Redbooks Paper

Frank Kyne
Glenn McGeoch

Achieving the Highest Levels of Parallel Sysplex Availability for DB2

Abstract

Maintaining high availability in a Parallel Sysplex® requires planning and effort on the part of many individuals in multiple disciplines. DB2® is just one component of a highly available environment.

This Redpaper provides additional information to the *Parallel Sysplex Availability Checklist*, SG24-6061 Redbook. It focuses on configuring DB2 to take advantage of Parallel Sysplex and to maintain high availability. We also include recommendations for maintaining high availability.

We discuss the following topics:

- Introduction and overview
- DB2 subsystem configuration
- DB2 data sharing configuration
- Database design and configuration
- Application design introduction
- End-user connectivity
- Operations and procedures
- Supporting material

Introduction and overview

In this section we provide an overview of the availability concepts with regards to DB2 and Parallel Sysplex. We discuss the following topics:

- What is availability for DB2?
- Components of an available DB2 environment
- Reasons for unavailability
- Configuring for high availability

What is availability for DB2?

When we talk about DB2 availability we need to understand that there are many components to availability. The following list contains some of the components:

- DB2 system software
- Operating system software
- Transaction manager software, such as CICS® or IMS™
- End-user application software
- Coupling Facility Control Code, if running in a Parallel Sysplex
- Central processor hardware
- Coupling Facility hardware, if running in a Parallel Sysplex
- Network software and hardware

When an end user calls and says that their DB2 application is not available, it may be that DB2 is available, but one or more of the other components are not available.

- In the case where the DB2 system is available but some other component is preventing access to DB2, the DB2 support staff needs to work with the support staff for the failing component to resolve the problem. For an application to be truly available, all of its components must be available. Plan to configure your DB2 systems or Parallel Sysplex to obtain the level of availability needed in your environment. Later in this chapter we discuss how to configure your DB2 environment for high availability.

Availability is often measured with regards to the end user. If their application is available then the system must be available.

- The required availability level is often stated in a service level agreement (SLA), between the end-user department and the provider of the service. If the availability level specified in the SLA is met, the system is considered highly available. If high availability continues for 24 hours a day, 365 days a year, then the system is considered continuously available. More detailed definitions of availability levels for DB2 systems are described in the redbook *DB2 UDB for OS/390 and Continuous Availability,* SG24-5486.

Data sharing changes how many people view availability. Since a data sharing group, to an end user, is viewed as one large DB2 subsystem, continuous availability of the data sharing group is what matters.

- If one of the members is down, but the application can continue to run on another member, then the system is considered available. Whether your application can run on another DB2 member may depend on whether there are any affinities between the application and the member on which the application usually runs. For example, if the application also accesses some non-DB2 resources, which are not available on the LPARs where the other DB2 members run, then your application will not be available when its usual DB2 execution environment is not available.

DB2 data sharing allows you to mask both planned and unplanned outages.

- In later sections of this chapter we define planned and unplanned outages and discuss how to configure your Parallel Sysplex to maximize availability during both types of outages.

  As a first recommendation we highly recommend you have a well defined problem management process in place to analyze the symptoms of a problem and determine the root cause.

Components of an available DB2 environment

High availability has long been one of the reasons for customers to migrate from other databases or file systems to DB2 on OS/390® or on z/OS®. A DB2 environment could consist of a single DB2 subsystem or a number of subsystems operating independently or together as a data sharing group.

Regardless of the definition of a DB2 environment, there are a number of components that are common to all highly available DB2 environments.

Table 1 Components of an available DB2 environment

Component	Description
Log data sets	The DB2 logs are a critical component of DB2 that are used to recover data when an outage or application error occurs.
Image copies	The backups of system and application data allow for speedier recovery when required, thus minimizing the amount of time that a DB2 subsystem or application is unavailable.
Subsystem parameters	There are a number of subsystem parameters (ZPARMs) that control how each DB2 subsystem operates. Specifying the correct values for your environment can greatly improve system availability. Many ZPARMs can be changed without bringing DB2 down, thus improving availability.
IRLM	DB2's Internal Resource Lock Manager (IRLM) is both a separate subsystem and a critical component of DB2. IRLM manages the locking mechanisms that serialize access to your DB2 data. IRLM ensures data integrity when there are many tasks that are attempting to access the same data.
Duplexing	Whether it is dual logging and dual image copies in any DB2 environment or duplexed group buffer pools in a data sharing environment, duplexing is an important concept for all highly-available DB2 environments.
Application management	Each application needs to be designed for high availability. Applications should be restartable and designed for high concurrency with other applications. For details on techniques for coding highly available applications refer to the IBM® Redbook *Parallel Sysplex Application Considerations*, SG24-6523.

Component	Description
Thread management	Since many tasks, running as DB2 applications, utilities or commands, may simultaneously request resources from DB2 and the operating system, a task-management system must be in place to ensure that one or more tasks do not monopolize the system resources. Each identifiable task in DB2 is known as a thread. There are a number of system parameters and application coding techniques that you can use to manage DB2 threads. The system parameters are discussed in this book. The application coding techniques are discussed in the IBM Redbook *Parallel Sysplex Application Considerations*, SG24-6523.
Backup and recovery procedures	Standardized backup and recovery procedures are critical to ensure that data is recovered quickly when required, thus minimizing the time that the data is unavailable.
Performance monitoring	Monitor DB2 systems and applications on a regular basis to ensure high performance and availability. In , “Operations and procedures” on page 36 we discuss some techniques to monitor for performance and concurrency problems.

Reasons for unavailability

Ideally you would like your DB2 environment continuously available 24 hours a day, 365 days a year. Unfortunately outages do occur, but with proper planning you can minimize those outages and mask those outages to the end user. We will discuss methods to minimize outages in subsequent sections.

Here we identify the major causes for planned and unplanned outages.

- Following are some of the most common reasons for planned outages:
  - Applying software maintenance
  - Migrating to a new version of DB2 or other system software
  - Running REORG or other utilities
  - Implementing a new version of an application
- Some of the most common reasons for unplanned outages are:
  - Hardware problems
  - Operating system software problems
  - DB2 software problems
  - Application problems

Although it is possible that you may experience an outage for any or all of the above reasons, proper planning for and exploitation of the availability features of DB2, can help you to minimize the outage or prevent the end user from even seeing the outage.

Configuring for high availability

We provided an overview of the components of a highly available DB2 environment and the reasons why you may experience unavailability. In this section, we provide an overview of

what you need to do to configure your DB2 environment for high availability. Each of these topics are further discussed in subsequent sections of this chapter.

DB2 subsystem configuration

DB2 is designed to incorporate many availability options, even when running as a single subsystem. Following is a brief list of some of the options you can configure:

- Changing checkpoint frequency to reduce restart and recovery times
- Implementing dual logging
- Changing ZPARMs dynamically
- Using features that allow for faster restart processing
- Developing procedures for taking sysplex-wide dumps when required
- Running DB2PLI8 to provide IBM support with information about your databases

See, “DB2 subsystem configuration” on page 6 for details on configuring a DB2 subsystem for high availability.

DB2 data sharing configuration

Implementing DB2 data sharing can provide benefits with regards to scalability and availability by taking advantage of the following features:

- Multiple DB2 subsystems that share the same DB2 catalog
- Splitting workloads across multiple DB2 subsystems
- Rolling maintenance across subsystems without bringing the entire group down
- Migrating to a new version of DB2 one member at a time

See, “DB2 data sharing configuration” on page 10 for details on configuring a DB2 data-sharing environment for high availability.

Database design and configuration

How you design your databases has a significant impact on the availability of your application data. Following is a list of some of the areas over which you have control:

- Use of partitioned tables and table spaces to allow for parallel operations
- Development of a buffer pool strategy to better control access characteristics of your data
- Choice of data set size and control interval (CI) size
- Changing some of your database attributes dynamically through online schema change
- Backup and recovery procedures
- Determining when and how to reorganize data

See, “Database design and configuration” on page 17 for details on designing and configuring your databases for high availability.

Application design

A poorly tuned application can be, perhaps, the biggest inhibitor to high availability. Following is a list areas you should concentrate on during the application design, development, and implementation process:

- Design applications with manageable units of work to enable fast restart and recovery
- Design for concurrency with other applications
- Use the power of DB2 by coding efficient SQL instead of program controlled access logic
- Develop well managed testing methodologies and application promotion procedures
- Develop procedures for application data recovery

See, “Application design introduction” on page 28 for details on designing, developing, and implementing your applications for high availability.

End-user connectivity to the single image

You can configure connectivity to a DB2 subsystem or to a data sharing group for maximum availability. Following are some of the alternative connectivity configurations:

- Direct access to a single member of the group
- Direct access to the data sharing group
- Use DB2 Connect™ to access remote data-sharing groups
- Use the Group Attach facility to connect to a data sharing group

See, “End-user connectivity” on page 33 for details regarding options for connecting to a DB2 subsystem or data sharing group.

Operations and procedures

Although there are many features of DB2 that provide for high availability, you need to develop some procedures to take best advantage of these features. Develop and test the following operational procedures for your DB2 environment:

- Tracking and applying DB2 maintenance
- Changing DB2 subsystem parameters
- Migrating to a new version of DB2
- Monitoring a data sharing group
- Maintaining DB2 statistics

See, “Operations and procedures” on page 36 for more details on developing procedures that will contribute to maintaining a highly available DB2 environment.

DB2 subsystem configuration

DB2 contains many features that provide for high availability, even when running as a single subsystem. In this section we discuss the features that provide for high availability within DB2, and we discuss how to configure your DB2 subsystems to take advantage of these features. We discuss the following topics:

- Checkpoint frequency recommendations
- Dual logging recommendations
- Dynamic ZPARMs
- Restart options
- Dump options
- DB2PLI8

Checkpoint frequency recommendations

- ❑ We suggest that you control how frequently system checkpoints are taken.

  Prior to DB2 Version 7 you could only control the checkpoint frequency by specifying the number of log records written by DB2 between checkpoints. Starting with Version 7 you can alternatively specify the checkpoint frequency in terms of the number of minutes between checkpoints. The checkpoint frequency and frequency type (log records or minutes) are set on panel DSNTIPL at installation time. Refer to *DB2 UDB for z/OS V8 Installation Guide,* GC18-7418 for instructions on setting these values.

- ❑ Consider the trade-offs between the overhead needed for frequent checkpoints and the time needed to restart a DB2 subsystem. The checkpoint frequency can impact how long it takes to restart DB2.

If your primary concern is DB2 restart time, we recommend you use a use a checkpoint frequency between 50,000 and 1,000,000 log records. Otherwise, use a checkpoint frequency of 2 to 5 minutes.

Dual logging recommendations

- We still strongly recommend that you maintain dual copies of your active logs, although advancements in the reliability and performance of storage devices in recent years have greatly minimized the number of outages on DASD and tape devices. With dual logging, if you do experience a failure on an active log data set, DB2 can continue to operate by using the second copy.
- We recommend that you place copy 1 and copy 2 of each active log data set on different DASD subsystems to ensure that both copies are not impacted by a DASD subsystem failure. Storage Management Subsystems (SMS) can also assist in controlling placement of your log data sets through use of the data set separation feature. This feature gives SMS the responsibility of ensuring that the named data sets are allocated in different physical control units.
- If you have a DB2 environment with a heavy volume of updates, or if you have the need to recover multiple DB2 objects quickly, then we recommend that you define each active log as an extended format data set. We also recommend that you enable VSAM input/output (I/O) striping for your active logs.
- You should size your active logs to minimize the number of times you need to access archive log data sets.

  For example, if you take daily image copies of your application data, and you have a need to recover data quickly, size your active logs to hold a full day of data. Otherwise, you may find that your recovery jobs are frequently requesting archive log data sets. If the archive log data sets are on tape this could slow your recovery process. If you have to access archive log data sets on a regular basis, write the archive log to DASD, which is managed by SMS. You can also use a tool, such as the DB2 Archive Log Accelerator for z/OS, to reduce the overhead of maintaining large archive log data sets and to enable faster writing and reading of archive logs.

See the instructions for installation panel DSNTIPH in *DB2 UDB for z/OS V8 Installation Guide,* GC18-7418 for instructions on activating dual logging.

Refer to the redbook *DB2 for z/OS and OS/390 Version 7 Performance Topics,* SG24-6129, for instructions on enabling VSAM I/O striping.

Refer to the redbook *DB2 for z/OS and OS/390 Version 7 Selected Performance Topics,* SG24-6894 for possible performance enhancements when using FICON® and striping for DB2 log data sets.

Dynamic ZPARMs

DB2 Version 7 provided the capability to dynamically change many system parameters (ZPARMs) while keeping the DB2 system up.

DB2 Version 8 supports even more dynamically changeable ZPARMs. You can dynamically change ZPARMs using the -SET SYSPARM command. See, “Changing system parameters” on page 37 for the procedures on how to dynamically change system parameters.

Restart options

There were a number of enhancements to DB2 restart processing in recent years. We provide an overview of those enhancements here and provide some recommendations in “DB2 restart and recovery recommendations” on page 16.

Normal restart

During a normal DB2 restart, with no special options, DB2 uses the log and BSDS to determine what to recover. For systems that do not share data, the restart process consists of the following four phases:

- Phase 1: Log initialization
- Phase 2: Current status rebuild
- Phase 3: Forward log recovery
- Phase 4: Backward log recovery

These four phases are described in detail in *DB2 UDB for z/OS Version 8 Administration Guide,* SC18-7413.

Restart Light

Restart Light allows you to restart the DB2 member on another LPAR purely for the purpose of freeing the retained locks, after which the member is terminated. It was introduced by DB2 Version 7.

This type of restart is useful in data sharing if you have an LPAR failure and the DB2 member that was running in that LPAR has retained locks in the Coupling Facility. The “light” restart requires a much smaller amount of storage than is usually required to restart a DB2 member. This means that you can restart the member on a different LPAR that would normally not be able to handle that DB2 member.

The purpose of the restart is to free the retained locks. Once the locks are freed, DB2 automatically terminates the member. You can bring that member up later on the original LPAR once it is available again.

Postponed recovery

We recommend that you design your applications with frequent commit points to minimize the amount of backout processing required at restart. For example, if you have a long running unit of work that has not completed processing when one of your DB2 systems terminates, it could take a long time to restart that DB2 system. This is because DB2 has to back out all the uncommitted changes for that unit of work.

Starting with DB2 Version 6, you can limit the amount of backward log processing that occurs at restart, thus preventing long running URs from delaying the availability of a DB2 system.

- The LIMIT BACKOUT ZPARM on installation panel DSNTIPL controls whether you want to postpone some backward log processing.
- The BACKOUT DURATION ZPARM on the same panel acts as a multiplier for your checkpoint interval.
- If the LIMIT BACKOUT ZPARM is set to YES or AUTO, then at restart DB2 limits the number of log records processed by any unit of work to the value of BACKOUT DURATION multiplied by the checkpoint frequency.
- If you specify AUTO for LIMIT BACKOUT, then DB2 postpones the recovery for units of work that meet the BACKOUT DURATION criteria. Then DB2 automatically recovers those units of work once the restart completes.

- If you specify YES for LIMIT BACKOUT, then DB2 will not attempt to recover those units of work that qualify for postponed recovery until you issue the RECOVER POSTPONED command.

See *"Active log data set parameters: DSNTIPL*" in *DB2 UDB for z/OS V8 Installation Guide,* GC18-7418 for instructions on setting these ZPARMs to take advantage of postponed recovery.

Automatic restart

If you are running DB2 in a Parallel Sysplex, you can have the z/OS Automatic Restart Manager (ARM) automatically restart DB2 or IRLM after a failure.

When DB2 or IRLM stops abnormally, z/OS:

1. Determines whether z/OS failed too
2. Determines where DB2 or IRLM should be restarted
3. Restarts DB2 or IRLM

You control how automatic restart works by using automatic restart policies available with z/OS Automatic Restart Manager (ARM).

For more details on ARM, see *MVS Setting Up a Sysplex*, SA22-7625 and the IBM Redpiece REDP0173, at the following Web site:

http://www.redbooks.ibm.com

Conditional restart recommendations

- You only perform a conditional restart if there are no other options available that will successfully restart your DB2 system.

  If you are running DB2 in a Parallel Sysplex note that conditional restarts are member specific. There is no concept of a group-wide conditional restart. Therefore there is a high likelihood that you will have data integrity problems after issuing the conditional restart, unless you have system affinities that eliminate the possibility that data could be shared across members.

- Before issuing a conditional restart you read the topic *"Restarting with conditions"* in *DB2 UDB for z/OS Version 8 Administration Guide,* SC18-7413.

Dump options

For customers in data sharing, it is often important to aid problem resolution by taking a sysplex-wide dump of all pertinent address spaces, usually DBM1, MSTR, and IRLM.

The three address spaces listed above are suspended while they are dumped. The other address spaces are affected only if they require services from the ones being dumped. The time to dump all members should be very similar to the time required to collect a dump on a single member. This is because the dumps are taken concurrently on all members.

With the level of IRLM shipped with DB2 Version 8 (IRLM 2.2), all locks are stored above the 2 GB bar.

Dump option recommendations

- For releases of DB2, prior to Version 8, we recommend that you specify PC=YES in the IRLM startup procedure. Specify QUIESCE=NO when taking an SVC dump.
  - Specifying PC=YES moves the IRLM locks from ECSA into IRLM's private above-the-line storage.

– Specifying QUIESCE=NO ensures that the system is not made non-dispatchable while common storage is being dumped.

DB2PLI8

DB2PLI8 is an IBM-supplied program that allows you to capture statistics about a DB2 system, its objects and applications, and send that information to the IBM DB2 for z/OS Support Center for assistance in resolving performance problems. You can access DB2PLI8 from the IBM FTP site by entering the following information:

```
Site: testcase.software.ibm.com
Directory: s390/fromibm/db2
Files to download: DB2PLI8.LKEDLE
                   DB2PLI8.JCL71
```

The Job Control Language (JCL) file contains instructions for installation and execution of DB2PLI8.

This program can prove extremely valuable when you have a performance problem and the support center needs to duplicate it in a lab environment. DB2PLI8 will capture DDL, statistics, and EXPLAIN data, and will allow the support center to build a similar environment to yours for testing purposes.

DB2 data sharing configuration

One of the most significant enhancements to DB2 availability was the introduction of DB2 data sharing in DB2 for MVS/ESA™ V4. This section describes data sharing and highlights its availability features. We discuss the following topics:

- DB2 data sharing overview
- Advantages of DB2 data sharing
- A history of DB2 data sharing enhancements
- Data sharing configuration recommendations

DB2 data sharing overview

DB2 data sharing was introduced with DB2 Version 4. Data sharing runs in a Parallel Sysplex environment, which is a group of z/OS systems that communicate and cooperate with one another using specialized hardware and software. A Parallel Sysplex contains, along with the processors that are running the DB2 subsystems, one or more processors that act as Coupling Facilities. Figure 1 on page 11 shows the basic components of a DB2 data sharing environment running in a Parallel Sysplex.

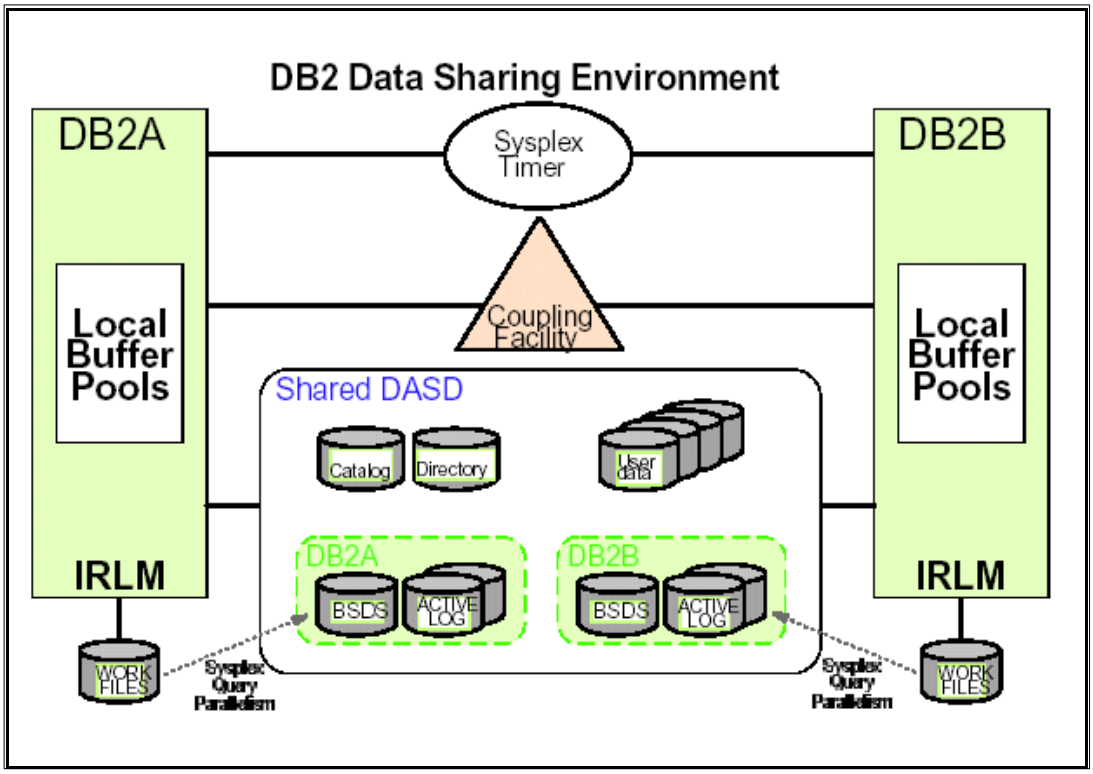

Figure 1 The data sharing environment

The Coupling Facilities run specialized control code that coordinates activity across the multiple subsystems (called members) of the data sharing group. The Coupling Facilities contain three major structures that are utilized by DB2, as shown in Figure 2 on page 12:

- Group Buffer Pools - These are cache structures that contain data and index pages for DB2 objects that are being accessed across the data sharing group. The group buffer pools, which usually have a one-to-one correspondence with the local buffer pools on each member, monitor which members are accessing which DB2 data sets and thereby keep track of any data being updated by one member that is subsequently accessed by a second member. This situation is called inter-DB2 read/write interest in a data set, and it causes the pages being updated to be written from the local buffer pools to the respective group buffer pools. The member that is reading the data then reads the data from the group buffer pools, and ensures that the most recent data is read and that data integrity is maintained.
- Lock structure - The lock structure ensures that the same data is not updated by multiple members at the same time. It does this by ensuring that data being updated by one member cannot be updated by another member until the first member commits or rolls back, thus releasing the locks that it held.
- Shared Communications Area (SCA) - The SCA is used to coordinate recovery and startup across the group. The SCA contains the following items:
  - DB2 member names
  - BSDS names for each member
  - Database exception status conditions for DB2 objects and DB2 members
  - Recovery information, such as log data set names and information about in doubt transactions

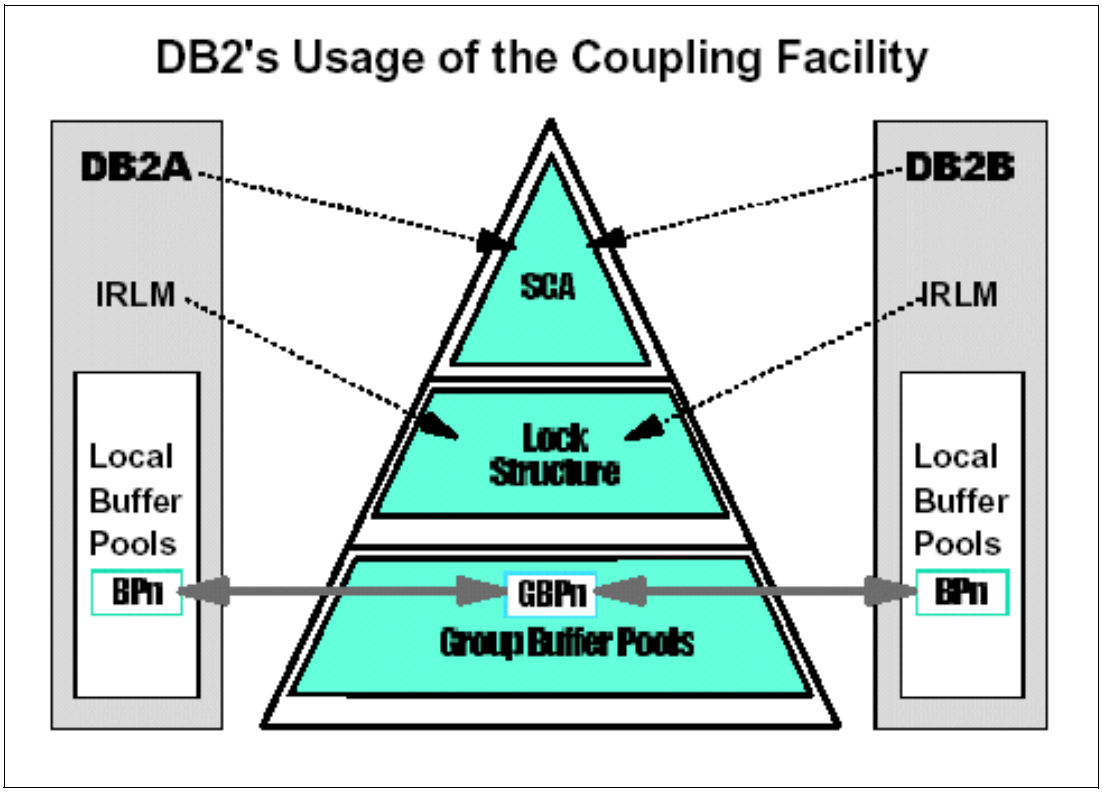

Figure 2 DB2's usage of the Coupling Facility

The Group Buffer Pools, Lock structures, and SCA reside in one or more Coupling Facilities. To use DB2 data sharing, you must have at least two Coupling Facilities with failover capability from each to the other. Later on in this section we recommend how you should configure your Coupling Facilities for maximum DB2 availability.

In addition to the Coupling Facilities, a Parallel Sysplex includes one or more sysplex timers, which synchronize the system clocks of all the members of the data sharing group. Since multiple members can update data from the same set of tables, and since any one of those members can issue recovery, the log data on each member uses a common log record sequence number (LRSN) that is derived from the store clock timestamp and synchronized across all members of the group by the sysplex timer.

Advantages of DB2 data sharing

One of the advantages of DB2 data sharing is to provide additional capacity without having to change applications or copy data. If you exhausted the resources of a single DB2 subsystem, without data sharing you would need to split your data across multiple subsystems and provide logic to determine which subsystems to access for which data. With data sharing, you can add another member to the group and use the same applications and data.

Even if you do not have capacity issues that preclude you from running your entire DB2 workload on one subsystem, still consider DB2 data sharing for the availability improvements provided by a properly configured Parallel Sysplex environment.

Data sharing provides the following availability features that are not provided when running a single DB2 subsystem:

- Multiple DB2 subsystems, called members of a data sharing group, that can provide failover when one of the subsystems is unavailable because of an unplanned outage
- Distribution of your workload over multiple members to reduce the CPU workload on any one member, without having to maintain multiple copies of the data
- Application of maintenance to one of the members without an outage to your production applications, which can continue to access DB2 data from the other members

  The maintenance can then be subsequently rolled across all members in a staggered manner.
- Migration to a new version of DB2 without having to bring the whole data sharing group down

  DB2 supports the coexistence of two versions within the same data sharing group. This means that you can migrate one member initially, run in coexistence mode until you are satisfied that the new DB2 code is working properly, then migrate each additional member one at a time. The only complete outage is during the time that CATMAINT is run.

A history of DB2 data sharing enhancements

As we previously discussed, data sharing became available with DB2 for OS/390 Version 4. There were enhancements to the data sharing functionality with each subsequent version of DB2. In this section we highlight the enhancements available with each version of DB2. Data sharing first became available in DB2 V4. DB2 Version 8 is the current version.

DB2 Version 4 features

- Group Buffer Pool (GBP) rebuild - This is accomplished by stopping all DB2 members, by altering the local buffer pool sizes to 0 or by stopping all objects using the GBP.
- GBP recovery - This is accomplished by issuing START DATABASE commands against all objects in GRECP status.

DB2 Version 5 enhancements

- GBP duplexing - DB2 writes changed pages to both a primary and secondary GBP. If connectivity to the primary GBP is lost, DB2 automatically switches over to the secondary GBP.
- GBP rebuild - If you are not using GBP duplexing, DB2 can automatically rebuild the GBP following a CF or CF link failure.
- Automatic GBP recovery - If you defined the GBP with AUTOREC(YES), DB2 can automatically recover the objects in GRECP status. You no longer have to issue START DATABASE commands.
- Improved restart performance - Forward log recovery ignores log records for disposition-complete URs that apply to GBP-dependent page sets.
- Disaster recovery enhancement - New ENDLRSN option on the CRCR record provides for a less disruptive quiesce point than issuing ARC LOG SCOPE(QUIESCE) or ARC LOG SCOPE(GROUP) without ENDLRSN.
- Selective partition locking (SPL) - This can reduce the amount of locking occurs by locking only the database partitions that are being used, instead of the entire table space.
- Inter-DB2 read/write tracking - Improvements were made to how DB2 tracks inter-DB2 read/write interest to allow GBP-dependency to be removed sooner.

- Space map page enhancement - The MEMBER CLUSTER option of the CREATE TABLESPACE command causes each member to acquire its' own space map page, thus reducing the chance that there will be hot space map pages during insert processing.
- IMMEDWRITE bind option - This option deals with order-dependent transactions where an update on one member may not be seen on another member if the transaction on the first member spawns a transaction that runs on the second member before completing phase 2 of commit processing.

  Specifying IMMEDWRITE(YES) on the bind statement forces updated pages that are GBP-dependent to be written immediately after the buffer update is complete. This option ensures that the spawned transaction sees the data, but this may also impact performance. IMMEDWRITE(NO) is the default.
- Changes to traces and DISPLAY GROUPBUFFERPOOL output - These changes help you to better monitor your data sharing environment.

DB2 Version 6 enhancements

- GBP caching changes - New caching options at the page set level as well as the GBP level
- At the page set level GBPCACHE NONE causes changed pages to be written directly to DASD, bypassing the GBP, while GBPCACHE SYSTEM writes only space map pages to the GBP.
- At the GBP level, GBPCACHE(YES) indicates that caching to this GBP should use the specifications for the page set, while GBPCACHE(NO) causes all changed pages to be written directly to DASD, regardless of what the values are at the page set level.
- Support of 8 KB and 16 KB page sizes - Since DB2 Version 6 introduced 8 KB and 16 KB page sizes, data sharing was also enhanced to allow for 8 KB and 16 KB group buffer pools.
- Space map update tracking - The TRACKMOD option of the CREATE TABLESPACE and CREATE INDEX statements controls whether DB2 tracks changed pages in the space map page of a table or index. Specifying YES, which is the default, causes DB2 to track the changes, thus improving the performance of incremental image copies. Specifying NO tells DB2 not to track the changes, which can result in improved concurrency at the cost of elongated incremental image copies.
- Fast log apply - The Fast Log Apply processing can reduce the time required for GRECP and LPL recovery.
- Changes to IMMEDWRITE behavior - Immediate Write processing was enhanced to be able to specify the behavior at the subsystem level or at the plan/package level. Additionally, IMMEDWRITE(PH1) was added to force updated pages that are GBP-dependent to be written at or before phase 1 of commit. PH1 does not incur the performance penalty of YES, and satisfies most customers' needs for order-dependent processing.

DB2 Version 7 enhancements

- Restart Light - The LIGHT(YES) option of the START DB2 command can be used to start a minimal DB2 footprint for the purpose of releasing the retained locks held by that member. This option allows you to bring up a DB2 member on a different processor without the need to have a large storage footprint on that processor. Once the member is restarted and the retained locks are freed, the member is automatically brought down.
- Persistent structure size changes - In earlier releases of DB2, any changes you made to structure sizes using the SETXCF START,ALTER command might have been lost when

you rebuilt a structure and recycled DB2. Now you can allow changes in structure size to persist when you rebuild or reallocate a structure.

- IMMEDWRITE enhancement - Prior to Version 7 the value of IMMEDWRITE at the system level was established by editing macro DSN6GRP. This enhancement adds the option to installation panel DSNTIP4 and reflects the current setting in catalog tables SYSPACKAGE and SYSPLAN in column IMMEDWRITE.

DB2 Version 8 enhancements

- CF lock propagation reduction - This enhancement remaps IX parent L-locks from XES-X to XES-S. Data sharing locking performance benefits because this allows IX and IS parent global L-locks to be granted without invoking global lock contention processing to determine that the new IX or IS lock is compatible with existing IX or IS locks.
- CF request batching - This enhancement utilizes new functionality in z/OS 1.4 and CF level 12 to enable DB2 to register and write multiple pages to a group buffer pool and to read multiple pages from a group buffer pool for castout processing. This reduces the amount of traffic to and from the Coupling Facility for writes to group buffer pools and for reads for castout processing. This reduces the data sharing overhead for most workloads.
- Improved LPL recovery - Prior to V8, you had to manually recover pages that DB2 put into the logical page list (LPL). DB2 Version 8 automatically attempts to recover LPL pages as they go into LPL, when it determines the recovery will probably succeed. Currently LPL recovery requires exclusive access to the table space page set. Version 8 introduces a new serialization mechanism whereby LPL recovery is much less disruptive.
- Restart Light enhancements - If indoubt units of recovery (UR) exist at the end of restart recovery, DB2 now remains running so that the indoubt URs are resolved. After all the indoubt URs are resolved, the DB2 member that is running in LIGHT(YES) mode shuts down and can be restarted normally.
- Change to the default for IMMEDWRITE bind option - This enhancement changes the default processing to write changed pages during phase 1 of commit processing. DB2 will no longer write changed pages during phase 2 of commit processing.
- Change to -DISPLAY GROUPBUFFERPOOL output - Currently, the CF level displayed by the -DISPLAY GROUPBUFFERPOOL command may be lower than the actual CF level as displayed by a D CF command. The -DISPLAY GROUPBUFFERPOOL is now enhanced to display both the operational CF level as before, and also the actual CF level. (The operational CF level indicates the capabilities of the CF from DB2's perspective. The actual CF level is the microcode level as displayed by the D CF command.)

Data sharing configuration recommendations

Data sharing has been available since DB2 Version 4 and is in use by many DB2 customers around the world. These customers implemented data sharing because they needed the high availability and extra capacity that data sharing provides. In order to gain these benefits, it is imperative that you properly configure your data sharing environment. In this section we provide some recommendations for configuring your DB2 data sharing environment for the highest possible availability.

Parallel Sysplex configuration recommendations

- We recommend that non-duplexed SCA and lock structures should be in a Coupling Facility (CF) that is failure-isolated from your DB2 subsystems. If one of your CFs is an Internal Coupling Facility (ICF), we recommend that the ICF not be used for the lock and SCA structures unless those structures are duplexed. Do not place your lock and SCA structures in the same CF as GBP0, which contains cached information for the DB2 catalog and directory.

- We recommend that you do not duplex your lock and SCA structures unless your availability needs require it. Duplexing the lock and SCA structures gives you faster recovery times because the structures can be rebuilt very quickly from in-memory data; consequently, this can have a significant impact on performance.
- We recommend that you duplex your group buffer pools. The writes to the secondary buffer pools occur after the primary writes are complete, so there is little or no impact on performance. Also, there is a noticeable improvement in availability compared to having to do GBP recovery.
- If you only have two CFs, make sure that you have enough memory in each CF to hold the lock, SCA, and GBP structures.

  For example, if you have the lock and SCA structures in CF1 and the GBPs in CF2, and you experience a failure on CF1, you will need enough memory on CF2 to allow the lock and SCA structures to be rebuilt there. We often recommend that you install three CFs to allow for continuous GBP duplexing in the event of the failure of one of the CFs.

DB2 restart and recovery recommendations

- Consider using Automatic Restart Manager (ARM) in conjunction with the RETAINED LOCK TIMEOUT option (ZPARM RETLWAIT) of installation panel DSNTIPI to ensure that retained locks are released in a timely manner. Please see *DB2 UDB for z/OS Version 8 Data Sharing: Planning and Administration,* SC18-7417 for more information on creating an ARM policy.
- If you intend to use Restart Light at some point, plan for it ahead of time by creating an ARM policy for the DB2 group that specifies LIGHT(YES) within the RESTART_METHOD(SYSTERM) keyword for the DB2 element name.

  For example:

  ```
  RESTART_METHOD(SYSTERM,STC,'cmdprfx STA DB2,LIGHT(YES)')
  ```

- If you require faster restarts, we recommend that you take more frequent checkpoints. See, “Checkpoint frequency recommendations” on page 6 for recommendations on checkpoint frequency. You can also enable fast log apply to speed up restarts if you have adequate DBM1 storage.
- Specify AUTOREC(YES) for each of your group buffer pools to enable automatic recovery of your GBPs. You do not need to specify AUTOREC(YES) for any GBPs that are defined with GBPCACHE(NO) since data is written directly to DASD.
- Data sharing generally requires more disk space for the active log, and you need to archive the logs more frequently. See *DB2 UDB for z/OS Version 8 Data Sharing: Planning and Administration,* SC18-7417 for more information.

Workload balancing recommendations

- We recommend that you use the Workload Manager (WLM) component of z/OS to balance work across the members of your data sharing group. DB2 works closely with WLM to ensure that incoming requests are optimally balanced across the members of a data sharing group. WLM is required if you plan to implement stored procedures.
- Consider using sysplex query parallelism for decision support queries or if you have queries that perform complex data analysis on large amounts of data. Several members of a data sharing group can be used to split the workload for a single query, amongst multiple processors, to reduce the elapsed time of the query. Be aware that sysplex query parallelism can be resource intensive. Therefore, you should use it in conjunction with WLM to ensure that these types of queries do not monopolize CPU resources.

Version migrations recommendations

- ❑ If you plan to take advantage of the version coexistence capabilities of a data sharing group, we recommend that you set the ABIND ZPARM to a value of COEXIST.

  The ABIND ZPARM controls automatic rebind operations in a data sharing coexistence environment when a plan or package that was previously bound on the newer version of DB2 is now running on the older version of DB2. If you specify ABIND YES while in coexistence mode, your plans and packages are automatically rebound every time they run on a different version from the version on which they were bound. So you can see considerable rebind activity that could have a negative impact on performance.

  Once all members of a data sharing group are migrated to the new version, a value of COEXIST is treated as if it was YES. If you do not want invalid packages and plans to be automatically rebound once you complete your migration, then set the ABIND ZPARM to NO.

Database design and configuration

In addition to configuring your DB2 systems for high availability, it is important to implement procedures to achieve high availability for your application data. There are a number of techniques available to achieve high availability throughout the database design, development, and implementation stages. We discuss some of those techniques in the sections that follow. We discuss the following topics:

- Database design
- Backup and recovery
- Reorganization

Database design

DB2 for z/OS provides quite a bit of flexibility when it comes to database design. In this section we talk about some of the areas you can control, and their impact on database availability.

Partitioned tables and table spaces

In DB2 Version 7, when you define a table space you have four options for the type of table space:

- simple
- segmented
- partitioned
- large object (LOB)

When you define a partitioned table space, by specifying the NUMPARTS keyword on the CREATE TABLESPACE statement, you create a separate physical data set for each partition. As a result, each partition can be processed separately for many database operations. You can REORG a partition without impacting the other partitions. You can run SQL or utilities on multiple partitions in parallel, thus reducing the elapsed time of jobs and making the data more available for other operations.

Prior to DB2 Version 8, if you wanted to create a secondary index on a partitioned table space, the secondary index—also called a non-partitioned index (NPI)—was created as a single physical data set. You tend to lose some of the advantages of partitioned table spaces when you have NPIs defined. Following are some of the challenges of managing NPIs:

- NPIs increase intersystem read-write interest in data sharing.

- The sheer size of NPIs over large partitioned tables makes their management as a single object difficult.
- Recovery of an NPI must be done at the entire index level.
- Partition-level operations, such as deleting all the data from a partition, are less clean with NPIs.
- Online REORG of a single partition requires a BUILD2 phase where the entire object is unavailable.
- Parallel LOAD PART jobs create contention on the NPI.

DB2 Version 8 introduces table-control partitioning and data-partitioned secondary indexes (DPSIs). Table-controlled partitioning eliminates the dependency of partitioning on a partitioning and clustering index, as shown in Figure 3.

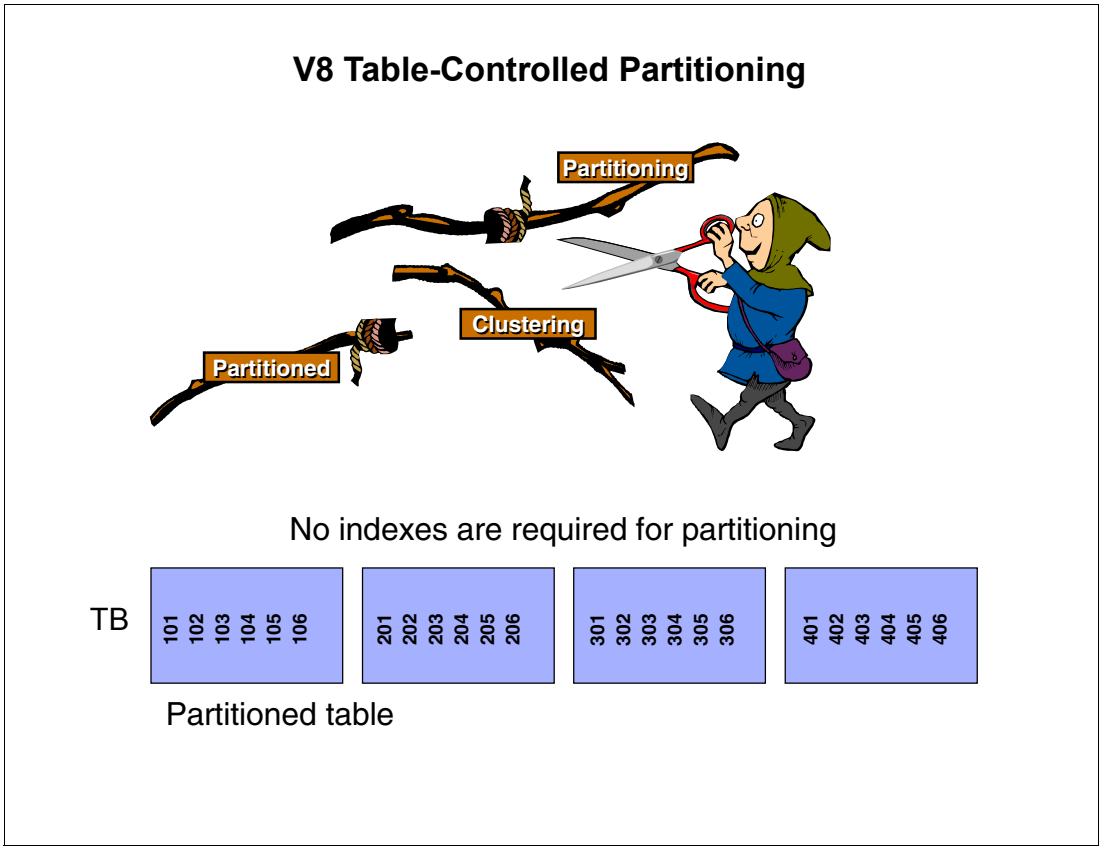

Figure 3 New partitioning and clustering concepts

By using table-controlled partitioning, clustering, being partitioned, and being the partitioning index are now separate concepts. When using table-controlled partitioning, a table does not require a partitioning index as the partitioning occurs, based on the PARTITION BY clause in the CREATE TABLE statement. Since the partitioning index is no longer required, you can use another index as a clustering index.

With DPSIs you can now physically partition your secondary indexes and eliminate many of the issues that NPIs introduced.

See the redbook *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ...and More,* SG24-6079 for details on defining table-control partitioning and data-partitioned secondary indexes.

Partitioned tables and table spaces recommendations

We recommend the following for partitioned tables and table spaces:

- In DB2 Version 7, use partitioned table spaces
- In DB2 Version 8, use partitioned tables

Partitioning enables the capability to run multiple processes in parallel without incurring contention on resources. This reduces the elapsed time for each process and makes the data more available.

Buffer pool strategy

Buffer pools are areas of storage where DB2 temporarily stores index pages or data pages after they are read from DASD or before they are written to DASD. Buffer pool storage is either virtual storage (in the DBM1 address space) or real storage (in data spaces). Starting with DB2 V8, which eliminates the 2 GB constraint on the DBM1 address space, all buffer pools reside in virtual storage.

The role of the buffer pool during read operations is to provide a repository for index and data pages such that data can be read with a minimum amount of waiting for I/O operations. Depending on how you define your buffer, you could potentially have critical OLTP applications needing to perform unnecessary I/O operations to re-read data because the data that it previously read was paged out of a buffer pool by a query that read in too many pages from a data warehouse stored on DB2 for z/OS. Devising a buffer pool strategy that fits your environment can provide the benefits of improved response times, greater system throughput, reduced CPU consumption, reduced I/O activity and, for customers at V7 and below, virtual storage constraint relief.

Buffer pool recommendations

Buffer pool tuning is more of an art than a science. There are tools, such as IBM DB2 Buffer Pool Analyzer for z/OS, that can assist you by recommending optimal groupings for DB2 objects in buffer pools, but you need to know your data to arrive at the best strategy. We recommend that you use the following buffer pool strategy as a starting point and modify it for any special needs in your environment:

- Use BP0 for catalog and directory data and index entries only.

  You do not want to mix your application data in the same buffer pool with critical operational data that DB2 uses.

- Specify default buffer pools for data and indexes at install time on panel DSNTIP1.

  If you do not specify a default buffer pool for data and for indexes, any DB2 objects created without an explicitly specified buffer pool will use BP0 by default, which is not what you want to do.

- Separate your data and your indexes into different buffer pools.

  Depending on the type of read access used by a SQL statement, you can benefit from separating these types of objects.

  For example, a query that uses an access path of non-matching index scan must read all the index entries sequentially, but may only read a small number of data entries in random fashion. In this case it makes sense to allow DB2 to sequentially prefetch index pages into the index buffer pool for faster I/O against the index, rather than interleaving index and data pages.

- Separate further your data and index objects by type of access.

  Data that is highly used and is most often read or updated randomly, such as transaction data, has very different behavioral characteristics than data that is most often read or

updated sequentially, such as historical data. You may also want to consider different buffer pool thresholds for each of these buffer pools.

For example, buffer pools designed for query only workloads should have a high sequential steal threshold (VPSEQT at or near 100), while buffer pools used exclusively for transaction workloads should have a low value for VPSEQT, perhaps even zero to disable sequential prefetch.

- ❑ Separate large DB2 objects from small DB2 objects.

  Large DB2 tables that often require reading many rows from the table, or for which a small number of rows take up an entire page, can easily fill a buffer pool and cause pages from smaller DB2 objects to be paged out. To reduce the number of times DB2 has to re-read data from the smaller tables, we recommend that you use a separate buffer pool for your large DB2 tables. Also, if you have tables that require rows larger than 4 KB, create a separate buffer pool with a page size of 8 KB, 16 KB, or 32 KB to hold that data.

- ❑ Separate LOBs into their own buffer pool.

  Due to their size, LOBs require more I/O than any other type of DB2 object. LOBs should use a buffer pool with a deferred write threshold (DWQT) of zero to avoid massive writes at commit (for LOB tables defined as LOG NO) or at checkpoint (for LOB tables defined with LOG YES). Setting DWQT to zero causes these writes to happen continuously in the background.

Developing a buffer pool strategy that is aligned with the type of data access used in your environment can reduce the impact that any one DB2 object can have on other DB2 objects, thus making your application data more available. See *DB2 UDB for z/OS Version 8 Administration Guide,* SC18-7413 for some guidelines on tuning buffer pools.

Data set size

DB2 supports a data set size of up to 64 GB. The value is specified on the DSSIZE keyword of the CREATE TABLESPACE statement. DB2 also supports a keyword of LARGE, instead of DSSIZE, for backwards compatibility.

- ❑ We recommend that you specify DSSIZE for all new table spaces. You should also be aware that the data set size for a table space can not be altered once it is chosen.

  To change the data set size you need to drop and recreate the table space. This could result in a lengthy period of unavailability. Therefore we recommend that you choose a data set size carefully when designing your databases.

CI size

Prior to DB2 Version 8 the only CI size available for DB2 data sets was 4 KB. DB2 Version 8 introduces support for CI sizes of 8, 16, and 32 KB. This support is activated by setting ZPARM DSVCI on installation panel DSNTIP7 to YES. Turning on this ZPARM causes the control interval for the underlying VSAM data set to correspond to the page size of the buffer pool used for the table space. The new CI sizes reduce integrity exposures, and relieves some restrictions on concurrent copy (of 32 KB objects) and the use of striping (of objects with a page size of 4 KB).

- ❑ Use this enhancement to potentially reduce elapsed time for table space scans.

Online schema changes

Perhaps the biggest availability enhancement in DB2 Version 8 is the ability to make a number of changes to your database schema without having to drop and recreate objects.

- Prior to Version 8, if you created a table and at a later date decided that you needed to change the data type or length of a column, you had to execute the following steps in sequence:
  - Extract the DDL and unload the data.
  - Drop the table.
  - Create the table.
  - Create any indexes on the table.
  - Create any views on the table.
  - Grant privileges to the DB2 objects.
  - Rebind plans and packages that use the table.

  Once the table was dropped the data was unavailable until the whole process was complete.

- DB2 Version 8 allows you to make some changes to the database schema while maintaining the availability of the table throughout the change process.

  For example, if table T1 has a column named COL1 that is defined as CHAR(10), you could issue the following SQL statement to increase COL1 from 10 bytes to 20 bytes:

  ```
  ALTER TABLE T1 ALTER COLUMN COL1 SET DATA TYPE CHAR(20)
  ```

  A new “version” of the table is created and any existing rows retrieved are materialized in the new format. Any rows inserted or updated are saved with the new definition. Existing rows are not immediately converted to the new format, but will be when the object is reorganized.

  The table space is placed in advisory REORG pending state (AREO) because there is some performance degradation to convert the data in the existing rows to the new format when accessed. Once a REORG is completed, the data is converted to the new format and there is no performance impact.

  In addition to changing the length of a column, you can also add a column to an index, add a partition, drop the partitioning index, change the clustering index, among other changes, without making the changed object unavailable.

  Figure 4 on page 22 summarizes the Alter Data Types currently supported with DB2 for z/OS Version 8.

For more details on what changes can be made online and what restrictions apply, see *“Online schema changes”* in the redbook *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ...and More,* SG24-6079.

The introduction of online schema change has lessened the impact of your initial choice for database design. You can now size columns based on your current need, knowing that you can come back later and increase the size without having to drop the table.

Supported Alter Data Types

From Data Type	To Data Type
smallint	integer
smallint	float(1-21) or real
smallint	foat(22-53) or double
smallint	>= decimal(5,0)
integer	float(22-53) or double
integer	>=decimal(10,0)
float(1-21) or real	float(22-53) or double
<=decimal(7,s)	float(1-21) or real
<=decimal(15,s)	float(22-53) or double
decimal(p,s)	decimal(p+a,s+b)
char(n)	char(n+x)
char(n)	varchar(n+x)
varchar(n)	char(n+x)
varchar(n)	varchar(n+x)
graphic(n)	graphic(n+x)
graphic(n)	vargraphic(n+x)
vargraphic(n)	vargraphic(n+x)
vargraphic(n)	graphic(n+x)

If a decimal is converted to floating point, column cannot have unique index or a unique constraint

For decimal data types "a" plus "b" must be greater than zero or there is no change

For character data types "x" can be greater than or equal to zero

Figure 4 Supported Alter Data Types

Backup and recovery

To ensure that your application data remains available in the event of an unplanned outage, periodically create backup copies of your DB2 objects by running the COPY utility. In this section we discuss the following procedures, and provide recommendations for each:

- How to create multiple backup copies of your DB2 objects
- How to speed up recovery of your DB2 objects
- How backing up indexes can improve recovery performance

Creating multiple backup copies of DB2 objects

If you use the default parameters for the COPY utility only one backup copy will be created for each execution of the utility.

- We recommend that you make multiple copies to allow for access to data in the event that the primary copy becomes unavailable. You can take multiple image copies in a single execution of the COPY utility or you can make a copy of an image copy data set.

  Creating multiple image copies in one execution of the COPY utility is controlled by the COPYDDN and RECOVERYDDN options. You can specify two DD names, or alternatively a template, for the COPYDDN option. These DD names represent the primary and backup copy data sets for the image copy at the local site. You can also specify two DD names, or alternatively a template, for the RECOVERYDDN option. These DD names represent the primary and backup copy data sets for the image copy at the

recovery site. See *"Making multiple image copies"* in *DB2 UDB for z/OS Version 8 Utility Guide and Reference,* SC18-7427 for instructions and a sample COPY utility control statement.

If the additional copies of a data set are taken to slower devices, such as remote attached tape drives for the recovery site copies, the image copy could take longer to run than it would for a single copy, thus decreasing data availability. An alternative solution is to take a single image copy, then run the COPYTOCOPY utility to create additional backups from the primary image copy data set. The COPYTOCOPY utility can create up to three copies using the image copy data set that was created from running the COPY utility.

Since COPYTOCOPY does not run against the actual DB2 object, you can run SQL statements against the DB2 objects concurrently with execution of the COPYTOCOPY utility. For more information on when it is appropriate to use the COPYTOCOPY utility, see the redbook *DB2 for z/OS and OS/390 Version 7 Using the Utilities Suite,* SG24-6289.

- Consider specifying the CHECKPAGE option for your image copies. The CHECKPAGE option checks each page in the table space or index space for validity. If any errors are found, the COPY utility issues a message that describes the type of error. The COPY utility incurs a 10% CPU overhead when you are using the CHECKPAGE option, so you should balance the additional security of knowing that your data and indexes are free of errors against the CPU cost of that assurance.

Maintaining image copies of indexes

Starting with DB2 Version 6 you can create image copies of your indexes and of your table spaces. If you create an image copy of your indexes for a table space, you can then recover those indexes in parallel with the recovery of the table space.

- We recommend that you consider copying large indexes with low update activity that require high availability. Since there will be very few log records due to the low volume of updates, running RECOVER on the index could run much faster than running REBUILD.

Speeding up recovery of DB2 objects

When a problem does occur that requires you to recover application data, you want to be able to recover that data as quickly as possible to maintain high availability for the application.

- In order to maintain the highest possible availability for your application data, we recommend that you implement procedures, and take advantage of DB2 features that speed the recovery process.

Image copy strategy

- We recommend that you develop a strategy for backing up your application data.

  How frequently you back up your DB2 objects and the type of image copy you run should depend on the volume and frequency of updates to the data and the need for speedy recovery.

  For example, it is unlikely that you can run a daily COPY SHRLEVEL REFERENCE on a large table that is updated continuously on a 24 x 7 basis because there is no point in time when the data can be accessed in read only mode for a lengthy period. For those tables you may need to run a COPY SHRLEVEL CHANGE to allow updates to occur against the table while the COPY utility is running. Alternatively you can use the CONCURRENT option on a COPY SHRLEVEL CHANGE utility to invoke DFSMSdss™ concurrent copy function, which creates a snapshot of your data while continuing to allow updates.

  For more information on the various options of the COPY utility, see the redbook *DB2 UDB for OS/390 and Continuous Availability,* SG24-5486.

Standardized recovery jobs

When you do have a need to recover application data, how quickly you can put together the necessary recovery jobs can have a high impact on availability.

- Building standardized recovery jobs in advance for all your DB2 tables can speed up recovery time. You can also use the wild carding capabilities introduced in Version 7 to build a list of related table spaces to be recovered in one recovery job step.

  Wild carding is accomplished by providing a LISTDEF command prior to the RECOVER command. Example 1 shows a sample job to recover a list of table spaces in database DSN8D71A. The OPTIONS PREVIEW command expands the list definitions and reports on the results in the SYSPRINT data set. In this example all table spaces in database DSN8D71A are recovered with the exception of table spaces DSN8S71D and DSN8S71E. We discuss the PARALLEL keyword in the next section.

- In addition to using LISTDEF to build RECOVER jobs for a series of related objects, we recommend that you maintain separate RECOVER jobs using the TOCOPY and TORBA options. Since you are likely to need both types of RECOVER utilities at some point in time, it is better to have each type of job prepared and tested in advance.

COPY and RECOVER parallelism

Prior to Version 6 you could only COPY or RECOVER one table space per job step. If you had a number of objects to recover because of an outage, you had to recover each object individually. This meant creating multiple jobs or running each recover serially in separate steps within a job.

Starting with Version 6 you can copy and recover DB2 objects in parallel in the same job step. You do this by specifying a list of objects with the LISTDEF command, then by specifying the PARALLEL option for the COPY and RECOVER utilities. The PARALLEL option specifies the maximum number of objects that can be processed in parallel in any one utility execution. So if you had ten table spaces that you needed to recover in the event of an application failure you could specify the option PARALLEL 10 as shown in Example 1.

Example 1 RECOVER using LISTDEF command with wildcard

```
//STEP1 EXEC DSNUPROC,SYSTEM=DB2G,UID=PAOLOR1,
//UTSTATS=''
//SYSIN DD *
   OPTIONS PREVIEW
   LISTDEF DB01A
      INCLUDE TABLESPACE DSN8D71A.*
      EXCLUDE TABLESPACE DSN8D71A.DSN8S71D
      EXCLUDE TABLESPACE DSN8D71A.DSN8S71E
   RECOVER LIST DB01A PARALLEL 10
```

The PARALLEL option specifies the maximum number of objects that can be processed in parallel in one utility execution. If you do not specify a value for the PARALLEL option or you specify 0, then DB2 determines the optimum number of objects to process in parallel based on the available storage and the values for the system parameters CTHREAD and IDBACK.

Fast Log Apply

One of the potential inhibitors to availability is the time it takes to read the DB2 logs during DB2 subsystem restart and during the recovery of DB2 objects. DB2 Version 6 introduced Fast Log Apply to reduce both elapsed time and CPU time during these tasks.

Fast Log Apply still performs a single log read task as in Version 5, but sorts the log records into groups that pertain to the same page set before applying the changes. As a result,

changes can be applied in parallel, and there is a reduction in the need to re-read the same data pages to apply the changes recorded on the log records.

DB2 uses Fast Log Apply during the following processes:

- Forward phase of DB2 restart
- RECOVER utility
- START DATABASE for:
  - LPL recovery
  - GRECP recovery

Fast Log Apply utilizes an intermediate buffer storage area to sort the log records. System parameter LOGAPSTG specifies the maximum DBM1 storage that can be used by the Fast Log Apply process. You activate this feature by specifying a non-zero value for LOGAPSTG. The default value is 0 MB, which means that the fast log-apply process is disabled except during DB2 restart. During DB2 restart, the fast log-apply process is always enabled.

- ❑ We recommend that you specify 10 MB of log apply storage for each RECOVER job that you want to have faster log apply processing.

Coordinated backups

- ❑ If your applications access data sources other than DB2, you need to develop a backup and recovery strategy that allows for coordinated backups of all the data that your application accesses.

  For example, if your DB2 application also accesses IMS or VSAM data, you need to ensure that if you experience a system or application failure, you can recover your IMS or VSAM data to the same point in time as your DB2 data.

  See *DB2 UDB for z/OS Version 8 Administration Guide,* SC18-7413 for details on backing up and recovering data across multiple database systems.

Reorganization

The availability of your application data may be impacted greatly by how organized your data is. Tables that are highly volatile, with a great deal of insert, update, delete, and load activity, may become disorganized on a regular basis. As a result you could experience longer I/O time to return your data. This could lead to increased contention as more pages need to be accessed to retrieve the required data and locks are held for longer periods.

Certain database schema changes may require you to reorganize your application data to pick up the changes. For example, altering a table to add a column will not impact existing rows until you run the REORG utility against the table space for that table.

- ❑ For each of the these scenarios, determine the following:
  - When you need to run a REORG.
  - What type of REORG you need to run.

When to run REORG

DB2 Version 6 provides the OFFPOSLIMIT and INDREFLIMIT keywords to the REORG TABLESPACE utility and the LEAFDISTLIMIT keyword to the REORG INDEX utility to allow a REORG to conditionally execute if the criteria specified for one of those keywords is met.

For example, if you specified an OFFPOSLIMIT of 10 for a REORG TABLESPACE utility, the REORG utility executes the following SQL statement before actually reorganizing the table space:

```
SELECT CARDF, (NEAROFFPOSF +FAROFFPOSF)*100/CARDF
```

```
FROM SYSIBM.SYSINDEXPART
WHERE CARDF >0
AND (NEAROFFPOSF + FAROFFPOSF)*100/CARDF > :offposlimit value
```

The SQL executes using the value of 10 for the host variable. Any rows returned indicate a degradation of the clustering index for the table space specified. The REORG is executed.

For more information on how to use keywords to conditionally execute the REORG utility, see the redbook*, DB2 for z/OS and OS/390 Version 7 Using the Utilities Suite,* SG24-6289.

What type of REORG to run

There are three types of the REORG utility that can run. Each type provides a different level of availability. Choose the type of REORG that fits your availability needs. Following are the three types of REORG:

Table 2 Types of REORG

Type of REORG	Description
SHRLEVEL NONE	This is the most restrictive of all REORGs. During the UNLOAD phase, read-only access is allowed to the data. Once the RELOAD phase starts all readers are drained and no access is allowed until the REORG finishes, assuming Inline COPY is taken during the REORG. Use this type of REORG if you have a sufficient batch window during which your data can be unavailable.
SHRLEVEL REFERENCE	Read-only access is allowed during the RELOAD phase by placing the object in UTRO status until the SWITCH phase begins. At this point the object is placed into UTUT status and readers are removed via a drain on the objects. This means that the table is unavailable to readers until the SWITCH is complete.

Type of REORG	Description
SHRLEVEL CHANGE	This type of REORG allows both readers and writers access to the data until the last log iteration and the SWITCH phase. During the RELOAD phase, applications can access the data in UTRW mode, with any updates being written to the log as normal. Once the RELOAD is finished, REORG begins processing the log records and applying them to the new “shadow” data sets. The SHRLEVEL CHANGE type of REORG is often called on-line REORG because the DB2 data sets are completely available in read-write mode, while the REORG is processing against the shadow data sets up until the time of the SWITCH phase. The log is processed in cycles, called iterations, until DB2 decides that there is little enough log to apply. Then it can meet the time scales specified in the MAXRO parameter. Next, the last iteration of the log is processed, the object status is set to UTUT (if not already done), the readers are drained (if not already done), and the SWITCH phase commences. When complete, the drains are released, status is set to RW, and REORG deletes unwanted data sets, including the original object data sets.

Prior to DB2 Version 7, during the SWITCH phase DB2 would rename the original data sets to a temporary name, then rename the shadow data sets to the original data set names. The renames could take some time, especially in the case of a large partitioned table space with many partitions.

Starting with Version 7 you can make use of the FASTSWITCH keyword of on-line REORG that updates the catalog to refer to the shadow data sets rather than issuing renames for every data set. As a result the SWITCH phase is much faster and the data is unavailable for a shorter period of time. The first character in the instance node can alternate between a value of ‘I’ or ‘J’ and the catalog and the directory are updated to point to the shadow data sets rather than to the original data sets, and the original data sets are deleted. This is shown in Figure 5 on page 28.

FASTSWITCH YES is the default for REORG TABLESPACE and for REORG INDEX.

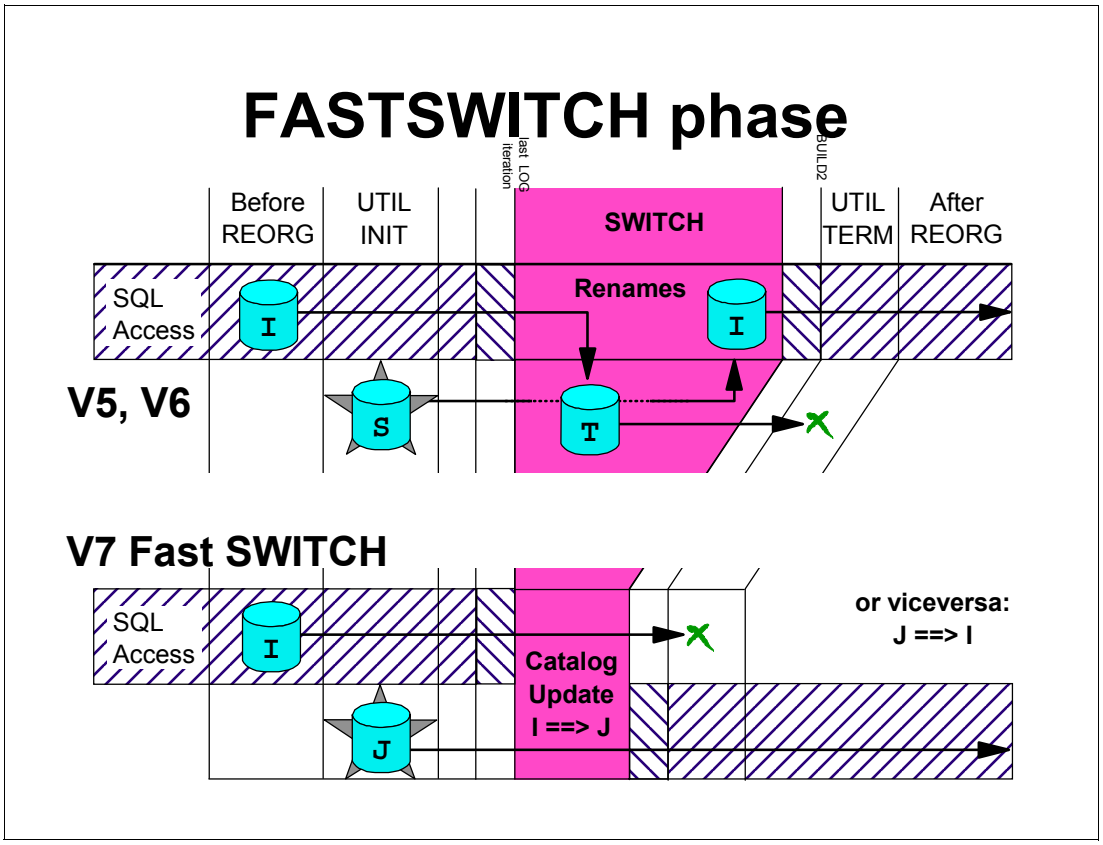

Figure 5 FASTSWITCH phase

For more details on the different types of REORG, a summary of availability features for each, and recommendations for running REORG, see the redbook *DB2 for z/OS and OS/390 Version 7 Using the Utilities Suite,* SG24-6289.

Application design introduction

In addition to configuring your DB2 systems and application data for high availability, it is important to implement proper coding procedures to prevent against application failures, performance problems, and contention problems, which can all impact system availability. In this section we discuss some of the techniques you can use during the application design, coding, testing, implementation, and recovery phases, in order to maintain high application availability.

We describe additional techniques on how to design your applications for high availability in detail in the redbook *Parallel Sysplex Application Considerations*, SG24-6523 and in*DB2 UDB for z/OS Version 8 Application Programming and SQL Guide,* SC18-7415. These redbooks cover the following topics:

- Application design
- Application coding
- Application management
- Application recovery

Application design

A poor application design can have a greater negative impact on application availability than just about any other phase of the application development process. In this section we discuss some techniques for designing highly available applications.

Manageable units of work

Applications that run for a long time without committing tend to hold resources that may be required by other applications. A failure to code frequent commits, especially for long running batch jobs that run concurrently with on-line transactions, can cause performance degradation and even deadlocks or time-outs with your transactions.

- Ensure that batch jobs include commit logic, perhaps every 100 updates, to release locks held on objects that other applications may require.
- If your applications access non-DB2 data in addition to DB2 data, then you must design your applications with coordinated sync points to ensure that both DB2 and non-DB2 data can be recovered to the same point in time. You can use a checkpoint/restart tool to facilitate restart processing for you.

Design for concurrency

You should design applications with the understanding that other applications may need access to the same data. This can especially be a problem if you have small control tables or static code tables that are accessed by many applications.

- If you are using control tables to generate keys for tables, you will likely experience contention on those control tables, especially as your volume of work increases. Consider using IDENTITY columns or SEQUENCES to generate keys for you instead of control tables.
- If you have many static code tables that are used across different applications, we recommend that you consider alternatives to allow each application its own copy of the data. One alternative is to use a declared temporary table (DTT) to store the information from the code table. This technique is especially useful for long running applications that need to load the code table information once but will use it throughout the length of the job.

  Using a DTT reduces contention because the only locking that occurs on the code table will be at the time it is read to populate the DTT. Once the DTT is populated you can use SQL to access the codes without requiring locks against the base code table.
- Consider using a technique called *lock avoidance* whenever possible. When you bind with option CURRENTDATA(NO), DB2 can test to see if a row or page has committed data on it. If it does, DB2 does not have to obtain a lock on the data at all.

  Unlocked data is returned to the application, and the data can be changed while the cursor is positioned on the row. Lock avoidance is also impacted by the frequency of commit logic. If applications that access the same data concurrently do not commit frequently, you will not be able to take advantage of lock avoidance.

These techniques and many more are described in more detail in the redbook *Parallel Sysplex Application Considerations*, SG24-6523 and in *DB2 UDB for z/OS Version 8 Application Programming and SQL Guide,* SC18-7415.

Application coding

Good application coding techniques are usually a result of good application design, but there are some areas to be aware of when coding for high availability.

Effective use of database design

- When coding your SQL statements in an application, take advantage of the database design. DB2 can often process data more efficiently than can an application's program logic (especially when there are indexes on the selection criteria). Therefore, whenever possible, code your SQL to restrict the amount of data returned to your program, as opposed to reading all the data in and letting the program eliminate the rows that are not needed.
- DB2 can more efficiently join data from multiple tables than can be done manually within your program. Logic that reads rows from one table, and then uses the results to read row by row from a second table, is more inefficient than a join would be. Also, this logic can elongate run times and potentially hold locks for longer periods of time.

Retrieve only columns the application needs

- Code your SQL statements to select only those columns that your application needs.

  Even though DB2 retrieves data at the page level and loads the pages into the buffer pool, only those columns selected are passed from the buffer pool to your application. There is a CPU cost associated with each column passed, so if you select only the columns you need you will reduce the CPU cost. Additionally, you may benefit from index-only access if you can limit your select to only columns that are in an index. In that case no locks are taken against the data pages, which makes that data more available to other applications.

Application management

Application management consists of processes to test, promote, and maintain applications once they are designed and coded. In this section we discuss some techniques you can use throughout the testing and implementation stages to ensure high availability for your applications and their data.

Testing procedures

One of the key phases of the application development process is the test phase. Companies that maintain a highly available environment typically have multiple degrees of testing to perform on an application before promoting it to production.

- We recommend that you perform the following levels of test for any application whose availability is critical. Not all of these test phases require separate processing environments. In many cases you can perform tests in the same environment with some additional parameters applied. Your test environments may be configured differently than what we recommend here, but we recommend that your test environments include all of the functionality encompassed by the test phases that follow.
- We also recommend that you run EXPLAIN at each level of testing. EXPLAIN can identify potential performance or contention problems before they occur in a higher level test environment or in production.

The following table contains the recommended levels of testing for any application whose availability is critical.

Table 3 Recommended levels of testing

Test	Description
Functional test	This level of testing verifies all code paths for the programs that were written or changed. The purpose is to verify that the new code correctly performs the intended function.

Test	Description
Product verification test	This level of testing verifies that the new or changed code works in conjunction with the other programs that make up the application. The purpose is to verify that the application, as a whole, works as designed.
System integration test	This level of testing verifies that the new or changed application works in conjunction with the other applications that share the same system. The purpose is to determine whether the new or changed application has any features that would negatively impact other applications running in the same environment. This is often the first level of testing where contention problems are discovered.
Stress test	This level of testing is similar to the system integration test, but it exposes the applications to transaction levels equivalent to a production environment. The purpose is to identify problems that only show up under stress situations. Both contention and performance problems are often identified here. This is the first level of testing where you should run your DB2 performance monitoring tool against the application being tested.
Performance test	This level of testing is performed only after all problems that the previous testing phases identified are resolved. The purpose is to measure throughput rates relative to previous benchmarks and/or service level agreements.
Regression test	This level of testing provides an environment similar to the production environment and can be used to verify system software maintenance, DSNZPARM changes, or other DB2 environmental changes such as a change in buffer pools, RID pool, EDM pool or CF structures. The purpose is to verify that a change in the DB2 environment does not negatively impact applications.

Application promotion procedures

Once an application is validated at all levels of testing, it is then ready to be moved into a production environment. Even if your stress and performance test environments were carefully created to mirror your production environment, it is possible that your application will behave differently once it is promoted to the production environment.

- We recommend that you bind your plans and packages with EXPLAIN against the production environment, and verify that the access paths chosen are the ones you expected.

You have two alternatives for verifying access paths prior to overlaying the production versions of the applications:

Bind into a verification collection

As part of your production promotion procedures you can bind your application into a collection designed specifically for access path verification. Since the bind is occurring on the production system, you will see the access paths that will be used in production but without overlaying the existing production application. You can then manually compare the access paths to determine whether it is appropriate to continue.

Integrate access path checking into promotion process

An alternative to binding into a separate collection and manually comparing access paths is to implement a tool that compares the new access path with the old one prior to issuing the bind. The IBM tool DB2 Path Checker for z/OS can generate the EXPLAIN output for the new or changed application prior to performing the bind. DB2 Path Checker for z/OS can be integrated into your JCL procedures to prevent binds, that will cause a regression in access paths, from occurring.

In addition to verifying that proper access paths are selected, your production promotion process should consider what is required if the application change needs to be backed out. If a problem is identified after a change is implemented, and there is a need to revert back to a prior version of the application, you must maintain backup versions of source code, DBRMs, and load modules.

- ❑ We recommend that your production promotion process include a step that copies the prior versions of the source code, DBRM, load module and, if applicable, the DDL to backup libraries.

Application recovery

Regardless of how careful you are during the application design, coding, and testing phases, there are times when you have to recover either application data or an entire application. In this section we discuss techniques to prepare for application recovery and the steps required to perform the recovery.

Application data recovery

If you need to recover your application data to a point in time, you need to establish a quiesce point to which you will recover. If your application fails it will be rolled back to the most recent commit point or sync point. Then determine whether any changes committed to that point need to be undone. If you need to back out changes committed by an application, run the RECOVER utility to a point in time prior to the application.

Use the QUIESCE utility to establish a consistent recovery point (the current log RBA or LRSN) for a set of DB2 objects. You can use the LISTDEF capabilities available with DB2 Version 7 to identify a set of table spaces that need to be quiesced for potential recovery purposes.

Assume that database DSN8D71A contains five table spaces, DSN8S71A through DSN8S71E. If application A accesses table spaces DSN8S71A, DSN8S71B and DSN8S71C, then you can issue the QUIESCE statement shown in Example 2 to establish a quiesce point for the application.

Example 2 QUIESCE utility using the LISTDEF command

```
//STEP1 EXEC DSNUPROC,SYSTEM=DB2G,UID=QUIESCE
//SYSIN DD *
   LISTDEF DB01A
      INCLUDE TABLESPACE DSN8D71A.*
      EXCLUDE TABLESPACE DSN8D71A.DSN8S71D
```

```
   EXCLUDE TABLESPACE DSN8D71A.DSN8S71E
QUIESCE LIST DB01A WRITE YES
```

Once you determine that you need to recover your application data, you can issue the standardized recovery job that you prepared for this application, as discussed in, “Backup and recovery” on page 22, and insert the appropriate RBA or LRSN as the recovery point.

Application code recovery

In cases where your application change needs to be backed out, you can restore your application code to the level prior to the change if you maintained copies of the prior versions of the load library and DBRM. For more details on backing up and recovering DB2 databases, see *DB2 UDB for z/OS Version 8 Administration Guide,* SC18-7413.

End-user connectivity

Access to a DB2 data sharing group can vary depending upon the needs of the application or the end user that requires access. In this section we will explore various methods for accessing DB2 data, and provide an overview of techniques you can use to redirect work to another member in the case of failure of one of the members. We discuss the following topics:

- Overview of access methods
- Access directed to a single member
- Access directed to the group
- Using DB2 Connect to access remote data sharing groups
- Using group attach to connect to a data sharing group

Overview of access methods

Applications and end users can communicate with a DB2 data sharing group by using one of two protocols: Transmission Control Protocol/Internet Protocol (TCP/IP) or Systems Network Architecture (SNA) protocol. Each of these protocols supports multiple methods to access data-sharing groups.

- A TCP/IP requester can use one of the following three methods to access a data sharing group:
  - Group access
  - Member-specific access
  - Single-member access
- An SNA requester can use one of the following three methods to access a data sharing group:
  - Member-specific access
  - Group-generic access
  - Single-member access

Although the types of accesses sound the same for each protocol, there are differences in how they behave. Each method is described in detail in *DB2 UDB for z/OS Version 8 Data Sharing: Planning and Administration,* SC18-7417.

Since many of the recent network computing enhancements to DB2 are only supported when using TCP/IP connections, we will only discuss the TCP/IP protocol for the remainder of this section.

Access directed to a single member

Single-member access is used to connect to a specific member of a data sharing group. All subsequent requests connect to the same member. If you have applications that must run on a specific member, because of affinities to other resources such as VSAM files or other non-DB2 databases, you can use the single-member access method to connect to that member. Single-member access is just like accessing a DB2 subsystem that does not share data.

- We recommend that you do not use single-member access because it does not allow for workload balancing across the data sharing group and does not allow for failover processing since the application can not connect to another member if the specific member is unavailable.

Access directed to the group

Group access is supported by either dynamic virtual IP address (VIPA) network addressing or domain server (DNS) network addressing. The application requesting the connection uses the dynamic VIPA or DNS name to connect to any member of the group. That member then provides the requester with a list of available members. Subsequent requests can connect to any of the available members.

Dynamic VIPA network addressing gives you the ability to assign a specific, virtual IP address to a data sharing group and to each member of the group. This address is independent of any specific TCP/IP stack within the Parallel Sysplex. Even if a member is moved to another z/OS system, as in the case of a failure or maintenance, the member remains accessible and retains the same virtual IP address. Dynamic VIPA is supported in OS/390 V2R8 and above.

Sysplex Distributor is the strategic IBM solution for connection workload balancing in a Parallel Sysplex and is built on dynamic VIPA. When a TCP connection request arrives, the Sysplex Distributor routing stack consults Workload Manager (WLM) to find the relative available capacities on the nodes (OS/390 or z/OS) hosting the stack and application. The routing node also consults Service Policy Agent for network performance and defines policies that might affect the distribution decision.

With all available relevant information, including precisely which target stacks have server applications ready for work, the routing stack selects a target stack and forwards the request to that target stack for processing. The routing stack also remembers the full connection information (source and destination IP addresses and ports), so that future TCP segments for that connection are sent on to the same target stack for processing.

When the connection ends, the target stack notifies the routing stack that the connection has ended, so the routing stack can discard the connection routing entry for the ended connection. The next connection from the same client is distributed independently of the earlier one.

- We recommend that you use dynamic VIPA and Sysplex Distributor to configure your Parallel Sysplex for the best distribution of workload and to ensure continuous connectivity in the event of failure of one of the members. For more details on dynamic VIPA and Sysplex Distributor, see the technical paper *Leveraging z/OS TCP/IP Dynamic VIPAs and Sysplex Distributor for higher availability,* GM13-0165 on the IBM zSeries® Literature web site:

  `http://www-1.ibm.com/servers/eserver/zseries/library/literature/reference.html`

Using DB2 Connect to access remote data sharing groups

DB2 Connect Enterprise Edition connects LAN-based systems and their desktop applications to your company's mainframe and minicomputer host databases. DB2 Connect can be

configured to access a single member or the entire group. Configuring DB2 Connect to access a single member requires disabling its sysplex support, which makes DB2 Connect dependent on that member being operational.

- ❑ We recommend that you configure DB2 Connect for group access.

DB2 Connect takes advantage of dynamic VIPA and Sysplex Distributor to balance the distributed workload across the members of the group. You need to configure DB2 Connect to access remote data-sharing groups. Use the Configuration Assistant to update the database directories that DB2 Connect uses to manage database connection information.

For more details on configuring DB2 Connect to access remote data-sharing groups, see *DB2 UDB for z/OS Version 8 Data Sharing: Planning and Administration,* SC18-7417 or *DB2 Connect Users'Guide Version 8,* SC09-4835.

Using group attach to connect to a data sharing group

When you submit a job on a z/OS system, you specify a DB2 subsystem name. That name can be either an actual subsystem name or a group attachment name. The group attachment name is defined on installation panel DSNTIPK and can be used by batch programs, the call attachment facility (CAF), the RRS attachment facility (RRSAF), IMS, CICS Transaction Server, and DB2 utilities. The group attachment name should not be the same as any of the member names in the group.

When one of the above listed requesters attempts to connect to the DB2 subsystem listed in the JCL, DB2 attempts to find a qualifying subsystem name.

- If that subsystem is found and it is started, DB2 attaches to that subsystem.
- If the subsystem is not found, then DB2 uses the group attachment name and connects to the first started DB2 member that is defined to this z/OS system, according to the order in which the subsystems were initialized at IPL time. DB2 will always connect to the first started subsystem on the list. There will be no attempt at load balancing.
- If the subsystem is found but it is not started and the connection request did not specify NOGROUP, then DB2 uses the group attachment name and connects to the first started DB2 member that is defined to this z/OS system, according to the order in which the subsystems were initialized at IPL time. DB2 will always connect to the first started subsystem on the list. There will be no attempt at load balancing.

Group attach recommendations

- ❑ We recommend that you use the group attachment name for jobs that are not sensitive to a particular member. Specify the group attachment name for CICS or IMS applications, but note that CICS and IMS applications must be aware of the particular member to which they are attached. This is so they can resolve indoubt units of recovery in the event of a failure.

  IMS Version 7, or later, allows IMS dependent regions to use the DB2 group attachment name to direct jobs to a data sharing group. CICS Transaction Server version 2.2 or later allows you to use the CICS RESYNCMEMBER=YES option to handle indoubt units of work. See *DB2 UDB for z/OS Version 8 Data Sharing: Planning and Administration,* SC18-7417 for more information about running CICS and IMS applications.

Operations and procedures

Operations and procedures play a significant role in maintaining high availability for DB2 subsystems and data-sharing groups. In this section we discuss the following activities for which you should develop standardized procedures:

- Tracking and applying DB2 maintenance
- Changing system parameters
- Migrating to a new version of DB2
- Monitoring a data sharing group
- Maintaining DB2 statistics

Tracking and applying DB2 maintenance

Many of the problems that customers experience with DB2 could have been avoided with better preventive maintenance of both DB2 and associated zSeries software products. There were a number of enhancements in recent years to the IBM Software Support process that make it easier for you to track and apply available DB2 maintenance.

- IBM established a service testing environment called Consolidated Service Test (CST). The CST team combines Recommended Service Upgrades (RSUs) for multiple zSeries products into a single testing environment so that they can recommend PTF service for z/OS and key subsystems together in one RSU sourceid to all z/OS customers. When you order service through ShopzSeries you can receive and install tested service for all of the following products and service levels:
  - z/OS V1R3, V1R4, and V1R4 z990 Exploitation Support
  - CICS TS 1.3/CPSM 1.4
  - CICS TS 2.2
  - DB2 UDB for OS/390 V7
  - WebSphere® Application Server V4 and V5 for z/OS and OS/390
  - WebSphere MQ 5.3 and 5.3.1
  - IMS V7 and V8
  - IRLM 2.1
  - Data Management Tools for DB2 and IMS

  For more details on the Consolidate Service Test and the Recommended Service Upgrade, see the following web site:

  `http://www-1.ibm.com/servers/eserver/zseries/zos/servicetst/`

- Additionally, IBM maintains a Software Support Handbook which provides information about tracking APARs and applying maintenance, as well as the various levels of support offerings. The handbook can be found at the following web site:

  `http://techsupport.services.ibm.com/guides/handbook.html`

- We also published a redpaper that provides details on the CST and RSU, as well as some general maintenance recommendations for OS/390 and z/OS products and a recommended installation procedure for preventive service. The redpaper is *Improvements in z/OS Service*, REDP-0324.

One of the advantages of data sharing is that you can apply maintenance on one member at a time, while still maintaining availability to the data from other members of the group. Table 4 on page 37 shows the action required to maintain data availability for various types of planned maintenance.

Table 4 Planned maintenance changes

Type of change	Action required
Early code	Bring down one z/OS system at a time and re-IPL.
DB2 code	Bring down and restart each member independently.
IRLM code	Bring down and restart each IRLM member independently.
Attachment code	Apply the change and restart the transaction manager or application.

Changing system parameters

DB2 system parameters (ZPARMs) apply to each individual subsystem in a data sharing group; therefore, you can change ZPARMs one member at a time without having to bring the entire group down. If the parameter is one that can be changed dynamically, then you do not have to bring even that one member down. If the parameter is not dynamically changeable then you will have to stop and start that one member to activate the change.

- We recommend that you dynamically change ZPARMs whenever possible to avoid the outage required by bringing the system down and back up to activate the parameter. To update the ZPARMs on a DB2 system follow these steps:
  - Run through the installation process in Update mode.
  - Produce a new subsystem parameter load module.
  - Issue the -SET SYSPARM command.

The -SET SYSPARM command was introduced in DB2 V7 to provide the capability to change system parameters while DB2 is up. See *DB2 UDB for z/OS Version 8 Command Reference,* SC18-7416 for the syntax of the -SET SYSPARM command and instructions for its use.

For a list of the ZPARMs you can update online, see *DB2 UDB for z/OS Version 8 Command Reference,* SC18-7416. For more details on how you update those parameters, see *"Updating parameters through the Update Selection Menu panel: DSNTIPB"* in *DB2 UDB for z/OS V8 Installation Guide,* GC18-7418.

Migrating to a new version of DB2

When you migrate your DB2 data sharing group to a new version of DB2 you have two options for how you will migrate each individual member:

- Migrate all members at once

  If you migrate all members at once you are experience an outage for the entire data sharing group for the time it takes to migrate all the members.

- Migrate each member individually

  If you migrate each member individually, parts of the DB2 catalog are unavailable during the migration of the first member, due to locks that are obtained by the catalog migration utility (CATMAINT). After the first member is migrated you can migrate the remaining members one at a time while the data remains available to the end user on the other members. During the time that the group consists of members in two different versions, your group is considered to be in coexistence mode.

  The purpose of coexistence is to allow continuous availability of your DB2 data.

- ❑ If you need continuous availability, we recommend that you migrate using the coexistence method. But do so at a period of low activity since there are some operational costs of running in coexistence mode. Also, set the ABIND ZPARM to COEXIST on each member prior to starting the migration to ensure that automatic rebinds do not happen repeatedly across the group. See, “Data sharing configuration recommendations” on page 15 for more details about the ABIND parameter.
- ❑ If you do not need continuous data availability, plan to shut down the entire group during migration.

Please see *“Migrating an existing data sharing group to a new release”* in *DB2 UDB for z/OS Version 8 Data Sharing: Planning and Administration,* SC18-7417 and *“Migrating a data sharing group”* in *DB2 UDB for z/OS V8 Installation Guide,* GC18-7418 for more details on migrating a data sharing group to a new version of DB2.

Monitoring a data sharing group

Monitoring of a data sharing group can exist at two levels:

- Monitoring the group and its structures
- Monitoring the performance of the DB2 members.

Monitoring the group and its structures

System monitoring is an integral component of a high availability strategy. You can display various information about a data sharing group and about the Coupling Facility structures and policies by issuing various DB2 and z/OS DISPLAY commands. The following tables contain the available DB2 and z/OS commands.

Table 5 DB2 commands available for monitoring data-sharing groups

DB2 and z/OS commands	Description
-DISPLAY GROUP	Displays information about a data sharing group to which a subsystem belongs. You can see all the members and whether or not they are active.
-DISPLAY GROUPBUFFERPOOL	Displays the status of the group buffer pools.

Detailed examples of each of the commands in Table 5, are shown in *“Monitoring the group”* in *DB2 UDB for z/OS Version 8 Data Sharing: Planning and Administration,* SC18-7417.

Table 6 z/OS commands available for monitoring data-sharing groups

DB2 and z/OS commands	Description
D XCF,STR	Displays information about the Coupling Facility structures and policies. Provides summary information about all structures.
D XCF,STR,STRNAME=strname	Displays more detailed information about the structure identified by *strname*. The structure can be a group buffer pool, the lock structure or the SCA.

Detailed examples of each of the commands in Table 6, are shown in *“Monitoring the group”* in *DB2 UDB for z/OS Version 8 Data Sharing: Planning and Administration,* SC18-7417.

Monitoring the performance of DB2 members

When you operate in a data sharing environment, you often can see performance levels at or near that of environments that do not share data. The overhead introduced by data sharing is proportional to the amount of inter-DB2 read/write interest experienced by your workload. This overhead can often be controlled by introducing levels of parallelism into your design or by controlling the locking in your applications.

Monitoring parallelism

- You can control inter-DB2 read/write interest by separating jobs by partition key ranges. Additionally, you can reduce elapsed times when running multiple jobs by partition in parallel. DB2 Version 8 introduces table-controlled partitioning and data partitioned secondary indexes, which make parallel processing even more viable than before by eliminating contention on non-partitioned indexes (NPIs).
- For complex read-only queries you may want to consider taking advantage of Sysplex query parallelism, which uses the full power of the data sharing group to process the query on multiple members simultaneously. Sysplex query parallelism is enabled by specifying a value of YES for the COORDINATOR ZPARM on installation panel DSNTIPK for at least one of your DB2 members and by specifying a value of YES for the ASSISTANT ZPARM on installation panel DSNTIPK for at least one other of your DB2 members.

  You can issue the DISPLAY GROUP command with the DETAIL option to see which members can act as coordinators and which members can act as assistants. A member can act as a coordinator for queries submitted on that member and as an assistant for queries submitted on a different member.

Monitoring locking

- Another area you should monitor for data sharing performance is the level of locking that occurs, both at the local level and the global level. Locking problems in a data sharing environment can be due to application design issues or due to an inadequately sized lock structure. There are multiple methods you can use to monitor locking in a data sharing group.

Table 7 Methods for monitoring locks

Monitor locking method	Description
-DISPLAY DATABASE LOCKS command	displays information about page set, partition or table locks that are held on resources within the database specified
Resource Measurement Facility (RMF™)	collects resource usage data for a Parallel Sysplex. The Coupling Facility Activity Report of RMF reports on locking activity to the Coupling Facility. It reports on the total number of lock requests, the number of requests deferred because of contention, and the number of deferred requests that were caused by false contention. You can use this report to determine whether your lock structure is sized properly.
DB2 statistics trace	keeps counters that track global locking activity and contention for each member of the group. This trace incurs very low overhead, so we recommend that you keep this trace running to provide continuous monitoring of each member.

Monitor locking method	Description
DB2 accounting trace	identifies which plans are experiencing global lock contention. You can see the number of suspensions as well as the amount of false contention experienced for each plan.
DB2 performance trace class 6 (IFCID 0045)	indicates whether the suspension is because of contention. You should only activate this trace as needed if you can not resolve the locking problem from information provided by one of the other methods. The performance trace incurs significantly more overhead than either the statistics or accounting traces. If you do start a performance trace, do so only for a specific plan or set of plans to identify a performance problem. Remember to stop the trace once you identify your problem.

All of the options listed Table 7 on page 39 for monitoring locking problems are discussed in more detail under the topic *"Monitoring DB2 locking"* in*DB2 UDB for z/OS Version 8 Data Sharing: Planning and Administration,* SC18-7417.

- You can also use a tool such as DB2 Performance Expert to assist you in interpreting the output of the statistics, accounting, and performance traces.
- We recommend that you aim for total global lock contention of less than 5%, preferably less than 1%. False contention should be less than half of total global lock contention. You can use the same tuning actions to reduce lock contention in data sharing as you do in a single DB2 subsystem.

For detailed recommendations for reducing contention please see the topic *"Tuning your use of locks"* in *DB2 UDB for z/OS Version 8 Data Sharing: Planning and Administration,* SC18-7417 or the topic *"Lock Tuning*" in *DB2 UDB for z/OS Version 8 Administration Guide,* SC18-7413.

General monitoring recommendations

- In addition to executing the specific commands and running the reports previously listed, we recommend that you implement a proactive monitoring strategy to identify potential problems before they impact system availability. Most performance monitoring tools allow you to set thresholds and provide alerts or work with your automation tools to take action when those thresholds are reached. Familiarize yourself with your monitoring and automation tools and configure them to identify problems in a proactive manner.
- Develop a process to regularly review performance reports to determine whether any applications are experiencing degradation in performance that could eventually cause availability problems.

Maintaining DB2 statistics

The maintenance of current DB2 statistics can be critical to maintaining high availability for your DB2 applications. If you do not have current statistics, it is possible that DB2 could choose an incorrect access path for your applications, which elongates elapsed time and the period of time in which locks are held. As a result you could have inefficient applications that tie up resources and makes those resources unavailable to other users and applications.

- We recommend that you specify procedures for running RUNSTATS on all your DB2 objects on a regular basis. You should especially take note whenever any major changes occur that could cause a change in the structure of some DB2 objects or a change in the

volume of transactions against those objects. You can use tools such as DB2 Path Checker for z/OS to identify potential unacceptable access path changes due to old or insufficient statistics.

Supporting material

There have been a number of publications and presentations over the years that address techniques for maintaining a highly available DB2 environment, both within a single DB2 subsystem and within a Parallel Sysplex. This section provides some high level recommendations and refers to other publications where you can find more details.

DB2 redbooks

The following redbooks provide information on DB2 availability:

- *Parallel Sysplex Application Considerations*, SG24-6523

  This book focuses on DB2 availability in a Parallel Sysplex from the application design perspective, as opposed to the system design perspective.

- *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ...and More, SG24-6079*, published in May, 2004.

  This book provides details on all the DB2 V8 features, including those that impact availability. You can find details about online schema changes, partitioned tables, data-partitioned secondary indexes, and many more availability enhancements.

- *DB2 for z/OS and OS/390 Version 7 Selected Performance Topics, SG24-6894*, published in November, 2002.

  This book is a follow-up to SG24-6129, and provides additional details on enhancements to DB2 logging that can impact availability.

- *DB2 for z/OS and OS/390 Version 7 Using the Utilities Suite, SG24-6289*, published in August, 2001.

  This book highlights changes to utilities processing in DB2 V7, such as wild carding and templates, parallelism for COPY, RECOVER and LOAD, and the FASTSWITCH enhancement to online REORG. Use this book in conjunction with *S*C18-7427 to plan your utility strategy for high availability.

- *DB2 for z/OS and OS/390 Version 7 Performance Topics, SG24-6129*, published in July, 2001.

  This book highlights new features of DB2 Version 7 from a performance perspective. You can find more details on many availability enhancements such as fast log apply, dynamically changeable ZPARMs, and load partition parallelism.

- *DB2 UDB for OS/390 and Continuous Availability,* SG24-5486, published in May, 2000.

  This book addresses how DB2 can be established to provide a continuously available solution. It focuses on features in DB2 V5 and V6, many of which still apply today, which focus on database and application availability.

- *Application Design Guidelines for High Performance,* SG24-2233, published in August, 1997.

  This book provides recommendations for database design and application design for high performance. It also includes a chapter on considerations for a data sharing environment.

- *Parallel Sysplex Availability Checklist*, SG24-6061, published in December, 2004.

This book provides recommendations for High Availability in a Parallel Sysplex environment from the hardware, the operating system, and systems management processes point of view.

Technical papers

- *Leveraging z/OS TCP/IP Dynamic VIPAs and Sysplex Distributor for higher availability,* GM13-0165, published in July, 2002.

  This paper provides details on configuring dynamic VIPA and using the Sysplex Distributor feature of z/OS. The paper can be found at the following web site:

  `http://www-1.ibm.com/servers/eserver/zseries/library/literature/reference.html`

- *Improvements in z/OS Service*, REDP-0324, published in September, 2002.

  This redpaper provides information on enhancements to the z/OS service stream such as the Consolidated Service Test and Recommended Service Upgrade, including recommendations on applying maintenance. The redpaper can be found on the IBM redbooks web site.

Authors

Frank Kyne is a Certified IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes and teaches IBM classes worldwide on all areas of Parallel Sysplex. Before joining the ITSO 6 years ago, Frank worked in IBM Ireland as an MVS™ Systems Programmer.

Glenn McGeoch is a Senior DB2 Consultant for IBM DB2 for z/OS Services organization in the United States, working out of San Francisco, CA. He has 26 years of experience in the software industry, with 18 years of experience working with DB2 for z/OS and OS/390. He holds a degree in Business Administration from the University of Massachusetts and an MBA from Rensselaer Polytechnic Institute. Glenn worked for 19 years as an IBM customer with a focus on CICS and DB2 application development, and spent the last 7 years with IBM assisting DB2 customers. His areas of expertise include application design and performance, DB2 data sharing, and DB2 migration planning. He has given presentations on data sharing at regional DB2 User Groups, and he has presented to customers on DB2 stored procedures, migration planning, and application programming topics.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include 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.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®	IBM®	Rational®
Redbooks (logo) ™	IMS™	RETAIN®
CICS®	Lotus®	RMF™
DB2 Connect™	MVS™	S/390®
DB2®	MVS/ESA™	Tivoli®
DFS™	OS/390®	WebSphere®
DFSMSdss™	Parallel Sysplex®	z/OS®
FICON®	Perform™	zSeries®

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of the The Open Group in the United States and other countries.

Linux is a trademark of Lunus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.