/db/dbfile02                 0   813        0     813      0.0     5.5

Mon May 11 16:21:25 2015

/db/dbfile01               816     0      816       0      0.3     0.0

/db/dbfile02                 0   816        0             816     0.0     5.3

Mon May 11 16:21:30 2015

/db/dbfile01                 0     0        0               0     0.0     0.0

/db/dbfile02                 0     0        0               0     0.0     0.0
```

About I/O statistics

When running your database through the file system, the read-write lock on each file allows only one active write per file. When you look at the disk statistics using `iostat`, the disk reports queueing time and service time. The service time is the time that I/O spends on the disk, and the queueing time is how long it waits for all of the other I/Os ahead of it. At the volume level or the file system level, there is no queueing, so `vxstat`and `qiostat`do not show queueing time.

For example, if you send 100 I/Os at the same time and each takes 10 milliseconds, the disk reports an average of 10 milliseconds of service and 490 milliseconds of queueing time. The `vxstat`and `qiostat`report an average of 500 milliseconds service time.

About tuning DB2 databases

To achieve optimal performance on your DB2 database, the database needs to be tuned to work with VxFS. There are a number of DB2 parameters that you can tune to improve your DB2 database performance.

DB2_USE_PAGE_CONTAINER_TAG

By default, DB2 stores a container tag in the first extent of each DMS container, whether it is a file or a device. The container tag is the metadata for the container. (Before DB2 v8.1, the container tag was stored in a single page, so it required less space in the container.) It is recommended that you keep this variable set to OFF.

The DB2_USE_PAGE_CONTAINER_TAG variable is set using the db2set command.

```
$ db2set DB2_USE_PAGE_CONTAINER_TAG=OFF
$ db2stop ;
$ db2start
```

If you set this registry variable to ON when you use RAID devices for containers, I/O performance might degrade. Because for RAID devices you create table spaces with an extent size equal to or a multiple of the RAID stripe size, setting the DB2_USE_PAGE_CONTAINER_TAG to ON causes the extents not to line up with the RAID stripes. As a result, an I/O request might need to access more physical disks than would be optimal. Users are strongly advised against enabling this registry variable.

DB2_PARALLEL_IO

This setting is used to force parallel I/O to occur on tablespaces. This is important in combination with the DB2_STRIPED_CONTAINERS setting, as RAID devices have more than one physical disk and therefore can sustain a greater I/O load than non-RAID devices. DB2 achieves this parallelism by enabling multiple prefetch threads on enabled tablespaces.

The DB2_PARALLEL_IO variable is set using the db2set command. To enable parallel I/O on all tablespaces, you would run the commands:

```
$ db2set DB2_PARALLEL_IO=*
$ db2stop ; db2start
```

To enable parallel I/O on a subset of all tablespaces, you need to know the tablespace identifying number and supply a list of tablespace ids, comma separated, to the db2set command:

```
$ db2 connect to PROD
$ db2 list tablespaces
$ db2 terminate
$ db2set DB2_PARALLEL_IO=3,4,8,9
$ db2stop ; db2start
```

As per the examples, you must stop and restart your instance after modifying the DB2_PARALLEL_IO setting. It is also recommended that DB2_PARALLEL_IO be enabled for tablespaces residing on RAID devices when PREFETCHSIZE > EXTENTSIZE.

PREFETCHSIZE and EXTENTSIZE

Prefetching is a behavior that increases database performance in DSS type environments, or environments where data are large enough that they cannot be maintained in the database memory. The extentsize is important in environments where DB2 tablespaces and containers reside upon RAID devices. In general, the EXTENTSIZE should always be equal to or a multiple of the RAID stripe size

By setting DB2_PARALLEL_IO, the tablespace PREFETCHSIZE takes on special meaning. PREFETCHSIZE is divided by the EXTENTSIZE to arrive at the degree of I/O parallelism. Without this environment variable set, the degree of I/O parallelism is normally derived from the number of containers. Because RAID often has only one container, it is important to set the PREFETCHSIZE as a multiple of the EXTENTSIZE, to provide a sufficient number of IO_SERVERS (at least one per physical disk), and to assign the tablespace to a bufferpool that is sufficiently large to accommodate to prefetch requests.

In the general case, we calculate EXTENTSIZE based on the physical attributes of the volume. PREFETCHSIZE should be at least EXTENTSIZE * the number of containers in order to obtain a good I/O parallelism. When dealing with RAID devices however, we may have only a single container within a tablespace and so the number of containers would be substituted with the number of devices or columns in the volume.

When using DMS device containers, such as Quick I/O files, the operating system does not perform any prefetching or caching.

See “About Quick I/O” on page 71.

When you need to have a greater control over when and where memory is allocated to caching and prefetching of DB2 tablespace data, use Cached Quick I/O.

See “About Cached Quick I/O” on page 93.

If you prefer to assign more system memory permanently to DB2 bufferpools, set PREFETCHSIZE and the DB2_PARALLEL_IO settings for tablespaces.

For example, we have a VxVM RAID0 volume striped across 10 physical disks with a stripe column size of 64k. We have created a VxFS file system on this volume and are about to create a tablespace of DMS containers:

```
qiomkfile -s 1G /db2_stripe/cont001
```

```
db2 connect to PROD
$ db2 create tablespace DATA1 managed by database \

using (device '/db2_stripe/cont001' 128000 ) \
pagesize 8k extentsize 8 prefetchsize 80

$ db2 terminate
```

In this example, we ensure that each read of an extent will span 1 physical drive (column width is 64k and our extentsize is 8 * 8k pagesize). When prefetching, we take a full stripe read at a time (there are 10 disks in the stripe, so 10 * an extent is 80 pages). Observe that the PREFETCHSIZE remains a multiple of the EXTENTSIZE. These settings would provide a good environment for a database which in general uses clusters of data around 640k or less. For larger database objects or more aggressive prefetch on data, the specified PREFETCHSIZE can be multiplied.

If the database's main workload requires good sequential I/O performance, such as a DSS workload, then the settings for Cached Quick I/O and PREFETCHSIZE becomes even more important.

There are some cases where setting the PREFETCHSIZE to large values or having prefetching at all may degrade performance. In OLTP environments where data access is very random, you may need to turn off prefetching on a tablespace, or minimize the effect by setting PREFETCHSIZE equal to EXTENTSIZE.

It is still very important in these types of environment to ensure that access to indexes is very fast and preferably all heavily accessed indexes are cached by Cached Quick I/O or in bufferpool memory.

INTRA_PARALLEL

The INTRA_PARALLEL setting is usually set on machines with multiple CPUs when large and complex queries are being executed. This may not provide any performance advantage in OLTP environments, as queries in these types of environments are normally very simple, short and highly repetitive. However, for DSS or OLAP environments, enabling this option may provide significant performance improvements.

NUM_IOCLEANERS

Specifies the number of async page cleaners. The cleaners flush dirty pages from the buffer pool, freeing the space for the threads pulling data in from storage. Important to tune this if the PREFETCH settings for the database are being modified. To avoid I/O wait, set this parameter higher if insert/update/delete is heavy or prefetch large.

NUM_IOSERVERS

Specifies the number of I/O servers for the database. These servers implement prefetch and async I/O operations. Should be set to at least the number of physical devices on the host system in order to maximize I/O parallelism.

CHNGPGS_THRESH

Specifies the threshold at which the IOCLEANERS start flushing dirty pages. A lower value indicates that cleaning should being earlier.

Table scans

Quick I/O in its default mode performs all I/O as direct I/O.

In the case of single-threaded sequential scans (common in decision support system (DSS) workloads), using buffered reads can yield better performance. Because the file system detects these sequential reads and performs read-aheads, the next few blocks that are requested by DB2 are readily available in the system buffer cache and are simply copied to the DB2 buffer pool. Because access from memory is inherently faster than access from disk, this achieves a significant reduction in response time.

To handle large sequential scans when using Quick I/O, two methods are available to improve performance:

- Modify the DB2 PREFETCH setting to force reading in data before it is required.
- The second method is to enable Cached Quick I/O for the files that would be read by the DB2 sequential scan process. Cached Quick I/O enables buffered reads, and the automatic file system read-ahead helps lower response times by pre-loading data. A major advantage of using Cached Quick I/O is that it does not require any database level changes and so does not require the database to be restarted for changes to take effect.

Asynchronous I/O

Asynchronous I/O allows the DB2 database to schedule multiple I/Os without waiting for the I/O to complete. When the I/O completes, the kernel notifies the `DB2` using an interrupt.

Quick I/O supports kernel asynchronous I/O (KAIO), which reduces CPU utilization and improves transaction throughput. The DB2 database engine, by default, will make use of asynchronous I/O when using DMS containers.

Buffer pools

The UNIX buffer cache plays an important role in performance when using UFS, HFS, or JFS in buffered I/O mode. However, when using Quick I/O, the database buffer pools must be tuned as if raw devices are being used. You can allocate more memory to the database buffer pools because Quick I/O bypasses the file system cache to improve database performance. Memory pages normally allocated to the file system cache can be allocated to the database buffer pools. Adjusting the size and number of buffer pools requires restarting the database. Cached Quick I/O can be used to dynamically modify memory allocation to the database without requiring a restart.

See "About Cached Quick I/O" on page 93.

Memory allocation

Never configure DB2 to use more than 75% of the physical memory available on the system. DB2 may have to compete with other processes for system memory resources, and all of these potential processes must be considered when sizing and allocating memory. In the ideal configuration, a system that is dedicated to DB2 simplifies the tuning and monitoring issues and ensures best performance.

TEMPORARY tablespaces

When more than one TEMPORARY tablespace exists in the database, they will be used in round-robin fashion in order to balance their usage. See the Administration Guide for information on using more than one tablespace, rebalancing and recommended values for EXTENTSIZE, PREFETCHSIZE, OVERHEAD, and TRANSFERRATE.

DMS containers

When you have more than one container in a DMS tablespace, it is important to ensure that all containers are the same physical, and logically declared, size. DB2 stripes data across available containers in a tablespace, writing in a round-robin fashion. If containers are not sized the same, then once the tablespace becomes sufficiently full, all I/O activity could be occurring to one physical file or device. This will incur a heavy performance penalty, especially when coupled with high values of the `NUM_IOCLEANERS`, `NUM_IOSERVERS` and `PREFETCHSIZE` configuration settings.

When extending tablespace containers using the `qiomkfile`command, ensure that you maintain this equal length across all containers in a tablespace.

Data, indexes, and logs

It is always important to separate database data and log files. The write patterns for these types of object are very different and so mixing them on the same device will adversely affect performance. Log writes are always sequential and high bandwidth, whereas writes to data tablespaces can range from random to large and sequential. It is important to ensure that log writes are fast and do not suffer from device latency in order to provide the highest performing database environment.

When using SMS tablespaces, it is not possible to separate data and indexes onto different devices. This means that there is no way to reduce contention for I/O and memory between these two types of database object. However, when using DMS devices, it is possible to place the data and indexes of tables into different tablespaces. This can provide much improved performance in environments which have very heavy usage of indexes and/or constrained memory.

In addition to being able to separate and therefore easily monitor I/O to the data and indexes, assigning indexes to a separate tablespace allows you to assign a dedicated bufferpool to the indexes or enable Cached Quick I/O on the index containers as required. This can greatly improve performance in environments where you want to ensure that indexes are always in memory and that there is no contention between data and indexes for a single bufferpools resources.

Database statistics

The DB2 database maintains internal information and statistics about the physical layout of data in the database. These internal statistics are used by the prefetch and I/O scheduling threads to plan operations in advance and can therefore have a very large impact on performance. With regular database activity, the statistics can become incorrect and therefore begin to have an adverse affect on I/O planning. This is especially true after major loads of new data, creating indexes on tables and heavy table activity involving large numbers of delete or update queries.

DB2 provides several tools to assist in updating these statistics and therefore enable continued and accurate I/O planning. These tools can be run from the db2 command prompt and are called RUNSTATS, REORG and REORGCHK tools. They should be run regularly to ensure optimal database performance.

See the System Catalog Statistics section in the *DB2 Administration Guide* and the section on CLP commands in the *DB2 Command Reference*.

About tuning Solaris for DB2

To achieve optimal performance using Veritas Storage Foundation for DB2, certain Solaris parameters need to be tuned. Changing these parameters requires modifying the Solaris kernel settings (specified in the `/etc/system` file) and rebooting the system.

You can add or change these tuning parameters in the `/etc/system` file using a text editor. The following example shows the contents of an `/etc/system` file:

```
* start DB2 *
set msgsys:msginfo_msgmax=65535
set msgsys:msginfo_msgmnb=65535
set msgsys:msginfo_msgseg=16384
set msgsys:msginfo_msgssz=16
set msgsys:msginfo_msgmap=258
set msgsys:msginfo_msgmni=256
set msgsys:msginfo_msgtql=512
*
set shmsys:shminfo_shmmax=134217728
set shmsys:shminfo_shmseg=16
set shmsys:shminfo_shmmni=300
*
set semsys:seminfo_semmni=256
set semsys:seminfo_semmap=258
set semsys:seminfo_semmns=512
set semsys:seminfo_semmnu=512
* end DB2 *
```

Note: The settings for all tunable parameters depend on such factors as the size of your system and database, the database load, and the number of users. Refer to the book for your version of DB2 to obtain precise settings for your hardware configuration.

maxuprc

This parameter sets the maximum number of processes that can be run concurrently by any one user. If you anticipate having a large number of users accessing the database concurrently, you may need to increase this parameter.

To increase the maxuprc parameter

1. Check the current setting for `maxuprc` as follows:

```
# echo "maxuprc/D" | adb -k
```

2. Modify or add the `maxuprc` setting in the `/etc/system` file as follows:

```
# set maxuprc=some_integer
```

msgmax

This parameter sets the maximum size (in bytes) of a single message. In general for Solaris systems, the value should not exceed 65535. See your DB2 documentation for the recommended value.

msgmnb

This parameter limits the total number of message bytes than can be on a single message queue at one time. The value should be some multiple of the `msgmax` setting. See your DB2 documentation for the recommended value.

msgseg

This parameter sets the number of msgssz segments available to all segments on all message queues. Memory for these segments is pre-allocated from kernel memory. See your DB2 documentation for the recommended value.

msgssz

This parameter sets the size of each message segment (in bytes) allocated for a message. The operating system allocates a total of msgseg * msgssz segments for use by application messages. The `msgssz` parameter is highly application dependent. See your DB2 documentation for the recommended value.

msgmap

This parameter sets the size of the message queue resource map. Each block of available, contiguous message segments requires one msgmap entry. See your DB2 documentation for the recommended value.

msgmni

This parameter sets the number of message queue identifiers available. Each message queue requires one message queue identifier and each identifier structure is approximately 144 bytes in size. See your DB2 documentation for the recommended value.

msgtql

This parameter sets the number of message queue headers available. Each queued message requires one message queue header and each header structure is approximately 12 bytes in size. See your DB2 documentation for the recommended value.

shmmax

This parameter sets the maximum size (in bytes) of a single shared memory segment. See your database documentation for the recommended value.

shmmni

This parameter sets the number of shared memory identifiers. See your database documentation for the recommended value.

shmseg

This parameter sets the maximum number of shared memory segments that can be attached by a process. See your database documentation for the recommended value.

semmap

This parameter sets the number of entries in semaphore map. The memory space given to the creation of semaphores is taken from `semmap`, which is initialized with a fixed number of map entries based on the value of `semmap`. The value of `semmap` should never be larger than `semmni`. See your database documentation for the recommended value.

semmni

This parameter sets the number of semaphore set identifiers in the system. The `semmni` parameter determines the number of semaphore sets that can be created

at any one time, and may need to be set higher for a large database. See your database documentation for the recommended value.

semmns

This parameter sets the maximum number of semaphores in the system. The `semmns` parameter may need to be set higher for a large database. See your database documentation for the recommended value.

semmnu

This parameter sets the system-wide maximum number of undo structures. Setting this parameter value equal to `semmni` provides for an undo structure for every semaphore set. Semaphore operations performed using `semop`(2) can be undone if the process terminates, but an undo structure is required to guarantee it. See your database documentation for the recommended value of `semmnu`.

Appendix A

Veritas Storage Foundation for DB2 command line interface

This appendix includes the following topics:

Overview of commands

Veritas Storage Foundation for DB2 provides a command line interface (CLI) to many key operations also supplied from within the GUI application. The command line interface lets you incorporate command operations into scripts and other administrative processes. These commands can be executed using the CLI or the Veritas Storage Foundation for DB2 GUI.

Note: The Veritas Storage Foundation for DB2 command line interface depends on certain tablespace, datafile, or container information (depending on your database) that is collected and stored in the SFDB repository. Some CLI commands update the repository by default. It is important to regularly ensure the repository is up-to-date by using the `db2ed_update` command.

All Veritas Storage Foundation for DB2 commands supported in the command line interface are located in the `/opt/VRTS/bin` directory. Online manual pages are located in the `/opt/VRTS/man` directory. Follow the installation instructions provided in the *Veritas Storage Foundation Installation Guide* to ensure you can use these commands and view the online manual pages.

Note: The commands ending with `_all` apply to multiple partitions of a partitioned DB2 database on the same host.

Table A-1 summarizes the commands available to you from the command line.

Table A-1 Veritas Storage Foundation for DB2 commands

Command	Description
`db2ed_update` and `db2ed_update_all`	Updates the repository in Veritas Storage Foundation for DB2. `db2ed_update_all` calls `db2ed_update` on every database partition for a partitioned DB2 database in a symmetric multiprocessing (SMP) environment.
`db2ed_checkconfig` and `db2ed_checkconfig_all`	Checks the configuration of a DB2 database in a Veritas Storage Foundation environment. `db2ed_checkconfig_all` calls `db2ed_checkconfig` on every database partition for a partitioned DB2 database in an SMP environment.
`db2ed_saveconfig` and `db2ed_saveconfig_all`	Saves the configuration of a DB2 database in a Veritas Storage Foundation environment. `db2ed_saveconfig_all` calls `db2ed_saveconfig` on every database partition for a partitioned DB2 database in an SMP environment.
`db2ed_ckptcreate` and `db2ed_ckptcreate_all`	Creates a Storage Checkpoint for a DB2 database. `db2ed_ckptcreate_all` creates a Version Checkpoint for a partitioned DB2 database. `db2ed_ckptcreate_all` calls `db2ed_ckptcreate` on every database partition.
`db2ed_ckptdisplay` and `db2ed_ckptdisplay_all`	Displays Storage Checkpoints for a DB2 database. `db2ed_ckptdisplay_all` displays Version Checkpoints for the current partitioned DB2 database.
`db2ed_ckptmount` and `db2ed_ckptmount_all`	Mounts a Storage Checkpoint for a DB2 database. `db2ed_ckptmount_all` mounts all Storage Checkpoints for a Version Checkpoint for a partitioned DB2 database.
`db2ed_ckptpolicy`	Creates and administers Storage Checkpoint allocation policies for a Multi-Volume File System (MVS). You can display, create, update, assign, and remove Storage Checkpoint allocation policies using this command.

Table A-1 Veritas Storage Foundation for DB2 commands *(continued)*

Command	Description
db2ed_ckptquota	Administers quotas for Storage Checkpoints.
db2ed_ckptremove and db2ed_ckptremove_all	Removes a Storage Checkpoint for a DB2 database. db2ed_ckptremove_all removes a Version Checkpoint for a partitioned DB2 database.
db2ed_ckptrollback and db2ed_ckptrollback_all	Rolls back a DB2 database to a Storage Checkpoint point-in-time image. db2ed_ckptrollback_all rolls back a a partitioned DB2 database to a Version Checkpoint. The db2ed_ckptrollback_all command calls db2ed_ckptrollback on every partition.
db2ed_ckptumount and db2ed_ckptumount_all	Unmounts a Storage Checkpoint for a DB2 database. db2ed_ckptumount_all unmounts a Version Checkpoint for a partitioned DB2 database.
db2ed_clonedb	Creates a copy of a DB2 database by cloning all existing database containers. This cloned database can only be started on the same host as the existing database.
qio_convertdbfiles **Note:** This command is not available on AIX and Linux.	Converts DB2 container files to Quick I/O files.
qio_getdbfiles	Extracts information on files used by the database and stores the names of these files in mkqio.dat. The mkqio.dat file is used by the qio_convertdbfiles command.
qio_recreate	Automatically recreates Quick I/O files when the database is recovered. The command expects to find a mkqio.dat file in the directory where the qio_recreate command is run.
db2ed_vmchecksnap	Creates and validates a snapplan that the db2ed_vmsnap command uses to create a volume snapshot of a DB2 database. The snapplan specifies snapshot scenarios (such as online_snapshot, online_mirror, or offline). The command can also be used to validate, list, copy, and remove a snapplan.

Table A-1 Veritas Storage Foundation for DB2 commands *(continued)*

Command	Description
`db2ed_vmsnap`	Creates a snapshot image of a DB2 database by splitting the mirror volumes used by the database. You can also use this command to resynchronize the snapshot image back to the current database. The command also allows you to reverse resynchronize a snapshot image of an DB2 database.
`db2ed_vmclonedb`	Mounts the file systems on the snapshot volumes and starts a clone database from snapshot volumes. You can also use this command to shut down or restart the clone database, unmount the file systems, or deport the clone database's volumes.
`edgetmsg2`	Manages message log files. You can use this utility to display and list message log files. You can also use this utility to write a message to a log file or to the console, or read the log file and print to the console.
`vxstorage_stats`	Displays storage object I/O statistics.

Command support

All commands ending with `_all` are supported in an SMP environment. These commands apply to multiple partitions.

The following commands are not supported in an SMP environment:

- `db2ed_clonedb`
- `db2ed_ckptpolicy`
- `db2ed_ckptquota`
- `db2ed_vmchecksnap`
- `db2ed_vmsnap`
- `db2ed_vmclonedb`

The following commands applies at the host level:

- `edgetmsg2`
- `vxstorage_stats`

About the command line interface

You can use the Veritas Storage Foundation for DB2 command line interface to perform administrative operations. For more detailed information about the commands and their syntax and available options, see the individual manual pages.

Updating the repository using db2ed_update

You can use the command to update the repository.

Any time you change the structure of the database (for example, by adding or deleting containers), you must run `db2ed_update`.

Before updating the repository, review the following information:

Prerequisites

- You must be logged on as the instance owner (typically, the user ID `db2inst1`).

Usage notes

- The `db2ed_update` command saves or updates the information related to the DB2 database in the SFDB repository.
- The database must be up and running, and the `$DB2DATABASE` variable argument must be specified with the `-D` option. The `$DB2INSTANCE` argument is optional.
- For multiple partitions, use the `db2ed_update_all` command. `db2ed_update_all` calls `db2ed_update` on every database partition for a partitioned DB2 database.
- See the `db2ed_update`(1M) manual page for more information.

Table A-2 lists the options for updating the repository.

Table A-2 Options for updating the repository

Option	Description
`-D DB2DATABASE`	Specifies the name of the DB2 database for which a snapshot image will be created.
`-I DB2INSTANCE`	Specifies the name of the DB2 instance.
`-G service_group`	Specifies the name of the VCS service group for the database if the DB2 instance is under VCS control.

To update the SFDB repository

- Use the command as follows:

```
$ /opt/VRTS/bin/db2ed_update -D PROD
```

Checking the database configuration environment using db2ed_checkconfig

You can use the `db2ed_checkconfig` command to verify and report on the database environment from the command line.

Before checking the configuration environment, the following conditions must be met:

Prerequisites
- You must be logged on as the instance owner (typically, the user ID `db2inst1`).

Usage notes
- The `db2ed_checkconfig` command is used to verify various elements of the database environment. The utility attempts to use certain basic rules on database settings, file system and volume parameters and layout options to verify how resilient and well configured a configuration is. The information provided is valid for the supplied database.
- For multiple partitions, use the `db2ed_checkconfig_all` command. `db2ed_checkconfig_all` checks the configuration for a partitioned DB2 database by calling `db2ed_checkconfig` on every database partition.
- See the `db2ed_checkconfig_all`(1M) manual page.
- See the `db2ed_checkconfig`(1M) manual page.

To check the database configuration environment

- Use the `db2ed_checkconfig` command as follows:

```
$ /opt/VRTS/bin/db2ed_checkconfig -D PROD


Examining File System and DB2 Container attributes.

Total of 13 containers over 2 file systems.

All file systems are VxFS.

SFDB2 db2ed_checkconfig WARNING V-81-3508: The VxFS layout of
/udb_home/db2inst1/prod_temp is not version 4.
```

```
SFDB2 db2ed_checkconfig WARNING V-81-3508: The VxFS layout of
/udb_home is not version 4.

VxFS file systems not in the version 4 layout do
not support all of the features of VxFS.

The database has:
  5 SMS Containers
  3 DMS File Containers
  0 DMS Device Containers

Examining DB2 container fragmentation.

0 SMS containers skipped fragmentation check.

11 files are fragmented.

Examining File System tunable settings.

Parameters for all VxFS file systems used by PROD.

Filesystem i/o parameters for /udb_home/db2inst1/prod_temp
read_pref_io = 65536
read_nstream = 1
read_unit_io = 65536
write_pref_io = 65536
write_nstream = 1
write_unit_io = 65536
pref_strength = 10
buf_breakup_size = 1048576
discovered_direct_iosz = 262144
max_direct_iosz = 1048576
default_indir_size = 8192
qio_cache_enable = 0
write_throttle = 0
max_diskq = 1048576
initial_extent_size = 8
max_seqio_extent_size = 2048
max_buf_data_size = 8192
hsm_write_prealloc = 0
read_ahead = 1
inode_aging_size = 0
inode_aging_count = 0
fcl_maxalloc = 31775
fcl_keeptime = 0
fcl_winterval = 3600
```

```
Filesystem i/o parameters for /udb_home
read_pref_io = 65536
read_nstream = 1
read_unit_io = 65536
write_pref_io = 65536
write_nstream = 1
write_unit_io = 65536
pref_strength = 10
buf_breakup_size = 1048576
discovered_direct_iosz = 262144
max_direct_iosz = 1048576
default_indir_size = 8192
qio_cache_enable = 0
write_throttle = 0
max_diskq = 1048576
max_diskq = 1048576
initial_extent_size = 8
max_seqio_extent_size = 2048
max_buf_data_size = 8192
hsm_write_prealloc = 0
read_ahead = 1
inode_aging_size = 0
inode_aging_count = 0
fcl_maxalloc = 222425
fcl_keeptime = 0
fcl_winterval = 3600
Examining DB2 Volume layout and attributes.

Data for database PROD is contained in one volume group.

SFDB2 db2ed_checkconfig WARNING V-81-3419: File system
/udb_home/db2inst1/prod_temp is not
mirrored using VxVM.

SFDB2 db2ed_checkconfig WARNING V-81-3419: File system
/udb_home is not mirrored using VxVM.

Examining Volume Group versions.

Volume Group version for db2dg is:
110
See the vxdg(1M) man page for more information on Volume Group
versions.
Examining DB2 internal information.
```

```
DB2 Version is 8.1.
Examining DB2 logging mode.
The database has transaction logs in directory .
/udb_home/db2inst1/prod_re.
SFDB2 db2ed_checkconfig WARNING V-81-3322: Transaction log directory
is not mirrored using VxVM.
The database is running in archivelog mode.
SFDB2 db2ed_checkconfig WARNING V-81-3322: The database has a
user-exit defined for archiving logs. Cannot determine where logs are
archived to.
Examining DB2 Database Free Space.

DB20000I  The SQL command completed successfully.
Name                                 = SYSCATSPACE
Type                                 = System managed space
Total pages                          = 1745
Used pages                           = 1745
Free pages                           = Not applicable
Page size (bytes)                    = 4096
Name                                 = USER1
Type                                 = Database managed space
Total pages                          = 12800
Used pages                           = 160
Free pages                           = 12608
Page size (bytes)                    = 4096
Name                                 = INDX1
Type                                 = Database managed space
Total pages                          = 12800
Used pages                           = 96
Free pages                           = 12672
Page size (bytes)                    = 4096
Name                                 = TEMPORARY1
Type                                 = System managed space
Total pages                          = 1
Used pages                           = 1
Free pages                           = Not applicable
Page size (bytes)                    = 4096
Examining File System and DB2 Container attributes.
```

Saving the database configuration environment using db2ed_saveconfig

You can use the command to save configuration information on the database, Symantec products, and system hardware from the command line.

Before saving the configuration environment, the following conditions must be met:

Prerequisites
- You must be logged on as the instance owner (typically, the user ID `db2inst1`).

Usage notes
- The `db2ed_saveconfig` command is used to collect and record configuration information on the database, Symantec products, and system hardware. Information is gathered in the context of a specified database. The utility attempts to gather enough information to allow an administrator to reconstruct a system and database from scratch, in the case of a complete system failure.
- Information collected is in the form of many system configuration files and the results of querying the system hardware, Symantec products, and the database. The location where configuration information has been saved is displayed as output from the `db2ed_saveconfig` command. Alternatively, you can use the `-l` option to designate this location.
- For multiple partitions, use the `db2ed_saveconfig_all` command. `db2ed_saveconfig_all` saves the configuration for a partitioned DB2 database by calling `db2ed_saveconfig` on every database partition.
- See the `db2ed_saveconfig_all`(1M) manual page for more information.
- See the `db2ed_saveconfig`(1M) manual page.

To save the database configuration environment

- Use the command as follows:

```
$ /opt/VRTS/bin/db2ed_saveconfig -D PROD

System configuration information saved to directory:

/tmp/vxdbed.DR.1148
```

Creating Storage Checkpoints using db2ed_ckptcreate

You can use the command to create a Storage Checkpoint from the command line.

Storage Checkpoints can be either online or offline. If `online` is specified, the database is put into write suspend mode when the Storage Checkpoint is created. If `offline` is specified, the database is expected to be down.

Before creating a Storage Checkpoint, the following conditions must be met:

Prerequisites

- You must be logged on as the instance owner (typically, the user ID `db2inst1`).
- For best recoverability, always keep the `LOGRETAIN` and/or `USEREXIT` database configuration parameters enabled when you create Storage Checkpoints.

Usage notes

- db2ed_ckptcreate stores Storage Checkpoint information under the following directory:
  /etc/vx/vxdbed/DB2,$DB2INSTANCE,$DB2DATABASE/NODEnum/*checkpoint_dir*
- For multiple partitions, use the `db2ed_ckptcreate_all` command. `db2ed_ckptcreate_all` creates a Version Checkpoint for a partitioned DB2 database by calling `db2ed_ckptcreate` on every database partition.
- See the db2ed_ckptcreate(1M) manual page for more information.

To create Storage Checkpoints while the database is online

- Use the `db2ed_ckptcreate` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptcreate -D PROD -o online

An online Storage Checkpoint Checkpoint_971672042
is created at GMT 2004-05-01-18.22.34.0000.
```

To create Storage Checkpoints without updating the repository while the database is online

- Use the `db2ed_ckptcreate` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptcreate -D PROD -o online -n

An online Storage Checkpoint Checkpoint_971672043
is created at GMT 2004-05-01-18.22.34.0000.
```

To create Storage Checkpoints while the database is offline

1 Terminate the database before creating the offline Storage Checkpoint:

```
# db2 terminate
```

2 Use the `db2ed_ckptcreate` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptcreate -D PROD -o offline

An offline Storage Checkpoint Checkpoint_971672044
is created at GMT 2004-05-01-18.22.34.0000.
```

The default option is `online`.

To assign a Storage Checkpoint allocation policy to a Storage checkpoint

- Use the `db2ed_ckptcreate` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptcreate -D PROD -o online -p \
ckpt_data,ckpt_metadata

Creating online Storage Checkpoint of database PROD.

Storage Checkpoint Checkpoint_971672044 created.
```

To create Storage Checkpoints on multiple partitions

- As the DB2 instance owner, use the `db2ed_ckptcreate_all` command as follows:

  ```
  $ /opt/VRTS/bin/db2ed_ckptcreate_all -I db2inst -D PROD
  ```

  where `db2inst` is the instance name.

  ```
  Creating Version Checkpoint Version_ckpt_1088639094 on all partitions.
  rah: omitting logical node 0

  Creating checkpoint on partition 1.
  An online Storage Checkpoint Checkpoint_1088639125 is created
  at GMT 2004-06-30-23.45.29.0000
  db2ed_ckptcreate -I ... completed ok

  Creating checkpoint on partition 0.
  An online Storage Checkpoint Checkpoint_1088639159 is created
  at GMT 2004-06-30-23.46.03.0000
  db2ed_ckptcreate -I ... completed ok
  ```

  The command output will contain "`rah: omitting logical node 0`". This is the normal behavior.

Scheduling Storage Checkpoints using db2ed_ckptcreate and cron

You can use the `db2ed_ckptcreate` command to schedule Storage Checkpoint creation in a `cron` job or other administrative script.

Before scheduling Storage Checkpoints, the following conditions must be met:

Prerequisites

- You must be logged on as the instance owner (typically, the user ID `db2inst1`).

Usage notes

- Create a new `crontab` file or edit an existing `crontab` file to include a Storage Checkpoint creation entry with the following space-delimited fields:
  minute hour day_of_month month_of_year day_of_week\
  /opt/VRTS/bin/db2ed_ckptcreate
  where:
  minute - numeric values from 0-59 or *
  hour - numeric values from 0-23 or *
  day_of_month - numeric values from 1-31 or *
  month_of_year - numeric values from 1-12 or *
  day_of_week- numeric values from 0-6, with 0=Sunday or *
  Each of these variables can either be an asterisk (meaning all legal values) or a list of elements separated by commas. An element is either a number or two numbers separated by a hyphen (meaning an inclusive range).
- See the `db2ed_ckptcreate`(1M), `cron`(1M), and `crontab`(1) manual pages for more information.

Scheduling Storage Checkpoint creation in a cron job

- To create a Storage Checkpoint twice a day, at 5:00 a.m. and 7:00 p.m., every Monday through Friday, include the following entry in your `crontab` file:

```
0 5,19 * * 1-5 /opt/VRTS/bin/db2ed_ckptcreate -D PROD \
-I db2inst1 -o online
```

- To create a Storage Checkpoint at 11:30 p.m., on the 1st and 15th day of each month, include the following entry in your `crontab` file:

```
30 23 1,15 * * /opt/VRTS/bin/db2ed_ckptcreate -D PROD \
-I db2inst1 -o online
```

- To create a Storage Checkpoint at 1:00 a.m. every Sunday while the database is offline, include the following entry in your `crontab` file:

```
0 1 * * 0 /opt/VRTS/bin/db2ed_ckptcreate -D PROD \
-I db2inst1 -o offline
```

Displaying Storage Checkpoints using db2ed_ckptdisplay

You can use the `db2ed_ckptdisplay`command to display the Storage Checkpoints associated with a DB2 instance from the command line.

You can also use it to display fileset quota values.

Before displaying Storage Checkpoints, the following conditions must be met:

Prerequisites

- You may be logged in as either the instance owner or `root`. If you execute the command as `root`, you must set up `$DB2INSTANCE` as the instance owner.

Usage Notes

- In addition to displaying the Storage Checkpoints created by Veritas Storage Foundation for DB2, `db2ed_ckptdisplay` also displays other Storage Checkpoints (for example, Storage Checkpoints created by the NetBackup).
- The **Status** field identifies if the Storage Checkpoint is partial (P), complete (C), invalid (I), mounted (`M`), read-only (`R`), or writable (W).
- For multiple partitions, use the `db2ed_ckptdisplay_all` command. `db2ed_ckptdisplay_all` displays Version Checkpoints associated with the current partitioned DB2 database.
- See the `db2ed_ckptdisplay`(1M) manual page for more information.

To display Storage Checkpoints created by Veritas Storage foundation for DB2

- Use the `db2ed_ckptdisplay` command as follows to display information for Storage Checkpoints created by Veritas Storage Foundation for DB2:

```
$ /opt/VRTS/bin/db2ed_ckptdisplay -D PROD

Checkpoint_975876659        Sun Apr 3 12:50:59 2005       P+R

Checkpoint_974424522_wr001  Thu May 16 17:28:42 2005      C+R

Checkpoint_974424522        Thu May 16 17:28:42 2005      P+R
```

To display other Storage Checkpoints

- Use the `db2ed_ckptdisplay` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptdisplay -D PROD -o other

NetBackup_incr_PROD_955133480           NBU   /db01
NetBackup_full_PROD_9551329         52  NBU   /db01
```

To display other Storage Checkpoints without updating the repository

- Use the `db2ed_ckptdisplay` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptdisplay -D PROD -o other

NetBackup_incr_PROD_955133480           NBU   /db01
NetBackup_full_PROD_9551329         52  NBU   /db01
```

To display all Storage Checkpoints

- Use the `db2ed_ckptdisplay` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptdisplay -D PROD -o all

Checkpoint_971672042       Sun May 15 13:55:53 2005        C+R

Checkpoint_903937870       Fri May 13 22:51:10 2005        C+R

Checkpoint_901426272       Wed May 11 16:17:52 2005        P+R

NetBackup_incr_PROD_955133480           NBU   /db01

NetBackup_full_PROD_9551329         52  NBU   /db01
```

To display all Storage Checkpoints without updating the repository

- Use the `db2ed_ckptdisplay` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptdisplay -D PROD -o all -n

Checkpoint_971672042       Sun May 15 13:55:53 2005        C+R

Checkpoint_903937870       Fri May 13 22:51:10 2005        C+R

Checkpoint_901426272       Wed May 11 16:17:52 2005        P+R

NetBackup_incr_PROD_955133480           NBU   /db01
NetBackup_full_PROD_9551329         52  NBU   /db01
```

To display fileset quota values

- Use the command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptdisplay -D PROD -c \
Checkpoint_903937870 -Q

Checkpoint_975876659        Wed Mar 19 9:12:20 2005        C+R

Filesystem                   HardLim   SoftLim   CurrentUse

/udb_home/db2inst1/indx1_1   100000    50000     2028

/udb_home/db2inst1/indx1_1   100000    50000     2028

/udb_home/db2inst1/temp      150000    80000     2142

/udb_home/db2inst1/system1   150000    70000     3092
```

Mounting Storage Checkpoints using db2ed_ckptmount

You can use the command to mount a Storage Checkpoint for the database from the command line.

Before mounting Storage Checkpoints, review the following information:

Prerequisites

- You may be logged in as either the instance owner or `root`. If you execute the command as `root`, you must set up `$DB2INSTANCE` as the instance owner.

Usage notes

- The `db2ed_ckptmount` command is used to mount a Storage Checkpoint into the file system namespace. Mounted Storage Checkpoints appear as any other file system on the machine and can be accessed using all normal file system based commands.
- The DB2 instance owner must have permission to mount the Storage Checkpoint on the mount point.
- Storage Checkpoints can be mounted as read-only or read-write. By default, Storage Checkpoints are mounted as read-only.
- If the `rw` (read-write) option is used, `_wrxxx`, where `xxx` is an integer, will be appended to the Storage Checkpoint name.
- If the specified mount point directory does not exist, then `db2ed_ckptmount` creates it before mounting the Storage Checkpoint, as long as the DB2 instance owner has permission to create it.
- For multiple partitions, use the `db2ed_ckptmount_all` command. `db2ed_ckptmount_all` mounts all Storage Checkpoints for a Version Checkpoint for a partitioned DB2 database.
- See the `db2ed_ckptmount_all`(1M) manual page.
- See the `db2ed_ckptmount`(1M) manual page for more information.

To mount Storage Checkpoints with the read/write option

- Use the `db2ed_ckptmount` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptmount -D PROD -c Checkpoint_971672042\
-m /tmp/ckpt_rw -o rw

Creating Storage Checkpoint on /tmp/ckpt_rw/udb_home/udb01a_home
with name Checkpoint_971672042_wr001
```

To mount Storage Checkpoints with the read-only option

- Use the command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptmount -D PROD -c Checkpoint_971672042\
-m /tmp/ckpt_ro -o ro
```

Unmounting Storage Checkpoints using db2ed_ckptumount

You can use the command to unmount a Storage Checkpoint from the command line.

Before unmounting Storage Checkpoints, the following conditions must be met:

Prerequisites
- You may be logged in as either the instance owner or `root`. If you execute the command as `root`, you must set up `$DB2INSTANCE` as the instance owner.

Usage notes
- The `db2ed_ckptumount` command is used to unmount a mounted Storage Checkpoint from the file system namespace. Mounted Storage Checkpoints appear as any other file system on the machine and can be accessed using all normal file system based commands. When mounted Storage Checkpoints are not required, they can be unmounted.
- See the `db2ed_ckptumount`(1M) manual page for more information.

Note: For multiple partitions, use the `db2ed_ckptumount_all` command.

To unmount Storage Checkpoints

- Use the command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptumount -D PROD \
-c Checkpoint_971672042_wr001
```

Creating and working with Storage Checkpoint allocation policies using db2ed_ckptpolicy

You can use the command to create and administer Storage Checkpoint allocation policies for Multi-Volume File Systems (MVSs). Storage Checkpoint allocation policies specify a list of volumes and the order in which to allocate data to them.

Before creating or working with Storage Checkpoint allocation policies, the following conditions must be met:

Prerequisites

- You must be logged on as the instance owner (typically, the user ID `db2inst1`).

Usage notes

- The `db2ed_ckptpolicy` command can be used only on file systems using disk layout Version 6.
- The VxVM volume set and VxFS Multi-Volume File System features must be enabled to use Storage Checkpoint allocation policies.
- If you want to set a Storage Checkpoint allocation policy for a particular file system in the database, the VxFS Multi-Volume File System feature must be enabled on that file system.
- The status of a Storage Checkpoint allocation policy is either `partial` or `complete`. A partial policy is one that does not exist on all file systems used by the database. A complete policy is one that exists on all file systems.
- After an allocation policy is assigned to a Storage Checkpoint, the allocation mechanism attempts to satisfy requests from each device in the order specified in the allocation policy. If the request cannot be satisfied from any of the devices in the allocation policy, the request will fail, even if other devices that have space exist in the file system. Only devices listed in the policy can be allocated.
- See the `db2ed_ckptpolicy`(1M) manual page for more information.

To create a Storage Checkpoint allocation policy

- Use the `db2ed_ckptpolicy` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptpolicy -D DB2DATABASE \
-o create -p ckpt_policy
```

Output similar to the following is displayed:

```
File System: /mvsfs/v2 (MVS volumes: mvsv4,mvsv5)
Assigned Data Policy: NONE (MVS Volumes: N/A)
Assigned Meta Data Policy: NONE (MVS Volumes: N/A)
Please enter the volume name(s), sperated by space,
for the policy ckpt_policy [skip,quit]: mvsv4

File System: /mvsfs/v1 (MVS volumes: mvsv1,mvsv2,mvsv3)
Assigned Data Policy: NONE (MVS Volumes: N/A)
Assigned Meta Data Policy: NONE (MVS Volumes: N/A)
Please enter the volume name(s), separated by space,
for the policy ckpt_policy [skip,quit]: mvsv2

The following information will be used to create policy ckpt_sample
```

```
ckpt_sample             /mvsfs/v2             mvsv4
ckpt_sample             /mvsfs/v1             mvsv2
```

This example assumes the following:

- Two MVS file systems `/mvsfs/v1` and `/mvsfs/v2` are used for datafiles.
- File system `/mvsfs/v1` is created on volume set mvsvset1.
- File system `/mvsfs/v2` is created on volume set mvsvset2.
- Volume set `mvsvset1` contains volumes `mvsv1`, `mvsv2`, and `mvsv3`.
- Volume set `mvsvset2` contains volumes `mvsv4` and `mvsv5`.

To display Storage Checkpoint allocation policy within the database

- Use the `db2ed_ckptpolicy` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptpolicy -D DB2DATABASE \
-n -o display [-c storage_ckpt | -p ckpt_policy]
```

If `-p ckpt_policy` and `-c storage_ckpt` options are not specified, output similar to the following is displayed:

```
Policy Name           File System Coverage
--------------------- --------------------
 ckpt                 Complete
 ckpt_data            Complete
 ckpt_metadata        Complete
 new_ckpt             Partial
 ckpt_sample          Complete
```

If `-p ckpt_policy` option is specified, output similar to the following is displayed:

```
Policy Name         File System    MVS volumes
----------------    -----------    -----------
ckpt_sample         /mvsfs/v2      mvsv4
ckpt_sample         /mvsfs/v1      mvsv2
```

If the `-c storage_ckpt` option is specified, output similar to the following is displayed:

```
Storage Checkpoint      File System   Data Policy Meta Data Policy
------------------      ------------  ----------- ----------------
Checkpoint_1095125037   /mvsfs/v2     ckpt_data   ckpt_metadata
Checkpoint_1095125037   /mvsfs/v1     ckpt_data   ckpt_metadata
```

To update a Storage Checkpoint allocation policy

- Use the command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptpolicy -D DB2DATABASE \
-n -o update -p ckpt_policy
```

Output similar to the following is displayed:

```
File System: /mvsfs/v2 (MVS volumes: mvsv4,mvsv5)
Policy: ckpt_sample (MVS volumes: mvsv4)
Please enter the volume name(s), separated by space,
for the policy ckpt_sample [skip,quit]: mvsv5

File System: /mvsfs/v1 (MVS volumes: mvsv1,mvsv2,mvsv3)
Policy: ckpt_sample (MVS volumes: mvsv2)
Please enter the volume name(s), separated by space,
for the policy ckpt_sample [skip,quit]: mvsv2,mvsv3

The following information will be used to create policy ckpt_sample
ckpt_sample             /mvsfs/v2               mvsv5
ckpt_sample             /mvsfs/v1               mvsv2,mvsv3
```

To assign a Storage Checkpoint allocation policy

- Use the `db2ed_ckptpolicy` command as follows to assign an allocation policy to a specified Storage Checkpoint:

```
$ /opt/VRTS/bin/db2ed_ckptpolicy -D DB2DATABASE \
-n -o assign -c ckpt_name -p ckpt_policy[,ckpt_metadata_policy]
```

To remove a Storage Checkpoint allocation policy

- Use the command as follows to remove an allocation policy from a specified Storage Checkpoint:

```
$ /opt/VRTS/bin/db2ed_ckptpolicy -D DB2DATABASE \
-n -o remove -p ckpt_policy
```

Administering Storage Checkpoint quotas using db2ed_ckptquota

You can use the `db2ed_ckptquota` command to administer file system quotas for Storage Checkpoint for a database from the command line.

Before administering Storage Checkpoint quotas, the following conditions must be met:

Prerequisites
- You must be logged on as the instance owner (typically, the user ID `db2inst1`).
- The repository entry for the database must exist and the DBA must be the owner of all file systems to be affected.

Usage notes
- See the `db2ed_ckptquota`(1M) manual page for more information.

To set quota limits for all file systems in the database and enable quota enforcement

- Use the `db2ed_ckptquota` command as follows to set the hard and soft limits for all file systems in the database and enable quota enforcement:

```
$ /opt/VRTS/bin/db2ed_ckptquota -D DB2DATABASE \
-o set=hardlimit,softlimit,enable
```

To set quota limits for all file systems specified in a list file

- Use the `db2ed_ckptquota` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptquota -D DB2DATABASE \
-o set=hardlimit,softlimit -f listfile
```

To disable quota limits for a file system

- Use the command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptquota -D DB2DATABASE -o disable \
filesystem_path
```

To display quota values for all file systems in the database

- Use the `db2ed_ckptquota` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptquota -D DB2DATABASE -o display

Filesystem                Hardlimit     Softlimit     CurrentUse

/db2/udb_home             50000         40000         136

/db2/testvol01            25000         20000         128

/db2/testvol02            50000         40000         128

/db2/testvol03            50000         40000         0

/db2/testvol04            25000         20000         128

/db2/testvol05            50000         40000         128
```

The numbers in the “Hardlimit” and “Softlimit” columns represent the total numbers of file system blocks allowed.

`CurrentUse` displays the number of filesystem blocks currently used by all Storage Checkpoints in the filesystem. If there are no Storage Checkpoints, or if quotas have been disabled, `CurrentUse` will display 0.

Performing Storage Rollback using db2ed_ckptrollback

You can use the `db2ed_ckptrollback` command to rollback a DB2 instance to a Storage Checkpoint.

Before performing a Storage Rollback, the following conditions must be met:

Prerequisites

- You must be logged in as the instance owner.

Usage notes

- The db2ed_ckptrollback rolls a DB2 database back to a specified Storage Checkpoint. You can perform a Storage Rollback for the entire database.
  Database rollback for the entire database requires that the database be inactive before Storage Rollback commences. The db2ed_ckptrollback command will not commence if the DB2 database is active.
- For multiple partitions, use the db2ed_ckptrollback_all command. db2ed_ckptrollback_all rolls back a partitioned DB2 database to a Version Checkpoint. The db2ed_ckptrollback_all command calls db2ed_ckptrollback on every partition.
- See the db2ed_ckptrollback(1M) manual page for more information.

To roll back a DB2 instance to a Storage Checkpoint

- Use the db2ed_ckptrollbackcommand as follows:

```
$ /opt/VRTS/bin/db2ed_ckptrollback -D PROD -c \
Checkpoint_903937870
```

Removing Storage Checkpoints using db2ed_ckptremove

You can use the db2ed_ckptremovecommand to remove a Storage Checkpoint for a DB2 instance at the command line.

Before removing Storage Checkpoints, the following conditions must be met:

Prerequisites

- You may be logged in as either the instance owner or root. If you execute the command as root, you must set up $DB2INSTANCE as the instance owner.

Usage notes

- The db2ed_ckptremove command is used to remove a Storage Checkpoint from the file system, or file systems, it is associated with. The Storage Checkpoint must have been created using the GUI or the db2ed_ckptcreate(1M) command.
- You must unmount the Storage Checkpoint before you can remove it.
- For multiple partitions, use the db2ed_ckptremove_all command. db2ed_ckptremove_all removes a Version Checkpoint for a partitioned DB2 database.
- See the db2ed_ckptremove(1M) manual page for more information.

To remove Storage Checkpoints

- Use the `db2ed_ckptremove` command as follows:

```
$ /opt/VRTS/bin/db2ed_ckptremove -D PROD \
-c Checkpoint_971672042_wr001
```

Cloning the DB2 database using db2ed_clonedb

You can use the `db2ed_clonedb` command to clone a DB2 database using a Storage Checkpoint. Cloning an existing database using a Storage Checkpoint must be done on the same host.

You have the option to manually or automatically recover the database when using the command:

- Manual (Interactive) recovery, which requires using the `-i` option, of the clone instance allows the user to control the degree of recovery by specifying which archive log files are to be replayed.
- Automatic (non-interactive) recovery, which is the default usage of the command, recovers the entire database and replays all of the archive logs. You will not be prompted for any archive log names.

Before cloning the DB2 database, review the following information:

Prerequisites

- You must first create a Storage Checkpoint.
  See "Creating Storage Checkpoints using db2ed_ckptcreate" on page 282.
- Before using `db2ed_clonedb` to clone a database either within the same instance or across instances, ensure that your locale is the same as the database locale and that the locale is the same across instances. If you do not know what locale to set, refer to the file `/opt/VRTSdb2ed/lib/DB2CODPAGE.tbl` for a mapping between the locale and the database Code Page.
- Make sure you have enough space and system resources to create a clone instance on your system.
- A clone database takes up as much memory and machine resources as the primary database.

Usage notes

- The `db2ed_clonedb` command is used to create a copy of a database, cloning all existing database files to new locations.
- The `db2ed_clonedb` command only works when the instance is up.
- It is assumed that the user has a basic understanding of the database recovery process.
- When cloning a database using `db2ed_clonedb`, any database within the target instance that has the same name as the source database to be cloned will be temporarily uncataloged and therefore unavailable. If the source database is being cloned within the same instance, it will be temporarily uncataloged while `db2ed_clonedb` is running. The database will be recataloged on completion of `db2ed_clonedb`.
- See the `db2ed_clonedb`(1M) manual page for more information.

Table A-3 lists the options for cloning the DB2 database.

Table A-3 db2ed_clonedb command options

Option	Description
`-I SOURCE_INSTANCE`	Specifies the source DB2 database. If the instance is not specified here, it is set as the current user.
`-S SOURCE_DATABASE`	Specifies the name of the source DB2 database.
`-T TARGET_DATABASE`	Specifies the name of the new DB2 database that will be created.
`-c CKPT_NAME`	Indicates the name of the Storage Checkpoint to use for creating the new database. The Storage Checkpoint is mounted automatically during the cloning process.
`-m MOUNT_POINT`	Indicates the location of the database containers to be mounted.
`-l`	Requires the argument `TARGET_DATABASE_REDOLOG_DIRECTORY`. Specifies the redo log directory of the target database.
`-i`	Runs the command in interactive mode where you must respond to prompts by the system. The default mode is non-interactive. (Optional)
`-a`	Requires the argument `RECOVERY_LOG_LOCATION`. If this option is specified, a minimal database recovery will occur automatically after the clone is created. (Optional)

Table A-3 db2ed_clonedb command options *(continued)*

Option	Description
`-o umount`	Shuts down the clone database and unmounts the Storage Checkpoint file system.
`-o restartdb`	Mounts the Storage Checkpoint file system and starts the clone database. The `-o restartdb` option will not attempt to recover the clone database.
`-d`	Used with the `-o umount` option. If the `-d` option is specified, the Storage Checkpoint used to create the clone database will be removed along with the clone database.

To clone a DB2 database with manual DB2 recovery

- Use the `db2ed_clonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -S source_db -T target_db1 \
-c Checkpoint_1049927758 -m /db2clone/target_db1

Clone database succeeded.
```

To clone a DB2 database with automatic DB2 recovery

- Use the `db2ed_clonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -S source_db -T target_db2 \
-c Checkpoint_1049927758 -m /db2clone/target_db2 -a \
/db2clone/db2inst1/NODE0000/SQL00002/SQLOGDIR/


Clone database succeeded.
The database will be rolled forward to 2005-04-09-22.36.02.0000.
/db2clone/db2inst1/NODE0000/SQL00002/SQLOGDIR/
will be used to be searched for archived logs during recovery.
Rollforward Status
Input database alias                   = TARGET_DB2
Number of nodes have returned status   = 1
Node number                            = 0
Rollforward status                     = DB  working
Next log file to be read               = S0000002.LOG
Log files processed                    =  -
Last committed transaction             = 2005-04-08-20.32.53.000000

DB20000I  The ROLLFORWARD command completed successfully.
Rollforward Status
```

```
Input database alias                    = TARGET_DB2
Number of nodes have returned status    = 1
Node number                             = 0
Rollforward status                      = not pending
Next log file to be read                =
Log files processed                     = S0000002.LOG - S0000002.LOG
Last committed transaction              = 2005-04-08-20.32.53.000000
DB20000I  The ROLLFORWARD command completed successfully.
```

To clone a DB2 database with interactive DB2 recovery

- Use the `db2ed_clonedb`command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -S source_db -T target_db3 \
-c Checkpoint_1049927758 -m /db2clone/target_db3 -i
/db2clone/db2inst1/NODE0000/SQL00002/SQLOGDIR/ will be used to
be searched for archived logs during recovery.


Press <Return> to continue
Or <r> to retry
Or <q> to quit:

The database will be rolled forward to 2005-04-09-22.36.02.0000.
The archived logs will be retrieved from
/db2clone/db2inst1/NODE0000/SQL00002/SQLOGDIR/.
The estimated minimum logs are S0000002.LOG-S0000002.LOG.

You can retrieve all log files by executing
cp /db2clone/db2inst1/NODE0000/SQL00002/SQLOGDIR/*.LOG
/db2clone/target_db3/REDOLOG
from another session and then
Press q to continue

Or Press Enter to retrieve the minimum logs.

The recovery process will stop if more logs is required.

Press <Return> to continue ...
   or <q> to skip ...

Rollforward Status

Input database alias                 = TARGET_DB3
Number of nodes have returned status = 1
Node number                          = 0
Rollforward status                   = DB  working
```

```
Next log file to be read             = S0000002.LOG
Log files processed                  =  -
Last committed transaction           = 2003-04-08-20.32.53.000000

DB20000I  The ROLLFORWARD command completed successfully.

Rollforward Status

Input database alias                 = TARGET_DB3
Number of nodes have returned status = 1
Node number                          = 0
Rollforward status                   = not pending
Next log file to be read             =
Log files processed                  = S0000002.LOG - S0000002.LOG
Last committed transaction           = 2003-04-08-20.32.53.000000

DB20000I  The ROLLFORWARD command completed successfully.
```

To shut down the clone database and unmount the Storage Checkpoint

- Use the `db2ed_clonedb`command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -T target_db3 -o umount
```

To mount a Storage Checkpoint file system and start the clone database

- Use the `db2ed_clonedb`command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -T target_db3 -o restartdb
```

To delete a clone database and the Storage Checkpoint used to create it

- Use the `db2ed_clonedb`command as follows:

```
$ /opt/VRTS/bin/db2ed_clonedb -T target_db3 -o umount -d
```

Creating and working with snapplans using db2ed_vmchecksnap

A snapplan specifies snapshot scenarios for a database (such as `online_snapshot`, `online_mirror`, or `offline`). You can name a snapplan file whatever you choose.

You can use the option to create the snapplan and set default values for the parameters. You may then modify the snapplan file using a text editor.

You can also use the command to validate, copy, list, or remove a snapplan and check the storage to make sure it is configured appropriately for the Database FlashSnap feature.

See "Validating a snapplan (db2ed_vmchecksnap)" on page 220.

Note: You must have the Enterprise Edition of Veritas Storage Foundation for DB2 to use this command.

Snapplan parameters

When using `db2ed_vmchecksnap -o setdefaults` option to create the snapplan, the following parameters are set:

Table A-4 Snapplan parameters

Parameter	Value
SNAPSHOT_VERSION	Specifies the snapshot version for this release of Veritas Storage Foundation for DB2
PRIMARY_HOST	Specifies the name of the host where the primary database resides.
SECONDARY_HOST	Specifies the name of the host where the clone database will reside. In a Veritas Storage Foundation for DB2 environment, if the primary and secondary hosts are the same, the snapshot volumes will not be deported.
PRIMARY_DG	Specifies the name of the Volume Manager disk group used by the primary database.
SNAPSHOT_DG	Specifies the name of the disk group containing the snapshot volumes. In a Veritas Storage Foundation for DB2 environment, the snapshot volumes will be put into this disk group on the primary host and deported if the primary and secondary hosts are different. The secondary host will import this disk group to start a clone database.
DB2DATABASE	Specifies the name of the DB2 database.
DB2HOME	Specifies the home directory of the database.

Table A-4 Snapplan parameters *(continued)*

Parameter	Value
ARCHIVELOG_DEST	Specifies the full path of the archive logs. There are several archive log destinations that can be used for database recovery if you are multiplexing the archive logs. You must specify which archive log destination to use. It is recommended that you have the archive log destination on a separate volume if SNAPSHOT_ARCHIVE_LOG is **yes** .
SNAPSHOT_MODE	**online_snapshot**, **online_mirror**, or **offline** Specifies whether the snapshot mode should be online_snapshot, online_mirror, or offline when the snapshot is created. By default, the SNAPSHOT_MODE is online_snapshot. If the snapshot is created while the SNAPSHOT_MODE for the database is set to online_snapshot or online_mirror, the db2ed_vmsnap command will first put the database into WRITE SUSPEND mode. After db2ed_vmsnap finishes creating the snapshot, it will take the database out of WRITE SUSPEND mode. If the database is offline, it is not necessary to put the database into WRITE SUSPEND mode. If the SNAPSHOT_MODE is offline, the secondary host must be different than the primary host. If the SNAPSHOT_MODE is online_mirror, the primary and secondary host must be the same, and the ALLOW_REVERSE_RESYNC parameter must be set to yes in the snapplan. If the SNAPSHOT_MODE is online_snapshot, the snapshot can be created on either the primary or a secondary host.
SNAPSHOT_PLAN_FOR	The default value is **database** and cannot be changed. Specifies the database object for which you want to create a snapshot.
SNAPSHOT_PLEX_TAG	Specifies the name of the tag set to the plexes that will be used by db2ed_vmsnap to take the snapshot. The db2ed_vmchecksnap command will use this tag name to search if all the volumes in the database have the plexes with this tag name set.

Table A-4 Snapplan parameters *(continued)*

Parameter	Value
SNAPSHOT_VOL_PREFIX	Specifies the snapshot volume prefix. Use this variable to specify a prefix for the snapshot volumes split from the primary disk group. A volume name cannot be more than 32 characters.
ALLOW_REVERSE_RESYNC	**yes** or **no** By default, reverse resynchronization is off (set equal to no). If it is set to yes, this parameter allows you to restore the original volume from a snapshot. The original database, however, must be down for this operation.
SNAPSHOT_MIRROR	Specifies the number of plexes to be snapshot. The default value is 1.

Creating a snapplan

Before creating a snapplan, the following conditions must be met:

Prerequisites

- You must be the DB2 instance owner.
- The disk group must be version 110 or later. For more information on disk group versions, see the vxdg(1M) manual page.
- Be sure that a DCO and DCO volume are associated with the volume(s) for which you are creating the snapshot.
- Snapshot plexes and their associated DCO logs should be on different disks than the original plexes, and should be configured correctly for creating snapshots by the system administrator.
- Persistent FastResync must be enabled on the existing database volumes and disks must be assigned for the snapshot volumes.
- The database must be running with LOGRETAIN mode on.

Usage notes

- The snapplan must be created on the primary host.
- After creating the snapplan using the `db2ed_vmchecksnap` command, you can use a text editor to review and update the file, if necessary.
- It is recommended that you create a local working directory to store your snapplans in.
- See the `db2ed_vmchecksnap`(1M) online manual page for more information.
- If `SNAPSHOT_MODE` is set to `online_snapshot`, the clone database can be recovered on either the primary or a secondary host. If `SNAPSHOT_MODE` is set to `online_mirror`, the primary host must be the same as the secondary host. If `SNAPSHOT_MODE` is set to `offline`, the clone database must be recovered on a secondary host.
- Database FlashSnap commands are not supported for partitioned DB2 databases in an SMP or MPP environment.

Table A-5 lists the options for creating a snapplan.

Table A-5 Options for creating a snapplan

Option	Description
`-D DB2DATABASE`	Specifies the name of the DB2 database for which a snapshot image will be created.
`-I DB2INSTANCE`	Specifies the name of the DB2 instance.
`-f SNAPPLAN`	Specifies the local path or the full path of the snapplan that you are creating.
`-o setdefaults`	Creates a default snapplan. This option can be used with the `-o validate` option to validate that the configuration is correct.
`-o validate`	Validates each parameter in the snapplan and checks whether the snapshot volumes have been configured correctly for creating snapshots, and copies the snapplan to the repository.
`-o list`	Lists all the snapplans associated with a specific $DB2DATABASE.
`-o copy`	Copies the snapplan from the repository to your current local directory.

Table A-5 Options for creating a snapplan *(continued)*

Option	Description
`-o remove`	Removes the snapplan from the repository and your local directory.
`-t SECONDARY_HOST`	Specifies the name of the host to which the snapshot image will be deported. If it is the same as the primary server, the snapshot volumes will not be deported. This argument is required if `-o setdefaults` is used. It is ignored if specified for `-o validate`.
`[-p plex_tag]`	Specifies the tag name for the plexes used to create the snapshot. If it is not specified, `db2ed_flashsnap` is used as the default plex tag.

To create a snapplan and set the default values for a single host

- Use the `db2ed_vmchecksnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap1 \
-o setdefaults -t host1 -p PRODtag
```

```
Snapplan snap1 for PROD.
=====================================================
SNAPSHOT_VERSION=5.0
PRIMARY_HOST=host1
SECONDARY_HOST=host1
PRIMARY_DG=PRODdg
SNAPSHOT_DG=SNAP_PRODdg
DB2DATABASE=PROD
DB2HOME=/PROD_HOME
REDOLOG_DEST=/PROD_HOME/inst1/NODE0000/SQL00001/SQLOGDIR/
SNAPSHOT_MODE=online_snapshot
SNAPSHOT_PLAN_FOR=database
SNAPSHOT_PLEX_TAG=PRODtag
SNAPSHOT_MIRROR=1
SNAPSHOT_VOL_PREFIX=SNAP_
ALLOW_REVERSE_RESYNC=no
```

To create a snapplan and set the default values in a two-host configuration

- Use the `db2ed_vmchecksnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD \
-f snap2 -o setdefaults -t host2 -p PRODtag


Snapplan snap2 for PROD.
=====================================================
SNAPSHOT_VERSION=4.0
PRIMARY_HOST=host1
SECONDARY_HOST=host2
PRIMARY_DG=PRODdg
SNAPSHOT_DG=SNAP_PRODdg
DB2DATABASE=PROD
DB2HOME=/PROD_HOME
REDOLOG_DEST=/PROD_HOME/inst1/NODE0000/SQL00001/SQLOGDIR/
SNAPSHOT_MODE=online_snapshot
SNAPSHOT_PLAN_FOR=database
SNAPSHOT_PLEX_TAG=PRODtag
SNAPSHOT_MIRROR=1
SNAPSHOT_VOL_PREFIX=SNAP_
ALLOW_REVERSE_RESYNC=no
```

Validating a snapplan

You can use the `db2ed_vmchecksnap` command with the `-o validate` option to validate a snapplan and check the storage to make sure it is configured appropriately for the Database FlashSnap feature.

To validate a snapplan for a snapshot image to be used on the primary host

- Use the `db2ed_vmchecksnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap1 -o validate


PRIMARY_HOST is host1
SECONDARY_HOST is host1
The version of PRIMARY_DG-PRODdg is 110.
SNAPSHOT_DB is SNAP_PRODdg
SNAPSHOT_PLAN_FOR is database
Examining DB2 volume and disk layout for snapshot
Volume prod_db on PRODdg is ready for snapshot.
Original plex and DCO log for prod_db is on PRODdg01.
Snapshot plex and DCO log for prod_db is on PRODdg02.
```

```
SNAP_PRODdg for snapshot will include: PRODdg02
ALLOW_REVERSE_RESYNC is yes
The snapplan snap1 has been created.
```

To validate a snapplan for a snapshot image to be used on the secondary host

- Use the `db2ed_vmchecksnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap2 -o validate
```

```
PRIMARY_HOST is host1
SECONDARY_HOST is host2
The version of PRIMARY_DG-PRODdg is 110.
SNAPSHOT_DB is SNAP_PRODdg
SNAPSHOT_PLAN_FOR is database
Examining DB2 volume and disk layout for snapshot.
Volume prod_db on PRODdg is ready for snapshot.
Original plex and DCO log for prod_db is on PRODdg01.
Snapshot plex and DCO log for prod_db is on PRODdg02.
SNAP_PRODdg for snapshot will include: PRODdg02
ALLOW_REVERSE_RESYNC is yes
The snapplan snap2 has been created.
```

Listing and viewing snapplans using db2ed_vmchecksnap

The `db2ed_vmchecksnap` command allows you to list and view existing snapplans.

To list all available snapplans for a specific DB2 database

- Use the command as follows:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -o list

The following snapplan(s) are available for PROD:

SNAP_PLAN     SNAP_STATUS     DB_STATUS    SNAP_READY

snap1         init_full       -            yes

snap2         init_full       -            yes

snap3         init_full       -            yes
```

The command output displays all available snapplans, their snapshot status (`SNAP_STATUS`), database status (`DB_STATUS`), and whether a snapshot may be taken (`SNAP_READY`).

See "Database FlashSnap snapshot status and database status" on page 335.

To view a snapplan

- Use the `db2ed_vmchecksnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap1 -o list


SNAPSHOT_VERSION=4.0
PRIMARY_HOST=host1
SECONDARY_HOST=host1
PRIMARY_DG=PRODdg
SNAPSHOT_DG=SNAP_PRODdg
DB2DATABASE=PROD
DB2HOME=/PROD_HOME
REDOLOG_DEST=/PROD_HOME/inst1/NODE0000/SQL00001/SQLOGDIR/
SNAPSHOT_MODE=online_snapshot
SNAPSHOT_PLAN_FOR=database
SNAPSHOT_PLEX_TAG=PRODtag
SNAPSHOT_MIRROR=1
SNAPSHOT_VOL_PREFIX=SNAP_
ALLOW_REVERSE_RESYNC=yes
STORAGE_INFO
PRODdg02
SNAP_PLEX=prod_db-02
STATUS_INFO
SNAP_STATUS=init_full
DB_STATUS=init
LOCAL_SNAPPLAN=/export/snap_dir/snap1
```

Copying or removing a snapplan using db2ed_vmchecksnap

The `db2ed_vmchecksnap` command allows you to copy or remove snapplans.

To copy a snapplan from the repository to your local directory

- Assuming that the snapplan is not already present in your local directory, use the `db2ed_vmchecksnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap1 -o copy

Copying 'snap1' to '/export/snap_dir'
```

To remove a snapplan

- Use the `db2ed_vmchecksnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmchecksnap -D PROD -f snap1 -o remove

The snapplan snap1 has been removed from the repository.
```

Creating, resynchronizing, or reverse resynchronizing a snapshot database using db2ed_vmsnap

You can use the `db2ed_vmsnap` command to create a snapshot image of a database. The snapshot can be used locally or on another host that is physically attached to the shared storage. You can also resynchronize the snapshot image back to the primary database.

Before creating, resynchronizing, or reverse resynchronizing a snapshot database, review the following information:

Prerequisites

- You must be logged in as the DB2 instance owner.
- You must create and validate a snapplan using `db2ed_vmchecksnap` before you can create a snapshot image with `db2ed_vmsnap`.

Usage notes

- The `db2ed_vmsnap` command can only be used on the primary host.
- When creating a snapshot volume, create the snapshot on a separate controller and on separate disks from the primary volume.
- The online redo log must be included in the snapshot if a clone database is to be started.
- Resynchronization speed varies based on the amount of data changed in both the primary and secondary volumes when the mirror is broken off.
- See the `db2ed_vmsnap`(1M) manual page for more information.

Options for the `db2ed_vmsnap` command are:

Table A-6 db2ed_vmsnap command options

Option	Description
`-D DB2DATABASE`	Specifies the name of the DB2 database for which a snapshot image will be created.
`-f SNAPPLAN`	Specifies the name of the snapplan you are using.

Table A-6 db2ed_vmsnap command options *(continued)*

Option	Description
`-o snapshot [-F] | resync`	Specifies whether to create a snapshot or synchronize the snapshot image with the current database image. The -F option prepares the volumes for being snapshot and forces snapshot creation.
`-o reverse_resync_begin`	Begins reverse resynchronization.
`-o reverse_resync_commit`	Commits the reverse resynchronization changes after you have verified that they are acceptable.
`-o reverse_resync_abort`	Aborts reverse resynchronization and mounts the original volumes back with the file systems that are configured to use the volume.

To create a snapshot image on the primary host

- Use the `db2ed_vmsnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap1 -o snapshot

db2ed_vmsnap started at 2006-03-10 13:10:54
DB20000I  The SET WRITE command completed successfully.
DB20000I  The SET WRITE command completed successfully.
A snapshot of DB2DATABASE PROD is in DG SNAP_PRODdg.
Snapplan snap1 is used for the snapshot.
DB2 server (server_name) is kagu

If -r <relocate_path> is used in db2ed_vmclonedb,
        make sure <relocate_path> is created and owned by DB2
        Instance Owner. Otherwise, the following mount points
        need to be created and owned by DB2 Instance Owner:

        /PROD_HOME.
        /PROD_TBS.

db2ed_vmsnap ended at 2006-03-10 13:11:17
```

To resynchronize a snapshot to your database

- Use the `db2ed_vmsnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap1 -o resync

db2ed_vmsnap started at 2005-03-15 10:07:10

The option resync has been completed.

db2ed_vmsnap ended at 2005-03-15 10:07:21
```

To resynchronize your database to a snapshot

1

2 Assuming the mount point for the primary database was created and owned by the DB2 instance owner before mounting the VxFS file system, use the `db2ed_vmsnap`command as follows:

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap1 \
-o reverse_resync_begin

db2ed_vmsnap started at 2004-05-26 13:37:11

Files and control structures were changed successfully.

Database was catalogued successfully.

DBT1000I  The tool completed successfully.

db2ed_vmsnap ended at 2004-05-26 13:37:48
```

To abort resynchronizing your database to a snapshot

- Use the `db2ed_vmsnap` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap1 -o \
reverse_resync_abort

db2ed_vmsnap started at 2004-05-26 13:38:16

DB20000I  The FORCE APPLICATION command completed successfully.

DB21024I  This command is asynchronous and may not be effective

immediately.

DB20000I  The UNCATALOG DATABASE command completed successfully.

DB21056W  Directory changes may not be effective until the directory
cache is refreshed.

Files and control structures were changed successfully.

Database was catalogued successfully.

DBT1000I  The tool completed successfully.

DB20000I  The UNCATALOG DATABASE command completed successfully.

DB21056W  Directory changes may not be effective until the directory
cache is refreshed.

DB20000I  The CATALOG DATABASE command completed successfully.

DB21056W  Directory changes may not be effective until the directory
cache is refreshed.

The option reverse_resync_abort has been completed.

db2ed_vmsnap ended at 2004-05-26 13:39:31
```

This option is only allowed when `reverse_resync_begin` has been run. It is not allowed if `reverse_resync_commit` has been executed.

If your snapshot was created with the snapshot mode set to `online_mirror`, you cannot run `db2ed_vmsnap -o reverse_resync_begin` after running `db2ed_vmsnap -o reverse_resync_abort`. You must take a new snapshot before performing reverse resynchronization again.

To commit reverse resynchronization changes

- Use the command as follows:

Warning: Upon completion of reverse resynchronization, the content of the original database is discarded. Storage Checkpoints taken on either the original database or the clone database before or after the snapshot was created are discarded. Storage Checkpoints taken before the snapshot was created are preserved. The `db2ed_vmsnap -o reverse_resync_commit` command cannot be undone and should be used with extreme caution.

```
$ /opt/VRTS/bin/db2ed_vmsnap -D PROD -f snap1 -o \
reverse_resync_commit

db2ed_vmsnap started at 2004-05-26 13:40:32

DB20000I  The FORCE APPLICATION command completed successfully.

DB21024I  This command is asynchronous and may not be effective
immediately.

DB20000I  The UNCATALOG DATABASE command completed successfully.

DB21056W  Directory changes may not be effective until the
directory cache is refreshed.

DB20000I  The CATALOG DATABASE command completed successfully.

DB21056W  Directory changes may not be effective until the
directory cache is refreshed.

db2ed_vmsnap ended at 2004-05-26 13:41:35
```

This option is only allowed after `reverse_resync_begin` has been run.

Creating or shutting down a clone database using db2ed_vmclonedb

You can use the command to create or shutdown a clone database on either the primary or secondary host using snapshot volumes from the primary host.

Before creating or shutting down a clone database, the following conditions must be met:

Prerequisites

- You must be logged in as the DB2 instance owner to use `db2ed_vmclonedb` command.
- Before you can use the `db2ed_vmclonedb` command, you must create and validate a snapplan and create a snapshot.
- The volume snapshot must contain the entire database.
- The system administrator must provide the database administrator with access to the necessary volumes and mount points.
- Before you can use the `db2ed_vmclonedb` command with the `-r relocate_path` option (which specifies the initial mount point for the snapshot image), the system administrator must create the mount point and then change the owner to the DB2 instance owner.
- The `SNAPSHOT_MODE` must be `online_snapshot` or `offline`. If `SNAPSHOT_MODE` is set to `offline`, a two-host configuration is required.

Usage notes

- The `db2ed_vmclonedb` command can be used on the secondary host.
- In a single-host configuration, the primary and secondary hosts are the same.
- In a single-host configuration, `-r relocate_path` is required.
- In a two-host configuration, the `server_name=`*`svr_name`* option is required.
- See the `db2ed_vmclonedb`(1M) manual page for more information.

Options for `db2ed_vmclonedb` are:

Table A-7 db2ed_vmclonedb options

Option	Description
`-D DB2DATABASE`	Specifies the name of the DB2 database for which a snapshot image will be created.
`-I DB2INSTANCE`	Specifies the name of the DB2 instance used to create the snapshot. This is required when cloning a database under a different instance.
`-g snap_dg`	Specifies the name of the disk group that contains all snapshot volumes.
`-o mount`	Mounts the file systems so you can use them to do a backup.
`-o mountdb`	Starts the database to allow manual database recovery.
`-o recoverdb`	Automatically recovers the database.

Table A-7 db2ed_vmclonedb options *(continued)*

Option	Description
`-o restartdb`	Restarts the database if the clone database is shut down. A clone database must exist to use the `-o restartdb` option.
`-o update_status`	Updates the database status information in the repository. This option is required only after the database has been manually recovered (`-o mountdb`).
`-o umount`	Shuts down the clone database and unmounts all snapshot files.
`server_name=`*`svr_name`*	Specifies the host on which the primary DB2 database runs.
`-f SNAPPLAN`	Indicates the name of the snapplan that you are using.
`-r` *`relocate_path`*	Specifies the initial mount point for the snapshot image. If you are creating a clone in a single-host configuration, -r is required. Otherwise, it is an optional argument. The system administrator needs to create and change the owner of this mount point to the DB2 instance owner. The `db2ed_vmclonedb` command will fail if the DB2 instance owner does not have access rights to this mount point. If `-r relocate_path` is used with the `-o mount | mountdb | reccoverdb` options, it will also be required to restart or unmount the clone database.

To clone the primary database automatically in a single-host configuration

- Use the `db2ed_vmclonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o recoverdb,new_db=NEWPROD,server_name=db2svr -f snap1 -r /clone

db2ed_vmclonedb started at 2004-04-01 13:18:02

Mounting /clone/testvol on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol.

Mounting /clone/udb_home on
/dev/vx/dsk/SNAP_PRODdg/SNAP_udb_home.

Files and control structures were changed successfully.
Database was catalogued successfully.

DBT1000I  The tool completed successfully.

Database relocation was successful.

DBT1000I  The tool completed successfully.

DB20000I  The CATALOG DATABASE command completed successfully.

DB21056W  Directory changes may not be effective until the directory
cache is refreshed.

db2ed_vmclonedb ended at 2004-04-01 13:18:40
```

To clone the primary database on a secondary host automatically in a two-host configuration

- Use the `db2ed_vmclonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o recoverdb,new_db=NEWPROD,server_name=db2svr -f snap2

db2ed_vmclonedb started at 2004-04-06 07:54:09

Mounting /clone/testvol on
/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol.

Mounting /clone/udb_home on

/dev/vx/dsk/SNAP_PRODdg/SNAP_udb_home.

Files and control structures were changed successfully.

Database was catalogued successfully.

DBT1000I  The tool completed successfully.

Database relocation was successful.

DBT1000I  The tool completed successfully.

db2ed_vmclonedb ended at 2004-04-06 07:55:00
```

To clone the primary database manually in a single-host configuration

1. Mount the file systems.
2. Create a clone using the `db2ed_vmclonedb` command.

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o mount,new_db=NEWPROD -f snap1,server_name=db2svr -r /clone

db2ed_vmclonedb started at 2004-04-01 13:30:12

Mounting /clone/testvol on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol.

Mounting /clone/udb_home on
/dev/vx/dsk/SNAP_PRODdg/SNAP_udb_home.

db2ed_vmclonedb ended at 2004-04-01 13:30:24
```

3 Recover the database manually.

4 Update the snapshot status (`database_recovered`) for the clone database on the primary host after manual recovery has been completed.

```
$ /opt/VRTS/bin/db2ed_vmclonedb -o
update_status,new_db=NEWPROD,server_name=db2svr -f snap1 -r /clone

db2ed_vmclonedb started at 2004-04-01 13:37:09

The snapshot status has been updated.

db2ed_vmclonedb ended at 2004-04-01 13:37:23
```

To clone the primary database manually in a two-host configuration

1 Mount the file systems.

2 Create a clone using the `db2ed_vmclonedb` command.

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o mount,new_db=NEWPROD,server_name=db2svr -f snap2

db2ed_vmclonedb started at 2004-04-06 09:06:09

Mounting /clone/testvol on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol.

Mounting /clone/udb_home on

/dev/vx/dsk/SNAP_PRODdg/SNAP_udb_home.

db2ed_vmclonedb ended at 2004-04-06 09:06:43
```

3 Recover the database manually.

4 Update the snapshot status (`database_recovered`) for the clone database on the secondary host after manual recovery has been completed.

```
$ /opt/VRTS/bin/db2ed_vmclonedb -o \
update_status,new_db=NEWPROD,server_name=db2svr -f snap2

db2ed_vmclonedb started at 2004-04-06 09:22:27

The snapshot status has been updated.

db2ed_vmclonedb ended at 2004-04-06 09:22:40
```

To shut down the clone database and unmount all snapshot file systems in a single-host configuration

- Use the command as follows:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -o umount,new_db=NEWPROD,\
server_name=db2svr -f snap1 -r /clone

db2ed_vmclonedb started at 2004-04-02 15:11:22

Umounting /clone/prod_db.

Umounting /clone/prod_ar.

db2ed_vmclonedb ended at 2004-04-02 15:11:47
```

To shut down the clone database and unmount all snapshot file systems in a two-host configuration

- Use the `db2ed_vmclonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D prod_db -f snap2 -o \
umount,new_db=clonedb,server_name=db2svr -r /clone

db2ed_vmclonedb started at 2004-04-09 23:09:21

Umounting /clone/arch.

Umounting /clone/prod_db.

db2ed_vmclonedb ended at 2004-04-09 23:09:50
```

  This shuts down the clone database, unmounts file systems, and deports the snapshot disk group for a clone on a secondary host.

To restart a clone database in a single-host configuration

- Use the `db2ed_vmclonedb` command as follows:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o restartdb,new_db=NEWPROD,server_name=db2svr -f snap1 -r /clone

db2ed_vmclonedb started at 2004-05-26 13:20:58

Mounting /clone/testvol01 on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol01.

Mounting /clone/testvol02 on
/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol02.

Mounting /clone/testvol03 on
/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol03.

Mounting /clone/testvol04 on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol04.

Mounting /clone/testvol05 on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol05.

Mounting /clone/udb_home on

/dev/vx/dsk/SNAP_PRODdg/SNAP_udb_home.

Files and control structures were changed successfully.

Database was catalogued successfully.

DBT1000I  The tool completed successfully.

db2ed_vmclonedb ended at 2004-05-26 13:21:48
```

To restart a clone database in a two-host configuration

- Use the command as follows:

```
$ /opt/VRTS/bin/db2ed_vmclonedb -D PROD -g SNAP_PRODdg \
-o restartdb,new_db=NEWPROD,server_name=db2svr -f snap2

db2ed_vmclonedb started at 2004-05-26 10:14:21

Mounting /clone/testvol01 on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol01.

Mounting /clone/testvol02 on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol02.

Mounting /clone/testvol03 on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol03.

Mounting /clone/testvol04 on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol04.

Mounting /clone/testvol05 on

/dev/vx/dsk/SNAP_PRODdg/SNAP_testvol05.

Mounting /clone/udb_home on

/dev/vx/dsk/SNAP_PRODdg/SNAP_udb_home.

Files and control structures were changed successfully.

Database was catalogued successfully.

DBT1000I  The tool completed successfully.

db2ed_vmclonedb ended at 2004-05-26 10:15:15
```

Managing log files using edgetmsg2

You can use the `edgetmsg2` utility to manage message log files. You can use the `edgetmsg2` utility to write a message to a log file or to the console, read the log file and print to the console, and display the available log files.

Before managing log files with the `edgetmsg2` command, review the following information:

Prerequisites
- You must be logged in as the instance owner or root to use this command.

Usage notes
- The default log file for a database is located in the following directory:
  `/etc/vx/vxdbed/logs/sfua_database.log`
  where *database* is the `DB2DATABASE`.
- By default, only messages with a severity equal to or greater than `ERROR` will be logged.
- See the `edgetmsg2`(1M) manual page for more information.

Table A-8 lists options for `edgetmsg2`.

Table A-8 edgetmsg2 options

Option	Description
`-s set_num`	Specifies the message catalogue set number. The default is 1.
`-M msgid[:severity]`	Specifies the message ID and severity to be printed.
`-f msg_catalog | logfile | log_directory`	Specifies the message catalogue path, log file, or log directory.
`-v severity | severity`	Overwrites the minimum log severity or creates a severity filter. The severity values are either 0-8 or 100-108.
`-p`	Pauses the cursor at the end of a display message. By default, a line feed is added to each display message. Use the `-p` option to indicate that no line feed is to be added.
`-o list [,suppress_time]`	Displays the content of a log file. You can specify `,suppress_time` to exclude time information in the utility output.
`-o report[,no_archive]`	Displays the available log files. You can specify ,no_archive to exclude log files from the utility output.
`-t from_time[,to_time]`	Reduces the length of the utility output by specifying the time range to include. This option must be used together with the `-o list` option. Use the following format: `yyy-mm-dd HH:MM:SS`.
`-I DB2INSTANCE`	Specifies the DB2 instance.
`-D DB2DATABASE`	Specifies the DB2 database.

Table A-8 edgetmsg2 options *(continued)*

Option	Description
"default format string"	Specifies the C language `printf()` format string.
[args]	Specifies arguments for the format string conversion characters.

To print a message

- Use the `edgetmsg2` command as follows:

```
$ /opt/VRTS/bin/edgetmsg2 [-s set_num] \

[-M msgid[:severity]] \
[-f msg_catalog] [-v severity] [-p] [-m value] \
["default format string" [args]]
```

To read a message log file

- Use the `edgetmsg2` command as follows:

```
$ /opt/VRTS/bin/edgetmsg2 -o list[,suppress_time] \
-I DB2INSTANCE -D DB2DATABASE | [-f logfile] \
[-v severity] [-t from_time,to_time]
```

To list available log files

- Use the `edgetmsg2` command as follows:

```
$ /opt/VRTS/bin/edgetmsg2 -o report[,no_archive] \
[-f log_directory]
```

Displaying I/O mapping and statistics using vxstorage_stats

You can use the `vxstorage_stats` command to display I/O mapping and statistics about Veritas File System files one file at a time. The statistics are recorded only for VxFS files and VxVM volumes. These statistics show I/O activity.

Before displaying I/O mapping and statistics, the following conditions must be met:

Prerequisites
- You may be logged in as either the database administrator or `root`.

Command usage for `vxstorage_stats` is as follows:

```
$ /opt/VRTS/bin/vxstorage_stats [-m] [-s] [-i interval -c count ] \
-f file_name
```

Table A-9 lists options for the `vxstorage_stats` command.

Table A-9 vxstorage_stats command options

Option	Description
`-m`	Displays the I/O topology for the specified file.
`-s`	Displays the file statistics for the specified file.
`-c` count	Specifies the number of times to display statistics.
`-i` interval	Specifies the interval frequency for displaying updated I/O statistics.
`-f` filename	Specifies the file to display I/O mapping and statistics for.

To display I/O mapping information

- Use the `vxstorage_stats` command with the `-m` option as follows:

  ```
  $ /opt/VRTS/bin/vxstorage_stats -m -f \
  /data/system01.dbf
  ```

  For file type (`fi`), the `SIZE` column is number of bytes, and for volume (`v`), plex (`pl`), sub-disk (`sd`), and physical disk (`da`), the `SIZE` column is in 512-byte blocks. Stripe sizes are given in sectors.

```
TY NAME                              DESCRIPTION  SIZE       EXT OFFSET
fi /data/system01.dbf                FILE         536870912  1   1048576
v  /dev/vx/rdsk/db2dg/mydb01_user    MIRROR       16777216   0   0
pl vxvm:db2dg/mydb01_user-01         STRIPE       16777600   0   0
rd /dev/vx/rdmp/c5t2d0s2             HOST_DEVICE  0          0   0
sd /dev/rdsk/c5t2d0s2                DEVICE       3355520    0   0
da c5t2d0                            DISK         71120064   0   0
rd /dev/vx/rdmp/c5t3d0s2             HOST_DEVICE  0          0   0
sd /dev/rdsk/c5t3d0s2                DEVICE       3355520    0   0
da  c5t3d0                           DISK         71120064   0   0
rd /dev/vx/rdmp/c5t4d0s2             HOST_DEVICE  0          0   0
sd /dev/rdsk/c5t4d0s2                DEVICE       3355520    0   0
da c5t4d0                            DISK         71120064   0   0
rd /dev/vx/rdmp/c5t0d0s2             HOST_DEVICE  0          0   0
sd /dev/rdsk/c5t0d0s2                DEVICE       3355520    0   0
da c5t0d0                            DISK         71120064   0   0
rd /dev/vx/rdmp/c5t1d0s2             HOST_DEVICE  0          0   0
```

```
sd /dev/rdsk/c5t1d0s2              DEVICE       3355520    0   0
da c5t1d0                          DISK         71120064   0   0
```

The output indicates that the file has a single extent, which usually indicates a Veritas Quick I/O file. The volume `mydb01_user`, where the file system is created, is a Veritas Volume Manager Volume with 5 column striping across 5 disks (`c5t0` to `c5t4`)

To display the entire I/O mapping and statistics for each I/O stack

- Use the `vxstorage_stats` command with the `-m` and `-s` options as follows:

```
$ /opt/VRTS/bin/vxstorage_stats -m -s -f /data/system01.dbf

TY NAME                 NSUB   DESCRIPTION    SIZE       OFFSET
PROPERTIES
fi /data/system01.dbf      1   FILE           262146048  1172128
Extents: 6 Sparse Extents:0
v  data_vol                2   MIRROR         8388608    0
pl vxvm:mapdg/data_vol-01  1   CONCAT_VOLUME  8388608    0
rd /dev/vx/rdmp/c1t10d0s2  1   PARTITION      8388608    0
sd /dev/rdsk/c1t10d0s2     1   PARTITION      35368272   0
da c1t10d0                 0   DISK           35368272   0
pl vxvm:mapdg/data_vol-03  1   CONCAT_VOLUME  8388608    0
rd /dev/vx/rdmp/c1t13d0s2  1   PARTITION      8388608    0
sd /dev/rdsk/c1t13d0s2     1   PARTITION      71127180   0
da c1t13d0                 0   DISK           71127180   0

                              OPERATIONS     FILE BLOCKS    AVG TIME(ms)
                                                            (512 byte)
OBJECT                      READ   WRITE   B_READ   B_WRITE  AVG_RD AVG_WR
/data/system01.dbf          615    19      20752    152      3.53   24.74
/dev/vx/rdsk/mapdg/data_vol 19444  33318   895903   1376825  9.26   16.14
vxvm:mapdg/data_vol-01      19444  33318   895903   1376825  9.24   14.00
/dev/rdsk/c1t10d0s2         19444  33318   895903   1376825  9.24   14.00
c1t10d0                     19444  33318   895903   1376825  9.24   14.00
vxvm:mapdg/data_vol-03      0      33318   0        1376825  0.00   14.18
/dev/rdsk/c1t13d0s2         0      33318   0        1376825  0.00   14.18
c1t13d0                     0      33318   0        1376825  0.00   14.18
```

For example, to display statistics two times with a time interval of two seconds:

```
$ /opt/VRTSdb2ed/bin/vxstorage_stats -s -i2 -c2 \
-f /data/system01.dbf
```

Identifying VxFS files to convert to Quick I/O using qio_getdbfiles

Note: This command is not available in Veritas Storage Foundation for DB2 on Linux.

You can use the `qio_getdbfiles` command to identify VxFS files before converting them to Quick I/O files. Only VxFS files may be converted to Quick I/O.

The `qio_getdbfiles` command queries the database and gathers a list of datafiles to be converted to Quick I/O. The command requires direct access to the database.

Before using the `qio_getdbfiles` command, the following conditions must be met:

Prerequisites

- To use this command for DB2, the `DB2DATABASE` environment variable must be set.
- You must be logged in as the instance owner.

Usage notes

- The `-T` option forces the behavior for a specific database type. The database options that are supported are `ora`, `syb`, and `db2`. Use this option in environments with more than one type of database.
- The `-a` option specifies that all datafiles should be included. By default, potential sparse files are excluded.
- See the `qio_getdbfiles`(1M) manual page for more information.
- See the `qio_getdbfiles`(1M) manual page for more information.

To identify the VxFS files to convert to Quick I/O

1 Use the `qio_getdbfiles` command as follows:

```
S /opt/VRTSdb2ed/bin/qio_getdbfiles [-T ora|syb|db2]

$ /opt/VRTSsybed/bin/qio_getdbfiles [-T syb] \
[-d <database_name>] [-m <master_device_pathname>]
```

where `-T syb` forces behavior for Sybase, <`database_name`> specifies the database device files, and <`master_device_pathname`> specifies the full path name of the master device for the Sybase ASE server.

The `qio_getdbfiles` command stores the filenames and file sizes in bytes in a file called `mkqio.dat`.

2 View the `mkqio.dat` file:

```
$ cat mkqio.dat
```

The `mkqio.dat` file contains the database filenames that can be converted to Quick I/O files. The format of the file is a list of paired file paths and file sizes. For example:

```
/database/dbfiles.001 1024000

/database/dbfiles.002 2048000
```

Converting VxFS files to Quick I/O using qio_convertdbfiles

After running `qio_getdbfiles`, you can use the `qio_convertdbfiles` command to convert database files to use Quick I/O. This command is for use with VxFS file systems only.

The `qio_convertdbfiles` command converts regular files or symbolic links that point to regular files on VxFS file systems to Quick I/O. The `qio_convertdbfiles` command converts only those files listed in the `mkqio.dat` file to Quick I/O. The `mkqio.dat` file is created by running `qio_getdbfiles`. It can also be created manually.

Before converting files, the following conditions must be met:

Prerequisites

- To use this command for DB2, the `DB2DATABASE` environment variable must be set.
- You must be logged in as the instance owner.
- Remove any non-VxFS files from mkqio.dat before running `qio_convertdbfiles`. The `qio_convertdbfiles` command will display an error message if any of the database files in mkqio.dat are not on a VxFS file system.

Usage notes

- The `qio_convertdbfiles` command expects all files to be owned by the database administrator.
- Converting existing database files to Quick I/O is not recommended if the files are fragmented. In this case, it is recommended that you create new files with the `qiomkfile` command (these files are guaranteed not to be fragmented) and then convert the data from the old files (using a command such as `dd`).
- Ensure that the database is shut down before running `qio_convertdbfiles`.
- See the `qio_convertdbfiles`(1M) manual page for more information.

Table A-10 lists options for the `qio_convertdbfiles` command.

Table A-10 qio_convertdbfiles command options

Option	Description
`-T`	Forces the behavior for a specific database type. The database options that are supported are `ora`, `syb`, and `db2`. Use this option in environments with more than one type of database.
`-a`	Changes regular files to Quick I/O files using absolute pathnames. Use this option when symbolic links need to point to absolute pathnames. By default, relative pathnames are used.
`-f`	Reports on current fragmentation levels for files listed in `mkqio.dat`. Fragmentation is reported at four levels: not fragmented, slightly fragmented, fragmented, and highly fragmented.
`-h`	Displays a help message.
`-i`	Creates extra links for all database files and log files in the /dev directory to support the SAP command.
`-u`	Changes Quick I/O files back to regular files.

To convert VxFS files to Quick I/O files

1 After running the `qio_getdbfiles` command, shut down the database:

Warning: Running `qio_convertdbfiles` with any option except `-f` while the database is up and running can cause severe problems for your database, including data loss and corruption. Make sure the database is shut down before running the `qio_convertdbfiles` command.

---

2 Run the `qio_convertdbfiles` command to convert the list of files in `mkqio.dat` to Quick I/O files:

```
$ /opt/VRTSdb2ed/bin/qio_convertdbfiles
```

You must remove any non-VxFS files from `mkqio.dat` before running `qio_convertdbfiles`. The `qio_convertdbfiles` command will display an error message if any of the database files in `mkqio.dat` are not on a VxFS file system.

3 Restart the database to access these database files using the Quick I/O interface.

To undo a previous run of qio_convertdbfiles

- Use the `qio_convertdbfiles` as follows:

```
$ /opt/VRTSdb2ed/bin/qio_convertdbfiles -u

.dbfile::cdev:vxfs: --> dbfile
```

This reverts a previous run of `qio_convertdbfiles` and changes Quick I/O files back to regular VxFS files.

If the database is up and running, an error message will be displayed stating that you need to shut it down before you can run `qio_convertdbfiles`.

Recreating Quick I/O files using qio_recreate

Note: This command is not available in Veritas Storage Foundation for DB2 on Linux.

You can use the `qio_recreate` command to automatically recreate Quick I/O files when the database is recovered.

Before converting files to Quick I/O, the following conditions must be met:

Prerequisites

- To use this command for DB2, the `DB2DATABASE` environment variable must be set.
  You must be logged in as the instance owner to use this command.

Usage notes

- The `qio_recreate` command expects to find a file named `mkqio.dat` in the directory where the command is run. The mkqio.dat file contains a list of the Quick I/O files used by the database and their sizes. If the `mkqio.dat` file is not in the directory, you will be prompted to create it using `qio_getdbfiles`.
  See "Identifying VxFS files to convert to Quick I/O using qio_getdbfiles" on page 327.
- The `qio_recreate` command supports conventional Quick I/O files only (that is, Quick I/O files in the following form: `file --> .file::cdev:vxfs:`). In creating a Quick I/O file, the `qio_convertdbfiles` command renames the regular VxFS file, `file`, to `.file` with the Quick I/O extension (`:cdev::vxfs:`) and creates a symbolic link to it. By default, the symbolic link uses a relative path name.
- There are no options for the `qio_recreate` command and no output is returned when the command runs successfully.
- See the `qio_recreate`(1M) manual page for more information.

The qio_recreate command follows these rules in recreating Quick I/O files when a database is recovered:

- If a Quick I/O file (`.file::cdev:vxfs:`) is missing, then `qio_recreate` recreates it.
- If both a symbolic link (`file`) and its associated Quick I/O file (`.file::cdev:vxfs:`) are missing, `qio_recreate` recreates both the symbolic link and the Quick I/O file.
- If a symbolic link (`file`) from a regular VxFS file to its associated Quick I/O file (`.file::cdev:vxfs:`) is missing, then `qio_recreate` recreates the symbolic link.
- If a Quick I/O file (`.file::cdev:vxfs:`) is missing and the regular VxFS file that is symbolically linked to it is not the same one that originally created it, then `qio_recreate` issues a warning message and does not recreate the Quick I/O file.
- If a Quick I/O file (`.file::cdev: vxfs:`) is smaller than the size listed in `mkqio.dat`, `qio_recreate` issues a warning message.

To automatically recreate Quick I/O files when the database is recovered

- Use the `qio_recreate` command as follows:

```
$ /opt/VRTSdb2ed/bin/qio_recreate
```

Appendix B

Using third-party software to back up files

This appendix includes the following topics:

About backing up and restoring Quick I/O files using Legato NetWorker

When Quick I/O files are created using the command `qiomkfile`, a hidden file with storage space allocated to it and a link are created. The link points to the Quick I/O interface of the hidden file. Using `qiomkfile` ensures that the space for the file is allocated in a contiguous manner, which typically improves performance.

Legato NetWorker does not follow symbolic links during backups because doing so would result in the data getting backed up twice: once using the symbolic link and once as the file itself. As a result, Quick I/O files must be backed up as two separate files and restored accordingly.

Because Legato NetWorker deletes and recreates files before they are restored, the restored files lose their contiguous allocation and could be restored as fragmented files with indirect extents. While this does not impact the integrity of the data being restored, it can degrade performance. Creating the file using `qiomkfile` before doing the backup does not resolve this problem because NetWorker deletes and recreates the file.

To avoid this potential performance degradation, Quick I/O files must be backed up and restored using the same methods used to back up and restore raw devices. This method involves using the NetWorker `rawasm` command to back up or `save` directories containing Quick I/O files. Because of the way the `rawasm` command works, NetWorker follows the Quick I/O symbolic link to back up the actual data

in the hidden file. Skip the hidden file to avoid backing up the data twice. During restore, NetWorker looks at the attributes of the saved file and restores it using `rawasm`, bypassing the file deletion and recreation steps.

For example, to view all files in the `db01` directory:

```
$ ls -al /db01

total 2192

drwxr-xr-x    2 root   root  96   Oct 20 17:39 .

drwxr-xr-x    9 root   root  8192   Oct 20 17:39 ..

-rw-r--r--     db2inst1   db2iadm1  1048576   Oct 20 17:39  .dbfile

lrwxrwxrwx    1 db2inst1   db2iadm1   22  Oct 20 17:39 dbfile    ->\

.dbfile::cdev:vxfs:
```

The command for backing up the `/db01` directory using `rawasm` would look like:

```
<< /db01 >>
rawasm: dbfile
skip: .dbfile
```

To restore the file, preallocate the Quick I/O file using the `qiomkfile` command and enter:

```
$ cd /db01
$ recover -a /db01/dbfile
```

Appendix C

Veritas Database FlashSnap status information

This appendix includes the following topics:

Database FlashSnap snapshot status and database status

The Veritas Database FlashSnap functionality provides both snapshot status information and snapshot database status information for various stages of snapplan and snapshot procedures. You can view the status information through the CLI and through the GUI.

For more information about Database FlashSnap GUI functionality, see the *Veritas Storage Foundation for DB2 Graphical User Interface Guide*.

You can obtain both the snapshot status and the database status from the command line using the `db2ed_vmchecksnap` command with the `-o` list option. The snapshot status and database status information may also appear in error messages.

Snapshot status details

To view snapshot status information from the command line, use the `db2ed_vmchecksnap` command with the `-o list` option to list all available snapplans for a specified database. Snapshot status information is displayed in the command output under the column heading `SNAP_STATUS`.

Table C-1 shows detailed information about each snapshot status (`SNAP_STATUS`) value.

Table C-1 Snapshot status values

SNAP_STATUS	Completed operations	Allowed operations
`init_full`	■ `db2ed_vmchecksnap -o validate` (successful) ■ `db2ed_vmsnap -o resync` (successful) ■ `db2ed_vmsnap -o reverse_resync_commit` (successful)	■ `db2ed_vmsnap -o snapshot`
`init_db`	■ `db2ed_vmchecksnap -o validate -f` *`snapplan`* (failed)	■ `db2ed_vmchecksnap -o validate | list -f` *`snapplan`* Ensure that your storage configuration has been set up correctly.
`invalid`	■ `db2ed_vmchecksnap -o validate` (failed)	■ `db2ed_vmchecksnap -o validate`
`snapshot_start`	■ `db2ed_vmsnap -o snapshot` (failed)	■ Contact your system administrator for help. Use Veritas Volume Manager commands to resynchronize the snapshot volumes, and use `db2ed_vmsnap -o snapshot -F` to force snapshot creation.
`snapshot_end`	■ `db2ed_vmsnap -o snapshot` (successful) ■ `db2ed_vmsnap -o reverse_resync_abort` (successful)	■ `db2ed_vmsnap -o resync` ■ `db2ed_vmsnap -o reverse_resync_begin` ■ `db2ed_vmclonedb -o mount|mountdb|recoverdb`

`snapshot_vol_start`
`snapshot_vol_end`
`snapshot_dg_start`
`snapshot_dg_end`
	■ `db2ed_vmsnap -o snapshot` (failed)	■ Re-run `db2ed_vmsnap -o snapshot`

`resync_vol_start`
`resync_vol_end`
`snapshot_dg_start`
`snapshot_dg_end`
	■ `db2ed_vmsnap -o resync` (failed)	■ Re-run `db2ed_vmsnap -o resync`

Table C-1 Snapshot status values *(continued)*

SNAP_STATUS	Completed operations	Allowed operations
`resync_start`	■ `db2ed_vmsnap -o resync` (failed)	■ Contact your system administrator for help. Use Veritas Volume Manager commands to resynchronize the snapshot volumes, and use `db2ed_vmsnap -o snapshot -F` to force snapshot creation.
`reverse_resync_begin_start`	■ `db2ed_vmsnap -o reverse_resync_begin` (failed)	■ Contact Veritas support.
`reverse_resync_begin_end`	■ `db2ed_vmsnap -o reverse_resync_begin` (successful)	■ `db2ed_vmsnap -o reverse_resync_abort` ■ `db2ed_vmsnap -o reverse_resync_commit`
`reverse_resync_commit_start`	■ `db2ed_vmsnap -o reverse_resync_commit` (failed)	■ Contact Veritas support.
`mount_end`	■ `db2ed_vmclonedb -o mount` (successful)	■ `db2ed_vmclonedb -o umount`
`restartdb_start`	■ `db2ed_vmclonedb -o restartdb` (failed)	■ `db2ed_vmclonedb -o umount` ■ Start the snapshot database manually.
`restartdb_end`	■ `db2ed_vmclonedb -o restartdb` (successful)	■ `db2ed_vmclonedb -o umount`
`recoverdb_start`	■ `db2ed_vmclonedb -o recoverdb` (failed)	■ Recover the snapshot database manually, then run `db2ed_vmclonedb -o update_status` ■ `db2ed_vmclonedb -o umount`
`recoverdb_end`	■ `db2ed_vmclonedb -o recoverdb` (successful)	■ `db2ed_vmclonedb -o umount`
`umount_start`	■ `db2ed_vmclonedb -o umount` (failed)	■ Verify that your file system(s) are not busy and retry the command.

Table C-1 Snapshot status values *(continued)*

SNAP_STATUS	Completed operations	Allowed operations
umount_end	■ db2ed_vmclonedb -o umount (successful)	■ db2ed_vmclonedb -o mount ■ db2ed_vmclonedb -o restartdb ■ db2ed_vmsnap -o resync ■ db2ed_vmsnap -o reverse_resync_begin

Database status details

To view snapshot status information from the command line, use the `db2ed_vmchecksnap` command with the `-o list` option to list all available snapplans for a specified database. Database status information is displayed in the command output under the column heading `DB_STATUS`.

Table C-2 shows detailed information about each database status (`DB_STATUS`) value.

Table C-2 Database status values

DB_STATUS	Completed operations
init	■ db2ed_vmchecksnap -o validate (successful) db2ed_vmsnap -o snapshot (successful) db2ed_vmsnap -o reverse_resync_begin (successful)
database_recovered	■ db2ed_vmclonedb -o recoverdb (successful)

Glossary

address-length pair	Identifies the starting block address and the length of an extent (in file system or logical blocks).
asynchronous I/O	A format of I/O that performs non-blocking reads and writes. This enables the system to handle multiple I/O requests simultaneously.
atomic operation	An operation that either succeeds completely or fails and leaves everything as it was before the operation was started. If the operation succeeds, all aspects of the operation take effect at once and the intermediate states of change are invisible. If any aspect of the operation fails, then the operation aborts without leaving partial changes.
block map	A file system is divided into fixed-size blocks when it is created. As data is written to a file, unused blocks are allocated in ranges of blocks, called extents. The extents are listed or pointed to from the inode. The term used for the data that represents how to translate an offset in a file to a file system block is the “block map” for the file.
boot disk	A disk used for booting an operating system.
buffered I/O	A mode of I/O operation (where I/O is any operation, program, or device that transfers data to or from a computer) that first transfers data into the Operating System buffer cache.
cache	Any memory used to reduce the time required to respond to an I/O request. The read cache holds data in anticipation that it will be requested by a client. The write cache holds data written until it can be safely stored on non-volatile storage media.
Cached Quick I/O	Cached Quick I/O allows databases to make more efficient use of large system memory while still maintaining the performance benefits of Quick I/O. Cached Quick I/O provides an efficient, selective buffering mechanism to complement asynchronous I/O.
cluster	A set of hosts that share a set of disks.
cluster-shareable disk group	A disk group in which the disks are shared between more than one host.
cold backup	The process of backing up a database that is not in active use.
command launcher	A graphical user interface (GUI) window that displays a list of tasks that can be performed by Veritas Volume Manager or other objects. Each task is listed with the object type, task (action), and a description of the task. A task is launched by

	clicking on the task in the Command Launcher. concatenation A Veritas Volume Manager layout style characterized by subdisks that are arranged sequentially and contiguously.
concurrent I/O	A form of Direct I/O that does not require file-level write locks when writing to a file. Concurrent I/O allows the relational database management system (RDBMS) to write to a given file concurrently.
configuration database	A set of records containing detailed information on existing Veritas Volume Manager objects (such as disk and volume attributes). A single copy of a configuration database is called a configuration copy.
container	A physical storage device that can be identified by a directory name, a device name, or a file name.
copy-on-write	A technique for preserving the original of some data. As data is modified by a write operation, the original copy of data is copied. Applicable to Storage Checkpoint technology, where original data, at the time of the Storage Checkpoint, must be copied from the file system to the Storage Checkpoint when it is to be overwritten. This preserves the frozen image of the file system in the Storage Checkpoint.
database	A database is a collection of information that is organized in a structured fashion. Two examples of databases are Relational Databases (such as Oracle, Sybase, or DB2), where data is stored in tables and generally accessed by one or more keys and Flat File Databases, where data is not generally broken up into tables and relationships. Databases generally provide tools and/or interfaces to retrieve data.
Decision Support Systems	Decision Support Systems (DSS) are computer-based systems used to model, identify, and solve problems, and make decisions.
defragmentation	The act of reorganizing data to reduce fragmentation. Data in file systems become fragmented over time.
device file	A block- or character-special file located in the /dev directory representing a device.
device name	The name of a device file, which represents a device. AIX syntax is Disk_#; HP-UX syntax is c#t#d#; Linux syntax is sda, where "a" could be any alphabetical letter; Solaris syntax is c#t#d#s#.
direct I/O	An unbuffered form of I/O that bypasses the kernel's buffering of data. With direct I/O, data is transferred directly between the disk and the user application.
Dirty Region Logging	The procedure by which the Veritas Volume Manager monitors and logs modifications to a plex. A bitmap of changed regions is kept in an associated subdisk called a log subdisk.

disk access name The name used to access a physical disk, such as Disk_1 on an AIX system, c1t1d1 on an HP-UX system, sda on a Linux system, or c0t0d0s0 on a Solaris system. The term device name can also be used to refer to the disk access name.

disk array A collection of disks logically and physically arranged into an object. Arrays provide benefits including data redundancy and improved performance.

disk cache A section of RAM that provides a cache between the disk and the application. Disk cache enables the computer to operate faster. Because retrieving data from hard disk can be slow, a disk caching program helps solve this problem by placing recently accessed data in the disk cache. Next time that data is needed, it may already be available in the disk cache; otherwise a time-consuming operation to the hard disk is necessary.

disk group A collection of disks that share a common configuration.

A disk group configuration is a set of records containing detailed information on existing Veritas Volume Manager objects (such as disk and volume attributes) and their relationships. Each disk group has an administrator-assigned name and an internally defined unique ID. The root disk group (rootdg) is a special private disk group

disk name A Veritas Volume Manager logical or administrative name chosen for the disk, such as disk03. The term disk media name is also used to refer to the disk name.

DMP See “Dynamic Multipathing.”

DSS See “Decision Support Systems.”

Dynamic Multipathing Dynamic Multipathing (DMP) is a Veritas Volume Manager feature that allows the use of multiple paths to the same storage device for load balancing and redundancy.

error handling Routines in a program that respond to errors. The measurement of quality in error handling is based on how the system informs the user of such conditions and what alternatives it provides for dealing with them.

evacuate Moving subdisks from the source disks to target disks.

exabyte A measure of memory or storage. An exabyte is approximately 1,000,000,000,000,000,000 bytes (technically 2 to the 60th power, or 1,152,921,504,606,846,976 bytes). Also EB.

extent A logical database attribute that defines a group of contiguous file system data blocks that are treated as a unit. An extent is defined by a starting block and a length.

extent attributes The extent allocation policies associated with a file and/or file system. For example, see “address-length pair.”

failover The act of moving a service from a failure state back to a running/available state. Services are generally applications running on machines and failover is the process of restarting these applications on a second system when the first has suffered a failure.

file system A collection of files organized together into a structure. File systems are based on a hierarchical structure consisting of directories and files.

file system block The fundamental minimum size of allocation in a file system.

fileset A collection of files within a file system.

fixed extent size An extent attribute associated with overriding the default allocation policy of the file system.

fragmentation Storage of data in non-contiguous areas on disk. As files are updated, new data is stored in available free space, which may not be contiguous. Fragmented files cause extra read/write head movement, slowing disk accesses.

gigabyte A measure of memory or storage. A gigabyte is approximately 1,000,000,000 bytes (technically, 2 to the 30th power, or 1,073,741,824 bytes). Also GB, Gbyte, and G-byte.

high availability (HA) The ability of a system to perform its function continuously (without significant interruption) for a significantly longer period of time than the combined reliabilities of its individual components. High availability is most often achieved through failure tolerance and inclusion of redundancy; from redundant disk to systems, networks, and entire sites.

hot backup The process of backing up a database that is online and in active use.

hot pluggable To pull a component out of a system and plug in a new one while the power is still on and the unit is still operating. Redundant systems can be designed to swap disk drives, circuit boards, power supplies, CPUs, or virtually anything else that is duplexed within the computer. Also known as hot swappable.

hot-relocation A Veritas Volume Manager technique of automatically restoring redundancy and access to mirrored and RAID-5 volumes when a disk fails. This is done by relocating the affected subdisks to disks designated as spares and/or free space in the same disk group.

inode list An inode is an on-disk data structure in the file system that defines everything about the file, except its name. Inodes contain information such as user and group ownership, access mode (permissions), access time, file size, file type, and the block map for the data contents of the file. Each inode is identified by a unique inode number in the file system where it resides. The inode number is used to find the inode in the inode list for the file system. The inode list is a series of inodes. There is one inode in the list for every file in the file system.

intent logging A logging scheme that records pending changes to a file system structure. These changes are recorded in an intent log.

interrupt key A way to end or break out of any operation and return to the system prompt by pressing Ctrl-C.

kernel asynchronous I/O A form of I/O that performs non-blocking system level reads and writes. This enables the system to handle multiple I/O requests simultaneously.

kilobyte A measure of memory or storage. A kilobyte is approximately a thousand bytes (technically, 2 to the 10th power, or 1,024 bytes). Also KB, Kbyte, kbyte, and K-byte.

large file A file more than two gigabytes in size. An operating system that uses a 32-bit signed integer to address file contents will not support large files; however, the Version 4 disk layout feature of VxFS supports file sizes of up to two terabytes.

large file system A file system more than two gigabytes in size. VxFS, in conjunction with VxVM, supports large file systems.

latency The amount of time it takes for a given piece of work to be completed. For file systems, this typically refers to the amount of time it takes a given file system operation to return to the user. Also commonly used to describe disk seek times.

load balancing The tuning of a computer system, network tuning, or disk subsystem in order to more evenly distribute the data and/or processing across available resources. For example, in clustering, load balancing might distribute the incoming transactions evenly to all servers, or it might redirect them to the next available server.

load sharing The division of a task among several components without any attempt to equalize each component's share of the load. When several components are load sharing, it is possible for some of the shared components to be operating at full capacity and limiting performance, while others components are under utilized.

Logical Unit Number A method of expanding the number of SCSI devices that can be placed on one SCSI bus. Logical Unit Numbers address up to seven devices at each SCSI ID on an 8-bit bus or up to 15 devices at each ID on a 16-bit bus.

logical volume See "volume."

LUN See "Logical Unit Number."

master node A computer which controls another computer or a peripheral.

megabyte A measure of memory or storage. A megabyte is approximately 1,000,000 bytes (technically, 2 to the 20th power, or 1,048,576 bytes). Also MB, Mbyte, mbyte, and K-byte.

metadata Data that describes other data. Data dictionaries and repositories are examples of metadata. The term may also refer to any file or database that holds information about another database's structure, attributes, processing, or changes.

mirror	A duplicate copy of a volume and the data therein (in the form of an ordered collection of subdisks). Each mirror is one copy of the volume with which the mirror is associated. The terms mirror and plex can be used synonymously.
mirroring	A layout technique that mirrors the contents of a volume onto multiple plexes. Each plex duplicates the data stored on the volume, but the plexes themselves may have different layouts.
mount point	The directory path name at which a file system attaches to the file system hierarchy.
multithreaded	Having multiple concurrent or pseudo-concurrent execution sequences. Used to describe processes in computer systems. Multithreaded processes are one means by which I/O request-intensive applications can use independent access to volumes and disk arrays to increase I/O performance.
NBU	See "Veritas NetBackup (NBU)."
node	One of the hosts in a cluster.
object (VxVM)	An entity that is defined to and recognized internally by the Veritas Volume Manager. The VxVM objects include volumes, plexes, subdisks, disks, and disk groups. There are two types of VxVM disk objects—one for the physical aspect of the disk and the other for the logical aspect of the disk.
OLTP	See "Online Transaction Processing."
online administration	An administrative feature that allows configuration changes without system or database down time.
Online Transaction Processing	A type of system designed to support transaction-oriented applications. OLTP systems are designed to respond immediately to user requests and each request is considered to be a single transaction. Requests can involve adding, retrieving, updating or removing data.
paging	The transfer of program segments (pages) into and out of memory. Although paging is the primary mechanism for virtual memory, excessive paging is not desirable.
parity	A calculated value that can be used to reconstruct data after a failure. While data is being written to a RAID-5 volume, parity is also calculated by performing an exclusive OR (XOR) procedure on data. The resulting parity is then written to the volume. If a portion of a RAID-5 volume fails, the data that was on that portion of the failed volume can be recreated from the remaining data and the parity.
partition	The logical areas into which a disk is divided.
persistence	Information or state that will survive a system reboot or crash.
petabyte	A measure of memory or storage. A petabyte is approximately 1,000 terabytes (technically, 2 to the 50th power).

plex A duplicate copy of a volume and its data (in the form of an ordered collection of subdisks). Each plex is one copy of a volume with which the plex is associated. The terms mirror and plex can be used synonymously.

preallocation Prespecifying space for a file so that disk blocks will physically be part of a file before they are needed. Enabling an application to preallocate space for a file guarantees that a specified amount of space will be available for that file, even if the file system is otherwise out of space.

Quick I/O Quick I/O presents a regular Veritas File System file to an application as a raw character device. This allows Quick I/O files to take advantage of asynchronous I/O and direct I/O to and from the disk device, as well as bypassing the UNIX single-writer lock behavior for most file system files.

Quick I/O file A regular UNIX file that is accessed using the Quick I/O naming extension (::cdev:vxfs:).

RAID A Redundant Array of Independent Disks (RAID) is a disk array set up with part of the combined storage capacity used for storing duplicate information about the data stored in that array. This makes it possible to regenerate the data if a disk failure occurs.

repository A repository holds the name, type, range of values, source, and authorization for access for each data element in a database. The database maintains a repository for administrative and reporting use.

Pertinent information, needed to display configuration information and interact with the database, is stored in the repository.

root disk The disk containing the root file system.

root disk group A special private disk group on the system. The root disk group is named rootdg. However, starting with the 4.1 release of Veritas Volume Manager, the root disk group is no longer needed.

root file system The initial file system mounted as part of the UNIX kernel startup sequence.

script A file, containing one or more commands that can be run to perform processing.

shared disk group A disk group in which the disks are shared by multiple hosts (also referred to as a cluster-shareable disk group).

sector A minimal unit of the disk partitioning. The size of a sector can vary between systems.

A sector is commonly 512 bytes.

segment Any partition, reserved area, partial component, or piece of a larger structure.

SGA See "System Global Area."

single threading The processing of one transaction to completion before starting the next.

slave node	A node that is not designated as a master node.
slice	The standard division of a logical disk device. The terms partition and slice can be used synonymously.
snapped file system	A file system whose exact image has been used to create a snapshot file system.
snapped volume	A volume whose exact image has been used to create a snapshot volume.
snapshot	A point-in-time image of a volume or file system that can be used as a backup.
snapshot file system	An exact copy of a mounted file system, at a specific point in time, that is used for online backup. A snapshot file system is not persistent and it will not survive a crash or reboot of the system.
snapshot volume	An exact copy of a volume, at a specific point in time. The snapshot is created based on disk mirroring and is used for online backup purposes.
spanning	A layout technique that permits a volume (and its file system or database) too large to fit on a single disk to distribute its data across multiple disks or volumes.
Storage Checkpoint	An efficient snapshot technology for creating a point-in-time image of a currently mounted VxFS file system. A Storage Checkpoint presents a consistent, point-in-time view of the file system by identifying and maintaining modified file system blocks.
storage class	Set of volumes with the same volume tag.
Storage Rollback	On-disk restore capability for faster recovery from logical errors, such as accidentally deleting a file. Because each Storage Checkpoint is a point-in-time image of a file system, Storage Rollback simply restores or rolls back a file or entire file system to a Storage Checkpoint.
stripe	A set of stripe units that occupy the same positions across a series of columns in a multi-disk layout.
stripe unit	Equally sized areas that are allocated alternately on the subdisks (within columns) of each striped plex. In an array, this is a set of logically contiguous blocks that exist on each disk before allocations are made from the next disk in the array.
stripe unit size	The size of each stripe unit. The default stripe unit size for VxVM is 32 sectors (16K). For RAID 0 stripping, the stripe unit size is 128 sectors (64K). For VxVM RAID 5, the stripe unit size is 32 sectors (16K). A stripe unit size has also historically been referred to as a stripe width.
striping	A layout technique that spreads data across several physical disks using stripes. The data is allocated alternately to the stripes within the subdisks of each plex.
subdisk	A consecutive set of contiguous disk blocks that form a logical disk segment. Subdisks can be associated with plexes to form volumes.

superuser	A user with unlimited access privileges who can perform any and all operations on a computer. In UNIX, this user may also be referred to as the "root" user. On Windows/NT, it is the "Administrator."
tablespace	A tablespace is a storage structure (containing tables, indexes, large objects, and long data) that allows you to assign the location of database and table data directly onto containers. Tablespaces reside in database partition groups.
terabyte	A measure of memory or storage. A terabyte is approximately 1,000,000,000,000 bytes (technically, 2 to the 40th power, or 1,000 GB). Also TB.
throughput	A measure of work accomplished in a given amount of time. For file systems, this typically refers to the number of I/O operations in a given period of time.
UFS	The Solaris name for a file system type derived from the 4.2 Berkeley Fast File System.
unbuffered I/O	I/O that bypasses the file system cache for the purpose of increasing I/O performance (also known as direct I/O).
Veritas Enterprise Administrator	Application that is required to access graphical user interface (GUI) functionality.
Veritas NetBackup (NBU)	A product that lets you back up, archive, and restore files, directories, or raw partitions that reside on your client system.
Veritas Volume Replicator (VVR)	A feature of Veritas Volume Manager, VVR is a data replication tool designed to contribute to an effective disaster recovery plan.
Version Checkpoint	In a DB2 UDB EEE or ESE Data Partitioning Feature (DPF) environment, a Version Checkpoint is a set of checkpoints that spans multiple partitions. A Version Checkpoint contains one Storage Checkpoint per partition. A Storage Checkpoint is an efficient snapshot technology for creating a point-in-time image of a currently mounted VxFS file system. A Storage Checkpoint presents a consistent, point-in-time view of the file system by identifying and maintaining modified file system blocks.
volume	A logical disk device that appears to applications, databases, and file systems as a physical disk partition. A logical disk can encompass multiple or one to many physical volumes.
volume layout	A variety of layouts that allows you to configure your database to meet performance and availability requirements. This includes spanning, striping (RAID-0), mirroring (RAID-1), mirrored stripe volumes (RAID-0+1), striped mirror volumes (RAID-1+0), and RAID 5.
volume manager objects	Volumes and their virtual components. See "object (VxVM)."
VVR	See "Veritas Volume Replicator (VVR)."
vxfs or VxFS	The acronym for Veritas File System.

vxvm or VxVM The acronym for Veritas Volume Manager.

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X