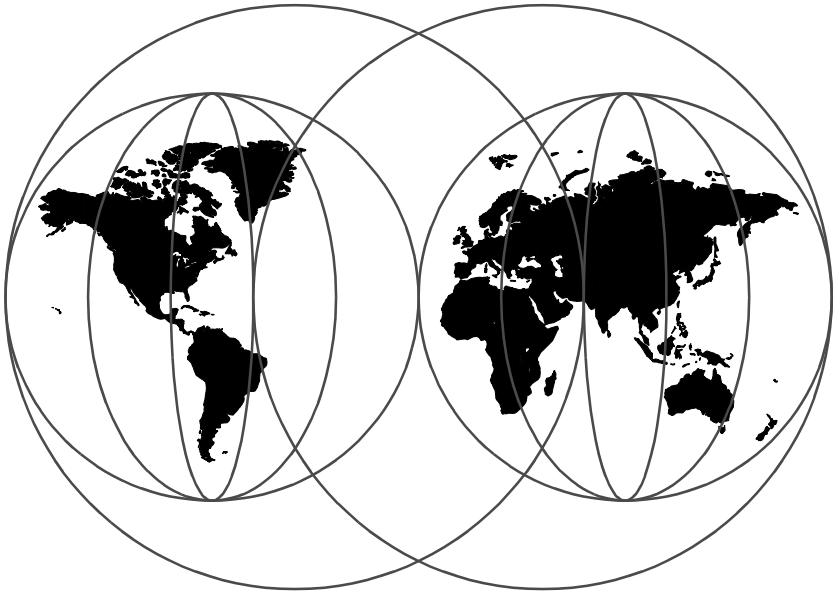
Managing VLDB Using DB2 UDB EEE

Jonathan Cook, Eduardo Fontana, George Latimer



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-5105-00

SG24-5105-00

International Technical Support Organization

Managing VLDB Using DB2 UDB EEE

June 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special Notices” on page 265.

First Edition (June 1998)

This edition applies to DB2 Universal Database Version 5.0 Enterprise - Extended Edition for use with the AIX V4 Operating System.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

This redbook is intended to help database administrators manage very large databases using DB2 Universal Database Enterprise-Extended Edition (EEE) on RS/6000 SP hardware. Specifically, it discusses how to plan the allocation of disk space and then load data into a large database.

A detailed explanation of how to use ADSTAR Storage Manager (ADSM) to backup and restore DB2 UDB EEE databases is included. A section is also devoted to archiving log files using ADSM.

Configuring your database system to be highly available using HACMP is also covered. The complexities inherent in using multiple database partitions on each SP node are explained.

All the programs and scripts that were used to create a large database and configure the system for use with ADSM and HACMP are included in this book.

Some knowledge of DB2 Universal Database or DB2 Parallel Edition is assumed.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

This project was designed and managed by:

Jonathan Cook is an Advisory Software Engineer at the International Technical Support Organization, Austin Center. He writes extensively and teaches IBM classes worldwide on DB2 for the UNIX and Intel platforms.

The authors of this document are:

Jonathan Cook
International Technical Support Organization, Austin Center.

Eduardo Fontana is a Data Management Pre-Sales Technical Specialist who works for the Software Group at IBM Argentina. He has extensive experience in AIX, HACMP, RS/6000 SP, DB2, and Intel Operating Systems.

George Latimer is a Senior Availability Services Specialist who works for IBM Global Services based in Charlottesville, Virginia. His areas of speciality are ADSM, RS/6000 SP, DB2, and HACMP.

Thanks to the following people for their invaluable contributions to this project:

Calene Janacek
International Technical Support Organization, Austin Center

Dwaine Snow
IBM Toronto Lab

Dale McInnis
IBM Toronto Lab

Bob Harbus
IBM Toronto Lab

A special thanks goes to the staff at the IBM Teraplex Center in Poughkeepsie, NY, without whose help this project would never have been completed:

Joe Labriola, Joe Catucci, Gus Branish, Kurt Sulzbach, and Paul Bildzok

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 281 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com`
  For IBM Intranet users `http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Chapter 1. Overview

The approach of this redbook is to address some of the issues in managing very large databases (VLDBs) using DB2 Universal Database Enterprise-Extended Edition (DB2 UDB EEE). Specifically, we discuss how to:

- Allocate disk space for the data
- Load the data into a database
- Backup and restore databases using ADSM
- Make the database system highly available using HACMP

A step-by-step guide to each of the above topics is provided in the following chapters. We were fortunate when writing this book to have the use of an RS/6000 SP system configured with four high nodes (eight CPUs per node), a high-speed switSch and around 300 GB of disk. This allowed us to run tests with a relatively large database (30 GB) and a configuration of DB2 UDB EEE with four database partitions (logical nodes) per SP node. This system was made available to us by the IBM TerSaplex Center in Poughkeepsie, New York.

1.1 DB2 Universal Database Enterprise - Extended Edition

DB2 Universal Database Enterprise-Extended Edition has features that help to satisfy the needs of decision support and data warehousing applications. These features include:

- Intelligent data distribution. DB2 UDB EEE supports parallel queries by using intelligent database partitioning. In an MPP (Massively Parallel Processor) or cluster configuration, DB2 UDB EEE distributes the data and database functions to multiple hosts. DB2 UDB EEE uses a hashing algorithm that enables it to manage the distribution (and redistribution) of data as required.
- A cost-based SQL optimizer. DB2's cost-based SQL optimizer makes use of information about how the data is distributed to evaluate the different access paths for an SQL query. It will then use the access path with the estimated shortest elapsed execution time. The optimizer supports SQL query rewrite, OnLine Analytical Processing (OLAP), SQL extensions, Dynamic Bit-Mapped Indexing ANDing (DBIA), and Star Joins. These features are commonly used in data warehousing applications.

- Parallel everything. DB2 UDB EEE's capabilities include taking advantage of cluster hardware configurations, partitioning of data, accessing plans automatically created for parallel execution with standard SQL, and parallel execution of utilities. DB2's parallel execution applies to SELECT, INSERT, UPDATE, and DELETE functions. Data scans, joins, sorts, load balancing, table reorganization, data load, index creation, indexed access, and backup and restore are all performed simultaneously on all hosts in the DB2 cluster.
- Scalability. As your business grows, DB2 can accommodate more users and more data with predictably scalable performance. It uses a concept called *shared nothing*. (This is discussed in more detail in "Parallel Architecture and Processing" on page 5). The Shared nothing architecture allows parallel queries to be processed with the minimum of contention for resources between the hosts in the DB2 cluster. Because the number of data partitions has little impact on traffic between hosts, performance scales in an almost linear manner as you add more machines to your DB2 cluster.

DB2 UDB EEE provides the ability for a database to be partitioned across multiple independent computers using the same operating system. To the end-user or application developer, the database appears to be a single logical entity. DB2 UDB Enterprise-Extended Edition is designed for applications where the database is simply too large for a single computer to handle efficiently. SQL operations can operate in parallel on the individual database partitions, thus improving the performance of a single query.

1.2 Terminology Used in DB2 UDB EEE

This section focuses on some of the terms and definitions we'll use for the remainder of the book.

A database is simply a collection of data. If you're familiar with DB2 or another RBDMS, you'll know that data is stored in tables in a database with tables having a certain number of columns and rows. In DB2 EEE, the database is distributed into many database partitions. A database partition is a part of the database that has its own portion of the user data, indexes, configuration files, and transaction logs. Let's look at a simple environment where there is one database partition for every machine in your cluster.

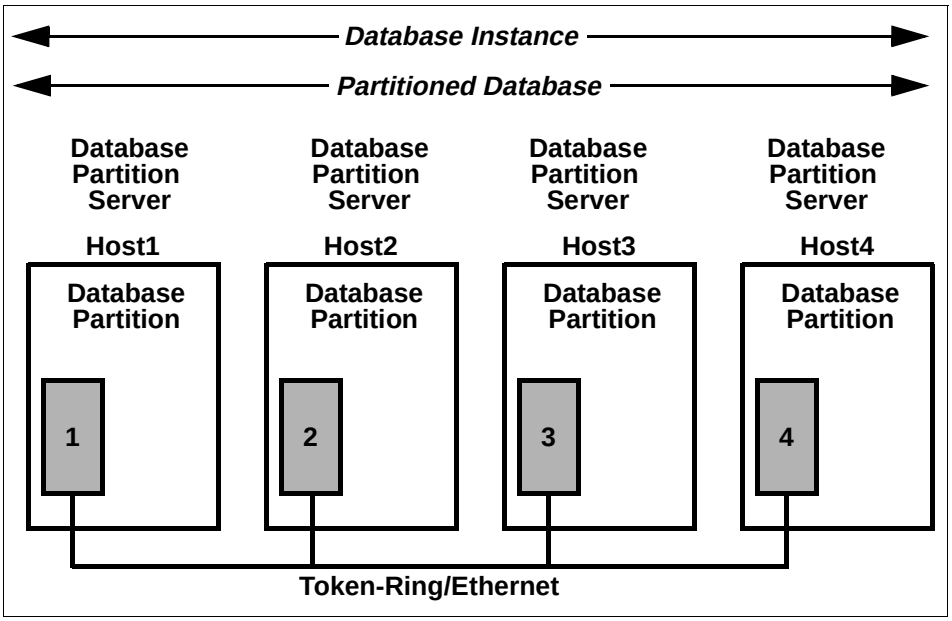


Figure 1. Database Partitions

Figure 1 is an example of a simple four host cluster. A configuration file determines which machines will be part of the DB2 UDB EEE environment. A database partition server holds a portion (partition) of the database. The hosts communicate using TCP/IP through a token-ring or Ethernet connection with DB2.

A database instance is a logical database manager environment. You can have several instances of the database manager product executing within the same group of hosts. You can use these instances to separate a development environment from a production environment, to tune the database manager to a particular environment, or to protect sensitive information from a particular group of people.

A database is created within a database instance. The database may either be partitioned or non-partitioned. It will be probably be a partitioned database when using DB2 UDB EEE because you have the ability to distribute the function and data among all the hosts in your DB2 cluster.

It is possible to have multiple database partitions on a single host.

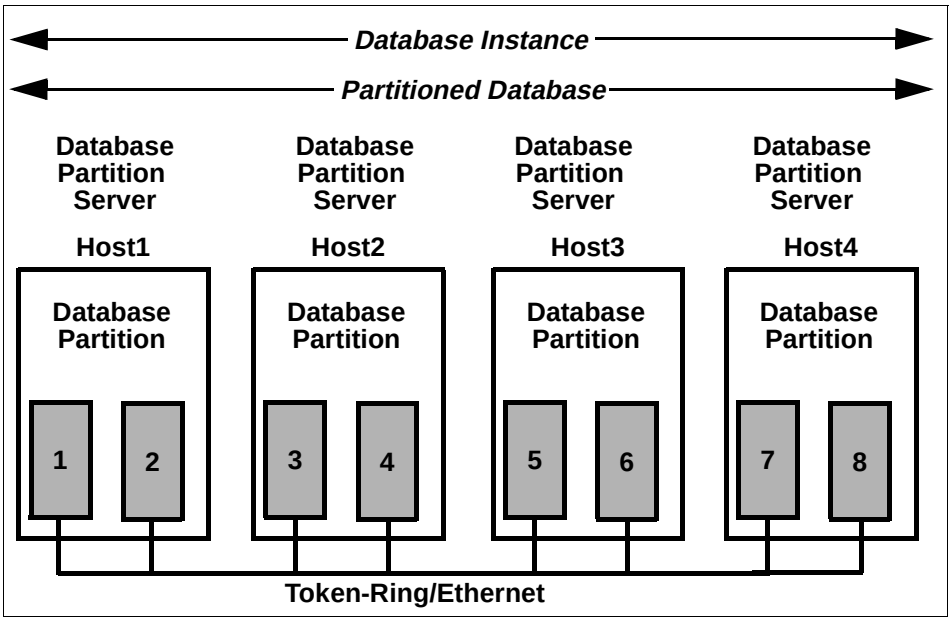


Figure 2. Multiple Database Partitions per Host

Figure 2 uses the same four-host DB2 cluster used in the last example (Figure 1 on page 3). Notice that each host has two database partitions. Quite often, this type of configuration is used to take advantage of SMP (Symmetric Multi-Processor) hardware. You can distribute the database function and processing among different hosts and among processors on the same host.

It is desirable to have the same number of database partitions on each host, though there may be exceptions to this statement. For example, if your original cluster consisted of single CPU machines and you added new hardware to your DB2 cluster that included SMP machines, you might consider creating multiple database partitions on the new SMP hardware.

You might also want to keep one database partition reserved to store only the System Catalogs and not contain any user data. The database partition that holds the System Catalog tables is referred to as the *catalog partition*. You may also come into contact with the term *catalog node*. The term *node* was used in DB2 Parallel Edition and caused some confusion because it referred to both software and hardware. A node in DB2 terminology is a database partition. However, we also refer to uniprocessor or symmetric multiprocessor machines as nodes. This book tries to avoid confusion in terminology. However, some of the DB2 commands and SQL statements use the old terminology.

1.2.1 Parallel Architecture and Processing

This section explains some more terminology that is used in a partitioned database environment. We show how DB2 UDB EEE uses this terminology.

DB2 UDB EEE is a product that takes advantage of the *shared nothing architecture*.

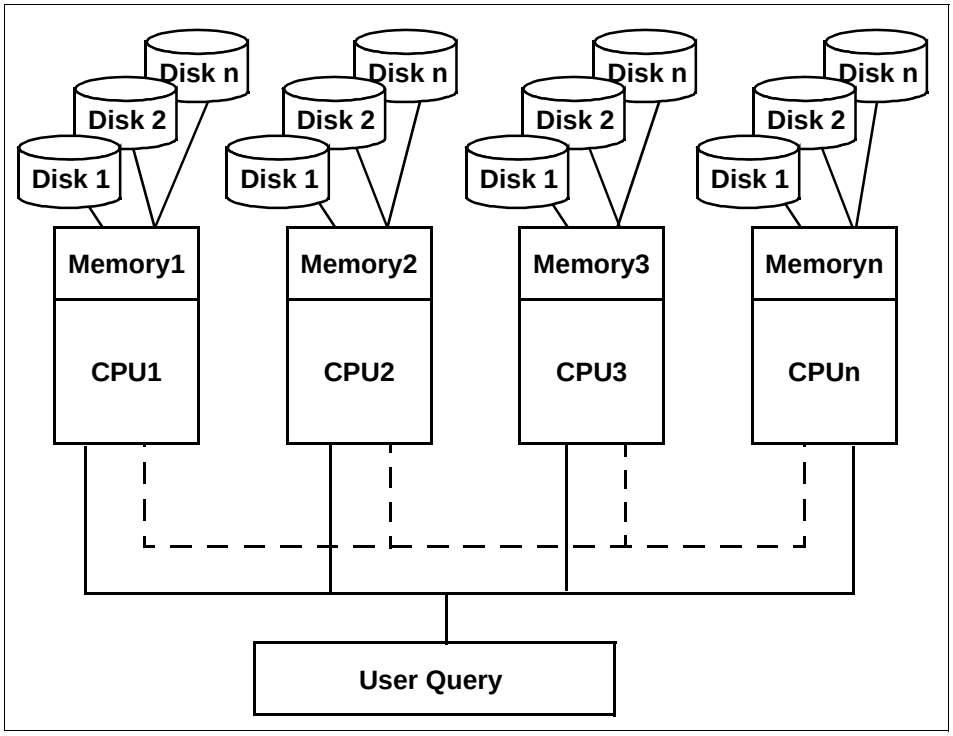


Figure 3. Shared Nothing Architecture

Figure 3 shows an example of the shared nothing architecture. Here we see loosely coupled processors are linked by some high-speed interconnection. Each processor has its own memory and accesses its own disks. The advantages of this type of architecture are the following:

- Scalability in terms of database size and number of processors
- Performance gains from not sharing resources across a network

Total memory has a fixed capacity. By increasing the number of machines, you can exceed that fixed amount because the memory is shared among machines. The same is true for total disk capacity. The other advantage that could be gained is in the number of operations that are performed. Each

machine only needs to do part of the work. So, processing is more distributed, and the database can manage a larger amount of data.

Performance gains are assisted by the concept of *function shipping*. Function shipping assists in the reduction of network traffic because functions, such as SQL queries are shipped instead of data. Function shipping means that relational operators are executed on the processor containing the data whenever possible. So, the operation (or the SQL) is moved to where the data resides. Function shipping is well suited to the shared nothing architecture model.

The high-speed interconnection used between hosts is represented in Figure 3 on page 5 by the dotted lines. In an SP cluster, this would be the high-speed switch. The smoother lines represent another type of network. The user query issues an SQL statement. For example, an SQL SELECT statement is issued. Every database partition receives the operation from one processor that works as a dispatcher. This dispatcher is sometimes referred to as the coordinator partition. Each database partition executes the operation on its own set of data. An exchange of information among the hosts may occur. The result from the operation is sent back to the coordinator partition. The coordinator partition assembles the data and returns it to the requestor.

Chapter 2. Building a Large DB2 UDB EEE Database

There are many issues to consider when building a large database. The approach of this section is to take you through an example of physically designing, implementing and loading a 30 GB DB2 UDB EEE database on a RS/6000 SP with four high nodes. Every step that was taken in real life is documented here, with comments where appropriate. Rather than explaining all the possible methods to perform each task, we merely explain the method we chose. Our choice of methods was based on either simplicity or performance. The tasks we cover are:

- “Designing and Implementing the DIsk Space Allocation” on page 8
- “Installing and Configuring DB2 UDB EEE” on page 17
- “Creating the Database, Nodegroups, Table Spaces and Tables” on page 27
- “Loading Data into the Database” on page 35

We also perform the necessary tasks to support an HACMP configuration. This subject is covered fully in “DB2 UDB EEE High Availability using HACMP” on page 129.

2.1 The Hardware Configuration

The hardware configuration used for this example is shown in Figure 4. It shows a RS/6000 SP with four high nodes.

Each high node has:

- 8 CPUs (604)
- 4 GB of internal SCSI disk

There is 288 GB of external SSA disk. These disks are physically connected to all four SP nodes using SSA loops.

The SP nodes have these network interfaces:

- Token-ring using tp3an01, tp3an05, tp3an09, tp3an13
- Ethernet using tp3an01b, tp3an05b, tp3an09b, tp3an13b
- Switch using tp3sn01, tp3sn05, tp3sn09, tp3sn13

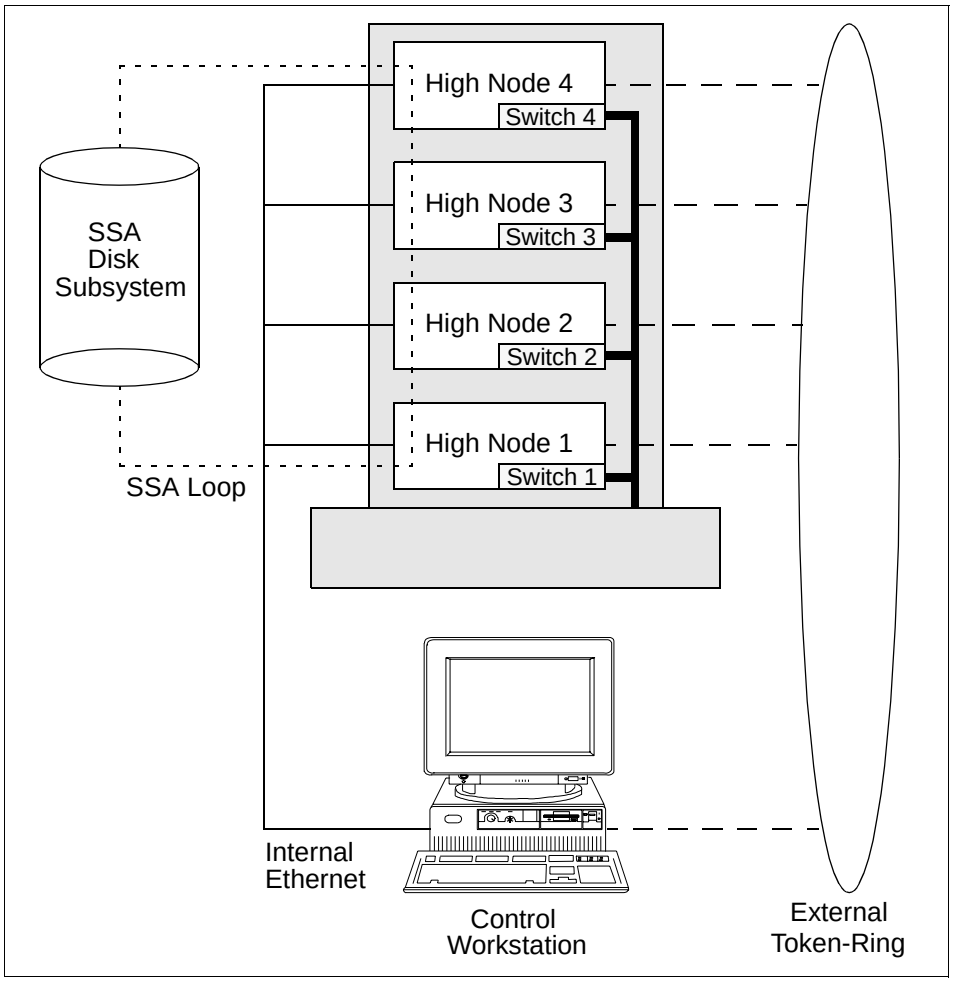


Figure 4. Hardware Configuration

2.2 Designing and Implementing the DIsk Space Allocation

The available disk was allocated to hold a 30 GB TPCD database. This database was to loaded with data using the TPCD generator program *dbgen* with a size factor of 30, resulting in the following tables:

- lineitem with 180,000,000 rows (69 percent)
- orders with 45,000,000 rows (16 percent)
- partsupp with 24,000,000 rows (11 percent)

- customer with 4,000,000 rows (2 percent)
- part with 6,000,000 rows (2 percent)
- supplier with 300,000 rows (<1 percent)
- nation with 25 rows and region with 5 rows

Indexes are also created against these tables.

2.2.1 Instance and Database Partition Servers

We decided to configure 16 database partition servers across the four SP nodes, making four servers per SP node. This is based on the sizing rule of one database partition server per two CPUs.

- Database partition 1 holds only the System Catalogs and very small tables. This is done for the following reasons:
  - For backup processing, the partition holding the System Catalogs must finish before the other partitions can start. So we store as little data as possible on this partition.
  - This frees up some CPU power on this host. This partition can thus be used as the coordinator partition, and also as an ADSM server.

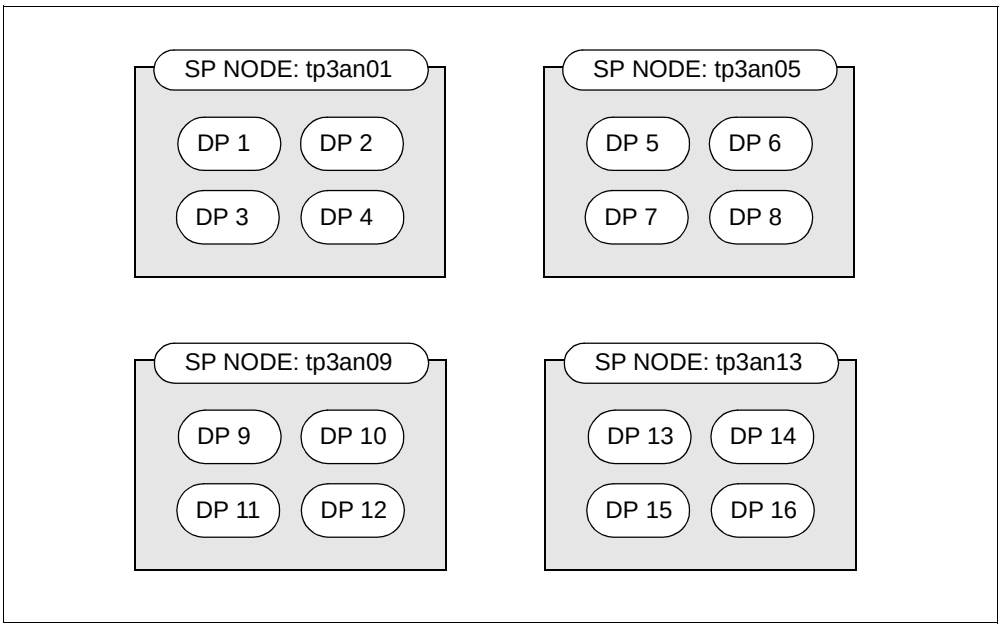


Figure 5. SP Nodes and DP Servers

In Figure 5, the relationship between SP nodes and DB2 database partition (DP) servers is shown.

2.2.2 Database

The database is created with the name TPCD30.

2.2.3 Nodegroups

Two user-defined nodegroups were defined:

- NG_BIG for the large and medium tables (lineitem to supplier). This nodegroup is defined on partitions (2-16), leaving partition 1 for the System Catalogs and very small tables.
- NG_LIT for the very small tables (nation and region). This nodegroup is defined on partition 1.

2.2.4 Table Spaces for Data and Index

Five Database Managed Space (DMS) table spaces for data and indexes were defined in total. Four of these were defined in nodegroup NG_BIG:

- TS_DAT_BIG for the lineitem table data
- TS_IND_BIG for the lineitem table indexes
- TS_DAT_MED for the medium tables (orders to supplier) data
- TS_IND_MED for the medium tables indexes

The lineitem and medium tables were separated into different table spaces because:

- We can backup and restore at the table space level. So lineitem could be backed up independently of the medium tables.
- When loading data, a load of lineitem can be run simultaneously with a load of one of the medium tables, for example, orders.

One DMS table space was defined in NG_LIT:

- TS_LIT for the very small tables (region and nation) data and indexes

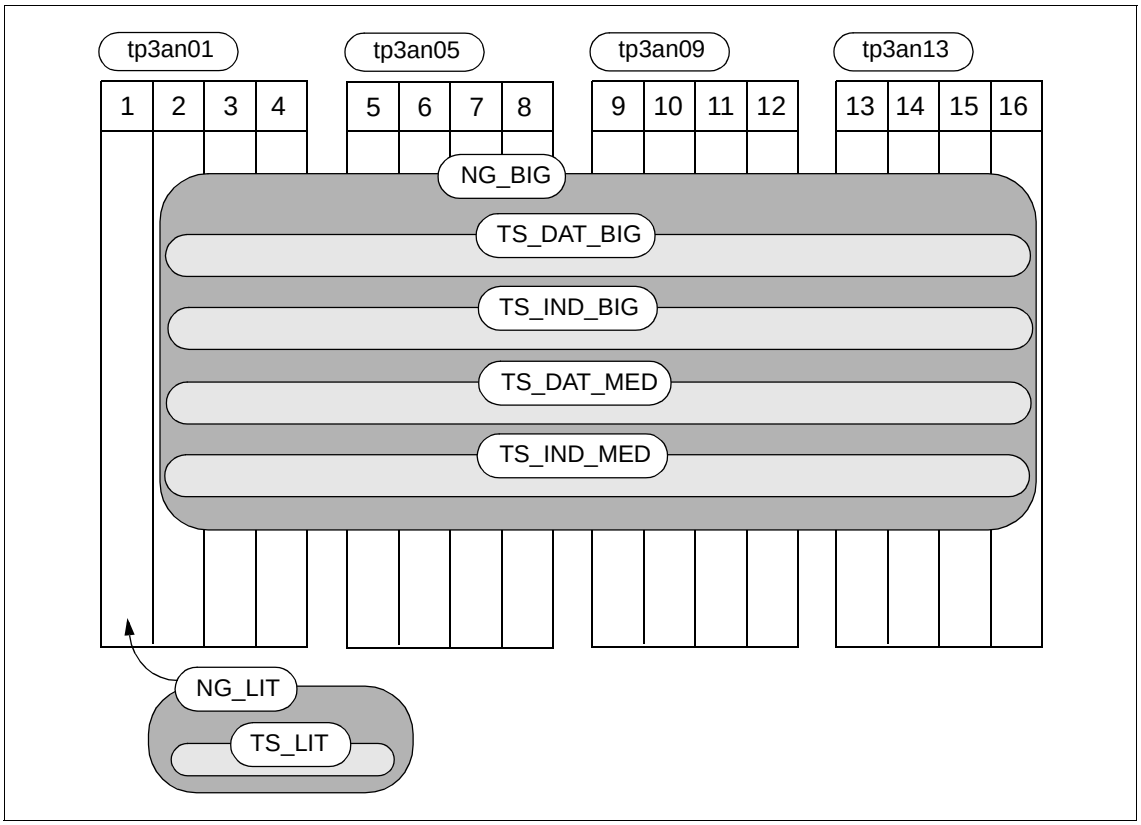


Figure 6. Nodegroups and Table Spaces for Data and Index

In Figure 6, we can see the relationship between the nodegroups and the table space that will hold the data and indexes.

2.2.5 Temporary Table Space

A temporary table space was created (TS_TMP) as System Managed Space (SMS) which uses containers which are separate file systems. This was done for these reasons:

- Performance: isolate log space and temp space: The logs and the temporary space are on separate file systems and so use different disks.
- Granularity of disk allocation: We can allocate the disk space for the temporary table separately from the disk space for the logs.

2.2.6 Container Definitions

On each of the SP nodes these Logical Volumes (LVs) or file systems (FSs) were defined as containers for the table spaces (SMS or DMS):

- Two raw LVs for the DMS table space TS_DAT_BIG
- Two raw LVs for the DMS tables space TS_IND_BIG
- Two raw LVs for the DMS table space TS_DAT_MED
- Two raw LVs for the DMS table space TS_IND_MED
- Two FSs for the SMS table space TS_TMP

In addition, on the first SP node alone, tp3an01:

- One raw LV for the table space which holds the System Catalogs
- One raw LV for the DMS table space TS_LIT

2.2.7 File Systems for Log Files

On each SP node, a separate file system is defined to hold the log files. The logical volume used by this file system is mirrored because any loss of log file data will cause the database to be non-recoverable.

2.2.8 Mapping LVs and FSs to Physical Disks

These LVs were mapped to disk in the following way:

- We have 16 physical disks (external SSA) available on each SP node.
- Since we have four database partitions per SP node, there are four physical disks available per partition.
- We place the index and data parts of each table on different disks. This is to reduce disk head movement when using an index to access the data portion of a table.
- For the large and medium tables, we define two disks each for data and index.
  - The two disks used for the data of the large table are also used for the indexes of the medium tables.
  - The two disks used for the index of the large table are also used for the data of the medium tables.
- Each container used for data or index is mapped to one physical disk so that DB2 can do intelligent striping.
- The two FSs for TS_TMP are placed on two disks each per partition.

Mirroring Disks

It is normally recommended to mirror all of the disks used for data and index because a loss of one of these disks would usually render the database unusable. However, this was not done in our configuration. This means that if a disk fails, after replacing the disk, the database would have to be restored from backup and rolled forward. This could easily take a considerable time during which the database is not available. So if long recovery times are unacceptable, you MUST mirror all data and index disk.

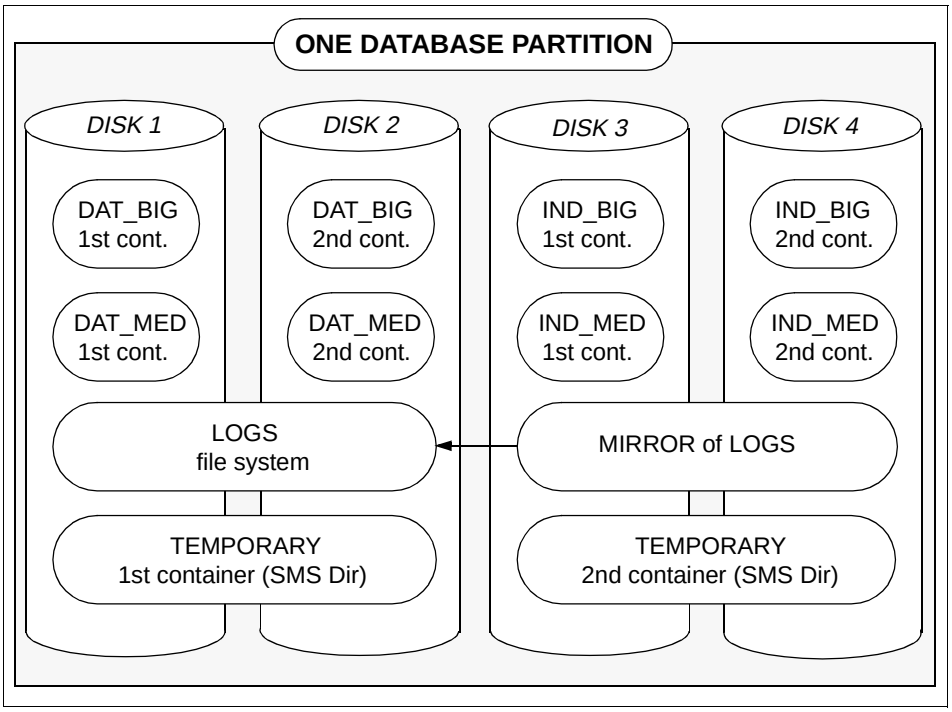


Figure 7. Containers, Logical Volume and Disks

Figure 7 shows the allocation of disk to containers and logs. Note that this scheme is repeated across all 15 partitions (2-16) which the NG_BIG nodegroup includes. On partition 1, the logs file system and SMS directories for the temporary table space are defined, but not the BIG and MED containers. Instead, there is one container for the data and indexes of the very small tables defined in TS_LIT.

2.2.9 Sizing of LVs and FSs

The sizing of the LVs and FSs was based on:

- TS_DAT_BIG, the table space for lineitem's data. We know the lineitem table will take around 70 percent of 30 GB for all its data across all the partitions. This equals 21 GB spread over the 15 partitions, or 1.4 GB per partition. To allow for a margin of growth, we allocated 2 LVs each of 150 physical partitions (PPs) each of 8 MB = 2.4 GB of disk.
- TS_IND_BIG, the table space for lineitem's indexes. By testing with smaller TPCD databases, we found that the indexes need up to the same amount of disk as we defined for the data. So we allocated 2.4 GB disk for the indexes per partition.
- TS_DAT_MED. The data of the medium tables takes 30 percent of the total data. This makes less than half the space allocated to lineitem. So we allocated 2 LVs each of 75 PPs of 8 MB = 1.2 GB of disk.
- TS_IND_MED. For the indexes of the medium tables, we allocated the same amount of disk as for their data–that is, 1.2 GB of disk.
- Logs: For the FS holding the log files, we allocated one FS of 1.2 GB spread over two disks. This disk was mirrored on the other two disks available for the partition.
- Temporary Space: For the 2 FSs holding TS_TMP, we allocated a total of 4.8 GB per partition. This comprised of two FSs of 2.4 GB, each using two disks. The size of the temporary table space was designed to be big enough to be able to hold another copy of the largest table in case of a REORG.

2.2.10 Naming of VGs, LVs and FSs

To be compatible with an HACMP configuration, volume groups, logical volumes and file systems must be named so that no conflict arises after HACMP takeover.

We used the following naming rules:

- Volume groups: *vg_<SP node>_<unique id>*.

  For example, vg_n01_01 is the first volume group on SP node tp3an01.

- Logical volumes: *lv_<SP node>_<VG id>_<LV id>*.

  For example, lv_n05_01_103 is a logical volume on SP node tp3an05, defined in the vg_n05_01 volume group.

- File systems:

- The file systems used for TS_TMP containers are named: */DB_TMP/db2inst1/NODE00<DP id>/T1* and */DB_TMP/db2inst1/NODE00<DP id>/T2*, where <DP id> is the database partition identifier.

  For example, /DB_TMP/db2inst1/NODE0013/T1 is one of the two FSs used as containers for TS_TMP for database partition 13.

- The file systems used to store the log files are named: */DB_LOG/db2inst1/NODE00<DP id>*, where *<DP id>* is the database partition identifier.

  For example, /DB_LOG/db2inst1/NODE0009 is the FS used to store the log files for database partition 9.

This table summarizes the logical volumes that are created per database partition:

LV Name	Used For	LV Type	TS Type	Disks	Partitions
`lv_n01_01_101`	Catalog data	raw	DMS	1	1
`lv_n01_01_102`	Small data	raw	DMS	1	1-16
`lv_nyy_0z_x03`	Logfiles	JFS	SMS	2(Mirror)	1-16
`lv_nyy_0z_x04`	Temp1	JFS	SMS	2	1-16
`lv_nyy_0z_x05`	Temp 2	JFS	SMS	2	1-16
`lv_nyy_0z_x06`	Large data1	raw	DMS	1	2-16
`lv_nyy_0z_x07`	Large data 2	raw	DMS	1	2-16
`lv_nyy_0z_x08`	Large index 1	raw	DMS	1	2-16
`lv_nyy_0z_x09`	Large index 2	raw	DMS	1	2-16
`lv_nyy_0z_x10`	Medium data 1	raw	DMS	1	2-16
`lv_nyy_0z_x11`	Medium data 2	raw	DMS	1	2-16
`lv_nyy_0z_x12`	Medium index 1	raw	DMS	1	2-16
`lv_nyy_0z_x13`	Medium index 2	raw	DMS	1	2-16

Table 1. Logical Volumes per Database Partition

The name of the logical volume is based on:

- Its usage (log, data, index, temp and so on)
- The database partition that uses the logical volume
- The SP node where the logical volume is defined

For example, for the LV used for the log files, the name is lv_nyy_0z_x03, where:

- x=1 for partitions 5, 9, and 13
- x=2 for partitions 2, 6, 10, and 14

- x=3 for partitions 3, 7, 11, and 15
- x=4 for partitions 4, 8, 12, and 16

and:

- yy=2-digit node number (01, 05, 09, 13)

and:

- z=1 for partitions 1, 2, 5, 6, 9, 10, 13, 14
- z=2 for partitions 3, 4, 7, 8, 11, 12, 15, 16

2.2.11 Volume Groups per SP node

Two volume groups were created per SP node. For example, for tp3an01, these VGs are: vg_n01_01 and vg_n01_02.

2.2.12 File Systems per Database Partition

If we look at database partition 1, the file systems used by the database are:

```
Filesystem       1024-blocks   Free     Mounted on

/dev/lv_n01_01_103  2031616   2015520   /DB_TMP/db2inst1/NODE0001/T1
/dev/lv_n01_01_104  2031616   2015520   /DB_TMP/db2inst1/NODE0001/T2
/dev/lv_n01_01_105  1228800   1206432   /DB_LOG/db2inst1/NODE0001
```

These file systems hold the temporary data and log data for partition 1 of the database. There are 3 similar file systems for each partition.

2.2.13 Logical Volumes per Database Partition

If we consider the logical volumes used as containers for data and index of the large and medium tables, database partition 2 has the following LVs assigned:

```
LV NAME             TYPE       LPs   PPs   PVs  LV STATE      MOUNT POINT

lv_n01_01_206       udb        150   150   1    closed/syncd  N/A
lv_n01_01_207       udb        150   150   1    closed/syncd  N/A
lv_n01_01_208       udb        150   150   1    closed/syncd  N/A
lv_n01_01_209       udb        150   150   1    closed/syncd  N/A
lv_n01_01_210       udb        75    75    1    closed/syncd  N/A
lv_n01_01_211       udb        75    75    1    closed/syncd  N/A
lv_n01_01_212       udb        75    75    1    closed/syncd  N/A
lv_n01_01_213       udb        75    75    1    closed/syncd  N/A
```

These logical volumes hold the data and indexes for the large and medium tables on database partition 2. Note that these tables are not defined on partition 1. The names of the logical volumes are decided according to the rules defined in Table 1 on page 15.

2.2.14 File System for the DB2 Instance Home Directory

A file system was also created at the first SP node, tp3an01, to hold the instance owner's home directory:

```
Filesystem              1024-blocks      Mounted on
/dev/lv_n01_01_114      819200           /home/tp3an01
```

2.2.15 How to Create the VGs, LVs and FSs

The volume groups, logical volumes and file systems were all defined using a shell script, setup_VG.ksh, as shown in "Allocate Disks and Logical Volumes" on page 218. This shell script also provides the ability to destroy all the definitions (VGs, LVs, FSs) if required, which is useful if we need to change the configuration in any way.

2.3 Installing and Configuring DB2 UDB EEE

To install and configure DB2 UDB EEE, we performed the following tasks:

- Make the file system for the DB2 instance's home directory (home/tp3an01) available across NFS to all the SP nodes.
- Install DB2 UDB EEE.
- Add a group and a user for the DB2 instance.
- Add entries to /etc/services on all SP nodes.
- Create the DB2 instance.
- Edit .profile.
- Edit db2nodes.cfg.
- Edit .rhosts.
- Setup db2diag.log and syslog.
- Start the DB2 instance.

2.3.1 Making /home/tp3an01 Available to all the SP Nodes

The /home/tp3an01 file system, which will hold the DB2 instance owner's home directory, was NFS-exported using SMIT:

```
smitty nfs
-->Network File System (NFS)
---->Add a Directory to Exports List
```

These fields were specified:

```
PATHNAME of directory to export ................/home/tp3an01
HOSTS allowed root access .....................tp3an05,tp3an09,tp3an13
```

Note: It is important to list the hostnames of the other three SP nodes in `HOSTS allowed root access` during the definition of `Add a Directory to Exports List`.

On the other three SP nodes (tp3an05, tp3an09, tp3an13), this directory was NFS-mounted. For example, on tp3an05:

```
smitty nfs
-->Network File System (NFS)
---->Add a File System for Mounting
```

These fields were specified:

```
PATHNAME of mount point............................/home/tp3an01
PATHNAME of Remote Directory........................./home/tp3an01
HOST where remote directory resides.................tp3an01
Remount file system now,
update /etc/filesystems or both?....................both
/etc/filesystems entry will mount the directory
on system RESTART..................................yes
Mount file system soft or hard.....................soft
```

NFS Mount /home/tp3an01 over the Switch

During testing, we noticed that any trap files (*.trp) produced by DB2 are written in the instance owner's home directory on tp3an01 (even if the trap occured one of the other SP nodes). For this reason, we found it necessary to NFS mount /home/tp3an01 over the switch.

2.3.2 Installing DB2 UDB EEE

To install DB2 UDB EEE, the method we used was to:

- Log in as root on the first SP node (tp3an01).
- Change directory into the directory where the images are located.

- Run `smitty installp`, then specify . (dot) as the input directory.
- Select **F4** against Software to install.
- Choose **License for DB2 UDB EEE** (this will pull in all the other LPPs as prerequisites).

Once that has finished, edit /smit.log to find the `installp` command that was run:

```
installp -acgNQqwX -d. 'db2_05_00.xsrv 5.0.0.1'
```

For the `installp` command to work on the other SP nodes, the DB2 images must be in a directory which is available through NFS to the other SP nodes. In our case, we placed the images under the file system that will be used for the home directory of the instance owner (db2inst1), which is /home/tp3an01.

Then we installed the other nodes using this command:

```
installp -acgNQqwX -d/home/tp3an01 'db2_05_00.xsrv 5.0.0.1'
```

from each of the three other SP nodes (tp3an05, tp3an09, tp3an13) while logged in as root at those SP nodes.

To see what LPPs were installed, look at /smit.log on each SP node. Here is the output from the first node:

```
Installation Summary
--------------------
Name                        Level           Part        Event       Result
-------------------------------------------------------------------------------
ifor_ls.client.base         4.2.0.0         USR         APPLY       SUCCESS
ifor_ls.client.base         4.2.0.0         ROOT        APPLY       SUCCESS
db2_05_00.xsrv              5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.pext              5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.db2.samples       5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.cs.sna            5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.cs.ipx            5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.cs.drda           5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.client            5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.db2.rte           5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.db2.engn          5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.das               5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.cs.rte            5.0.0.1         USR         APPLY       SUCCESS
db2_05_00.cnvucs            5.0.0.1         USR         APPLY       SUCCESS
```

Note that the DB2 EEE components installed in this example take around 60 MB in /usr/lpp/db2_05_00.

To install the documentation in HTML format, we chose to install to one SP node only to avoid wasting disk space. This is the output from running the install:

```
Installation Summary
--------------------
Name                        Level           Part        Event       Result
-------------------------------------------------------------------------------
db2_05_00.html.en_US        5.0.0.1         USR         APPLY       SUCCESS
```

This takes an additional 22 MB. The documentation is installed in compressed format. To decompress these files, we ran:

```
cd /usr/lpp/db2_05_00/doc
./db2insthtml en_US
```

After completion, /usr/lpp/db2_05_00 takes up around 130 MB.

Space Required to Install the Online Manuals

For the db2insthtml script to complete, you need enough space in /usr/lpp/db2_05_00to hold the compressed (22 MB) and decompressed (70 MB) versions.

2.3.3 Adding a Group and User

Since the RS/6000 SP is configured to use File Collections to manage user and group information, users and groups must be defined on the Control Workstation and then copied to all the SP nodes using the *supper* utility.

To add a group called db2asgrp on the Control Workstation when logged in as root, enter:

```
smitty group
-->Add a Group
```

This field was specified:

```
Group NAME ........................... db2asgrp
```

To add a user called db2inst1 on the Control Workstation when logged in as root, enter:

```
smitty user
-->Add a User
```

These fields were specified:

```
User NAME.............................. db2inst1
Primary GROUP.......................... db2asgrp
HOME directory ........................ /home/tp3an01/db2inst1
```

Instance Owner as a Normal AIX User

This is a normal AIX user, not an SP user. An SP user uses AMD (Automounter) to manage the home directory. Since the DB2 instance owner's home directory must be constantly available, mounting it over NFS is a better option.

Before distributing this user to the SP nodes we need to:

- Set its password because to be able to log in as the new user, a password must exist.
- Change its password because, by default, when we first log in as this new user, the system will prompt us to change the password.

To set the password on the Control Workstation when logged in as root, enter:

```
sp-tp3cw[/]> passwd db2inst1
Changing password for "db2inst1"
db2inst1's New password:
Enter the new password again:
```

To make the system prompt you to change the password, we logged in as db2inst1 using the AIX login command:

```
sp-tp3cw[/]> login db2inst1
3004-610 You are required to change your password.
       Please choose a new one.

db2inst1's New password:
Enter the new password again:
```

Now we are ready to distribute the new user and group information to the SP nodes. From the Control Workstation, when logged in as root, enter:

```
sp-tp3cw[/] dsh -a /var/sysman/supper update user.admin
```

Here is the output from the `supper` command:

```
tp3an01.ppd.pok.ibm.com: Updating collection user.admin from server sp-tp3.ppd.p
ok.ibm.com
tp3an01.ppd.pok.ibm.com: File Changes:  3 updated, 0 removed, 0 errors.
tp3an05.ppd.pok.ibm.com: Updating collection user.admin from server sp-tp3.ppd.p
ok.ibm.com
tp3an05.ppd.pok.ibm.com: File Changes:  3 updated, 0 removed, 0 errors.
tp3an09.ppd.pok.ibm.com: Updating collection user.admin from server sp-tp3.ppd.p
ok.ibm.com
tp3an09.ppd.pok.ibm.com: File Changes:  3 updated, 0 removed, 0 errors.
tp3an13.ppd.pok.ibm.com: Updating collection user.admin from server sp-tp3.ppd.p
ok.ibm.com
tp3an13.ppd.pok.ibm.com: File Changes:  3 updated, 0 removed, 0 errors.
```

2.3.4 Adding Services Entries

Before creating the DB2 UDB EEE instance, entries must be made in the /etc/services file on all SP nodes to be included in the instance. As we are using 4 database partitions (DPs) per SP node, we need to reserve 4 ports per SP node. To support HA/CMP failover, another 4 ports must be reserved, making 8 in total.

On tp3an01, tp3an05, tp3an09, tp3an13 these lines were added to /etc/services:

```
DB2_db2inst1            30000/tp
DB2_db2inst1_END        30007/tp
```

This means that ports 30000 to 30007 inclusive are reserved for DB2.

2.3.5 Creating an Instance

Now we are ready to create the DB2 UDB EEE instance. Logged in as root at tp3an01:

```
tp3an01[/]> cd /usr/lpp/db2_05_00/instance
tp3an01[/usr/lpp/db2_05_00/instance]> ./db2icrt -udb2inst1 db2inst1
DBI1070I Program db2icrt completed successfully.
```

Fenced User

The fenced user ID is specified (using -u) as the instance owner because security is not an issue.

2.3.6 Adding Profile Entries

Entries need to be added to .profile of db2inst1 so that environment variables and DB2 profile variables are set. On tp3an01, logged in as db2inst1, these lines were added to /home/tp3an01/db2inst1/.profile:

```
. ~/sqllib/db2profile
# Set the default DB
db2set -i db2inst1 db2dbdft=tpcd30
```

2.3.7 Editing db2nodes.cfg

The file db2nodes.cfg in the directory $INSTHOME/sqllib defines which database partition servers will be started when a `db2start` is issued. Using vi, we added these lines:

```
1  tp3an01  0 tp3sn01
2  tp3an01  1 tp3sn01
3  tp3an01  2 tp3sn01
4  tp3an01  3 tp3sn01
5  tp3an05  0 tp3sn05
6  tp3an05  1 tp3sn05
7  tp3an05  2 tp3sn05
8  tp3an05  3 tp3sn05
9  tp3an09  0 tp3sn09
10 tp3an09  1 tp3sn09
11 tp3an09  2 tp3sn09
12 tp3an09  3 tp3sn09
13 tp3an13  0 tp3sn13
14 tp3an13  1 tp3sn13
15 tp3an13  2 tp3sn13
16 tp3an13  3 tp3sn13
```

Note that:

- We started numbering the database partitions (DPs) (nodenums or first column) at 1 so that they would be synchronized with the hostnames of the SP nodes.
- Hostname (2nd column) is defined as the first Ethernet interface (en0) on each SP node.
- Logical port (3rd column) must be 0, 1, 2, 3 for the four DPs on each SP node.
- Switchname (4th column) is the network interface of the switch on each SP node.

2.3.8 Creating .rhosts Entries

Before issuing a `db2start`, we have to make sure that remote execution permission is defined across the SP nodes. In other words, if we execute `rsh tp3an05 date` from tp3an01, we will not be prompted for a password. This needs to be true for all the network interfaces defined in db2nodes.cfg.

This permission is set by adding these lines to .rhosts in /home/tp3an01/db2inst1 on tp3an01:

```
tp3an01  db2inst1
tp3sn01  db2inst1
tp3an05  db2inst1
tp3sn05  db2inst1
tp3an09  db2inst1
tp3sn09  db2inst1
tp3an13  db2inst1
tp3sn13  db2inst1
```

The permission on this file MUST be 600 by running:

```
chmod 600 .rhosts
```

Because /home/tp3an01 is a file system that is shared across the SP nodes, we only have to create these entries once.

It is recommended that you test that remote execution permission has been enabled for each network interface before issuing a `db2start`.

2.3.9 Setting up db2diag.log

Most DB2 informational and error messages are written to db2diag.log. This file is, by default, stored in the $INSTHOME/sqllib/db2dump directory, which is NFS-shared to all the SP nodes. It is a good idea to change this path so that the db2diag.log is written locally on each SP node.

To do this, logged in as db2inst1 at tp3an01:

```
db2 update dbm cfg using diagpath /tmp/db2
```

where `/tmp/db2` is a file system that exists locally on each SP node (not NFS-shared). Since this parameter (`diagpath`) is defined at the database manager level, this command is executed once for the DB2 instance.

Note that all the DPs on the same SP node will write to the same file. So output from DPs 1, 2, 3, and 4 will all go into /tmp/db2/db2diag.log on tp3an01.

2.3.10 Setting up syslog.conf

Some DB2 informational and error messages apply to the SP node and are independent of the DB2 instance. These messages are captured by configuring the AIX syslog daemon.

To do this, logged in as root:

1. Add this line to /etc/syslog.conf:

   ```
   user.warn /tmp/db2/syslog.db2
   ```

   Note: /tmp/db2 is a non-NFS directory

2. Create the /tmp/db2/syslog.db2 file

   ```
   touch tmp/db2/syslog.db2
   ```

3. Find the process id (pid) of the syslogd process:

   ```
   ps -ef|grep syslogd
   ```

4. Send a `-1` signal to this process using `kill`:

   ```
   kill -1 <syslogd-pid>
   ```

This set of commands should be run at each one of the SP nodes.

2.3.11 Starting the Instance

Now we are ready to start the instance. To do this, logged in as db2inst1, enter:

```
db2start
```

This is the output from running a `db2start`:

```
03-09-1998 14:25:00     4   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:00     1   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:00     3   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:01     2   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:06     6   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:07     8   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:07     7   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:07     5   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:07    11   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:08    10   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:08    12   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:08     9   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:08    14   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:08    13   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:09    16   0   SQL1063N  DB2START processing was successful.
03-09-1998 14:25:09    15   0   SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

Note that the 3rd column indicates the number of the database partition server. They are started in random order.

If any DP servers fail to start, for example DP server 16, you can use `db2start` to start a single DP server:

```
db2start nodenum 16
```

When you specify the nodenum parameter, you can only supply a single DP server number.

To see the processes on one SP node (tp3an01), enter:

```
db2_local_ps
```

This is a typical output from db2_local_ps:

```
Node 1
     UID    PID   PPID   C    STIME    TTY  TIME CMD
db2inst1  25428  22862   0 14:36:05      -  0:01 db2sysc 1
db2inst1  21614  25428   0 14:36:06      -  0:01 db2fcmdm 1
db2inst1  25686  25428   0 14:36:05      -  0:00 db2gds 1
db2inst1  28786  25428   0 14:36:07      -  0:00 db2pdbc 1
db2inst1  29044  25428   0 14:36:07      -  0:00 db2ipccm 1
db2inst1  30076  25428   0 14:36:07      -  0:00 db2panic (idle) 1
db2inst1  29818  25686   0 14:36:07      -  0:00 db2resyn 1
Node 2
     UID    PID   PPID   C    STIME    TTY  TIME CMD
db2inst1  27750  26976   0 14:36:06      -  0:01 db2sysc 2
db2inst1  18050  27750   1 14:36:07      -  0:01 db2fcmdm 2
db2inst1  23684  27750   0 14:36:07      -  0:00 db2pdbc 2
db2inst1  28008  27750   0 14:36:06      -  0:00 db2gds 2
db2inst1  30342  27750   0 14:36:07      -  0:00 db2ipccm 2
db2inst1  30858  27750   0 14:36:07      -  0:00 db2panic (idle) 2
db2inst1  30600  28008   0 14:36:07      -  0:00 db2resyn 2
Node 3
     UID    PID   PPID   C    STIME    TTY  TIME CMD
db2inst1  24392  24134   0 14:36:05      -  0:01 db2sysc 3
db2inst1  24650  24392   0 14:36:05      -  0:00 db2gds 3
db2inst1  26460  24392   0 14:36:06      -  0:01 db2fcmdm 3
db2inst1  27234  24392   0 14:36:06      -  0:00 db2pdbc 3
db2inst1  27492  24392   0 14:36:06      -  0:00 db2ipccm 3
db2inst1  28524  24392   0 14:36:06      -  0:00 db2panic (idle) 3
db2inst1  28266  24650   0 14:36:06      -  0:00 db2resyn 3
Node 4
     UID    PID   PPID   C    STIME    TTY  TIME CMD
db2inst1  25944  25170   0 14:36:05      -  0:01 db2sysc 4
db2inst1  20096  25944   0 14:36:07      -  0:00 db2panic (idle) 4
db2inst1  22640  25944   0 14:36:07      -  0:01 db2fcmdm 4
db2inst1  26202  25944   0 14:36:05      -  0:00 db2gds 4
db2inst1  29302  25944   0 14:36:07      -  0:00 db2pdbc 4
db2inst1  29560  25944   0 14:36:07      -  0:00 db2ipccm 4
db2inst1  15486  26202   0 14:36:07      -  0:00 db2resyn 4
```

Here we can see that there are four sets of processes per SP node, representing the four database partition servers. The name of each process is followed by the DP server number.

2.4 Creating the Database, Nodegroups, Table Spaces and Tables

Now that the DB2 UDB EEE instance is running, we can start to create the database and DB2 objects within it. We cover:

- Creating the database
- Configuring Archival Logging
- Backing up the database
- Creating nodegroups
- Creating a temporary table space
- Creating table spaces for data and index
- Creating tables

2.4.1 Creating the Database

To create the database, we issued the following commands:

```
db2 terminate
export DB2NODE=1
db2 -v "create db tpcd30 on /DB_LOG catalog tablespace managed by
database using (device '/dev/rlv_n01_01_101' 10240)"
```

Note:

- We have explicitly set `DB2NODE`; so we know which DP the System Catalogs will be stored on. We will need to know this for backup because the catalog DP must be backed up first.
- The `db2 terminate` is issued before the setting of `DB2NODE` just to make sure that any existing connections using another value of `DB2NODE` are terminated.
- The database is created on /DB_LOG so that the file systems for the logfiles will be used (for example: /DB_LOG/db2inst1/NODE0001 for the first DP).
- We have overridden the defaults for the container used for the System Catalog table space (SYSCATSPACE). We use the DMS raw device, lv_n01_01_101.

- The size is set to 10240 4 KB pages or 40 MB. This should be sufficient for the number of DB2 objects we intend to store in the System Catalogs.

2.4.2 Setting Archival Logging

To enable archival logging, we set `logretain` on at this point. This will force a backup, which will be much faster now rather than after we load the data. To run `db2 update db cfg` at all DPs, we issued:

```
db2_all "||db2 update db cfg for tpcd30 using logretain on"
```

The double pipe (||) before the DB2 command means that the command will be run in parallel on all DPs.

See also “Running Commands on Multiple Database Partitions” on page 255.

2.4.2.1 Backing up the Database

To backup the database requires a `db2 backup db` command be run on all DPs. The Catalog DP backup must finish before the other DPs can start in parallel. To achieve this:

```
db2_all "<<+1<db2 backup db tpcd30 to /backdb"
db2_all "<<-1<||db2 backup db tpcd30 to /backdb"
```

Note that:

- `<<+1<` means runs the following command only at DP 1.
- `<<-1<||` means run the following command in parallel at all DPs except DP 1.

2.4.3 Creating Nodegroups

We decided to create two nodegroups to hold the user table spaces and tables. The first, NG_BIG, covers DPs 2 to 15 and will hold all tables apart from the very small tables (nation and region). The second, NG_LIT, is defined on DP 1 only and is for the very small tables.

```
create nodegroup NG_BIG on nodes (2 to 16);
create nodegroup NG_LIT on nodes (1);
```

2.4.4 Creating a Temporary Table Space

A temporary table space, TS_TMP, is created to make use of the file systems that were created for temporary usage. First, the new temporary table space is created, and then the default temporary table space is dropped. This order

of events is necessary because the database must have at least one temporary table space defined in it.

```
create temporary tablespace TS_TMP in nodegroup IBMTEMPGROUP
managed by system
using ('/DB_TMP/db2inst1/NODE0001/T1','/DB_TMP/db2inst1/NODE0001/T2')
on node (1)
using ('/DB_TMP/db2inst1/NODE0002/T1','/DB_TMP/db2inst1/NODE0002/T2')
on node (2)
using ('/DB_TMP/db2inst1/NODE0003/T1','/DB_TMP/db2inst1/NODE0003/T2')
on node (3)
using ('/DB_TMP/db2inst1/NODE0004/T1','/DB_TMP/db2inst1/NODE0004/T2')
on node (4)
using ('/DB_TMP/db2inst1/NODE0005/T1','/DB_TMP/db2inst1/NODE0005/T2')
on node (5)
using ('/DB_TMP/db2inst1/NODE0006/T1','/DB_TMP/db2inst1/NODE0006/T2')
on node (6)
using ('/DB_TMP/db2inst1/NODE0007/T1','/DB_TMP/db2inst1/NODE0007/T2')
on node (7)
using ('/DB_TMP/db2inst1/NODE0008/T1','/DB_TMP/db2inst1/NODE0008/T2')
on node (8)
using ('/DB_TMP/db2inst1/NODE0009/T1','/DB_TMP/db2inst1/NODE0009/T2')
on node (9)
using ('/DB_TMP/db2inst1/NODE0010/T1','/DB_TMP/db2inst1/NODE0010/T2')
on node (10)
using ('/DB_TMP/db2inst1/NODE0011/T1','/DB_TMP/db2inst1/NODE0011/T2')
on node (11)
using ('/DB_TMP/db2inst1/NODE0012/T1','/DB_TMP/db2inst1/NODE0012/T2')
on node (12)
using ('/DB_TMP/db2inst1/NODE0013/T1','/DB_TMP/db2inst1/NODE0013/T2')
on node (13)
using ('/DB_TMP/db2inst1/NODE0014/T1','/DB_TMP/db2inst1/NODE0014/T2')
on node (14)
using ('/DB_TMP/db2inst1/NODE0015/T1','/DB_TMP/db2inst1/NODE0015/T2')
on node (15)
using ('/DB_TMP/db2inst1/NODE0016/T1','/DB_TMP/db2inst1/NODE0016/T2')
on node (16);

drop tablespace TEMPSPACE1
```

Note that relative paths can also be used for the container definitions. For example, if the database had been created on /DB_TMP, then we could run this command:

```
create temporary tablespace TS_TMP in nodegroup IBMDEFAULTGROUP managed
by system using ('T1','T2')
```

2.4.5 Creating Table Spaces for Data and Index

These are the commands that were issued to create the five table spaces to be used for data and index:

For TS_LIT, which will hold the very small tables (data and index):

```
create tablespace TS_LIT in NG_LIT
managed by database
using (device '/dev/rlv_n01_01_102' 10240);
```

For TS_DAT_MED, which will hold the medium tables data:

```
create tablespace TS_DAT_MED in NG_BIG
managed by database
using (device '/dev/rlv_n01_01_210' 153600,
       device '/dev/rlv_n01_01_211' 153600) on node (2)
using (device '/dev/rlv_n01_02_310' 153600,
       device '/dev/rlv_n01_02_311' 153600) on node (3)
using (device '/dev/rlv_n01_02_410' 153600,
       device '/dev/rlv_n01_02_411' 153600) on node (4)
using (device '/dev/rlv_n05_01_110' 153600,
       device '/dev/rlv_n05_01_111' 153600) on node (5)
using (device '/dev/rlv_n05_01_210' 153600,
       device '/dev/rlv_n05_01_211' 153600) on node (6)
using (device '/dev/rlv_n05_02_310' 153600,
       device '/dev/rlv_n05_02_311' 153600) on node (7)
using (device '/dev/rlv_n05_02_410' 153600,
       device '/dev/rlv_n05_02_411' 153600) on node (8)
using (device '/dev/rlv_n09_01_110' 153600,
       device '/dev/rlv_n09_01_111' 153600) on node (9)
using (device '/dev/rlv_n09_01_210' 153600,
       device '/dev/rlv_n09_01_211' 153600) on node (10)
using (device '/dev/rlv_n09_02_310' 153600,
       device '/dev/rlv_n09_02_311' 153600) on node (11)
using (device '/dev/rlv_n09_02_410' 153600,
       device '/dev/rlv_n09_02_411' 153600) on node (12)
using (device '/dev/rlv_n13_01_110' 153600,
       device '/dev/rlv_n13_01_111' 153600) on node (13)
using (device '/dev/rlv_n13_01_210' 153600,
       device '/dev/rlv_n13_01_211' 153600) on node (14)
using (device '/dev/rlv_n13_02_310' 153600,
       device '/dev/rlv_n13_02_311' 153600) on node (15)
using (device '/dev/rlv_n13_02_410' 153600,
       device '/dev/rlv_n13_02_411' 153600) on node (16);
```

Note that the size of 153600 4 KB pages (or 600 MB) is equal to the size of the LVs that have already been created. See “Sizing of LVs and FSs” on page 14 for more details.

There is no `ON NODE` clause for node 1 because the TS_DAT_MED table space is not defined on the first database partition. The relationship between the table space name, device name and database partition is shown in Table 1 on page 15.

For TS_IND_MED, which will hold the medium tables indexes:

```
create tablespace TS_IND_MED in NG_BIG
managed by database
using (device '/dev/rlv_n01_01_212' 153600,
       device '/dev/rlv_n01_01_213' 153600) on node (2)
using (device '/dev/rlv_n01_02_312' 153600,
       device '/dev/rlv_n01_02_313' 153600) on node (3)
using (device '/dev/rlv_n01_02_412' 153600,
       device '/dev/rlv_n01_02_413' 153600) on node (4)
using (device '/dev/rlv_n05_01_112' 153600,
       device '/dev/rlv_n05_01_113' 153600) on node (5)
using (device '/dev/rlv_n05_01_212' 153600,
       device '/dev/rlv_n05_01_213' 153600) on node (6)
using (device '/dev/rlv_n05_02_312' 153600,
       device '/dev/rlv_n05_02_313' 153600) on node (7)
using (device '/dev/rlv_n05_02_412' 153600,
       device '/dev/rlv_n05_02_413' 153600) on node (8)
using (device '/dev/rlv_n09_01_112' 153600,
       device '/dev/rlv_n09_01_113' 153600) on node (9)
using (device '/dev/rlv_n09_01_212' 153600,
       device '/dev/rlv_n09_01_213' 153600) on node (10)
using (device '/dev/rlv_n09_02_312' 153600,
       device '/dev/rlv_n09_02_313' 153600) on node (11)
using (device '/dev/rlv_n09_02_412' 153600,
       device '/dev/rlv_n09_02_413' 153600) on node (12)
using (device '/dev/rlv_n13_01_112' 153600,
       device '/dev/rlv_n13_01_113' 153600) on node (13)
using (device '/dev/rlv_n13_01_212' 153600,
       device '/dev/rlv_n13_01_213' 153600) on node (14)
using (device '/dev/rlv_n13_02_312' 153600,
       device '/dev/rlv_n13_02_313' 153600) on node (15)
using (device '/dev/rlv_n13_02_412' 153600,
       device '/dev/rlv_n13_02_413' 153600) on node (16);
```

For TS_DAT_BIG, which will hold lineitem's data:

```
create tablespace TS_DAT_BIG in NG_BIG
managed by database
using (device '/dev/rlv_n01_01_206' 307200,
       device '/dev/rlv_n01_01_207' 307200) on node (2)
using (device '/dev/rlv_n01_02_306' 307200,
       device '/dev/rlv_n01_02_307' 307200) on node (3)
using (device '/dev/rlv_n01_02_406' 307200,
       device '/dev/rlv_n01_02_407' 307200) on node (4)
using (device '/dev/rlv_n05_01_106' 307200,
       device '/dev/rlv_n05_01_107' 307200) on node (5)
using (device '/dev/rlv_n05_01_206' 307200,
       device '/dev/rlv_n05_01_207' 307200) on node (6)
using (device '/dev/rlv_n05_02_306' 307200,
       device '/dev/rlv_n05_02_307' 307200) on node (7)
using (device '/dev/rlv_n05_02_406' 307200,
       device '/dev/rlv_n05_02_407' 307200) on node (8)
using (device '/dev/rlv_n09_01_106' 307200,
       device '/dev/rlv_n09_01_107' 307200) on node (9)
using (device '/dev/rlv_n09_01_206' 307200,
       device '/dev/rlv_n09_01_207' 307200) on node (10)
using (device '/dev/rlv_n09_02_306' 307200,
       device '/dev/rlv_n09_02_307' 307200) on node (11)
using (device '/dev/rlv_n09_02_406' 307200,
       device '/dev/rlv_n09_02_407' 307200) on node (12)
using (device '/dev/rlv_n13_01_106' 307200,
       device '/dev/rlv_n13_01_107' 307200) on node (13)
using (device '/dev/rlv_n13_01_206' 307200,
       device '/dev/rlv_n13_01_207' 307200) on node (14)
using (device '/dev/rlv_n13_02_306' 307200,
       device '/dev/rlv_n13_02_307' 307200) on node (15)
using (device '/dev/rlv_n13_02_406' 307200,
       device '/dev/rlv_n13_02_407' 307200) on node (16);
```

Note that the size of 307200 4 KB pages (or 1200 MB) equals the size of the LVs that have been created in “Sizing of LVs and FSs” on page 14.

For TS_IND_BIG, which will hold lineitem’s indexes:

```
create tablespace TS_IND_BIG in NG_BIG
managed by database
using (device '/dev/rlv_n01_01_208' 307200,
       device '/dev/rlv_n01_01_209' 307200) on node (2)
using (device '/dev/rlv_n01_02_308' 307200,
       device '/dev/rlv_n01_02_309' 307200) on node (3)
using (device '/dev/rlv_n01_02_408' 307200,
       device '/dev/rlv_n01_02_409' 307200) on node (4)
using (device '/dev/rlv_n05_01_108' 307200,
       device '/dev/rlv_n05_01_109' 307200) on node (5)
using (device '/dev/rlv_n05_01_208' 307200,
       device '/dev/rlv_n05_01_209' 307200) on node (6)
using (device '/dev/rlv_n05_02_308' 307200,
       device '/dev/rlv_n05_02_309' 307200) on node (7)
using (device '/dev/rlv_n05_02_408' 307200,
       device '/dev/rlv_n05_02_409' 307200) on node (8)
using (device '/dev/rlv_n09_01_108' 307200,
       device '/dev/rlv_n09_01_109' 307200) on node (9)
using (device '/dev/rlv_n09_01_208' 307200,
       device '/dev/rlv_n09_01_209' 307200) on node (10)
using (device '/dev/rlv_n09_02_308' 307200,
       device '/dev/rlv_n09_02_309' 307200) on node (11)
using (device '/dev/rlv_n09_02_408' 307200,
       device '/dev/rlv_n09_02_409' 307200) on node (12)
using (device '/dev/rlv_n13_01_108' 307200,
       device '/dev/rlv_n13_01_109' 307200) on node (13)
using (device '/dev/rlv_n13_01_208' 307200,
       device '/dev/rlv_n13_01_209' 307200) on node (14)
using (device '/dev/rlv_n13_02_308' 307200,
       device '/dev/rlv_n13_02_309' 307200) on node (15)
using (device '/dev/rlv_n13_02_408' 307200,
       device '/dev/rlv_n13_02_409' 307200) on node (16);
```

2.4.6 Creating the Tables

For the small tables, the index and data are stored in TS_LIT:

```
create table NATION   (              N_NATIONKEY  integer not null,
  N_NAME        char(25) not null,   N_REGIONKEY  integer not null,
  N_COMMENT     varchar(152))
in TS_LIT;

create table REGION   (              R_REGIONKEY  integer not null,
  R_NAME        char(25) not null,   R_COMMENT    varchar(152))
in TS_LIT;
```

For the medium tables, the data is stored in TS_DAT_MED, and the indexes are stored TS_IND_MED:

```
create table PART     (              P_PARTKEY    integer not null,
  P_NAME        varchar(55) not null, P_MFGR       char(25) not null,
  P_BRAND       char(10) not null,    P_TYPE       varchar(25) not null,
  P_SIZE        integer not null,     P_CONTAINER  char(10) not null,
  P_RETAILPRICE float not null,       P_COMMENT    varchar(23) not null)
in TS_DAT_MED index in TS_IND_MED;

create table SUPPLIER (              S_SUPPKEY    integer not null,
  S_NAME        char(25) not null,    S_ADDRESS    varchar(40) not null,
  S_NATIONKEY   integer not null,     S_PHONE      char(15) not null,
  S_ACCTBAL     float not null,       S_COMMENT    varchar(101) not null)
in TS_DAT_MED index in TS_IND_MED;

create table PARTSUPP (              PS_PARTKEY   integer not null,
  PS_SUPPKEY    integer not null,     PS_AVAILQTY  integer not null,
  PS_SUPPLYCOST float not null,       PS_COMMENT   varchar(199) not null)
in TS_DAT_MED index in TS_IND_MED;

create table CUSTOMER (              C_CUSTKEY    integer not null,
  C_NAME        varchar(25) not null, C_ADDRESS    varchar(40) not null,
  C_NATIONKEY   integer not null,     C_PHONE      char(15) not null,
  C_ACCTBAL     float not null,       C_MKTSEGMENT char(10) not null,
  C_COMMENT     varchar(117) not null)
in TS_DAT_MED index in TS_IND_MED;

create table ORDERS   (              O_ORDERKEY    integer not null,
  O_CUSTKEY       integer not null,   O_ORDERSTATUS char(1) not null,
  O_TOTALPRICE    float not null,     O_ORDERDATE   date not null,
  O_ORDERPRIORITY char(15) not null,  O_CLERK       char(15) not null,
  O_SHIPPRIORITY  integer not null,   O_COMMENT     varchar(79) not null)
in TS_DAT_MED index in TS_IND_MED;
```

For the large table, the data is stored in TS_DAT_BIG, and the indexes are stored TS_IND_BIG:

```
create table LINEITEM (              L_ORDERKEY    integer not null,
  L_PARTKEY       integer not null,   L_SUPPKEY     integer not null,
  L_LINENUMBER    integer not null,   L_QUANTITY    float not null,
  L_EXTENDEDPRICE float not null,     L_DISCOUNT    float not null,
  L_TAX           float not null,     L_RETURNFLAG  char(1) not null,
  L_LINESTATUS    char(1) not null,   L_SHIPDATE    date not null,
  L_COMMITDATE    date not null,      L_RECEIPTDATE date not null,
  L_SHIPINSTRUCT  char(25) not null,  L_SHIPMODE    char(10) not null,
  L_COMMENT       varchar(44) not null)
in TS_DAT_BIG index in TS_IND_BIG;
```

2.4.7 Creating Indexes

The indexes are created after the load. Because we will be using concurrent db2splits, the order of input data cannot be guaranteed. To have the data in each table clustered by the desired index, we will have to REORG each table

after loading it. So building indexes during the load is not a good idea as they would have to be rebuilt after the REORG. See “Reorganizing the Table on the Clustering Index” on page 44 for more details.

2.5 Loading Data into the Database

The task of loading data into each table involved these steps:

- Create the input data.
- Load the data into the table using Autoloader.
- Create the index used for clustering.
- Reorganize the table on that index.
- Create the other indexes.

This methodology assumes that we know by which index the table should be clustered.

When loading this volume of data (for example 21 GB for lineitem), using concurrent db2splits with Autoloader is ideal. This is because:

- If we use a single db2split process, the db2split processing, not the input data stream, is the bottleneck in the complete load cycle. Using multiple db2split processes speeds up the splitting part of the process.
- Using multiple db2split processes will mean sending large amounts of data between the SP nodes. We have a high-speed switch, so this task is very fast.

2.5.1 Creating the Input Data

The Transaction Processing Council D (TPCD) package contains a program called *dbgen* which allows you to generate data for a TPCD database of a user-defined size (in this case 30 GB), and for one particular table in the TPCD database. The output from dbgen is piped into Autoloader. The tables must be treated individually since one Autoloader job can only work on one table at a time.

We tested with the lineitem table at the 1 GB TPCD database size first before trying the 30 GB version. From our testing it became apparent that the bottleneck for loading the data was the data stream coming from dbgen. So for the purposes of the tests, we decided to find a way to speed up the input stream. The way chosen was to create a lineitem table at the 1 GB size, output this to disk, and then cat this file 30 times sequentially into a pipe used as the input for the Autoloader job. The implications of this are:

- The database is not a "real" 30 GB TPCD database, but a 1 GB TPCD duplicated 30 times.
- Any unique indexes will have to be made non-unique.

These implications are acceptable as the objectives of these tests were:

- Design the disk storage and load a 30 GB database
- Implement a backup and recovery strategy using ADSM
- Implement a failover strategy using HACMP

We do not test the TPCD queries themselves.

2.5.2 Using Autoloader with Concurrent db2splits

Here is the lineitem.aload script used:

```
# L1 is 1 GB version of lineitem table

date
rm lineitem.tbl
mkfifo lineitem.tbl

db2autold -c lineitem.cfg &

cat L1 L1 L1 L1 L1 L1 L1 L1 L1 L1 \
    L1 L1 L1 L1 L1 L1 L1 L1 L1 L1 \
    L1 L1 L1 L1 L1 L1 L1 L1 L1 L1 \
> lineitem.tbl

wait

rm lineitem.tbl
date
```

Note that this script must be run from a directory which is available (using NFS) from all the SP nodes.

Here is the Autoloader configuration file, lineitem.cfg:

```
RELEASE=V5.0

db2 "load from lineitem.tbl of del modified by coldel| replace into lineitem \
     nonrecoverable using /work"

DATABASE=tpcd30

OUTPUT_NODES=(2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)

SPLIT_NODES=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)

MODE=SPLIT_AND_LOAD

LOGFILE=LOG

NOTNFS_DIR=/work

CHECK_LEVEL=NOCHECK

TRACE=1
```

Note that:

- The option `nonrecoverable` was used in the `load` command because this will leave the table space for lineitem (TS_DAT_BIG) in a normal state after the load has finished. Without this option, the table space would be put into a "Backup Pending" state, forcing us to do a backup before the next step (create index).
- After a table space has been put into "Backup Pending" as the result of a load without "nonrecoverable", you are permitted to:
  - Perform another load of a table in the same table space
  - Select data from tables in the table space

  You are not permitted to change the data of tables in the table space in any way (such as insert, update or delete on its tables), or add or drop indexes on the tables in the table space. You will receive this error:

  ```
  SQL0290N Table space access is not allowed.
  ```

2.5.3 When to Create Indexes

We chose to create the indexes after Autoloader processing for the following reasons:

- When we use concurrent db2splits, the order of the input data cannot be guaranteed, meaning that there is less value in creating the indexes during the load as we will have to REORG the data anyway.
- In order to create indexes during the load:

- Sort directories must be defined in the load command. This requires a large amount of additional disk space, enough to hold all the index keys for all the indexes defined on the table.
- Since we can only supply one load command to Autoloader, each of the four DPs per SP node must share the same sort directories.

- If we try to create indexes during the load and we do run out of sort space, the whole Autoloader process must be restarted.

2.5.4 The Number of db2split Processes

We have 15 DPs and 32 CPUs in total over four SP nodes. Therefore 15 loads will be run in parallel. If we configure Autoloader to run 16 db2split processes, then there should be at least one db2psplit, 16 db2splits and 15 loads (making a total of 32) processes running concurrently across the 32 CPUs. So all the CPUs should be kept busy during this Autoloader job. In fact, subsequent testing showed that a lower number of db2split processes will keep the CPUs busy.

2.5.5 Notes on Using Autoloader

1. To monitor Autoloader, you can copy the db2autold script and then modify the copy. By changing debug=0 to debug=1 at line 49, this will cause `set -x` to be applied throughout the script, and so all executed lines will be echoed to the screen.
2. Because we were running remotely, and we needed the Autoloader job to keep running even if our network connection to the RS/6000 SP was broken, we used `nohup`, as in:

   ```
   nohup timex lineitem.aload &
   ```

   If you use `nohup` before a command:

   - The output from the command is sent to a file called nohup.out
   - If the session that initiated the command is terminated, the command will continue to run.
3. When running a lot of db2split and load processes, the time taken for Autoloader to prepare the named pipes and directories is relatively long. This is because the pipes and directories are created serially with error checking after each one.
4. In our environment, namely 16 DPs on four SP nodes, the following changes to the supplied db2autold script improved the startup time:

- The function `check_remote_access_permission` was commented out. Normally, this checks that `rsh` commands can be run on SP nodes using the switch.
- Creating the named pipes. For our tests, 240 named piped were created (16*15) just for the db2split and load processing. These pipes are created in serial normally. We changed the code to run `mknod` in the background and then `wait` for all to finish.
- Creating the "psplitload" and "psplittemp" directories
  - For our tests, we changed the script to run `mkdir` in the background and then `wait` for all to finish.

These changes reduced the startup time from 20 mins to 5 mins.

Note that these changes speed up the startup processing of the db2autold script, especially for our environment (16 db2splits, 15 loads). They bypass error checking. We felt safe doing this because we knew that permissions were already in place to perform these functions (`rsh`, `mknod` and `mkdir`). In the worst case, if a problem arose from making these changes, we could resume using the supplied db2autold script. (After having run a cleanup!).

These changes improved the cleanup time:

- All `rsh` jobs were started in background and then wait to finish

2.5.6 Problems running Autoloader

Sometimes db2autold fails, and in one or several of the db2split or load logs you see:

```
SQL2043N Unable to start a child process or thread.
```

We changed `maxuproc` from 500 to 1000 on all SP nodes and the problem did not occur again.

2.5.7 Autoloader Log files

The log files for the load processes are kept in files called load_LOG.<DP>. For example, for DP 2, the file is load_LOG.2:

```
 Database Connection Information

 Database product       = DB2/6000 5.0.0
 SQL authorization ID   = DB2INST1
 Local database alias   = TPCD30

SQL3109N  The utility is beginning to load data from file
"/work/db2inst1/psplittemp/lineitem.tbl.002".

SQL3500W  The utility is beginning the "LOAD" phase at time "03-06-1998
15:22:42.692650".

SQL3519W  Begin Load Consistency Point. Input record count = "0".

SQL3520W  Load Consistency Point was successful.

SQL3110N  The utility has completed processing.  "12082560" rows were read
from the input file.

SQL3519W  Begin Load Consistency Point. Input record count = "12082560".

SQL3520W  Load Consistency Point was successful.

SQL3515W  The utility has finished the "LOAD" phase at time "03-06-1998
17:53:34.725487".


Number of rows read         = 12082560
Number of rows skipped      = 0
Number of rows loaded       = 12082560
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 12082560

0
DB20000I  The TERMINATE command completed successfully.
```

We can see from this log file that:

- 12,082,560 lines were loaded at DP 2.
- The load for DP 2 started at 15:22:42 and finished at 17:53:34.
- This makes a total elapsed time of: 2 hours, 30 minutes, 52 seconds.

The log files for the split processes are kept in files called splt_LOG.<DP>. For example, for DP 2, the file is splt_LOG.2:

```
The log file "splt_LOG.2" was opened successfully.
Starting time: "03-06-1998 15:23:08.303928".
The type of the input date file is "1" (0-ASC, 1-DEL, 2-BIN).
The input file was not found, using stdin as input.
Program is running with check level: "NOCHECK".
The string delimiter is """, the column delimiter is "|", and the decimal point
is ".". Tracing "1" delimited record(s).
The output partitioning map file "LINEITEM.lineitem.tbl.map" was opened
successfully.
Read of input partitioning map is in progress.
Input partitioning map was successfully read.
The run type is "PARTITION".
The output partitioning map file "OutMap.2" was opened successfully.
Distribution file name: "DISTFILE.2".
The distribution file "DISTFILE.2" was opened successfully for writing.
This utility is using " 1" partitioning keys.
"L_ORDERKEY  "  Start:"   0"  Len:"   0"  Position:"   1"  Type:"NN(-1)
INTEGER".
The output data file will be "/work/db2inst1/psplitload/lineitem.tbl2".
All output data files were opened successfully.
Processing record number "           1".
Key Index: "0". Data: "L_ORDERKEY" "1" "0" "0" "33".
Paritioning number returned from hash function: "0349" (hex) " 841" (decimal).
Processed "       50000" records (or lines).
Processed "      100000" records (or lines).
(.......................SKIPPED.............................................)
Processed "    11050000" records (or lines).
Processed "    11100000" records (or lines).
Writing output partition map to file "OutMap.2".
Writing distribution map to "DISTFILE.2".
Total number of records processed: " 11112444".
Total number of records discarded: "        0".
Stop time "03-06-1998 17:53:33.755367".
Elapsed time: " 2" hours, "30" minutes, "25" seconds.
Throughput: "1231" records/sec.
Record counts for output nodes:
Node: "16". Record count: "743869".
Node: "15". Record count: "739931".
Node: "14". Record count: "732357".
Node: "13". Record count: "743043".
Node: "12". Record count: "737117".
Node: "11". Record count: "737207".
Node: "10". Record count: "743003".
Node: "9". Record count: "739060".
Node: "8". Record count: "742680".
Node: "7". Record count: "743197".
Node: "6". Record count: "747024".
Node: "5". Record count: "743851".
Node: "4". Record count: "735586".
Node: "3". Record count: "737253".
Node: "2". Record count: "747266".
Complete.
Program ran successfully with "0" warning message(s) and "0" discarding
record(s).
```

We can see from this log file that:

- The db2split for DP 2 started at 15:23:08 and finished at 17:53:33
- The total elapsed time was 2 hours, 30 minutes, 25 seconds.
- The db2split processes start just after the load processes and finish just before the end of the load processes.
- The number of lines processed not same as loaded at DP 2. This is because the output from each db2split is divided among the 15 DPs.

2.5.8 Verifying the Load

To verify how many rows were actually loaded in the lineitem table, enter:

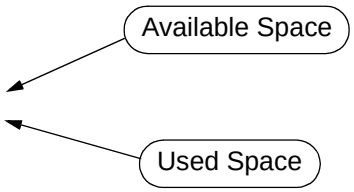
```
 > timex db2 "select count(*) from lineitem"

1
-----------
  180036450

  1 record(s) selected.


real 252.12
user 0.01
sys  0.14
```

To see how much of the table space for lineitem's data (TS_DAT_BIG) has been used up, enter:

```
db2 terminate; export DB2NODE=2
db2 list tablespaces show detail
```

Then if we look at the output for TS_DAT_BIG:

```
 Tablespace ID                        = 6
 Name                                 = TS_DAT_BIG
 Type                                 = Database managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal
 Total pages                          = 614400
 Useable pages                        = 614336
 Used pages                           = 419552
 Free pages                           = 194784
 High water mark (pages)              = 419552
 Page size (bytes)                    = 4096
 Extent size (pages)                  = 32
 Prefetch size (pages)                = 32
 Number of containers                 = 2
 Minimum recovery time                = 1998-03-06-20.19.21.000000
```



We can see that 419,552 pages out of a total of 614,336 (or 68 percent) usable pages have been filled. This leaves some room for growth.

2.5.9 Creating the Index on which the Data is Clustered

Now that the data portion of the table has been loaded, we must ensure that the data is clustered with respect to one of the indexes. We chose the index on L_ORDERKEY for clustering, which is called FK_L_OKEY. First, we must create the index:

```
create index FK_L_OKEY on LINEITEM (L_ORDERKEY)
DB20000I  The SQL command completed successfully.


real 753.62
user 0.02
sys  0.13
```

Next, we must run REORGCHK to check the current CLUSTERRATIO of the data with respect to this index, FK_L_OKEY:

```
db2 terminate
export DB2NODE=2
db2 -v "reorgchk on table db2inst1.lineitem"
```

Here is the part of the output from REORGCHK relating to the index, FK_L_OKEY:

```
Index statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
F5: 100*(KEYS*(ISIZE+10)+(CARD-KEYS)*4) / (NLEAF*4096) > 50
F6: 90*((4000/(ISIZE+10))**(NLEVELS-2))*4096/ (KEYS*(ISIZE+10)+(CARD-KEYS)*4)<10
0

CREATOR  NAME                CARD  LEAF  LVLS ISIZE   KEYS   F4   F5  F6 REORG
-------------------------------------------------------------------------------
Table: DB2INST1.LINEITEM
DB2INST1 FK_L_OKEY           2e+08 2e+05     3     4  2e+06   74   73    0 *--
-------------------------------------------------------------------------------
```

We can see that the CLUSTERRATIO (F4) is 74 percent . If we run a REORG based on this index, the CLUSTERRATIO should change to 100 percent, meaning that the data is perfectly clustered on the index FK_L_OKEY.

Note that we set explicitly the value of DB2NODE to 2 before running the REORGCHK. This ensures that the analysis will be based on the data at DP 2. If we happen to have a connection to the database still available at DP 1, (which has no data for this table) the REORGCHK will still work. DB2 will find the first DP where the lineitem table has data.

2.5.10 Reorganizing the Table on the Clustering Index

The next step is to REORG the lineitem table on the clustering index, FK_L_OKEY:

```
reorg table db2inst1.lineitem index db2inst1.fk_l_okey use ts_tmp
DB20000I  The REORG TABLE command completed successfully.


real 3369.62
user 0.02
sys  0.30
```

Note that:

- We specified **use ts_tmp**. This means that the temporary table space, TS_TMP, will be used to hold a temporary copy of the table during REORG processing. If we don't specify a value, then the REORG utility will use the same table space where lineitem's data is held, namely TS_DAT_BIG. By using TS_TMP, we will also be using different disks (compared to those used by TS_DAT_BIG) to hold the temporary copy. This will result in a much shorter elapsed time for the REORG.
- There must be enough log files (LOGPRIMARY plus LOGSECOND) defined for the REORG to complete. We changed the number of secondary log files to 48 using:

```
db2_all "||db2 -v update db cfg for tpcd30 using LOGSECOND 48"
```

The contents of the log file directory for DP 2 (/DB_LOG/db2inst1/NODE0002/SQL00001/SQLOGDIR) after the REORG was:

```
-rw-------   1 db2inst1 dbadmin1  139264 Mar 06 23:37 S0000003.LOG
-rw-------   1 db2inst1 dbadmin1  811008 Mar 07 00:14 S0000004.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 00:36 S0000005.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 00:39 S0000006.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 00:42 S0000007.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 00:46 S0000008.LOG
-rw-------   1 db2inst1 dbadmin1 3891200 Mar 07 00:49 S0000009.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:17 S0000010.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:20 S0000011.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:24 S0000012.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:27 S0000013.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:30 S0000014.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:33 S0000015.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:36 S0000016.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:40 S0000017.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:43 S0000018.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:46 S0000019.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 08:49 S0000020.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 09:08 S0000021.LOG
-rw-------   1 db2inst1 dbadmin1  184320 Mar 07 09:10 S0000022.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 09:10 S0000023.LOG
-rw-------   1 db2inst1 dbadmin1 4104192 Mar 07 09:10 S0000024.LOG
```


We can see that as our job started at 08:17 and finished at 09:10, that 15 log files were needed in this UOW; so LOGPRIMARY + LOGSECOND must be greater than 15 for this UOW to finish. Note that the LOGFILSZ was left at the default of 1000 (so each log files is 4 MB in size).

You can use the `GET SNAPSHOT` command to estimate the amount of log space needed. For instance, after the REORG completed, we ran a snapshot:

```
db2 terminate
export DB2NODE=2
db2 get snapshot for all on tpcd30 | egrep 'Log pages w|Rows read'
```

This is the result:

```
Log pages written                          = 11410
Rows read                                  = 11625924
```

We can see that:

```
log space used = 11410 * 4K = 11.1 4 MB log files
```

The value of Rows read and Log pages written can be used to calculate the amount of log space required for the REORG. If we test with 1 percent of the

data for example, the log pages written value for 1 percent can be multiplied by 100 to give the required log space at full size.

We must also be careful not to run out of sort space when running REORG. Since we are using TS_TMP for sort space, if we run:

```
du -ak /DB_TMP/db2inst1/NODE0002/T1
```

We can see the temporary file used for REORG processing, in this case SQL00002.DTR:

```
Sat Mar  7 09:01:41 EST 1998
4       /DB_TMP/db2inst1/NODE0002/T1/SQLTAG.NAM
839552  /DB_TMP/db2inst1/NODE0002/T1/SQL00002.DTR
839560  /DB_TMP/db2inst1/NODE0002/T1
```

This file reached a maximum size of 820 MB. As there are two containers (T1 and T2) per DP, the space required by REORG per DP is 1.6 GB. If we look at the space taken by the data for lineitem per DP, we can see from doing a `db2 list tablespaces show detail`, that for TS_DAT_BIG at DP 2:

```
Used pages = 419552
```

Since each page is 4 KB, this equals 1.638 GB of data per DP. This means that for the REORG to complete, the table space used for REORG temporary processing must be at least as big as the space taken by the data for the table.

Note that as we have done no updates or deletes to the lineitem table, the space taken by REORG is easy to calculate.

2.5.11 Running REORGCHK to Check Clustering

Now that the REORG has completed, we run REORGCHK again to see the effect on the CLUSTERRATIO:

```
CREATOR  NAME                CARD  LEAF  LVLS ISIZE   KEYS   F4   F5  F6 REORG
------------------------------------------------------------------------------
Table: DB2INST1.LINEITEM
DB2INST1 FK_L_OKEY          2e+08 2e+05     3     4  2e+06  100   73    0 ---
------------------------------------------------------------------------------
```

This shows that CLUSTERRATIO (F4) has changed to 100 percent.

2.5.12 Creating Other indexes

Now we will create the other indexes on lineitem:

```
create index L_SDATE on LINEITEM (L_SHIPDATE)
DB20000I  The SQL command completed successfully.

create index L_RDATE_CDATE on LINEITEM (L_RECEIPTDATE,L_COMMITDATE)
DB20000I  The SQL command completed successfully.


real 2148.77
user 0.03
sys  0.14
```

We run REORGCHK again to see the CLUSTERRATIO for each index:

```
CREATOR  NAME                 CARD  LEAF  LVLS ISIZE   KEYS   F4   F5  F6 REORG
-------------------------------------------------------------------------------
Table: DB2INST1.LINEITEM
DB2INST1 FK_L_OKEY          2e+08 2e+05     3     4  2e+06  100   73    0 ---
DB2INST1 L_RDATE_CDATE      2e+08 2e+05     3     8 246247   83   73    0 ---
DB2INST1 L_SDATE            2e+08 2e+05     3     4   2525   83   73    0 ---
-------------------------------------------------------------------------------
```

As the CLUSTERRATIO (F4) is over 80 percent for all indexes, they will all be considered by the optimizer for index access to the table.

2.5.13 Space Taken by the Indexes

If we look at the output relating to TS_IND_BIG from `db2 list tablespaces show detail` at DP 2, we see:

```
 Tablespace ID                        = 7
 Name                                 = TS_IND_BIG
 Type                                 = Database managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal
 Total pages                          = 614400
 Useable pages                        = 614336
 Used pages                           = 49056
 Free pages                           = 565280
 High water mark (pages)              = 49056
 Page size (bytes)                    = 4096
 Extent size (pages)                  = 32
 Prefetch size (pages)                = 32
 Number of containers                 = 2
 Minimum recovery time                = 1998-02-20-19.44.27.000000
```

This shows that there is enough space to create more indexes; which we may need as this is a Decision Support System (DSS) style database.

Chapter 3. DB2 UDB EEE Backup and Recovery using ADSM

This chapter first covers the backup and recovery features of DB2 UDB EEE. This includes recommendations for implementing a backup and recovery strategy.

This chapter also includes a detailed guide to configuring and managing a DB2 UDB EEE backup and recovery environment using ADSM Server Version 3 and AIX Version 4.2.1 on an RS/6000 SP with a switch. In addition, it addresses the installation and customization of the ADSM server and the ADSM clients, shows how to modify DB2 UDB EEE to use ADSM, and provides sample procedures to perform online backups and to manage database backup files and archived logs.

3.1 Overview of DB2 UDB EEE Backup and Recovery

To begin to create a backup and recovery strategy, the recovery requirements of each database must be defined. Once these requirements are determined, the database administrator can develop backup procedures to enable recovery.

Some items to consider when developing a recovery plan follow:

- How critical is the database?
- How large is the database?
- How volatile is the database?
- How current must the database be?
- What about application errors?
- How much space can be allocated for backup copies and archived logs?

Creating a backup and recovery strategy also requires an understanding of the features provided by the product to support backup and recovery. Subsequent sections cover:

- Methods of recovery. DB2 UDB supports crash recovery, restore (or version recovery), and roll-forward recovery.
- Logging. DB2 UDB supports two basic types of logging: circular logging and log retention logging.
- Online versus offline backup.
- Managing logs and backups.

Some issues concerning backup and recovery are best considered early in the planning process.

- If the recovery plan requires use of logs, mirroring of the logs is highly recommended. A disk failure causing loss of one or more logs could prevent meeting recovery objectives.
- If ADSM is to be used to store backups and archived logs and multiple database instances are to be run on the same host, do not use the same database alias in each instance. ADSM uses the database alias name as part of the filename of each object. ADSM cannot distinguish objects for one instance from objects in the other instance.
- For a database to be online, DB2 UDB EEE requires that its catalog partition be available. Consequently, we highly recommend that you configure the catalog partition to tolerate a disk failure. Allocate both the catalog partition's log files and the catalog tablespace on mirrored or RAID-5 disks.

3.2 Recovery Methods

DB2 UDB supports three methods of recovery: crash recovery, version recovery (also called "restore recovery"), and roll-forward recovery.

3.2.1 Crash Recovery

Transactions (more accurately, units of work) against the database can be interrupted unexpectedly. For example, if power fails before all of the changes that are part of a logical unit of work are committed, the database becomes inconsistent and unusable. Crash recovery returns the database to a consistent, usable state.

3.2.2 Version (or Restore) Recovery

This recovery method requires loading a backup copy of the database or table space(s). The database will be restored to exactly the same state that it was in when it was backed up.

3.2.3 Roll-Forward Recovery

This recovery method also requires loading a backup copy of the database or table space(s) and then applying log records to recover changes made after the backup image was created. This method provides recovery from media, hardware, operational, and software failures. The recovery can be to a point in time or to the last committed unit of work.

If a multipartition tablespace is to be recovered to a point in time, then the load of the backup image and the application of the logs must be performed on all partitions where the tablespace is defined.

Roll-forward recovery requires enabling "log retention." Retention logging is further discussed in the next section.

3.3 Logging

Transaction logging records each change to a database to permit recovery of the database to a consistent state. As data in the database buffer pool is modified, log records reflecting these changes are written into a log buffer. At commit time, all of the log records reflecting changes to the database during this unit of work MUST be written to the log files on disk. Once the log records are successfully stored to disk, recoverability can be guaranteed (at least as long as the log files are available).

After commit, the only certainty is that the log records are written to disk. The modified data in the buffer pool is not written to the database files until some later time. But even if a crash occurs before the database is updated, the log records stored on disk contain all the information needed to rebuild the committed changes.

If you desire a fully recoverable database, allocate the log files on a fault-tolerant disk complex: mirrored, duplexed, or RAID-5 for example. Imagine a situation where your log files are allocated on a disk. The disk is not a member of a RAID configuration. The log files are not mirrored to a separate disk. There is no backup copy of the log file. If you now lose this disk, you cannot recover the database up to the current unit of work. Make an immediate backup of your database!

DB2 UDB supports two basic types of logging: circular logging and archival logging. Archival logging is also referred to as log retention logging

In a partitioned database environment, log files must be allocated for each database partition.

3.3.1 Circular Logging

Circular logging initially uses a specified number of primary log files. The log files reflect changes by in-process transactions. The logs are used sequentially. A log file cannot be reused until all units of work contained within it are either committed or rolled back AND the committed changes are written to the disks used by the database.

If the database manager requests the next log in sequence and that log is not available for reuse, a secondary log file will be allocated. After the secondary log fills, the database manager again checks if the next sequential primary log is available for reuse. If the primary log is still unavailable, another secondary log file is allocated. This process continues until either the primary log file becomes available for reuse or until the number of secondary logs permitted for allocation is exceeded.

If DB2 UDB cannot continue logging due to a log full condition, the database manager will halt.

Circular logging supports crash and version/restore recovery. It does not support roll-forward recovery. For more information on the recovery methods (crash, version/restore, and roll-forward) see “Recovery Methods” on page 50.

3.3.2 Archival (or Log Retention) Logging

The second type of logging supported by DB2 UDB is archival logging. Archival logging is also called log retention logging. This type of logging is activated in UDB by setting the `logretain` parameter in the database configuration to `ON`. And don’t forget, each database partition has its own set of database configuration parameters!

With archival logging, when a log file fills, the database manager allocates another. The database administrator normally configures several primary log files so that the next log file can be allocated before it is needed. The database configuration parameter `logprimary` specifies how many primary log files are allocated when the database is created. With archival logging, the database manager will allocate no more log files than the total of the primary and secondary database configuration parameters (`logprimary` plus `logsecond`). Once that total has been allocated, the need for more logging space will cause the database to halt with a log full condition. Allocate a sufficient number of adequately sized log files to handle the workload. Beware the application that tries to process too large a unit of work. Consider, too, that in-doubt transactions can prevent the freeing of a log file. Be sure to check for in-doubt transactions following a database recovery action. Quickly resolve each that exists to free its log space (and any locks on the database it may hold).

At any point in time, a log file associated with log retention logging will be in one of three states:

1. Active. These logs contain records for units of work which have not yet committed (or rolled back). Active logs also contain information for

transactions which have committed but have not yet had the changes written from the buffer pool in memory to the database files on disk.

Active log files are used for crash recovery.

2. Online Archive. These logs contain information for completed transactions which no longer require protection from crash recovery. All changes in the log have been written to the database files. They are called online because these logs still reside in the same subdirectory (LOGPATH) as the active logs.
3. Offline Archive. The log files have been moved out of the active log file subdirectory. A manual process or a process invoked through a user exit can be used to move the files.

When archival logging is used, the database manager will truncate and close the current active log when the last application disconnects from the database. This is a positive feature if the database is to be inactive for some period of time. But for a low-activity database where there are short periods when no application will be connected to the database, the overhead of continuously truncating the last active log and then reallocating the primary log files when a new application connects can be costly. In such situations, the DBA should consider the `ACTIVATE DATABASE` command. This command will keep the database active and log file truncation will not occur. But if log files are not mirrored (or otherwise protected from a disk failure), the administrator must evaluate the impact on recovery. To lose the log through disk failure will make the point of recovery for the database to be less than the current unit of work. The `DEACTIVATE DATABASE` command can be entered to allow log file truncation to occur for databases on which the `ACTIVATE DATABASE` command has been used.

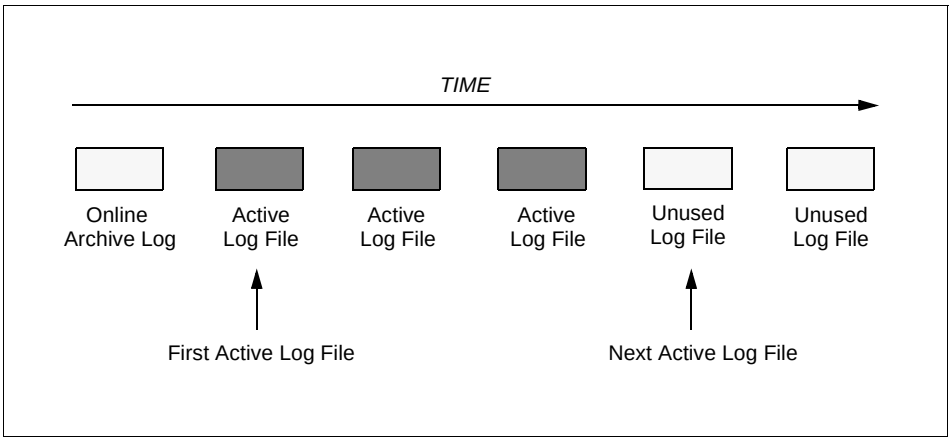


Figure 8. Online, Active and Unused Log Files

The database manager uses a control file to track the status of the log files. The control file identifies the active log with the lowest name, the oldest active log. As shown in Figure 8, this log file is called the *First active log file*. The control file also specifies the name of the next log file to be used. Appropriately enough, this log is called *Next active log file*.

The values for first active log file and next active log file can be displayed by using the following command:

```
db2_all ';db2 get database configuration for TPCD100' | grep "active
log"
```

where `TPCD100` is our database alias. Log files that are older than the first active log file are known as archive log files. They are not required for crash recovery and can be moved to different media.

By using the values of nextactive and firstactive, the database administrator can determine the number of active logs that are currently allocated. If this number is abnormally large, an application may not be committing on a timely basis. If the number of allocated active logs is close to the total of primary and secondary logs, the installation may be approaching a log full condition. And when a log full condition occurs, the database will hang.

The log file naming convention starts at S0000000.LOG and sequentially names logs until S9999999.LOG is reached. After S9999999.LOG the name wraps back to S0000000.LOG, and the sequence starts again. The higher log file numbers will only be used when archive logging is configured for the

database. For circular logging, only the numbers corresponding to the actual number of primary and secondary circular logs will be used.

3.4 Recovery History File

The recovery history file resides in the same directory as the database configuration file. There is a file for each database partition. The database manager updates the recovery history file whenever one of the following operations are performed:

- Backup a database or table space
- Restore a database or table space
- Load a table
- Quiesce a table space
- Roll forward a database or table space

When a recovery action is required, the summary information contained in the file can help the database administrator to create the recovery plan. Information that may be useful includes:

- The part of the database that has been copied by a backup, load, or copy operation.
- When the database was copied
- Where the copy is located
- Time of the last restore
- Quiesce entry with local and CUT timestamps provided
- Table space point-in-time recover information

DB2 UDB provides two commands to extract information from the recovery history file: `LIST HISTORY` and `LIST BACKUP`. The commands differ in the amount of information provided. `LIST BACKUP` retrieves only information pertaining to backups and restores. `LIST HISTORY` retrieves the full spectrum of records. The database administrator can control how long history file information is kept before it is purged. The database configuration parameter `REC_HIS_RETN` can be set to specify the retention period for history file records. The database software automatically prunes records older than the value of `REC_HIS_RETN`, which by default is 366 days. Records can also be removed manually using the `PRUNE HISTORY` command. The timestamp may be abbreviated to as little as yyyy (4-digit year). All entries with timestamps equal to or less than the timestamp provided are deleted from the file. Using the `WITH FORCE` option specifies that entries will be pruned according to the specified timestamp

even if the command causes entries from the most recent restore set to be deleted.

3.5 Choosing A Backup Strategy

Defining a robust backup strategy for your database requires careful consideration of your business requirements, including factors such as the requirements for database availability and the maximum time the database can be down for backups. The Backup Database SmartGuide (included in the DB2 UDB V5 Control Center on Intel platforms), helps you with the decision-making. This section describes the considerations and issues in more detail in order to provide you with a better understanding of this important area.

To begin, you must decide on the types of failure that you are protecting against. The purpose of running a backup is to be able to use it to perform a recovery. Therefore, you must test your processes and procedures to ensure that your backups will be useful in the event of a failure. Backups that cannot be used to restore data are pointless.

The following are general guidelines for planning a recovery strategy:

- The type of data contained in your database is relevant. Databases that contain read-only data do not need to be protected through archive logging. Off-line backups can be run following each new data load activity. The use of circular logging would be sufficient in this case.
- With continuously updated data that is deemed important to your business, you must use archive logging.
- If your database must be continuously available, you must take online backups. This requires the use of archive logs.
- If, in the event of a failure, your database must be recovered in a short time, you will need to run more frequent backups. In this case, you need to establish how long it would take to recover from a failure (the sum of the time to restore the database from a backup plus the time needed to roll the log forward).

Beyond transaction failures and system crashes, both of which DB2 will recover from, you should consider application errors. This refers to the general case of data being damaged in some manner. Clearly, there is no way to prevent an authorized user from altering data inappropriately. The best strategy for dealing with this type of problem is to ensure that you have archive logs to roll forward the database to the point just prior to the corruption of the data. Be sure to factor into your schedule the maximum time

the database can be down for backups, and whether these are online or offline.

Probably the most common type of failure is caused by media problems. This is not limited to disk problems, but can extend to other I/O devices, including disk controllers and tape devices. As a starting point, it is suggested that you do not back up your database to the same disk on which the production version exists: use either a separate disk or external media. The handling of your logs should be similar: Consider directing these to a separate physical disk from that of the database. In addition to protecting against a disk failure affecting both, this may also result in performance improvements.

Though unlikely, it is possible that your backup media could suffer a problem just when it is needed to enable you to recover from a disk failure. Consider the impact of a tape becoming unusable. If your data is absolutely critical, you should consider having duplicate tape media. Another strategy is to minimize the potential for impact caused by a failed disk. This applies to the disks that both the database and logs reside upon. Using disk arrays for your database volumes or logs (or both) is perhaps the best defense against disk media failures. See the *DB2 UDB V5 Administration Guide* for information on disk arrays. If you extend redundancy to disk controllers as well, it is highly unlikely that your database will ever be unavailable or that logs will be lost due to a media failure.

When a database is first connected to, or activated, the database manager formats the primary logs for that database. The more logs to be formatted, the more time DB2 UDB requires to initialize. To minimize startup time, specify no more primary logs than needed to support crash recovery. Use secondary logs to handle the rare times when many logs are required. Let's say DB2 UDB starts logging with file S0000000.LOG and after some time the log fills. DB2 UDB allocates the next log and calls the user exit, *db2uext2*, to copy the full log to ADSM storage. The full log, though copied, is still active. The full log will remain in active status until every unit of work having changes in that log commits and until every logged (and now committed) change is written to the database disk. Once all the changes are committed and written to the database disk, the log becomes inactive. It is no longer required for crash recovery. At this time, the now inactive log is renamed. It becomes the highest numbered allocated log.

For example, say `logprimary` equals 5. When a connect or activate is issued, DB2 UDB will allocate five primary logs. Assume the log file names are S0000017.LOG through S0000021.LOG. UDB starts recording changes in S0000017.LOG. Eventually, the logs fills. UDB starts using the next log, S0000018.LOG, and calls `db2uext2` to archive S0000017.LOG to ADSM

storage. Eventually, all the changes reflected in S0000017.LOG will be committed or rolled back. Later yet, all the changes will finally be written to the database disk files. At this time, DB2 UDB no longer requires S0000017.LOG to support crash recovery. S0000017.LOG is renamed to S0000022.LOG.

Now what if a database partition must be restored? Or worse, what if an application error requires that the entire database be restored and rolled forward to a point just before the errant application ran? Not only must all of the database backups be restored from tape, but the necessary logs must be retrieved as well. This may involve many tape mounts. Bear in mind also that even query-only database systems do their share of logging.

Wouldn't it be nice to have on disk all of the logs needed to roll forward from the most recent backup? To be able to avoid waiting for log files to be recalled from ADSM tape storage? Wouldn't it be nice if DB2 UDB wasn't so quick to rename those inactive log file too allow some number of log files to remain in the LOGPATH directory? Maybe just enough to roll forward from the most recent backup?

A possible solution to this log-on-tape dilemma is to use the sample exit, which writes logs to an alternate directory instead of to ADSM. A daemon program could be written to archive the older logs to ADSM. Also you would need to update the database configuration parameter to specify this disk directory as an alternate log path.

3.6 Introducing ADSTAR Distributed Storage Manager (ADSM)

ADSTAR Distributed Storage Manager (ADSM) is an enterprisewide storage management application. ADSM is a client/server application that provides automated storage management services to multivendor workstations, personal computers, and local area network (LAN) file servers. This section aims to introduce the components and terminology of ADSM. A later section of this chapter describes how to set up and use ADSM for database backup and recovery.

ADSM includes the following components:

- Server

  The server provides backup, archive, and space management services. It maintains a database and a recovery log. The database contains information about ADSM resources and users and points to the location of user data that is stored by ADSM. ADSM stores user data in storage pools

(defined below), not in the ADSM database. ADSM records changes to its database in the recovery log.

- Administrative client

  This component allows ADSM administrators to control and monitor server activities, to define how stored user data is to be managed, and to set up schedules to provide services at regular intervals. The administrative client will be used in a later section that explains how to install and set up ADSM for use by DB2 UDB EEE.

- Backup-archive client

  This component allows users to maintain backup versions of their workstation files, which they can then restore if the original files are lost or damaged. Users can also archive files for long-term storage and retrieve the archived files when necessary.

- Hierarchical storage management (HSM) client

  ADSM users can free workstation storage space by migrating less frequently used files to ADSM server storage. A user accesses a migrated file just as if it were an ordinary file. The migrated file is automatically recalled from ADSM server storage and made available to the user.

- Application programming interface (API)

  Application programs can use defined program calls to request ADSM services like backup, restore, archive, and retrieve. When ADSM is used to store its backups and archived log files, UDB requests the services through the API.

The ADSM server supports two methods for storing user data: backup and archive. The purpose of backup is to guard against loss of information. Multiple copies (called *versions*) of user data files can be stored by the ADSM server. The process to get a backup version from ADSM storage is called *restore*. The restored backup version remains in ADSM storage; it is not deleted. ADSM provides two types of backup:

- Incremental backup copies all files that are new or that have changed since the last incremental backup.
- Selective backup copies only specific files named by the user.

The archive method preserves files for long-term storage. Unlike backup, archive does not support versions. The process to return an archived file from ADSM storage to the client node is called *retrieve*. The archived file remains in ADSM storage.

So, an ADSM backup version is restored; an archived file is retrieved.

DB2 UDB EEE can use both ADSM processes: backup/restore and the archive/retrieve. When the `USE ADSM` clause is included in the DB2 UDB command to backup (or restore) a database or tablespace, the ADSM backup (restore) function is used. The ADSM version of the db2uext2 user exit uses the ADSM archive and retrieve functions to handle copied database log files.

To store user data, the ADSM server requires two things: where to put the copied data and how to manage those copies.

During the example configuration detailed in “Installing and Configuring ADSM and DB2 UDB EEE” on page 63, the following ADSM objects are defined:

- Library. An administrator-defined collection of one or more drives that share similar media mounting requirements.
- Device class. Each device is associated with an ADSM device class. A device class contains information about the device type and the way the device manages its media.
- Storage pool. A named collection of storage volumes that are associated with one device class.
- Backup and Archive copy groups. Where you specify parameters that control the generation and expiration of backup and archive data.
- Management class. Associates backup and archive groups with files and specifies if and how client node files are migrated to storage pools.
- Policy set. Specifies the management classes that are available to groups of users. Policy sets contain one or more management classes consisting of a default management class and any number of additional management classes.
- Policy domain. Lets an administrator group client nodes by the policies that govern their files and by the administrators who manage their policies. A policy domain contains one or more policy sets, but only one policy set (named ACTIVE) can be active at a time. ADSM uses the active policy set to manage files for client nodes assigned to a policy domain.
- Include/exclude list. This allows users to:
  - Exclude files or directories from backup operations
  - Include any previously excluded files
  - Bind a file to a specific management class

After all these ADSM objects have been defined, there is an overview diagram which shows how the relationships between the definitions in “Overview of Backing up a Database Using ADSM” on page 102.

For additional information about ADSM, consult the ADSM product manuals or visit the ADSM home page at:

`http://www.storage.ibm.com/software/adsm/adsmhome.htm`

3.7 Planning for ADSM

To develop a hardware configuration for ADSM, a number of items must be evaluated. The following is by no means an exhaustive list:

- How many tape transports? The number of drives required is mostly a function of the amount of data and the duration of your backup window. The number of concurrent backup tasks directed to an ADSM server should not exceed the number of drives on that server. If there exist more tasks than drives, the extra processes will hang in a *media wait* condition. If the wait is longer than a configurable time, the waiting process is cancelled. Perhaps the amount of data on a single partition is so large that multiple sessions must be used to complete the backup in the time required. The number of concurrent backup sessions should be no more than the number of available drives. You might consider spreading your backups over several days. Say, backup one-third of your database partitions on one day, the second third on the next day, and the following third on the day after. To support ADSM tasks like reclaiming tapes and copying storage pools, consider no fewer than two tape drives.
- How many adapters are needed? How many tape drives can be attached to an adapter and still provide reasonable performance?
- Will backup ADSM server(s) be needed in case the server host crashes? The failover host will need adapters to at least take over the tape and disk(s) storing the ADSM logs, database, and storage pools. To provide a non-TCP/IP heartbeat path for HACMP, additional serial adapters be called for. These considerations lead to the next question.
- On what kind of node(s) will the ADSM server(s) run? Most likely your choice for an ADSM node will be at least a wide node, perhaps even larger. The node must provide a sufficient number of slots to contain the adapters. Consider dedicating a host to the ADSM server. Avoid running the server function on one of your database hosts. Remember that the response of a DB2 UDB EEE database is as fast as the slowest database partition (or host). To give processing cycles to ADSM requires taking cycles from the database server. Increased contention for the processor(s)

leads to more waiting. And the additional wait time will slow down the speed at which the database server on the partition can do its work and reply back to the coordinator node.

- Where will the ADSM server run? If the server is located within the RS/6000 SP complex, the clients can send their files to the ADSM server across the SP switch. Network bottlenecks are less likely.
- Will tape drives, if used, be installed in an automated library? Using an automated library like the 3575 Tape Dataserver for small complexes or the 3494 Dataserver for larger installations will minimize the need for personnel to perform the tasks of mounting and dismounting tapes. The library will likely be more responsive to tape mount requests and will work 24 hours a day every day of the year.
- So, what about recovery? As discussed above, the hardware plan must allow backups to complete in the allotted time. And while a backup of all the database partitions might be spread across several days, you probably cannot get away with that in a recovery situation. What if an application error necessitates recovering the entire database to the time just before the errant program started? How does an unavailable database impact the business? This and other factors determine how quickly the database must be restored to operation.
- Another factor to evaluate is how the backup images are stored on tape. For example, what is the likelihood that you will try to restore two different database partitions at the same time only to discover that both backup images are on the same tape? The question applies mostly to installations running multiple database partitions on the same host. Let's say you configure four database partitions on each of your high nodes. ADSM provides a feature called collocation, which if enabled on a storage pool, makes ADSM keep each client's files on a minimal number of volumes within the storage pool. This feature can be enabled for hosts or for file systems. But here's the problem. Host collocation may not work because you have four database partitions on the same host. Collocation by file space also fails to guarantee separation because the entire set of backups (today, yesterday's, last week's, node01's, node23's) are all part of the same file space (the database alias name translated to uppercase). Perhaps you might backup all the partitions on the same host at the same time. Just be sure that you have enough drives. If you run four simultaneous backups each using a single ADSM session, ideally four drives should be available. But even this plan could be undone when tape reclamation runs. The reclamation process consolidates the active, unexpired data on many volumes onto fewer volumes. ADSM will just consolidate those mostly empty tapes and fill new tapes. And the current

backups for partition 1 and for partition 2 are now perhaps on the same tape! Another option is to assign each partition to a separate backup storage pool. Or, using host collocation, assign the first partition on each host to one storage pool, the second logical partition to another, and so on.

3.8 Installing and Configuring ADSM and DB2 UDB EEE

This section goes through the steps necessary to configure ADSM and DB2 UDB EEE so that DB2 backups will use ADSM storage and full log files will be copied to ADSM storage.

3.8.1 Hardware Configuration

Throughout the steps, the same example will be referenced, namely a four high-node RS/6000 SP with a high sped switch (HPS). Each SP node has 64 GB of external SSA disk. Four 3590 tape libraries are attached to the first SP node. The SP nodes have hostnames tp3an01, tp3an05, tp3an09, and tp3an13. The Control Workstation's hostname is sp-tp3cw.

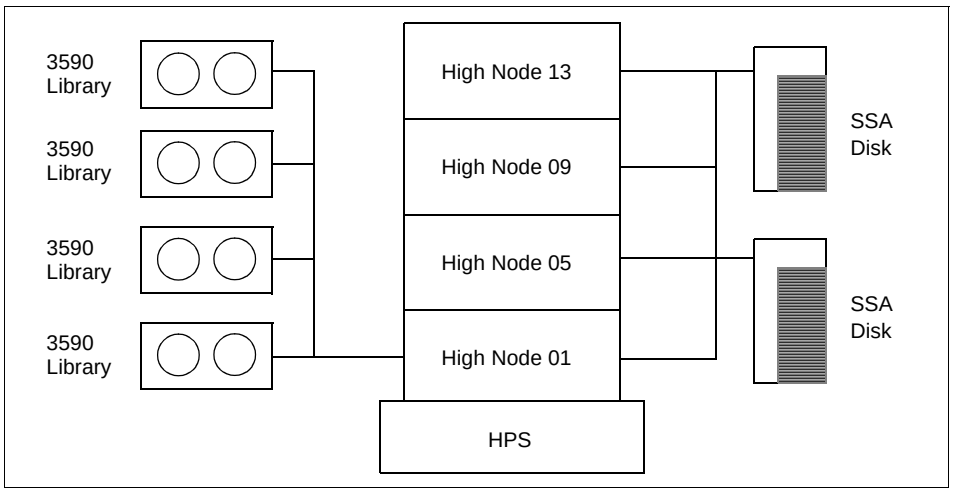


Figure 9. Hardware Configuration

3.8.2 Software Used

The four high nodes have the following software installed:

- AIX 4.2.1
- PSSP 3.1

- DB2 UDB EEE V5.0.0.1
- ADSM Client for AIX V2.1.20.7

The first high node (tp3an01) has in addition:

- ADSM Server for AIX V3.1

The DB2 UDB EEE database referenced is a 30 GB TPCD database.

Advice will be given where appropriate for configurations that differ from the configuration used in our tests.

3.9 ADSM Server and Client Installation

The following steps describe a new install of ADSM Version 3. If migrating from an earlier version, refer to the *ADSM V3 for AIX Quick Start* manual.

Having determined the number and location of the ADSM server(s), the ADSM Server product must be installed on the chosen host(s). For this project, one ADSM server is sufficient. It will be installed on the first node in the RS/6000 SP complex. The ADSM client software will be installed on all SP nodes and on the Control Workstation. For our example, the 3590 tape drives require the Atape.driver device driver. If using the 3494 tape drive, you will require (in addition to Atape.driver) the atldd.driver product.

One of two basic methods can be used to install the software: the `installp` command or `smitty`. The ADSM software images were copied to an install directory, /usr/sys/software. The /usr/sys/software directory was exported using NFS in read-only mode to all SP nodes.

3.9.1 Install the ADSM Server Software

For this project, we installed ADSM Server Version 3, PTF U452223. The PTF brings the adsm.server.rte fileset to level 3.1.0.1.

```
# dsh -w tp3an01 mount sp-tp3cw:/usr/sys/software /mnt

# dsh -w tp3an01 'installp -acgNqX -d /mnt adsm.server \
       adsm.devices adsm.license 2>&1 \
       | tee /tmp/install.adsmserv.$$' | dshbak | more

# dsh -w tp3an01 umount /mnt
```

Notice the single quotes surrounding the `installp` and `tee` command sent to node 1. This prevents the shell on our host from interpolating variable `$$` and

from implementing the redirection of STDERR and the pipe from `installp` to tee. If you wish to preview, but not actually perform, the software installation, add the `-p` flag to the `installp` command above.

After a successful installation, a list of installed products can be obtained by using the `lslpp` command at the ADSM server SP node (tp3an01):

```
[tp3an01][/]> lslpp -L 'adsm*'

Fileset                      Level  State  Description
----------------------------------------------------------------------------
adsm.devices.rte           3.1.0.0    C    ADSM Device Support runtime
adsm.license.rte           3.1.0.0    C    ADSM Server License Registration
adsm.server.gif            3.1.0.0    C    ADSM Server Web Administrator
                                           Icons
adsm.server.rte            3.1.0.1    C    ADSM Server Runtime
adsm.server.util           3.1.0.0    C    ADSM Server Utilities
```

3.9.2 Install the ADSM Client Software

We installed the ADSM Version 2 client code on all SP nodes and on the Control Workstation, sp-tp3cw.

ADSM Version 3 Client and DB2 UDB

ADSM Version 2 Client was used in our tests because at the time of writing, DB2 UDB did not support the ADSM Version 3 Client. The support of ADSM Version 3 Client by DB2 UDB is planned for June 1998. We did try to use the Version 3 Client and encountered problems with offline backups and the `db2adutl` command.

The /usr/sys/software directory contains the install images for the ADSM client. Using the `installp` command, we install all hosts at the same time. (As with the ADSM server install above, notice the use of single quotes.) To preview, add the `-p` flag to the `installp` command below.

```
# dsh -a -w sp-tp3cw mount sp-tp3cw:/usr/sys/software /mnt

# dsh -a -w sp-tp3cw 'installp -acgNqX -d /mnt \
      adsm.client 2>&1 | tee /tmp/install.adsmcli.$$' \
      | dshbak| more

# dsh -a -w sp-tp3cw umount /mnt
```

After a successful installation, a list of installed products can be obtained by using the `lslpp` command at a client SP node (in this case tp3an05):

```
[tp3an05][/]> lslpp -L 'adsm*'

Fileset                      Level  State  Description
----------------------------------------------------------------------------
adsm.client.admin         2.1.20.7    C    ADSM Client - Administrative GUI
adsm.client.api           2.1.20.7    C    ADSM Client - Application
                                           Programming Interface
adsm.client.base          2.1.20.7    C    ADSM Client - Backup/Archive
adsm.client.common        2.1.20.7    C    ADSM Client - Common Files
adsm.web.client           2.1.20.7    C    ADSM Client - WebClient
```

It is recommended to check that the device driver for the tape drives are installed. For example, IBM 3590 Magstar Tape (which we used) and IBM 3570 Magstar MP Tape require Atape.driver. The automated IBM 3494 Magstar Tape Library requires the atldd.driver product. (The IBM 3575 Magstar MP Tape Library Dataserver does not require the atldd.driver product.)

3.10 ADSM Server Configuration

This section describes how to set up and customize Version 3 of ADSM server. To integrate DB2 UDB EEE into existing ADSM installations, only a few of the steps apply. Specifically, your ADSM administrator should consider the modification of the current policy domain to add a management class for UDB backup and log archive files. Additionally, insure that the storage volumes are sufficient to handle the increased requirement. More tape volumes may need to be formatted and inserted into the library. Automated schedules may require adjustment.

The tasks outlined below and in the following section on ADSM client customization will serve as a guide for those installations which are setting up ADSM for the first time. The following sequence of steps describes the setup and customization of the ADSM server:

- Allocate and format the ADSM log and database files.
- Create a set of customized server options, dsmserv.opt.
- Register and grant authority to ADSM administrators.
- Revoke system authority from SERVER_CONSOLE ID.
- Register ADSM licenses.
- Define libraries, devices and devices classes.
- Define storage pools.

- Establish storage management policy (domain, policy set, and management class definitions).
- Register client nodes.
- Set up schedules to automate client and server operations.
- Initialize storage pool volumes.
- Automate the start of the ADSM server and/or the client scheduler when a node is booted.
- Implement procedures to manage the ADSM environment.

3.10.1 Allocate ADSM Database and Log

First, let's discuss some considerations to bear in mind when creating the ADSM database and log files.

3.10.1.1 ADSM Database and Log Configuration

In addition to storing server configuration data, the ADSM database contains information about (and pointers to) all versions of a client node's backup and archive files which reside in the storage pools. When a client requests that a specific version of a backup file be restored, the ADSM server consults its database to locate the requested file in the storage pool. The ADSM recovery log reflects changes to the database. Without the recovery log, all changes to the ADSM database since the last backup are lost. And if you lose your ADSM database, you lose all the backup/archive data of all the clients. To provide greatest availability and protection, mirror both the database and the recovery logs used by ADSM. For resource constrained systems, at least try to mirror the ADSM log. Also, enable roll-forward recovery mode. This mode allows the database to be recovered to its most current state or the point at which the database was lost.

The normal rules for mirroring apply. Put the file and its mirror on different disks. Attach the mirroring disk to a different adapter and to a different power circuit than the mirrored disk.

If you cannot tolerate the unavailability of the ADSM server for a prolonged node failure, consider using HACMP to switch the server functions to a backup node.

ADSM supports the use of either journaled file system (JFS) files or raw logical volumes for the database, recovery log, and disk storage pools. Either ADSM or AIX can provide the mirroring function. (Note: If you decide to use raw logical volumes, be sure to use ADSM, not AIX, mirroring. AIX overwrites ADSM control information when raw logical volumes are mirrored.)

3.10.1.2 Sizing the ADSM Database and Log Files

The *ADSM V3 Administrator's Guide* details how to calculate the sizes for the database, for the recovery log, and for disk storage pools. The current setup plans to use roll-forward logging. For this mode, the guide recommends an initial log size of 25 MB. Disk storage pools will not be used. The allocation size of the ADSM database was computed as follows:

To determine the size of the ADSM database, you must estimate the total number of client files to be stored, the number of versions of each file, the amount of database storage needed to describe each object, and factors to account for overhead, growth, and a margin of safety.

These are the steps we took:

1. Determine the number of files on the clients. The following command was run on the Control Workstation and on the four SP nodes.

   ```
   # find / -fstype jfs | wc -l
   ```

   This resulted in:

   ```
   42,000 files on the control workstation
   25,000 files on each SP node
   142,000 total files
   ```

2. Three backup and/or archive versions of each file will be stored.
3. Each stored object requires six hundred (600) bytes of database storage.
4. The *ADSM Administration Guide* suggests 50 percent additional space for overhead.
5. The size in bytes is the product of the total number of files (142,000), the number of versions per file (3) and the number of bytes per entry (600). Calculate the size:

   ```
   (142,000 * 3 * 600) * 1.50 = 383400000 bytes
   = 383.4 MB
   ```

   To allow for a safety factor, we allocated 501 MB to the ADSM database.

Why 501 MB and not 500? ADSM subtracts one (1) MB from the allocated space for overhead. The remaining space is divided into 4 MB partitions. If any partition is less than 4 MB, it is not used. So, had the database been allocated with 500 MB, ADSM would have taken 1 MB for overhead. The remaining 499 MB would be divided into one hundred twenty-four 4 MB partitions and 3 MB of unused space. With an allocation of 501 MB, ADSM removes 1 MB for overhead and divides the remaining 500 MB into one hundred twenty-five 4 MB partitions and no unused space.

ADSM and AIX Partition Sizes

Do not confuse ADSM partition size (always 4 MB) with AIX partition size, which varies. For this example, it is 8 MB.

To summarize, the ADSM log is to be 25 MB, the database 501 MB. Both log and database will be mirrored. Sometime in the future, the ADSM server may be made highly available by using HACMP to switch the server to a backup host. To prepare for that possibility, a new mirrored file system will be created. You may even wish to create a separate volume group consisting of disks dedicated to ADSM. Since our current system is disk-constrained, we will use an existing volume group. In addition to the ADSM log and database, the new file system will also contain a directory for configuration macros, ADSM server options file (dsmserv.opt), and an ADSM start script.

3.10.1.3 Creating the File System for ADSM Database and Log

Since the log and database require a total of 526 MB, an allocation of sixty-six 8 MB logical partitions (528 MB) would seem to be sufficient. Surely, the remaining 2 MB would be enough to contain the inodes and the planned configuration directory and files. In fact, if the file system is allocated with the default which creates one inode for every 4096 bytes of file system storage (nbpi=4096), you cannot even allocate the database and the log! By changing the number of bytes per inode to 131072 (which also requires changing the aggregation value to 64 (ag=64)) both files can be allocated. There's even enough space, barely, to store the planned configuration directory and files. For the curious, the impact of different nbpi values is summarized below:

Using the default file system creation values of ag=8 and nbpi=4096:

```
> df -k /tmp/test_fs

Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/test_lv       540672    523652    4%       17     1% /tmp/test_fs
```

The result is that not enough space remains to allocate both the recovery log and the database. Free space is a little less than 524 MB; however, the two files together require 526 MB.

Using creation values ag=64 and nbpi=131072:

```
> df -k /tmp/test_fs
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/test_lv       540672    540004    1%       17     1% /tmp/test_fs
```

Now both files can be allocated and enough space will be left over to contain the ADSM configuration source files.

So, a sixty-six partition logical volume will be created for the ADSM files, where each partition is 8 MB. The file system will be created using the higher (non-default) values for nbpi and ag. As mentioned above, you may wish to create a separate volume group to contain the ADSM files. So, the first three commands below (`mkvg` to create an ADSM volume group, `mklv` to allocate a JFS log and `logform` to format the JFS log) are included as optional commands. Our exercise, however, does not require a separate volume group.

These are the commands you might use to create the volume group, allocate a jfslog and format the jfslog:

```
# mkvg -f -y vg_adsm -s 8 hdisk9 hdisk10

# mklv -y lv_adsm_log01 -t jfslog -a c -c 2 \
       vg_adsm 1 hdisk9 hdisk10

# yes | /usr/sbin/logform /dev/lv_adsm_log01
```

Note that the JFS log will be one partition allocated in the center of one disk. The JFS log mirror will be one partition in the center of the second disk.

To set up and mount the file system to contain the ADSM files for this example:

```
# mklv -y lv_n01_01_115 -t jfs -a e -c 2 vg_n01_01 66 \
       hdisk13 hdisk39

# crfs -v jfs -d lv_n01_01_115 -m /adsmserv/tp3an01 \
       -A yes -p rw -t no -a frag=4096 -a nbpi=131072 -a ag=64

# mount /adsmserv/tp3an01
```

To allocate the ADSM recovery log and database (the `-m` flag indicates that the sizes (25 and 501) are in megabytes):

```
# dsmfmt -m -log /adsmserv/tp3an01/log.1 25

# dsmfmt -m -db  /adsmserv/tp3an01/db.1  501
```

In the ADSM file system (/adsmserv/tp3an01) created above, we made an additional subdirectory to contain customization source files: the server options file, setup macros, the ADSM startup script, and more.

```
# mkdir -p /adsmserv/tp3an01/config
```

3.10.2 Format the ADSM Database and Log

The ADSM server locates its recovery log and its database(s) by referencing the dsmserv.dsk file. Whenever a new ADSM log and/or database file is formatted, the file dsmserv.dsk is created in the current working directory unless it already exists. If dsmserv.dsk does exist, it is modified when the format program runs.

For our example, the ADSM server program was started from our custom directory, /adsmserv/tp3an01/config.

To format the ADSM log and database and create the dsmserv.dsk file:

```
# cd /adsmserv/tp3an01/config
# dsmserv format 1 /adsmserv/tp3an01/log.1 \
                 1 /adsmserv/tp3an01/db.1
```

The number "1" before each file name specifies that one ADSM log file and one database file will be formatted. The `dsmserv format` command creates a file, /adsmserv/tp3an01/config/dsmserv.dsk, which identifies the recovery log and database files and contains:

```
/adsmserv/tp3an01/log.1
/adsmserv/tp3an01/db.1
```

3.10.3 Customize ADSM Server Options

During startup, the ADSM server requires an options file, dsmserv.opt. The ADSM server program product provides an annotated sample server options file, /usr/lpp/adsmserv/bin/dsmserv.opt.smp. Using the sample file as a guide,

we customized a server options file for our installation. The configuration file, /adsmserv/tp3an01/config/dsmserv.opt, created for this exercise follows:

```
*  ============================================
*  File:  dsmserv.opt
*  Platform: AIX
*
*  Customized ADSM V3.1 server options file
*
*  ============================================
COMMmethod         SHAREDMEM
SHMPort            1510
COMMmethod         TCPIP
TCPPort            1500
TCPWindowsize      640
TCPNODELAY         YES
ENABLE3590LIBrary  Yes
MAXSessions        25
BUFPoolsize        131072
LOGPoolsize        2048
TXNGroupmax        256
COMMTimeout        6000
IDLETimeout        15
VOLUMEHistory      /adsmserv/tp3an01/config/volumehistory1
DEVCONFig          /adsmserv/tp3an01/config/devconfig1
MOVEBatchsize      1000
MOVESizethresh     500
USELARGebuffers    Yes
```

Most of the above parameters aim to optimize performance. In general, ADSM will complete a process faster when it can work with larger blocks of data.

- We must have a `COMMMETHOD` parameter entry set to TCPIP as we will be using the SP switch.
- The `TCPPORT` parameter specifies the TCP/IP port to be used by the ADSM server to wait for requests. You must ensure that this port is neither already defined in /etc/services nor in use by another program.
- The `TCPWINDOWSIZE` parameter, set to 640 KB, specifies the size of the TCP/IP sliding window for the ADSM clients. This can vary from 0 to 2048 KB. Larger window sizes may improve performance while using more memory.
- The `MAXSESSIONS` parameter specifies the maximum number of simultaneous client sessions. The default value, and the value coded above, is 25 client sessions. We need at least four (one per SP node). Your installation may require a greater number. The maximum value is limited only by available virtual memory size or communication resources.

- The ENABLE3590LIBRARY parameter is only needed when 3590 Tape Drives are installed in an IBM 3494 Dataserver.
- The COMMTIMEOUT parameter defines the amount of time that the ADSM server waits during a ADSM database update, or for an expected message from a ADSM client, before terminating the session with the client. This is set to 6000 seconds as recommended in the *DB2 UDB Administration Guide*.

For a full description of all the parameters, see the *ADSM for AIX V3.1 Administrator's Reference,* GC35-0275-00.

ADSM Online Manuals

A full set of ADSM online manuals can also be found at:

`http://www.storage.ibm.com/software/adsm/pubs`

3.10.4 Start the ADSM Server

The ADSM server can be started using one of several methods: the ADSM GUI, AIX commands or by using a script. A entry can be added to /etc/inittab to automatically start the ADSM server when the host is booted.

The installation of ADSM server puts a start-up entry into /etc/inittab. We will change that entry to use our custom start script.

The ADSM server can be started in either console mode or quiet mode (also called background mode). In console mode, ADSM runs in a workstation window. A default user ID called SERVER_CONSOLE is associated with the session. To start an ADSM console session, enter the `dsmserv` command without options. You may choose to run the ADSM server in the background. An administrative client session must be used to control the server. To start the server in background, run the `dsmserv` command with the `-quiet` option.

When the ADSM server product is installed, the administrator ID SERVER_CONSOLE is created and granted system class (the highest) authority. The SERVER_CONSOLE user ID does not have (nor can it be given) a password. We will register an installation system administrator and then reduce the authority of SERVER_CONSOLE as soon as possible.

You may decide to not reduce the authority of SERVER_CONSOLE. With ADSM password protection enabled (the default), the SERVER_CONSOLE ID cannot sign on to an administrative client. The signon requires a password, and SERVER_CONSOLE cannot be assigned one. Furthermore, two copies of the same server cannot be run simultaneously. (You can,

however, run multiple DIFFERENT servers at the same time.) So, if the ADSM server is automatically started in background (quiet) mode by the boot process, a console session cannot be started as long as the original server program continues to run. Without a console session, someone with bad intentions could not use your SERVER_CONSOLE ID with system authority to delete objects in your ADSM database.

We decided to take no chances (other than perhaps all the administrators with system class authority forget their ADSM password at the same time!). We reduced the authority of SERVER_CONSOLE at the earliest opportunity.

To start ADSM for the very first time:

```
# cd /adsmserv/tp3an01/config
# ulimit -d unlimited
# dsmserv
ANR7800I DSMSERV generated at 11:17:30 on Oct 6 1997.

ADSTAR Distributed Storage Manager for AIX-RS/6000
Version 3, Release 1, Level 0.1

Licensed Materials - Property of IBM

5765-C43 (C) Copyright IBM Corporation 1990, 1997. All rights reserved.
U.S. Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corporation.

ANR0900I Processing options file dsmserv.opt.
ANR0990I ADSM server restart-recovery in progress.
ANR0200I Recovery log assigned capacity is 24 megabytes.
ANR0201I Database assigned capacity is 500 megabytes.
ANR0306I Recovery log volume mount in progress.
ANR0353I Recovery log analysis pass in progress.
ANR0354I Recovery log redo pass in progress.
ANR0355I Recovery log undo pass in progress.
ANR0352I Transaction recovery complete.
ANR2100I Activity log process has started.
ANR2803I License manager started.
ANR8200I TCP/IP driver ready for connection with clients on port 1500.
ANR8285I Shared Memory driver ready for connection with clients on port 1510
ANR2560I Schedule manager started.
ANR0993I ADSM server initialization complete.

ADSM Server for AIX-RS/6000 - Version 3, Release 1, Level 0.1

adsm>
```

After a series of messages, the ADSM prompt (`adsm>`) should appear.

Note that the `ulimit` command is issued to avoid having the start of the ADSM server cancelled because of insufficient memory. If the start of the server is cancelled for this reason, you will see messages similar to the following:

```
System error - error data is:/usr/lpp/adsmserv/bin/dsmserv
System error: There is not enough memory available now.
```

If the ADSM server is started by a script called from /etc/inittab, add the command `ulimit -d unlimited` to that script ahead of the `dsmserv` command. In a later step, after ADSM server customization is complete and tested, an ADSM startup script will be developed and included in the /etc/inittab file to automatically start ADSM when the system is booted.

Paging Space Requirements

The ADSM Server program requires a significant amount of paging space to start. In our tests we noticed 170 MB more paging space was used after starting ADSM Server. Make sure that you have sufficient paging space defined.

3.10.5 Define an ADSM System Administrator

Next we defined an ADSM system administrator for our installation. The administrator ID is *sysadm*; the password is *password*. Then we granted system class authority to the new administrator:

```
adsm>  register admin sysadm password
adsm>  grant authority sysadm classes=system
```

From now on, we plan to customize the ADSM server using ADSM macro files. These files contain one or more ADSM commands. Macro files document the ADSM customization commands and can be useful to speed recovery in a disaster recovery situation. Macros cannot be entered through the ADSM server console session. They must be run from an ADSM administrative client.

Create a macros subdirectory in our configuration directory. All configuration macros will be kept and run from this subdirectory.

```
# mkdir /adsmserv/tp3an01/config/macros
```

Start an ADSM administrative client by opening an AIXterm window. Change to the macros directory and enter the `dsmadmc` command. When prompted, enter the administrator ID (sysadm) created above and the password (password):

```
> cd /adsmserv/tp3an01/config/macros
> dsmadmc
ADSTAR Distributed Storage Manager
Command Line Administrative Interface - Version 2, Release 1, Level 0.7
(C) Copyright IBM Corporation, 1990, 1996, All Rights Reserved.

Enter your user id:  sysadm
Enter your password:
ANS5100I Session established with server ADSM: AIX-RS/6000

adsm>
```

Revoke system class authority from the SERVER_CONSOLE administrator ID. Make sure that you know your ADSM system administrator password before you issue this command.

```
adsm>  revoke authority server_console classes=system
```

3.10.6 Register Additional ADSM Administrators

If you need to register additional ADSM administrators, you should use a macro file. For example, we created a macro file admin.mac in the configuration directory, /adsmserv/tp3an01/config/macros:

```
/* ---------------------------------------------------------------- */
/*           R E G I S T E R   A D M I N I S T R A T O R S          */
/* ---------------------------------------------------------------- */
/* This macro file defines administrators and, if required, grants  */
/* authorities.                                                     */
/* ---------------------------------------------------------------- */

register admin ssmith    -        /* System administrator           */
     2easy4u             -        /* initial password               */
     forcepwreset=yes    -        /* Change password at first login  */
     contact="S. Smith,  pager:  1-xxx-xxxxxxx pin 123456"

grant authority ssmith   -
     classes=system              /* Authorize sysadmin              */

register admin jjones    -        /* ADSM administrator (query only) */
     see4miles           -        /* initial password               */
     forcepwreset=yes    -        /* Change password at first login  */
     contact="J. Jones   pager:  1-xxx-xxxxxxx pin 987654"
```

It is recommended to not put the command to revoke system authority from SERVER_CONSOLE in a macro file. You need to be certain that you can login to the ADSM client with another ID and execute a system command before reducing SERVER_CONSOLE's authority.

The FORCEPWRESET option forces an ADSM administrator to change the password at the next login.

Notice a couple of things. Macro files can contain blank lines and comments. A comment begins with the character string /* (slash-asterisk) and ends with the string */ (asterisk-slash). To continue a command to the next line, use the line-continuation character - (hyphen).

Back in the ADSM administrative client window (dsmadmc session), run the macro:

```
adsm> macro /adsmserv/tp3an01/config/macros/admin.mac
```

Note that relative path names can be used. If you are in the directory /adsmserv/tp3an01/config/macros, you could enter the shorter:

```
adsm> macro admin.mac
```

3.10.7 Register ADSM Licenses

Although our system currently has only five potential client hosts (the SP Control Workstation and four SP nodes), nine client licenses will be registered to allow for growth. To use the TCP/IP transport, ADSM requires a network license. So we created a macro file called license.mac:

```
/* ------------------------------------------------ */
/*        R E G I S T E R   L I C E N S E S         */
/* ------------------------------------------------ */
/* Macro to install licenses to support 9 ADSM clients */

register license file(/usr/lpp/adsmserv/bin/1client.lic)
register license file(/usr/lpp/adsmserv/bin/1client.lic)
register license file(/usr/lpp/adsmserv/bin/1client.lic)
register license file(/usr/lpp/adsmserv/bin/5client.lic)

register license file(/usr/lpp/adsmserv/bin/network.lic)
```

This macro file installs licenses to support nine ADSM clients. Note that licenses can be repeated. In our case, since we want to allow up to nine ADSM clients to be registered, one 5-client and three 1-client licenses are used. One client license is installed with the ADSM Server product; so only eight additional licenses need to be added. A network license is also installed. Licenses are stored in the nodelock file in the directory in which the ADSM client session is running. For our example, this file is /adsmserv/tp3an01/config/nodelock.

To register the additional licenses, go to the ADSM administrative client (dsmadmc) and enter:

```
adsm> macro license.mac
```

The previous macro command and all subsequent macro commands will specify the relative path name only.

3.10.8 Define Tape Drives to ADSM

Before the drives can be defined to ADSM, the hardware must be installed and configured by AIX. To use the 3590 IBM Tape Drive with Automated Cartridge Facility as an SCSI library, set the drive to Random mode. To verify that the drives are configured correctly to AIX:

```
> lsdev -Cc tape

rmt0 Available 00-06-01-0,0 IBM 3590 Tape Drive and Medium Changer
rmt1 Available 00-02-01-0,0 IBM 3590 Tape Drive and Medium Changer
rmt2 Available 00-17-01-4,0 IBM 3590 Tape Drive and Medium Changer
rmt3 Available 00-18-01-0,0 IBM 3590 Tape Drive and Medium Changer

> ls -l /dev/*smc*

crw-rw-rw-   1 root     system    18,  2 Mar 16 18:03 /dev/rmt0.smc
crw-rw-rw-   1 root     system    18,258 Mar 16 18:03 /dev/rmt1.smc
crw-rw-rw-   1 root     system    18,514 Mar 16 18:03 /dev/rmt2.smc
crw-rw-rw-   1 root     system    18,770 Mar 16 18:03 /dev/rmt3.smc
```

In our example, we see four 3590 tape devices defined to AIX. We also see four library devices, which are named /dev/rmt*n*.smc where *n*=1,2,3,4.

To define these devices to ADSM, we created a macro called library.mac:

```
/* ------------------------------------------------- */
/*      D E F I N E    T A P E    D E V I C E S       */
/* ------------------------------------------------- */
/* Macro to define four 3590 tape devices to ADSM     */

define library l3590_01 libtype=scsi device=/dev/rmt0.smc
define drive   l3590_01 d3590_01 device=/dev/rmt0
define devclass devc3590_01 devtype=3590 library=l3590_01 -
       mountlimit=drives mountretention=2 mountwait=120
commit

define library l3590_02 libtype=scsi device=/dev/rmt1.smc
define drive   l3590_02 d3590_02 device=/dev/rmt1
define devclass devc3590_02 devtype=3590 library=l3590_02 -
       mountlimit=drives mountretention=2 mountwait=120
commit

define library l3590_03 libtype=scsi device=/dev/rmt2.smc
define drive   l3590_03 d3590_03 device=/dev/rmt2
define devclass devc3590_03 devtype=3590 library=l3590_03 -
       mountlimit=drives mountretention=2 mountwait=120
commit

define library l3590_04 libtype=scsi device=/dev/rmt3.smc
define drive   l3590_04 d3590_04 device=/dev/rmt3
define devclass devc3590_04 devtype=3590 library=l3590_04 -
       mountlimit=drives mountretention=2 mountwait=120
commit
```

There are three steps to complete for each tape device:

1. Define the tape library to ADSM using the `define library` command. Here we must specify the AIX device for the library, /dev/rmt*n*.smc
2. Define the tape drive to ADSM using the `define drive` command. Here we must reference the library name assigned in the first step and specify the AIX device for the tape drive, /dev/rmt*n*.
3. Define a device class to ADSM using the `define devclass` command. Here we must reference the library name assigned in the first step and specify other parameters for this device class.

   An important parameter is `MOUNTWAIT`, which defines the amount of time an ADSM client will wait for a tape device to become available before timing out. This is important during backup processing when we will be backing up 15 database partitions in parallel. These 15 backup images will be divided equally over the four available tape devices; so while each tape device is processing its first backup image, it will have a waiting list of up to three additional backup images. `MOUNTWAIT` needs to be set sufficiently high so that none of the ADSM clients time out. See “Archiving DB2 Log Files Using ADSM” on page 110 for more details.

For a full explanation of these commands, see the *ADSM for AIX 3.1 Administrator's Reference Guide*.

Using the ADSM administrative client (dsmadmc session), run the macro library.mac:

```
adsm> macro library.mac
```

To verify the defined libraries, use the `q library` ADSM command from dsmadmc. For instance:

```
adsm> q library

Library Name  Library     Device            Private   Scratch   External
              Type                          Category  Category  Manager
------------  ----------  ----------------  --------  --------  --------------
L3590_01      SCSI        /dev/rmt0.smc
L3590_02      SCSI        /dev/rmt1.smc
L3590_03      SCSI        /dev/rmt2.smc
L3590_04      SCSI        /dev/rmt3.smc
```

To verify the defined drives, use the `q drive` ADSM command from dsmadmc. For instance:

```
adsm> q drive

Library Name  Drive Name    Device Type  Device            ON LINE
------------  ------------  -----------  ----------------  -------------------
L3590_01      D3590_01      3590         /dev/rmt0         Yes
L3590_02      D3590_02      3590         /dev/rmt1         Yes
L3590_03      D3590_03      3590         /dev/rmt2         Yes
L3590_04      D3590_04      3590         /dev/rmt3         Yes
```

To verify the defined device classes, use the `q devclass` ADSM command from dsmadmc. For instance:

```
adsm> q devclass

Device       Device        Storage    Device       Format     Est/Max     Mount
Class        Access           Pool    Type                   Capacity     Limit
Name         Strategy        Count                               (MB)
---------    ----------    -------    ---------    ------    --------    ------
DEVC3590-    Sequential          2    3590         DRIVE          0.0    DRIVES
 _01
DEVC3590-    Sequential          1    3590         DRIVE          0.0    DRIVES
 _02
DEVC3590-    Sequential          1    3590         DRIVE          0.0    DRIVES
 _03
DEVC3590-    Sequential          1    3590         DRIVE          0.0    DRIVES
 _04
DISK         Random              3
```

3.10.9 Define Storage Pools to ADSM

Storage Pools are used by ADSM to access storage devices, which may be groups of tape devices or disks. Create a macro, storage.mac, to define storage pools to ADSM.

```
/* ---------------------------------------------------- */
/*      Define ADSM storage pools                       */
/* ---------------------------------------------------- */

define stgpool austin_tapepool_01 devc3590_01           -
       pooltype=primary                                 -
       description="austin primary pool - 3590_01"      -
       access=readwrite maxsize=nolimit collocate=no    -
       reclaim=100 maxscratch=30 reusedelay=0
commit

define stgpool austin_tapepool_02 devc3590_02           -
       pooltype=primary                                 -
       description="austin primary pool - 3590_02"      -
       access=readwrite maxsize=nolimit collocate=no    -
       reclaim=100 maxscratch=30 reusedelay=0
commit

define stgpool austin_tapepool_03 devc3590_03           -
       pooltype=primary                                 -
       description="austin primary pool - 3590_03"      -
       access=readwrite maxsize=nolimit collocate=no    -
       reclaim=100 maxscratch=30 reusedelay=0
commit

define stgpool austin_tapepool_04 devc3590_04           -
       pooltype=primary                                 -
       description="austin primary pool - 3590_04"      -
       access=readwrite maxsize=nolimit collocate=no    -
       reclaim=100 maxscratch=30 reusedelay=0
commit

define stgpool austin_copypool devc3590_01              -
       pooltype=copy                                    -
       description="austin copy storage pool"           -
       access=readwrite collocate=no                    -
       reclaim=100 maxscratch=30 reusedelay=3
commit
```

Note that the second parameter of the `define stgpool` command is the device class (for instance, devc3590_01). This must match the devclass names created in the library.mac macro above.

The `MAXSCRATCH` parameter (set to 30) defines the maximum number of labelled tapes that can be dynamically allocated to the storage pool.

For a complete description of the parameters of the `define stgpool` command, see the *ADSM for AIX V3.1 Administrator's Reference*.

To verify the storage pool definitions, use the `q stgpool` ADSM command from the ADSM client, dsmadmc. For example:

```
adsm> q stgpool

Storage      Device       Estimated    Pct    Pct  High  Low  Next
Pool Name    Class Name    Capacity   Util   Migr   Mig  Mig  Storage
                               (MB)               Pct  Pct  Pool
-----------  ----------  ----------  -----  -----  ----  ---  -----------
ARCHIVEPOOL  DISK               0.0    0.0    0.0    90   70
AUSTIN_COP-  DEVC3590_-          0.0    0.0
 YPOOL        01
AUSTIN_TAP-  DEVC3590_-    992,207.0    2.9    3.3    90   70
 EPOOL_01     01
AUSTIN_TAP-  DEVC3590_-  1,232,854.     3.0    3.3    90   70
 EPOOL_02     02                 0
AUSTIN_TAP-  DEVC3590_-  1,265,558.     3.0    3.3    90   70
 EPOOL_03     03                 5
AUSTIN_TAP-  DEVC3590_-  1,263,671.     3.0    3.3    90   70
 EPOOL_04     04                 0
BACKUPPOOL   DISK               0.0    0.0    0.0    90   70
SPACEMGPOOL  DISK               0.0    0.0    0.0    90   70
```

3.10.10 Create ADSM Policy Domains

Policy domains define the characteristics to be associated with ADSM client requests (such a backup). Each ADSM client uses the `register` command to specify its policy domain.

Within a policy domain, we define management classes. A management class holds detailed information such as which file names to include in the client request.

Also within a policy domain, we define copygroups for archive and backup. These define details such as how many versions of a object such be retained.

Create a macro to define the policy domains to ADSM. This macro file defines four policy domains, one for each 3590 tape library. We will show the full details here of only the first policy domain, which relates to the first 3590 tape library. The other three policy domains are identical in definition and only differ in their name and which ADSM client they serve. The complete policy.mac macro file is listed in “Policy.mac” on page 123.

Here is the first part of the policy.mac macro:

```
/* ------------------------------------------------------------*/
/*  S E T   U P   P O L I C Y   D O M A I N S                  */
/* ------------------------------------------------------------*/

/* ------------------------------------------------------------ */
/*                   austin_domain_01                           */
/* ------------------------------------------------------------ */

copy domain standard austin_domain_01

update domain austin_domain_01                                  -
    description="Custom domain for Austin"                      -
    backretention=30  archretention=365
commit

update mgmtclass austin_domain_01 standard standard             -
    description="austin_domain_01 standard management class." -
    spacemgtechnique=none
commit

update copygroup austin_domain_01 standard standard standard  -
    type=backup destination=austin_tapepool_01                 -
    frequency=0 verexists=2 verdeleted=1                       -
    retextra=30 retonly=60                                     -
    mode=modified serialization=shrdynamic

update copygroup austin_domain_01 standard standard standard  -
    type=archive destination=austin_tapepool_01                -
    frequency=cmd retver=15                                    -
    mode=absolute serialization=dynamic
commit
```

When creating domains, management classes and copygroups, we use the copy command to copy a standard version of the object, and then the update command to change only the fields that differ from the standard definition.

These commands above:

- Created a new domain called austin_domain_01 and defined its properties.
- Defined the standard management class for this domain.
- Defined the standard backup and archive copy groups for the standard management class for this domain. The `DESTINATION` parameter must match the storage pool name defined in stgpool.mac.

Here is the rest of the policy.mac macro for the first policy domain:

```
copy mgmtclass austin_domain_01 standard standard              -
    udb_mgmt_class

update mgmtclass austin_domain_01 standard udb_mgmt_class      -
    desc="Management class for DB2 UDB backups and log archives"
commit

update copygroup austin_domain_01 standard                     -
    udb_mgmt_class standard                                    -
    type=backup destination=austin_tapepool_01                 -
    frequency=0 verexists=1 verdeleted=0                       -
    retextra=0 retonly=60                                      -
    mode=modified serialization=shrdynamic

update copygroup austin_domain_01 standard                     -
    udb_mgmt_class standard                                    -
    type=archive destination=austin_tapepool_01                -
    frequency=cmd retver=365                                   -
    mode=absolute serialization=dynamic
commit

assign defmgmtclass austin_domain_01 standard standard

validate policyset austin_domain_01 standard

activate policyset austin_domain_01 standard

commit
```

These commands have:

- Created a new management class called udb_mgmt_class within the austin_domain_01 domain.
- Defined the backup and archive copy groups for the udb_mgmt_class management class. Again, the `DESTINATION` parameter must match the storage pool name defined in storage.mac.
- Assigned a default management class to the austin_domain_1 domain.
- Validated and activated the policy set for the austin_domain_1 domain.

A full explanation of these commands can be found in the *ADSM for AIX V3.1 Administrator's Reference*.

The ADSM commands `q domain`, `q mgmtclass` and `q copygroup` can be used from the ADSM client dsmadmc to verify the definitions.

3.10.11 Customize ADSM Server

The ADSM server can be further customized by using the `set` ADSM command. Create a macro, set.mac, to customize the ADSM server:

```
/* ------------------------------------------------------- */
/*                S E T   C O M M A N D S                  */
/* ------------------------------------------------------- */

set servername tp3an01
set authentication on
set actlogretention 30
set eventretention 30
set invalidpwlimit 0
set logmode rollforward
set passexp 60
set registration closed
set schedmodes any
```

These settings make the following customizations:

- SERVERNAME. Identifies the server by its host's short name (hostname `-s`). The system default value is ADSM.
- ACTLOGRETENTION. Number of days (here 30) to keep messages in the server activity log.
- EVENTRETENTION. Number of days (here 30) to keep event records.
- INVALIDPWLIMIT. When set to 0, this means that the node will not be locked. Invalid administrator login attempts are not counted.
- LOGMODE. When set to `rollforward`, this allows full recovery of the ADSM database. Save record of all changes since last database backup. This requires periodic database backups of the ADSM database.
- PASSEXP. Number of days (here 60) to expire password.
- REGISTRATION. When set to closed, this means that the system administrator must register all client nodes to the ADSM server.
- SCHEDMODES. When set to any, this means that an ADSM client can select its scheduling mode.

ADSM `set` commands are described in detail in the *ADSM for AIX V3.1 Administrator's Reference*.

3.10.12 Create ADSM Administrative Schedules

Administrative schedules allow you to define regular administrative tasks, such as when to delete inactive backup images through the process known as expiring the inventory. Create a macro, admin.mac, to create ADSM administrative schedules:

```
/* ---------------------------------------------------- */
/*   A D M I N I S T R A T I V E   S C H E D U L E S    */
/* ---------------------------------------------------- */

delete schedule * type=admin
delete schedule austin_domain_01 * type=client
delete schedule austin_domain_02 * type=client
delete schedule austin_domain_03 * type=client
delete schedule austin_domain_04 * type=client

define schedule EXPIREINVENTORY type=administrative -
     cmd="expire inventory" -
     starttime=08:30 -
     duration=1 durunits=hour -
     period=1 perunits=day dayofweek=any -
     active=yes
```

The `define schedule` command specifies that the `EXPIRE INVENTORY` task:

- Will take place at 08:30 (`STARTTIME`).
- Has as a one hour window to get started (`DURATION` and `DURUNITS`).
- Will take place once a day (`PERIOD` and `PERUNITS`) on all days of the week (`DAYOFWEEK`).
- Will run at the first opportunity because it is immediately active (ACTIVE).

3.10.13 Configure ADSM Server to Start at Boot

The installation of the ADSM Server program product adds an entry in the /etc/inittab file. The entry, which has the identifier "autosrvr," automatically starts the ADSM server when the operating system is booted. We want to initialize the ADSM server using the customized options file, dsmserv.opt, created in an earlier step. To do so, we will replace the default inittab entry with an entry that invokes our own ADSM startup script. First, create and test the custom start script which will be called *startadsm.rc* and stored in the custom configuration directory /adsmserv/tp3an01/config.

```
#!/bin/ksh

# Script name:  startadsm.rc
# Owner:  root
# Group:  system
# Permissions:  rwxr-----  (740)

# If installed, start up the 3494 tape library....

if [ -x /etc/methods/startatl ]; then
   /etc/methods/startatl
fi

# Get the language correct....

export LANG=en_US

ulimit -d unlimited

# Start the server

print "$(date '+%D %T') [$(basename $0)] Starting Server"
cd /adsmserv/tp3an01/config
export DSMSERV_CONFIG=./dsmserv.opt
export DSMSERV_DIR=/usr/lpp/adsmserv/bin
$DSMSERV_DIR/dsmserv quiet &

exit 0
```

Update /etc/inittab to use our custom script to start the ADSM server instead of the one inserted when the ADSM Server product was installed.

```
# rmitab autosrvr
# mkitab "autosrvr:2:once:/adsmserv/tp3an01/config/startadsm.rc "\
> "2>&1 >/dev/console # Start the ADSM server"
```

When the ADSM Server is started at boot time, a message like this will be displayed at the console:

```
03/26/98 12:06:56 [startadsm.rc] Starting ADSM Server
```

To verify that the ADSM Server is running, do:

```
[tp3an01][/]> ps -ef|grep dsmserv
    root 10722     1   0 12:06:57  pts/0  0:13 /usr/lpp/adsmserv/bin/dsmserv quiet
```

3.10.13.1 Stopping the ADSM Server

To shut down the ADSM server, use the `dsmadmc` command to request an administrative client session. When prompted, provide your administrator name and your password. When the `adsm>` prompt appears, enter `halt`.

```
[tp3an01][/]> dsmadmc
ADSTAR Distributed Storage Manager
Command Line Administrative Interface - Version 2, Release 1, Level 0.7
(C) Copyright IBM Corporation, 1990, 1996, All Rights Reserved.

Enter your user id:  sysadm
Enter your password:
ANS5100I Session established with server ADSM: AIX-RS/6000

adsm> halt
ANR2234W This command will halt the server; if the command is issued from a
remote client, it may not be possible to restart the server from the remote
location.
Do you wish to proceed? (Yes/No) y

ANS5103I Highest return code was 0.
```

Stopping ADSM Server

Be certain that you are at the ADSM prompt (`adsm>`) when you issue halt. If you are at the AIX prompt and you have the necessary permissions, down will come AIX!

3.10.14 Prepare IBM 3590 Tape Drives

When the IBM 3590 Tape Drive with Automatic Cartridge Facility (ACF) is set to random mode, it acts as a small, self-contained library of up to 10 cartridges, controlled by ADSM. ADSM uses the SCSI Medium Mover commands to select a cartridge from a given cell and move it to its destination cell. Control of the source and destination cells is left entirely to the ADSM server.

To ensure the integrity of the library, the magazine has a lock. When the magazine is installed in the ACF and locked, cartridges cannot be added or removed by the operator. Input and output of cartridges is through the *priority cell* under the control of ADSM (using the ADSM `CHECKIN` and `CHECKOUT` commands). In this way, ADSM can maintain a correct inventory of the cartridges in the library.

Prepare each 3590 drive for ADSM:

- Load up to ten cartridges into the ACF magazine.
- Insert the magazine into the ACF and press the **Lock** button.

- Using the operator panel on the ACF, set the ACF mode to Random Mode.
- Again using the operator panel, start ACF.

All of the magazine status lights are activated to *in use* and remain so unless random mode is disabled.

3.10.15 Prepare Tape Media

When the ADSM server accesses a tape volume, it checks the volume name in the tape header to ensure that the correct volume is used. To prepare the tapes for use by ADSM, they must be labelled and identified (that is, checked in) to ADSM.

To label the cartridges for ADSM, log in as root and run the following commands (one for each drive):

```
# dsmlabel -drive=/dev/rmt0 -library=/dev/rmt0.smc -search -keep -overwrite
# dsmlabel -drive=/dev/rmt1 -library=/dev/rmt1.smc -search -keep -overwrite
# dsmlabel -drive=/dev/rmt2 -library=/dev/rmt2.smc -search -keep -overwrite
# dsmlabel -drive=/dev/rmt3 -library=/dev/rmt3.smc -search -keep -overwrite
```

where:

- `drive` is the AIX device of the tape drive (/dev/rmt*n*).
- `library` is the AIX device of the tape library (/dev/rmt*n*.smc).
- `search` means all usable tapes in the library will be labelled.
- `keep` means that the tapes will kept in the library after labelling.
- `overwrite` means that any existing label will be overwritten. Use with care.

After the tapes are labelled, ADSM must build the initial inventory of each library. Using the ADSM administrative client, dsmadmc, log in to the ADSM server as a system administrator.

```
adsm> checkin libvolume l3590_01 status=scratch search=yes
adsm> checkin libvolume l3590_02 status=scratch search=yes
adsm> checkin libvolume l3590_03 status=scratch search=yes
adsm> checkin libvolume l3590_04 status=scratch search=yes
```

- The parameter search=yes specifies that ADSM should search the library for volumes that can be checked in automatically. One after another, each cartridge in the ACF is loaded into the 3590 drive. If ADSM does not already have the volume in inventory (which it should not), the volume is added.
- The CHECKIN command also requires a library name (for example, l3590_01). During ADSM server customization, a library name was assigned to each 3590 tape drive. Refer to the library.mac macro file which was used to define ADSM devices and device related objects.

The CHECKIN commands will run as ADSM background processes. The administrative client will not be notified when a process completes. If you wish to check the progress, periodically use either the ADSM QUERY PROCESS command (which will indicate whether the process is running or not) or the QUERY ACTLOG command. As each tape is checked in (or not), a record is written to the activity log.

3.10.16 High Availability Considerations for ADSM Server

To use HACMP to switch the ADSM Server function to a backup host, the following must be performed:

1. Select a backup host. Cable the external disk(s) containing the ADSM Server database, recovery logs, disk storage pools, tape devices to the backup host.
2. Install ADSM Version 3 filesets, adsm.server, adsm.license, and adsm.devices, on the backup host.
3. On the primary host, remove from /etc/inittab the entry that starts the ADSM server during boot.
4. Modify HACMP. Add application server scripts to start and stop the ADSM server. Add the ADSM server's volume group to the HACMP resource list. Make sure that the switch service address of the primary ADSM host is a failover resource. Add commands to make the tape devices available (mkdev -l <tape>) on whichever host is the current ADSM server host; make them unavailable (rmdev -l <tape>) on the non-server host.
5. Modify the ADSM client file dsm.sys on all hosts. Set parameter TCPServeraddress to be the switch service address of the primary ADSM server host. When an HACMP failover occurs, this service address will be initialized as a switch alias on the takeover host.
6. Test the failover. Make sure that the ADSM server starts on the backup host and that remove clients can backup files successfully.

3.11 ADSM Client Configuration

This section describes the configuration of Version 2.1.20.7 of ADSM Client for AIX 4.2, which was installed in an earlier section. Note that, at the time of writing, DB2 UDB EEE V5 does not support Version 3 of the ADSM Client for AIX. This support is planned for June 1998.

3.11.1 Create Client System Options File (dsm.sys)

The required client system options file, dsm.sys, identifies one or more ADSM servers to contact for services. The file contains communication options, authorization options, backup and archive processing options, and scheduling options for each server. ADSM provides default values for most of the options. So, except for a few required communication entries, most can be omitted.

ADSM provides a sample system options file, /usr/lpp/adsm/bin/dsm.sys.smp, that contains the minimum entries required to start using ADSM. For AIX hosts, the system options file, dsm.sys, must reside in directory /usr/lpp/adsm/bin. As we only used four client SP nodes, we simply copied this file to each of the SP nodes in /usr/lpp/adsm/bin. Optionally, if you have a large number of client nodes, you could choose one of these options:

- Include /usr/lpp/adsm/bin/dsm.sys into the user.admin file collection and propagate the file collection using `supper update user.admin`.
- Move the dsm.sys to a NFS shared directory, such as the instance owner's home directory, and create a link in /usr/lpp/adsm/bin.

The contents of dsm.sys follow:

```
************************************************************
* File:  dsm.sys
*
* ADSM V2 Client System Options file for AIX
************************************************************

SErvername         tp3an01

COMMmethod         TCPip
TCPPort            1500
TCPServeraddress   tp3sn01.ppd.pok.ibm.com  * Via switch

* --------------------------
* Performance options
* --------------------------

TCPBuffsize        32
TCPNodelay         YES
TCPWindowsize      640
TXNBytelimit       25600
USELARGebuffers    Yes      * V2 client above PTF 6

* --------------------------
* Security options
* --------------------------

Groups             system
Users              root
Mailprog           /usr/bin/mail root
Passwordaccess     Generate

* --------------------------
* Miscellaneous options
* --------------------------

InclExcl           /admin/adsm/inclexcl.list
SCHEDLOGname       /tmp/dsmsched.log
SCHEDLOGRetention  30
```

Some important parameters are:

- SERVERNAME. This parameter defines the name of the ADSM server this client will use.
- COMMMETHOD. Needs to be TCP/IP in our example as we are using the SP switch.
- TCPPORT. Must be unused and match the port that was defined in the ADSM server options file, dsmserv.opt.
- TCPSERVERADDRESS. Defines the network interface to be used at the ADSM server. In our example, we should use the switch interface to ensure optimal performance.

- PASSWORDACCESS. The installation of an ADSM server automatically sets password authentication on. To obtain services, the client node must provide present a valid password to the server. In the above options set, the Passwordaccess option specifies how to handle the ADSM password for the client nodes. When the option is set to generate, ADSM automatically creates a new password for the client node each time the client's password expires. The new password is encrypted and stored on the client node, in the directory /etc/security/adsm. When the client requests ADSM services, the password is retrieved from the local file and provided to the server. In most cases, ADSM does not prompt the client user for a password. If the ADSM administrator manually changes the client node password at the server (using the ADSM UPDATE NODE command), ADSM will prompt for the new password when the client next requests services. API programs, such as the db2uext2 user exit and the DB2 UDB backup and restore utilities, will fail in this situation. Once the initial password has been set, avoid changing it manually. If such a change must be made, an authorized user (usually root) should run the following command on the client node for which the password was modified:

```
dsmc set password <old_password> <new_password>
```

  ADSM will then prompt the user to confirm the new password by reentering it. If the old password is not known, the root user can delete the encrypted password file from /etc/security/adsm directory at the client node, and rerun the above dsmc command. Enter anything for the old password.

  Whenever a new password is automatically generated, ADSM will send the new password using the user ID and program specified in the option MAILPROG.

- GROUPS and USERS. If you do not specify group names with the GROUPS option or user IDs with the USERS option, all users can request ADSM services. With the options coded as they are, only root and members of the system group can request ADSM services. All other users are prevented from accessing the server.
- INCLEXCL. The file specified by the INCLEXCL parameter defines specific groups of files to include in or to exclude from a backup. This file is created at the Control Workstation (in /admin) and made available to the SP nodes through NFS because each SP node has the same requirements. See “Create Include-Exclude Options File” on page 96 for more details.

The full list of options are listed in *ADSM V2 Installing the Clients,* SH26-4049.

3.11.2 Create Client User Options File (dsm.opt)

Entries in this file control processing for sessions with ADSM, including some additional options that relate to backup, archive, restore, and retrieve operations. A default client user options file, dsm.opt, is created by root. We chose to write the file to the default directory, /usr/lpp/adsm/bin, but it could be placed elsewhere. Use the environment variable `DSM_CONFIG` to locate the dsm.opt if the file is stored in other than the default directory. Note that if you use `DSM_CONFIG`, you will also have to set `DSMI_CONFIG` to the same directory for the DB2 UDB EEE instance owner. You can do this either by using an environment variable or by using `db2set`.

The contents of the customized client user options file appear below.

```
*********************************************
* File:  dsm.opt
*
* ADSM V2 Client User Options file for AIX
*********************************************

SErvername          tp3an01
DOMain              all-local
REPlace             Yes
Tapeprompt          No
Quiet
ERRORLOGRetention   30
```

- The `DOMAIN` option specifies the default list of file systems to include during an incremental backup. Quite logically, "all-local" instructs ADSM to back up all file systems currently mounted on the client host. Instead of "all-local", a list of file systems could be provided. But what if a new file system is added? If you forget to add the new file system to the domain list, it will not be backed up. To avoid the possibility of omitting files from the backup, we prefer to include them all. The include/exclude list, described below, specifies files to exclude from the backup. So, by default, we will backup all files except those specifically excluded in the include/exclude list.
- The `TAPEPROMPT` parameter must be set to no, otherwise an error message will be generated when using the DB2 `BACKUP` command.

The full list of options are listed in *ADSM V2 Installing the Clients,* SH26-4049.

3.11.3 Create Include-Exclude Options File

This optional file identifies files the you want to explicitly include in or exclude from backup services. For example, you can exclude core files, local caches of network file systems, files that contain compiled object code that can be easily reproduced, and operating system files.

Another important use of this file is to assign a special management class to a file or group of files. Management classes specify how ADSM should handle certain files or groups of files in regard to how many backup versions to keep and how long to retain backup versions and archive copies. If you do not assign a management class to a file, ADSM assigns a default. Use the INCLEXCL option of the dsm.sys file (see “Create Client System Options File (dsm.sys)” on page 92) to identify the include-exclude options file to ADSM.

The include-exclude file below excludes some operating system files and assigns a special management class, UDB_MGMT_CLASS, to UDB objects stored by ADSM.

```
exclude /unix/
exclude /.../core
exclude /tmp/.../*
exclude /etc/ibmatl.pid
exclude /u/.../.sh_history
exclude /home/.../.sh_history
exclude /usr/games/.../*
exclude /.../.sh_history
exclude /.../smit.log
exclude /.../smit.script
exclude /adsmserv/tp3an01/db.1
exclude /adsmserv/tp3an01/log.1
* ------------------------------------------------------------
* Exclude DB2 UDB files for TEMPORARY tables
* ------------------------------------------------------------
exclude /DB_TMP/.../*
* ------------------------------------------------------------
* Assign DB2 UDB backups and logs to special management class
* ------------------------------------------------------------
include /*/NODE????/FULL_BACKUP.??????????????.*  UDB_MGMT_CLASS
include /*/NODE????/TSP_BACKUP.??????????????.*   UDB_MGMT_CLASS
include /*/NODE????/LOAD_COPY.??????????????.*    UDB_MGMT_CLASS
include /*/NODE????/S???????.LOG  UDB_MGMT_CLASS
```

Notice how the DB2 objects are defined in the include clause. When DB2 UDB uses ADSM to copy DB2 backup images, load copies or full log files to ADSM storage, the names given to the objects are:

- A full database backup object is:
  /<database>/NODEnnnn/FULL_BACKUP.timestamp.seq_no

- A table space backup object is: /<database>/NODEnnnn/TSP_BACKUP.timestamp.seq_no.
- A load copy object is: /<database>/NODEnnnn/LOAD_COPY.timestamp.seq_no.
- A full log files is:/<database>/NODEnnnn/Syyyyyyy.LOG.

where <database> is the database alias name, NODEnnnn is the node number, and Syyyyyyy.LOG is the log file name.

Wildcards are used to specify the ADSM object names. Asterisk (*) means one or many characters; a question mark (?) means a single character. So, the entry:

```
include /*/NODE????/FULL_BACKUP.??????????????.*  UDB_MGMT_CLASS
```

assigns a full backup image from any database, and on any node, to the UDB_MGMT_CLASS management class.

In this way, using the include/exclude file allows us to define one ADSM management class for all DB2 UDB instances and databases on the ADSM client nodes.

If you need to have different management classes for different databases, you could either:

- specify the database name in an include entry in the include/exclude file. For example, to assign full backups of the database TPCD30 to a management class called TPCD30_MGMT_CLASS:

  ```
  include /TPCD30/NODE????/FULL_BACKUP.??????????????.* TPCD30_MGMT_CLASS
  ```

- use the database configuration parameter `ADSM_MGMTCLASS`. For example, to assign ADSM management class TPCD30_MGMT_CLASS for TPCD30, the database instance owner would execute:

  ```
  db2_all "db2 update db cfg for TCPD30 using adsm_mgmtclass TPCD30_MGMT_CLASS"
  ```

  Since each database partition has its own set of database configuration parameters, the `db2_all` command is used to update the value on all database partitions.

3.11.4 Define the Client Nodes to the ADSM Server

Before a workstation can request ADSM services like backup and archive, it must be registered with the ADSM server. Using the ADSM Server customization command `SET REGISTRATION`, an authorized administrator can specify that ADSM client node registration be *open* or *closed*. With closed registration, a system or policy administrator defines:

- The workstation's node name. This should be set to the value returned by the hostname command.
- The ADSM client node password.
- The policy domain to which the client node belongs.
- Whether the user can choose to compress files before sending them to server storage.
- Whether the user is allowed to delete backup or archive files from server storage.

Using db2adutl to delete log files

For the DB2 UDB system administrator to use the db2adutl utility to delete obsolete archived log files, the option ARCHDEL of the register (or update) node command must be set to YES.

With open registration, when a user attempts to access the server from an unregistered client node, the server prompts the user for a node name, password, and contact information, and registers the workstation. On UNIX systems, only the root user can register a workstation as a client node with the server. The server sets the following defaults:

- Each client node is assigned to the policy domain named STANDARD.
- Each user defines whether data compression is used before files are sent to server storage.
- Each user is allowed to delete archived files from server storage. The user cannot delete backup files.

The administrator can reassign domains or change node attributes using the UPDATE NODE command.

To register ADSM client nodes at the ADSM server, create a macro file, nodes.mac, and file it in directory /adsmserv/tp3an01/config/macros with the other customization macros.

```
/* ------------------------------------------------ */
/*    R E G I S T E R   C L I E N T   N O D E S     */
/* ------------------------------------------------ */
/* This macro file informs the ADSM server of client  */
/* nodes that are authorized to connect to the server */

/* %1 = register or update                          */
/* %2 = client password                             */

%1 node sp-tp3cw.ppd.pok.ibm.com                     -
   %2                                                -
   domain=austin_domain_01

%1 node tp3an01.ppd.pok.ibm.com                      -
   %2                                                -
   domain=austin_domain_01                           -
   archdel=yes

%1 node tp3an05.ppd.pok.ibm.com                      -
   %2                                                -
   domain=austin_domain_02                           -
   archdel=yes

%1 node tp3an09.ppd.pok.ibm.com                      -
   %2                                                -
   domain=austin_domain_03                           -
   archdel=yes

%1 node tp3an13.ppd.pok.ibm.com                      -
   %2                                                -
   domain=austin_domain_04                           -
   archdel=yes
```

Notice a new twist in this macro. The strings %1 and %2 are called *substitution variables*. Parameters entered with the ADSM macro command replace these variables. The first parameter replaces %1, the second replaces %2, and so on. With the nodes.mac macro, the first parameter specifies the command (either register new nodes or update existing nodes). The second parameter supplies the password that the node will use when requesting services from ADSM. For example, to register the ADSM client nodes and reset their passwords, enter the following at the ADSM client (dsmadmc):

```
adsm> macro nodes.mac register newpasswd
```

The previous command registers five client nodes (the Control Workstation and four RS/6000 SP nodes) with the ADSM server. The initial password to be used when the node contacts the server is *nodepasswd*.

3.11.5 Set the Initial ADSM Password on Client Nodes

When the client nodes were registered to the ADSM server, a password (nodepasswd) was assigned. If password authentication is on (as it is for us), the client node must provide its password when ADSM services are requested. The ADSM client retrieves the password from a local file on the client host. The encrypted password is stored in /etc/security/adsm directory.

When the client node's password is initially set (`REGISTER NODE` command) or changed (`UPDATE NODE` command) at the server, the local password file on the client node must be updated as well. To do this, the root user can run the DSMC SET PASSWORD command at each client host for which the password has been modified or changed. When a lot of client nodes are affected, the process of logging into each client, running the DSMC command, logging out and repeating the process at each client host can become tedious and error-prone.

Automating the Process

Two scripts are provided below as samples of how the password change process might be automated. Instead of using the DSMC client command to change the password on the node, a program provided by DB2 UDB is used. This program, dsmapipw, is located in the /usr/lpp/db2_05_00/adsm directory.

The first sample script, dsmapipw.ksh, prompts the user, who must be root, for some passwords. The script then queries the RS/6000 SP System Data Repository (SDR) for all SP node names. The script was written with the assumption that all SP hosts are DB2 UDB hosts. Once dsmapipw.ksh has the necessary passwords and has determined the client hosts for which the password will be changed, it calls dsmapipw.exp. Script dsmapipw.exp, which is written using the Expect language, logs into each host as root, runs the DB2 UDB program dsmapipw to set the password, and then logs out.

The Korn Shell script, dsmapipw.ksh, is basically a front-end to dsmapipw.exp. It prompts the user to enter several passwords: the current ADSM password, the new ADSM password, and the root password. For both the current and the new ADSM passwords, enter the password that was specified to the ADSM server when the client node was registered or updated. When prompted, enter the root password.

Note that to use the script in its current form, the same ADSM password must be used for all the client nodes. The root password must be consistent across all nodes as well. The root password does not, of course, have to be the same as the ADSM password.

```
#!/bin/ksh
#
# ------------------------------------------------------------------------
# dsmapipw.ksh : This script sets/resets the adsm password on the SP2 nodes
# and the Control Workstation.
# ------------------------------------------------------------------------

admin_db2="/admin/db2"

echo "\nEnter the current ADSM password:  \c"
stty -echo
read OLD_PWD junk
stty echo
echo "\nEnter the new ADSM password:  \c"
stty -echo
read NEW_PWD junk
stty echo
echo "\nEnter the new ADSM password again:  \c"
stty -echo
read NEW_PWD2 junk
stty echo

if [[ "${NEW_PWD}" != "${NEW_PWD2}" ]]; then
   echo "\nNew password does not verify.  Try again!\n"
   exit 1
fi

echo "\nEnter the root password:  \c"
stty -echo
read ROOT_PWD junk
stty echo
echo "\nEnter the root password again:  \c"
stty -echo
read ROOT_PWD2 junk
stty echo

if [[ "${ROOT_PWD}" != "${ROOT_PWD2}" ]]; then
   echo "\nRoot password does not verify.  Try again!\n"
   exit 1
fi

HOST_NAMES=/tmp/host.names.$$
rm -f $HOST_NAMES

for i in $(SDRGetObjects -x host_responds|egrep -v "   0 $" \
   |awk '{print $1}')
do
   SDRGetObjects -x Node node_number==$i \
      reliable_hostname >>$HOST_NAMES
done

$admin_db2/dsmapipw.exp ${ROOT_PWD} $HOST_NAMES $OLD_PWD $NEW_PWD
```

This Korn Shell script, dsmapipw.ksh, after collecting passwords and making a list of hosts, calls the Expect script, dsmapipw.exp. This script, which is listed below, logs into each host in the list, changes the ADSM client password, and exits.

```
#!/usr/lpp/ssp/expect/bin/expect
# --------------------------------------------------------
#  dsiapipw.exp : Change ADSM passwords on multiple nodes
# --------------------------------------------------------

set password [lindex $argv 0]
set hostlist [lindex $argv 1]
set ADSM_oldpw [lindex $argv 2]
set ADSM_newpw [lindex $argv 3]
set list [exec cat $hostlist]
set l [exec wc -l $hostlist]
set len [lindex $l 0]

for {set i 0} {$i < $len} {set i [expr $i+1]} {
  set h [lindex $list $i]
  spawn telnet $h
    expect {login: }
      send root\r
    expect {*Password: }
      send $password\r
    expect {*}
      send /usr/lpp/db2_05_00/adsm/dsmapipw\r
    expect {*current password:}
      send $ADSM_oldpw\r
    expect {*your new password:}
      send $ADSM_newpw\r
    expect {*your new password again:}
      send $ADSM_newpw\r
         expect {*}
         send exit\r
  expect eof
}
```

3.12 Overview of Backing up a Database Using ADSM

Figure 10 on page 104 illustrates how the ADSM definitions relate together to take a DB2 backup image and store it onto a 3590 tape library. These are the steps:

1. The DB2 `BACKUP` command specifies the clause `USE ADSM`. This instructs DB2 to call ADSM and take the DB2 backup image as input. In the diagram, only the backup image for database partition 1 is shown.
2. The ADSM client system options file, dsm.sys defines the ADSM server to be used (tp3an01).
3. The DB2 backup image is named for ADSM: /TPCD30/NODE0001/FULL_BACKUP.19980323130942, where the database name is TPCD30, the database partition number is 1, and the timestamp shown is for 13:09:42 on 23 Mar 1998. This backup image name matches an entry in the include/exclude file, which defines that images of this name will use the udb_mgmt_class ADSM management class. The include/exclude file name is defined in the ADSM client system options file, dsm.sys.

4. The ADSM client node entry defines that this client will use the ADSM policy domain called austin_domain_01 defined at the ADSM server, tp3an01.
5. The destination storage pool, austin_tapepool_01, is defined in the policy set entry which has:
   - domain=austin_domain_01
   - mgmt_class=udb_mgmt_class
   - copygroup=backup
6. The device class for this storage pool is defined as devc3590_01.
7. The library for this device class is l3590_01.
8. The AIX devices are defined as /dev/rmt0 for the tape and /dev/rmt.smc for the library.

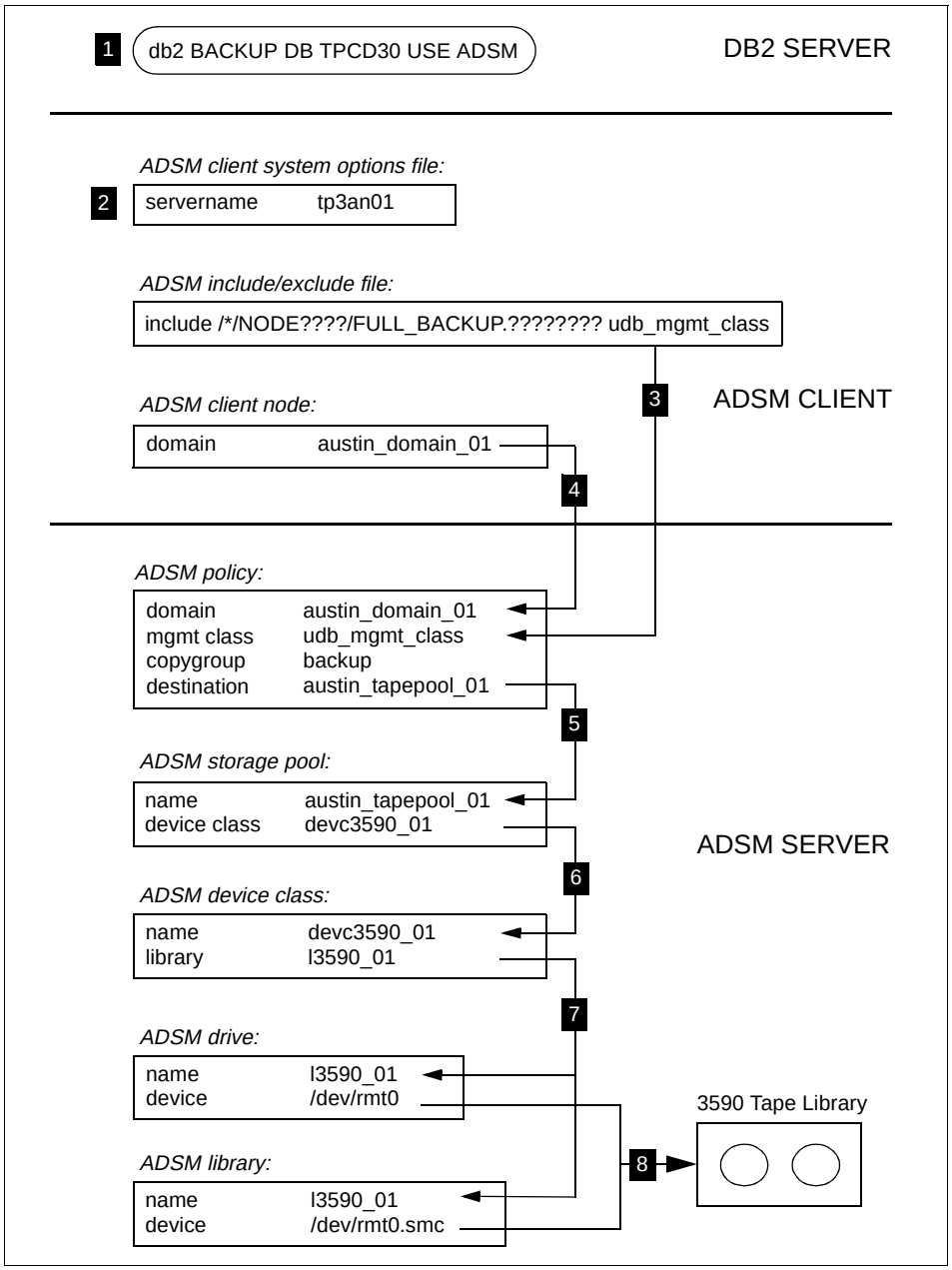


Figure 10. Overview of Backing up a Database using ADSM

3.13 Online Database Backup Using ADSM

To run a backup of a partitioned database using ADSM, you first need to run a backup of the catalog database partition. Once that has finished, run a backup of all the other partitions in parallel. If your catalog partition is not necessarily on the first partition of the database (which is recommended), then you can use the script below (DB_backup.ksh). This script first queries the database directory to find the catalog partition. It then carries out the backup on that partition, followed by a parallel backup of the other partitions.

```
#!/usr/bin/ksh
#
# Sample KornShell script to which uses db2_all to perform
# a full online backup of a database.
#
  ONLINE="online"
  ALIAS=${1:-TPCD30}   # Set database alias
  SCRIPT_NAME=$(basename $0)

# Determine catalog partition

  CAT_PART=‘db2 list database directory    \
       | awk -F"=[ ]+"                     \
         ’/^ Database alias/ {             \
                   if ($2 == ALIAS) {      \
                      alias_found="y"      \
                   }                       \
              }                            \
          /^ Catalog node number/ {        \
                   if (alias_found=="y") { \
                      print $2             \
                   }                       \
              }                            \
         ’ ALIAS=$ALIAS‘
  if [ -z "$CAT_PART" ]; then      # If cannot find catalog partition, exit
      print "$SCRIPT_NAME:  Unable to determine catalog partition for"\
            "\n\tdatabase $ALIAS.  Check alias and verify that database"\
            "\n\tis defined on this host, $(uname -n)."\
            "\n-----ABORTING-----\n"
      exit 1
    else
      print "$SCRIPT_NAME:  Catalog partition for database $ALIAS is"\
            "$CAT_PART.\n"
  fi

# Backup the catalog partition.  The 2-character string \" in the prefix of
# the db2_all command causes all occurrences of ## to be replaced by the
# database partition number; the leading zero in RAHWAITTIME assignment
# suppresses those rah status messages.

  print "$SCRIPT_NAME:  Backing up catalog partition."

  db2_all "<<+$CAT_PART<\"export RAHWAITTIME=060;db2 terminate >/dev/null;echo DB
Partition=##;db2 backup db $ALIAS $ONLINE use adsm;db2 terminate >/dev/null"

# Backup the rest of the partitions.  The "||" in the db2_all prefix causes
# the script to wait until all backround jobs complete.  See notes above
# concerning RAHWAITTIME and the \" in the db2_all prefix.
```

```
  print "$SCRIPT_NAME:  Backing up remaining partitions....."

  db2_all "<<-$CAT_PART<||\"export RAHWAITTIME=060;db2 terminate >/dev/null;echo
DB Partition=##;db2 backup db $ALIAS $ONLINE use adsm;db2 terminate >/dev/null"

  print "$SCRIPT_NAME:  All DB backups for $DB_ALIAS are complete!"
exit 0
```

To monitor the backup in ADSM, start an ADSM client session using dsmadmc and use the `q session` command. Here is some example output:

```
adsm> q sess

  Sess Comm.  Sess     Wait   Bytes   Bytes Sess  Platform Client Name
Number Method State    Time    Sent   Recvd Type
------ ------ ------ ------ ------- ------- ----- -------- --------------------
   367 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN01.PPD.POK.IBM-
                                                          .COM
   368 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN01.PPD.POK.IBM-
                                                          .COM
   369 Tcp/Ip RecvW    0 S    4.4 K   1.3 G Node  DB2/6000 TP3AN01.PPD.POK.IBM-
                                                          .COM
   370 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN05.PPD.POK.IBM-
                                                          .COM
   371 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN05.PPD.POK.IBM-
                                                          .COM
   372 Tcp/Ip RecvW    0 S    4.4 K   1.2 G Node  DB2/6000 TP3AN05.PPD.POK.IBM-
                                                          .COM
   373 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN05.PPD.POK.IBM-
                                                          .COM
   375 Tcp/Ip RecvW    0 S    5.0 K   1.4 G Node  DB2/6000 TP3AN09.PPD.POK.IBM-
                                                          .COM
   376 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN09.PPD.POK.IBM-
                                                          .COM
   377 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN09.PPD.POK.IBM-
                                                          .COM
   378 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN09.PPD.POK.IBM-
                                                          .COM
   379 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN13.PPD.POK.IBM-
                                                          .COM
   380 Tcp/Ip RecvW    0 S    4.9 K   1.4 G Node  DB2/6000 TP3AN13.PPD.POK.IBM-
                                                          .COM
   381 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN13.PPD.POK.IBM-
                                                          .COM
   382 Tcp/Ip MediaW 4.9 M    1.3 K     581 Node  DB2/6000 TP3AN13.PPD.POK.IBM-
                                                          .COM
   383 Tcp/Ip Run      0 S   49.6 K     576 Admin AIX      SYSADM
```

Here we can see that at any one time, one database partition from each SP node is writing to ADSM storage. The other partitions have to wait for access to their assigned tape libraries. The column `Bytes Recvd` shows the amount of data received by ADSM from DB2 UDB for each ADSM session.

3.14 Database Restore Using ADSM

To restore a partitioned database using ADSM, we must first identify the backup images we wish to restore. The following scripts automate the

process on the assumption that you wish to restore the last backup image that exists on each partition, and then roll the logs forward to the end of logs.

The script DB_restore_all.ksh determines the catalog partition for a database. Then for each database partition, the script DB_restore_partition.ksh is run. This script determines the most current ADSM backup file for a partition, determines the target directory, and runs the RESTORE command WITHOUT PROMPTING.

Note that the catalog partition must be restored before the other partitions. Finally, when all partition restores are complete, the ROLLFORWARD command is submitted to the catalog partition.

```
#!/usr/bin/ksh
#
# Script : DB_restore.ksh
#
# Restore all database partitions and then perform a ROLLFORWARD to END OF LOGS.
#
  DB_ALIAS=${1:-TPCD30}      # Default DB alias name
  SCRIPT_NAME=$(basename $0)

  if [[ -z $DB2INSTANCE ]]; then
      print "$SCRIPT_NAME:  Variable DB2INSTANCE is not set."\
            #\n-----ABORTING-----\n"
      exit 1
  fi

  CAT_PART=‘db2 list database directory    \
       | awk -F"=[ ]+"                     \
         ’/^ Database alias/ {             \
                   if ($2 == ALIAS) {      \
                      alias_found="y"      \
                   }                       \
               }                           \
          /^ Catalog node number/ {        \
                   if (alias_found=="y") { \
                      print $2             \
                   }                       \
               }                           \
         ’ DB_ALIAS=$DB_ALIAS‘
  if [ -z "$CAT_PART" ]; then
      print "$SCRIPT_NAME:  Unable to determine catalog partition for"\
            "\ndatabase $DB_ALIAS.  Check alias and verify that database"\
            "\nis defined on this host, $(uname -n)."
      exit 1
    else
      print "$SCRIPT_NAME:  Catalog partition for database $DB_ALIAS is"\
            "$CAT_PART.\n"
  fi

  LOG_PATH=‘db2 get db cfg for $DB_ALIAS   \
       | awk -F"=[ ]+"                     \
         ’/^ Path to log files/ {          \
                   print $2                \
                   }                       \
         ’ ’
  if [ -z "$LOG_PATH" ]; then
      print "$SCRIPT_NAME:  Unable to determine log path for database"\
```

```
            "\n $DB_ALIAS.   -----ABORTING-----"\n
        exit 2
      else
        print "$SCRIPT_NAME:  Log path for database $DB_ALIAS is $LOG_PATH.\n"
    fi

    set -- $(echo $LOG_PATH | tr '/' ' ')

    typeset -i DIRECTORY_COUNT=$#
    typeset -i loop_count=0
    TARGET_DIRECTORY="/"
    while [[ $loop_count -le $# && "$1" != "$DB2INSTANCE" ]]; do
        TARGET_DIRECTORY="${TARGET_DIRECTORY}${1}/"
        shift 1
    done
    TARGET_DIRECTORY=${TARGET_DIRECTORY%*/}       # Chop off trailing slash
    if [[ $loop_count -gt "$DIRECTORY_COUNT" ]]; then
        print "$SCRIPT_NAME:  Unable to find instance name in path $DB_PATH."\
              "\n\t-----ABORTING-----\n"
        exit 3
    fi

    print "$SCRIPT_NAME:  Restoring catalog partition."

    db2_all "<<+$CAT_PART<||\" export
RAHWAITTIME=060;$HOME/bin/DB_restore_partition.ksh $DB_ALIAS ##
$TARGET_DIRECTORY"

    print "$SCRIPT_NAME:  Restoring remaining partitions..."

    db2_all "<<-$CAT_PART<||\" export
RAHWAITTIME=060;$HOME/bin/DB_restore_partition.ksh $DB_ALIAS ##
$TARGET_DIRECTORY"

    print "$SCRIPT_NAME:  Rolling forward to end-of-logs and complete...\n"

    db2_all "<<+$CAT_PART<||\" export RAHWAITTIME=060;echo DB Partition=##;db2
rollforward db $DB_ALIAS to end of logs and complete;db2 terminate >/dev/null"

    print "$SCRIPT_NAME:  All done!"

exit 0
```

This is the DB_restore_partition.ksh script:

```
#!/usr/bin/ksh
#
# Script: DB_restore_partition.ksh
#
# This script restores a DB2 UDB partition using the most
# current ADSM backup file.

    SCRIPT_NAME=$(basename $0)

    if [[ "$#" -ne 3 ]]; then
        print "$SCRIPT_NAME:  Invalid number of parameters...."\
              "\n\t-----ABORTING-----\n"
        exit 1
    fi
    DB_ALIAS=$1
    DB_PARTITION=$2
    TARGET_DIRECTORY=$3
    print "P1=$1, P2=$2,P3=$3"
```

```
# Get timestamp of most recent full backup.

  LATEST_TIMESTAMP=‘db2adutl query full node $DB_PARTITION \
       | grep "Time:"   \
       | awk ’BEGIN {FS="Time: |, Oldest"}{print $2}’ \
       | sort | tail -1 ‘

  if [ -z "$LATEST_TIMESTAMP" ]; then
      print "$SCRIPT_NAME:  Unable to determine most recent backup for"\
            "\n\tdatabase $DB_ALIAS, partition $DB_PARTITION."\
            "\n-----ABORTING-----\n"
      exit 2
  fi

  print "$SCRIPT_NAME:  Restoring database $DB_ALIAS, partition $DB_PARTITION,"\
        "\n\tusing backup with timestamp $LATEST_TIMESTAMP.\n"

  export DB2NODE=$DB_PARTITION    # Not really necessary

  db2 restore db $DB_ALIAS use adsm taken at $LATEST_TIMESTAMP \
      to $TARGET_DIRECTORY without prompting

  RC=$?
  case "$RC" in
      0) print "$SCRIPT_NAME:  Restore of database $DB_ALIAS, partition"\
             "$DB_PARTITION,"\
              "\n\tcompleted OK.\n"
         ;;
      2) print "$SCRIPT_NAME:  Restore of database $DB_ALIAS, partition"\
             "$DB_PARTITION,"\
              "\n\tcompleted with warning.  Warning number 2539 indicates that"\
              "\n\tthe restore command specified WITHOUT PROMPTING clause..\n"
         ;;
      *) print "\n$SCRIPT_NAME:  Restore of database $DB_ALIAS, partition"\
             "$DB_PARTITION,"\
             "\n\t--->> FAILED ---- FAILED ---- FAILED.  Return code=$RC.\n"
         ;;
  esac

  db2 terminate

exit 0
```

As long as you restore to the same database name, everything is automated. If you want to restore to a new database name, it is gets a little bit more difficult because the logs are archived under the old name, but the roll forward will use the new database name when requesting the logs from ADSM.

3.15 Tablespace Restore Using ADSM

To ensure that restored table spaces are synchronized with the rest of the database, the table spaces must be rolled forward to the end of the logs. For this reason, table-space-level backup and restore can only be performed if roll-forward recovery is enabled. If roll-forward recovery is disabled at any

time after a table space level backup is executed, it will not be possible to restore from the backup and then roll the table space forward to the current point in time.

Using ADSM to archive UDB log files does have one drawback where table space restores are concerned. When a table space is rolled forward, the db2uext2 user exit will not be called. Consequently, archived log files will not be automatically retrieved from ADSM. So before the ROLLFORWARD command is issued, the database administrator must retrieve the logs and store them in either the database log path or in an alternate log path. Be sure there is sufficient space to contain the retrieved logs. If the retrieved log files are stored in a different directory from the active log files, use the OVERFLOW LOG PATH option of the ROLLFORWARD command. The following command shows how to retrieve log files from ADSM by running the db2uext2 user exit manually:

```
db2uext2 -RQRETRIEVE -DBTPCD30 -NNNODE0001 -LP/work/NODE0001 -LNS0000012.LOG
```

Option -RQ above instructs the user exit to retrieve from database (-DB) TPCD 30 the log file named (-LN) S0000012.LOG for database partition (-NN) 1 and store it in directory /work/NODE0001 as file named S0000012.LOG. There can be no spaces between an option and its argument. A slash must terminate the log path (option -LP) name. With the exception of log path directory (-LP), all option arguments must be capitalized.

After all necessary log files have been retrieved and stored, the ROLLFORWARD TABLESPACE command can be run. The ROLLFORWARD command must be run on the host containing the database catalog partition.

3.16 Archiving DB2 Log Files Using ADSM

For this project, ADSM will store the UDB database log files. IBM supplies three sample user exit programs: one for tape, one for disk, and one for ADSM. After configuring the ADSM user exit, as soon as a log file becomes full (even if it contains log records for active transactions), DB2 will call the db2uext2 user exit program to copy the log file to ADSM storage.

Use of these sample programs is not mandatory. You may choose to create your own user exit program, perhaps using one of the sample programs as a model. Useful information can be found in comments at the start of each sample program. Appendix J of the *DB2 UDB Administration Guide* describes a number of considerations that apply to calling a user exit program for archiving and retrieving log files. The guide also describes the database

manager format for calling the exit program on a UNIX operating system, and it explains the specific return codes that the exit can provide back to the calling database manager.

The steps to install the sample ADSM user exit follow:

1. To install the ADSM exit requires a C compiler, the ADSM user exit source file, and access to the ADSM API library and header files. The library (libApiDS.a) and the header files (dsmrc.h, dsmapifp.h, and dsmapitd.h) are located in directory /usr/lpp/adsm/api/bin, which is part of the ADSM Client for AIX Version 2.

   Note: ADSM Client Version 2 must be installed on the DB2 UDB hosts. At the time of writing, DB2 UDB did not currently support the Version 3 ADSM client. Version 3 client support was planned for June 1998.

2. Log in to a UDB host as the instance owner. Change to a working directory, the home directory perhaps.

3. Copy the ADSM sample user exit to the working directory:

```
cp -p $HOME/sqllib/misc/c/db2uext2.cadsm ./db2uext2.c
```

4. Edit the copied file, db2uext2.c, to specify "installation defined variables" which suit your environment. A few considerations apply:

   - Comments at the beginning of the db2uext2.c source code provide useful information including several sample scenarios of changing installation defined variables.
   - Both path names, TEMP_DIR and AUDIT_ERROR_PATH, must end with a slash. These directories must exist; the user exit will not create them. TEMP_DIR provides a temporary repository for files being archived to or retrieved from ADSM. The directory AUDIT_ERROR_PATH contains error and audit logs. The audit logs report ADSM activity. One log documents the use of ADSM to archive UDB log files. A separate file tracks the retrieval of UDB log files from ADSM. If an ADSM archive or retrieve operation fails, the event is recorded in the error log.
   - Although the comments in the source code state otherwise, the same directory can be used for TEMP_DIR and for AUDIT_ERROR_PATH.
   - For each database partition, only one UDB log file will be archived or retrieved at a time. When multiple database partitions run on the same host, the number of archive (or restore) operations could equal the number of database partitions. If a host runs four partitions of a

database, TEMP_DIR could contain up to four copies of UDB log files at the same time. Plan file system free space accordingly.

- Periodically, delete obsolete entries from the user exit audit and error logs. By default, the exit collects both error and audit information; the files are opened in append mode. Without purging, the files continue to grow and may eventually fill up the file system.

For this project, the variables BUFFER_SIZE, TEMP_DIR, and AUDIT_ERROR_PATH will be modified. Observe that the same directory is used for both error logs and for audit logs. Also, ignore the comment that TEMP_DIR will be created and removed by the user exit. If TEMP_DIR (here /var/tmp/UserExits/) does not exist, you must create it. Here are the define statements we used from the db2uext.c file:

```
#define BUFFER_SIZE         32768   /* transmit or receive the  */
                                    /* log file in 32K portions */
#define TEMP_DIR            "/var/tmp/UserExits/"
#define AUDIT_ACTIVE        1       /* enable audit logging     */
#define ERROR_ACTIVE        1       /* enable error logging     */
#define AUDIT_ERROR_PATH    "/var/tmp/UserExits/"
#define AUDIT_ERROR_ATTR    "a"     /* append to text file      */
```

Note that the directory names end in a forward slash (/).

5. Compile the user exit. The exit must be named db2uext2. The -I (as in Include) command flag directs the compiler to the location of the ADSM API header files. An additional flag -l (lowercase L) instructs the C compiler to include the ADSM API library, which will be found in directory specified by the -L flag.

```
> cc -o db2uext2 db2uext2.c -L/usr/lpp/adsm/api/bin -lApiDS \
        -I/usr/lpp/adsm/api/bin
```

6. Copy (or move) the exit program to the sqllib/adm directory and insure that the instance owner has execute permission.

```
> cp -p db2uext2 $HOME/sqllib/adm
```

7. Create the directories (or directory) specified for the TEMP_DIR and AUDIT_ERROR_PATH installation variables. Verify that file system free space is sufficient to contain the files.
8. Make the directory (root authority is not required to create a subdirectory in /var/tmp):

```
> mkdir /var/tmp/UserExits
```

Since this project runs four database partitions on each high node, there may be as many as four copies of database log files in the TEMP_DIR directory at the same time, one for each database partition. The user exit's audit and error logs, if enabled, require additional free space. Remember to prune the user exit log files. If the file system capacity is insufficient to handle the additional requirements, work with your UNIX system administrator to expand the file system.

9. Update the USEREXIT database configuration parameter for all partitions to direct the database manager to begin using the db2uext2 user exit when it is next started. Run the following command (tpcd30 is our database alias):

```
db2_all "db2 update db cfg for tpcd30 using userexit on"
```

10. To activate the user exit, stop and start the database.

```
> db2stop
> db2start
```

11. Perform a full offline backup of the database. The first time the database is started after USEREXIT is enabled, the database will be in Backup Pending state. Before it can be used, an offline backup must be made. First, back up the catalog database partition (here nodenum 1). After that completes, back up the remaining partitions.

```
> db2_all "<<+1< db2 backup db tpcd30 use adsm"
> db2_all "<<-1<;db2 backup db tpcd30 use adsm"
```

Note the use of the db2_all special modifiers:

- <<+1< means run the command on only the first database partition.
- <<-1<; means run the command on all database partitions apart from the first one in background simultaneously.

With our four-tape ADSM server setup, the full backup of a 30 GB database completed in approximately thirty-five minutes.

After the backup of the catalog database partition finished, the commands to back up the remaining 15 partitions were all submitted at the same time. When the number of jobs requiring tape exceeds the number of tape

drives available, the excess jobs will queue waiting for drives to be freed by current processes. When ADSM detects that a job has been waiting for a tape mount for longer than a configurable time limit, the job will be cancelled. The ADSM device class parameter `MOUNTWAIT` defines the length of time a job is allowed to wait for a tape mount before the job is terminated. The initial value for `MOUNTWAIT` is set when the device class is defined using the ADSM `DEFINE DEVCLASS` statement (the default value is 60 minutes). The `MOUNTWAIT` value can be changed with the ADSM `UPDATE DEVCLASS` statement. To submit the backup commands in parallel as above, the `MOUNTWAIT` value needs to be at least 30 (minutes) to avoid the possibility that ADSM will terminate one or more of the backup commands. In fact, for this project, a `MOUNTWAIT` value of 120 minutes was chosen. Our device class definitions can be found in the ADSM configuration macro file, library.mac, in “Define Tape Drives to ADSM” on page 78.

3.17 Using db2adutl to Manage Backups and Logs

The db2adutl utility provides query, extract, and deletion of backups, logs and copy images saved by ADSM. You can build your backup/log archive management process around the function offered by db2adutl. This section introduces some of the concepts of ADSM and UDB, highlights the options of the db2adutl utility, and closes with some examples of how you might use db2adutl to automatically manage your ADSM images.

3.17.1 ADSM and DB2 Concepts

ADSM manages UDB backups and logs differently. ADSM backups of UDB log files are called (in ADSM) *archive files*. UDB backups are stored in ADSM as *backup files*. An ADSM management class is associated with every ADSM object. A management class informs ADSM of your requirements for the assigned set of objects. For instance, how many backup versions of a file should be kept, how long should backup versions and archive copies be kept and so on. See “Create Include-Exclude Options File” on page 96 for details of how to associate an ADSM management class with DB2 objects.

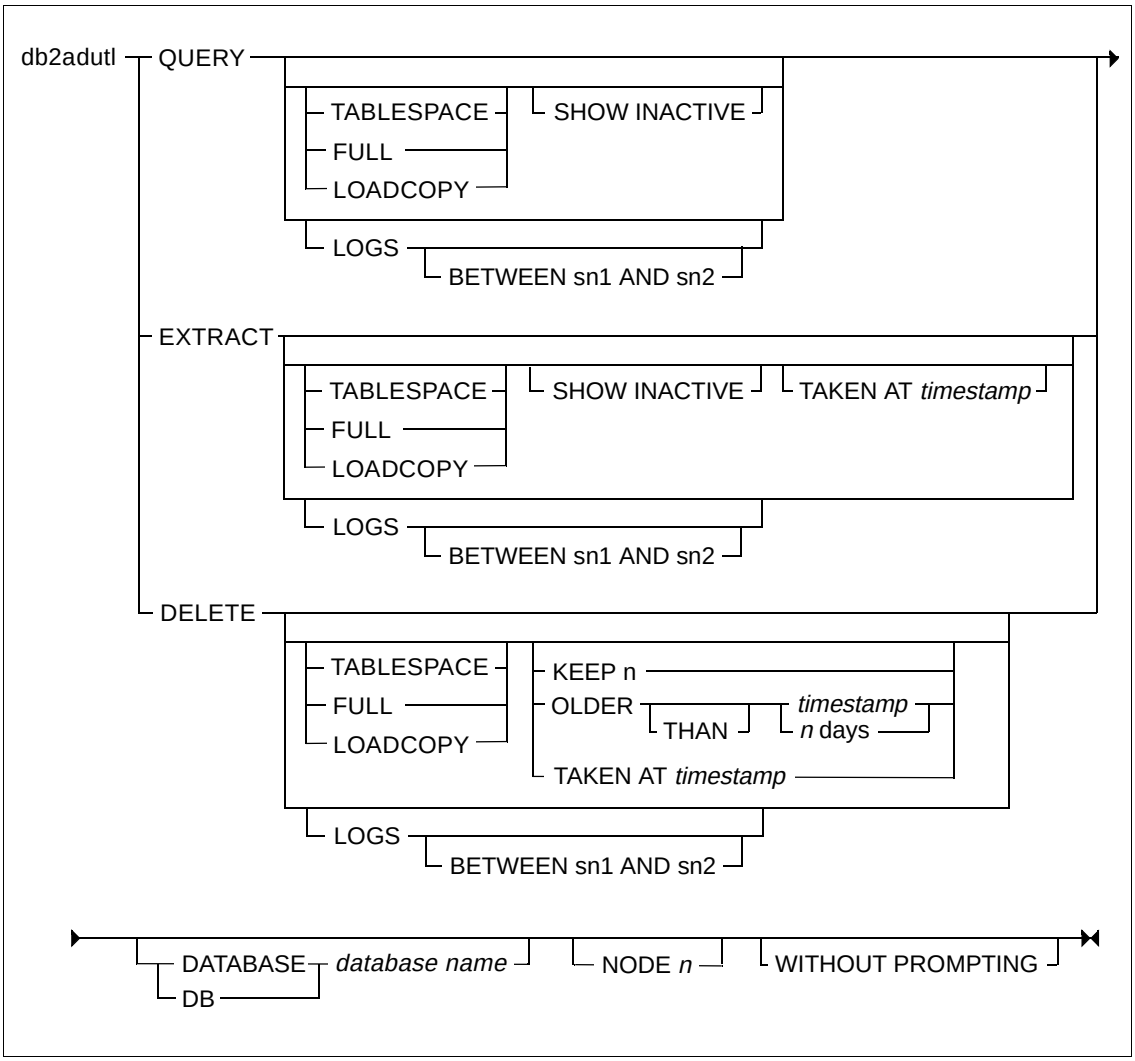


Figure 11. db2adutl Syntax

3.17.2 Query Option of db2adutl

With no other parameters, `db2adutl QUERY` lists all active database and tablespace backups, load copy images, and log archive files for all databases and in our case, all database partitions. To only list images and files for one database partition, use the `AT NODE` parameter. An additional parameter specifies a particular database alias where an instance includes more than one database. Active tablespace images or backup images or load copy

images can be displayed. The SHOW INACTIVE clause lists, in addition to the active ADSM images, those which have been deleted using db2adutl DELETE. For more details on db2adutl DELETE and active versus inactive images, see the following section.

3.17.3 Delete Option of db2adutl

The DELETE option of db2adutl deletes log files or deactivates backup and load copy files. Note the distinction here. DELETE LOGS removes entries from the ADSM database. DELETE FULL (or TABLESPACE or LOADCOPY) deactivates the database (or tablespace or loadcopy) image(s) maintained by ADSM. Unlike the logs that are deleted, the backups are flagged as inactive (but not deleted) by db2adutl. ADSM flags these inactive files for delete based upon the backup copy group definition in the associated management class. When the ADSM EXPIRE INVENTORY command runs, the files flagged for deletion will be removed from the ADSM storage volume(s).

Using the management class definitions, you can set up ADSM so that deletion of ADSM images is controlled by db2adutl, by ADSM, or by a combination of both. Section 3.17.5, “An Example Usage of db2adutl” on page 117, provides a step-by-step description of how a process might be set up to manage backups and log archives.

The db2adutl DELETE command, with no other qualifiers, lists all active backups and each logs. From this list, you can select the backups to deactivate and/or the logs to delete. For database and tablespace backup files and for load copy image files, db2adutl allows several methods to specify what you want to be deactivated. You can:

1. Keep a number, n, of the most recent backup copies using the command qualifier KEEP n.
2. Deactivate all backups older than a certain date by including the phrase OLDER [THAN] *timestamp_value*. The brackets indicate that the word THAN is optional.
3. Deactivate images older than n days with the OLDER [THAN] n DAYS qualifier.
4. Use the phrase TAKEN AT *timestamp_value* to deactivate a specific backup or loadcopy image.

db2adutl DELETE will delete one or more archive log files. To delete a range of log files use DELETE LOGS BETWEEN *log_number1* AND *log_number2*. To delete a single log, DELETE LOGS BETWEEN *log_number1* AND *log_number1* (a "range" of one!).

To be able to delete the log files, verify that:

- The node is defined to ADSM with ARCHDEL=YES.
- If the client options file, dsm.sys, specifies authorized users or groups, be sure that the instance owner (or group) is included.

3.17.4 Extract Option of db2adutl

The EXTRACT command of the utility copies backups, logs, or both from the ADSM server to your current directory. As with DELETE and QUERY, the default without qualifiers is to list all the active images and each log. And, as with DELETE, you can specify a range of images or a specific image to copy to your working directory.

3.17.5 An Example Usage of db2adutl

To illustrate the use of db2adutl, consider the following requirement. Say we want to maintain the two most recent backups and all logs necessary to restore the database from as far back as the oldest retained backup. By using db2adutl DELETE on the backup images that we don't need (all apart from the two most recent) this command will not automatically delete the log files that predate the latest two backups; you must do this as an additional step.

An additional factor is that our database is spread over 16 database partitions on 4 SP nodes. Each database partition has its own log files. So each database partition will have to be queried individually to find out which log files to delete. If you miss off the NODE parameter, then db2adutl will list all the backup images for all the partitions on the SP node, which is probably not what you want.

We'll start by running a db2adutl query on partition 1:

```
> db2adutl query node 1

Query for database TPCD30

Retrieving full database backup information.
   full database backup image: 1, Time: 19980311173855, Oldest log: S0000011.LOG
   full database backup image: 2, Time: 19980312152249, Oldest log: S0000011.LOG
   full database backup image: 3, Time: 19980312165637, Oldest log: S0000011.LOG
   full database backup image: 4, Time: 19980312171520, Oldest log: S0000011.LOG
   full database backup image: 5, Time: 19980313172308, Oldest log: S0000014.LOG
   full database backup image: 6, Time: 19980320181156, Oldest log: S0000017.LOG

Retrieving log archive information.
   Log file: S0000011.LOG
   Log file: S0000012.LOG
   Log file: S0000013.LOG
   Log file: S0000014.LOG
   Log file: S0000015.LOG
   Log file: S0000016.LOG
```

We can see that:

- If we delete the first four backup images, then the log files before S0000014.LOG can also be deleted.
- Tablespace and loadcopy output have been removed for clarity.
- As we only have one database defined, we don't need the DATABASE parameter.

Now, if we look at partition 2:

```
> db2adutl query node 2

Query for database TPCD30

Retrieving full database backup information.
   full database backup image: 1, Time: 19980311175452, Oldest log: S0000025.LOG
   full database backup image: 2, Time: 19980312152434, Oldest log: S0000025.LOG
   full database backup image: 3, Time: 19980312171621, Oldest log: S0000025.LOG
   full database backup image: 4, Time: 19980313172459, Oldest log: S0000033.LOG
   full database backup image: 5, Time: 19980320181514, Oldest log: S0000035.LOG

Retrieving log archive information.
   Log file: S0000025.LOG
   Log file: S0000026.LOG
   Log file: S0000027.LOG
   Log file: S0000028.LOG
   Log file: S0000029.LOG
   Log file: S0000030.LOG
   Log file: S0000031.LOG
   Log file: S0000032.LOG
   Log file: S0000033.LOG
   Log file: S0000034.LOG
```

We can see that:

- The log sequence numbers are much higher than on partition 1. This is because partition 1 holds only the System Catalogs.
- The numbering of the backup images does not necessarily agree across the partitions. In this case, the backup image numbered 5 (on partition 2) correlates with the backup image numbered 6 (on partition 1). We can tell this by the timestamp of 19980320nnnnnn, which means March 20, 1998. There is only one backup image for each partition on this date.
- If we delete the first three backup images, then the log files before S0000033.LOG can also be deleted.
- The output does not identify the partition number.

Note that db2adutl queries the local logging information on each SP node. So if we need to run db2adutl query node 5, we must run this from the SP node which holds database partition 5, namely tp3an05.

To delete all the backup images, apart from the last two on partition 1:

```
> db2adutl delete full keep 2 node 1
```

```
Query for database TPCD30

Retrieving full database backup information.
   full database backup image taken at: 19980312171520, Sessions used: 1
   full database backup image taken at: 19980312165637, Sessions used: 1
   full database backup image taken at: 19980312152249, Sessions used: 1
   full database backup image taken at: 19980311173855, Sessions used: 1

   Do you want to deactivate these backup images (Y/N)? y

      Are you sure (Y/N)? y
```

If we now run `db2adutl QUERY` on partition 1:

```
> db2adutl query node 1

Query for database TPCD30

Retrieving full database backup information.
   full database backup image: 1, Time: 19980313172308, Oldest log: S0000014.LOG
   full database backup image: 2, Time: 19980320181156, Oldest log: S0000017.LOG

Retrieving log archive information.
   Log file: S0000011.LOG
   Log file: S0000012.LOG
   Log file: S0000013.LOG
   Log file: S0000014.LOG
   Log file: S0000015.LOG
   Log file: S0000016.LOG
```

Note that the numbers assigned to the backup images change. The backups known as 5 and 6 before the delete are now known as 1 and 2. In fact the backup image numbers are assigned sequentially, oldest first, and change if any backup images are added or deleted.

The logs S0000011.LOG to S0000013.LOG are now not needed and can be deleted:

```
> db2adutl delete logs between S0000011.LOG and S0000013.LOG node 1 without
prompting

Query for database TPCD30

Retrieving log archive information.
   Log file: S0000011.LOG
   Log file: S0000012.LOG
   Log file: S0000013.LOG
```

Note that we used the `WITHOUT PROMPTING` clause here; so no confirmation messages were displayed.

Another `db2adutl QUERY` on partition 1 to check:

```
> db2adutl query node 1

Query for database TPCD30

Retrieving full database backup information.
```

```
      full database backup image: 1, Time: 19980313172308, Oldest log: S0000014.LOG
      full database backup image: 2, Time: 19980320181156, Oldest log: S0000017.LOG

   Retrieving log archive information.
      Log file: S0000014.LOG
      Log file: S0000015.LOG
      Log file: S0000016.LOG
```

To complete the job for the 16 database partitions in our example, as each partition has different log sequence numbers, these steps (query, delete backup images, delete unnecessary log files) must be repeated for each partition. Care must be taken to run the `db2adutl` command from the SP node that holds each database partition.

Here are some example scripts which perform this function. The delfullandlogsall script calls "delfullandlogs once for each SP node. At each SP node, for the database partitions which are defined there, delfullandlogs deletes the log files to which precede the last two backup images, then deletes all but the last two backup images.

```
# script: ~/bin/delfullandlogsall

rsh tp3an01 ". ~/.profile;~/bin/delfullandlogs 1"
rsh tp3an05 ". ~/.profile;~/bin/delfullandlogs 5"
rsh tp3an09 ". ~/.profile;~/bin/delfullandlogs 9"
rsh tp3an13 ". ~/.profile;~/bin/delfullandlogs 13"
```

```
# script : ~/bin/delfullandlogs

# $1 is the first of the four partitions on the host

a=$1
b=‘echo $1 + 1|bc‘
c=‘echo $1 + 2|bc‘
d=‘echo $1 + 3|bc‘

for n in $a $b $c $d
do

echo "DB2NODE=$n"

db2adutl query node $n | awk ’/Oldest/ {print $10}’ \
         | cut -c1-12 > /tmp/logs.$n

firstdelname=‘head -1 /tmp/logs.$n‘

firstdel=‘echo $firstdelname|cut -c2-8‘

keep=‘tail -2 /tmp/logs.$n | tail -2 | head -1 | cut -c2-8‘

lastdel=‘echo $keep - 1|bc‘

lastdelname=‘echo $lastdel | awk ’{ printf("S%07d.LOG\n",$1) }’‘

#### check that we have logs to delete

if [[ $firstdel -lt $lastdel ]] then
   db2adutl delete logs between $firstdelname and $lastdelname \
            node $n without prompting
fi

##### delete all but the last 2 backups

db2adutl delete full keep 2 node $n without prompting

db2adutl query node $n

done
```

If for any reason some of the individual partition backups fail when doing a overall database backup, then the practice of keeping the two latest backups will still guarantee the integrity of the database. That is, the backup images do not have to be have been generated from the same `db2_all db2 backup` command. We can roll forward as long as we have all the log files available. However, the recovery time will be severely impacted if some partitions have to roll forward more logs than others.

3.18 Using ADSM to Query Archived UDB Log Files

If enabled, the UDB db2uext2 user exit is called when a log file fills. The exit creates an ADSM archive object named /<db_alias>/NODEnnnn/Syyyyyyy.LOG, where *nnnn* is the database partition number and *yyyyyyy* is the log sequence number. For example, suppose DB2 UDB EEE log file S0000016.LOG in partition 3 of database alias TPCD30 fills. The database manager calls db2uext2, which in turn calls ADSM to create an archive file named /TPCD30/NODE0003/S0000016.LOG.

An ADSM administrator can use the SQL feature of ADSM Version 3 to query all or a subset of ADSM archive objects. For example, to list all archived log files for database alias TPCD30:

```
adsm>  select * from archives where filespace_name='/TPCD30'
  ANR2963W This SQL query may produce a very large result table, or may require a
  significant amount of time to compute.

  Do you wish to proceed?y

       NODE_NAME: TP3AN01.PPD.POK.IBM.COM
  FILESPACE_NAME: /TPCD30
            TYPE: FILE
         HL_NAME: /NODE0003/
         LL_NAME: S0000016.LOG
       OBJECT_ID: 3075
    ARCHIVE_DATE: 1998-03-11 13:19:39.000000
           OWNER: db2inst1
     DESCRIPTION: Log file for DB2 database TPCD30
      CLASS_NAME: UDB_MGMT_CLASS

       NODE_NAME: TP3AN01.PPD.POK.IBM.COM
  FILESPACE_NAME: /TPCD30
            TYPE: FILE
         HL_NAME: /NODE0003/
         LL_NAME: S0000017.LOG
       OBJECT_ID: 3076
    ARCHIVE_DATE: 1998-03-11 13:25:13.000000
           OWNER: db2inst1
     DESCRIPTION: Log file for DB2 database TPCD30
      CLASS_NAME: UDB_MGMT_CLASS

  adsm>
```

3.19 Tuning Considerations for ADSM on the RS/6000 SP

More often than not, performance tuning seems a black art. An ADSM environment is no exception. ADSM is a client/server application. Consequently, a tuning effort involves the server host, the client hosts, and the network in between. This section attempts to remove some of the mystery by providing a starter-set of recommendations. As always, the normal caveat applies: What's good for one system is not necessarily good for another.

Because ADSM is probably not the only process running on a host, some of these values may not be reasonable in your environment. As with any tuning effort, consult with your own network and system tuning specialists.

Basically, all tuning can be reduced to a small set of items: real memory, I/O, network bandwidth (really just another form of I/O), and CPU power. You figure out (most likely by being told) the critical applications. These are your loved ones. Then you take care of them, especially when demand for resource exceeds the supply. To give to one, you must take from another.

For ADSM, to improve performance, consider the TCP/IP `no` command options, ADSM server configuration parameters (dsmserv.opt), ADSM client options (dsm.sys and dsm.opt), and SP switch adapter parameters (`RPOOLSIZE` and `SPOOLSIZE`).

The options used for this project are listed in “TCP/IP Options Script (tuning.cust)” on page 127.

Use of these sample programs is not mandatory. You may choose to create your own user exit program, perhaps using one of the sample programs as a model. Useful information can be found in comments at the start of each sample program. Appendix J of the *DB2 UDB Administration Guide* describes a number of considerations that apply to calling a user exit program for archiving and retrieving log files. The guide also describes the database manager format for calling the exit program on a UNIX operating system, and it explains the specific return codes that the exit can provide back to the calling database manager.

3.20 Scripts Used in the Test Configuration

The scripts listed in this section perform the following tasks:

- Setup the ADSM Policy Domains (see 3.20.2, “TCP/IP Options Script (tuning.cust)” on page 127.)
- Configure TCP/IP Options

3.20.1 Policy.mac

```
/* ------------------------------------------------------------*/
/*  S E T   U P   P O L I C Y   D O M A I N S                  */
/* ------------------------------------------------------------*/

/* ------------------------------------------------------------ */
/*                  austin_domain_01                            */
/* ------------------------------------------------------------ */

copy domain standard austin_domain_01
```

```
update domain austin_domain_01                                -
    description="Custom domain for Austin"                    -
    backretention=30  archretention=365
commit

update mgmtclass austin_domain_01 standard standard           -
    description="austin_domain_01 standard management class." -
    spacemgtechnique=none
commit

update copygroup austin_domain_01 standard standard standard  -
    type=backup destination=austin_tapepool_01                -
    frequency=0 verexists=2 verdeleted=1                      -
    retextra=30 retonly=60                                    -
    mode=modified serialization=shrdynamic

update copygroup austin_domain_01 standard standard standard  -
    type=archive destination=austin_tapepool_01               -
    frequency=cmd retver=15                                   -
    mode=absolute serialization=dynamic
commit

copy mgmtclass austin_domain_01 standard standard              -
    udb_mgmt_class

update mgmtclass austin_domain_01 standard udb_mgmt_class      -
    desc="Management class for DB2 UDB backups and log archives"
commit

update copygroup austin_domain_01 standard                     -
    udb_mgmt_class standard                                    -
    type=backup destination=austin_tapepool_01                 -
    frequency=0 verexists=1 verdeleted=0                       -
    retextra=0 retonly=60                                      -
    mode=modified serialization=shrdynamic

update copygroup austin_domain_01 standard                     -
    udb_mgmt_class standard                                    -
    type=archive destination=austin_tapepool_01                -
    frequency=cmd retver=365                                   -
    mode=absolute serialization=dynamic
commit

assign defmgmtclass austin_domain_01 standard standard

validate policyset austin_domain_01 standard

activate policyset austin_domain_01 standard

commit

/* ------------------------------------------------------------ */
/*                   austin_domain_02                           */
/* ------------------------------------------------------------ */

copy domain standard austin_domain_02

update domain austin_domain_02                                 -
    description="Custom domain for Austin"                     -
    backretention=30  archretention=365
commit
```

```
update mgmtclass austin_domain_02 standard standard           -
    description="austin_domain_02 standard management class." -
    spacemgtechnique=none
commit

update copygroup austin_domain_02 standard standard standard  -
    type=backup destination=austin_tapepool_02                -
    frequency=0 verexists=2 verdeleted=1                      -
    retextra=30 retonly=60                                    -
    mode=modified serialization=shrdynamic

update copygroup austin_domain_02 standard standard standard  -
    type=archive destination=austin_tapepool_02               -
    frequency=cmd retver=15                                   -
    mode=absolute serialization=dynamic
commit

copy mgmtclass austin_domain_02 standard standard              -
    udb_mgmt_class

update mgmtclass austin_domain_02 standard udb_mgmt_class      -
    desc="Management class for DB2 UDB backups and log archives"
commit

update copygroup austin_domain_02 standard                     -
    udb_mgmt_class standard                                    -
    type=backup destination=austin_tapepool_02                 -
    frequency=0 verexists=1 verdeleted=0                       -
    retextra=0 retonly=60                                      -
    mode=modified serialization=shrdynamic

update copygroup austin_domain_02 standard                     -
    udb_mgmt_class standard                                    -
    type=archive destination=austin_tapepool_02                -
    frequency=cmd retver=365                                   -
    mode=absolute serialization=dynamic
commit

assign defmgmtclass austin_domain_02 standard standard

validate policyset austin_domain_02 standard

activate policyset austin_domain_02 standard

commit

/* ------------------------------------------------------------ */
/*                   austin_domain_03                           */
/* ------------------------------------------------------------ */

copy domain standard austin_domain_03

update domain austin_domain_03                                 -
    description="Custom domain for Austin"                     -
    backretention=30  archretention=365
commit

update mgmtclass austin_domain_03 standard standard            -
    description="austin_domain_03 standard management class." -
    spacemgtechnique=none
commit

update copygroup austin_domain_03 standard standard standard  -
```

```
    type=backup destination=austin_tapepool_03               -
    frequency=0 verexists=2 verdeleted=1                     -
    retextra=30 retonly=60                                   -
    mode=modified serialization=shrdynamic

update copygroup austin_domain_03 standard standard standard  -
    type=archive destination=austin_tapepool_03              -
    frequency=cmd retver=15                                  -
    mode=absolute serialization=dynamic
commit

copy mgmtclass austin_domain_03 standard standard             -
    udb_mgmt_class

update mgmtclass austin_domain_03 standard udb_mgmt_class     -
    desc="Management class for DB2 UDB backups and log archives"
commit

update copygroup austin_domain_03 standard                    -
    udb_mgmt_class standard                                   -
    type=backup destination=austin_tapepool_03                -
    frequency=0 verexists=1 verdeleted=0                      -
    retextra=0 retonly=60                                     -
    mode=modified serialization=shrdynamic

update copygroup austin_domain_03 standard                    -
    udb_mgmt_class standard                                   -
    type=archive destination=austin_tapepool_03               -
    frequency=cmd retver=365                                  -
    mode=absolute serialization=dynamic
commit

assign defmgmtclass austin_domain_03 standard standard

validate policyset austin_domain_03 standard

activate policyset austin_domain_03 standard

commit

/* ------------------------------------------------------------ */
/*                  austin_domain_04                            */
/* ------------------------------------------------------------ */

copy domain standard austin_domain_04

update domain austin_domain_04                                -
    description="Custom domain for Austin"                    -
    backretention=30  archretention=365
commit

update mgmtclass austin_domain_04 standard standard           -
    description="austin_domain_04 standard management class." -
    spacemgtechnique=none
commit

update copygroup austin_domain_04 standard standard standard  -
    type=backup destination=austin_tapepool_04                -
    frequency=0 verexists=2 verdeleted=1                      -
    retextra=30 retonly=60                                    -
    mode=modified serialization=shrdynamic

update copygroup austin_domain_04 standard standard standard  -
```

```
    type=archive destination=austin_tapepool_04              -
    frequency=cmd retver=15                                  -
    mode=absolute serialization=dynamic
commit

copy mgmtclass austin_domain_04 standard standard             -
    udb_mgmt_class

update mgmtclass austin_domain_04 standard udb_mgmt_class     -
    desc="Management class for DB2 UDB backups and log archives"
commit

update copygroup austin_domain_04 standard                    -
    udb_mgmt_class standard                                   -
    type=backup destination=austin_tapepool_04                -
    frequency=0 verexists=1 verdeleted=0                      -
    retextra=0 retonly=60                                     -
    mode=modified serialization=shrdynamic

update copygroup austin_domain_04 standard                    -
    udb_mgmt_class standard                                   -
    type=archive destination=austin_tapepool_04               -
    frequency=cmd retver=365                                  -
    mode=absolute serialization=dynamic
commit

assign defmgmtclass austin_domain_04 standard standard

validate policyset austin_domain_04 standard

activate policyset austin_domain_04 standard

commit
```

3.20.2 TCP/IP Options Script (tuning.cust)

This is the tuning.cust file.

```
#!/bin/ksh
# Module: <tuning.cust>
#-------------------------------------------------------------------
# Description: This script is a sample which is designed to be used by the
#       user as a guide to change the network options for TCP/IP and UDP/IP
#       communications on the SP system
#
#   These settings are only a sample and should be changed to the preferred
#   settings for your system.
#
#   For further recommendations on tuning, refer to the PSSP Administration
#   Guide
#-------------------------------------------------------------------
#
#   Set default mbuf pool size for all TCP/IP and thread buffer
#
   /usr/sbin/no -o thewall=65536
#
#   Set default sb_max size for all IP connections
#
#  /usr/sbin/no -o sb_max=163840
   /usr/sbin/no -o sb_max=6000000
#
#   Set default TCP send window size for all TCP/IP connections
```

```
#
#  /usr/sbin/no -o tcp_sendspace=65536
   /usr/sbin/no -o tcp_sendspace=655360
#
#   Set default TCP receive window size for all TCP/IP connections
#
#  /usr/sbin/no -o tcp_recvspace=65536
   /usr/sbin/no -o tcp_recvspace=655360
#
#   Set default UDP send window size for all UDP/IP connections
#   Effective maximum is limited to 65536
#
   /usr/sbin/no -o udp_sendspace=65536
#
#   Set default UDP receive window size for all UDP/IP connections
#
   /usr/sbin/no -o udp_recvspace=655360
#
#   Set rfc1323 feature on for faster throughput for High Performance Networks.
#
   /usr/sbin/no -o rfc1323=1
#
#   Set default Maximum Transmission Unit size for transfers across networks.
#   (Value equals MTU size minue size of TCP header (=52 bytes))
#
#  /usr/sbin/no -o tcp_mssdflt=1500
   /usr/sbin/no -o tcp_mssdflt=1448       # Optimized for ethernet
#  /usr/sbin/no -o tcp_mssdflt=4300       # Optimized for FDDI
```

Chapter 4. DB2 UDB EEE High Availability using HACMP

In this chapter, we explain how to configure DB2 UDB EEE for high availability using HACMP on an RS/6000 SP. A real example is given, namely a four high node (8-way) SP with a 30 GB TPCD database spread over 16 DB2 UDB EEE database partitions.

4.1 Overview of HACMP

High-Availability Cluster Multi-Processing (HACMP) is used to provide high availability services on the RISC System/6000. A set of systemized, shared resources are utilized which cooperate to guarantee essential services.

HACMP/6000 Version 3 for AIX Version 3.2.5 and HACMP for AIX Version 4.2 and AIX Version 4 both support up to eight processors in a cluster. DB2 UDB EEE configurations running under HACMP can exceed these limits by spanning multiple HACMP clusters. SP nodes are defines as cluster nodes to HACMP.

4.1.1 Components of HACMP

High Availability Cluster Multi-Processing/6000 Version 4.2 has two distinct subsystems, which can be purchased together or separately.

- The high availability subsystem provides a highly available environment on a cluster of RS/6000s. This provides the function to allow independent nodes, each running separate applications and accessing separate data, to provide failure protection for each other.
- The loosely coupled multi-processing or Concurrent Resource Manager (CRM) subsystem allows two or more machines to concurrently access the same data and run the same application, also providing failure protection for each other.

The mode of operation of DB2 UDB EEE suits the high availability subsystem of HACMP.

4.1.2 Considerations for High Availability of DB2 UDB EEE

Today, DB2 is a critical part of many businesses. Once a DB2 system is in production, its consistent availability can become key to a business. A reduction of system availability will incur costs to the business (either directly or indirectly), taking one or more of these forms:

- **Direct Revenue loss**

If system availability has a direct effect on a business's ability to take revenue, then revenue will be lost while the system is down.

- **Staff Productivity**

  The staff dependent upon a system will often be unable to perform any useful work when a system is not available. This tends to induce frustration in the users, causing them to think of the system as an impediment to their work rather than an aid.

- **Customer satisfaction**

  If a business transaction cannot be performed because the system is unavailable, customers may use an alternative source. Many potential customers for a particular service may also base their choice of a provider on their service level. Lower service levels generally lead to lower levels of customer satisfaction and sometimes the loss of customers.

- **Circumvention costs**

  When a system and its supported business processes are unavailable, it may be possible to provide an alternative solution, rather than simply denying service altogether. There will probably be additional costs associated with such solutions.

4.1.3 Points of Failure

For an HACMP/6000 cluster to be effective, single points of failure should be eliminated. A single point of failure exists when a critical cluster function is provided by a single component. If that component fails, the function can no longer be provided, and an essential service becomes unavailable.

Examples of cluster components which are potential single points of failure are:

- SP nodes
- Power sources
- Network adapters and networks
- Disk adapters and disks

4.1.3.1 SP Nodes

SP nodes can be eliminated as a single point of failure by having standby SP nodes ready to take over their workload should they fail. These standby SP nodes can be configured with HACMP/6000 to behave in three different ways:

1. Idle Standby. A standby SP node can be provided which will take over the work of a failed SP node. When the failed SP node is fixed and

reintegrated into the cluster, it will reclaim its resources. The standby SP node must have access to all resources required for the provision of the essential services: disks, networks and so on. This method results in a cluster which will lose no performance after a failure, as long as the standby SP node has the same capacity. The disadvantage is the cost since the standby SP node is not used except after an SP node failure.

2. Rotating Standby. A standby SP node is provided to take over the work of a failed SP node, as in the idle standby scenario. However, when the failed SP node is reintroduced, it does not reclaim its resources, but becomes the new standby machine. This configuration has the same cost characteristics as idle standby. Its advantage is that it avoids the impact of the originally failed SP node by not reclaiming its resources when it rejoins the cluster.
3. Mutual Takeover. There are no standby SP nodes; all SP nodes are utilized in a normal state. After an SP node failure, the failed SP nodes resources and essential services are taken over by one of the surviving SP nodes in addition to its normal services. This method uses hardware resources more efficiently because redundant SP nodes are not used. The disadvantage is that there may be performance degradation after an SP node failure.

4.1.3.2 Power Sources

Either uninterruptable power supplies, dual power sources or both should be used.

4.1.3.3 Networks

Secondary networks should be available to cope with the failure of the primary network for the SP nodes connected to the external network. Standby network adapters are used to cope with an SP node failure in a mutual takeover or hot standby cluster. After an SP node failure, the SP node that takes over the failed nodes services will be required to have two network addresses: its own network address and the address of the failed SP node. To do this, two adapters are required for the machine on the same physical network. The standby adapter can also take over the same machine's primary address if the primary network adapter of the machine fails. In an SP cluster, it is not necessary to use double switch adapters in each node. HACMP should be configured to react to a switch adapter failure as a complete node failure. This also allows the use of thin nodes as part of the HACMP cluster.

The following TCP/IP networks are supported in an HACMP/6000 environment:

- Ethernet

- Token-Ring
- Fiber Distributed Data Interchange (FDDI)
- Asynchronous Transfer Mode (ATM)
- IBM Serial Optical Channel Converter (SOCC)

It is strongly recommended that all nodes sharing an external disk have a point-to-point, non-TCP/IP connection with each node sharing the disk. Should TCP/IP fail, HACMP/6000 will still be able to connect using the non-TCP/IP connection to prevent unwanted disk takeovers.

The following options are available:

- Raw RS232 link
- SCSI-2 differential bus

4.1.3.4 Disks

Disks can be attached to more than one SP node. This allows a disk to be accessed by a takeover SP node. Data held on disks other than RAID disks should be mirrored to prevent loss of data after a disk failure. To ensure no single point of failure, these mirrors should be accessed through different adapters. Disks that are supported include external SCSI disks and enclosures, 9333 serial disk subsystems, and various RAID subsystems.

- SCSI. SCSI-2 differential disk subsystems should be used. A chain of SCSI disks can connect to up to four nodes. Each disk is only "owned'' by one node at a time. This ownership can be transferred after a failure. This solution has the benefit of lower cost, but access is relatively slow. Because access time is usually important, SCSI disks are rarely used in HACMP configurations.
- 7133 Serial Disk Subsystems. These provide increased system throughput, disk capacity and performance. With the SSA loop design, a single cable failure will not cause loss of access to data. If there is a failure on a loop, the SSA adapter will automatically continue accessing the devices in a non-loop configuration. Once the path is restored, the adapter will automatically reconfigure to resume the normal mode of operation. If there is a disk failure, the hot-swappable disks can be removed without loss of communication between the adapter and the other disks on the loop. Similar automatic reconfigurations will set things back to normal once the disk is repaired and/or replaced. The 7133 is also designed with hot-swappable disk modules, redundant power supplies and fans which can be replaced without affecting system operations. The 7133 provides superior performance due to its greater bandwidth for

multiple concurrent, full duplex I/O operations. The 7133 can connect to eight nodes with Release 2.1 (and later releases) of HACMP/6000.

- 7135 RAIDiant Array. This is a disk array controller with an SCSI-2 differential fast/wide host interface and multiple SCSI buses which have attached disk drives. The array supports RAID levels 0, 1, 3, and 5. Levels 1, 3 and 5 offer data redundancy; so mirroring is not necessary.
  - Under level 0, each group of disks is used without any form of data redundancy. This provides the most storage, but does not offer high availability.
  - With level 1, all disks are mirrored on a one-for-one basis within a group. This provides the fastest read access and the best performance in case of a failure, but it is costly.
  - For level 3, one disk within the group is used for parity, and the other disks hold the data. The parity disk contains sufficient information about all the other disks in the group to be able to rebuild any data after the loss of a single disk. After a disk failure, no data is lost, but the time taken to access the data increases considerably.
  - Level 5 is similar to level 3, except the parity data is striped across all disks within the group. This provides faster access than level 3. Level 5 provides fast write access, but read access is slower than level 1. It has very good price/performance characteristics.

  Level 1 and level 5 are the two RAID levels which should be considered for DB2 UDB EEE. If cost is an issue, then level 5 is recommended. If optimum performance is needed, and this must be maintained after a failure, then level 1 should be considered.
- 7137 High Availability External Disk Array - This provides functionality similar to the 7135 RAIDiant Array. It only contains one controller, but allows hot standby disks to be configured. After a disk failure, the data from the failed disk will be re-created onto the standby disk. If a second disk failure occurs, then the parity disk can be used to provide continuous service. Only RAID-5 and RAID-0 are supported. RAID-0 does not provide availability. The 7137 stripes the data sequentially across the full set of disks.

4.1.4 Example Scenarios

An SP cluster is a group of RISC System/6000 machines connected through at least one internal network. If a high-performance (HPS) switch is used as an internal network, then data can be sent at a rate of up to 100 MB/sec in both directions between any pair of SP nodes.

There are numerous configurations possible with HACMP/6000 and DB2 UDB EEE. Any implementation will have specific requirements which may well dictate the properties of the implemented HACMP cluster. Two examples of networked clusters are given in this section which will provide good solutions in many cases. The first example should provide excellent price/performance, but will suffer degradation after a failure. The second example is a more costly solution, but performance after a failure should suffer little degradation.

4.1.4.1 Mutual Takeover Scenario

The first example demonstrates the use of a mutual takeover configuration:

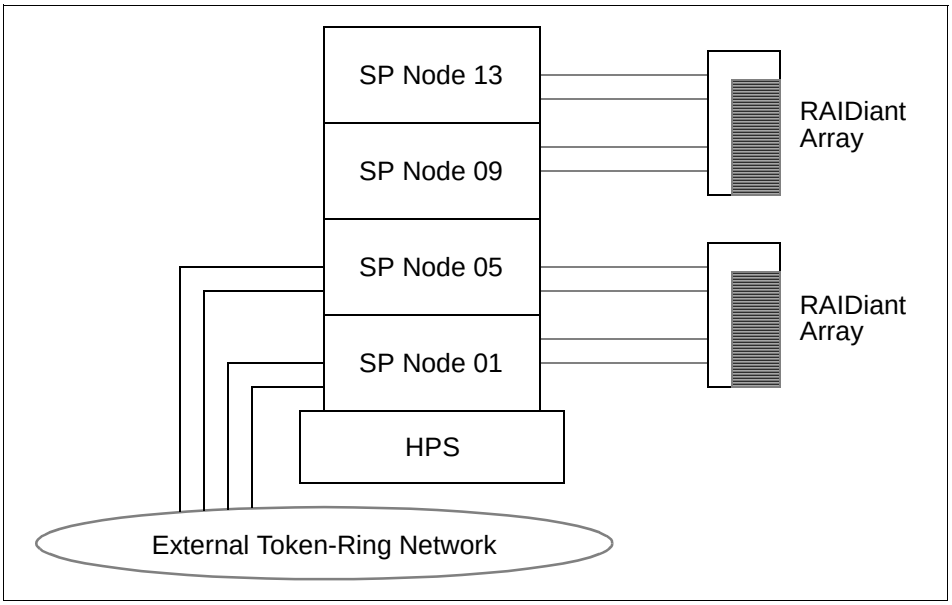


Figure 12. HACMP Mutual Takeover Scenario

This is a HACMP Mutual Takeover Cluster that uses RAIDiant arrays as the storage method for the database. The SP nodes are connected through a primary 16 Mb token-ring to the external network. The HPS and the internal Ethernet network provide communication between the SP nodes. The token-ring network has standby adapters to allow for takeover on nodes 01 and 05.

The connections to the RAIDiant arrays should be duplicated in each machine to protect against a SCSI adapter failure or SCSI cable failure. RAID level 5 would be implemented in the disk groups.

The cluster consists of two mutual takeover pairs of machines. Each pair has access to a shared set of RAIDiant arrays. In normal use, some of the disks will be used by one of the nodes and some by the other. If a node fails, its partner will take over the rest of the RAIDiant array disks and the primary logical and hardware IP address and will start up the failed node's DB2 UDB EEE processes. For example, after a failure of Node 05, Node 01 will take over the resources (disks, IP address) from Node 05 and start the database partitions originally running on Node 05.

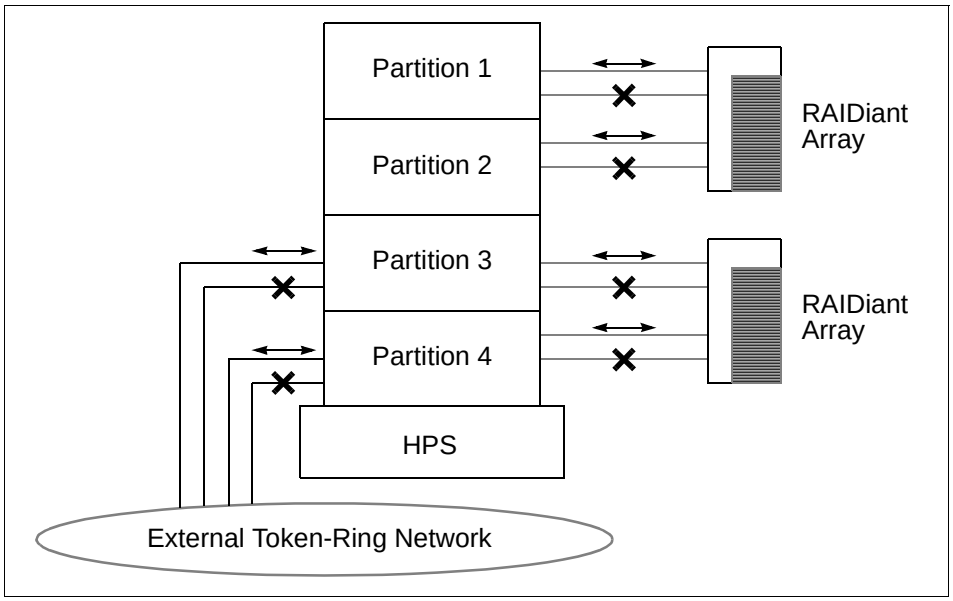


Figure 13. HACMP Mutual Takeover before Takeover

After an SP node failure, the database will be running on three SP nodes instead of four. This will probably result in a high degradation in performance.

The standby adapter will be used to provide TCP/IP services by the takeover node with the same IP service address used normally by the failing node.

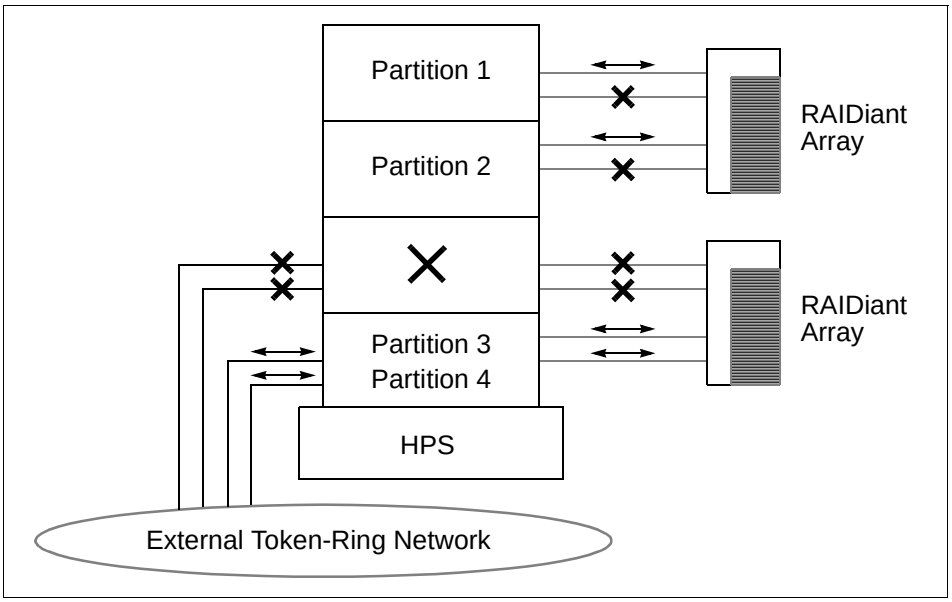


Figure 14. HACMP Mutual Takeover after Takeover

In this case, Node 09 and Node 13 (holding database partitions 1 and 2) are used for the DB2 EEE engine only and are not used as coordinator nodes since they do not have an external network.

After a disk failure, the RAIDiant arrays will continue to provide uninterrupted access to data. However, access time will increase significantly since the data will have to be extracted using the parity information.

This configuration provides a system that is very cost-effective. During normal use, the database system is configured to take full advantage of the SP nodes. However, after a failure, although the database will continue to be available, the performance is likely to degrade significantly.

4.1.4.2 Rotating Standby Scenario

The first example demonstrates the use of a rotating standby configuration:

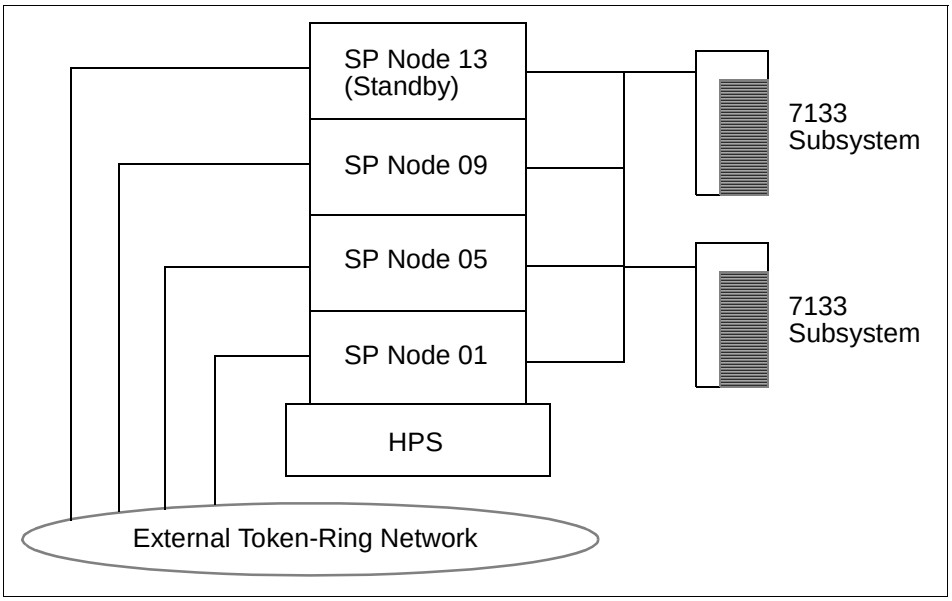


Figure 15. HACMP Rotating Standby Scenario

This is a HACMP Standby Cluster example where a cluster uses a 7133 high-performance serial disk subsystem as the disk storage. The nodes are connected through a primary 16 Mb token-ring network and the HPS.

Three IP addresses and three database partitions are defined for the cluster, and the first three nodes to be started will take these resources. The last node will become a standby node for these resources. All the SP nodes are connected to all the 7133 subsystems to allow them to access the correct disks for their database partitions.

The data held on the 7133 subsystems should be mirrored to protect against failure. These mirrors should be across different 7133 subsystems and adapters to ensure no single point of failure.

Raw RS232 null modem connections could be connected between all nodes. These non-TCP/IP connections would be used to prevent a node from attempting to take over disks after a failure of TCP/IP. The serial ports must be provided using eight serial port adapters because SP nodes do not have serial ports.

After an HPS adapter failure, the cluster would be configured to run a node_down script to allow the instance to move to a standby node.

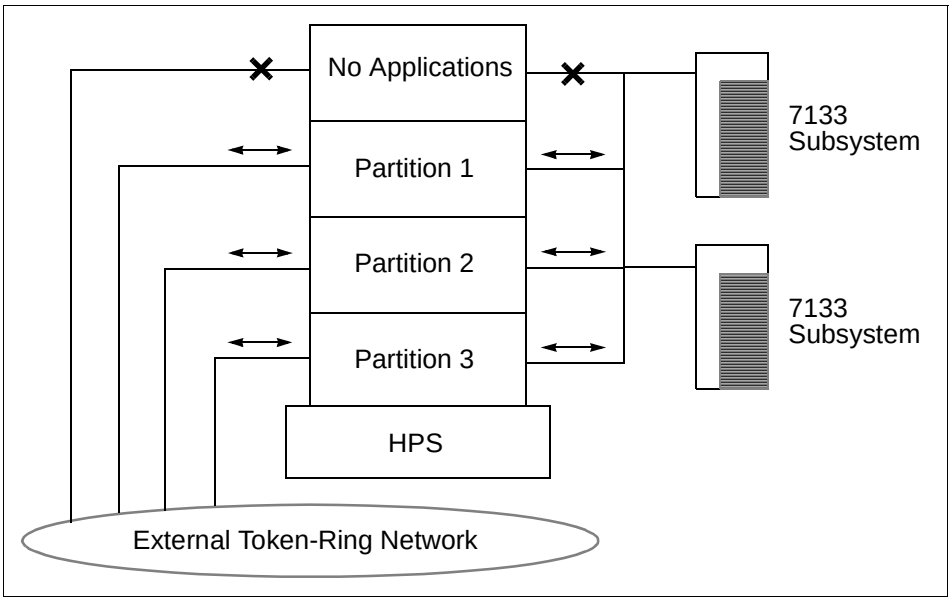


Figure 16. HACMP Rotating Standby before Takeover

This example illustrates a configuration that should provide a high level of performance. It should also provide a system which has little or no degradation of performance after a failure.

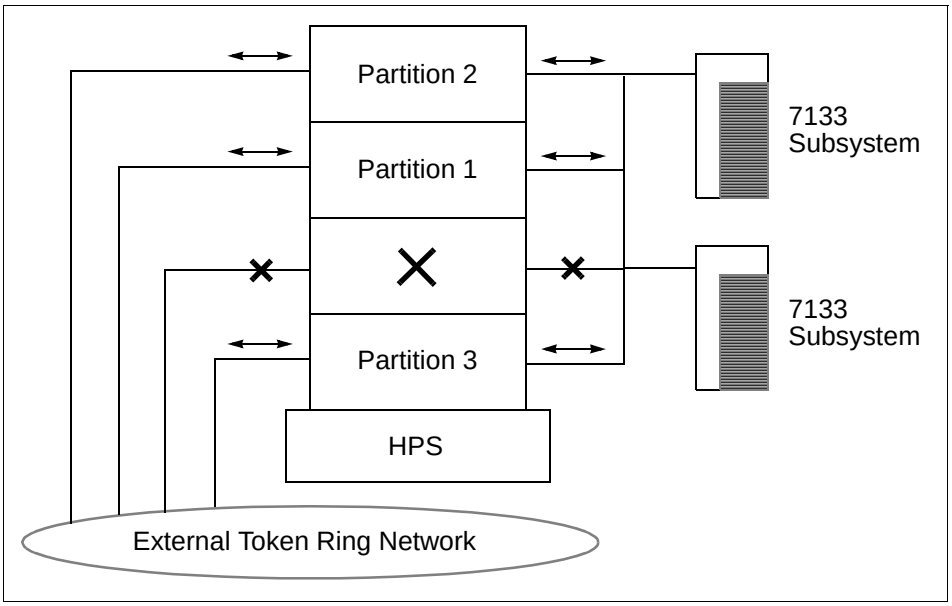


Figure 17. HACMP Rotating Standby After Takeover

As you can see in the figure above, the system does not need to return to the original configuration. When the failing node is repaired, it becomes the standby node that will be used in case of failure.

When using HACMP/6000 with DB2 UDB EEE and your system experiences a failure, the HACMP software will execute. By using HACMP/6000, you can take over resources such as disk. A machine could serve as a standby SP node in the event of an SP node failure. The HACMP software will cause the following to occur:

- Takeover of the network address
- Takeover of some file systems (user-defined)
- Takeover of volume groups used for raw table spaces
- Execute a user-defined script

4.2 High Availability for DB2 UDB EEE on RISC/6000 SP

Generally speaking, the goal of this chapter is to list and suggest alternatives to those special issues raised by using DB2 UDB EEE on the RISC/6000 SP. In this sense, the emphasis will be on High Availability Cluster Multi-Processing (HACMP) from a DB2 UDB EEE perspective.

After reviewing the hardware and software configuration used for the tests, we provide an overview of the available design alternatives, then cover HACMP installation and configuration to work with DB2 UDB EEE.

The approach followed in this chapter is to start from an existing standard RISC/6000 SP environment with DB2 UDB EEE installed and to detail the changes necessary in order to configure HACMP on the existing RISC/6000 SP system. However, there are customers who may want to include planning for HACMP prior to the installation of the RISC/6000 SP and DB2 UDB EEE. Those readers interested in the latter approach should refer to the redbook *An HACMP Cookbook,* SG24-4553.

4.2.1 Hardware Configuration

The configuration used for DB2 UDB EEE and HACMP testing consists of a four-high node RISC/6000 SP (see Figure 18) with eight processors and 2 GB of memory per high node. The nodes are interconnected via a High-Performance Switch (HPS). In addition, the standard administrative Ethernet (SP Ethernet) connects the nodes to the RS/6000 Control Workstation.

Nodes node01 and node05 shown in Figure 18 have two Ethernet adapters each connected to allow TCP/IP address takeover in case of failure, and will be used as a DB2 coordinator nodes. The other nodes, node09, and node13 will be used for the DB2 engine only, without connection to an external network.

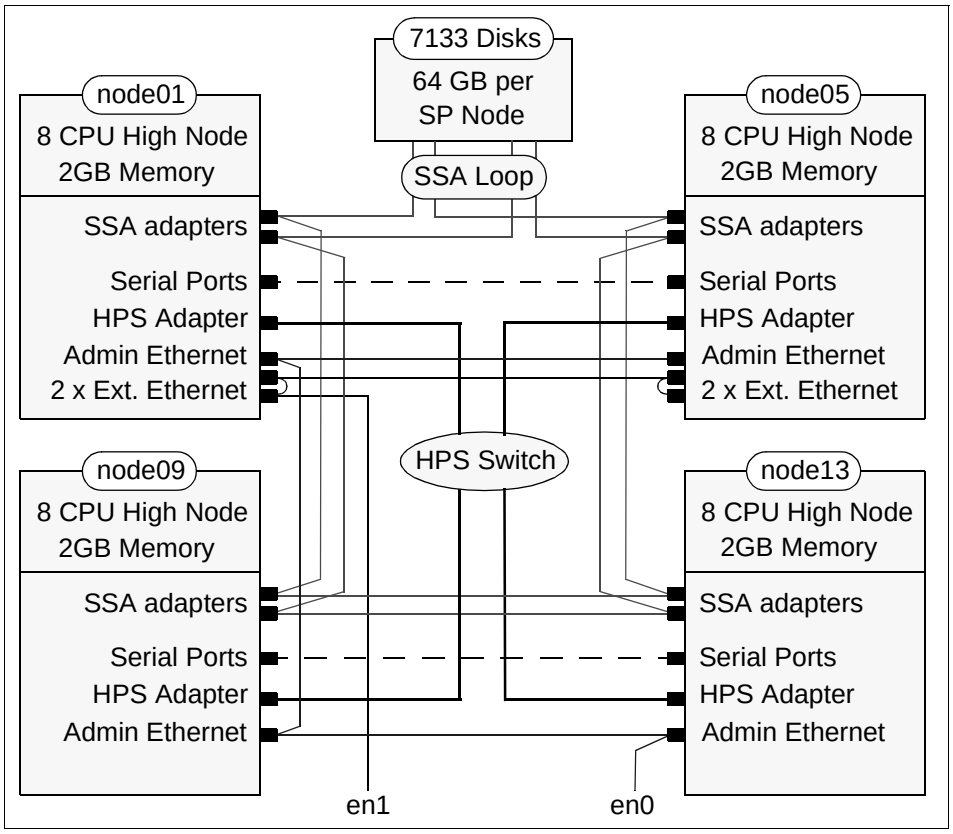


Figure 18. Hardware Configuration Used for the Test

Serial storage (7133) units are connected to the four SP nodes in a loop configuration that allows access to disks from any node. There are a total of 64 x 4 GB disks in two independent loops of 32 disks, making a total of 256 GB of serial disk.

These disks are used to store any critical data, such as DB2 UDB EEE table data, that needs to be protected against hardware failures. To protect against disk failure, we use mirroring, and to protect against cable failure in the SSA loop, we use a double loop.

Each node has an 8-port serial adapter connected with null modem cables between pairs of nodes (node01 to node05 and node09 to node13). These connections are used to carry the HACMP heartbeat, so that TCP/IP is not a single point of failure.

If you are using 7135 RAIDiant disks, for a two-node HACMP cluster with a 7135 disk unit, five cables and terminators are required, including two 52G7348 Y-cables, one 52G7349 system-to-system cable, and two 52G7350 terminators. The two resistive terminators on each SCSI-2 Differential adapter have to be removed.

If you are using Target Mode SCSI for the TCP/IP heartbeat, then before the SCSI adapters can be cabled together, it is necessary to check that they have different SCSI identifiers by choosing an appropriate value for the Adapter card SCSI ID field in the menu displayed by the SMIT `chgscsi` command. Since each device (adapter) on an SCSI bus needs to have a unique identifier and target mode SCSI connects two adapters together, the two adapters cannot have the same identifier. It is recommended to select a value other than 7 for each adapter's SCSI identifier.

4.2.2 Installed Software

The Control Workstation and all the SP nodes are using AIX V4.2.1 with PSSP V3.1, with HACMP V4.2.2 on the SP nodes only. Because HACMP V4.1.1 and HACMP V3.1.1 have similar functionality to HACMP V4.2.2, the descriptions in this chapter apply to these releases also. The only difference is that some SMIT short paths have different names in older versions of HACMP. The version we used (V4.2.2) has also some graphical administrative tools that will not be covered here, since they provide a lesser degree of functionality than SMIT.

If you experience security problems or high delays during cluster synchronization, make sure that the latest available HACMP PTF has been applied.

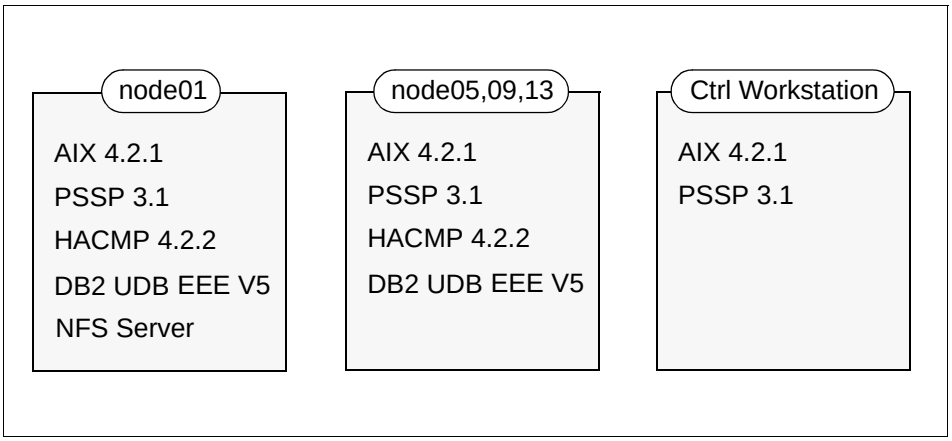


Figure 19. Software Configuration Used for the Test

4.2.3 Network Interfaces

The four node's host names are hnode01, hnode05, hnode09, and hnode13. The digits 01, 05, 09, and 13 are used because a high node uses four slots in the SP2. It is recommended to number the SP nodes in this way to allow you to easily map a host name to a slot in the SP. These names are the ethernet (en0) network interface names. For the HPS, the network interface names are switch1 to switch4. These are the names of the switch adapters in the SDR (System Data Repository) of the SP. IP address takeover through HACMP uses one switch adapter per node. As a result, it is not possible to use the SDR switch names for IP address takeover. Each switch adapter has an alias, called sw01, sw05, sw09, and sw13, corresponding to switch01, switch05, switch09, and switch13. Finally, four switch boot addresses, swboot01, swboot05, swboot09, and swboot13 are assigned. All these network interface names are defined on the same subnet because the SP switch does not use standby addresses. In our configuration, only the ethernet standby adapters will use a different subnet. For a summary of available network interfaces on each node, see Figure 20 on page 144.

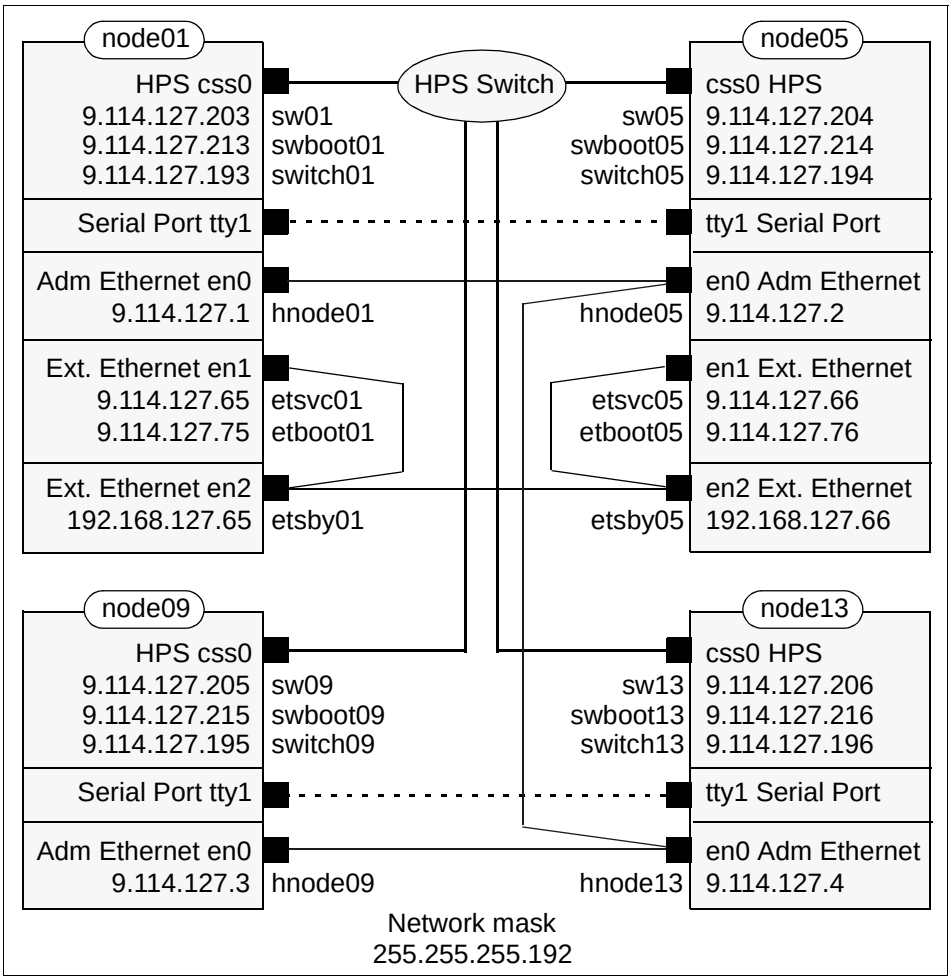


Figure 20. Network Interfaces Available in Each Node

4.2.4 DB2 UDB EEE Configuration

DB2 UDB EEE is installed on each node. The instance owner is called db2inst1 and belongs to the dbadmin1 group. There are four database partitions defined in each SP node, making a total of 16 in all. The instance owner's home directory is made available to the other nodes using NFS. If you use AMD (Automounter) to manage the instance owner's home directory, you risk intermittent failures.

One database is defined, called TPCD30, and the catalog node for this database is node01. Eight tables are defined in this database: CUSTOMER,

LINEITEM, NATION, ORDERS, PART, PARTSUPP, REGION, and SUPPLIER.

The LINEITEM table is the largest with 180,036,450 rows and, as shown in Table 2, is defined in the TS_DAT_BIG table space which exists at all database partitions apart from the catalog partition. The other tables, apart from REGION and NATION, are defined in the TS_DAT_MED table space, which also exists at all database partitions apart from the catalog partition. The REGION and NATION tables are defined in the TS_LIT table space and only use the catalog partition.

Table 2. Tables Defined in the TPCD30 Database

Table Name	Table Space (data)	Nodegroup	Partitions
LINEITEM	TS_DAT_BIG	NG_BIG	2-16
ORDERS	TS_DAT_MED	NG_BIG	2-16
PARTSUPP	TS_DAT_MED	NG_BIG	2-16
CUSTOMER	TS_DAT_MED	NG_BIG	2-16
PART	TS_DAT_MED	NG_BIG	2-16
SUPPLIER	TS_DAT_MED	NG_BIG	2-16
NATION	TS_LIT	NG_LIT	1
REGION	TS_LIT	NG_LIT	1

Nodegroup NG_BIG is defined across all partitions except the catalog partition. The NG_LIT nodegroup uses only the catalog partition.

DB2 UDB EEE uses the sw01, sw05, sw09, and sw13 switch network addresses for inter-node communications in the db2nodes.cfg configuration file.

A more detailed explanation of the database physical layout can be found in “Designing and Implementing the DIsk Space Allocation” on page 8.

4.3 HACMP Takeover for this Configuration

In this section, we explain, step by step, how HACMP manages the takeover of a failed SP node.

We will use the symbols shown in Figure 21. DPxx corresponds to the four database partitions running on the SP node nodexx. When we move the

DP01 symbol, for instance to node05, this means that the DB2 EEE processes used to manage these four database partitions are running on node05.

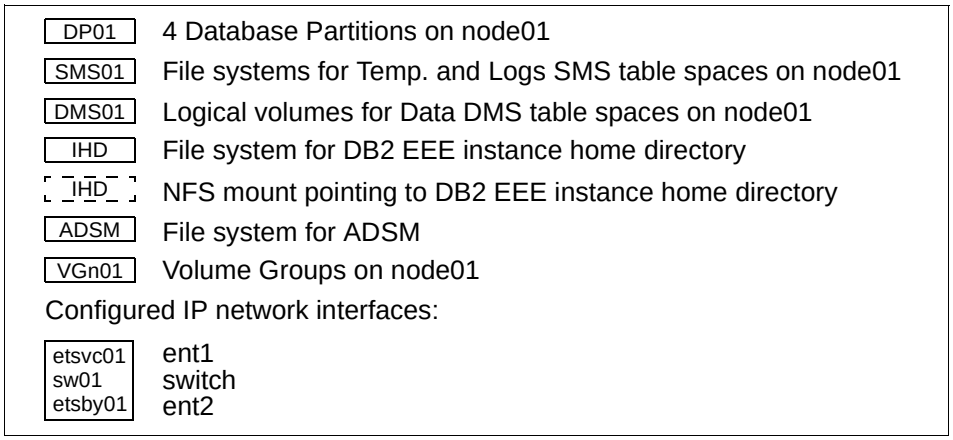


Figure 21. Symbols Used in HACMP

In Figure 22, we show the test machine in normal usage with four database partitions on each SP node. For example, node01 has database partitions 1 to 4, node05 the next four and so on.

Each SP node uses two volume groups for DB2 EEE shown as VGnxx. For example, node01 uses volume groups vg_n01_01 and vg_n01_02 (shown as VGn01).

In these volume groups, there are four DMS raw device containers for data table spaces and four more for indexes per database partition, except for partition 1 which is reserved for database catalogs and very small tables. The symbol DMS01 corresponds to all the DMS table spaces of database partition 1–in this case 3 x (4+4)= 24 containers or raw logical volumes.

There are two file systems used as SMS containers for temporary table spaces and one file system used as SMS container for logs per database partition, shown as SMSxx, giving a total of 12 on each SP node.

The database instance owner’s home directory file system is shown as IHD in a dotted rectangle when it is mounted remotely through NFS.

Current active IP network interfaces are shown also.

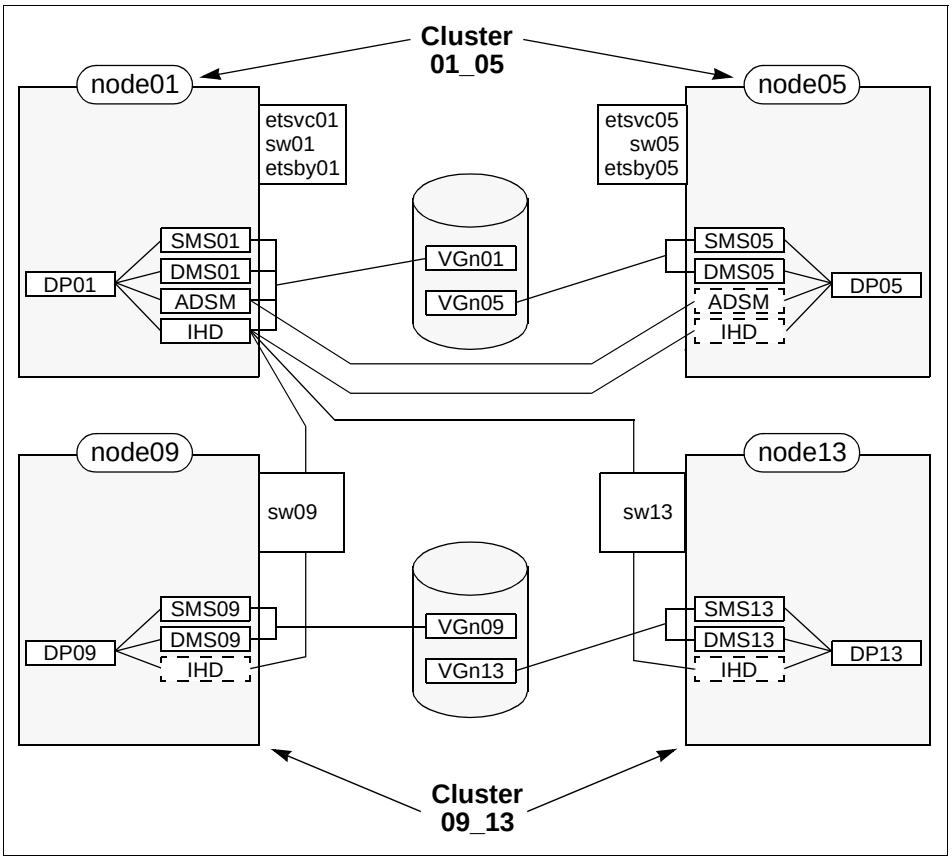


Figure 22. Configuration before HACMP Takeover

Now let's say that SP node node01 loses power:

- Since node01 holds the instance owner's home directory,all the HACMP clusters are affected.
- After some seconds (depending on the type of networks involved), the node01 failure is detected in cluster01_05, and node node05 initiates the takeover procedure.
- The ent2 adapter on node node05 is configured to the etsvc01 address to replace the external TCP/IP services of node01.
- In the switch interface of node05, an alias for sw01 will be configured to replace the switch TCP/IP services of node01.

- The VGn01 volume groups are varied on by node05 to gain access to the disks used by the database partitions (1-4) which were running on node01.
- A file system check (fscs) of all the SMS01 file systems is performed, and these file systems are mounted.
- The instance owner's home directory is mounted locally by HACMP in node05. Since this SP node uses the alias sw01 for the switch, the SP nodes in the HACMP cluster cluster09_13 regain access through NFS to IHD.
- We use an HACMP start script to start DB2 UDB EEE database partitions on node05. Once this script completes, we have access to all the database partitions in the DB2 UDB EEE database.

If we compare the configuration before and after HACMP takeover, we notice that node05 experiences some performance degradation because it has eight database partitions to manage.

The configuration after HACMP takeover is shown in Figure 23.

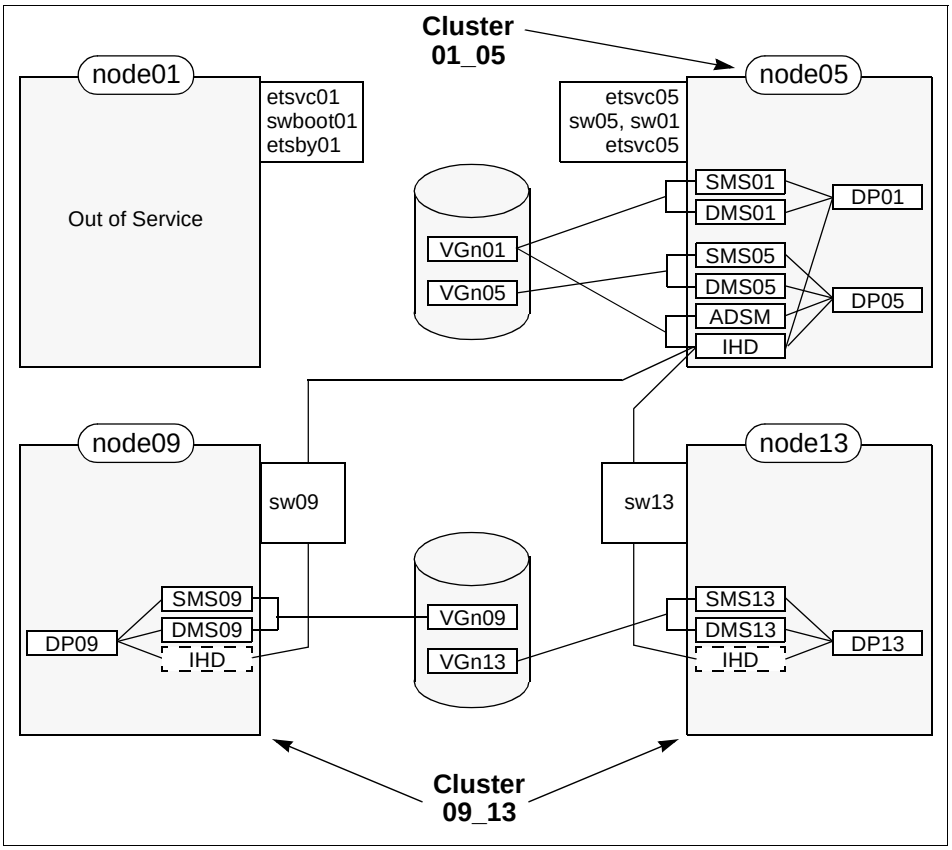


Figure 23. Configuration after HACMP Takeover of node1 by node5

When node01 is booted successfully, it will use the boot address swboot1 to avoid any conflict with node node05, because node05 is still using the address of node01.

When we choose (either manually or automatically) to return to normal operation, the database partitions designated by DP01 are stopped by another script that we wrote to stop DB2 EEE database partitions (see 4.11.2, “Start and Stop DB2 UDB EEE” on page 211).

Then, on node05, HACMP does the following:

- The SMS01, IHD and ADSM file systems are unmounted.
- The VGn01 volume groups are varied off.
- The adapter ent2 is reconfigured to use etsby05, and switch alias sw01 is removed.

Then, on node01:

- An alias sw01 is configured for the switch.
- The adapter ent1 is changed to use etsvc01.
- The VGn01 volume groups are varied on.
- The SMS01, ADSM and IHD file systems are mounted.
- The IHD is made available through NFS from node01 to the other SP nodes.

Now we are once more in the normal operating condition.

Figure 24 shows the configuration after a HACMP takeover of node09 by node13:

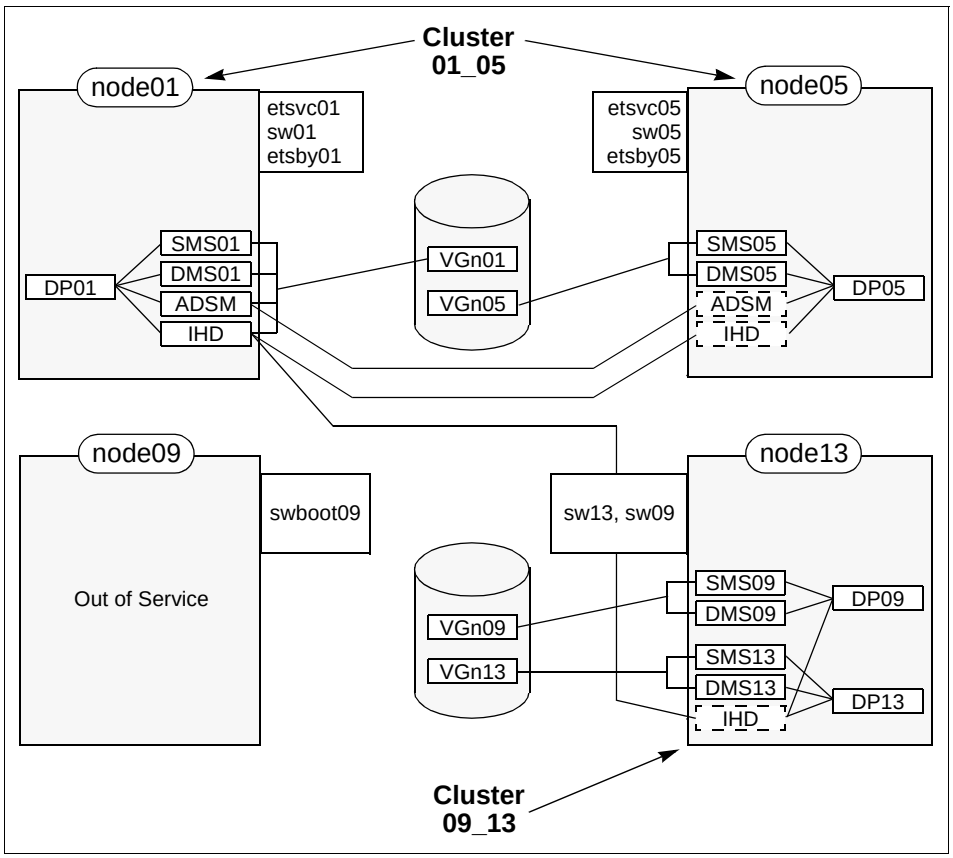


Figure 24. Configuration after HACMP Takeover of node9 by node13

4.4 HACMP Considerations for DB2 UDB EEE

This section provides guidelines on configuring a HACMP environment involving DB2 UDB EEE on the RS/6000 SP.

4.4.1 SP Ethernet Considerations

According to HACMP guidelines, the SP Ethernet should not be used for IP address takeover since this could conflict with the node information stored in the SDR (System Data Repository) on the Control Workstation.

4.4.1.1 Non-IP Network Considerations

Even though it would be possible to use the SP Ethernet and the HPS as the only network for HACMP heartbeat, it is not recommended to do so. In case of TCP/IP failure on one node, several nodes may be trying to acquire the same shared resources at the same time because of TCP/IP failure. Most customers will probably want to protect themselves against this type of problem, making the use of a non-IP link for heartbeat a necessity. This link can either be a serial RS-232 connection between nodes, requiring multiport serial adapters in each node, or a target mode SCSI connection between nodes. Actually, it is highly recommended to use two networks for heartbeat, one being a non-IP network, the other being the SP Ethernet. When the `Estart` command is issued, which happens each time the HPS needs to be restarted on one of the nodes and in particular for each HACMP takeover and node reintegration, the switch becomes momentarily unavailable. For HACMP, if the SP Ethernet was not included in the configuration, this would mean that there would be no TCP/IP link available during Estart processing. To avoid possible problems caused by this, the SP Ethernet is included in the HACMP configuration as backup heartbeat network, which guarantees an TCP/IP communication between nodes at all times.

4.4.2 DB2 UDB EEE Executables

DB2 UDB EEE executables are located in /usr/lpp/db2_05_00. There are two ways to make these files available on all nodes:

- Install DB2 UDB EEE on each SP node
- Export DB2 UDB EEE executables using NFS or AFS

It is recommended by the HACMP manuals to install the DB2 code on SP nodes because:

- The code may have keys to activate the software related to the processor of the node.

- If an upgrade of the DB2 code is related to a AIX upgrade, you will have to do this node by node.

In our example configuration, we installed DB2 EEE on each node.

If DB2 is installed on one node and NFS is used to make the code available to the other nodes, this can make software maintenance easier since DB2 UDB EEE upgrades can be done on one node only. In this case, for availability reasons, the exported file system should be placed on external shared disk hardware. Having binaries accessed through NFS does not impact performance in most cases (except for the first access to the executables).

In both cases, if the instance owner’s home directory is not available, it is not possible to access the DB2 executables since access to these files goes through soft links in the instance owner's home directory.

4.4.3 Cluster Size

HACMP supports cluster sizes ranging from two to eight nodes. If you are sizing the cluster for use with DB2 UDB EEE, then at the hardware, configuration, and maintenance level, there is probably no significant advantage in using small versus large clusters. Since it is possible to have multiple independent pairings of resource groups within one cluster, it might even be more convenient to manage large eight-node clusters on the RISC/6000 SP. There are several drawbacks associated with two-node clusters:

- If both nodes fail, there is no backup node. However, the probability of this happening is minimal.
- In a mutual takeover configuration, when one node fails, the surviving node must support twice its usual load, possibly leading to performance degradation.
- Since two-node clusters are usually associated with cascading resources, database operations are interrupted twice: first when a node fails and a second time when it reintegrates into the cluster and the surviving node needs to release the failing node's resources.

Large HACMP clusters can be configured to eliminate these problems, as will be seen in the next section. But, when using large HACMP clusters, extreme care must be exercised when configuring the resources under the control of HACMP. For example, if you have a large amount of disk in each node, HACMP takeover after the failure of a node involves a resync of all the disks, and recovery time may become unacceptably long.

4.4.4 Standby Nodes or Mutual Takeover

The question of using standby nodes versus mutual takeover is linked to the previous section about cluster size. In small clusters, it is usually too expensive to waste up to 50 percent of the computing resources by configuring standby nodes. In this case, it is more sensible to implement mutual takeover with cascading resources. For large clusters, however, it can be advantageous to use rotating resources, allowing for one standby node. In this case, it is possible to avoid the problem of degraded performance and double database operation interruption associated with two-node clusters.

For these reasons, we should consider two types of HACMP configurations for DB2 UDB EEE. For smaller clusters, consider two-node clusters in mutual takeover setups with cascading resources. For larger clusters, consider four- to eight-node clusters with one standby node and rotating resources. The last issue is how many DB2 UDB EEE database partitions should be configured per RISC/6000 SP node.

4.4.5 DB2 UDB EEE Database Partitions per SP Node

The *DB2 UDB V5 Administration Guide,* S10J-8157, on page 865, provides several examples of DB2 UDB EEE environments involving HACMP. From an HACMP perspective, implementations using two or more database partitions per SP node allow you to minimize the performance degradations normally associated with mutual takeover configurations.

If you have four DB2 UDB EEE database partitions per SP node, when a SP node fails, two database partitions could be restarted on two different SP nodes, leading to a 50 percent load increase on each takeover node, instead of 100 percent in a setup with one database partition per SP node.

Another point to consider is whether DB2 UDB EEE databases are well suited to having two or more database partitions per SP node. It is recommended to configure 1 database partition per two CPUs in an SMP SP node. So, in our example using eight CPU high nodes, we have configured four database partitions.

The catalog partition is accessed more frequently than the other database partitions, which can lead to performance problems in some environments. For this reason, consider keeping the catalog partition free of user tables.

4.4.6 Instance Home Directory Considerations

The management of the instance home directory can cause problems when configuring HACMP with DB2 EEE. In order to have access to the DB2

commands, you need access to the instance home directory. Each time that one node in a cluster does takeover of the instance home directory, all the DB2 database partitions will wait to execute DB2 commands until the instance home directory is made available through NFS from the takeover node. This means that any applications using data from these database partitions will be stopped during the HACMP takeover.

Also, when the takeover node gains access to the volume group where the instance home directory resides, HACMP must destroy the old NFS mount pointing to the instance home directory and replace it with a local mount. To be able to perform the NFS unmount, the database partitions running on the takeover node must be stopped.

4.4.6.1 A Very Important Tip

There is a way to simplify the management of the instance home directory. The key is make all the nodes equal and always mount the instance home directory across NFS. You may see some performance degradation because the database partitions on the SP node that holds the instance home directory will use NFS to access this directory across the switch, when really it is locally available. The gain is in making the whole system more available if an SP node fails.

Here is an overview of the steps:

- Make a file system with a name different from instance home directory in a shared volume group. For example: /db2home.
- Then, if you mount /db2home across the switch as sw01:/home/db2inst1 (in our example), any database partition will access /db2home using NFS.
- Now if you need mount or umount /db2home as part of a takeover, HACMP does not need to stop any database partitions. The database partitions will wait for the instance home directory to be made available through NFS.

The scripts in “NFS-Mounting /home/db2inst1 in cluster_09_13” on page 198 are needed on each node to mount and umount sw01:/home/db2inst1 after HACMP has started.

Here is what happens when a takeover occurs:

- As soon as /db2home is mounted as a local file system on the takeover node, it is NFS exported.
- The sw01 network interface is made available.
- The access to sw01:/home/db2inst1 is available.

- The database partitions which have been migrated to the takeover node are started.

A minor change to the HACMP script, cl_activate_nfs, is required to make this work. Copy this script to another directory and modify the line:

```
mount -o "$OPTIONS" $HOST:$FS $FS
```

to allow the mount of a different name. Then use the new code in scripts in “NFS-Mounting /home/db2inst1 in cluster_09_13” on page 198.

We will detail the configuration steps to set up an HACMP system where the SP node which holds the instance home directory mounts this file system locally (not using NFS).

4.4.7 Considerations for Our Example

Due the amount of disk involved in our example, we chose a configuration with two clusters of two 8-way SMP SP nodes using mutual takeover in each cluster. We did this in order to have the shortest disk resync time during a takeover.

We chose four database partitions per SP node (two CPUs per database partition) to allow a good balance between I/O intensive operations (backup/restore) and CPU-bound operations.

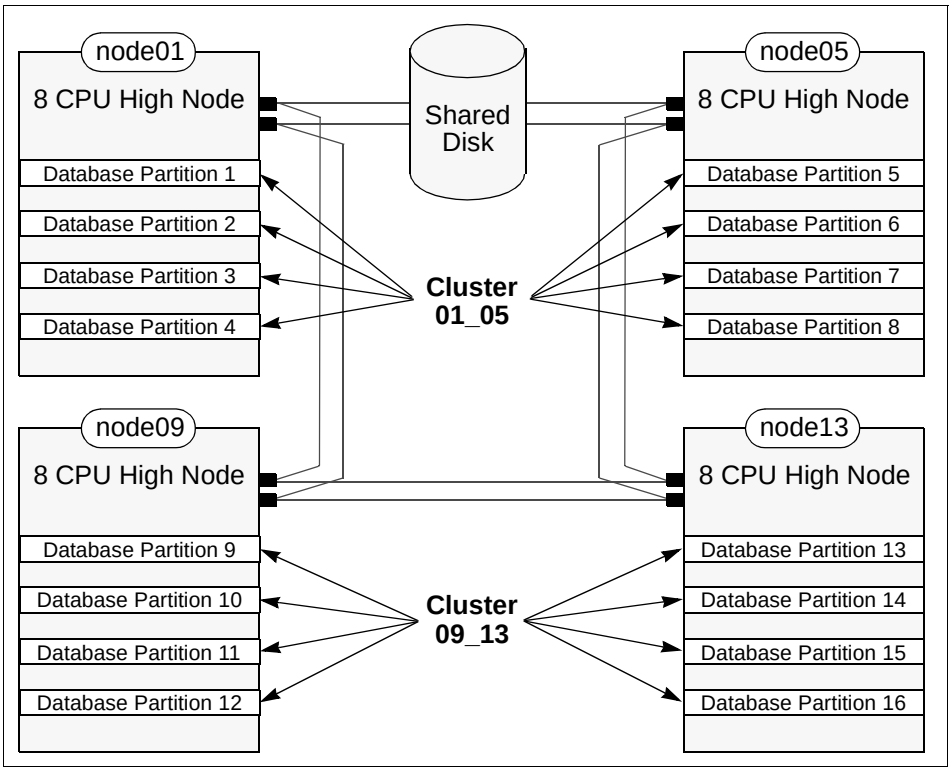


Figure 25. Two HACMP Clusters with Two Nodes Each

4.4.8 Effect of Switch Restart on DB2 UDB EEE

In an HACMP configuration, the `Estart` switch initialization command needs to be issued for each node takeover and each cluster reintegration. When Estart is issued, NFS becomes briefly unavailable, since our test configuration mounts NFS file systems over the switch. Running DB2 UDB EEE transactions are delayed and resumed after Estart has completed. Connections to databases are not lost.

4.4.9 DB2 UDB EEE Behavior in Case of Node Failure

This section gives some information on DB2 UDB EEE behavior when one of the DB2 UDB EEE partitions fails. This corresponds to the case where HACMP is not involved and can serve as a reference for the minimum level of disruption one should aim at when implementing HACMP, as seen from a DB2 UDB EEE viewpoint. In “Takeover of the DB2 Instance Owner’s Home Directory” on page 205, we will compare this best-case behavior to the actual behavior observed in the test configuration.

- **Coordinator Partition Failure**

  When the coordinator partition for a transaction fails, the transaction needs to be restarted, and connection to the database is lost. When the partition becomes available again, the transaction is rolled back. Processing can resume after recovery has occurred, including database roll back and roll forward.

- **Database Partition Failure**

  Queries or updates will hang forever until the missing database partition is restarted and initiates recovery. Then, the user receives a SQL1229 error message (transaction rolled back because of a system failure), and the transaction is rolled back. Refer to “Takeover of the DB2 Instance Owner’s Home Directory” on page 205, for more information about the SQL1229 error message.

- **Database Partition Not Used in Transaction**

  If a partition fails but it is not used in a running transaction, the transaction completes undisturbed. This is the case, for instance, if a transaction accesses a nodegroup that doesn't include all database partitions, and one of the non-used database partitions becomes unavailable.

- **Catalog Partition Failure**

  Any running transactions will receive the SQL1229 (see above) error message when the catalog partition is restarted and initiates recovery. While the catalog partition is down:

  - System Catalog tables cannot be accessed.
  - The redistribute command cannot be run.
  - Other database partitions cannot be restored (because the restore command needs a connection to the catalog partition).
  - Access to static SQL packages is not possible.
  - Issuing DDL (Data Definition Language) statements is not possible. Databases cannot be rolled forward.
  - New connection attempts will fail and return an SQL1229 error message.

  This clearly shows how critical it is to place a special importance on the catalog partition node when implementing HACMP for DB2 UDB EEE.

  With no catalog partition, very little if anything can be done with DB2 UDB EEE. The `connect reset` command will time-out with SQL1475 and system error -25567 (connect reset successful but error occurred during termination). As far as existing connections to databases are concerned,

they will be lost after MAX_CONNRETRIES times CONN_ELAPSE number of seconds, which is 50 seconds by default. CONN_ELAPSE and MAX_CONNRETRIES can be modified through the DB2 `update database manager configuration` command.

4.5 Prerequisite Tasks for Installation of HACMP with DB2 UDB EEE

A number of preparation steps are required before installing HACMP. They include creating and modifying shared volume groups and file systems for HACMP and installing DB2 UDB EEE, among other tasks.

4.5.1 Creating ttys for Serial Null Modem Lines

This step is required if you are using serial lines to carry the HACMP heartbeat. For clusters of two nodes, you need only one tty defined on each node. For larger clusters, you must connect and define enough serial lines to carry the heartbeat between all the nodes of the cluster.

Run `smitty maktty`, and make a tty without login on the port where you connect the null modem cable on each node, as shown in the following screen:

```
                                Add a TTY

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
[TOP]                                                  [Entry Fields]
  TTY type                                             tty
  TTY interface                                        rs232
  Description                                          Asynchronous Terminal
  Parent adapter                                       sa1
* PORT number                                         [s2]                        +
  Enable LOGIN                                         disable                    +
  BAUD rate                                           [9600]                      +
  PARITY                                              [none]                      +
  BITS per character                                  [8]                         +
  Number of STOP BITS                                 [1]                         +
  TIME before advancing to next port setting          [0]                         +#
  TERMINAL type                                       [dumb]
  FLOW CONTROL to be used                             [xon]                       +
  OPEN DISCIPLINE to be used                          [dtropen]                   +
  STTY attributes for RUN time                        [hupcl,cread,brkint,icrn> +
  STTY attributes for LOGIN                           [hupcl,cread,echoe,cs8]
  LOGGER name                                         []
  STATUS of device at BOOT time                       [available]                 +
  TRANSMIT buffer count                               [16]                        +#
[MORE...23]
F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Press PF4 against Port Number and select the port where the null cable is connected. In our example, this is s2.

Or if your configuration is equal in all the nodes, do:

```
dsh mkdev -c tty  -t tty -s rs232 -p sa1 -w s2
```

Replace `sa1` by the name of the parent adapter and `s2` by the port number of your configuration. A tty will be created on each node.

We assume you have set the environment variable, `WCOLL`, to use `dsh` over all the required nodes. Otherwise, use `dsh -a` to send the command to all the nodes.

In our example tty1 was created on each node.

To test the lines use:

```
stty </dev/tty1
```

on the two nodes where the line is connected. When you enter the second command you should see tty data from both commands.

4.5.2 Enabling Target Mode SCSI

We are not using this hardware in our example, but if you are using SCSI disks in your cluster, you could use Target Mode SCSI as an alternative to serial ports.

First, make sure the resistor blocks on the SCSI adapters have been removed and the adapters have different SCSI IDs, as explained in “Hardware Configuration” on page 140. Then, on each node, put the SCSI adapter in the Defined state. The name of the SCSI adapter can be found by typing the following command:

```
lsdev -Cc adapter | grep scsi
```

Assuming that this command returned scsi1, enter:

```
rmdev -l scsi1
```

This will put the adapter in the defined state. To enable target mode SCSI, type `smitty chgscsi`. Select your adapter; then press **Enter**. The following menu appears:

```
           Change / Show Characteristics of a SCSI Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  SCSI Adapter                                        scsi0
  Description                                         SCSI I/O Cont>
  Status                                              Available
  Location                                            00-03
  Adapter card SCSI ID                               [7]                      +#
  BATTERY backed adapter                              no                      +
  DMA bus memory LENGTH                              [0x202000]               +
  Enable TARGET MODE interface                        yes                     +
  Target Mode interface enabled                       no
  PERCENTAGE of bus memory DMA area for target mode  [50]                     +#
  Name of adapter code download file                  /etc/microcode/8d77.44>
  Apply change to DATABASE only                       no                      +
F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Change the Enable Target Mode Interface field to yes, and press **Enter**. The next step is to make the adapter available:

```
cfgmgr
```

You should now see several files in the /dev directory—tmscsin.im for the initiator or sending interfaces and tmscsin.tm for the target or receiving interfaces, with n going from 0 to 6.

After completing the above steps on all nodes, the connection between two nodes can be tested in the following way. As an example, on node09, we would type:

```
cat < /dev/tmscsi0.tm
```

On node13, we can send data to node09 by typing:

```
cat /etc/hosts > /dev/tmscsi0.im
```

The contents of node13 's /etc/hosts file should be displayed on node09.

4.5.3 Creating Our Own File Collection

To simplify the process of changing TCP/IP-related files, we will update the TCP/IP files in the Control Workstation and create a file collection that updates the files in all the nodes using the supper program.

To do this, we create a new file collection called db2.admin. This collection will update the files /etc/hosts, /etc/services and so on in all the nodes.

Execute the script install_db2_coll.ksh (See “Install the db2.admin File Collection” on page 208) with the file db2_coll.list, from the Control Workstation as follows:

```
install_db2_coll.ksh db2.admin ~/db2_coll.file
```

Then copy this script to each node or to a directory common to all nodes, and run it on each node:

```
install_db2_coll.ksh db2.admin
```

To update the collection on a regular basis, add (or modify) in each node a crontab entry periodically. To force propagation of changes from the Control Workstation, do:

```
dsh -a /var/sysman/supper update db2.admin
```

4.5.4 Creating /.rhosts Files on all Nodes

On all nodes, create or add the following lines to the /.rhosts file as user root using your favorite text editor:

```
hnode01 root
hnode05 root
hnode09 root
hnode13 root
switch01 root
switch05 root
switch09 root
switch13 root
sw01 root
sw05 root
sw09 root
sw13 root
swboot01 root
swboot05 root
swboot09 root
swboot13 root
```

This /.rhosts file is required by HACMP for cluster startup, and it is also used by the RS/6000 SP. Make sure that the permissions on the .rhosts file are set to the correct permissions by using the following command:

```
chmod 600 /.rhosts
```

If you have `rsh` permission problems, you can optionally add a line: `+ root`. This will suppress security on remote commands.

If you add this line and you still have problems synchronizing your cluster, remove the line and check the /.rhosts file, the name server definitions and the Kerberos definitions for the TCP/IP network interface names involved in

the cluster. Both TCP/IP and Kerberos are queried by HACMP to find out if you have permission to use rsh, which is used by the synchronization process.

4.5.5 Updating the /etc/hosts File on All Nodes

Even if your environment has been set up to use a name server, it is recommended to include all boot and service adapters in the /etc/hosts file on all nodes to keep the cluster working in the event of the name server being unavailable. For the tested system, we added the following lines to the /etc/hosts in the Control Workstation (refer to Figure 20 on page 144 for an overview of the network setup) and then we updated the file collection file, db2.admin, on each node (see “Creating Our Own File Collection” on page 160).

```
#############################################################
#       Network interface en0 setup
#       9.114.127.0 Network  Netmask=255.255.255.192
#
9.114.127.61   sp-tp3cw.ppd.pok.ibm.com            sp-tp3cw
9.114.127.1    tp3an01.ppd.pok.ibm.com             hnode01
9.114.127.2    tp3an05.ppd.pok.ibm.com             hnode05
9.114.127.3    tp3an09.ppd.pok.ibm.com             hnode09
9.114.127.4    tp3an13.ppd.pok.ibm.com             hnode13
#############################################################
#       Network Interface en1 service Ethernet addresses
#       9.114.127.64 Network  Netmask=255.255.255.192
#
9.114.127.65   tp3an01b.ppd.pok.ibm.com            etsvc01
9.114.127.66   tp3an05b.ppd.pok.ibm.com            etsvc05

#############################################################
#       Network Interface en1 boot Ethernet addresses
#       9.114.127.64 Network  Netmask=255.255.255.19
#
9.114.127.75   etboot01.ppd.pok.ibm.com            etboot01
9.114.127.76   etboot05.ppd.pok.ibm.com            etboot05

#############################################################
##       Network Interface en2 standby Ethernet addresses
#       192.168.127.64 Network  Netmask=255.255.255.19
#
192.168.127.65   etsby01.ppd.pok.ibm.com           etsby01
192.168.127.66   etsby05.ppd.pok.ibm.com           etsby05

#############################################################
#       Network Interface css0 setup - SDR Switch Addresses
#       9.114.127.192 Network  Netmask=255.255.255.192
#
9.114.127.193  tp3sn01.ppd.pok.ibm.com             switch01
9.114.127.194  tp3sn05.ppd.pok.ibm.com             switch05
```

```
9.114.127.195  tp3sn09.ppd.pok.ibm.com              switch09
9.114.127.196  tp3sn13.ppd.pok.ibm.com              switch13
#############################################################
#        Network Interface css0 setup - Service Switch Addresses
#        9.114.127.192 Network  Netmask=255.255.255.192
#
9.114.127.203  tp3sn01s.ppd.pok.ibm.com             sw01
9.114.127.204  tp3sn05s.ppd.pok.ibm.com             sw05
9.114.127.205  tp3sn09s.ppd.pok.ibm.com             sw09
9.114.127.206  tp3sn13s.ppd.pok.ibm.com             sw13
#############################################################
#        Network Interface css0 setup - Boot Switch Addresses
#        9.114.127.192 Network  Netmask=255.255.255.192
#
9.114.127.213  tp3sn01b.ppd.pok.ibm.com             swboot01
9.114.127.214  tp3sn05b.ppd.pok.ibm.com             swboot05
9.114.127.215  tp3sn09b.ppd.pok.ibm.com             swboot09
9.114.127.216  tp3sn13b.ppd.pok.ibm.com             swboot13
```

We add the second alias in order to clarify the function of each IP address, and we use those aliases throughout this chapter.

4.5.6 TCP/IP Definitions

Some of the adapters involved were configured during SP installation (for example: ccs0, en0, en1, tr0, tr1, and so on). Because of some limitations in the PSSP programs, some adapters need to be configured manually.

In this case, we are using an additional Ethernet adapter en2 and we must configure it manually. The best way to do this is by using the script that performs the customize-installation stage, namely /tftpboot/script.cust. In this way, if the node is reinstalled, you will preserve your configuration.

For the additional en2 adapter:

```
##### First setup cable type $CABLETYPE to bnc, dix or tp
##### $ADAPTER= ent2 $INTERFACE= en2
chdev -l $INTERFACE -a state=detach > /dev/null 2>&1
rmdev -l $ADAPTER > /dev/null 2>&1
chdev -l $ADAPTER -a bnc_select=$CABLETYPE > /dev/null 2>&1
mkdev -l $ADAPTER > /dev/null 2>&1
##### Now setup interface
chdev -l $INTERFACE -a netaddr=$IPADDRESS -a netmask=$SUBNETMASK \
-a state=up
##### where $SUBNETMASK is the network mask
##### and $IPADDRESS the ip address.
##### update ODM
chdev -l $INTERFACE -a netaddr=$IPADDRESS
```

You can run `smitty mktcpip` to do the configuration manually.

Because `smitty mktcpip` changes the hostname to the name of the network interface, after you configure en2, use the `hostname` command to restore the original hostname.

```
hostname original_host_name
```

We must do this process for adapter ent2 and interface en2 on nodes node01 and node05 using the standby addresses.

Additionally, HACMP expects that the service adapters will be configured before HACMP starts using its boot address. That means that if you are adding HACMP to a running DB2 EEE installation, you must change the SDR configuration or customization script, to set up boot address and reconfigure the nodes. If you do this manually and the node is reinstalled or recustomized by PSSP software, then the boot configuration will be lost.

4.5.7 Creating Aliases for the Switch

Run the following command on each node to set up service and boot addresses for the switch:

```
ifconfig css0 inet ipaddress netmask 255.255.255.192 alias up
```

where `ipaddress` is the IP address to be added as an alias.

For example, on node01, we ran:

```
ifconfig css0 inet 9.114.127.203 netmask 255.255.255.192 alias up
ifconfig css0 inet 9.114.127.213 netmask 255.255.255.192 alias up
```

This defines a switch service and boot address. It is not necessary define a standby address because we are using aliases. Due to the absence of standby addresses, we do not use a different subnet for switch addresses.

Never put the normal SDR switch address under HACMP control because you will run in conflicts with the PSSP software.

The alias definitions will be lost on reboot. They are only necessary before synchronization, after which time HACMP takes care of the aliases.

4.5.8 Disk Logical Name Definition

First, we use script in “Create Disk Devices” on page 218 to assign disk logical names to our SSA disks.

Because the same script will run on each node, the disk names will be the same on each node. This is very important because it is very easy to make

mistakes if you use the same disk name in another node pointing to a different physical disk.

You should choose the logical name of a disk by using the name of adapter, drawer and position in the drawer. This information will be needed to match the disks to volume groups and make mirroring easier.

In a production environment, you should assign names related to the physical position of the disks in the drawer and adapter. In this way, you could easily find the disk when a failure requires physical inspection of disk. This is important because there are usually many disks in a typical VLDB configuration.

4.5.9 Creating Shared Volume Groups

Each node holds shared volume groups named vg_nii_yy, where ii is the slot number and yy the sequential number of the volume group in the node. For instance, on node01, to create and activate an eight-disk volume group, you could use the command, as root:

```
mkvg -f -y vg_n01_01 -n hdisk2 hdisk3 hdisk4 hdisk5 hdisk6 hdisk7 \
                        hdisk8 hdisk9
varyonvg vg1_n01_01
```

The `-n` option means that the volume group will not be automatically activated at system restart.

If you are planning a HACMP cluster of more than two nodes, because a volume group can be activated by only one SP node at the same time, you must define enough volume groups in each node to allow the volume groups to be varied on in the takeover node. For example, if you have four database partitions in each SP node and you plan to use two SP nodes for takeover relocating two database partitions to one node and two to the other, you will need at least two volume groups on this node. If you define a volume group for each database partition, this will allow you to choose a different takeover node for each database partition.

Additionally, you always should have more than three physical disk in each volume group in order to allow quorum in case of failure of one disk. The maximum permitted number of disks in a volume group is 32. In our example, we chose to have two volume groups of eight disks in each SP node.

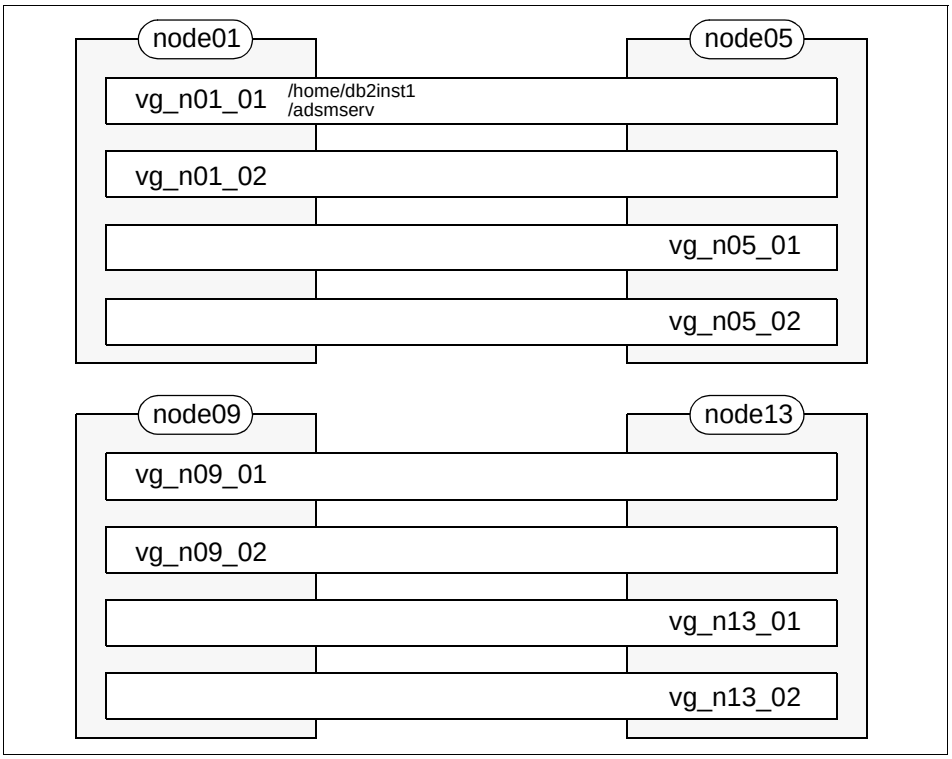


Figure 26. Shared Volume Group and File System Definition

Similar commands would be used on the remaining nodes to create the corresponding volume groups. Refer to Figure 26 for a description of the volume groups on the different nodes.

When configuring HACMP with a very large database, there are many considerations related to disks, volume groups, logical volumes, and so on to keep the HACMP administration manageable.

4.5.9.1 Example of Definition of Volume Groups

An example script can be found in Chapter 4.11.3.3, “Create Volume Groups and Logical Volumes” on page 223.

It is important to choose carefully which disks you will use in each volume group. If you using disk mirroring, the second copy should be in a different drawer from the first copy, so that a power failure in one drawer would only affect one of the two copies.

The set of disks that you choose for the volume group must be connected to a SSA loop to which the owner node and all the involved takeover nodes of this node have access. If you have more than one loop or adapter, you could split the disks across the adapters to optimize the bandwidth of the adapters and the SSA loop.

You should follow a naming convention which results in unique names. We chose to include the node number and a sequential number. Future HACMP requirements could force to you to import the volume group in another takeover node, and to avoid name conflicts, the names must be different.

If your volume group becames corrupted, all the disks involved will be lost; so consider using more volume groups in each node. Remember that mirroring is not allowed between different volume groups.

Use a physical partition (PP) size at volume group (VG) creation compatible with the biggest disk expected to be added to the VG in the future. An 8 MB PP size will be fine for 4 GB disks but not for 9 GB disks. If you need to change this value, you will have to destroy the VG. This task is time consuming if you have to restore data to the volume group.

It very often useful to exclusively dedicate a DB2 EEE database partition to manage the catalog tables and the very small tables. This removes some of the workload from this database partition. In this case, the volume group assigned to this database partition will be smaller.

4.5.9.2 Preserving the Volume Group Major Number for NFS

NFS requests retain the major device number of the volume group when a takeover occurs. To be prepared for current and future NFS configurations, it is a good idea to find the free major numbers on each node, using the `lvlstmajor` command. Then assign to each volume group in the cluster a different major number and force the major volume in the `mkvg` using the `-V` option during creation.

```
mkvg -f -y v vg_n01_01 -s16 -V56 hdisk3 .....
```

Major device number assignment is not provided by the script that we use to make volume groups.

We use another script to deal with major number assignment (see “Synchronize Volume Groups” on page 238). This script also deals with other issues related to an installation already in operation when HACMP is installed, such as NFS conflict definitions, mounted file systems, automount on a shared file system, and DB2 permissions. The `importvg` in the takeover node is made by this script, too.

HACMP comes now with a set of tools called C-SPOC (See the *HACMP Administration Guide*, SC23-1941, Chapter 4). If you use these tools, you could do volume group synchronization using `smit cl_updatevg.hdr`. These tools prevent you from many common mistakes that happen during shared volume group administration tasks.

4.5.10 Creation of Logical Volumes

It is strongly recommended to use a naming convention for logical volumes that results in names that are unique and easy to locate. You could use a combination of volume group name (because you already made this unique in previous step), a logical volume sequential number, and the usage (Journaled File System log, database log, temporary or data, and so on). We used the naming convention shown in Table 3 on page 168:

Table 3. ogical Volumes and Their Usage

LV_name	Usage	DB partitions
lv_nyy_0z_log	jfslog	1-16
lv_n01_01_101	catalog TS (raw DMS)	1
lv_n01_01_102	single NG table space	1
lv_nyy_0z_x03	UDB logfiles (jfs/SMS)	1-16
lv_nyy_0z_x04	UDB temp1 (jfs/SMS/2 drives)	1-16
lv_nyy_0z_x05	UDB temp2 (jfs/SMS/2 drives)	1-16
lv_nyy_0z_x06	UDB large data1 (raw/DMS)	2-16
lv_nyy_0z_x07	UDB large data2 (raw/DMS)	2-16
lv_nyy_0z_x08	UDB large index1 (raw/DMS)	2-16
lv_nyy_0z_x09	UDB large index2 (raw/DMS)	2-16
lv_nyy_0z_x10	UDB small data1 (raw/DMS)	2-16
lv_nyy_0z_x11	UDB small data2 (raw/DMS)	2-16
lv_nyy_0z_x12	UDB small index1 (raw/DMS)	2-16
lv_nyy_0z_x13	UDB small index2 (raw/DMS)	2-16
lv_nyy_0z_114	home/tp3an01 (jfs)	1

LV_name	Usage	DB partitions
where: x=1 for partitions 5, 9, and 13 x=2 for partitions 2, 6, 10, and 14 x=3 for partitions 3, 7, 11, and 15 x=4 for partitions 4, 8, 12, and 16 and yy= 2 digit node number and z=1 for partitions 1, 2, 5, 6, 9, 10, 13, 14 z=2 for partitions 3,4,7,8,11,12,15,16		

We used the script listed in “Create Volume Groups and Logical Volumes” on page 223.

To assign the logical volumes, you must understand the logical design of the database. The creation of logical volumes is very important to the performance and availability of the database. It is often not easy to modify the logical volume definitions once data has been loaded.

To create a logical volume manually, you could used `smit mklv` or the `mklv` command directly:

```
                              Add a Logical Volume

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
  Logical volume NAME                                 [lv_n13_01_106]
* VOLUME GROUP name                                    vg_n13_01
* Number of LOGICAL PARTITIONS                        [150]                    #
  PHYSICAL VOLUME names                               []                      +
  Logical volume TYPE                                 [udb]
  POSITION on physical volume                          middle                 +
  RANGE of physical volumes                            minimum                +
  MAXIMUM NUMBER of PHYSICAL VOLUMES                  []                       #
    to use for allocation
  Number of COPIES of each logical                     2                      +
    partition
  Mirror Write Consistency?                            yes                    +
  Allocate each logical partition copy                 yes                    +
    on a SEPARATE physical volume?
  RELOCATE the logical volume during                   yes                    +
    reorganization?
  Logical volume LABEL                                []
  MAXIMUM NUMBER of LOGICAL PARTITIONS                [512]
[MORE...6]

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

The disk I/O performance of your database will be affected by the position of each logical volume on the disk. Remember that the minimal unit of backup for a database is a table space, where each container points to a logical volume.

We chose to use SMS table spaces for temporary and log table spaces. For these table spaces, we create the logical volume first to ensure the desired placement and name, and then we define a file system over it.

For data and indexes, we chose DMS table spaces. For these table spaces, you only need to define the logical volumes to be used as raw containers. Because the raw devices must be owned by the instance owner, and are lost each time that HACMP activates the volume group, we set the type to udb to allow easy identification of DMS raw devices. See “Start DB2 UDB EEE on each resource” on page 211 for details on how to define DMS containers with the type udb and set permissions before starting DB2.

If you want to use mirroring, set the number of copies to 2.

DB2 EEE will split the data of a table across the containers of the corresponding table space and balance disk usage as long as the raw devices are on different disks.

The log logical volume (LV) is created automatically the first time that you create a file system. If you never create a file system, one easy way to create a log LV is create one file system and drop it, and then rename the log LV according your naming conventions. Another method is create a logical volume with the correct type and name and format it with the `logform` command.

If you already have many file systems, then the log LV will have already created by AIX. In this case, rename the logical volume and after that edit the /etc/filesystems file and change all the references to the old log logical volume name to the new name.

4.5.11 Creating Shared File Systems

Once that logical volumes are defined we need to define a file system for each DB2 instance and another for each SMS container or DMS file container. Raw devices don't need file systems. One additional file system was required for ADSM configuration files.

We installed DB2 EEE on each node. If you choose to install the code on one node and then use NFS to make the DB2 EEE code available to the other nodes, you will need a shared file system for the DB2 code.

On all the nodes, we created file systems for logs and temporary table spaces.

On node01, two additional shared file systems were defined: one to store the instance owner's home directory and another for ADSM configuration files (see Figure 26 on page 166). Shared file systems should not have disk accounting activated.

4.5.11.1 Creating the /home/db2inst1 File System

The /home/db2inst1 file system is used on node01 to store the DB2 instance owner's home directory.

The *Quick Beginnings Guide for DB2 UDB EEE*, on Page 58 under "Prepare for Installation," advises you to use an NFS-mounted file system. It is not recommended to use AMD or Automounter because these utilities can cause mounting or locking problems.

There are several methods of creating the file system, and shown below is the command we used. On node01, this file system was created over the logical volume previously defined, as root:

```
crfs -v jfs -d'lv_n05_01_103' -m /home/db2inst1 -A no -p'rw' -t no -a
frag='4096' -a nbpi='4096' -a ag='8'
```

You should allocate at least 60 MB for /home/db2inst1.

Note that we specified that the file system should not be mounted automatically at system restart (-A no) because we will do that using HACMP. Since the file system will be NFS-mounted on the other nodes over the switch, we need to export it on node01 using the `smit mknfsexp` command:

```
                    Add a Directory to Exports List

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* PATHNAME of directory to export                    [/home/db2inst1]
* MODE to export directory                            read-only
  HOSTS & NETGROUPS allowed client access            [sw05,sw09,sw13]
  Anonymous UID                                      [-2]
  HOSTS allowed root access                          []
  HOSTNAME list. If exported read-mostly             []
  Use SECURE option?                                 no
* EXPORT directory now, system restart or both       both
  PATHNAME of alternate Exports file                 []

F1=Help           F2=Refresh        F3=Cancel         F4=List
F5=Reset          F6=Command        F7=Edit           F8=Image
F9=Shell          F10=Exit          Enter=Do
```

Alternatively, you could use the `mknfsexp` command:

```
/usr/etc/mknfsexp -d /home/db2inst1 -t rw -r sw05,sw09,sw13 -B
```

The main reason we NFS-mount this directory over the switch is not performance, but availability, since IP takeover cannot be configured for the SP Ethernet. As the number of nodes in the configuration increases, the switch will also provide scalability.

On node05, make sure that the /home/db2inst1 directory entry exists. If it doesn't, create it using the command:

```
mkdir /home/db2inst1
```

If the /home/db2inst1 file system is not mounted, mount it first with the command:

```
mount /home/db2inst1
```

Then, on the remaining nodes, create the home directory for DB2 and set the correct permissions:

```
mkdir /home/db2inst1
chown db2inst1.dbadmin1 /home/db2inst1
```

This file system will be made available to the other nodes through NFS across the switch for availability reasons. We will put DB2's home directory on node node01 in the volume group vg_n01_01 with the NFS mount under the control of HACMP to allow file system takeover for node05.

The second cluster (node09 and node13) will have normal NFS mounts with automatic mount. Either use `mknfsmnt`:

```
/usr/sbin/mknfsmnt -f /home/db2inst1 -d /home/db2inst1 -h sw01 -n -B -A
-t 'rw' -w 'bg' -H -Y -Z -X
```

or using SMIT:

```
                          Add a File System for Mounting

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                [Entry Fields]
* PATHNAME of mount point                           [/home/db2inst1]          /
* PATHNAME of remote directory                      [/home/db2inst1]
* HOST where remote directory resides               [sw01]
  Mount type NAME                                   []
* Use SECURE mount option?                          no                       +
* MOUNT now, add entry to /etc/filesystems or both? both                     +
* /etc/filesystems entry will mount the directory   yes                      +
   on system RESTART.
* MODE for this NFS file system                     read-write               +
* ATTEMPT mount in foreground or background         background               +
  NUMBER of times to attempt mount                  []                        #
  Buffer SIZE for read                              []                        #
  Buffer SIZE for writes                            []                        #
  NFS TIMEOUT. In tenths of a second                []                        #
F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Due the fact that we are not using AMD, the Control Workstation is not involved in the DB2 home directory operation. This means also that it is not necessary to install High Availability Control Workstation (HACWS) to protect the control workstation against failure. Since we are not using HACWS, we will attempt to become independent of the control workstation by eliminating possible points of failure associated with it.

4.5.11.2 Creating File Systems for Log Files and Temporary TS

The next step is to create file systems for:

- Database log files
- SMS directory containers for the temporary tablespaces

When defining these file systems, we need to select different names on each node. Otherwise, it would not be possible to import the shared volume groups between two nodes, a step required by HACMP. The naming convention we used for the database log file systems is:

```
(database path)/(instance name)/NODE00(database partition number)
```

with database partition number ranging from 01 to 16. So, for partition 5, the log files we be stored in:

```
/DB_LOG/db2inst1/NODE0005
```

To create this file system on node05 (since this SP node holds partition 6):

```
crfs -v jfs -d'lv_n05_01_103' -m /DB_LOG/db2inst1/NODE0005 -A no -p'rw'
-t no -a frag='4096' -a nbpi='4096' -a ag='8'
```

The remaining file systems should be created accordingly on node01, node05, node09, and node13.

We follow a similar convention for temporary table space containers, using names like:

```
/DB_TMP/db2inst1/NODE0005/T1
```

There are two containers per database partition, T1 and T2.

See the “Creating the Database, Nodegroups, Table Spaces and Tables” on page 27 for more details about the commands used to create the DB2 objects that use these file systems.

We used the script in “Create Volume Groups and Logical Volumes” on page 223 to make all these file systems.

4.5.12 Enabling Disk Mirroring

Since the test configuration doesn't involve RAID disks, it is necessary to protect ourselves against disk failures. You can do this by using mirroring at the AIX logical volume level. When creating a logical volume, set the "number of copies" to 2. For an example, see Chapter 4.5.10, “Creation of Logical Volumes” on page 168. The following DB2 objects may be mirrored:

- Catalog Tables

- Log Files
- Data
- Index

To mirror a logical volume after creation, you can use the `mklvcopy` command. As an example, to mirror the lv_n05_01_103 logical volume, at node05:

```
mklvcopy lv_n05_01_103 2
```

You must ensure that the two copies are on different physical disks and if possible connected to different loops and power sources in order to avoid single points of failure.

Note that, in the test configuration, the serial link adapters and controllers are single points of failure. It would be possible to correct this by using multiple adapters and 7133 units and by mirroring the logical volumes across two adapters. In “Using AIX Error Notification” on page 191, we see that another way to eliminate this single point of failure is through AIX error notification.

Quorum Considerations

Quorum can be enabled or disabled. With quorum disabled, if a physical volume is not available, the volume group cannot be varied on unless `varyonvg -f` (force option) is used, leading to unpredictable results. On the other hand, with quorum enabled, at least three disks are necessary in each volume group to authorize varyon after one disk failed. This might not always be feasible (as in the tested configuration). Moreover, since the varyon command is under HACMP control, disk failures can go undetected. HACMP guidelines are usually to disable quorum for non-concurrent access volume groups.

4.5.13 Renaming the Shared Logical Volumes

Since we are starting from an existing RISC/6000 SP environment where logical volumes probably have default names, it is necessary to change these. Each shared logical volume needs to have a unique name in the cluster.

In our example, logical volume names on each node have names like lv_nii_0z_yyy, where ii is the SP node number, z is the volume group number in the node, and yyy is a sequential number.

In particular, it is necessary to rename each volume group’s log logical volume since they probably have the same default loglv00 name, and you will

get a name conflict when you import the volume group in another SP node. If you have any other default logical volume names in shared volumes groups, you should also rename them too.

In our configuration, we have already created logical volume names which conform to this rule in step Chapter 4.5.10, “Creation of Logical Volumes” on page 168. As an example, the following command will rename a logical volume to a new name:

```
chlv -n loglv_n01_01_01 loglv00
```

After changing the name of the log logical volume, make sure you change the corresponding entries in the /etc/filesystems file. In particular, when you rename a log logical volume, all the file system entries that use this log logical volume must be corrected manually.

4.5.14 Varying Off Shared Volume Groups on All Nodes

To allow the volume group to be imported at the corresponding takeover node, we need to take the volumes groups offline.

On node01, node05, node09, and node13, type:

```
varyoffvg vg_nxx_01
varyoffvg vg_nxx_02
```

This will deactivate the shared volume groups on all nodes.

4.5.15 Importing Shared Volume Groups

Since we have a pair of two-node clusters, we need to make node05's volume groups known on node01, and vice versa. Then we must repeat this step on the second cluster (node09 and node13).

When a node failure occurs, HACMP will `varyonvg` the volume group in the takeover node. This only works if the takeover node has already processed a `importvg` before the failure to update its volume group information about the failing node.

In our configuration, we have two internal disks and 64 external disks in each node. Assuming that vg_n13_01 includes hdisk21, as seen from node13, we would type:

```
importvg -y vg_n13_01 hdisk21
varyonvg vg_n13_01
```

This will import, then activate vg_n13_01 on node13. On node09, we have to specify:

```
importvg -y vg_n13_01 hdisk21
varyonvg vg_n13_01
```

Similar commands need to be run for the other volumes groups and must be repeated on node01 and node05.

Also, these commands assume that the disk logical names are the same on all the nodes. If you refer to the same physical disk by a different name on each node, you might run an `importvg` which affects another volume group (not the one you intended to `importvg`) and alter information about file systems, logical volumes, and so on.

You must do an `importvg` for each volume group on each node that is set up to do HACMP takeover. In our test, that means two volume groups on each node.

We used the script listed in “Synchronize Volume Groups” on page 238 to perform this task.

4.5.16 Changing Volume Groups on Destination Nodes

Shared volume groups should not be activated at system startup. To prevent a given volume group from being activated on the service and the takeover node at the same time, HACMP performs this operation.

To set activation at startup to off on node01:

```
chvg -a n -Q y vg_n01_01
chvg -a n -Q y vg_n01_02
```

Similarly, on node05:

```
chvg -a n -Q y vg_n05_01
chvg -a n -Q y vg_n05_02
```

These steps are repeated on node09 and node13 with the appropriate parameters.

4.5.17 Varying Off Volume Groups on Destination Nodes

To free the volume groups for normal use, we must vary offline these volume groups in the takeover nodes.

On node01:

```
varyoffvg vg_n05_01
varyoffvg vg_n05_02
```

Similarly, on node05:

```
varyoffvg vg_n01_01
varyoffvg vg_n01_02
```

These steps are repeated on node09 and node13 with the appropriate parameters.

4.5.18 Creating a DB2 Instance and Databases

The DB2 UDB EEE instance is created on node01 using the DB2 `db2icrt` command. An user different from the instance owner administrator and a new group will be required if you plan to use unfenced user defined functions; otherwise, the security of DB2 will be compromised.

Update /etc/services (on all SP nodes) with entries for the TCP/IP ports to be used by DB2 EEE. You must allocate as many ports as the maximum number of database partitions that would run per SP node after a takeover situation. This is twice as many ports compared to normal usage. In our example, we have four database partitions per SP node in normal usage. After an HACMP takeover, we will have eight database partitions running on the takeover SP node; so we need to configure eight ports.

Assuming that instance owner is db2inst1 and the user for unfenced UDF is db2inst1uf, to create a DB2 instance, as root:

```
cd /usr/lpp/db2_05_00/instance
./db2icrt -u db2inst1uf db2inst1
```

You must run this only in one node, and then set up the db2nodes.cfg file. In our example, db2nodes.cfg contains:

```
1 hnode01 0 sw01
2 hnode01 1 sw01
3 hnode01 2 sw01
4 hnode01 3 sw01
5 hnode05 0 sw05
6 hnode05 1 sw05
7 hnode05 2 sw05
8 hnode05 3 sw05
9 hnode09 0 sw09
10 hnode09 1 sw09
11 hnode09 2 sw09
12 hnode09 3 sw09
13 hnode13 0 sw13
14 hnode13 1 sw13
15 hnode13 2 sw13
16 hnode13 3 sw13
```

Normally, four ports are used, but in a takeover situation we will need eight, four for the database partitions that normally run in the node and four more for those that are taken over.

Databases can then be created with the DB2 `create database` command. For example, as db2inst1:

```
db2 terminate
export DB2NODE=1
db2 -v "create db tpcd30 on /DB_LOG"
```

Note that `DB2NODE` is set to `1` to ensure that the System Catalog tables are created at the first database partition. The `db2 terminate` makes sure that any existing connections are released.

For more details on the DB2 commands used to create the DB2 database and the objects inside it, see “Creating the Database, Nodegroups, Table Spaces and Tables” on page 27.

4.5.19 HACMP Installation

HACMP must be installed locally on each node. More specifically, the following components of the product were installed in our test machine:

Table 4. HACMP LPPs Installed

LPP	Packages	Description
cluster.adt.client	4.2.2.0 HACMP Client Include Files 4.2.2.0 HACMP Client Clstat Samples	To monitor cluster
cluster.base.client	4.2.2.0 HACMP Base Client Libraries 4.2.2.0 HACMP Base Client Runtime 4.2.2.0 HACMP Base Client Utilities	To allow access to HACMP manager
cluster.base.server	4.2.2.0 HACMP Base Server Diags 4.2.2.0 HACMP Base Server Events 4.2.2.0 HACMP Base Server Runtime 4.2.2.0 HACMP Base Server Utilities	HACMP manager
cluster.cspoc	4.2.2.0 HACMP CSPOC commands 4.2.2.0 HACMP CSPOC dsh and perl 4.2.2.0 HACMP CSPOC Runtime commands	Smit tools to manage shared volume groups
cluster.man.en_US	4.2.2.0 HACMP Client Man Pages-U.S.English 4.2.2.0 HACMP CSPOC Man Pages-U.S.English 4.2.2.0 HACMP Server Man Pages-U.S.Englis	Manuals
cluster.msg.en_US	4.2.2.0 HACMP Client Messages-U.S.English 4.2.2.0 HACMP CSPOC Messages-U.S.English 4.2.2.0 HACMP Server Messages-U.S.English	Messages for HACMP

Using `smit install_latest`, enter the name of the directory or device where the software resides, then select the products to be installed. You need

around 10 MB of free space in the /usr file system, depending on the options chosen. After the installation has completed, you should verify it using the /usr/sbin/cluster/diag/clverify utility. Select the software option, then follow the instructions. If all the nodes in the cluster have been installed from the same image, the verification step needs to be performed only once.

4.6 HACMP Configuration of cluster_09_13

In this section, we follow an incremental approach to configuring the HACMP clusters, starting from a relatively simple two-node mutual takeover configuration to progressively include more resources in the failover scenario. We have two HACMP clusters: one between node01 and node05 and one between node09 and node13. The first cluster is the most complex since it will be necessary to protect against NFS failure and also switch Eprimary node failure. For this reason, we will start with the second cluster.

Each time it was necessary to change one of the HACMP scripts, we tried to do it through the use of pre- and post-event scripts. This makes software maintenance easier since these scripts are not affected by new releases of HACMP or PTFs.

As a reminder, node09 and node13 share the same 7133 disk units. A serial cable between the nodes provides a non-IP heartbeat network. The configuration steps described below can be found with more detail in the *HACMP 4.1 for AIX Installation Guide,* SC23-2769.

4.6.1 Defining the Cluster ID and Name

On node09, enter `smit cm_config_cluster.add`. In our example, we entered `10` in the Cluster ID field and `cluster09_13` for the Cluster Name.

The cluster ID number must be unique in the connected networks in order to allow distinguish each cluster in the SP2, and from others outside if there more connected in the network.

4.6.2 Defining Nodes

On node09, enter `smit cm_config_nodes.add`. Set Node Names to `node09`, `node13`.

4.6.3 Defining Adapters

For this initial cluster, we have four adapters per node: one switch adapter, and one tty and SP Ethernet adapter for heartbeat communications and an

external Ethernet adapter. The reason why we define two networks for the heartbeat is explained in 4.4.1, “SP Ethernet Considerations” on page 151.

On node09, type `smit cm_config_adapters.add`.

For node09's tty adapter, the fields for our example were:

```
                                Add an Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Adapter IP Label                                   [node09tty1]
* Network Type                                       [rs232]                +
* Network Name                                       [serial1]              +
* Network Attribute                                   serial                +
* Adapter Function                                    service               +
  Adapter Identifier                                 [/dev/tty1]
  Adapter Hardware Address                           []
  Node Name                                          [node09]               +

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Repeat this command for the definition of Adapter Label node13tty1 for node13.

For node09's Ethernet adapter, en0:

```
                                Add an Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Adapter IP Label                                   [hnode09]
* Network Type                                       [ether]                +
* Network Name                                       [ether1]               +
* Network Attribute                                   private               +
* Adapter Function                                    service               +
  Adapter Identifier                                 []
  Adapter Hardware Address                           []
  Node Name                                          [node09]               +

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Additionally, en0 is a private network because it is for cluster communications. Repeat this command for definition of hnode13 Adapter Label for node13.

For network name, we chose ether1 for en0 and ether2 for en1 and en2 because en1 and en2 are connected to the same physical network.

On nodes node01 and node05 only, we have a third Ethernet adapter; so this next step is only for the cluster01_05 cluster:

```
                                Add an Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Adapter IP Label                                   [etsby01]
* Network Type                                       [ether]                 +
* Network Name                                       [ether2]                +
* Network Attribute                                   public                 +
* Adapter Function                                    standby +
  Adapter Identifier                                 []
  Adapter Hardware Address                           []
  Node Name                                          [node01]                +

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Note that this network name is defined as `ether2`.

This step must be repeated for all the other adapters. In our example, for the cluster01_05 cluster, these are:

- etsvc01 as service adapter in node01
- etsby01 as standby adapter in node01
- etboot01 as boot adapter in node01
- etsvc05 as service adapter in node05
- etsby05 as standby adapter in node05
- etboot05 as boot adapter in node05

For node09's switch adapter:

```
                                Add an Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Adapter IP Label                                    [sw09]
* Network Type                                        [hps]                +
* Network Name                                        [HPS1]               +
* Network Attribute                                    private             +
* Adapter Function                                     service             +
  Adapter Identifier                                  []
  Adapter Hardware Address                            []
  Node Name                                           [node09]             +

F1=Help           F2=Refresh        F3=Cancel         F4=List
F5=Reset          F6=Command        F7=Edit           F8=Image
F9=Shell          F10=Exit          Enter=Do
```

Note that we don't use the SDR switch address (switch3). Instead, HACMP knows only the swxx address, which is defined on the same subnetwork as the SDR switch addresses because there is no standby address for the switch.

Note that the network name for the switch should contain the string HPS, which will be required below when we enable IP address takeover. The network type should always be private for the HPS.

This step must be repeated for the service switch address of node13.

Since we plan to use switch takeover, due to the NFS mounts across the switch, do the same for the boot switch address of each node, using boot in the adapter function field.

In the previous SMIT screens, the Adapter Identifier field has been left blank. This is possible because HACMP looks up the address in the /etc/hosts file.

The remaining adapters can then be easily defined to HACMP in a way similar to that shown above.

SDR Switch Addresses

Don't use the SDR switch address as the boot address for HACMP. This might result in the nodes hanging with an error message.

4.6.4 Synchronizing Cluster Definition on All Nodes

This step will copy the ODM definitions entered on node09 to node13. Execute `smit cm_cfg_top_menu`, then select **Synchronize Cluster Topology**, and press **Enter**.

If any errors are reported, synchronization is stopped.

If you have problems related to TCP/IP permissions, make sure that you have applied the latest HACMP PTFs.

4.6.5 Configuring Resource Groups

We have one resource group on each node. These are called resource09 and resource13 in cluster09_13. The resource groups contain the shared disks and volume groups, file systems, network interfaces, and application servers for each node. Since we are not using standby nodes, required for rotating resource groups, nor the concurrent logical volume manager for concurrent access resource groups, the groups must be cascading resource groups.

To define the resource09 resource group, type `smit cm_add_grp`, then enter the following information:

```
                              Add a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* Resource Group Name                              [resource09]
* Node Relationship                                 cascading              +
* Participating Node Names                         [node09 node13]        +

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

This step must be repeated for the resource13 resource group:

```
                                Add a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Resource Group Name                                [resource13]
* Node Relationship                                   cascading              +
* Participating Node Names                           [node13 node09]        +

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Note that the first node in the node list specifies the high priority node, sometimes called the owner, of the resources.

4.6.6 Configuring Application Servers

Since we want DB2 UDB EEE to be restarted when one of the nodes in the cluster fails, we have to create start and stop scripts for DB2 UDB EEE. For availability reasons, we put these scripts in the instance owner's home directory. Normally, each node has its own set of start/stop scripts. Because of the large number of possibilities with 16 database partitions we wrote start and stop scripts controlled by a configuration file (see "Start and Stop DB2 UDB EEE" on page 211).

They proved to work reliably in our configuration, but would probably have to be modified in order to be integrated into other environments.

The configuration file used by the scripts allows you to choose the correct network interfaces and ports for normal and takeover operation for each resource and database partition. To handle post-takeover, remember that the /etc/services files on each node must have twice as many ports available for DB2 UDB EEE as are used in normal operation.

The scripts use the resource name and hostname to locate in the configuration file the database partitions involved and will also reconfigure DB2 EEE.

The core of the script is the `db2start restart` command. The syntax of this command is:

```
db2start nodenum W restart hostname X netname Y port Z
```

This command causes database partition W (corresponding to the first column of the db2nodes.cfg file) to be restarted on hostname X, using Y as network interface, on logical port Z.

Note that Port Z is the logical port number used by DB2 EEE, not the name that you include in /etc/services. Port 0 will be the first port number allocated in /etc/services to be used by DB2 EEE, port 1 the second number and so on.

Typically, if you have four database partitions per SP node, you use ports 0 to 3, and during takeover, you need to map the ports of the failing partitions to ports 4 to 7, because 0 to 3 are already in use by the partitions normally owned by the takeover node.

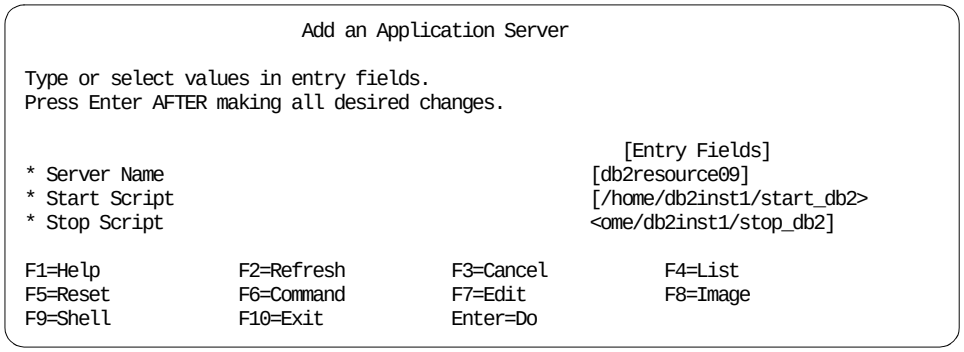
The `db2start` command with the restart option will update the db2nodes.cfg file and override the old values.

Before issuing `db2stop`, the stop script stops all DB2 UDB EEE applications using the DB2 `force application` command. In a more tailored version, the script could force only those applications that need access to the database partitions we need to stop.

Since most applications need access to all database partitions, we felt that it was not worth checking which database partitions the applications were using.

This script highlights the benefits one would derive from using rotating resources instead of cascading ones. With cascading resources, all applications are forced out twice, the first time when a node fails, the second time when it reintegrates the cluster. This may be unacceptable for some critical environments.

To define the start and stop scripts to HACMP, we define two application servers, called db2resource09 and db2resource13, on node09 and node13. As an example, for db2resource09, enter `smit claddserv.dialog`:



where Start Script contains:

```
/home/db2inst1/start_db2.ksh \
    db2inst1 \
    resource09 \
    /home/db2inst1/db2nodes.cfg.hacmp
```

and Stop Script contains:

```
/home/db2inst1/stop_db2.ksh \
    db2inst1 \
    resource09 \
    /home/db2inst1/db2nodes.cfg.hacmp
```

The application server information is propagated automatically to node13.

We use the complete path name to avoid problems with the path used internally by HACMP.

This step must be repeated for node01 and node05.

db2start problems

Make sure that the statd and lockd daemons are running on each node before starting DB2 UDB EEE . Otherwise, SQL error 5005 will be returned. These daemons are part of the nfs group that can be started with the `startsrc -g` NFS command.

4.6.7 Configuring Resources for Resource Groups

Resources need now to be configured for the resource09 and resource13 resource groups. For example, for resource09, type `smit cm_cfg_res.select`, select the group; then enter the following data:

```
                Configure Resources for a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                 [Entry Fields]
  Resource Group Name                                 resource09
  Node Relationship                                   cascading
  Participating Node Names                            node09 node13

  Service IP label                                   [sw09] +
  HTY Service Label                                  []
  Filesystems                                        [/DB_LOG/db2inst1/NODE000] +
  Filesystems Consistency Check                       fsck                    +
  Filesystems Recovery Method                         sequential              +
  Filesystems to Export                              []                       +
  Filesystems to NFS mount                           []                       +
  Volume Groups                                      [vg_n09_01 vg_n09_02] +
  Concurrent Volume groups                           []                       +
  Raw Disk PVIDs                                     []                       +
  Application Servers                                [db2resource09] +
  Miscellaneous Data                                 []

  Inactive Takeover Activated                         false                   +
  9333 Disk Fencing Activated                         false                   +
  SSA Disk Fencing Activated                          false                   +
  Filesystems mounted before IP configured            false                   +
[BOTTOM]

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

For each resource definition you must specify the service IP label, which is sw01, sw05, sw09, or sw13 depending on the node.

For node01 and node05, we also have service adapters etsvc01 and etsrc05. Specify sw01 etsvc01 in resource01 definition for Service IP Label and sw05 etsvc05 in resource05. Note that pressing F4 does not allow you to choose this value. You must type this value.

For file system definition, “Importing Shared Volume Groups” on page 176 must be done before this step; otherwise F4 will not generate a list of options.

Press **F4** in the entry field for Filesystems, and you obtain the list of file systems defined in the shared volume groups. Choose the file systems needed for the corresponding resource under normal operating conditions; HACMP will take care of takeover situations.

In our example, we selected the file systems used by the database log files and temporary table spaces on each node. On node01 only, we also selected /home/db2inst1 and /adsmserv.

Note that the Volume Groups field must be specified to allow the raw devices used for the DMS tablespaces to be accessed after takeover. The Raw Disk PVIDs field can be left blank.

In a two-cluster configuration like our example, HACMP is unable to manage the NFS-exported file systems as required. Leave Filesystems to Export blank. This function is covered in “Configuring NFS Access to /home/db2inst1” on page 199.

If you have only one cluster, you could specify the directories to be exported using NFS.

You must include, in the resource01 on node01, the directories /home/db2inst1 and /adsmserv in Filesystems to NFS mount. This allows HACMP to automatically mount these file systems through NFS in the takeover of node05 during normal operation and replace this NFS mount by a direct mount during node01 takeover by node05.

Finally, in the field Application Servers, use **F4** to generate a list, and select all the application servers related to this resource.

This step must also be completed for resource13, changing the resources to those appropriate for that resource group.

4.6.8 Synchronizing Node Environment

Propagate the above information to node13 by executing `smit cm_cfg_res_menu` and selecting the **Synchronize Cluster Resources** option.

4.6.9 Verifying Cluster Configuration

To verify the cluster configuration, run the cluster option of the /usr/sbin/cluster/diag/clverify utility on node09.

4.6.10 Configuring Client Nodes

The clinfo (cluster.client) program is required in order to update the Address Resolution Protocol (ARP) caches after IP address takeover. It is also needed to use the clstat cluster status monitoring program. Clinfo uses a configuration file located in /usr/sbin/cluster/etc/clhosts. Edit this file, comment out the line starting with 127.0.0.1, and add the following lines:

```
sw01
sw05
sw09
sw13
etsvc01
etsvc05
```

If we had clients, we would add their names to the `PING_CLIENT_LIST` variable in the /usr/sbin/cluster/etc/clinfo.rc file. You may want to include any service addresses in your system, such as Ethernet addresses.

In order for HACMP to work properly, add /usr/sbin/cluster/utilities and /usr/sbin/cluster/events/utils to the `PATH` variable in the /.kshrc file on the Control Workstation. This assumes that the /.kshrc file is propagated to the SP nodes through the file collection mechanism, as is usually the case if file collections are enabled. If they are not, modify this procedure to fit your environment.

4.6.11 Starting Cluster Services

On node09 and node13, type `smit clstart.dialog`:

```
                        Start Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                  [Entry Fields]
* Start now, on system restart or both            now                    +

  BROADCAST message at startup?                   true                   +
  Startup Cluster Lock Services?                  false                  +
  Startup Cluster Information Daemon?             true                   +


F1=Help          F2=Refresh       F3=Cancel        F4=List
F5=Reset         F6=Command       F7=Edit          F8=Image
F9=Shell         F10=Exit         Enter=Do
```

Start only one node at a time, and let each node complete its startup before starting the next one. The startup is not complete when SMIT displays OK. It is also necessary to monitor the HACMP log files (/var/adm/cluster.log or /tmp/hacmp.out) to check for completion of the startup.

DB2 Instance Owner's Password

To avoid being prompted by the operating system to change the password of the DB2 instance owner at logon time (which would interfere with the execution of the DB2 UDB EEE start and stop scripts), delete the line starting with `flags=` in the /etc/security/passwd file on the Control Workstation (assuming you are using file collections) for the DB2 instance owner.

4.6.12 Activating I/O Pacing

To avoid HACMP having to compete with I/O-bound applications for the CPU, it is a good idea to activate I/O pacing. With `smit chgsys`, set the HIGH water field to 33, and the LOW water field to 24. This will guarantee a correct failover behavior for HACMP in most environments.

4.6.13 Using AIX Error Notification

HACMP provides a way to associate user-defined scripts (so-called Notify methods) with errors logged by the AIX error notification services. The steps shown below should be repeated on all cluster nodes. In each case, the information to be entered can be accessed by typing `smit cm_add_notifymeth.dialog`. More error messages can be found in the *IBM RISC System/6000 Scalable POWERparallel Systems Diagnosis and Messages Guide,* GC23-3899, and in the *AIX Version 4.1 Problem Solving Guide and Reference,* SC23-2606, to suit other environments' needs.

4.6.13.1 TTY Failure

For tty failure on the tty serial line, we sent a mail message to the system administrator to notify him or her of the problem. Since failure of the tty1 is not critical as long as there is a TCP/IP connection available, it is acceptable to wait for some off-shift period until the adapter is serviced.

```
                              Add a Notify Method

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Notification Object Name                           [tty1]
* Persist across system restart?                      No                    +
  Process ID for use by Notify Method                []                    +#
  Select Error Class                                  Hardware              +
  Select Error Type                                   PERM                  +
  Match Alertable errors?                             None                  +
  Select Error Label                                 []           +
  Resource Name                                      [tty1]
  Resource Class                                     [All]
  Resource Type                                      [All]
  Notify Method
[echo "TTY1 line Problem. Check errorlog \" | mail root ]

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

4.6.13.2 Switch Failure

As far as switch failures goes, there are two error messages to watch out for: HPS_ER9, for switch adapter failures, and HPS_ER6, for switch adapter, micro-channel bus slot, or external clock source failures leading to termination of the Worm process. In both cases, it is necessary to determine whether the failure happened on both cluster nodes, or only locally.

In the first case, for switch adapter failure (HPS-ER9), we should perform a network failover or send a message to the root user if there is no backup network available (as in our sample configuration).

In the second case, for failure of the Worm process (HPS_ER6), graceful shutdown with takeover is performed, which allows DB2 UDB EEE to restart the failing database partitions on the takeover SP node.

For example, for the HPS_ER6 error:

```
                              Add a Notify Method

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Notification Object Name                           [HPS_ER6]
* Persist across system restart?                      Yes +
  Process ID for use by Notify Method                []                        +#
  Select Error Class                                  All +
  Select Error Type                                   All +
  Match Alertable errors?                             None                      +
  Select Error Label                                 [HPS_FAULT6_ER]            +
  Resource Name                                      [All]
  Resource Class                                     [All]
  Resource Type                                      [All]
  Notify Method                    [/usr/sbin/cluster/utilities/clstop -yNgr]

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

4.6.13.3 Disk and Adapter Failure

For 7133 adapter failures, we will do a graceful shutdown with takeover as for the switch failures:

```
                              Add a Notify Method

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Notification Object Name                           [7133_adapt_0]
* Persist across system restart?                      Yes +
  Process ID for use by Notify Method                []                          +#
  Select Error Class                                  Hardware                   +
  Select Error Type                                   PERM                       +
  Match Alertable errors?                             None                       +
  Select Error Label                                 [SDA_ERR1]                +
  Resource Name                                      [serdasda0]
  Resource Class                                     [adapter]
  Resource Type                                      [serdasda]
  Notify Method                  [/usr/sbin/cluster/utilities/clstop -yNgr]

F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Similarly, for 7133 controller failures:

```
                          Add a Notify Method

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* Notification Object Name                          [7133_contr_0]
* Persist across system restart?                     Yes +
  Process ID for use by Notify Method               []                      +#
  Select Error Class                                 Hardware               +
  Select Error Type                                  PERM                   +
  Match Alertable errors?                            None                   +
  Select Error Label                                [SDC_ERR1]           +
  Resource Name                                     [serdasdc0]
  Resource Class                                    [adapter]
  Resource Type                                     [serdasdc]
  Notify Method                 [/usr/sbin/cluster/utilities/clstop -yNgr]

F1=Help           F2=Refresh        F3=Cancel         F4=List
F5=Reset          F6=Command        F7=Edit           F8=Image
F9=Shell          F10=Exit          Enter=Do
```

4.6.13.4 Memory Failures

For memory failures:

```
                          Add a Notify Method

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* Notification Object Name                          [MEM_1]
* Persist across system restart?                     Yes +
  Process ID for use by Notify Method               []                      +#
  Select Error Class                                 Hardware               +
  Select Error Type                                  PERM                   +
  Match Alertable errors?                            None                   +
  Select Error Label                                [MEM1]           +
  Resource Name                                     [All]
  Resource Class                                    [memory]
  Resource Type                                     [All]
  Notify Method                 [/usr/sbin/cluster/utilities/clstop -yNgr]

F1=Help           F2=Refresh        F3=Cancel         F4=List
F5=Reset          F6=Command        F7=Edit           F8=Image
F9=Shell          F10=Exit          Enter=Do
```

The same procedure should be used for the MEM2 and MEM3 error labels. MEM1 indicates the absence of a memory card out of a memory card pair; MEM2 indicates the failure of up to two SIMMs on a memory card, and MEM3 indicates the failure of a memory card out of a memory card pair.

4.7 Review of Configuration in Non-Catalog Cluster

What have we achieved so far? We have a running two-node HACMP cluster with cascading resources in a mutual takeover configuration. From a DB2 UDB EEE perspective, we are protected against:

- Disk failure through mirroring
- TCP/IP failure through the heartbeat daemon
- CPU failure through shared disks

We are not protected against:

- Switch adapter failure
- Switch primary node failure
- Switch power failure or global failure
- Node or frame power failure
- Node main memory failure
- Serial line failure
- Disk adapter or controller failure

These points are addressed below. For the time being, in normal operating mode, each node (node09 and node13) has its own DB2 UDB EEE database partitions. In case of a node's failure, the surviving node will take over the other node's DB2 UDB EEE partitions, releasing them when the failing node reintegrates into the cluster. Time for takeover, assuming no DB2 UDB EEE rollback and rollforward is required, is less than one minute.

During node reintegration, the switch is reinitialized automatically by HACMP.

4.7.1 Switch Primary Node Failure

In order to activate Eprimary node takeover, execute the `/usr/sbin/cluster/events/utils/cl_HPS_Eprimary manage` command on node09. This feature ensures switch availability even in the case of the Eprimary node being unavailable. Should node09 be defined as Eprimary (by executing the Eprimary node09 command on the Control Workstation) and fail, HACMP will move the Eprimary node to node13. The Eprimary resource is defined as a rotating resource, so that node13 will keep the Eprimary function until it becomes unavailable and node09 becomes Eprimary again.

As we will see below, it makes more sense to define the Eprimary function in the other cluster (node01 and node05). If the switch is not running, HACMP

cannot start properly. If NFS is down, it cannot start either. In our setup, NFS files are mounted over the switch. Since node01 is the NFS server, it should also be the Eprimary node. The `unmanage` option of the `cl_HPS_Eprimary` command can be used to move the Eprimary takeover function to another cluster. Only one cluster can have this feature enabled.

4.7.2 Switch IP Address Takeover

Because of the way `db2start` using the `restart` option works, DB2 UDB EEE does not really need switch IP address takeover. The failing DB2 UDB EEE database partition is restarted through another network interface on the surviving node. For clients and applications, however, it can certainly be useful to implement IP address takeover. The following points need to be considered:

- ARP must be enabled for the HPS network. You can check this by executing `smit list_node_switch` on the Control Workstation. The RISC/6000 SP documentation recommends enabling ARP in all situations. If ARP is not enabled, change the boot response field to customize, enable ARP, and press **Enter**. Then, network boot the SP nodes by selecting the global commands submenu from the `spmon -g` interface.
- HACMP HPS network names must contain the HPS string.
- HPS networks must be private networks.
- Standby addresses are not required. Since IP takeover cannot use the SDR switch addresses, we use the `swnn` and `swbootnn` addresses as service and boot addresses, where `nn` is 01, 05, 09, or 13. These addresses are on the same subnet as the SDR switch addresses.

4.7.3 Node and Frame Power Recovery

RISC/6000 SP frames contain between one and three AC/DC 48 volt power supplies, depending on the type of frame. If the N+1 feature is installed, this implies that there are at least two power supplies per frame. A failure with one of these power supplies will not interrupt the operation of the RISC/6000 SP because of the backup that will be provided by the other power supply. Furthermore, the defective power supply is hot-pluggable and can be replaced without interruption to the system.

Each power supply can service up to eight nodes. With this in mind, if you have more than eight nodes per frame, it becomes necessary to consider configuring three power supplies to protect your system against unforeseen power supply failure.

However, there is only one power cord from the external power network to the SP. Customers should implement uninterrupted power sources (UPS) to guard themselves against a global power loss. The same is true concerning the power source for the external disks.

4.8 HACMP Configuration of cluster_01_05

Now we have completed the HACMP configuration of the cluster which includes node09 and node13. This section covers the additional steps necessary on the cluster that includes node01 and node05. These extra steps are needed as:

- node01 holds the instance owner's home directory which is made available to the other nodes through NFS.
- node01 is the switch Eprimary node.

The steps listed above for the cluster comprised of node09 and node13 should be repeated for node01 and node05 with the following differences:

- "Creating the /home/db2inst1 File System" on page 171 is done only on node01.
- Assuming the resource groups are now called resource01 and resource05, for resource01, we have to specify (see "Configuring Resources for Resource Groups" on page 187):

```
Filesystems to Export = /home/db2inst1 /adsmserv
Filesystems to NFS mount = /home/db2inst1 /adsmserv
```

- The vg_n01_01 volume group must have the same major number on node01 and node05 because it contains NFS-mounted file systems. This number can be specified when importing and creating the volume group. Available major numbers are given by the `lvlstmajor` AIX command. To change an existing volume group's major number, it is possible to export it then import it and specify the major number in the `importvg` command. (See "Synchronize Volume Groups" on page 238).
- The switch Eprimary node should be node01 since node01 is also the NFS server. Assuming that the cluster, cluster09_13, is still managing Eprimary takeover, to deactivate this function on node09:

```
/usr/sbin/cluster/events/utils/cl_HPS_Eprimary unmanage
```

  Then, to activate this function on node01:

```
/usr/sbin/cluster/events/utils/cl_HPS_Eprimary manage
```

  Then, at the Control Workstation:

```
Eprimary hnode01
```

4.8.1 NFS-Mounting /home/db2inst1 in cluster_09_13

There are two scenarios to consider:

- Mount Time - when a node comes up
- Unmount Time - when a node goes down

4.8.1.1 Mount Time

To NFS-mount /home/db2inst1 on node05 over the switch from node01, we use HACMP (see “Configuring Resources for Resource Groups” on page 187).

However, in cluster_09_13, we need to mount /home/db2inst1 after HACMP has been started in order to have the switch available. In both nodes in cluster09_13, we add a post-event script to the node_up_local HACMP script to mount the file system after the switch starts.

By typing `smit clcsclev.select`, the list of HACMP event scripts is displayed. Select the event you want to add a post- or pre-event script to (node_up_local in this case). The following menu appears:

```
                          Change/Show Cluster Events

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]

  Event Name                                        node_up_local

  Description         Script run when it is the local node joining the cluster

* Event Command                        [/usr/sbin/cluster/events/node_up_local]

  Notify Command                                    []
  Pre-event Command                                 []
  Post-event Command                    [/admin/hacmp/post_node_up_local]
  Recovery Command                                  []
* Recovery Counter                                  [0]                        #

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

In the Post-event Command field, add the name of the script you want to create. For maintenance reasons, it is not be a good idea to locate this script in the HACMP install directory since its contents could be overwritten by applying a new release of the product. Instead, you might want to choose a separate directory to store the pre- and post-event scripts, such as

/admin/hacmp, by creating the necessary entry with the `mkdir` command. The naming convention for the scripts should follow a simple scheme such as pre_ resp. post_<event name>.

The contents of the /admin/hacmp/post_node_up_local are:

```
#!/bin/sh
STATUS=0
/usr/sbin/cluster/events/utils/cl_activate_nfs 1 sw01 /home/db2inst1
if [ $? -ne 0 ]
then
        echo Failed to mount /home/db2inst1 from sw01
        echo Manual intervention required
        STATUS=1
fi
exit $STATUS
```

The script should be made executable. Assuming that it belongs to the root user:

```
chmod +x /usr/hacmp/post_node_up_local
```

4.8.1.2 Unmount Time

We need to unmount /home/db2inst1 when a node comes down; so we added a post_node_down_local script as a post-event to node_local_down. This script will unmount the instance home directory:

```
#!/bin/sh
STATUS=0
/usr/sbin/cluster/events/utils/cl_deactivate_nfs /home/db2inst1
if [ $? -ne 0 ]
then
        echo Failed to umount /home/db2inst1 from sw01
        echo Manual intervention required
        STATUS=1
fi
STATUS=0
exit $STATUS
```

4.8.2 Configuring NFS Access to /home/db2inst1

In our example configuration, making the DB2 instance owner's home directory available to the other nodes through NFS required some changes to the HACMP scripts.

The problems relate to how /etc/xtab file is updated. Since we declared /home/db2inst1 to be mounted by NFS in resource01 in cluster01_05, HACMP will overwrite the export permissions in the /etc/xtab file with permissions to allow automatic access to cluster09_13's nodes.

This works fine if you have only one cluster, but if you have more than one cluster using the same file system, as in our example, this creates problems. For example, we found that the nodes in cluster09_13 were not allowed to mount the file system.

As stated in the *HACMP Admin Guide,* SC23-1941, Chapter 11, you must modify the cl_export_fs script and remove the `-i` flag in the two places where `exportfs` command is used. This will force HACMP to use /etc/exports file when exporting using NFS.

However, as a result of this change, you must perform the NFS exports manually in each node using the /etc/exports file.

To perform an NFS export manually, use `smitty mknfsexp`:

```
                        Add a Directory to Exports List

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* PATHNAME of directory to export                    [/home/db2inst1]
* MODE to export directory                            read-write              +
  HOSTS & NETGROUPS allowed client access            [sw05,sw09,sw13,switch01>
  Anonymous UID                                      [-2]
  HOSTS allowed root access                          []
  HOSTNAME list. If exported read-mostly             []
  Use SECURE option?                                  no                     +
* EXPORT directory now, system restart or both        both                   +
  PATHNAME of alternate Exports file                 []

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

You must do this in node01 and node05 and allow access to any host where you plan to NFS mount /home/db2inst1 and /adsmserv.

In this example, due to the use of aliases, we must specify the switchxx network interface name in addition to the swxx name.

4.8.3 Modifying the cl_deactivate_nfs Script

The node_down_remote script, executed by the surviving node after failure of a node, calls the cl_deactivate_nfs script. By default, cl_deactivate_nfs will try to umount the /home/db2inst1 file system by first killing all processes having open file descriptors in this file system, then issuing a `umount -f` against this file system.

For DB2 UDB EEE, HACMP needs to stop node05's database partitions when node01 goes down in order to free up the /home/db2inst1 file system mount point. To do this on node05, comment out the lines in the cl_deactivate_nfs script (located in /usr/sbin/cluster/events/utils) containing the `cl_nfskill` command without the `-t` flag:

```
# cl_nfskill -k -u $fs
```

and uncomment the lines with the `-t` flag:

```
cl_nfskill -k -u -t $fs
```

Consider also amending this script so that the `umount` command is issued once, instead of repeatedly in a loop. If the file system is not busy, it will be unmounted. If it is busy, the umount is unlikely to be any more successful if it is executed several times. Please read the comments in the HACMP script because this change could kill another applications not related to the resources to be taken over.

To restart node05's database partitions, you must add a post-event script to the node_down_remote_complete script. Here is an example:

```
#!/bin/sh
STATUS=0
# added for DB2 UDB EEE
i=0
while (( i < 240 ))
do
    if ! su - db2inst1 -exec date 2>&1 |grep "Unable to change"
    then
        break
    fi
sleep 3
(( i = i + 1 ))
done
if (( i == 240 ))
then
    echo Problem with NFS...Manual intervention required
    STATUS=1
fi
/home/db2inst1/start_db2.ksh \
                    db2inst1 \
                    resource05 \
                    /home/db2inst1/db2nodes.cfg.hacmp
if [ $? -ne 0 ]
then
    STATUS=1
fi
exit $STATUS
```

On node09 and node13, this is not required because the NFS mounts for /home/db2inst1 are outside HACMP control.

4.8.4 Adding a Post-Event Script to the node_up_remote_complete Script

When node01 reintegrates into the cluster, the cl_deactivate_fs script is executed on node05. This script will call the AIX `fuser` command, which will kill those processes having open file descriptors in the file systems to be unmounted. For DB2 UDB EEE, this means the DB2 processes used by node05's database partitions are killed. Since DB2 applications are forced out during cluster reintegration, stopping node05 's database partitions will not make any difference for most applications. To restart DB2 UDB EEE on node05 after node01's reintegration, create the post_node_up_remote_complete script as a post-event script to the node_up_remote_complete event as follows:

```
#!/bin/sh
STATUS=0
# added for DB2 UDB EEE


i=0
while (( i < 240 ))
do
    if ! su - db2inst1 -exec date 2>&1 |grep "Unable to change"
    then
        break
    fi
sleep 3
(( i = i + 1 ))
done
if (( i == 240 ))
then
    echo Problem with NFS...Manual intervention required
    STATUS=1
fi

/home/db2inst1/start_db2.ksh \
                    db2inst1 \
                    resource05 \
                    /home/db2inst1/db2nodes.cfg.hacmp

if [ $? -ne 0 ]
then
    STATUS=1
fi
exit $STATUS
```

The lines before the call to start_db2.ksh prevent DB2 UDB EEE starting before NFS has stabilized. This assumes that we know the name of the instance.

We use the command `su - db2inst -exec date` to test if the instance home directory, /home/db2inst1, is already available because it is NFS-mounted from node01.

If /home/db2inst1 is not available after 240 seconds, a failure is reported. The wait time is to allow for node01 to become available and to be taken over by node05, and to have /home/db2inst1 ready to use.

4.8.5 Adding a Post-Event Script to the stop_server Script

As explained in the previous section, by default, HACMP will kill node05's database partitions when node01 reintegrates into the cluster. To avoid the risk of damaging a database by a brute force kill of DB2, create the post_stop_server as a post-event script to the stop_server event.

```
#!/bin/sh
STATUS=0
/home/db2inst1/stop_db2.ksh \
                    db2inst1 \
                    resource05 \
                   /home/db2inst1/db2nodes.cfg.hacmp

if [ $? -ne 0 ]
then
    STATUS=1
fi
exit $STATUS
```

4.8.6 Adding a Pre-Event Script to start_server

We create the pre_start_server script as a pre-event script to the script start_db2.ksh event on node05 in order to make sure that `start_db2.ksh` is not issued before NFS has stabilized.

```
#!/bin/sh
STATUS=0
i=0
while (( i < 30 ))
do
    if ! su - db2inst1 -exec date 2>&1 |grep "Unable to change"
    then
        break
    fi
    sleep 3
    (( i = i + 1 ))
done
```

```
if (( i == 30 ))
then
    echo Problem with NFS...Manual intervention required
    STATUS=1
fi
exit $STATUS
```

4.8.7 Adding a Post-Event Script to node_down_remote_complete

During failover on node05, sometimes we observed that restarting database partition 01 on node05 would cause the node05's four database partitions to terminate. To prevent this, we created the post_node_down_remote_complete script as a post-event script to the node_down_remote_complete event as follows:

```
#!/bin/sh
/home/db2inst1/start_db2.ksh \
                    db2inst1 \
                    resource05 \
                    /home/db2inst1/db2nodes.cfg.hacmp
```

This has no effect if node05 's database partitions are already running, as they should be.

4.9 How HACMP Takeover Affects DB2 UDB EEE

During HACMP takeover, DB2 transactions are terminated with SQL error message SQL1229 when the database partitions are restarted (using db2start restart). During restart, DB2 performs a roll-back of all uncompleted transactions, then a roll-forward to the last commit.

Shown below is the full text of the SQL1229 error message:

```
SQL1229N The current transaction has been rolled back because of a
system error.
Explanation:  A system error, such as node failure or connection
failure, has occurred.  The application is rolled back to the previous
COMMIT.
```

This is the behavior of various DB2 utilities when they are rolled back:

- Import of a table

  The application is rolled back. If the `COMMITCOUNT` parameter was used, the operation is rolled back to a previous committed point.

- Table Reorganization

  The operation is aborted and must be resubmitted.

- Nodegroup Redistribute

The operation is aborted. Some of the tables in the nodegroup may have been successful redistributed. Issuing the request again will restart the operation from the point of failure.

- Roll forward of a database

  The operation is aborted and the database is still in Roll Forward Pending state. The command must be resubmitted.

- Backup/Restore of a database

  The operation is aborted and must be resubmitted.

4.9.1 Takeover of the DB2 Instance Owner's Home Directory

If a node comes up while node01 is down, /home/db2inst1 will be automatically mounted as soon as node01 becomes available.

In this example, node05 will take over node01's file systems in the event of node01's failure. From a DB2 UDB EEE perspective, if node01 goes down, no DB2 command can be issued until node05 takes over node01.

Normally, transactions that are already running are not affected since the executables used by DB2 are already in local memory. Commands can be issued again when the server comes back up or takeover has occurred. Connection to a DB2 database is not lost.

4.9.2 DB2 UDB EEE Database Partition Failure

This section describes the behavior of the test system in the event of the failure of a DB2 UDB EEE partition, depending on the partition type. This should be compared to the behavior of DB2 UDB EEE without HACMP.

- Coordinator Partition failure

  All applications need to reconnect to the databases. The behavior is the same as without HACMP.

- Database Partition failure

  After takeover (within one minute of failure), DB2 UDB EEE needs to recover. The SQL1229 (transaction rolled back) error message is issued when the database partition is restarted, and the transactions are rolled back. Because the ha_stopscript stop script forces all applications out, any connections to databases are lost. Until the DB2 UDB EEE partition is restarted, transactions hang, as in the case where HACMP is not used.

- Failure of Data Partition not used in transaction

When the failure occurs, transactions proceed undisturbed. During cluster reintegration, however, the ha_stopscript will force all applications out, leading to transactions being rolled back (SQL1229) and connections being lost.

- Catalog Partition failure

  At the time the failure occurs, the behavior is the same as without HACMP. After takeover, new connections and transactions are possible again. When the catalog partition reintegrates into the cluster, the force application command issued by the ha_stopscript script will force all applications out, and connections will be lost. If takeover occurs within CONN_ELAPSE * MAX_CONNRETRIES seconds, connections are not lost during takeover. It might be advisable to increase the value of MAX_CONNRETRIES to avoid losing connections during takeover.

4.9.3 Other SP Component Failures

In this section, we provide a quick overview of the effect of some RISC/6000 SP component failures on DB2 UDB EEE as recorded on the tested configuration:

- RISC/6000 SP Ethernet failure

  Since the RISC/6000 SP Ethernet is used as a backup heartbeat network in our setup, it is important that it be available. However, DB2 UDB EEE operation is not directly impacted by failure of the RISC/6000 SP Ethernet.

- PSSP software failure

  The only relevant component for DB2 UDB EEE is the Worm switch daemon (fault_service_Worm_RTG). It has to run in order for the switch to be usable. If this daemon dies on any of the nodes, DB2 loses communications. As seen in “Using AIX Error Notification” on page 191, it is possible to use HACMP's support of AIX error notification mechanism to alleviate this problem.

- Control Workstation failure

  In normal operation, the Control Workstation is not used. However, to restart the switch or restart AMD on a node, the Control Workstation needs to be running. This means that HACMP takeovers and node cluster reintegrations are not possible if the Control Workstation is down. For this reason, is should be protected with the HACWS software.

- Serial line between the nodes and the Control Workstation

  Failure of this serial line has no influence on DB2 UDB EEE.

4.9.4 Failover and Cluster Reintegration Times

Time for failover of node01 is about one to two minutes, assuming no DB2 UDB EEE recovery is necessary. If transactions need to be rolled back that have been running for N minutes before the crash, add N minutes for a rough estimate of the time needed for takeover. Cluster reintegration is a little bit longer, 3 minutes 30 seconds to 4 minutes, the difference coming from the fact that it takes about two minutes to stop node05's database partitions when node01's database partitions have been running on node05.

4.10 Miscellaneous Configuration Issues

This section details some problems that occurred during the configuration of the system.

4.10.1 Use of /etc/netsvc.conf

During Synchronize Cluster Topology, we got an error message:

```
'ERROR: IP label etsvc01 not found on node node01
```

This occurs due to a combination of problems in DB2 UDB EEE and HACMP. At the time of writing, there was a known problem in HACMP V4.2 related to incorrect use or non-existing definition of /etc/netsvc.conf. You can use fully qualified TCP/IP addresses to circumvent the problem.

You must define the /etc/netsvc.conf file with at least one line containing:

```
hosts=local,bind
```

in order to tell TCPIP to look at /etc/hosts first, and pass short TCP/IP names to HACMP. Otherwise, HACMP V4.2 can fail during synchronization if fully qualified names are used. Also, you may have to apply the latest PTFs to DB2 UDB EEE.

4.10.2 Use of /etc/xtab

We used /etc/services as source of permissions for NFS in order to preserve permissions already in /etc/xtab outside of the control of HACMP. This file must be preserved during "telinit a" processing. In the event scripts, acquire_service_addr and acquire_takeover_addr HACMP has code to preserve /etc/xtab. Some versions of these scripts use the `copy` command and therefore fail. You should replace `copy` by `cp`.

4.10.3 NFS Permissions

We used alias to manage HACMP communications over the switch on the same subnet. Under these circumstances, when you are mounting a NFS file system across the switch service address, you must include the corresponding SDR address in the export file, or you will get a permission denied error.

4.11 Scripts Used in the Test Configuration

The scripts listed in this section perform the following tasks:

- “Install the db2.admin File Collection” on page 208
- “Start and Stop DB2 UDB EEE” on page 211
- “Allocate Disks and Logical Volumes” on page 218
- “Synchronize Volume Groups” on page 238

4.11.1 Install the db2.admin File Collection

This is the install_db2_coll.ksh script, which must be made executable.

```
#!/usr/bin/ksh
#-----------------------------------------------------------------------#
# File:         install_db2_coll.ksh
# Version:      1.1.0
#
# Description:  This script installs a file collection (db2.admin by
#               default) unless the file collection is already installed.
##
# Syntax:
#  install_db2_coll.ksh [file_collection_name] [collection_list]
#
#  where 'file_collection_name' is the name of the file
#  collection to be installed. If no file_collection_name
#  is specified, the default collection name of "db2.admin"
#  will be used. 'collection_list' is the name of the list file
#  of collection to be installed. If no file collection_list
#  is specified, the default collection file name of
#  "db2_coll.list" will be used. This parameter is only needed
#  when it is run from the control workstation.
#
# Example:
#  install_db2_coll.ksh db2.admin db2_coll.list
#

ProgramName=$(basename $0)
FileCollection=${1:-db2.admin}
FileCollectionList=${2:-~/db2_coll.list}
```

```
function install_db2_CW {
#  ------------------------------------------------------------------
#  Build and install db2.admin file collection.
#  ------------------------------------------------------------------

if [ ! -f "$FileCollectionList" ]; then
        print "\n\n[$ProgramName]:  File collection List file"\
                            " $FileCollectionList  not found \n\n"
        return 99
    fi
    CollDir=/var/sysman/sup/$FileCollection
    mkdir -p -m755 $CollDir
    chown root.system $CollDir
    set -o noclobber
    /usr/bin/cp /var/sysman/sup/sup.admin/* $CollDir
    /usr/bin/rm -f $CollDir/when $CollDir/last $CollDir/scan
    /usr/bin/cp -p $FileCollectionList $CollDir/list
    set +o noclobber
    cd /var/sysman/sup        # Return to previous directory
    /usr/bin/ln -sf /var/sysman/sup/$FileCollection/list \
                    /var/sysman/sup/lists/$FileCollection
    if [[ ! $(grep -c $FileCollection /var/sysman/file.collections) \
             -gt 0 ]];then
        /usr/bin/ed -s /var/sysman/file.collections <<-EOF!
    $
    a
    # ------------------------------------------------------------------
    # $FileCollection - file collection to sync files
    # ------------------------------------------------------------------
    primary $FileCollection       - / - / EO power no
    .
    w
    q
    EOF!
    fi
    return
}

function install_db2_SPnode {
    print "\n\n[$ProgramName]: Installing collection on node.\n\n"
    /var/sysman/supper update sup.admin
    /var/sysman/supper install $FileCollection
    if [[ $? -ne 0 ]]; then
        print "\n\n[$ProgramName]:  Install of file collection"\
                            "$FileCollection FAILED, RC=$RC\n\n"
    fi
    /var/sysman/supper update  $FileCollection
    return $RC
}
# ------------------------------------------------------------------
#      M A I N
# ------------------------------------------------------------------
{
# Check parameters
```

```
    if [[ "$(/var/sysman/supper status \
                | awk '$1==FILEC {print $2}' FILEC="$FileCollection")" \
                = Yes ]]; then
        print "\n\n[$ProgramName]:  File collection $FileCollection is "\
                "already installed\n\n"
    else
        print "\n\n[$ProgramName]:  Installing file collection"\
                " $FileCollection.\n\n"
        if [[ "$(/usr/lpp/ssp/install/bin/node_number)" -eq 0 ]]; then
            install_db2_CW
        else
            install_db2_SPnode
        fi
    fi
} 2>&1

exit
```

Important: After the collection is defined in the Control Workstation, copy the script to each node and run it to install the collection in the nodes. If you want this collection to be updated automatically, you must add the following line in each node in the root crontab file:

```
10 * * * * /var/sysman/supper update [collec_name] >/dev/null 2>&1
```

where: `[collec_name]` is the name of the collection.

Make sure that the user ID running the `supper` command has read access to any files in the file collection.

4.11.1.1 List Collection File Used by the db2.admin Collection

This file is called db2_coll.list:

```
symlinkall
always  ./etc/services
always  ./etc/security/limits
always  ./etc/inetd.conf
execute /usr/bin/refresh -s inetd >/dev/null 2>&1 (./etc/inetd.conf)
always  ./etc/syslog.conf
execute /usr/bin/refresh -s syslogd >/dev/null 2>&1 (./etc/syslog.conf)
always  ./etc/tftpaccess.ctl
always  ./etc/ftpusers
always  ./etc/resolv.conf
always  ./etc/netsvc.conf
always  ./etc/hosts
always  ./etc/environment
upgrade ./etc/aliases
execute /usr/sbin/newaliases >/dev/null 2>&1 (./etc/aliases)
upgrade ./.profile
upgrade ./.kshrc
upgrade ./.sh_logout
```

4.11.2 Start and Stop DB2 UDB EEE

There are two scripts here, one to start DB2 and to stop DB2.

4.11.2.1 Start DB2 UDB EEE on each resource

This script, start_db2.ksh, must be made executable.

```
#--------------------------------------------------------------------#
# File:         start_db2.ksh
# Version:      1.1.0
#
# Description:  This script starts db2 UDB EEE, allowing start of each
#               partition separately for recovery purposes.
#
# Syntax:       # start_db2 instance resource config conf_file
#
#               where 'instance' is the name of the DB2 instance
#               and 'resource' is the HACMP resource name to be
#               started and 'conf_file' is the file that describes
#               the configuration
#
# Example:      # start_db2 db2inst1 resource09
#
#--------------------------------------------------------------------
#
function start_one_dp {
# Start Data Partition
# Args Instance NumDP NewDir NewPort NewSw OldDir OldPort OldSw
    RUN='ps -fu $1 |grep db2sysc | awk '{print $9;}' \
                                 | grep $2 |grep -v grep'
    if [ "$RUN" != "" ]; then
        print "\n\n[$ProgramName]:In $1 Partition $2 already running.\n\n"
        return
    fi
    #Needs restart or only a start?
    if [ "$5" = "$8" -a "$3" = "$6" ] then
       MYRESTART=""
       MYMSG='Starting'
    else
       MYMSG='Restarting'
       MYRESTART=" restart hostname $3 netname $5 port $4 "
    fi
    # temp file used to issue the db2 start command
    TEMP_FILE=/tmp/start_DB2_$1_$2
    rm -rf $TEMP_FILE
    cat >| $TEMP_FILE << EOF
j=0
print "\\n[$ProgramName]: $MYMSG $1 Partition $2\\n" \\
"               on $3 through interface $5 port $4.\\n"
while (( j == 0 )) do
if ! db2start nodenum $2 $MYRESTART |grep SQL6036
# start or stop in progress
then
j=1
```

```
fi
done
EOF
    # make temp file executable
    chmod ogu+x $TEMP_FILE
    # execute as user DB2 administrator
    su - $1 -c "$TEMP_FILE"
}
# ------------------------------------------------------------------------
#                    M A I N
# ------------------------------------------------------------------------
#!/bin/ksh
# get parameters  Instance Resource and Config File
ProgramName=$(basename $0)
MYINSTANCE=${1:-db2inst1}
RESOURCE=${2:-resource01}
FileConfigStart=${3:-db2nodes.cfg.hacmp}

# Test instance name requested
if [ ! '/usr/lpp/db2_05_00/instance/db2ilist|grep $MYINSTANCE' ]; then
    print "\n\n[$ProgramName]: $MYINSTANCE is not a valid instance "\
          "name.\n\n"
    exit 98
fi

# Look up instance home directory
MYHOMEDIR='su - $MYINSTANCE -c 'echo $HOME' 2>/dev/null '

# Look for start map file
if [ ! -f "$FileConfigStart" ]; then
    print "\n\n[$ProgramName]:  DB2-HACMP start config file"\
          " $FileConfigStart not found \n\n"
    return 99
fi

# Build config of this resource
MYRESOURCES="/tmp/start_db2_"$RESOURCE
rm -rf $MYRESOURCES
grep -v '^#' $FileConfigStart | \
  awk '{ if ( $1 == "'$RESOURCE'" ) print $2,$3,$4,$5;}' >$MYRESOURCES

# what SP node are we on, look for alias choose in config tables
HOSTALIAS='hostname'
HOSTALIAS='host $HOSTALIAS |awk '{print $1,$5,$6,$7,$8;}''
MYHOST=""
for i in 'awk '{print $2;}' $MYRESOURCES'
do
    j='echo $HOSTALIAS | grep $i'
    if [ "$j" != "" ]; then
       MYHOST=$i
       break
    fi
done
if [ $MYHOST = "" ]; then
```

```
    print "\n\n[$ProgramName]: $RESOURCE is not configured in this node"\
          " \n\n"
    return 99
fi

# Check DB2 Logical Volume Permissions
VGLST=‘/usr/sbin/cluster/utilities/clshowres -g \
           "$RESOURCE"|grep ’^Volume Groups’‘
MYGRP=‘id $MYINSTANCE |awk ’{print $2;}’|cut -f 2 -d ’(’|cut -f 1 -d ’)’‘
echo "Seting DB2 access to raw LVs with type equal udb"
for vg in ‘echo $VGLST|sed -e ’{s/^Volume Groups//;}’‘ do
    MYDB2LVS=‘lsvg -l $vg \
              | awk ’{if ( $2 == "udb" ) print "/dev/r"$1" ";}’‘
    chown $MYINSTANCE.$MYGRP $MYDB2LVS
done

# Select resources for this host
awk ’{ if ($2=="’$MYHOST’") print $1,$2,$3,$4;}’ $MYRESOURCES
>$MYRESOURCES.dp

# Which partitions ?
MYDPS=‘awk ’{print $1;}’ $MYRESOURCES.dp‘
# Start them
for DPNUM in $MYDPS
do
    #Compute new parameters
    MYNEW=‘awk ’{ if ( $1 == "’$DPNUM’" ) print $2,$3,$4;}’ \
           $MYRESOURCES.dp‘
    MYDB2NODES=$MYHOMEDIR"/sqllib/db2nodes.cfg"
    MYOLD=‘cat $MYDB2NODES | awk ’{if ($1 == ’$DPNUM’) print $2,$3,$4;}’‘
    start_one_dp  $MYINSTANCE $DPNUM $MYNEW $MYOLD
done
```

Note: This function uses a config file (conf_file) in order to for each start option. The data is similar to db2nodes.cfg file, but each row is preceded by the resource name. In the form:

```
resource data_partition ip_address port_num switch_ip_address
```

When HACMP calls this start script with the resource and hostname, all the database partitions defined in the resource will start. You must take care to specify for each HACMP resource and database partition the starting conditions for normal operation and also for each takeover or rotating situation possible. You need one for each resource, hostname and database partition involved. Be careful with port planning when many database partitions are taken over by one machine. Port numbers must be different for each partition running on the same node. The script will check db2nodes.cfg and will restart the database partition with the specifications of conf_file. The db2nodes.cfg file will be updated with the new values.

4.11.2.2 Stop DB2 UDB EEE on each resource

This script, start_db2.ksh, must be made executable.

```
#--------------------------------------------------------------------#
# File:         stop_db2.ksh
# Version:      1.1.0
#
# Description:  This script stops db2 UDB EEE, allowing stop of each
#               separately for recovery purposes.
#
# Syntax:       # stop_db2 instance resource conf_file
#               where 'instance' is the name of the DB2 instance
#               and 'resource' is the HACMP resource name stoping
#               and 'conf_file' is the file that describes the
#               configuration
#
# Example:      # stop_db2 db2inst1 resource09

#--------------------------------------------------------------------
#
# Force Applications
function force_applications {
# Args Instance NumDP Node
# If not runnig I quit
    RUN=‘ps -fu $1 |grep db2sysc | awk '{print $9;}' \
                                 | grep $2 |grep -v grep‘
    if [ "$RUN" = "" ]; then
        print "\n\n[$ProgramName]:In $1 Partition $2 already stopped.\n\n"
        return
    fi

    # temp file used to issue the db2 force applications
    TEMP_FILE=/tmp/stop_DB2_$1
    rm -rf $TEMP_FILE
    cat >| $TEMP_FILE << EOF
# force out applications
db2 terminate
export DB2NODE=$2
db2 force applications all
# wait for applications to finish
j=0
print "\\n[$ProgramName]: Forcing all applications on $1 nodenum $2\\n."
while (( j == 0 ))
do
if db2 list applications | grep SQL1611 || db2 list applications | grep
SQL1032
then
j=1
fi
done
EOF

    # make temp file executable
```

```
    chmod ogu+x $TEMP_FILE
    # execute as user DB2 administrator
    su - $1 -c "$TEMP_FILE"
}
#--------------------------------------------------------------------
#
function stop_one_dp {
# Start Data Partition
# Args Instance NumDP Node
    RUN=‘ps -fu $1 |grep db2sysc | awk ’{print $9;}’ \
                                 | grep $2 |grep -v grep‘
    if [ "$RUN" = "" ]; then
        print "\n\n[$ProgramName]:In $1 Data Partition $2 already
stoped.\n\n"
        return
    fi
    # temp file used to issue the db2 stop command
    TEMP_FILE=/tmp/stop_DB2_$1_$2
    rm -rf $TEMP_FILE
    cat >| $TEMP_FILE << EOF
# Stopping data partition
j=0
print "\\n[$ProgramName]: Stopping $1 Partition $2 on $3.\\n"
while (( j == 0 ))
do
if ! db2stop nodenum $2 |grep SQL6036
# start or stop in progress
then
j=1
fi
done
EOF
    # make temp file executable
    chmod ogu+x $TEMP_FILE
    # execute as user DB2 administrator
    su - $1 -c "$TEMP_FILE"
}
# --------------------------------------------------------------------
#                T H E    M A I N    L I N E
# --------------------------------------------------------------------
#!/bin/ksh
# get parameters  Instance Resource and Config File
ProgramName=$(basename $0)
MYINSTANCE=${1:-db2inst1}
RESOURCE=${2:-resource01}
FileConfigStart=${3:-db2nodes.cfg.hacmp}

# Test instance name requested
if [ ! ‘/usr/lpp/db2_05_00/instance/db2ilist|grep $MYINSTANCE‘ ]; then
    print "\n\n[$ProgramName]: $MYINSTANCE is not a valid instance "\
          " name.\n\n"
    exit 98
fi
```

```
# Look up instance home directory
MYHOMEDIR=‘su - $MYINSTANCE -c ’echo $HOME’ 2>/dev/null‘

# Look for stop map file
if [ ! -f "$FileConfigStart" ]; then
    print "\n\n[$ProgramName]:  DB2-HACMP stop config file"\
          " $FileConfigStart not found \n\n"
    return 99
fi

# Check for DB2 code access
if [ $MYHOMEDIR = "/home/guest" ]; then
    print "\n\n[$ProgramName]: DB2 code not accesible, allow HACMP "\
          "to kill DB2"
    return 98
fi

# Build config of this resource
MYRESOURCES="/tmp/stop_db2_"$RESOURCE
rm -rf $MYRESOURCES
grep -v ’^#’ $FileConfigStart | \
    awk ’{ if ( $1 == "’$RESOURCE’" ) print $2,$3,$4,$5;}’ >$MYRESOURCES

# what SP node are we on, look for alias choose in config tables
HOSTALIAS=‘hostname‘
HOSTALIAS=‘host $HOSTALIAS |awk ’{print $1,$5,$6,$7,$8;}’‘
MYHOST=""
for i in ‘awk ’{print $2;}’ $MYRESOURCES‘ do
    j=‘echo $HOSTALIAS | grep $i‘
    if [ "$j" != "" ]; then
        MYHOST=$i
        break
    fi
done
if [ $MYHOST = "" ]; then
    print "\n\n[$ProgramName]:  $RESOURCE is not configured in this node"\
          " \n\n"
    return 99
fi

# Select resources for this host
awk ’{ if ($2=="’$MYHOST’") print $1,$2,$3,$4;}’ $MYRESOURCES
>$MYRESOURCES.dp

# Which partitions ?
MYDPS=‘awk ’{print $1;}’ $MYRESOURCES.dp‘
# Force applications connected to DB2 data partition.
for DPNUM in $MYDPS
do
    # Compute new parameters
    MYNEW=‘awk ’{ if ( $1 == "’$DPNUM’" ) print $2,$3,$4;}’ \
           $MYRESOURCES.dp‘
    # Stop all applications
    force_applications $MYINSTANCE $DPNUM
```

```
done
# Stop each data partition on the data partition server
for DPNUM in $MYDPS
do
    # Compute new parameters
    MYNEW=‘awk ’{ if ( $1 == "’$DPNUM’" ) print $2,$3,$4;}’ \
           $MYRESOURCES.dp‘
    stop_one_dp  $MYINSTANCE $DPNUM $MYNEW
done
```

This script uses the same resource file as start_db2.ksh. All the database partitions defined in the resource will be stopped.

4.11.2.3 Configuration File for DB2 Start and Stop Scripts

This is conf_file:

```
# Resource name,Data partition,Node name,Port Number,Switch Name
# Normal conditions
resource01 1  hnode01  0 sw01
resource01 2  hnode01  1 sw01
resource01 3  hnode01  2 sw01
resource01 4  hnode01  3 sw01
resource05 5  hnode05  0 sw05
resource05 6  hnode05  1 sw05
resource05 7  hnode05  2 sw05
resource05 8  hnode05  3 sw05
resource09 9  hnode09  0 sw09
resource09 10 hnode09  1 sw09
resource09 11 hnode09  2 sw09
resource09 12 hnode09  3 sw09
resource13 13 hnode13  0 sw13
resource13 14 hnode13  1 sw13
resource13 15 hnode13  2 sw13
resource13 16 hnode13  3 sw13
# Takeover conditions
resource01 1  hnode05  4 sw05
resource01 2  hnode05  5 sw05
resource01 3  hnode05  6 sw05
resource01 4  hnode05  7 sw05
resource05 5  hnode01  4 sw01
resource05 6  hnode01  5 sw01
resource05 7  hnode01  6 sw01
resource05 8  hnode01  7 sw01
resource09 9  hnode13  4 sw13
resource09 10 hnode13  5 sw13
resource09 11 hnode13  6 sw13
resource09 12 hnode13  7 sw13
resource13 13 hnode09  4 sw09
resource13 14 hnode09  5 sw09
resource13 15 hnode09  6 sw09
resource13 16 hnode09  7 sw09
```

4.11.3 Allocate Disks and Logical Volumes

There are two main tasks performed by the scripts in this section:

- “Create Disk Devices” on page 218
- “Create Volume Groups and Logical Volumes” on page 223

4.11.3.1 Create Disk Devices

This script, 7133_Config.ksh, is used is reserve disk numbers to be used at the SP nodes and to use the same logical names for all the nodes in the cluster. For example, hdisk32 will refer to the same device at all the nodes in the cluster.

```
#!/usr/bin/ksh
# --------------------------------------------------------------------
# name:  7133_Config.ksh
#        This script creates "hdisk" and "pdisk" AIX operation system
#        definitions for the SSA 7133 disks.
# --------------------------------------------------------------------

function update_ODM {

  echo "\nChecking ODM for user update of connwhere_shad attribute..."
  for i in $(egrep -v "^#|^$|^ *$" ${Tables_Dir}/${FileName} \
       | awk 'BEGIN {FS=":"} {print $4}' | sort -u); do
      echo $i
      /usr/bin/odmchange -o PdAt -q "attribute=connwhere_shad and \
           uniquetype=pdisk/ssar/$i"  <<-EOF!
PdAt:
        generic = DU
EOF!

      RC=$?
      if [ $RC != 0 ]; then
          echo "\tODM upd of PdAt for /pdisk/ssar/$i failed with RC=$RC"
      fi
  done

  /usr/bin/odmchange -o PdAt -q "attribute=connwhere_shad and \
uniquetype=disk/ssar/hdisk"  <<-EOF!
PdAt:
        generic = DU
EOF!

  RC=$?
  if [ $RC != 0 ]; then
      echo "\tODM upd of PdAt for disk/ssar/hdisk failed with RC=$RC"
  fi

  return 0
}
```

```
function remove_SSA_all {

  for i in $(lsdev -C  \
        |egrep "SSA C Physical Disk Drive|SSA Logical Disk Drive" \
        |cut -f1 -d' ') ; do
    echo "\nRemoving SSA device ${i}..."
    /usr/sbin/rmdev -l ${i} -d
    ReturnCode=$?
    echo "rmdev return code:  ${ReturnCode}"
  done
  return
}

function remove_SSA_table {

  for i in $(egrep -v "^#|^$|^ *$" ${Tables_Dir}/${FileName} \
       | awk 'BEGIN {FS=":"} {print $1, $2}'); do
    echo "\nRemoving SSA device ${i}..."
    /usr/sbin/rmdev -l ${i} -d
    ReturnCode=$?
    echo "rmdev return code:  ${ReturnCode}"
  done
  return
}

function define_hdisk {

  echo "\nMaking hdisk ${HdiskName}..."
  /usr/sbin/mkdev -l ${HdiskName} -p ssar -s ssar -t hdisk -c disk \
        -w ${ConnectionID} \
        -a "connwhere_shad=${ConnectionID}"
  ReturnCode=$?
  echo mkdev hdisk return code:  ${ReturnCode}
  return $ReturnCode
}

function make_PVID {

  echo "\nMaking physical volume ID for ${HdiskName}..."
  /usr/sbin/chdev -l ${HdiskName} -a pv=yes
  ReturnCode=$?
  echo chdev hdisk return code:  ${ReturnCode}
  return $ReturnCode
}

function define_pdisk {

  echo "\nMaking pdisk ${PdiskName}..."
  /usr/sbin/mkdev -l ${PdiskName} -p ssar -s ssar -t ${Type} -c pdisk \
        -w ${ConnectionID} \
        -a "connwhere_shad=${ConnectionID}"
  ReturnCode=$?
  echo mkdev pdisk return code:  ${ReturnCode}
  return $ReturnCode
```

```
}

  SETUP_LOG=${SETUP_LOG:-"/tmp/SSA_Config.PID$$"}
  Msg_Prefix="[$(basename $0)]"
  Tables_Dir="/admin/ITSO_setup"         # Location of defn table
  typeset -L1 first_character
{

  echo "\n\n${Msg_Prefix}: Defining 7133 SSA devices on $(date)."

# mount jid030e0:/home/latimer /mnt

  while [[ ! -f ${Tables_Dir}/${FileName} ]] ; do
    echo "\nEnter configuration file name ==> \c"
    read -s FileName x
  done

  echo "\n\n${Msg_Prefix}: Using table file \"${Tables_Dir}/${FileName}.\"
"

# --------------------------------------------------------------------
# Make sure ODM allows user update
# --------------------------------------------------------------------
  update_ODM
  sleep 4             # Allow time to read msgs before following menu

# --------------------------------------------------------------------
# Ask if any SSA devices should be removed.
# --------------------------------------------------------------------

   SSA_BOLD=$(tput smso)
   SSA_NORM=$(tput rmso)
   PS3="
Enter selection number:  "
   clear
   echo "\n\tSelect an option to remove SSA pdisks/hdisks:\n"
   select one in \
        "Remove only those SSA hdisks and pdisks listed in the table" \
        "Remove ${SSA_BOLD}ALL${SSA_NORM} SSA hdisks and pdisks and
continue" \
       "Remove ${SSA_BOLD}ALL${SSA_NORM} SSA hdisks and pdisks; then QUIT"
\
        "Do not remove any SSA pdisks/hdisks" \
        "quit"
        do
       case $REPLY in
           1) remove_SSA_table;break;;
           2) remove_SSA_all;break;;
           3) remove_SSA_all;exit 0;;
           4) break;;
           5) exit 0:;;
           *) ;;
       esac
   done
```

```
   unset SSA_BOLD
   unset SSA_NORM

# --------------------------------------------------------------------
# Make SSA hdisks and pdisks
# --------------------------------------------------------------------
  while read Table_row
    do
      first_character=${Table_row}
      if [ ${first_character} != '#' ] ; then  # Skip comments
        Table_row_fields=$(echo ${Table_row}|tr ':' ' ')
        set -- ${Table_row_fields}
        HdiskName=${1}          # Field 1 is "hdisk"
        PdiskName=${2}          # Field 2 is "hdisk"
        ConnectionID=${3}       # Field 3 is "connection ID"
        Type=${4}               # Field 4 is physical device type

        define_hdisk && make_PVID
        define_pdisk
      fi
    done  < $Tables_Dir/${FileName}

  echo "\n${Msg_Prefix}:  7133 SSA device setup completed on $(date)."\
       "\n\tSee file ${SETUP_LOG} for details.\n"
} 2>&1 |tee -a ${SETUP_LOG}

exit
```

4.11.3.2 Map disk addresses to logical names

This file is called node01.ssa and is used by the script 7133_config.ksh above with the map of logical name, physical name and disk ID.

```
# ------------------------------------------------------#
SSA Disk Configuration for 9076 SP2 nodes 01, 05, 09, & 13
# ------------------------------------------------------#
# -->tp3an01 adapter 2 port A2
#
hdisk29:pdisk27:0004AC9E48D800D:4000mbC:
hdisk14:pdisk12:0004AC9E110300D:4000mbC:
hdisk17:pdisk15:0004AC9E112500D:4000mbC:
hdisk19:pdisk17:0004AC9E112A00D:4000mbC:
# to tp3an05 adapter 2 port A1-->A2
hdisk24:pdisk22:0004AC9E1AE900D:4000mbC:
hdisk20:pdisk18:0004AC9E19EA00D:4000mbC:
hdisk22:pdisk20:0004AC9E19F700D:4000mbC:
hdisk7 :pdisk5 :0004AC9E08B200D:4000mbC:
# to tp3an09 adapter 2 port A1-->A2
hdisk18:pdisk16:0004AC9E112700D:4000mbC:
hdisk16:pdisk14:0004AC9E111300D:4000mbC:
hdisk21:pdisk19:0004AC9E19F100D:4000mbC:
hdisk25:pdisk23:0004AC9E1AEA00D:4000mbC:
# to tp3an13 adapter 2 port A1-->A2
```

```
hdisk15:pdisk13:0004AC9E111200D:4000mbC:
hdisk8 :pdisk6 :0004AC9E08D300D:4000mbC:
hdisk13:pdisk11:0004AC9E10F500D:4000mbC:
hdisk23:pdisk21:0004AC9E1AE600D:4000mbC:
#  to tp3an01 adapter 2 port A1-->
#
# -->tp3an01 adapter 2 port B2
hdisk5 :pdisk3 :0004AC7C548400D:4000mbC:
hdisk4 :pdisk2 :0004AC7C547A00D:4000mbC:
hdisk30:pdisk28:0004AC9E579300D:4000mbC:
hdisk31:pdisk29:0004AC9E579600D:4000mbC:
# to tp3an05 adapter 2 port B1-->B2
hdisk33:pdisk31:0004AC9E602000D:4000mbC:
hdisk32:pdisk30:0004AC9E5FFD00D:4000mbC:
hdisk2 :pdisk0 :0004AC7C542B00D:4000mbC:
hdisk3 :pdisk1 :0004AC7C547700D:4000mbC:
# to tp3an09 adapter 2 port B1-->B2
hdisk27:pdisk25:0004AC9E1B7F00D:4000mbC:
hdisk6 :pdisk4 :0004AC9E06F900D:4000mbC:
hdisk26:pdisk24:0004AC9E1B6B00D:4000mbC:
hdisk28:pdisk26:0004AC9E1B9100D:4000mbC:
# to tp3an13 adapter 2 port B1-->B2
hdisk9 :pdisk7 :0004AC9E092D00D:4000mbC:
hdisk11:pdisk9 :0004AC9E092F00D:4000mbC:
hdisk10:pdisk8 :0004AC9E092E00D:4000mbC:
hdisk12:pdisk10:0004AC9E093700D:4000mbC:
#  to tp3an01 adapter 2 port B1-->
#
# -->tp3an01 adapter 3 port A2
hdisk39:pdisk37:0004AC7C537100D:4000mbC:
hdisk57:pdisk55:0004AC9E2DE300D:4000mbC:
hdisk40:pdisk38:0004AC7C537500D:4000mbC:
hdisk45:pdisk43:0004AC7C53B800D:4000mbC:
# to tp3an05 adapter 3 port A1-->A2
hdisk37:pdisk35:0004AC7C534200D:4000mbC:
hdisk63:pdisk61:0004AC9E599200D:4000mbC:
hdisk56:pdisk54:0004AC9E23A100D:4000mbC:
hdisk62:pdisk60:0004AC9E328600D:4000mbC:
# to tp3an09 adapter 3 port A1-->A2
hdisk52:pdisk50:0004AC7C542200D:4000mbC:
hdisk59:pdisk57:0004AC9E301B00D:4000mbC:
hdisk58:pdisk56:0004AC9E2FE000D:4000mbC:
hdisk47:pdisk45:0004AC7C540300D:4000mbC:
# to tp3an13 adapter 3 port A1-->A2
hdisk51:pdisk49:0004AC7C542000D:4000mbC:
hdisk43:pdisk41:0004AC7C538200D:4000mbC:
hdisk64:pdisk62:0004AC9E59EE00D:4000mbC:
hdisk53:pdisk51:0004AC7C543400D:4000mbC:
#  to tp3an01 adapter 3 port A1-->
#
# -->tp3an01 adapter 4 port B2
hdisk38:pdisk36:0004AC7C536300D:4000mbC:
hdisk61:pdisk59:0004AC9E321000D:4000mbC:
hdisk54:pdisk52:0004AC7C546800D:4000mbC:
```

```
hdisk60:pdisk58:0004AC9E30EF00D:4000mbC:
# to tp3an05 adapter 4 port B1-->B2
hdisk44:pdisk42:0004AC7C539300D:4000mbC:
hdisk46:pdisk44:0004AC7C53BC00D:4000mbC:
hdisk35:pdisk33:0004AC7C532900D:4000mbC:
hdisk50:pdisk48:0004AC7C541900D:4000mbC:
# to tp3an09 adapter 4 port B1-->B2
hdisk48:pdisk46:0004AC7C540500D:4000mbC:
hdisk41:pdisk39:0004AC7C537800D:4000mbC:
hdisk49:pdisk47:0004AC7C540B00D:4000mbC:
hdisk65:pdisk63:0004AC9E5A0B00D:4000mbC:
# to tp3an13 adapter 4 port B1-->B2
hdisk55:pdisk53:0004AC9DCD6600D:4000mbC:
hdisk34:pdisk32:0004AC7C532600D:4000mbC:
hdisk36:pdisk34:0004AC7C533E00D:4000mbC:
hdisk42:pdisk40:0004AC7C537C00D:4000mbC:
#  to tp3an01 adapter 4 port B1-->
```

4.11.3.3 Create Volume Groups and Logical Volumes

This script, setup_VG.ksh, creates the required VGs and LVs required. We create two volume groups on each node, for a total of 16 volumes groups, 16 Journaled File System logs, 16 logs file systems, 32 temporary table spaces, 30 large data table spaces, 30 large index table spaces, 30 small data table spaces, and 30 small index table spaces, plus space for catalogs and so on.

```
#!/usr/bin/ksh
# ----------------------------------------------------------------------
# name:  setup_VG.ksh
# author:   George Latimer, IBM
# date:     02/17/98
# version:  1.0.0
# ------------------------------------------------------
# Add db2untag to all rlvs
# Verify correct LV seq numbers and sizes.  Check disks to
# see how they lay out.
# ------------------------------------------------------
# M O D I F I C A T I O N S:
#
# 18 Feb 1998:  Added deletion of lost+found from UDB SMS containers.
# ------------------------------------------------------
# This script sets up UDB volume groups, logical volumes, and
# filesystems.  Log files and catalog tablespace will be mirrored.
# Catalog tablespace will be on db partition 1.
# Single partition tables will be on db partition 1.
# Multiple partition tables will be split across db partitions 2 thru
# 16 (15 partition nodegroup).
# ADSM and coordinator node will be SP physical node 01 (host=tp3an05).
# ----------------------------------------------------------------------

set -x
INSTANCE=db2inst1           #     <------------------------------ Verify!
GROUP=dbadmin1              #     <------------------------------ Verify!
```

```
VG_DELETE_FILE=/admin/ITSO_setup/delete_VGs.tab  # Optional file
INSTHOME=${INSTHOME:-~$INSTANCE}   # Home directory of UDB instance owner
THIS_HOST=$(hostname -s)     # Short hostname of this host

typeset -Z2 NODE_NUMBER=$(hostname -s|cut -c6-7)  # 2-digit SP node number
LVS_TO_SYNC=""               # initialize

UNTAG_PGM=$INSTHOME/sqllib/misc/db2untag   # Location of UDB db2untag
program
LOGS_HLQ=/DB_LOG          # High-level qualifier of path for UDB log
                          # files.  To this will be appended instance
                          # name and database partition, eg.
                          # <hlq>/<instance>/NODEnnnn where nnnn=partition
                          # number.
TEMP_HLQ=/DB_TMP          # High-level qualifier of path for tempspace
                          # files.  To this will be appended instance
                          # name and database partition, eg.
                          #<hlq>/<instance>/NODEnnnn where nnnn=partition
                          # number.
case "$(/usr/lpp/ssp/install/bin/node_number)" in
      0)
          echo "\n$(basename $0):  Do not run on the Control Workstation."\
               "\n----- ABORTING -----"
          exit 1
          ;;
      1|5|9|13)
          ;;
      *)
          echo "\n$(basename $0):  Must be run on an SP node."\
               "\n----- ABORTING -----"
          exit 1
          ;;
esac
if [[ "$(whoami)" != "root" ]]; then
      print "\n$(basename $0):  You must be root to run this script."\
            "\n----- ABORTING -----"
      exit 2
fi

trap 'sleep 3'  0 1 2 3 15          # Bad things can happen to ODM if
                                      # cancel during LVM update!
. /admin/ITSO_setup/disk_table.ksh  # Read hdisks per VG per SP node

if [[ -r $INSTHOME/sqllib/db2nodes_cfg ]]; then
      NODES_CFG=$INSTHOME/sqllib/db2nodes_cfg
else
      NODES_CFG=/admin/ITSO_setup_nodes.cfg
fi

function blow_it_all_away {
#
# This function removes everything allocated for UDB.  Any JFS files
# (like logs, tempspace, etc) are unmounted and removed.  Then for each
```

```
# UDB volume group, all logical volumes -- including any jfslog(s) -- are
# removed. Then the volume group itself is removed.

if [[ -e  "$VG_DELETE_FILE" ]]; then
      VGs_to_delete=$(grep "vg_n${NODE_NUMBER}_" $VG_DELETE_FILE)
else
      VGs_to_delete=$(set|grep "vg_n${NODE_NUMBER}_"|cut -f1 -d'=')
fi


#  To delete LV's and filesystems, need to have VG varied on.
for vg in ${VGs_to_delete}; do
      varyonvg $vg
done
# First, determine JFS filenames.  Unmount filesystem if mounted.  Then
# remove the filesystem and it's mount point (this should also remove the
# underlying logical volume).
#
# N O T E:  The following DOES NOT DELETE /home/tp3an01!
#

for i in $(lsfs|tail +2|egrep "$LOGS_HLQ/|$TEMP_HLQ/"|awk '{print $3}');
do
      if [[ ! -z "$i" ]]; then
          print "\n$(basename $0):  Removing filesystem $i..."
          umount $i  >/dev/null 2>&1
          rmfs -r $i
          RC=$?
          if [[ $RC != 0 ]]; then
             print "\n$(basename $0):  ERROR:  Could not remove filesystem
$i."
            else
              print "\n$(basename $0):  Filesystem $i deleted OK."
          fi
      fi
done

### Delete all logical volumes, including jfslog.  (To remove jfslog,
### all filesystems must be previously removed.  This should hav been done
### in preceeding step.)
for VG in ${VGs_to_delete}; do
      for LV in $(lsvg -l $VG|tail +3|cut -f1 -d' '); do
          print "\n$(basename $0):  Removing LV $LV in VG $VG...."
          rmlv -f $LV
          RC=$?
          if [[ $RC != 0 ]]; then
              print "\n$(basename $0):  ERROR:  Could not remove LV $LV."
            else
              print "\n$(basename $0):  Logical volume $LV deleted OK."
          fi
      done
      print "\n$(basename $0):  Removing VG $VG...."
      reducevg -df $VG $(lsvg -p $VG|tail +3|cut -f1 -d' ')
      RC=$?
```

```
      if [[ $RC != 0 ]]; then
          print "\n$(basename $0):  ERROR:  Could not remove VG $VG."
        else
          print "\n$(basename $0):  Volume Group $VG deleted OK."
      fi
done
unset VGs_to_delete

### Everything should be removed!  Ready to allocate.

return 0
}

function create_VG {
export PARTN_SIZE=${3:-8}           # Default is 8MB partition

echo "\nCreating volume group $1 with $2..."
COMMAND="mkvg -f -y $1 -s${PARTN_SIZE} $2"
print "\n$(basename $0):COMMAND=$COMMAND"
VG=$(eval $COMMAND)
RC=$?
if [ $RC != 0 ]; then
      echo "\tmkvg command failed for VG $1, RC=$RC."\
           "\n\t----- ABORTING -----"
      exit 3
fi

return 0
}

function alloc_jfslog {

echo "\nCreating jfslog $1..."
COMMAND="mklv -y $1 -t jfslog -m /tmp/map_jfslog $VG 1"
print "\n$(basename $0):COMMAND=$COMMAND"
eval $COMMAND
RC=$?
if [[ "$RC" != 0 ]]; then
      echo "\tmklv for jfslog $1 FAILED, RC=$RC."\
           "\n\t----- ABORTING -----"
      exit 4
fi

yes | /usr/sbin/logform /dev/$1

return 0
}

function alloc_jfslog_mirror {


COMMAND="mklvcopy -u 2 -m /tmp/map_jfslog $1 2"
print "\n$(basename $0):COMMAND=$COMMAND"
eval $COMMAND
```

```
RC=$?
if [[ "$RC" != 0 ]]; then
      echo "\tmklvcopy for jfslog $1 FAILED, RC=$RC."\
           "\n\t----- ABORTING -----"
      exit 4
fi

syncvg -l $1

return 0
}
function create_LV {
# p1=LVname; p2=VG; p3=map_name; p4=#partitions; p5=LVtype (jfs, udb, etc)

echo "\nCreating logical volume $1 ..."
COMMAND="mklv -y $1 $6 -t $5 -m $3 $2 $4"
print "\n$(basename $0):  COMMAND=$COMMAND"
eval "$COMMAND"
RC=$?
if [[ "$RC" != 0 ]]; then
      echo "\tmklv for logical volume $1 FAILED, RC=$RC."
fi

return 0
}

function untag_container {
# p1=LVname

echo "\nUntagging container $1 ..."
$UNTAG_PGM -f $1
RC=$?
if [[ "$RC" != 0 ]]; then
      echo "\tuntag for logical volume $1 FAILED, RC=$RC."
fi

return 0
}


# --------------------------------------------------------------------
# main        MAIN         Main
# --------------------------------------------------------------------

set -x

SETUP_LOG=${SETUP_LOG:-"/tmp/setup_VG_beta2.PID$$"}
Msg_Prefix="[$(basename $0)]"

{

# INITIALIZATION:
#     Blow away all filesystems, jfs logical volumes, raw logical volumes,
#     raw logical volumes.  Need to unmount any UDB filesystems first.
```

```
#     To be safe, probably want to run untag after re-allocation.
#

blow_it_all_away

#
# SETUP VOLUME GROUP(s)
#     1.  How many VG's per physical node?  One per database node?  One
#             per SP physical node.  One per 16 drives?
#     2.  LV and its mirror must be in same VG.


VGs_to_alloc=$(set|grep "^vg_n${NODE_NUMBER}_"|cut -f1 -d'=')
for i in ${VGs_to_alloc}; do
      print $i
      if [[ -z "$(lsvg|egrep "^${i}$")" ]]; then  # if VG does not exist
          disks="$(eval echo \$$i)"               # ...create it
          create_VG $i "$disks"
        else                                      # else, print message
          print "\n$(basename $0):  VG $i will not be made.  VG already
exists."
      fi
done

# SETUP JFSLOG, LOGICAL VOLUMES, FILESYSTEMS, mount filesystems, run untag.

PARTITIONS_THIS_HOST=$(grep " ${THIS_HOST} " $NODES_CFG \
          |awk '{print $1}')
PARTITIONS_COUNT=$(echo ${PARTITIONS_THIS_HOST}|wc -w|tr -d ' ')
if [[ "${PARTITIONS_COUNT}" -ne 4 ]]; then
      print "$(basename $0):  Expected 4 partitions per host.  Found "\
            "${PARTITIONS_COUNT}"\
            "\ndefined database partitions for this host, ${THIS_HOST}."\
"\n---- ABORTING ----"
      exit 5
fi
for PARTITION in ${PARTITIONS_THIS_HOST}; do
      MOD_2=$(( $PARTITION % 2 ))   # Determine VG
      MOD_4=$(( $PARTITION % 4 ))   # Determines numbering of LV's
      typeset -Z4 PARTITION_4DIGITS=$PARTITION # used for NODEnnnn

      print "\n$(basename $0): ---------------------------------------"\
            "\n\tUDB DATABASE PARTITION:  $PARTITION"\
           "\n-------------------------------------------------------\n"
      case $PARTITION in
          1)
            VG=vg_n${NODE_NUMBER}_01
            LV_PREFIX="lv_$(echo $VG|cut -c4-9)_"
            set -A VG_DISKS $(eval echo \$$VG)
            set -A DISKS ${VG_DISKS[0]} ${VG_DISKS[1]} ${VG_DISKS[6]} \
                ${VG_DISKS[7]}
            /admin/ITSO_setup/setup_VG_maps_catalog.ksh ${DISKS[*]}
            typeset -Z3 SEQ_NO=101
            alloc_jfslog ${LV_PREFIX}log $VG   # jfslog for VG1
```

```
            ;;
          5|9|13)
            VG=vg_n${NODE_NUMBER}_01
            LV_PREFIX="lv_$(echo $VG|cut -c4-9)_"
            set -A VG_DISKS $(eval echo \$$VG)
            set -A DISKS ${VG_DISKS[0]} ${VG_DISKS[1]} ${VG_DISKS[6]} \
                ${VG_DISKS[7]}
            /admin/ITSO_setup/setup_VG_maps.ksh ${DISKS[*]}
            typeset -Z3 SEQ_NO=103
            alloc_jfslog ${LV_PREFIX}log $VG   # jfslog for VG1
            ;;
          2|6|10|14)
            VG=vg_n${NODE_NUMBER}_01
            LV_PREFIX="lv_$(echo $VG|cut -c4-9)_"
            set -A VG_DISKS $(eval echo \$$VG)
            set -A DISKS ${VG_DISKS[4]} ${VG_DISKS[5]} ${VG_DISKS[2]} \
                ${VG_DISKS[3]}
            /admin/ITSO_setup/setup_VG_maps.ksh ${DISKS[*]}
            typeset -Z3 SEQ_NO=203
                                        # next mirrors log for 1,5,9 & 13
            alloc_jfslog_mirror ${LV_PREFIX}log  # mirror jfslog for VG1
            ;;
          3|7|11|15)
            VG=vg_n${NODE_NUMBER}_02
            LV_PREFIX="lv_$(echo $VG|cut -c4-9)_"
            set -A VG_DISKS $(eval echo \$$VG)
            set -A DISKS ${VG_DISKS[0]} ${VG_DISKS[1]} ${VG_DISKS[6]} \
                ${VG_DISKS[7]}
            /admin/ITSO_setup/setup_VG_maps.ksh ${DISKS[*]}
            typeset -Z3 SEQ_NO=303
            alloc_jfslog ${LV_PREFIX}log $VG   # jfslog for VG1
            ;;
          4|8|12|16)
            VG=vg_n${NODE_NUMBER}_02
            LV_PREFIX="lv_$(echo $VG|cut -c4-9)_"
            set -A VG_DISKS $(eval echo \$$VG)
            set -A DISKS ${VG_DISKS[4]} ${VG_DISKS[5]} ${VG_DISKS[2]} \
                ${VG_DISKS[3]}
            /admin/ITSO_setup/setup_VG_maps.ksh ${DISKS[*]}
            typeset -Z3 SEQ_NO=403
                                        # next mirrors log for 3,7,11 & 15
            alloc_jfslog_mirror ${LV_PREFIX}log  # mirror jfslog for VG2
            ;;
          *)
            print "\n$(basename $0):  Partition #$PARTITION is invalid?"\
                  "\n---- ABORTING ----"
            exit 6
            ;;
      esac

### --------------------------------------------------------------------
### disk1             disk2             disk3               disk4
###  .6G SML_I       .6G SML_I         .6G SML_D           .6G SML_D
### 1.2G BIG_D      1.2G BIG_D        1.2G BIG_I          1.2G BIG_I
```

```
###  .6G LOGS        .6G LOGS          .6G LOGS_MIRR       .6G LOGS_MIRR
### jfslog (1)                       jfslog_mirr (1)
### 1.2G TEMP       1.2G TEMP        1.2G TEMP           1.2G TEMP
###
###  Note 1:  jfslog will be allocated disk 1  for partitions 1, 5, 9, & 13
###           jfslog mirror will be allocated on disk 3 for partitions
###           1, 5, 9, & 13
### ------------------------------------------------------------------
if [[ "$PARTITION" = 1 ]]; then

          # -------------------------------------#
          #      Allocate catalog partition      #
          # -------------------------------------#
          #allocate catalog_TS (1 container on disk 1; mirror on disk 3)
          LV=lv_$(echo $VG|cut -c4-9)_101                        # seq 101
          create_LV $LV $VG /tmp/map_catalog 5 udb
          untag_container /dev/r$LV
          mklvcopy -u 2 -m /tmp/map_catalog_mirror $LV 2
#         syncvg -l $LV
          LVS_TO_SYNC="$LVS_TO_SYNC $LV"
          chown -R $INSTANCE.$GROUP /dev/r$LV
#         chmod u+x /dev/r$LV
          #
         #allocate single_node_TS (1 container on disk 2; mirror on disk 4)
          LV=lv_$(echo $VG|cut -c4-9)_102                        # seq 102
          create_LV $LV $VG /tmp/map_single1 5 udb
          untag_container /dev/r$LV
          mklvcopy -u 2 -m /tmp/map_single1_mirror $LV 2
#         syncvg -l $LV
          LVS_TO_SYNC="$LVS_TO_SYNC $LV"
          chown -R $INSTANCE.$GROUP /dev/r$LV
#         chmod u+x /dev/r$LV
          #
          #allocate temp space across all 4 disks
          LV=lv_$(echo $VG|cut -c4-9)_103                        # seq 103
          FS=${TEMP_HLQ}/$INSTANCE/NODE${PARTITION_4DIGITS}/T1
          create_LV $LV $VG /tmp/map_temp1 248 jfs
          crfs -v jfs  -d $LV -m $FS -A yes      \
              -p rw -t yes -a frag=4096 -a nbpi=16384 -a compress=no
          mount $FS
          LV=lv_$(echo $VG|cut -c4-9)_104                        # seq 104
          FS=${TEMP_HLQ}/$INSTANCE/NODE${PARTITION_4DIGITS}/T2
          create_LV $LV $VG /tmp/map_temp2 248 jfs
          crfs -v jfs  -d $LV -m $FS -A yes      \
              -p rw -t yes -a frag=4096 -a nbpi=16384 -a compress=no
          mount $FS
                          # Get rid of AIX 4.2's lost+found directory:
          find ${TEMP_HLQ}/$INSTANCE/NODE${PARTITION_4DIGITS} \
              -name lost+found -exec rmdir {} \;
          chown -R $INSTANCE.$GROUP $TEMP_HLQ
          #
         #Allocate log files (copy 1 on disks 1 & 2; copy 2 on disks 3 & 4)
          LV=lv_$(echo $VG|cut -c4-9)_105                        # seq 105
         FS=${LOGS_HLQ}/$INSTANCE/NODE${PARTITION_4DIGITS}
```

```
            create_LV $LV $VG /tmp/map_log1 150 jfs
            crfs -v jfs  -d $LV -m $FS -A yes      \
                -p rw -t yes -a frag=4096 -a nbpi=16384 -a compress=no
            mklvcopy -u 4 -m /tmp/map_log1_mirror $LV 2
#           syncvg -l $LV
            LVS_TO_SYNC="$LVS_TO_SYNC $LV"
            mount $FS
                            # Get rid of AIX 4.2's lost+found directory:
            find ${TEMP_HLQ}/$INSTANCE/NODE${PARTITION_4DIGITS} \
                -name lost+found -exec rmdir {} \;
            chown -R $INSTANCE.$GROUP $LOGS_HLQ
            #
           #Allocate /home/tp2an01 (copy 1 on disks 1 & 2; copy 2 on disks 3
& 4)
            LV=lv_$(echo $VG|cut -c4-9)_114                         # seq 114
            FS=/home/tp3an01
            create_LV $LV $VG /tmp/map_home 100 jfs
            crfs -v jfs  -d $LV -m $FS -A yes      \
                -p rw -t yes -a frag=4096 -a nbpi=16384 -a compress=no
            mklvcopy -u 4 -m /tmp/map_home_mirror $LV 2
#           syncvg -l $LV
            LVS_TO_SYNC="$LVS_TO_SYNC $LV"
            mount $FS
            chown -R $INSTANCE.$GROUP $LOGS_HLQ

          else

            # -------------------------------------#
            #        Allocate all remaining        #
            #              partitions              #
            # -------------------------------------#

            #allocate temp space across all 4 disks
            LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                     # seq x03
            FS=${TEMP_HLQ}/$INSTANCE/NODE${PARTITION_4DIGITS}/T1
            create_LV $LV $VG /tmp/map_temp1 248 jfs
            crfs -v jfs  -d $LV -m $FS -A yes      \
                -p rw -t yes -a frag=4096 -a nbpi=16384 -a compress=no
            mount $FS
            SEQ_NO=$((SEQ_NO + 1))
            LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                     # seq x04
            FS=${TEMP_HLQ}/$INSTANCE/NODE${PARTITION_4DIGITS}/T2
            create_LV $LV $VG /tmp/map_temp2 248 jfs
            crfs -v jfs  -d $LV -m $FS -A yes      \
                -p rw -t yes -a frag=4096 -a nbpi=16384 -a compress=no
            mount $FS
                            # Get rid of AIX 4.2's lost+found directory:
            find ${TEMP_HLQ}/$INSTANCE/NODE${PARTITION_4DIGITS} \
                -name lost+found -exec rmdir {} \;
            chown -R $INSTANCE.$GROUP $TEMP_HLQ
            #
           #Allocate log files (copy 1 on disks 1 & 2; copy 2 on disks 3 & 4)
            SEQ_NO=$((SEQ_NO + 1))
            LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                     # seq x05
```

```
            FS=${LOGS_HLQ}/$INSTANCE/NODE${PARTITION_4DIGITS}
            create_LV $LV $VG /tmp/map_log1 150 jfs
            crfs -v jfs  -d $LV -m $FS -A yes      \
                -p rw -t yes -a frag=4096 -a nbpi=16384 -a compress=no
            mklvcopy -u 4 -m /tmp/map_log1_mirror $LV 2
#           syncvg -l $LV
            LVS_TO_SYNC="$LVS_TO_SYNC $LV"
            mount $FS
                           # Get rid of AIX 4.2's lost+found directory:
            find ${TEMP_HLQ}/$INSTANCE/NODE${PARTITION_4DIGITS} \
                -name lost+found -exec rmdir {} \;
            chown -R $INSTANCE.$GROUP $LOGS_HLQ
            #
            #allocate large_TS_data (1 container on each of disk 1 & 2)
            SEQ_NO=$((SEQ_NO + 1))
            LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                    # seq x06
            create_LV $LV $VG /tmp/map_large_data1 150 udb
            untag_container /dev/r$LV
            chown -R $INSTANCE.$GROUP /dev/r$LV
#           chmod u+x /dev/r$LV

            SEQ_NO=$((SEQ_NO + 1))
            LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                    # seq x07
            create_LV $LV $VG /tmp/map_large_data2 150 udb
            untag_container /dev/r$LV
            chown -R $INSTANCE.$GROUP /dev/r$LV
#           chmod u+x /dev/r$LV
            #
            #allocate large_TS_index (1 container on each of disk 3 & 4)
            SEQ_NO=$((SEQ_NO + 1))
            LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                    # seq x08
            create_LV $LV $VG /tmp/map_large_index1 150 udb
            untag_container /dev/r$LV
            chown -R $INSTANCE.$GROUP /dev/r$LV
#           chmod u+x /dev/r$LV

            SEQ_NO=$((SEQ_NO + 1))
            LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                    # seq x09
            create_LV $LV $VG /tmp/map_large_index2 150 udb
            untag_container /dev/r$LV
            chown -R $INSTANCE.$GROUP /dev/r$LV
#           chmod u+x /dev/r$LV
            #
            #allocate small_TS_data (1 each on disk 3 & 4)
            SEQ_NO=$((SEQ_NO + 1))
            LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                    # seq x10
            create_LV $LV $VG /tmp/map_small_data1 75 udb
            untag_container /dev/r$LV
            chown -R $INSTANCE.$GROUP /dev/r$LV
#           chmod u+x /dev/r$LV

            SEQ_NO=$((SEQ_NO + 1))
            LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                    # seq x11
            create_LV $LV $VG /tmp/map_small_data2 75 udb
```

```
          untag_container /dev/r$LV
          chown -R $INSTANCE.$GROUP /dev/r$LV
#         chmod u+x /dev/r$LV
          #
          #allocate small_TS_index (1 each on disk 1 & 2)
          SEQ_NO=$((SEQ_NO + 1))
          LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                    # seq x12
          create_LV $LV $VG /tmp/map_small_index1 75 udb
          untag_container /dev/r$LV
          chown -R $INSTANCE.$GROUP /dev/r$LV
#         chmod u+x /dev/r$LV

          SEQ_NO=$((SEQ_NO + 1))
          LV=lv_$(echo $VG|cut -c4-9)_$SEQ_NO                    # seq x13
          create_LV $LV $VG /tmp/map_small_index2 75 udb
          untag_container /dev/r$LV
          chown -R $INSTANCE.$GROUP /dev/r$LV
#         chmod u+x /dev/r$LV
      fi
done

print "$(basename $0):  Synchronizing mirrors (syncvg -l <lvname>)...."
for lv in $LVS_TO_SYNC; do
      nohup syncvg -l $lv >/dev/null 2>&1 &
done


echo "\n${Msg_Prefix}:  VG/LV setup completed on $(date)."\
       "\n\tSee file ${SETUP_LOG} for details.\n"
} 2>&1 |tee -a ${SETUP_LOG}

exit

# LV_name        used for                          DB partitions
# -------------  -------------------------------   -----------------
# lv_nyy_0z_log  jfslog                            1-16
# lv_n01_01_101  catalog TS (raw DMS)              1
# lv_n01_01_102  single NG tablespsace             1
# lv_nyy_0z_x03  UDB logfiles (jfs/SMS)            1-16
# lv_nyy_0z_x04  UDB temp1 (jfs/SMS/2 drives)      1-16
# lv_nyy_0z_x05  UDB temp2 (jfs/SMS/2 drives)      1-16
# lv_nyy_0z_x06  UDB large data1 (raw/DMS)          2-16
# lv_nyy_0z_x07  UDB large data2 (raw/DMS)          2-16
# lv_nyy_0z_x08  UDB large index1 (raw/DMS)         2-16
# lv_nyy_0z_x09  UDB large index2 (raw/DMS)         2-16
# lv_nyy_0z_x10  UDB small data1 (raw/DMS)          2-16
# lv_nyy_0z_x11  UDB small data2 (raw/DMS)          2-16
# lv_nyy_0z_x12  UDB small index1 (raw/DMS)         2-16
# lv_nyy_0z_x13  UDB small index2 (raw/DMS)         2-16
# lv_nyy_0z_114  home/tp3an01 (jfs)                1
#
#  where x=1 for partitions    5,  9, and 13
#        x=2 for partitions 2, 6, 10, and 14
#        x=3 for partitions 3, 7, 11, and 15
```

```
#        x=4 for partitions 4, 8, 12, and 16
#
#    and yy=2-digit node number (01, 05, 09, 13)
#
#    and z=1 for partitions 1, 2, 5, 6,  9, 10, 13, 14
#        z=2 for partitions 3, 4, 7, 8, 11, 12, 15, 16
# ------------------------------------------------------------------
#  All SP Nodes except node01, have 4 database partitions each.  Large
#  tables are split across 15 database partitions:  3 partitions on
#  node01 and 4 partitions on each of nodes 05, 09, and 13.
#
#  SP node 01, in addition to supporting 3 database partitions, also
#  provides the following functions:
#        1. NFS server for user home directories
#        2. ADSM server
#        3. Logical partition 1 contains catalog tablespace and single
#           nodegroup tables (nation and region)
#        4. Coordinator node functions
# ------------------------------------------------------------------
```

The following script, setup_VG_maps.ksh, is called by setup_VG.ksh:

```
#!/usr/bin/ksh
# ------------------------------------------------------------------------
# name:  setup_VG_maps.ksh
#
#        This script sets up mapfiles for logical volume allocation.  See
#        also "setup_VG.ksh" whick calls this script.
# ------------------------------------------------------------------------

/bin/rm -f /tmp/map_*

# ----------------
# Disk #1
# ----------------

cat >> /tmp/map_small_index1 <<-EOF!
        $1:1-75
EOF!

cat >> /tmp/map_large_data1 <<-EOF!
        $1:76-225
EOF!

typeset -i i=226
while [[ $i -le 287 ]]; do
  cat >> /tmp/map_temp1 <<-EOF!
        $1:$i
        $2:$i
        $3:$i
        $4:$i
EOF!
  i=i+1
done
```

```
typeset -i i=288
while [[ $i -le 349 ]]; do
  cat >> /tmp/map_temp2 <<-EOF!
        $1:$i
        $2:$i
        $3:$i
        $4:$i
EOF!
  i=i+1
done

cat >> /tmp/map_jfslog <<-EOF!          # Use for both log and mirror
        $1:350
EOF!

typeset -i i=351
typeset -i j=350
while [[ $i -le 425 ]] && [[ $j -le 424 ]]; do
  cat >> /tmp/map_log1 <<-EOF!
        $1:$i
        $2:$j
EOF!
  i=i+1;j=j+1
done

# ----------------
# Disk #2
# ----------------

cat >> /tmp/map_small_index2 <<-EOF!
        $2:1-75
EOF!

cat >> /tmp/map_large_data2 <<-EOF!
        $2:76-225
EOF!

#       $2:226-349      #This temp space is allocated under disk1

#       $2:350-424      #This log space is allocated under disk1

# ----------------
# Disk #3
# ----------------

cat >> /tmp/map_small_data1 <<-EOF!
        $3:1-75
EOF!

cat >> /tmp/map_large_index1 <<-EOF!
        $3:76-225
EOF!
```

```
typeset -i i=350
while [[ $i -le 424 ]]; do
  cat >> /tmp/map_log1_mirror <<-EOF!
        $3:$i
        $4:$i
EOF!
  i=i+1
done

# ----------------
# Disk #4
# ----------------

cat >> /tmp/map_small_data2 <<-EOF!
        $4:1-75
EOF!

cat >> /tmp/map_large_index2 <<-EOF!
        $4:76-225
EOF!

exit
```

The following script setup_VG_maps_catalog.ksh, is called by setup_VG.ksh:

```
#!/usr/bin/ksh
# ------------------------------------------------------------------------
# name:  setup_VG_maps_catalog.ksh
#
#        This script sets up mapfiles for logical volume allocation.  See
#        also "setup_VG.ksh" which calls this script.
# ------------------------------------------------------------------------

/bin/rm -f /tmp/map_*

# ----------------
# Disk #1
# ----------------

cat >> /tmp/map_catalog <<-EOF!
        $1:200-224
EOF!

typeset -i i=225
while [[ $i -le 286 ]]; do
  cat >> /tmp/map_temp1 <<-EOF!
        $1:$i
        $2:$i
        $3:$i
        $4:$i
EOF!
i=i+1
done
```

```
typeset -i i=287
while [[ $i -le 348 ]]; do
  cat >> /tmp/map_temp2 <<-EOF!
        $1:$i
        $2:$i
        $3:$i
        $4:$i
EOF!
  i=i+1
done

  cat >> /tmp/map_jfslog <<-EOF!          # Use for both log and mirror
        $1:349
EOF!

typeset -i i=350
typeset -i j=349
while [[ $i -le 424 ]] && [[ $j -le 423 ]]; do
  cat >> /tmp/map_log1 <<-EOF!
        $1:$i
        $2:$j
EOF!
  i=i+1;j=j+1
done

typeset -i i=425
typeset -i j=424
while [[ $i -le 474 ]] && [[ $j -le 473 ]]; do
  cat >> /tmp/map_home <<-EOF!
        $1:$i
        $2:$j
EOF!
  i=i+1;j=j+1
done
# ----------------
# Disk #2
# ----------------
cat >> /tmp/map_single1 <<-EOF!
        $2:200-224
EOF!
  # ----------------
  # Disk #3
  # ----------------
cat >> /tmp/map_catalog_mirror <<-EOF!
$3:200-224
EOF!

typeset -i i=349
while [[ $i -le 423 ]]; do
    cat >> /tmp/map_log1_mirror <<-EOF!
        $3:$i
        $4:$i
EOF!
```

```
    i=i+1
done

typeset -i i=424
while [[ $i -le 473 ]]; do
  cat >> /tmp/map_home_mirror <<-EOF!
        $3:$i
        $4:$i
EOF!
  i=i+1;j=j+1
done
# ----------------
# Disk #4
# ----------------
cat >> /tmp/map_single1_mirror <<-EOF!
        $4:200-224
EOF!

exit
```

4.11.3.4 Map disks to volume groups

This file is called disk_table.ksh. It is used by the setup_VG.ksh script listed above.

```
export vg_n01_01="hdisk13 hdisk23 hdisk29 hdisk14  \
                  hdisk64 hdisk53 hdisk39 hdisk57"
export vg_n01_02="hdisk10 hdisk12 hdisk5  hdisk4   \
                  hdisk36 hdisk42 hdisk38 hdisk61"
export vg_n05_01="hdisk17 hdisk19 hdisk24 hdisk20  \
                  hdisk40 hdisk45 hdisk37 hdisk63"
export vg_n05_02="hdisk30 hdisk31 hdisk33 hdisk32  \
                  hdisk54 hdisk60 hdisk44 hdisk46"


export vg_n09_01="hdisk22 hdisk7  hdisk18 hdisk16  \
                  hdisk56 hdisk62 hdisk52 hdisk59"
export vg_n09_02="hdisk2  hdisk3  hdisk27 hdisk6   \
                  hdisk35 hdisk50 hdisk48 hdisk41"
export vg_n13_01="hdisk21 hdisk25 hdisk15 hdisk8   \
                  hdisk58 hdisk47 hdisk51 hdisk43"
export vg_n13_02="hdisk26 hdisk28 hdisk9  hdisk11  \
                  hdisk49 hdisk65 hdisk55 hdisk34"
```

4.11.4 Synchronize Volume Groups

We used this script, vg_mach.ksh, to map volume groups to a new (available) major number in the SP nodes. It also handles DB2 raw device permissions, detects NFS conflicts and so on.

We use `lvlstmajor` to detect a free major number in both owner and takeover node. We only use the actual VG major number of owner node if this is free or used by the same VG on the takeover node.

This script, vg_mach.ksh, must run in the node that owns the volume group.

```
#-------------------------------------------------------------------------#
# File:         vg_mach.ksh
# Version:      1.1.0
#
# Description:  This script updates the definitions of volumes groups in
#               takeover node.
#
# Syntax:       # vg_mach.ksh vol_grp tko_node major_num
#               where 'vol_grp' is the name of the volume group
#               to be updated; 'tko_node' is the ip name of the
#               takeover node
#               (you need rsh access to it) and 'major_num' is a major
#               number not used in all the nodes in the cluster.
#
# Example:      vg_mach.ksh vg_n01_01 hnode01 34
#
# Fix parameter values
ProgramName=$(basename $0)
VGN=${1:-"vol_name_miss"}
DMVGN=`ls -l /dev/$VGN | awk '{print $5;}' | awk -F, '{print $1;}'`
TKONODE=${2:-"takeover_miss"}
MVGN=${3:-$DMVGN}
echo "actual major number $DMVGN "

#************************************************
#                                               *
#    U P D A T E     T H I S   V A R I A B L E S    *
DB2ADM="db2inst1"
DB2GRP="dbadmin1"
#                                               *
#************************************************

#temp files
TMPFILE="/tmp/$ProgramName.info"
TMPFILE2="/tmp/$ProgramName.disks"
TMPFILE3="/tmp/$ProgramName.nfs"
TMPFILE4="/tmp/$ProgramName.confnfs"
NFSLOG="$ProgramName.$VGN.nfs.conf"
rm /tmp/$ProgramName.*

echo
"*************************************************************************"
echo "Are you sure to run update for volume group $VGN in local node"
echo "and node $TKONODE using major number $MVGN ?"
echo "DB2 LVs (type UDB) will be set to owner $DB2ADM.$DB2GRP."
echo
"*************************************************************************"
```

```
echo "y or n =>"
read yesno
if [ "$yesno" != "y" ]; then
    exit 99
fi

#get VG info.
lsvg -l $VGN | tail -n +3 >$TMPFILE
lsvg -p $VGN | tail -n +3 >$TMPFILE2
#filesystems
FS=‘awk ’{ if ( $7 != "N/A" ) print $7;}’ $TMPFILE‘
#raw devices that have type as UDB
UDB=‘awk ’{ if ( $2 != "UDB" ) print $1;}’ $TMPFILE‘
DISKS=‘awk ’{ print $1;}’ $TMPFILE2‘
FIRSTDISK=‘echo $DISKS | awk ’{ print $1;}’‘
IDOWN=‘lsattr -El $FIRSTDISK|grep pvid|awk ’{print $2;}’‘
IDTKO=‘rsh $TKONODE lsattr -El $FIRSTDISK|grep pvid|awk ’{print $2;}’‘
if [ $IDOWN != $IDTKO ]; then
        print "\\n[$ProgramName]: $DISK have different disk id on owner " \
         "                an takeover node $TKONODE.\\n"
         exit 99
fi
# got NFS definitions?
rsh $TKONODE lsnfsmnt | tail -n +1 >$TMPFILE3
touch $TMPFILE4
# Mounted filesystems
MOUNTS=‘df| tail -n +2|awk ’{print $7 " ";}’‘
# Automount filesystems
STRMOUNTS=‘lsfs| tail -n +2|awk ’{if($7== "yes" ) print $3 " ";}’‘

# umounting filesystems and setting to no mount in start
for i in $FS
do
   FMNT=‘echo $MOUNTS | grep $i‘
   if [ "$FMNT" != "" ]; then
      echo "Umount $i."
      umount $i
   fi
   FSTR=‘ echo $STRMOUNTS | grep $i‘
   if [ "$FSTR" != "" ]; then
      echo "Automount set to no for $i."
      chfs -A’’‘locale nostr | awk -F: ’{print $1}’’’’ $i
   fi
   #check NFS conflicts?
   if grep $i $TMPFILE3
   then
      grep $i $TMPFILE3 >>$TMPFILE4
      echo "ERROR: NFS conflict in $i filesystem"
      echo "       storing definition in $NFSLOG."
      cp $TMPFILE4 $NFSLOG
      echo "       Removing NFS definition in $TKONODE."
      echo "       Use HACMP to do NFS mount in $TKONODE."
      rsh $TKONODE /usr/sbin/rmnfsmnt -f $i -B
   fi
```

```
done

echo Free VG
varyoffvg $VGN
if rsh $TKONODE lsvg | grep $VGN
then
   echo "Attention: A VG called $VGN detected in $TKONODE."
   echo "           The existing volume group is exported now."
   rsh $TKONODE exportvg $VGN
fi
echo Get new volume group in correct major number WAIT
echo Play attention to name conflicts here
rsh $TKONODE importvg -V $MVGN -y $VGN $FIRSTDISK
echo Put online $VGN on $TKONODE
rsh $TKONODE varyonvg $VGN
echo Set varyon to manual
rsh $TKONODE chvg -a n -Q y $VGN
# permisions of UDB raw devices
echo "set UDB permisions to correct values in $TKONODE"
for i in $UDB do
    chown $DB2ADM.$DB2GRP /dev/r$i
done
echo Set VG off
rsh $TKONODE varyoffvg $VGN
if ! ls -l /dev/$VGN | awk '{print $5;}' | grep "$MVGN" then
    echo Local definition of VG gone
    exportvg $VGN
    echo Adjust local major number
    importvg -V $MVGN -y $VGN $FIRSTDISK
fi
echo Put online $VG locally
varyonvg $VGN
# permisions of UDB raw devices
echo "set UDB permisions to correct values here"
for i in $UDB do
    chown $DB2ADM.$DB2GRP /dev/r$i
done
echo Set varyon to manual
chvg -a n -Q y $VGN
```

Note: This script can delete and re-create volume groups across SP nodes; so use it with care as a volume group corruption can result in loss of the data in the volume group. We assume there are no duplicate volume group names, logical volume names or file system names in the cluster. If this is not the case, conflicts may arise and must be manually resolved. Logical volumes of class UDB will be set with permissions for DB2. The `DB2ADM` and `DB2GRP` variables in the script are the DB2 owner and group of the instance administrator. The script must be running in the node that owns the volume group.

Appendix A. Autoloader Examples

This appendix provides some examples of using the Autoloader tool to load data into a DB2 UDB EEE database. The information given here is designed to complement the two sections on Autoloader in the *DB2 UDB Administration Guide,* S10J-8157. These sections are in Chapter 5, Using the Autoloader Utility, and Appendix O, Supplemental Autoloader Information.

A.1 Autoloader Example 1: Split on 1 DP, Load on 15 DPs

In this example, we use the Autoloader tool to:

- Receive input from a file produced by the TPCD data generator (dbgen)
- Split and Load in one run
- Split on 1 database partition only
- Load on 15 database partitions

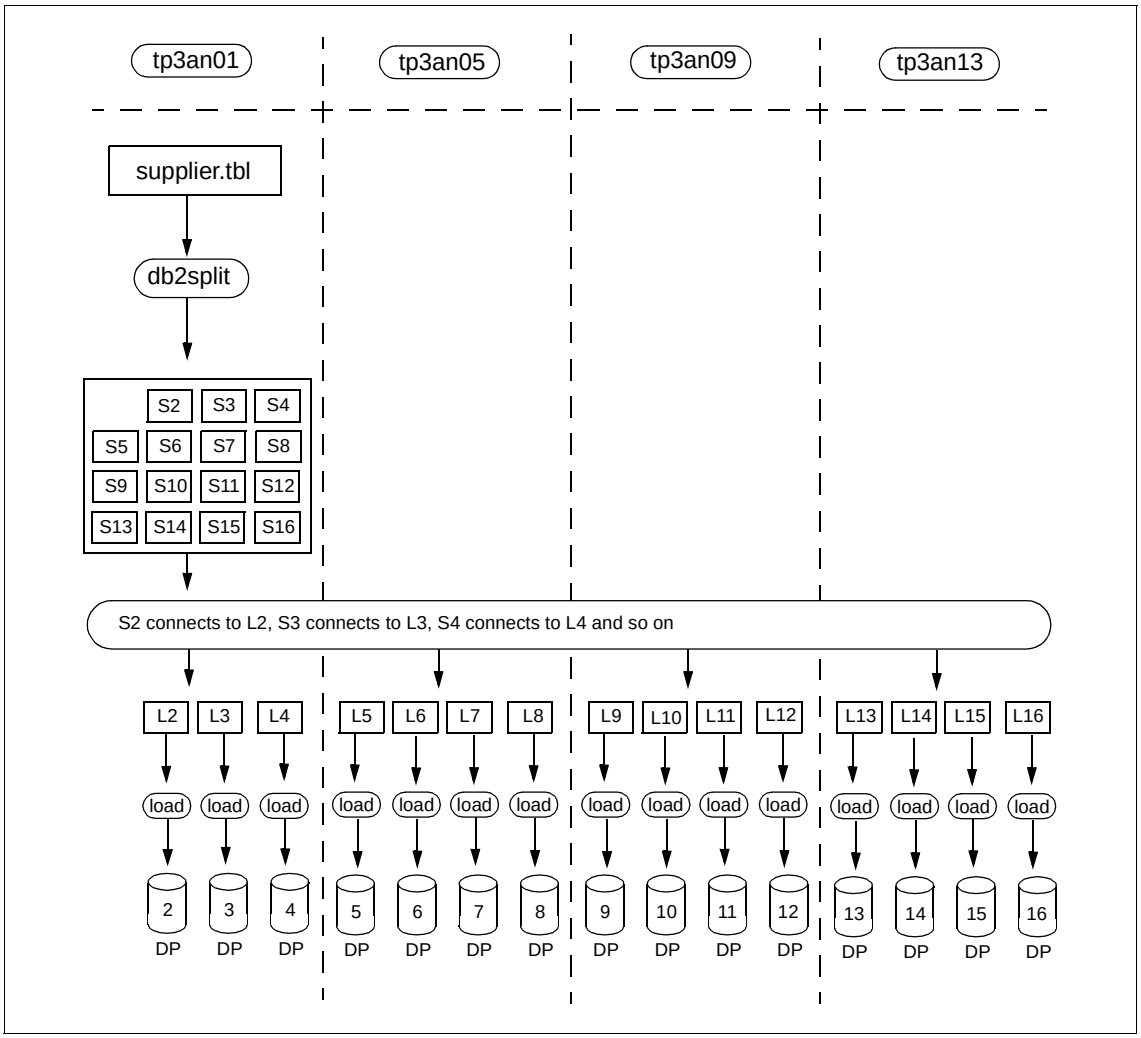


Figure 27. Autoloader Example 1

In Figure 27 on page 244, an overview of the Autoloader processing is shown. Note that:

- The supplier table has been defined in a 15 database partition nodegroup.
- There are four database partitions per SP node.
- The four SP nodes are tp3an01, tp3an05, tp3an09, and tp3an13.
- Database partition 1 is reserved for the System Catalogs.

- S2,S3,S4 ... S16 represent the 15 output named pipes of the split process. These pipes are named:
  - supplier.tbl2.00n (where n = 2,3,4 ... 16) and are stored in the /work/db2inst1/psplitload directory.
  - All these pipes exist on tp3an01.
- L2,L3,L4 ...L16 represent the 15 input named pipes of the 15 load processes. These pipes are named:
  - supplier.tbl.00n (where n = 2,3,4 ... 16) and are stored in the /work/db2inst1/psplittemp directory.
  - The pipes supplier.tbl.002,003,004 exist on tp3an01.
  - The pipes supplier.tbl.005,006,007,008 exist on tp3an05.
  - The pipes supplier.tbl.009,010,011,012 exist on tp3an09.
  - The pipes supplier.tbl.013,014,015,016 exist on tp3an13.

A.1.1 TPCD Generator

First, dbgen is run to create the input data:

```
dbgen -Ts -v -s1
```

dbgen is run with these flags:

`-Ts`	Only produce data for the supplier table
`-v`	Verbose
`-s1`	Produce supplier data for a 1 GB TPCD database

This will produce this output file:

```
-rw-r--r--   1 db2inst1 sys      1409633 Feb 27 11:53 supplier.tbl
```

Note: For clarity, we will run the dbgen into a file, and then db2autold will use that file as input. We could also use a named pipe to avoid having to store the supplier.tbl file on disk.

4.11.5 Autoloader Configuration File

This is the Autoloader configuration file, supplier.cfg:

```
RELEASE=V5.0

db2 "load from supplier.tbl of del modified by coldel| \
    replace into supplier statistics yes and indexes all using /work"

DATABASE=tpcd30

OUTPUT_NODES=(2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)

SPLIT_NODES=(2)

MODE=SPLIT_AND_LOAD

LOGFILE=LOG

NOTNFS_DIR=/work

CHECK_LEVEL=NOCHECK

TRACE=1
```

We must also create a non-NFS directory on each SP node. This directory is used by Autoloader to store the named piped it uses to channel to output of the splitting processes into the loading processes. In the supplier.cfg file, we have specified NOTNFS_DIR=/work.

The supplier table and all its indexes has been created prior to running db2autold; so during the load processing, all the indexes for supplier will be built.

The load command will:

- Generate statistics during the load (statistics yes and indexes all). Since we have not specified a value for `RUN_STAT_NODE` in the configuration file, statistics will be generated using the data at database partition 2, the first in the list of OUTPUT_NODES. In a EEE database, statistics are only ever generated from the data at a single database partition.
- By specifying `using /work` in the `load` command, we will use /work (a local non-NFS directory) for sort space when creating indexes. If we don't specify a non-NFS directory here, the default is $INSTHOME/sqllib/tmp, which is shared across all database partitions. A local non-nfs directory should be used in place of the default directory as:
  - The sort space will be managed locally on each SP node so all writes will be to local disk (not writes across NFS). This all means the performance will be much improved because there will be less contention for disk.

- $INSTHOME will fill up very quickly as all the load commands (15) will use this one directory for sort space.

A.1.2 Autoloader Command

When we run the `db2autold` command we must run it from a directory which is shared across all the SP nodes in the instance. We created a subdirectory in the instance owner's home directory, (in our case /home/tp3an01/db2inst1) as this directory must be shared for EEE to function.

This is the command which drives Autoloader:

```
db2autold -c supplier.cfg
```

A.1.3 Autoloader Output

When we run this command, this is the output:

```
Start reading autoloader configuration file: supplier.cfg.
Start initializing autoloader process.
Start moving data from db2split process(es) to target loading nodes in
background.
Start loading data on "15" node(s) in background.
Start 1 db2split process(es) in background.
Autoloader completed with detailed log message in file "autoload.LOG".
Cleanup was done!
```

A.1.4 The Phases of Autoloader Processing

Lets consider each phase in detail:

1. During the phase called "Start initializing autoloader process":
   - The values of NOTNFS_DIR (/work) and the username (db2inst1) are used to make the path /work/db2inst1.
   - In this path, directories called psplitload and psplitemp are created (if not already created).
     - psplitload is used to store the named pipes used as the output of the split processing.
     - psplittemp is used to store the named pipes used as the input for the load processing.
   - This happens on all SP nodes referenced in the db2nodes.cfg file for this instance.

2. During the phase called "Start moving data from db2split process(es) to target loading nodes in background":
   - The named pipes used for the output of the split processing are connected to the named pipes used for load processing. These latter pipes exist on the SP nodes where the data will be loaded for each of the split output files.
3. During the phase called "Start loading data on "15" node(s) in background."
   - The load processes are started (15 in this case). Each load process:
     - Takes its input from a named pipe and is started in the background.
     - Is run on the SP node where its data belongs
4. During the phase called "Start 1 db2split process(es) in background."
   - The one split process is started. This split process:
     - Takes its input from the file supplier.tbl
     - Outputs to 15 named pipes
5. The message `Cleanup was done` indicates that:
   - All the named pipes are removed.
   - All the processes associated with Autoloader are terminated.
   - All the temporary message files used by Autoloader are removed.

Note that the 15 load processes are started in the background; then the single split process is started. For split and load to communicate using pipes, the reader of the pipe (load) must be started before the writer (split) starts.

A.1.5 Log Files

After cleanup has finished, these log files are produced:

- autoload.LOG:
  - This is the log file of the complete autoloader processing.
- Load_LOG.<dp>, where dp = 2,3,4 16
  - These files, one per value in OUTPUT_NODES, are the messages from each load command.
  - In each file, we can see the named pipe used as input. For example, for database partition 2, the named pipe is /work/db2inst1/psplittemp/supplier.tbl.002

- We can also see the time when the load started and finished, plus the number of rows loaded.

- splt_LOG.2
  - This file contains the messages from the single split process used in this example.
  - In this file, we can see the output data file defined as: /work/db2inst1/psplitload/supplier.tbl2. This means that 15 named pipes called supplier.tbl2.002, supplier.tbl2.003 ... supplier.tbl2.016 in the /work/db2inst1/psplitload directory will be used as the output of the split process. The "2" after "tbl" indicates that split was run on database partition 2.
  - We can also see the time when the split started and finished, plus the number of processed input lines.

A.1.6 In Case of Problems

If any problems occur during Autoloader processing, cleanup may not take place. In this case, you should rerun `db2autold` with the same configuration file and add the `-d` flag. For example:

```
db2autold -c supplier.cfg -d
```

If you run Autoloader from a non-NFS directory, you will see output similar to the following:

```
ksh: /work/frame:  not found.
ksh: ./loadscript_supplier.tbl_5:  not found.
```

This messages indicate that the load script for database partition 5 cannot be found. This is because the load process for partition 5 is run on tp3an05. But as the autoloader job was started from a non-nfs directory on tp3an01, this directory is not available on tp3an05.

A.2 Autoloader Example 2: Split on 4 DPs, Load on 15

In this example, we use the Autoloader tool to:

- Split and Load in one run
- Split on four database partitions (compared to 1 in Example 1)
- Load on 15 database partitions

Running split on multiple database partitions will result in:

- The input data being converted into a number of smaller pieces (in this case four). This conversion is not done using db2split but by db2psplit, which cuts the input data into four equally sized pieces irrespective of the data.
- Each of these data pieces will be split using db2split. In this case, there are 15 outputs from each db2split, making a total of 60 outputs.
- Each of these outputs will be piped to a load running on the correct partition for the data.

We chose to split on four database partitions and specified the partitions 1, 2, 3, and 4 since these partitions are all located on the first SP node, tp3an01. If the "split partitions" were located on other SP nodes, more data would have to be sent between the SP nodes.

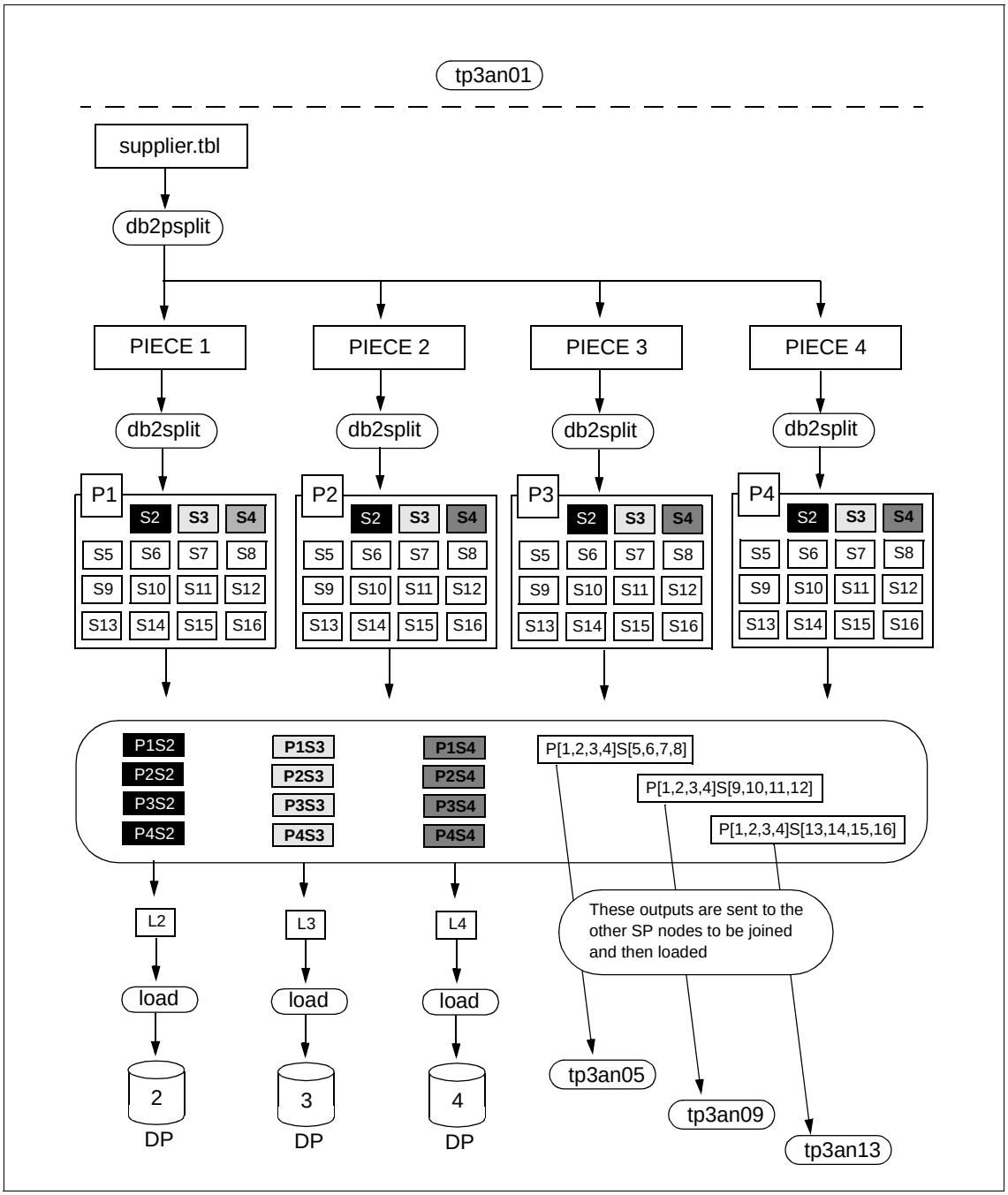


Figure 28. Autoloader Example 2

Note that, in Figure 28:

- Only tp3an01, the first SP node, is shown in this diagram.
- The notation P1S2 refers to the split output for database partition 2 running against piece 1 (P1) of the input data, supplier.tbl.
- The named pipes which receive the output of the split processes are named supplier.tbl<P>.<dp>, where <P> is the number of the piece (ranges from 1 to 4), and <dp> is the partition where this output data belongs (ranges from 001 to 015). In this example, there are 60 such named pipes.

This is the Autoloader configuration file, supplier.cfg:

```
RELEASE=V5.0

db2 "load from supplier.tbl of del modified by coldel| \
    replace into supplier statistics yes and indexes all using /work"

DATABASE=tpcd30

OUTPUT_NODES=(2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)

SPLIT_NODES=(1,2,3,4)

MODE=SPLIT_AND_LOAD

LOGFILE=LOG

NOTNFS_DIR=/work

CHECK_LEVEL=NOCHECK

TRACE=1
```

Note that:

- Compared to Example 1, the SPLIT_NODES clause has changed. Now it is: SPLIT_NODES=(1,2,3,4).
- The split log files, splt_LOG.1, splt_LOG.2, splt_LOG.3 and splt_LOG.4 show the output from each split.

A.2.1 Autoloader Command

This is the command that drives Autoloader:

```
db2autold -c supplier.cfg
```

A.2.2 Autoloader Output

When we ran this command, this is the output:

```
Start reading autoloader configuration file: supplier.cfg.
Start initializing autoloader process.
Start moving data from db2split process(es) to target loading nodes in
background.
Start loading data on "15" node(s) in background.
Start 4 db2split process(es) in background.
Autoloader completed with detailed log message in file "autoload.LOG".
Cleanup was done!
```

Note that now four db2split processes were used.

A.2.3 How Many db2split Processes to Use?

Now that we can use multiple split processes against the input data, how can we decide how many to use? The answer to this question depends really on where the slowest part of the complete data loading process is. If it is the split part, then using multiple split processes may well improve the performance of the overall loading cycle.

Bear in mind that:

- The more split processes used:
  - The more data has to pass between the SP nodes.
  - The greater the overhead to create and manage the named pipes
  - The greater the number of processes running simultaneously
  - The smaller the number of lines each split has to process. For example, if 16 split processes are run simultaneously, each one only has to process 6.25 percent of the input data.
- The faster the speed at which data is input into Autoloader the greater the benefit of having multiple split processes, such as with a very fast mainframe connection using ESCON channels.
- Conversely, if the input into Autoloader is the bottleneck, there is no point running more split processes, such as when running a data generator like dbgen when using SP high nodes. The dbgen program is single threaded and cannot exploit the multiple CPUs.

Appendix B. Running Commands on Multiple Database Partitions

In an environment with a lot of database partitions over several SP nodes, the job of running commands on multiple partitions poses some performance problems. With 16 DPs to cover, running a simple command like `db2 update db cfg` can take a relatively long time. For example, we recorded these times:

1. db2_all with no special modifiers

   ```
   db2_all "db2 update db cfg for TPCD30 using logretain on
   ```

   This takes around 43 seconds.

2. db2_all using ||

   ```
   db2_all "|| db2 update db cfg for TPCD30 using logretain on
   ```

   This takes around 23 seconds.

3. db2_all using ;

   ```
   db2_all "; db2 update db cfg for TPCD30 using logretain on"
   ```

   This takes around 51 seconds.

B.1 Creating Tailored Scripts

To improve the performance of tasks like this, we wrote some scripts which were tailored to our environment. These scripts reduce the number of remote shells invoked and run as much function as possible at each SP node where a particular DP resides. The following pages provide example scripts that allow us to run `db2 update db cfg` in 3 to 4 seconds. This is an overview of the function:

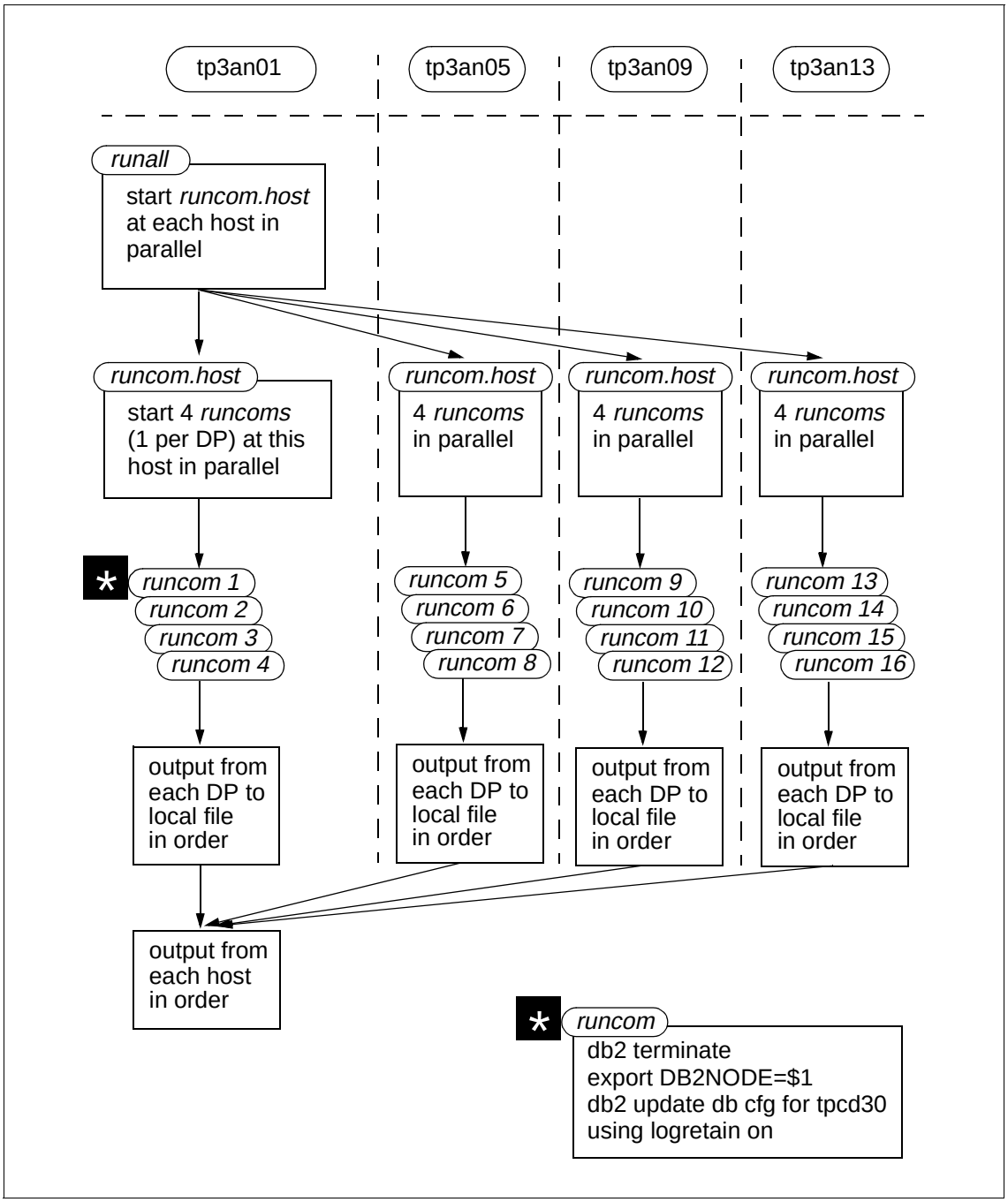


Figure 29. Using Tailored Scripts to Run Commands in Parallel

Figure 30.

Here is the ~/bin/runall script:

```
wd=‘pwd‘
c=$@      #the command

x=$USER
t=/tmp/$USER
cd $t

echo "Running $c on 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16"
rsh tp3an01 ". ~/.profile;cd $t;\
 ~/bin/runcom.host 1 2 3 4 $c" 2>&1 > $x.h1 &
rsh tp3an05 ". ~/.profile;cd $t;\
 ~/bin/runcom.host 5 6 7 8 $c" 2>&1 > $x.h5 &
rsh tp3an09 ". ~/.profile;cd $t;\
 ~/bin/runcom.host 9 10 11 12 $c" 2>&1 > $x.h9 &
rsh tp3an13 ". ~/.profile;cd $t;\
 ~/bin/runcom.host 13 14 15 16 $c" 2>&1 > $x.h13 &

wait

cat $x.h1 $x.h5 $x.h9 $x.h13 > $x.all
cat $x.all

cd $wd
```

Note that:

- Like `db2_all`, `/tmp/$USER` is used to store output
- A `wait` command is used. This means that execution of the command after the wait will only take place after all the child and lower generations of processes have terminated.
- The output from the 16 DPs will be output to the terminal in the right order.

Here is the ~/bin/runcom.host script:

```
n1=$1;n2=$2;n3=$3;n4=$4

shift;shift;shift;shift

c=$@     #the command

x=$USER     #get a id

for i in $n1 $n2 $n3 $n4
do
~/bin/runcom $i $c 2>&1 > $x.$i &
done

wait

cat $x.$n1 $x.$n2 $x.$n3 $x.$n4 > $x.$n1$n2$n3$n4

cat $x.$n1$n2$n3$n4
```

Here is the ~/bin/runcom script:

```
n=$1
shift
c=$@    # the command

h=‘hostname | cut -d’.’ -f1‘

db2 terminate 2>&1 >/dev/null
export DB2NODE=$n && echo "$h:DB2NODE=$DB2NODE"
$c  2>&1
```

Some important points about these scripts:

- They are all placed in a directory accessible from all SP nodes (in this case ~/bin, which has been added to the PATH).
- It is much more efficient to run four `rsh` commands which themselves initiate other local processes at each host, than to run 16 `rsh` commands.
- These scripts are very much tailored to our particular environment. If the hostnames or number of DPs change then the scripts would have to be modified. They are really designed to be examples of how to run commands at all the DPs efficiently.
- The reason that *runcom.host* calls *runcom* to run the DB2 command itself is that:
  - If you try to run DB2 commands in parallel in the background from the same script and each DB2 command addresses a different DP, for example:

```
db2 terminate;export DB2NODE=1;db2 "list tablespaces" &
db2 terminate;export DB2NODE=2;db2 "list tablespaces" &
```

The following error message is often generated:

```
DB21016E The command line processor encountered a system
          error while sending the command to the back-end
          process.
```

This problem goes away if we instead call another script to execute the DB2 command.

Now if we try:

```
runall "db2 update db cfg for TPCD30 using logretain on"
```

This takes around three seconds and produces this output:

```
Running db2 update db cfg for tpcd30 using logretain on on 1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16
tp3an01:DB2NODE=1
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  All applications must disconnect from this database before the
changes become effective.

tp3an01:DB2NODE=2
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  All applications must disconnect from this database before the
changes become effective.

.............(OMITTED OUTPUT FROM DPs 3-14)............................

tp3an13:DB2NODE=15
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  All applications must disconnect from this database before the
changes become effective.

tp3an13:DB2NODE=16
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  All applications must disconnect from this database before the
changes become effective.


real 3.80
user 0.04
sys  0.44
```

Note that the random script can be used by itself to run a DB2 command at a single partition. For example, to check the number of log pages written on DP 2 using `db2 get snapshot`:

```
runcom 2 db2 get snapshot for all on tpcd30|grep 'Logs pages w'
```

Running tasks which themselves are time-consuming (for example a load across all DPs of a large database) will not gain much advantage from using these tailored scripts. The scripts are really intended for commands that are quick to execute and must be run at the DP level, such as:

- `db2 list application`
- `db2 force application all`
- `db2 get snapshot`
- `db2 list tablespaces`

B.2 Performing Backups

Once these scripts have been written, it is relatively easy to make similar scripts to perform specialized functions such as backing up a small database to disk. Here are the scripts we wrote to perform this function:

Here is the script ~/bin/backdball:

```
id=$1   # id for this backup

wd=‘pwd‘

x=$USER
t=/tmp/$USER
cd $t

# as its a backup, we need to finish DP 1 first !
echo "Running backup db using id $id on 1"

db2 terminate 2>&1 >/dev/null
export DB2NODE=1
mkdir /backdb/NODE0001/$id 2>&1 >/dev/null
db2 -v "backup db tpcd30 to /backdb/NODE0001/$id"

# now the rest in parallel

echo "Running backup db using id $id on 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16"
rsh tp3an01 ". ~/.profile;cd $t;\
 ~/bin/backdb.host $id 1 2 3 4" 2>&1 > $x.h1 &
rsh tp3an05 ". ~/.profile;cd $t;\
 ~/bin/backdb.host $id 5 6 7 8" 2>&1 > $x.h5 &
rsh tp3an09 ". ~/.profile;cd $t;\
 ~/bin/backdb.host $id 9 10 11 12" 2>&1 > $x.h9 &
rsh tp3an13 ". ~/.profile;cd $t;\
 ~/bin/backdb.host $id 13 14 15 16" 2>&1 > $x.h13 &

wait

cat $x.h1 $x.h5 $x.h9 $x.h13 > $x.all
cat $x.all

cd $wd
```

Note that:

- The catalog DP must finish its backup before the other DPs start their backups.
- The backup images are stored in /backdb/NODE00nn, where nn is the DP number.
- This script was used to back up the database just after `logretain` was set to `on` (which puts the table spaces in the database into Backup Pending status). So there is very little data in the database and not a lot of disk space will be used by the backup images.

Here is the ~/bin/backdb.host script:

```
id=$1
shift
n1=$1;n2=$2;n3=$3;n4=$4

x=$USER     #get a id

#as its backup, don't do it on 1, but empty 1's output file
> $x.1

for i in $n1 $n2 $n3 $n4
do
[[ $i -ne 1 ]] && ~/bin/backdb $i $id 2>&1 > $x.$i &
done

wait

cat $x.$n1 $x.$n2 $x.$n3 $x.$n4 > $x.$n1$n2$n3$n4

cat $x.$n1$n2$n3$n4
```

Note that the output from DP 1 must be cleared; otherwise, we will see the output from a previous task.

Here is the ~/bin/backdb script:

```
n=$1
id=$2

db2 terminate 2>&1 >/dev/null
export DB2NODE=$n && echo "DB2NODE=$DB2NODE"
[[ $n -lt 10 ]] && out=NODE000$n || out=NODE00$n
mkdir /backdb/$out/$id 2>&1 >/dev/null
db2 -v backup db tpcd30 to /backdb/$out/$id 2>&1
```

Note that by storing the backup image from each DP in /backdb/NODE00nn, where nn is the DP number. This script could be used to back up much larger databases to disk. Each /backdb/NODE00nn directory could be a separate filesystem.

To run a backup, the command is:

```
backdball <id>
```

where <id> is an identifier for this backup.

Similar scripts exist for restore. If we need to restore, we would use:

```
restdball <id>
```

B.3 Examples of db2_all and rah

This section lists some useful examples of running commands across database partitions or hosts using the supplied utilities, `db2_all` and `rah`.

This information is designed to supplement Appendix P in the *DB2 UDB Administration Guide*.

B.3.1 Suppress Execution of the User's .profile

By default, the user's profile will be executed before the command. If we use a close parenthesis) before the command, the user's .profile file will not be run before executing the command. For example:

```
rah ")ps -F pid,ppid,etime,args -u $USER"
```

This makes sense in this example because the command to run (ps) does not require any environment variables, such as `$PATH` to be set in order to execute at the other hosts. By not running the .profile file, the `rah` command executes faster.

B.3.2 Display the Number of the DP where the Command was Run

In an configuration with multiple DPs per SP node, if we run a command like `db2_all db2 list tablespaces`, the output is tagged with the hostname, but not the DP number. To display at the number of the DP where the DB2 command was run, we can use the double quotes prefix. For example:

```
db2_all "\" echo DB2NODE=## && db2 list tablespaces | egrep Name\|State"
```

where the double quotes `"` tell `db2_all` to substitute any occurrence of double hash `##` with the DP number. Since we are enclosing the whole command with a pair of double quotes, we must escape the double quotes that `db2_all` treats as special with a backslash \.

The output of this command is:

```
DB2NODE=1
 Name                                = SYSCATSPACE
 State                               = 0x0000
 Name                                = TS_LIT
 State                               = 0x0000
 Name                                = USERSPACE1
 State                               = 0x0000
 Name                                = TS_TMP
 State                               = 0x0000
tp3an01: echo DB2NODE=## && ... completed ok

DB2NODE=2
 Name                                = USERSPACE1
 State                               = 0x0000
 Name                                = TS_TMP
 State                               = 0x0000
 Name                                = TS_DAT_MED
 State                               = 0x0000
 Name                                = TS_IND_MED
 State                               = 0x0000
 Name                                = TS_DAT_BIG
 State                               = 0x0000
 Name                                = TS_IND_BIG
 State                               = 0x0000
tp3an01: echo DB2NODE=## && ... completed ok

DB2NODE=3 and so on
```

Note the difference between the table spaces at DP 1 compared to DP 2.

Another way to use the double quotes special modifier is to protect it using single quotes. Here is an example where `db2_all` is used to stop the Governor utility on each DP:

```
db2_all ';"' db2gov stop $Database nodenum '##'
```

Note that since the semicolon, double quotes and hash (#) are all special to the Korn Shell, they must be protected with single quotes. In this way, they can be passed on to `db2_all` for interpretation.

Appendix C. Special Notices

This publication is intended to help system or database administrators to manage very large databases using DB2 Universal Database Enterprise-Extended Edition. The information in this publication is not intended as the specification of any programming interfaces that are provided by DB2 Universal Database. See the PUBLICATIONS section of the IBM Programming Announcement for DB2 Universal Database Enterprise-Extended Edition for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate

them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

You can reproduce a page in this document as a transparency, if that page has the copyright notice on it. The copyright notice must appear on each page being reproduced.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ADSTAR
AIX
AT
DB2
DB2 Universal Database
HACMP/6000
IBM ®
Magstar
RS/6000
RISC System/6000
SP
Scalable POWERparallel Systems
9076 SP2

The following terms are trademarks of other companies:

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix D. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 271.

- *Migrating to DB2 Universal Database Version 5*, SG24-2006
- *Migrating and Managing Data on RS/6000 SP*, SG24-4658
- *Backup, Recovery and Availability on RS/6000 SP,* SG24-4695

D.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

D.3 Other Publications

These publications are also relevant as further information sources:

- *IBM DB2 Universal Database Administration Guide*, S10J-8157
- *IBM DB2 Universal Database Extended Enterprise Edition for AIX Quick Beginnings*, S72H-9620
- *ADSM for AIX V3.1 Administrator’s Guide, GC35-0320*

- *ADSM Installing the Clients*, SH26-4080
- *HACMP Administration Guide*, SC23-1941

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at `http://www.redbooks.ibm.com/`.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Redbooks Web Site on the World Wide Web**

  `http://w3.itso.ibm.com/`

- **PUBORDER** – to order hardcopies in the United States
- **Tools Disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLCAT REDPRINT
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BokkManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

- **REDBOOKS Category on INEWS**
- **Online** – send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online** Orders – send orders to:

	IBMMAIL	**Internet**
In United States	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** – send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** – send orders to:

United States (toll free)	1-800-445-9269
Canada	1-800-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 408 256 5422 (Outside USA)** – ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

L

M

T

U

V

W

ITSO Redbook Evaluation

Managing VLDB Using DB2 UDB EEE
SG24-5105-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
❒ **Customer** ❒ **Business Partner** ❒ **Independent Software Vendor** ❒ **IBM employee**
❒ **None of the above**

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____

_____

_____

_____

SG24-5105-00
Printed in the U.S.A.