APPLICATION NOTE

V-SERIES DISK SYSTEMS IN THE IBM DB2 ENVIRONMENT

BACKUP AND RECOVERY OF DB2 DATABASES USING STORAGETEK SNAPSHOT SOFTWARE

MARCH 2004

1 MANAGEMENT OVERVIEW

Backup and recovery for large, critical database systems is an important and complex requirement in today's data processing environment. The traditional process of dump and restore has been used effectively for a long time. A major problem with this approach is the long outages with the attendant loss of service during these processes. The StorageTek® V-Series Shared Virtual Array® (SVA™) disk system can reduce or eliminate most of these outages.

This document describes how the virtual architecture of the SVA disk system helps to solve a pressing problem for many database users. To restore a database after an error as quickly as possible, frequent backups of the data are required. Unfortunately, goal and means often contradict themselves. The highest possible availability has been incompatible with long backup runs and how they are executed with classic database utilities.

Below, it is shown how the SVA disk system with its StorageTek SnapShot software functionality helps many ERP (Enterprise Resource Planning) users of databases like SAP/R3. Of the disk subsystems equipped with advanced copying techniques (some type of instant copy solution), the SVA disk system takes a special position. The SVA disk system has the most efficient storage architecture on the market and provides the lowest total cost of ownership (TCO). Furthermore, it offers the framework for duplicating (both replicating and restoring) large amounts of data quickly and easily. This feature can provide exceptional customer service, unprecedented backup and recovery, and strategic advantage. Also, with the SVA disk system, copies can be made so frequently that recovery times are reduced and more predictable with significant savings in time, money and resources.

However, it is not sufficient just to deploy intelligent hardware. Even if the hardware plays an indispensable role, it must be part of an integrated solution that consists of hardware, software and process changes in the enterprise. The advantages of the SVA disk system can only be exploited optimally when a backup and recovery strategy is coordinated correspondingly. This requires software to be an interface between storage hardware and process control to comprehensively automate and administer all tasks.

The successful interaction of the three components is described in the remainder of this document.

2 INTRODUCTION

Backup and recovery scenarios with modern ERP systems like SAP R/3, Siebel and others suffer from two typical problems. First, large amounts of data need to be moved for backups and, if an error occurs, the data must be restored. When using traditional methods this can be very time consuming and can create expensive production outages. The second problem is a classic dilemma. On one hand, data must be backed up as frequently as possible to keep loss of production at a minimum if a failure occurs and a recovery is necessary. On the other hand, every backup is a time intensive process during which the data usually is not available. Since both issues are mutually dependent, it is a classic vicious circle.

A solution in such a situation often requires eliminating one of the problem factors. The architectures of modern disk subsystems create possibilities, which account for the requirements of high availability and security, and can form the backbone for a new backup and recovery strategy. Many subsystems can copy large amounts of data quickly and easily. At least for backup and restoration, this feature significantly reduces the high time expenditure. Large improvements can also be achieved for many other processes where copying large amounts of data is important. It might be worthwhile to consider other opportunities as well.

This document deals with two complex issues. First, in which special way does the SVA disk system offer the best facility for an efficient backup and recovery strategy in an ERP environment? It is shown, that the completely different and unique architecture of this disk subsystem is the basis for a particularly simple and minimum-cost solution. Secondly, which requirements have to be fulfilled by software tools in this area and what advantages do they offer? Are the standard database utilities sufficient or are additional tools required for decisive process improvements? The goal of this document is to present a solution for an efficient backup and recovery strategy that also satisfies the criterion of the highest possible data availability and affordability.

Virtual Concurrent Copy (VCC), an IBM supported facility, is another approach that can be used for improved backup and recovery of DB2 databases. VCC is not addressed in this document as the techniques described below can provide superior performance and functionality for both the backup and restore tasks.

3 THE STORAGETEK SHARED VIRTUAL ARRAY (SVA) DISK SYSTEM

3.1 VIRTUAL ARCHITECTURE

A topmost goal of a virtual architecture is the complete separation of the user view from the hardware in order to use the available resources more efficiently and easily. This approach is also true for operating systems, tape, disk and many other components. For a virtual disk system, logical views of volumes and LUNs must be presented to the operating system. The SVA disk system from StorageTek accomplishes this through a table in which the LUNs or volumes are stored as addressable devices of the desired size and device type. Because they are not mapped on the real storage medium of the disk subsystem (back end), only table space but no physical disk storage is consumed when defining the volumes. Physical space is allocated dynamically and only when data is actually written. The functional volumes, although not really existing, are available to the operating system. With this technique, the goal of a virtual architecture is accomplished and, also, the real purpose: to provide an exceptionally flexible and simple handling of disk storage.

The characteristics of the virtual disk architecture are presented for an IBM DB2 database environment in this document. First, it is required to briefly describe the operation of the architecture and its essential features. Basically, the SVA disk system contains two tables. The first one holds the entries for the functional volumes and is called the "Functional Device Table" (FDT). The number of configurable functional volumes is restricted by the size of the FDT and the maximum size of the addressable subsystem capacity, the "functional capacity." The functional capacity is independent of the physical storage. Every table entry represents a volume of a certain size, which is a full-functional device for the operating system. When deployed in the Open Systems environment, several entries can be combined to form larger devices, i.e., more capacity associated with a specific address. LUNs of almost arbitrary size can be created to meet business needs without being limited by any restrictions of the physical disk hardware.

Figure 1 shows how the FDT creates a virtual disk environment with OS/390 volumes and Open System LUNs. The StorageTek SVA™ (Shared Virtual Array) Administrator (SVAA) software must be running at least on one of the attached servers and is used to create volumes and LUNS. Configuration details include characteristics like size and type of a new volume or LUN. The user-supplied information is transmitted to the SVA disk system by SVAA software. After the configuration process is completed the new volume or LUN can be used immediately. The configuration can be executed dynamically and, more importantly, nondisruptively by the customer.

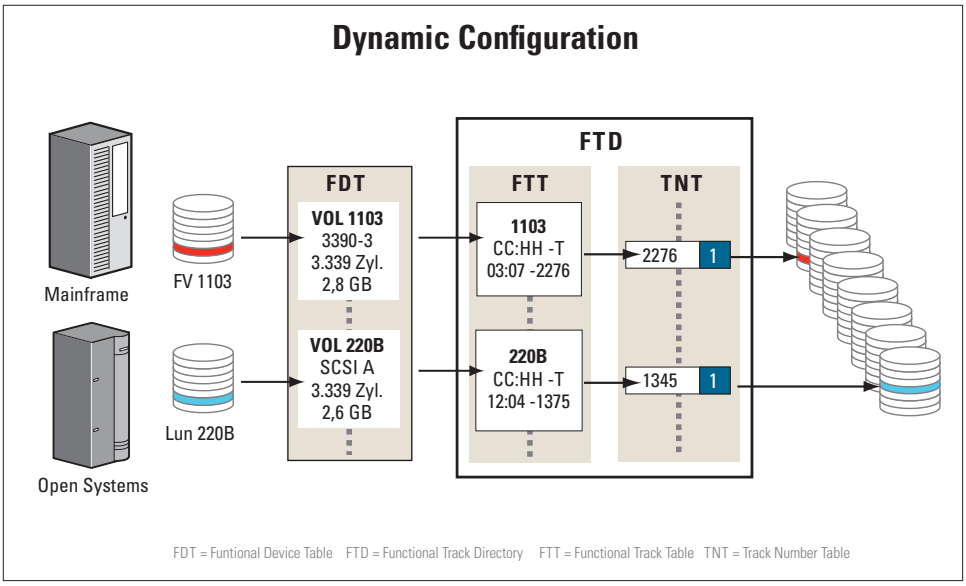

Figure 1. Dynamic configuration of virtual volumes and LUNs.

When user data is written to a volume or LUN (which are just table entries), the SVA disk system assigns the required amount of storage on the real disks. Thus, physical storage is not consumed until data is written to the SVA disk system. This feature is unique to the SVA disk system and provides many benefits. For this operation, a second table, the so-called "Functional Track Directory" (FTD), is also required (see **Figure 1**.). For every data unit stored, the FTD, which is managed on the functional level by the operating system, contains an entry describing its physical location and size. Because this space allocation is performed dynamically, the procedure is called "dynamic mapping." All other manufacturers use "fixed mapping," i.e., the physical storage space for data to be written in the future is already assigned and reserved at allocation time and is therefore no longer available.

The control unit in the SVA disk system chooses the location of new data from the entire physical disk pool and new data is always written to the less busy location. This process avoids hot spots that affect all other contemporary storage systems.

Of course, the SVA disk subsystem also incorporates a RAID protection mechanism: RAID 6+. RAID 6+ is similar to RAID 5 but has an additional level of redundancy that allows data to be recreated transparently even if two disks within an array should fail. The "+" indicates that the RAID protection is guaranteed without introducing additional overhead. For a write operation, the RAID 5 implementation demands reading all blocks of the RAID 5 group and before the actual update can take place, the parity information needs to be recalculated and then all blocks can be written. These additional I/O operations are called the "write penalty." For the SVA disk system this disadvantage is eliminated through the implementation of LSF (Log Structured File).

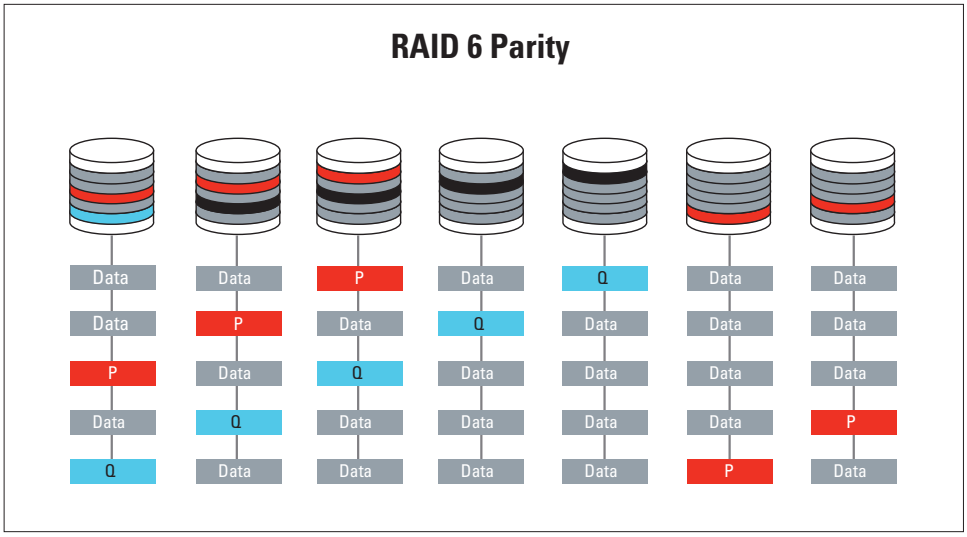

Figure 2. Distribution of data and parity with RAID 6.

3.2 LOG STRUCTURED FILE (LSF)

Another unique feature of the SVA disk system's virtual architecture is that new or changed data is always written to a new location on the back end based on performance and space. For changed data, new free space is always located and assigned dynamically. This procedure is called "no update-in-place" or LSF because it works like a log, i.e., new entries are always written to the end and the old ones left unchanged. A permanently running background task collects the old, invalid space and returns it to the free space pool.

LSF is also the enabler of the virtual SnapShot that creates copies of arbitrarily large amounts of data virtually instantaneously without moving data and without consuming physical storage. This replication technique and "point-in-time" quality is the heart of the high availability solution described in this document.

LSF is also the basis for RAID 6+ where "write updates" are performed without the otherwise unavoidable "write penalty." Furthermore, the compression of all data within the SVA disk system is also possible. With the update-in-place approach used by all other storage vendors, data compression is impossible because an update may enlarge a data block so that the space does not suffice anymore. These above explanations are illustrated in **Figures 3 and 4.**

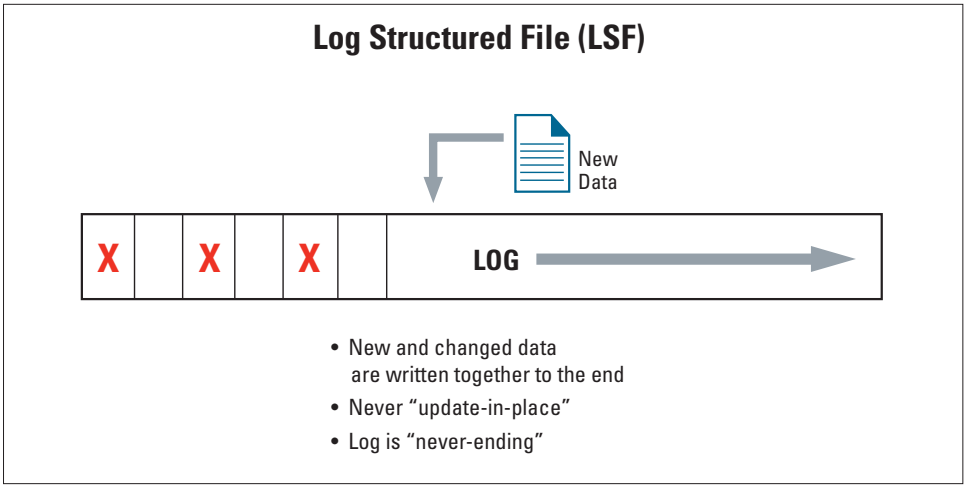

Figure 3. No “update-in-place” with LSF.

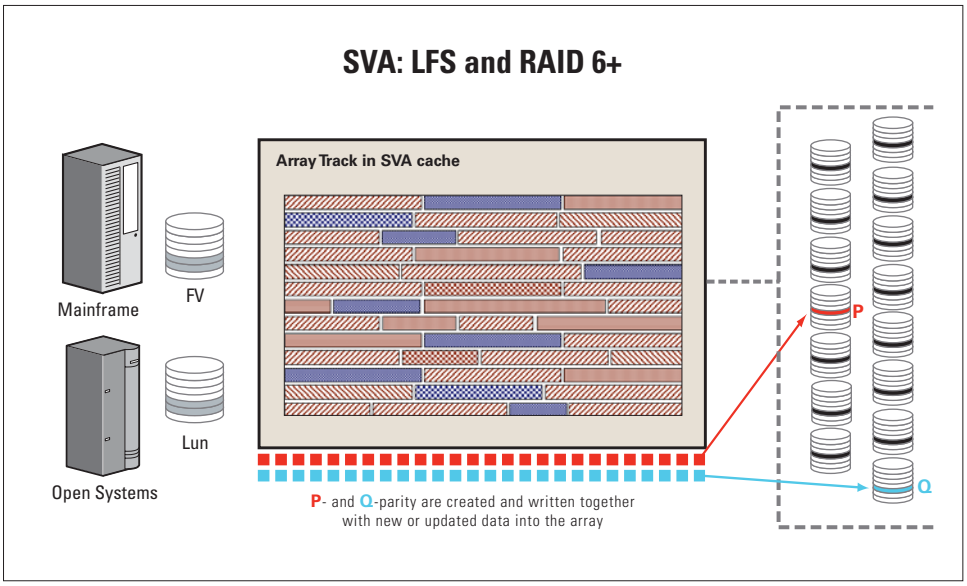

Figure 4. Write operation with LSF and RAID 6+.

3.3 STORAGETEK SNAPSHOT SOFTWARE

Only the SVA disk system allows data duplication by just copying table entries in the FTD and assigning them to another data set, partition or functional volume. The LSF implementation not only makes sure that the same data can be read from both the original and the copy, but can also be updated independently. Since every update is always written to a new location, the copy is unaware of any changes made to the original. The same is true for the original when the copy is changed. The virtual replication of a volume is shown in **Figure 5. Figure 6.** illustrates the virtual replication of a data set with two tracks. Finally, **Figure 7.** shows how an update of one of the copies is performed without impact to the other copy. The FTD is a two-stage table in which the Functional Track Table (FTT) contains the logical device addresses, as they are known to the operating system. The "Track Number Table"(TNT) has an entry for every logical address in the FTT that points to the physical data address in the back end. The duplication process consists of just copying the FTT pointers of a data set, a partition, or an entire functional volume and assigning them to the target volume within the FTT. This process is extremely fast and takes only a few microseconds, even for large amounts of data. This data duplication is at electronic speeds and has been measured at over 50 gigabytes per second.

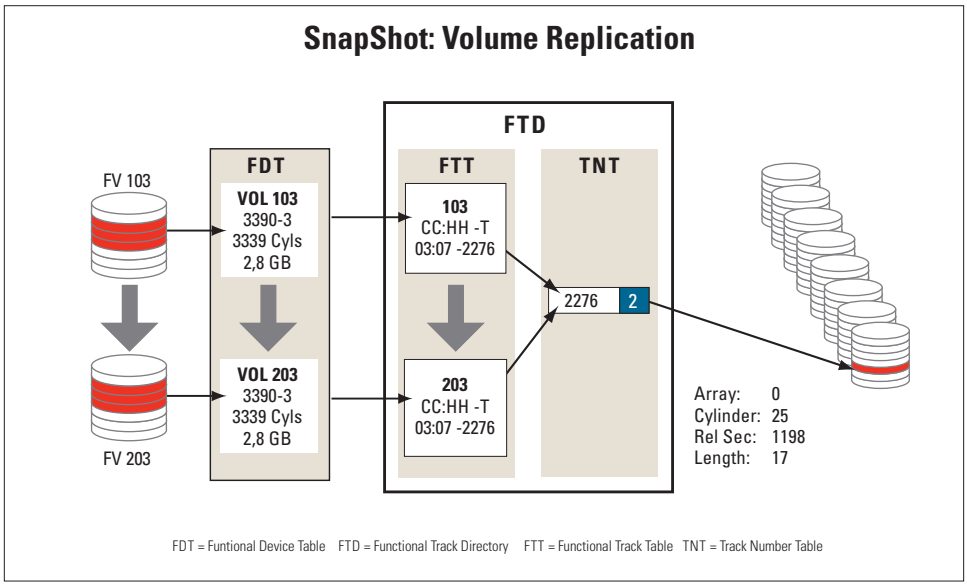

Figure 5. Volume SnapShot by replicating pointers in FDT/FTT.

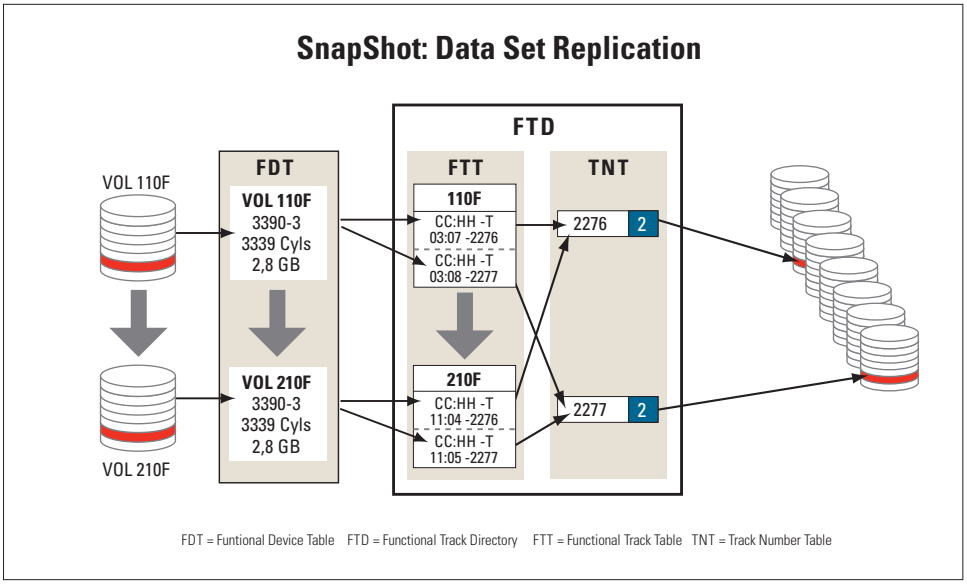

Figure 6. Data set SnapShot by replicating FTT pointers.

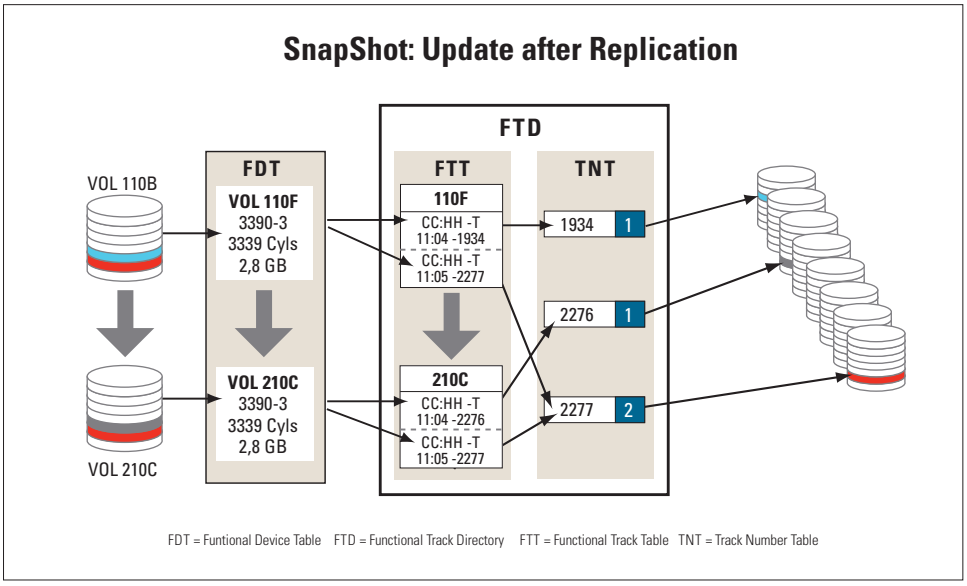

Figure 7. "Write Update" after SnapShot.

Two principal data replication techniques are generally used today. The first method permanently mirrors the data for which a copy will be needed in the future. When needed, the mirror is detached from its primary volume. With the second method, the control unit makes it appear that a copy is created at the moment the copying process is started. In reality, however, the required physical space is reserved and a copy process is started. The copy process can be byte for byte or the copy may occur when the original data is changed. Both procedures simulate a copy process and can be used with some limitations: Physical storage space of sufficient size must be available and reserved at the outset. Moreover, only a restricted number of simultaneous copies are possible.

A third and unique approach is provided by the SVA disk system. SnapShot holds a special position and cannot be assigned to either techniques identified above because snap-copies do not require a physical copy, do not consume any additional physical space, are executed at electronic speeds (versus electro-mechanical) and do not have any other restrictions on their use. Although data is neither replicated nor moved, they are "genuine" copies from the users' viewpoint and their emergence is fully transparent. An unlimited number of snap-copies can be made and these copies can be used for snap-backs to restore the original environment.

The two original replication methods described above using physical copies are subject to considerable restrictions. The arbitrary making of copies is impossible because physical space is consumed for each copy. Also, there is no concept of a snap-back, but only to make another physical copy. One will realize quickly that the replication is just a simulation. The bottom line is that SnapShot fulfills a major goal of virtual disk architecture by overcoming hardware-dependent restrictions.

SnapShot's mode of operation is not only an academic interest, but has positive effects on all processes dealing with data duplication. In this document, a range of applications for SnapShot is examined, particularly for databases and, especially, for IBM DB2. The uniqueness of SnapShot contributes fundamentally to the efficiency of the solutions described.

3.4 SVA DISK SYSTEM HARDWARE

The current StorageTek V2X Shared Virtual Array® (SVA™) disk system represents the seventh hardware generation of the virtual disk architecture and has been generally available since October 2002. It differs from its predecessor models by a newly structured control unit with much larger cache, through which performance and throughput are again considerably increased. The back end uses Fibre Channel disks instead of the SSA (subsystem service aid) drives used in earlier models of the SVA disk system. Particularly important in connection with this document is the four-fold increase in functional capacity, i.e., 4096–3390 mod 3 volumes can be configured independently of the physical capacity installed versus 1024 in previous versions.

The virtual disk architecture and the SnapShot functionality are common to all generations of the SVA disk system. Since the fifth generation, the SVA 9500 disk system, StorageTek PowerPPRC and StorageTek PPRC SnapShot software suite as well as other software products are available with the StorageTek Virtual Power Suite™ software. All statements made about SnapShot and the virtual disk architecture are true for all generations of the SVA disk system. **Figure 8** provides a hardware overview of the current V2X SVA disk system.

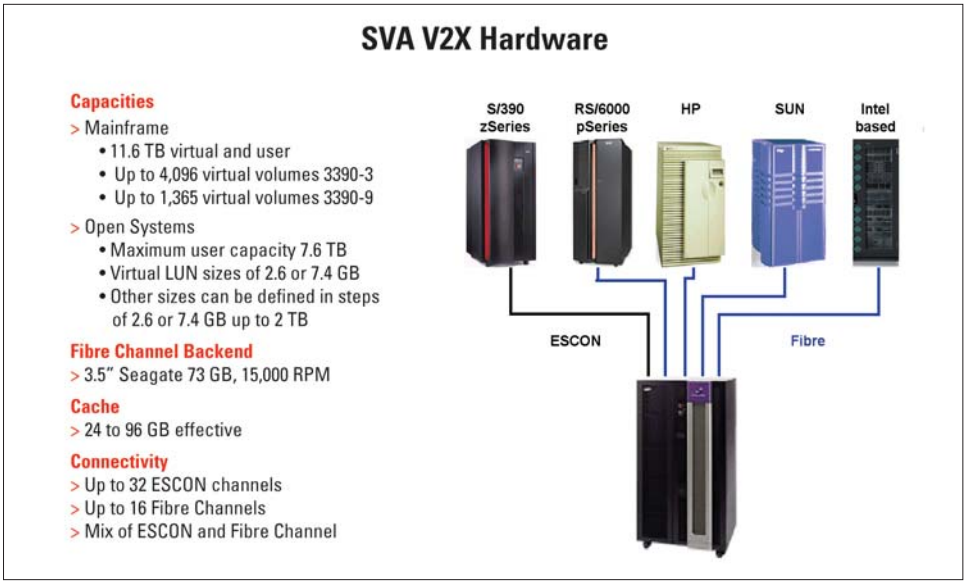

Figure 8. V2X SVA disk system hardware and connectivity.

4 DESCRIPTION OF THE TEST ENVIRONMENT

4.1 TEST VOLUMES

A V2X SVA disk subsystem was used for this testing. In preparation for the tests, a number of 3390 volumes were generated, configured and initialized. The first job was done quickly and easily by using the Shared Virtual Array Administrator (SVAA) software. SVAA software is the successor of IXFP and, for the most part, functionally identical with this IXFP software. Configuring is even simpler if the ISPF (interactive system productivity facility) panels of the SVAA software are used instead of the command line interface.

Under OS/390 or z/OS, the 3390-3 volumes are generated in the same way as disks with any other vendor. The initialization is carried out with the IBM utility ICKDSF and minimal physical storage is required on the SVA disk system for the definition of the volumes except for a label, a VTOC and, if possible, an indexed VTOC that can provide better performance.

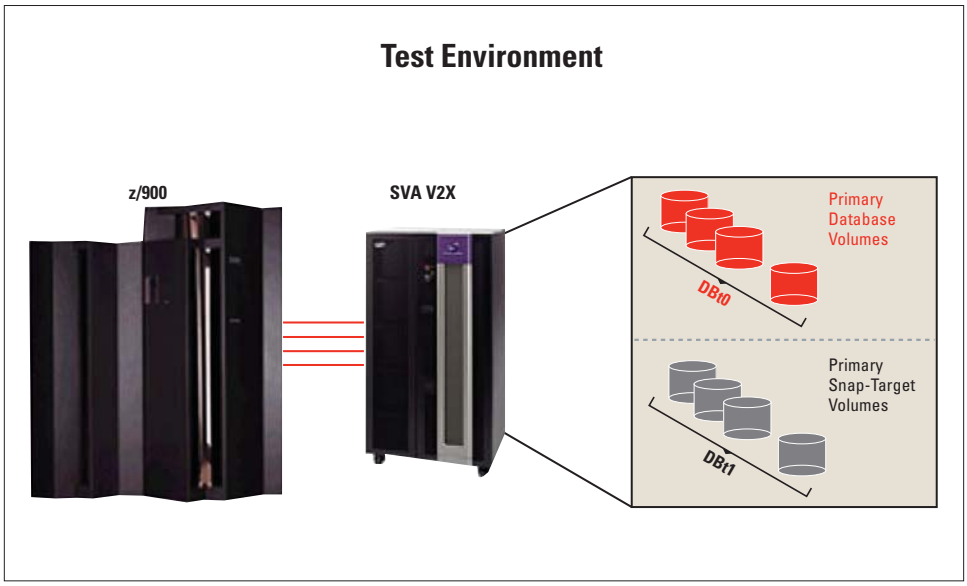

Figure 9. Test environment with 2 x 24 volumes 3390-3 for backup and recovery.

The V2X disk subsystem supports up to 4,096 functional volumes. For this test, two sets of 24 volumes 3390-3, marked as DBt0 and DBt1, were defined. The volumes of set DBt0 represent production volumes whereas set DBt1 serves as target for the SnapShot copies of set DBt0 (see **Figure 9**).

The 48 volumes did not have to be provided as groups with sequential addressing. But, for our testing this approach was chosen only for practical reasons as it simplified the execution of the tests and JCL coding. Further, there is no concern about physical volume placement. The SVA disk system handles the concept of volume placement automatically and avoids hot spots found in all other contemporary disk subsystems.

4.2 SOFTWARE RELEASES

The tests were executed under OS/390 Version 2.10. The database system was DB2 Version 6 and the application was SAP R/3 Version 4.6c.

4.3 GLOSSARY

The following abbreviations are used in the document:

Term	Description
Snap-copy	Copy of a data set or volume created with SnapShot
Snap-source	Original of a data set or volume replicated with SnapShot
Snap-target	Target data set or target volume of a SnapShot
Snap-back	Reversal of a SnapShot, i.e., copying back a snapped data set or volume to the original
Re-Snap	Repeating a SnapShot

5 BACKUP AND RECOVERY: TABLESPACE SNAPSHOT

Because of its unique approach, SnapShot provides a very fast, convenient and resource-saving method for creating backups. Contrary to ImageCopy, it can even copy the largest tablespaces within seconds. Therefore, it is acceptable to stop DB2 write activities with the command SET LOG SUSPEND for the duration of the snaps and continue afterwards with the SET LOG RESUME command.

The DB2 command SET LOG SUSPEND stops any write activities to the log and the tablespaces. After issuing this command, all tablespaces are in a frozen state. In addition, all log buffers not yet written to disk are emptied; a system checkpoint is set (only in a "non-data sharing" environment), and the highest RBA (relative byte address) (before suspending) is stored in the BSDS. The highlighted message DSNJ372I appears on the console and is frozen until SET LOG RESUME is issued and logging can continue.

Like any "normal" copy program, SnapShot can now be invoked immediately for replicating some or all tablespaces. Since this task merely consists of duplicating pointers, no physical space is required for the copied tablespaces and no data is moved either! Only data changed by later updates of the original tablespaces consume additional storage. There is no comparable method by any vendor that is so efficient in terms of CPU and disk subsystem and yet so easy to execute.

As mentioned earlier, the environment is a DB2 system with 24 functional 3390-3 volumes. In the first step we wanted to examine the possibility of creating snap-copies of single tablespaces that could be used for a recovery. If that is possible, one has a very fast and simple backup procedure for tablespaces. The feasibility and practicality of this is proven with the following small test. The snap-copies were created in online mode, i.e., without stopping DB2. The DB2 commands SET LOG SUSPEND and SET LOG RESUME were issued to prevent changes to the tablespaces only for the short time while the snap-copies were created.

5.1 TEST SEQUENCE

Step	Description
0. Initial state	DB2 is started. No updates are taking place and the system is in the static state A.
1. Job 1	Implements changes to a particular tablespace X.
2. Job 2	Update activity of Job 1 is stopped with SET LOG SUSPEND. DB2 is in state B now.
3. Job 3	Meanwhile, changed tablespace X is copied by "snap dataset" while DB2 is in state B.
4. Console command	SET LOG RESUME is issued and Job 1 continues with write activities.
5. Job 1	Finishes its write activities. The state of DB2 has changed from B to C.
6. Console command	Stop DB2.
7. Job 4	Create a tablespace report, which documents the state changes of tablespace X.
8. Job 5	Snap-back tablespace X copied with Job 3 above. Now the system is in an inconsistent state. System tables are in state C, whereas tablespace X is in state B.
9. Job 6	Invokes the DB2 recover utility to perform a forward recovery of tablespace X using the snap-copy from Job 3 and the DB2 log. Tablespace X is transferred to state C. Afterwards an "index rebuild" is performed and DB2 is consistent again.
10. Job 7	Creates a tablespace report to document the successful execution of Job 6.

Table 1: Test job description.

Documentation of the two SnapShot jobs (job 3 and job 5) used in the test:

```
//Job-Card
//STEP1 EXEC PGM=SIBBATCH,PARM='PARMLIB=SIBSTK88'
//STKPARMS DD DISP=SHR,DSN=XXX.PARMLIB
//SYSPRINT DD SYSOUT=*
//SYSTERM DD SYSOUT=*
//SYSIN DD *
 SNAP DATASET( -
 SOURCE(DB2K99.DSNDBC.A001XC0W.XSAP.I0001.A001) -
 TARGET(SPC.RV9AB8.SAP.SNAP) -
 VOLUME(RV9AB8) -
 CAT(YES) TOLENQF(YES))
```

Parameter of the "snap dataset" command:

- CAT(YES): Target data set will be cataloged.
- TOLENQF(YES): Source and target data set will be used, even if no exclusive control is possible. In our test, all data sets to be snapped are always open and write-accessible by DB2, although they are temporarily locked with SET LOG SUSPEND.

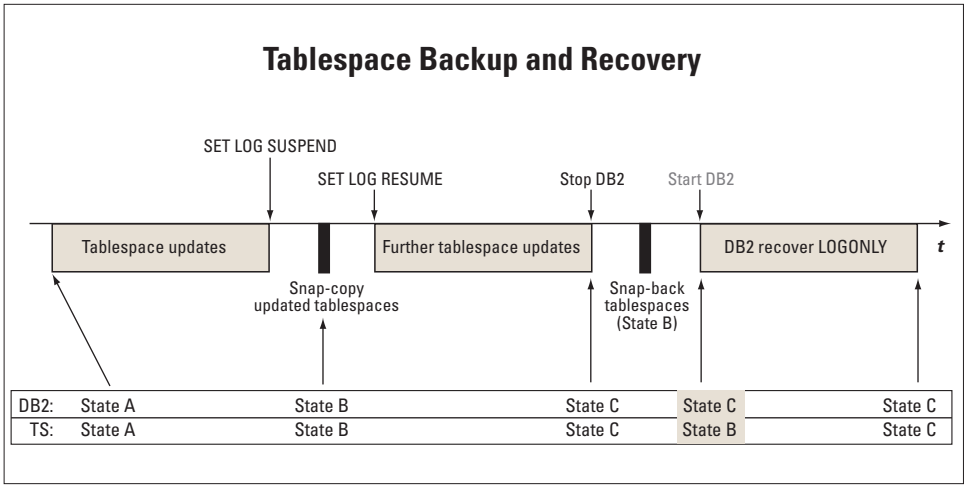

Figure 10. Tablespace backup and recovery with snap/snap-back.

5.2 TEST RESULTS

This small test proved that snap-copies created beyond the control of DB2 can be used as the basis for a tablespace recovery. This can be executed elegantly and quickly if the snap-copies are still resident on the SVA disk system. It is simply performed by a snap-back. SnapShot can be used in both directions, i.e., for backup and restore. All parameters were controlled and it was known exactly which tablespaces were changed by which updates.

5.3 EVALUATION

If it is known which tablespaces need to be recovered quickly and they are frequently replicated, snap-copies of tablespaces on the data set level offer a very fast, reliable, convenient and economic solution. However, this approach is reasonable for only a limited number of tablespaces. Since the copies created with SnapShot are not managed by DB2, image copies may still be required. However, this approach is quite valuable as an additional tool that can increase the availability of a database. Quite often there is a limited number of really business-critical tablespaces in a database. The DB2 administrator usually knows them and can use this approach. In daily practice, SnapShot can be used before REORGs and batch processing to help save time as it may be possible to make image copies of the SnapShot volumes. Virtual Concurrent Copy, which also exploits SnapShot, can be used to expedite the image copy process. See the IBM Redbook: *Implementing DFSMSdss SnapShot and Virtual Concurrent Copy*, IBM publication number SG24-5268, for details.

6 BACKUP AND RECOVERY: VOLUME SNAPSHOT

Backup and recovery on tablespace level is no longer practical when the number of tablespaces becomes extremely large. Today, a SAP DB2 database may have 30,000 or more tablespaces and certainly more will be added with future releases. This is not SAP-specific but typical for all ERP systems. An additional obstacle is that dependencies between the tablespaces are unknown.

According to SAP OSS Note 108469 (/51/) an R/3 database must:

> ...from the operational and semantical viewpoint (...) in its entirety needs to be considered a unit of recovery. The reason is a high level of integration of the R/3. That places some specific requirements on the backup and recovery procedures that are not common in traditional DB2 environments, e.g., a prior point in time recovery cannot be normally done on a per-tablespace basis but rather for the whole R/3 database which means all the tablespaces and indexes that make up the R/3 database including the DB2 catalog and directory…must be recovered.

Therefore, a strategy based on backup and restore of entire volumes should be used. Beginning with this in mind, we executed a test in which a complete database, including system files and tables, were copied with volume SnapShots. A goal of the test was to perform a recovery from volume snap-copies, not single data sets. The same prerequisites as in the case of data set snaps applied, i.e., the DB2 update activity was locked for a short time with SET LOG SUSPEND and SET LOG RESUME while the volume snaps were executed. Of course, for a database backup this procedure assumes that the interruption is so short that no time-out problems occur, i.e., the SnapShots must be executed considerably below this critical time threshold. IBM recommends in /1/:

> Do not keep log activity suspended during periods of high activity or for long periods of time. Suspending update activity can cause timing-related events such as lock timeouts or DB2 and IRLM diagnostic dumps.

For SAP R/3 applications the recommended time out value for locking a resource is 600 seconds, which is sufficient for creating volume snaps. The time out default is 60 seconds. If one uses the above procedure for applications whose DB2 subsystem uses the default value, then it can be set higher dynamically with the DB2 command MODFY irlmproc, SET TIMOUT=nnnn,subsysname and after execution of SET LOG RESUME be lowered again.

This test also took place under strictly controlled conditions, i.e., it was defined exactly which changes would be carried out for which tablespace. Only the time of the interruption by SET LOG SUSPEND was chosen coincidentally during the update activities. Because it was our goal to be able to comprehend every single step in detail, this approach was justified. Of course, one does not have such ideal conditions in a normal production environment and some analysis is required to determine the best possible time for a point-in-time recovery.

6.1 TEST SEQUENCE

Step	Description
0. Initial state	DB2 is started. No updates are taking place and the system is in static state A. DB2 system and log data are separated from the data tablespaces on different functional volumes.
1. Update database	Job 1 updates a particular tablespace.
2. SET LOG SUSPEND	Update activity of Job 1 is interrupted with SET LOG SUSPEND. DB2 is in state B now.
3. Volume snaps	All 24 DB2 volumes, including logs, system and data tablespaces, are replicated with several jobs, i.e., state B is saved. Every database volume of set DBt0 is duplicated on a functional volume of set DBt1 within the same SVA disk system. We used the parameter CONDVOL(LBL) which prevents the original VOLSER of the target volume to be overwritten. This approach has the advantage that the targets can remain online after the snaps. Of course, it is not possible to start DB2 from these volumes because the VTOC indexes and the VVDSs contain the names of the source volumes. Nevertheless, the targets could still be copied physically or snapped again.
4. SET LOG RESUME	SET LOG RESUME is issued and Job 1 continues with its write update activities.
5. Update database	Job 1 finishes write activities and the state of DB2 changes from B to C.
6. Console command	Stop DB2.
7. Tablespace reports	Create reports documenting the state changes of the tablespaces. Filter out "manually" all volumes containing changed tablespaces.
8. Volume snap-back	Snap-back of all volumes with tablespaces changed after SET LOG RESUME. It must be paid attention to that only volumes with data tablespaces are snapped-back. Volumes with DB2 system data, like catalog, Logs and BSDS, may NEVER be snapped-back because after the snap-back the data table-spaces are in state B whereas the system tables are in state C; the system is in an inconsistent state.
9. DB2 recovery	The DB2 recover utility performs a forward recovery for all tablespaces, which were brought back into state B by snap-back. In our test, and also in a normal production environment, there are lots of tablespaces remaining unchanged after performing the snaps in step 3. This fact reduces the recovery to those tablespaces, which were changed after the last SnapShot (state B). The databases for the recovery are the target volumes created with the volume SnapShots in step 3. However, there is one difficulty: The tablespaces on the snap-targets aren't immediately useable because the volume labels do not match the data, i.e., the names of the VTOC index and the VVDS still contain the volume labels of the source volumes. By not using the tablespaces of the target volumes for the recovery but the tablespaces snapped-back to the original volumes instead solves the problem. The bases for the recovery (restore the affected tablespaces to current state C) are the copies created in step 3 and snapped-back in step 5 as well as the actual (not snapped-back) DB2 log. After the recovery the database is consistent again.
10. Tablespace report	The recovery result was verified with SPUFI.

Table 2: Test job description.

JCL for the volume SnapShots used for testing (steps 3 and 8).

Copy all 24 DB2 volumes of set DBt0 (production volumes) to set DBt1 (snap-target):

```
//JOB-Card
//STEP1 EXEC PGM=SIBBATCH,PARM='PARMLIB=SIBSTK88'
//STKPARMS DD DISP=SHR,DSN=XXX.PARMLIB
//IN01 DD DISP=SHR,UNIT=3390,VOL=SER=XGK901
//OUT01 DD DISP=SHR,UNIT=3390,VOL=SER=RV9AC0
//IN02 DD DISP=SHR,UNIT=3390,VOL=SER=XGK902
//OUT02 DD DISP=SHR,UNIT=3390,VOL=SER=RV9AC1
//IN03 DD DISP=SHR,UNIT=3390,VOL=SER=XGK903
//OUT03 DD DISP=SHR,UNIT=3390,VOL=SER=RV9AC2
//SYSPRINT DD SYSOUT=*
//SYSTERM DD SYSOUT=*
//SYSIN DD *
 SNAP VOLUME( -
 INDD(IN01) -
 OUTDD(OUT01) -
 COPYVOLID(NO) REPLACE(YES) CONDVOL(LBL) )
 SNAP VOLUME( -
 INDD(IN02) -
 OUTDD(OUT02) -
 COPYVOLID(NO) REPLACE(YES) CONDVOL(LBL) )
 SNAP VOLUME( -
 INDD(IN03) -
 OUTDD(OUT03) -
 COPYVOLID(NO) REPLACE(YES) CONDVOL(LBL))
```

Eight jobs in total with three volume SnapShots each were executed. The volume snaps may also be performed sequentially in one task within a single job.

Parameters of the SNAP VOLUME command:

> COPYVOLID(NO): The VOLSER of the target will not be overwritten by the VOLSER of the source.

> REPLACE(YES): All data on the target will be replaced by the source data.

> CONDVOL(LBL): The names of the indexed VTOC and the VVDS get the VOLSER of the target, this is SYS1.VTOCIX.volser and SYS1.VVDS.Vvolser.

JCL to snap-back the volumes (step 8).

Snap-back volumes of set DBt1 containing tablespaces changed after SET LOG RESUME. Any volume with system data should never be snapped-back:

```
//JOB-Card
//STEP1 EXEC PGM=SIBBATCH,PARM='PARMLIB=SIBSTK88'
//STKPARMS DD DISP=SHR,DSN=XXX.PARMLIB
//OUT02 DD DISP=SHR,UNIT=3390,VOL=SER=XBK901
//IN02 DD DISP=SHR,UNIT=3390,VOL=SER=RV9AC4
//OUT03 DD DISP=SHR,UNIT=3390,VOL=SER=XBK902
//IN03 DD DISP=SHR,UNIT=3390,VOL=SER=RV9AC5
//SYSPRINT DD SYSOUT=*
//SYSTERM DD SYSOUT=*
//SYSIN DD *
 SNAP VOLUME( -
 INDD(IN02) -
 OUTDD(OUT02) -
 COPYVOLID(NO) REPLACE(YES) CONDVOL(LBL) )
 SNAP VOLUME( -
 INDD(IN03) -
 OUTDD(OUT03) -
 COPYVOLID(NO) REPLACE(YES) CONDVOL(LBL) )
```

Ten jobs were executed to snap-back 10 volumes in total.

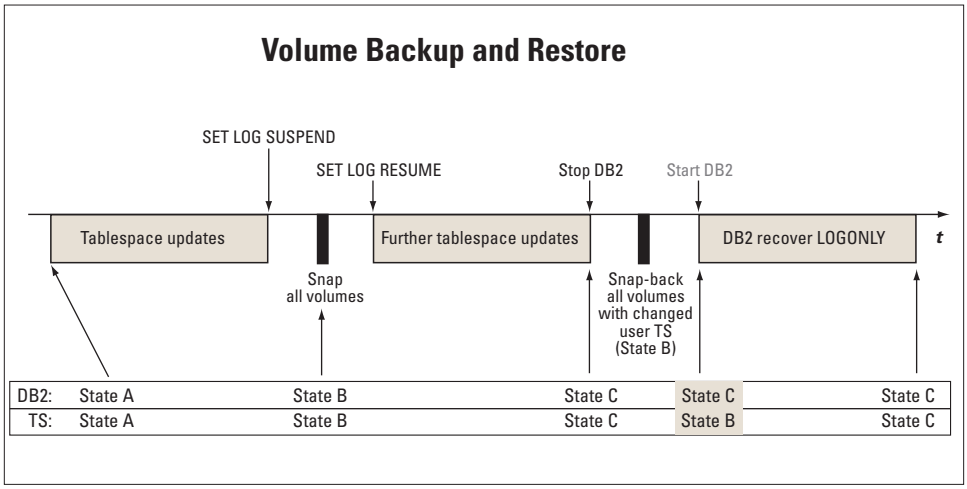

Figure 11. Volume backup and restore.

6.2 TEST RESULTS

Volume SnapShots can be used without any restrictions for an automatic DB2 recovery with the recover utility by supplying LOGONLY and REBUILD INDEX. However, it is imperative that the current DB2 logs, system tablespaces and the BSDS are available. Because of this restriction, no volumes with system data in state B are recovered. That is only possible if the application data is separated from the DB2 system data on different functional volumes.

6.3 EVALUATION

The procedure described for creating database backups with volume snaps is a cost-effective and easy to implement and manage alternative to classic backup procedures that require physical data duplication, movement to another storage location, and, often most importantly, time. SnapShot is exceptionally fast because no data is moved. It is economical because it does not require additional physical storage. For the same reason, it is also easy to implement and manage. The greatest advantage is the added flexibility because for selecting the target volumes, no preparations are required on the hardware level. As opposed to traditional solutions, the user can now decide from an organizational viewpoint and is no longer restricted by physical hardware.

As already mentioned earlier, snap-copies can be executed like "normal" copies, i.e., the advantages of SnapShot do not impose any hardware restrictions on the disk subsystem. The ability to execute a snap-back makes it fundamentally easier to reset a database quickly to a known state (state B in this document). Of course, this assumes that older snap-copies remain resident on the SVA disk system, although they were probably already backed-up to tape. The price the user is paying consists only of the consumption of functional addresses for the snap-targets and additional physical storage for changed data.

Of course, in a production environment, when and which tablespaces are changed cannot be controlled. Therefore, the conversion of the tested procedure into a practical approach requires further efforts.

7 BACKUP AND RECOVERY: VOLUME SNAPSHOT WITH INFOMAT

7.1 FREQUENT BACKUPS VERSUS RESOURCE USAGE

To protect against data loss, regular backups are required. For a DB2 database the backups are normally performed with the DB2 copy utility. The extent of the data modifications carried out after a backup determines the runtime of a possible recovery. The runtime for the regular DB2 recovery utility is the sum of the recovery phase (restore of the tablespace cluster) and the so-called "log-apply phase" (search the logs for changes since last backup).

If the data is backed up only once per week, the data backed up from a week ago must be used in the extreme case. Because of the time required to read the logs, the recovery times can be long and cannot be avoided. More frequent backups reduce the recovery time and the amount of data that is affected by the restore operation. However, more frequent backups require additional expenses for resources, like CPU, channels, storage, time and staff.

If an error requiring a database reset occurs, the latest backup copy from the time before the event has to be used. With this backup and the logs, one tries to find the optimal time to which the database can be reset to a consistent and error-free state. The larger the time interval between the last backup and the time of the error, the longer the recovery will last. During the recovery phase, access to the data being restored is not possible.

Every company must decide on its own for how long it can afford data to be unavailable and define how frequently and on which media backups should to be taken. An individual solution needs to be developed for solving the conflict between the expenses for backups and the data availability requirements. This is the classic dilemma of ERP users: either frequent backups with a high overhead in time, administration, and money or an elaborate recovery with long, expensive production outages.

7.2 IMAGECOPY VERSUS SNAPSHOT

The classic DB2 utilities for backup and recovery are COPY and RECOVER. The big advantage of using COPY is the administration of the copies in the DB2 catalog (SYSIBM.SYSCOPY), where all relevant information for a successful tablespace restoration is kept. The restore is executed with the recover utility. If there are several image copies, the utility selects the most current version automatically. Depending on its age, additional information is retrieved from the active log and, if necessary, from the archive log as well. The procedure is safe and proven, but reaches its limits for ERP systems with many thousands of tablespaces. For many enterprises even using the COPY option SHRLEVEL (REFERENCE) (OFFLINE backup) causes problems because a consistent copy is created, but write access to the tablespace is denied until the copy is done. Therefore, many users deploy the option SHRLEVEL (CHANGE) (ONLINE backup), which will not stop write access to the tablespaces being copied. Because the copies created are inconsistent ("fuzzy images") the recovery must include more log information and will last longer.

The fact that often very large amounts of data prohibit creating image copies at the frequency required for a sufficient availability of enterprise-critical applications is more serious. Lower backup frequencies can increase downtime for a recovery in case of an error. Of course, this also depends on the number of data modifications.

SnapShot offers a solution for this dilemma and improves the described situation for backups and restores significantly. No data is moved and no physical resources are required at the time of the snap. Duplicating data with SnapShot takes a matter of seconds and is very easy as well. The two large cost factors, resource consumption and administration, are reduced to a minimum. SnapShot saves time in multiple ways: first, for the backup of the database volumes, which is carried out so fast that it can virtually be executed during normal production. Secondly, a volume restore can also be done with SnapShot if the backup copies are still resident in the SVA disk system. Using SnapShot allows a considerably higher backup frequency, which in turn shortens the elapsed time between backups and, when necessary, a recovery can be executed much faster.

Because of the economies of SnapShot, it can and should be considered to keep not only one but several snap-copies in the SVA disk system because they consume functional space only and hardly any physical storage. Current SVA disk subsystems support up to 4,096 functional 3390-3 volumes. This corresponds to a total functional storage capacity of approximately 11.6 terabytes and is independent of the physical capacity installed. Only data changed after a snap consumes additional disk space. Experience indicates that this amount of data is very small compared to the total size of a database. A snap-back is executed just as quickly and easily as the snap before. Technically, there is no difference between a snap and a snap-back; merely the copy direction is different. The circle closes here: Because of the way SnapShot is implemented, it is possible to replicate data very frequently with a minimum of time and cost. Recovery time will also be reduced dramatically because SnapShot can also be used to restore data sets or volumes.

DB2 manages image copies in the DB2 catalog. The user must administer the backups of the volumes created with SnapShot, either manually or with a tool. This leads us to the issue of a practical solution for a backup and recovery strategy with SnapShot.

7.3 BACKUP/RECOVERY: STRATEGY WITH SNAPSHOT

The scenarios discussed so far have explained the operation and principle usefulness of snap-copies for backup and recovery purposes. Now the attention must be turned from tests with "laboratory character" to a useful SnapShot solution in an arbitrary production environment. For this case, one needs to solve the following problems:

1. The time frame between SET LOG SUSPEND and SET LOG RESUME must be held so small that no serious time-out problems (batch processing) occur.
2. Automatic administration of the snap-copies.
3. Automatic selection of the best-suited snap-copy for the recovery process.
4. A procedure to find those tablespaces, which have changed since the selected snap-copy was taken. The unchanged tablespaces don't need to be recovered. Significant time savings can be achieved if they are excluded from the recovery.
5. Investigating which tablespaces were deleted or newly created between the time of the backup and the time selected for the recovery.
6. In case of a point-in-time recovery: determination of the optimal time the database should be recovered to. This should be selected according to the lowest possible data loss and a guaranteed database consistency.

7.4 POINT-IN-TIME RECOVERY WITH INFOMAT

Many system administrators won't be able to fulfill the listed requirements without a suitable tool. Of the solution providers available on the market, we decided to partner with InfoDesign and their tool InfoMat. It fulfills the requirements 1–5 defined above and was designed to invoke SnapShot automatically if the data to be replicated resided on a SVA disk system from StorageTek or RVA from IBM. InfoMat uses DFDSS as the data mover that in turn invokes SnapShot automatically. The API to call SnapShot is integrated in DFDSS for z/OS and OS/390 and no customization is required for the activation. If DFDSS recognizes that all prerequisites for a SnapShot operation are fulfilled, it invokes a SnapShot and prevents the data from being moved physically (/13/).

The write activities of the DB2 database are stopped temporarily with SET LOG SUSPEND and, after snapping all 24 volumes from set DBt0 to DBt1, continued with SET LOG RESUME. Thereafter, a logical error was deliberately injected by deleting certain data records in the database. By supplying the time of the error, now InfoMat was supposed to reset DB2 as close as possible to the time when the "error" occurred in order to minimize data loss. The database was stopped and the volumes backed-up before a snap-back to the original volumes of set DBt0 was executed. The volumes containing DB2 system data were excluded. The snap-back jobs were created and executed by InfoMat automatically.

7.4.1 INTERMEDIATE RESULTS

At this time the following components were available:

- A database with current DB2 system components (DB2 catalog, directory, log and BSDS)
- User tablespaces with the contents of the last SnapShot (state B)

Now, a forward recovery was executed for the tablespaces that had changed since the last snap. The unchanged tablespaces were excluded. After the target time for the PIT recovery was chosen via the InfoMat selection panels, the required jobs were created and executed automatically. The process and the result of the test are shown in **Figure 12**.

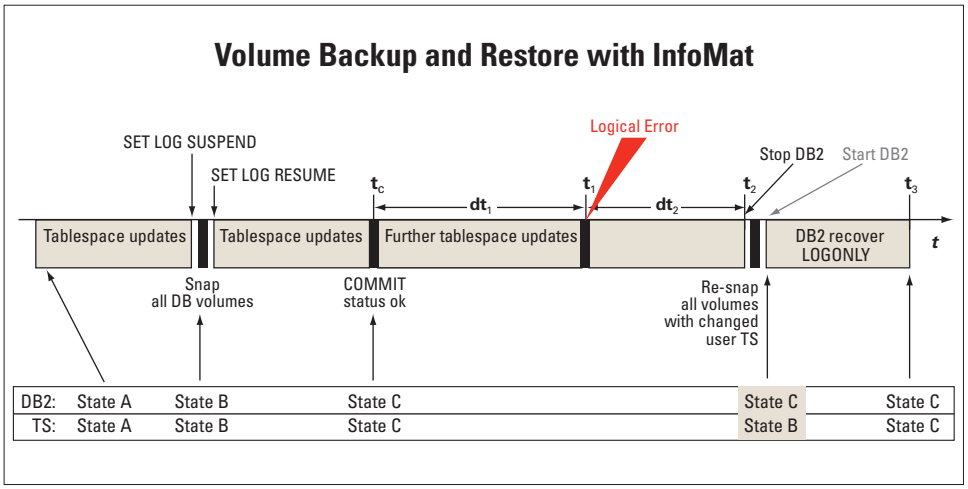

Figure 12. Recovery phases after a logical error.

7.4.2 Detailed test description

We began testing by submitting a job with SQL update statements after DB2 was started. We marked the initial state of the system state A. Update activity was interrupted with SET LOG SUSPEND. At this time the database was in state B. Generally, this is not a consistent state because "in-flight" transactions (started before issuing the command but not completed by a COMMIT at the time of SUSPEND) are also interrupted.

All database volumes containing system and application data were replicated with a volume SnapShot. It was important that the snapped volumes also included all user catalogs. In addition, InfoMat requires the separation of the application from the system data and the user catalogs on a volume level, i.e., at least one user catalog is required for each volume with application or system data. The catalogs for the application data must reside on the application data volumes and the catalogs for the system data on the volumes with the system data. With the replicated volumes it is now possible to:

1. Start DB2 with these volumes, though only simultaneously with a production system on another completely separate LPAR. At startup time the in-flight transactions would be rolled back and DB2 consistent, but of course, not up-to-date.
2. Use the copies to create a homogeneous database copy at an arbitrary copying point. For this purpose InfoMat provides corresponding functionalities.
3. Use the tablespaces on the data volumes for a recovery if a logical error requires a point-in-time recovery, i.e., resetting DB2 from an earlier state independent of the backup time.

Variant 3 was the goal of our test.

Write locking was lifted with SET LOG RESUME and the update activities could continue after the snap. At time t1, a logical, error was "discovered," which was simulated in our test. The organizational recovery started at time t1. During this phase a person responsible must take care that the error does not spread further within the database system and take appropriate actions to remove the consequences entered until then. For our test, we assumed that at time t2 the decision was made to stop the database and to perform a recovery. Now, the phase of the "technical" restoration started.

First, it had to be determined when the logical error entered the system. To investigate the corresponding system time from the details available, the help of the responsible department is required. This was the input to InfoMat, which figured out the correct RBA from which the point-in-time recovery should start. The selection criterion for the RBA is to restore the DB2 to a consistent, error-free state with minimum data loss. To achieve this goal, InfoMat explores different sources, like active logs, archive logs, BSDS and SYSLGRNX. From the RBA setting, a generation job created further jobs for a conditional restart of DB2 by selecting a previously created snap-copy.

The generated jobs performed the following tasks:

1. Entry of a conditional restart in the BSDS.
2. Selection and entry of a start- and end-RBA for the recovery after the restart in the BSDS. The start-RBA corresponds to the time of SET LOG SUSPEND before the youngest snap-copy before the error occurred. The end-RBA corresponds to the most suitable time for the point-in-time recovery discovered from different sources.
3. Compile and link DSNZPARM with parameter DEFER. This prevents DB2 from starting a recovery automatically at startup time.
4. Printing of SYSLGRNX to retrieve the tablespaces changed between start- and end-RBA. Only these need to be restored. The tablespaces unchanged after the snap can be reused immediately after the snap-back.
5. Printing the logs and evaluation to find those tablespaces, which were deleted or newly created between start- and end-RBA. Deleted tablespaces should not be included in the recovery right from the beginning. The recovery for the newly created tablespaces couldn't be based on snap-copies because they did not exist at the time of the snap. For this reason image copies made after their creation need to be used instead. Special cases are those tablespaces that were deleted and newly created using the same name. They certainly will reside in another place than the original. The current user catalog only knows about newly created tablespaces and points to positions where these candidates cannot be found on the snap-back volumes. On the other hand, because the old tablespaces with the same name still exist on the snap-back volumes, the old, saved user catalogs can be used to find the candidates and delete them again. The newly created tablespaces with the same name have to be handled like the other newly created tablespaces mentioned above.

The analysis phase was completed at that time and the real recovery was started.

6. Snap-back the application data volumes. If several snap-copies exist, the last one created before the error is selected. InfoMat generates the JCL for the snap-back and executes the job automatically. Only volumes with application data and accompanying user catalogs are copied to the current volumes. The tool allows determining the number of concurrent snap-back tasks, but the optimal number was not investigated during the test. However, it is proven that users snapping several hundred volumes achieved clear performance gains when running multiple snaps in parallel. This tuning possibility plays only a minor role when testing with 24 volumes and the system data volumes are excluded from the snap-back anyway.

Finally, DB2 was restarted and the real recovery was executed with the DB2 recover utility. First the DB2 catalog and the directory had to be restored. Afterwards the tablespace recoveries of the application volumes were performed. For the affected tablespaces (state B) on these volumes a forward recovery by using the current log was executed using the selected start- and end-RBA. As already mentioned, the deleted and newly created tablespaces, which have to be treated in a special way, were excluded. InfoMat created the required recovery jobs, which optionally can be submitted by a scheduler

After a successful execution of all steps we checked the database and noticed that the "error" had been removed and DB2 was in state C (our goal) again. Now DB2 can be started "normally" and released for general use again. The test was completed successfully at that time. In a production environment, the decision still has to be made on how to handle the loss of the newly created data during time periods dt1 and dt2 (**Figure 12.**). This is the responsibility of the application group again.

InfoMat cannot take into account the partitioning in source volumes and snap-targets automatically. This must be carried out "manually" by filling out a table once. After changes (deleting or adding volumes), an automated matching is optionally possible. However, after such "hard" structure changes, older snap-copy versions are no longer useable.

The question that might arise now is: What can be done to achieve a ZERO data loss? There is no data loss from the viewpoint of InfoMat. It only arises from the "characteristic" of the database system to treat long running units of work as one unity. Therefore the case data loss=0 exists in theory only. But a data loss can be minimized by avoiding long running units of work, for example by a frequent commit. A data loss also occurs by the logical cohesion of a unit of work in case of a rollback because all changes must be reversed. Additionally, practical experience indicates that nobody knows which bookings from an hour ago were correct and which were not.

Further information about the InfoMat software and the vendor InfoDesign is available under: **http://www.infodesigner.biz/**.

8 FINAL REMARKS

Compared to traditional methods, the point-in-time recovery could be shortened considerably by:

> Using snap-back instead of a physical copy for the restore.

> Deploying a tool that creates and manages SnapShots and selects the right version for the snap-back. This is important because it saves considerable administration overhead and time.

> Restoring only those tablespaces that were changed in the questionable time period. The more frequently backups are created, the fewer tablespaces will be affected and fewer tablespaces will have to be restored, i.e., the overhead to find out the affected candidates is then worthwhile.

Only by combining the snap-back technique, which copies even the largest amounts of data very fast and efficiently, and the "intelligent" determination of objects changed in between the primary goal can be accomplished: a fast and consistent recovery of a database at any time.

9 REFERENCES

9.1 IBM DOCUMENTS

/1/ DB2 UDB for OS/390 and z/OS V7 – Command Reference

/2/ DB2 UDB for OS/390 and z/OS V7 – Utility Guide and Reference

/3/ DB2 UDB for OS/390 and z/OS V7 – Data Sharing: Planning and Administration

9.2 IBM REDBOOKS

/10/ SAP on DB2 for z/OS and OS/390: Homogeneous System Copy (SG24-6287-00)

/11/ High Availability Considerations: SAP R/3 on DB2 for OS/390 (SG24-2003-00)

/12/ SAP R/3 on DB2 UDB for OS/390: Database Availability Considerations

(SG24-5690-00)

/13/ Implementing SnapShot (SG24-2241-01)

9.3 WHITE PAPERS

/30/ SAP R/3 Storage Management for OS/390:

/31/ Split Mirror Backup Recovery on IBM's RVA Storage Control

SAP Service Marketplace: http://www.service.sap.com/split-mirror -> Backup/Recovery

SAP OSS Notes

/50/ SAP OSS Note 363189

/51/ SAP OSS Note 108469

/52/ SAP OSS Note 83000

ABOUT STORAGETEK®

Storage Technology Corporation (NYSE: STK), a $2 billion worldwide company with headquarters in Louisville, CO, has been delivering a broad range of storage management solutions designed for IT professionals for over 30 years. StorageTek offers solutions that are easy to manage, integrate well with existing infrastructures and allow universal access to data across servers, media types and storage networks. StorageTek's practical and safe storage solutions for tape automation, disk storage systems and storage integration, coupled with a global services network, provide IT professionals with confidence and know-how to manage their entire storage management ecosystem today and in the future.

StorageTek products are available through a worldwide network. For more information, visit www.storagetek.com, or call 1.800.275.4785 or 01.303.673.2800.

WORLD HEADQUARTERS

Storage Technology Corporation
One StorageTek Drive
Louisville, Colorado 80028 USA
1.800.877.9220 or 01.303.673.5151

MO 9254 A 5/04