

Db2 for z/OS

Backup and Recovery Handbook

Reference Manual

Copyright © 2022 Broadcom. All Rights Reserved. The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries. For more information, go to www.broadcom.com. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Broadcom reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design. Information furnished by Broadcom is believed to be accurate and reliable. However, Broadcom does not assume any liability arising out of the application or use of this information, nor the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

Table of Contents

Chapter 1: About This Document	8
1.1 Purpose and Audience	8
1.2 Authors	8
1.3 About the Mainframe Division at Broadcom®	9
1.4 Document Structure	9
1.5 References	9
Chapter 2: Identifying Current Backup and Recovery Procedures	10
Chapter 3: Introduction to Backup and Recovery in Db2	11
3.1 Prepare an Efficient Db2 Backup Strategy	11
3.2 Know The Applications	12
3.2.1 Grouping Business Applications	12
3.2.2 Determine the Recovery SLA (Recovery Time Objective, RTO)	13
3.2.3 Map the Database Objects to Application Groups	13
3.2.4 Review	13
3.3 Introduction to Recovery	14
Chapter 4: Db2 Logging and Creating Data Backups	15
4.1 The Db2 Logging Environment	15
4.1.1 Bootstrap Data Set	15
4.1.2 Active and Archive Log Data Sets	15
4.1.3 Log Buffers	16
4.1.4 Checkpoints	17
4.1.5 Logging in a Data Sharing Environment	17
4.1.6 SYSLGRNX Directory Table	18
4.1.7 Closing and Opening Data Sets	18
4.1.8 Reducing Log Records by Converting to Read-Only Page Sets	18
4.1.9 Summary	19
4.2 Image Copy Considerations	19
4.2.1 Availability of Copied Objects	19
4.2.1.1 Availability Recommendations and Considerations	19
4.2.1.2 Choosing Between SHRLEVEL REFERENCE and CHANGE Image Copies	20
4.2.1.3 DD Statements	20
4.2.1.4 Dynamic Allocation	20
4.2.1.5 TEMPLATE Utility	20
4.2.2 Types of Image Copies	21
4.2.2.1 Sequential Image Copies	21
4.2.2.2 Concurrent Image Copies	21
4.2.2.3 FlashCopy Image Copies	22

4.2.2.4 FlashCopy and SHRLEVEL CHANGE	22
4.2.2.5 FlashCopy Processing	22
4.2.2.6 FlashCopy Image Copies Considerations	23
4.2.2.7 FlashCopy Requirements and Consideration	23
4.2.3 Tips and Recommendations for Creating Image Copies	24
4.2.4 Copying Multiple Objects with LISTDEF	24
4.2.4.1 LISTDEF	24
4.2.4.2 Using LISTDEF for Partition Selection	24
4.2.5 Parallel Processing Best Practice	25
4.2.6 Stacking Multiple Image Copies on One Tape	26
4.2.7 Deleting Old Copies from SYSCOPY	26
4.2.8 Copying Partitioned Objects	27
4.2.8.1 Sequential and Concurrent Image Copies	27
4.2.8.2 FlashCopy Image Copies	27
4.2.9 Considerations for Image Copies of Indexes	27
4.2.10 Copying Db2 Catalog and Directory	27
4.2.11 Multiple Image Copies	28
4.2.12 Deciding Between Full and Incremental Image Copies	28
4.2.12.1 Incremental and Differential Backup	28
4.2.12.2 Storage Implications for Full Image Copies	29
4.2.12.3 Time Required to Create an Image Copy	29
4.2.12.4 Time Required to Recover Database Objects from Image Copies	29
4.2.12.5 Automatic Switching to Full Copy	30
4.2.13 Creating Inline Copies	30
4.2.14 Copying Image Copies	30
4.2.15 DSN1COPY Considerations	30
4.2.15.1 Typical Usage of DSN1COPY	31
4.3 Determine Image Copies	31
4.3.1 Registration of Image Copies and the COPY Utility	31
4.3.1.1 Full SHRLEVEL REFERENCE Sequential Image Copy	32
4.3.1.2 Full SHRLEVEL CHANGE Sequential Image Copy	32
4.3.1.3 Incremental Sequential Image Copy	32
4.3.1.4 SHRLEVEL REFERENCE FlashCopy Image Copy	32
4.3.1.5 SHRLEVEL CHANGE: Inconsistent FlashCopy Image Copy	32
4.3.1.6 SHRLEVEL CHANGE: Consistent FlashCopy Image Copy	32
4.3.1.7 Concurrent Image Copy	32
4.4 System-Level Backup	33
4.4.1 Important Notes and Considerations	33
4.4.2 Can System-Level Backup Be the Only Backup?	33
4.4.3 Setup Considerations	33

Chapter 5: Db2 Recovery	35
5.1 Recovery Scenario Key Points	35
5.2 Recovery Point Objective	35
5.3 Report on Recovery	35
5.4 Recovery for NOT LOGGED Objects	37
5.5 Recovery from Image Copy	37
5.5.1 Requirements for Image Copies	37
5.5.1.1 Recovery from Sequential Image Copies	38
5.5.1.2 Recovery from Concurrent Image Copies	38
5.5.1.3 Recovery from FlashCopy Image Copies	38
5.5.2 Non-Recoverable Scenarios	38
5.5.3 Recovery Process and Db2 Logs	38
5.5.4 Log Data Sets and Log Ranges	38
5.5.4.1 Applying Log Records	38
5.5.5 Recovering vs. Rebuilding Indexes	39
5.5.6 Recovery Registration to SYSCOPY	39
5.5.7 Recovering Partitioned Objects	39
5.5.7.1 Migrating Data from One Db2 Subsystem to Another	39
5.5.8 OBID Translation	39
5.5.9 Fast Log Apply	40
5.5.10 Recovering the Db2 Catalog	40
5.5.10.1 Recovery from System-Level Backup	40
5.5.10.2 Backout Recovery	41
5.5.10.3 Recovery with a Log Analysis Tool	41
5.5.10.4 Recovery with LOAD RESUME	41
5.6 Point-In-Time Recovery	41
5.6.1 Selecting an RBA for a PIT Recovery	42
5.6.2 PIT Recovery Considerations	42
5.6.3 Recovery Avoidance for Unchanged Objects	42
5.7 Recovering Erroneous Db2 Transactions	43
5.7.1 Reconstructing the Transaction	43
5.7.2 Summary	44
5.8 Dropped Object Recovery	44
5.8.1 Implications of Dropping a Table	44
5.8.2 Dropped Table Recovery Considerations	45
Chapter 6: Recommendations and Guidelines	46
6.1 Determine Db2 Objects Critical to Business	46
6.2 Plan for Evolving Requirements	46
6.3 Leverage the Contents of the Db2 Catalog and Real-Time Statistics	46
6.4 Consider the Recovery SLA	47

6.5 Review, Practice, and Document Db2 Recovery Procedures	47
6.6 Ensure Existing Procedures and Jobs Take Advantage of New Features	47
6.7 Read-Only Db2 Objects	47
6.8 Keep (at Least) the Latest Full Image Copy on DASD	47
6.9 Use Incremental Image Copies	48
6.10 Index Backups	48
6.11 Keep Db2 Log Information	48
6.12 Consider Upgrading Segmented Tablespaces to Partition-by-Growth (UTS) Tablespaces	48
6.13 Consider Partition By Date	48
6.14 Regularly Use the MODIFY RECOVERY Utility	48
6.15 Accidental Dropping of Objects	48
6.16 Automated Techniques to Adjust Backup Strategy	49
6.17 Optimize Recovery with IBM Db2 zPARMs	49
Appendix A: Understanding Database Objects	50
A.1 Obtain Database Information	50
A.2 Obtain Table Information	50
A.3 Obtain Tablespace Information	51
A.4 Obtain Index Information	52
A.5 Obtain Views Information	53
A.6 Obtain Storage Group Information	53
A.7 Obtain Primary and Foreign Key Information	53
A.8 Obtain Check Constraints Information	55
A.9 Obtain LOB Information	55
A.10 Obtain Routine and User-Defined Function Information	55
A.11 Obtain Trigger Information	56
A.12 Obtain Sequence Information	56
A.13 Obtain Recoverability Information	57
A.14 Identify Log Ranges for a Tablespace	57
A.15 Retrieve All Grantees with Granted Privileges	57
A.16 Retrieve All Plans and Packages with Access to a Specified Table	58
Appendix B: Database Management Solutions for Db2 for z/OS	59
B.1 Database Administration Suite	59
B.2 Database Backup and Recovery Suite	60
B.3 Database Performance Suite	60
B.4 SQL Performance Suite	60
B.5 Report Facility	61
Appendix C: Acronyms and Abbreviations	62
Revision History	63
Db2-zOS-RM100; April 29, 2022	63

Db2-zOS-RM100; May 14, 2019 63

Chapter 1: About This Document

The database administrator (DBA) is responsible for creating a backup and recovery strategy for the Db2 for z/OS environment. This handbook provides a guideline for creating and evaluating the backup and recovery strategy and is meant to adequately prepare a DBA when a recovery is required. This handbook also focuses on tips and techniques to consider while formulating procedures and best practices. It is not intended to be read from beginning to end, but rather as a reference for information on Db2 backup and recovery topics.

1.1 Purpose and Audience

This handbook is intended for beginner to intermediate DBAs that design, document, and execute backup and recovery strategies over Db2 objects. However, an experienced DBA can also benefit from the contents of this handbook by jumping directly to the chapters that cover the technical details of relevant utilities. For every topic, special considerations, exceptions, and best practices are included. In most sections, there is no distinction between Db2 versions when discussing various topics. The reader is encouraged to confirm the availability of specific features in the Db2 documentation.

1.2 Authors

Philippe Dubost works on the Mainframe since 2005, and is Offering Manager at Broadcom. In this role, he is responsible for product planning and strategy, presenting and representing the products portfolio to customers and industry analysts, and collecting customer requirements and transforming them into actionable Agile/Scrum stories in the engineering backlog. Philippe is an IBM Champion and regularly speaks at Db2 for z/OS Regional User Groups; he is also the founder of csDUG (a Db2 User Group for the Czech Republic and Slovakia). Philippe has an MBA from Prague University of Economics (VŠE, Czech Republic), and a master's degree in Computer Sciences from ISIMA (France).

Javier Estrada Benavides is a Product Owner at Broadcom for backup and recovery. His previous experience includes 11 years working as a Db2 for z/OS System Programmer. He is the founder of the regional user group MEXDUG in Mexico and a member of many different Db2 user groups. He is also a member of the IDUG Content Committee and has been an IBM Champion for Analytics since 2018.

Michael Kalouš is a Principal Architect, currently responsible for development of the Database Management Solutions for Db2 for z/OS. Michael joined Broadcom in 2009, and after a brief period with the Database Management Solutions for IMS development team, he performed various roles in research and development, with a focus on Db2 for z/OS backup and recovery utilities. Michael has a master's degree in Computer Science and Engineering from CTU in Prague.

Emil Kotrč is a Principal Architect at Broadcom. Emil has been working in the Prague Technology Center since 2005. He started with the resource management products for mainframe and later moved into Db2 for z/OS database management development. Before joining Broadcom, Emil worked in an academic environment and he holds a PhD degree in applied mathematics. He is an active member of the IDUG Content Committee and Emil is also honored to have been added to the global list of IBM Champions for Analytics.

Jan Marek is a Principal Product Owner at Broadcom. He has been working in software engineering since 2005. Starting with mainframe workload automation, he is now enjoying the world of Db2 for z/OS. His main areas of focus are backup and recovery, utilities, and application development. Jan is an IBM Champion for Analytics.

Jacek Rafalak is a Senior System Engineer at Comarch Polska S.A. working as a Db2 specialist for one of the largest mainframe projects in CEE. He is responsible for capacity management and applications performance. He spent 2 years as a Product Owner for backup and recovery. He also worked as a DBA for 17 years. Jacek is an active volunteer in Db2 for z/OS as a board member in the Polish Db2 Users Group. He has been an IBM Champion for Analytics since 2012.

1.3 About the Mainframe Division at Broadcom®

The Mainframe Division at Broadcom® continues to drive the next horizon of open, cross-platform, enterprise solutions. We specialize in DevOps, security, AIOps, and infrastructure software solutions that allow customers to embrace open tools and technologies, make mainframe an integral part of their cloud, and enable innovation that drives business forward. We are committed to forging deep relationships with our clients at all levels. We go beyond products and technology to partner with you in creative ways that support your success.

1.4 Document Structure

The structure of this book is as follows:

- [Chapter 2](#) provides an introduction on the broad topic of Db2 backup and recovery strategies. These topics include implications and points to consider when creating backups for Db2 objects.
- [Chapter 3](#) describes the sources for backup information in Db2 such as its logging environment and the types of copies that are available.
- [Chapter 4](#) describes Db2 recovery, scenarios, and implications for recovering Db2 objects.
- [Chapter 5](#) provides guidelines to help evaluate alternatives for backup and recovery procedures.
- [Chapter 6](#) describes the Database Management Solutions for Db2 for z/OS that help complete DBA tasks related to backup, recovery, and database administration.

A beginner to intermediate DBA diving into Db2 backup and recovery will benefit from [Chapter 2](#). An experienced DBA can jump directly to [Chapter 3](#) to read about the sources that Db2 uses for recovery points.

NOTE: In this guide, Db2 refers to IBM Db2 for z/OS.

1.5 References

The references below may be used in conjunction with this document.

- IBM, *Db2 12 for z/OS Administration Guide*, 2016-2019. (n.d.). Retrieved from https://www.ibm.com/support/knowledgecenter/SSEPEK_12.0.0/nav/src/tpc/db2z_managementofdb2operations.html
- IBM, *Db2 for z/OS Installation and Migration Guide*, 2016-2019. (n.d.). Retrieved from https://www.ibm.com/support/knowledgecenter/SSEPEK_12.0.0/inst/src/tpc/db2z_installingmigratingdb2.html
- IBM, *Db2 for z/OS Managing Performance*, 2016-2019. (n.d.). Retrieved from https://www.ibm.com/support/knowledgecenter/SSEPEK_12.0.0/perf/src/tpc/db2z_managingdb2perf.html
- IBM, *Db2 for z/OS Utility Guide and Reference*, 2016-2019. (n.d.). Retrieved from https://www.ibm.com/support/knowledgecenter/SSEPEK_12.0.0/ugref/src/tpc/db2z_introutilities.html

Chapter 2: Identifying Current Backup and Recovery Procedures

The vast topic of backup and recovery requires contingency plans according to various points of view. Planning presents many trade-offs depending on the nature of the data, applications, and SLAs. However, in real practice backup and recovery is an ever-changing scenario.

Begin with a simple exercise: on the topic of data recoverability, consider the following general questions for every application in the domain, with many additional aspects to also be considered:

- Is the application running 24x7?
- Is the application accessed only or mostly by batch programs or distributed transactions?
- (If applicable) Does the application have a defined production cycle with specific times for batch processing?
- Are the Db2 objects grouped into their own separate databases, or are they grouped by creator (schema)?
- Does this application have a defined group of critical tables?

It is also crucial to consider potential customer requests and even unplanned contingencies. For any given Db2 object or group of objects, consider the following questions:

- Are objects backed up individually or in groups?
- What is the availability of the data during the copy process?
- Can a list of the last backups be generated, with their creation dates and times?
- Can the contents of any arbitrary requested backup be shown?
- Can the contents of a backup be recovered to a different Db2 object so that it can be validated?
- Can you recover from the last backup created?
 - How about the one before it or any other arbitrary backup?
 - Can you recover to a point in time for which you do not have a corresponding backup?
- Is there LOB data? How does this affect the copy or recovery JCLs?
- Are the tables linked by referential integrity constraints?
- If the contents of one or many disk volumes are lost, how does that affect the recovery process?
- How much do the answers to these questions change if one table or a whole database is recovered?

Use these questions as a basic checklist to assess knowledge on data and backup recovery alternatives. In the following sections, the technical aspects, special considerations, and limitations for backup and recovery features and utilities that answer these questions are discussed.

Chapter 3: Introduction to Backup and Recovery in Db2

Before creating a backup strategy for Db2 objects, consider all factors that influence the future and existing strategies and the kinds of possible scenarios. A superior backup and recovery strategy aligns with business processes and cycles to provide recovery points at critical times in the application.

3.1 Prepare an Efficient Db2 Backup Strategy

Backup strategies must simultaneously address the needs of the application and the business. Prepare to comply with audit reports and respond to likely scenarios by first identifying the types of failures your application may encounter as well as the specifics of relevant regulations.

Types of Encountered Failures

To be prepared for different recovery scenarios, create a backup and recovery plan or strategy. Numerous threats can cause system failures that affect data integrity and availability. A plan must consider all of them while addressing any special business rules that apply to specific applications. Various database failures fall into the following categories:

- Application failures occur when an application uses incorrect input, is executed in the wrong order, or when any other unexpected event influences execution. Application failures usually result in corrupted data and require a surgical recovery to undo all unwanted changes.
- Hardware failures usually cause loss of data and might be caused by storage device failure or damage. Whereas these failures were once common, they are now unlikely to occur. However, they still must be considered a potential threat.
- Database subsystem failures occur because of an internal exception within a Db2 subsystem. Usually these failures do not corrupt data and a Db2 restart can re-establish normal operation.
- Human errors are estimated to be the single greatest cause of most failures. The effects of these kinds of failures are similar to application failures.

Regardless of the type of failure, a fundamental component of every solid backup and recovery plan is creating copies of data (also known as image copies). Determining how often image copies are taken, the kind of image copies to take, how long the copies are stored, which options to specify, and which isolation levels to use for utilities to avoid workload interruptions is not an easy task. Before focusing on image copies, preparation work must be done to determine the strategy that best fits business needs.

Before Designing a Backup Strategy

One of the most common mistakes when designing a backup strategy is to consider only the backup procedure itself. A good backup strategy does not need to provide every recovery alternative, but it must be flexible enough to provide the options that are most relevant to the application itself.

Example: It is unrealistic to plan to always recover a table to its latest copy. Frequently, backup plans must also match the backup strategies for an application's data files since a recovery may be followed by jobs that reprocess data.

To maintain a realistic and efficient strategy, keep constant communication with your users and application support teams as they may enforce requirements that determine the rules of your backup plans. Below are a few of the most common examples of such requirements:

- Law requirements: Keeping two separate copies of your objects in different types of media with specific retention periods, as an example.
- SLA requirements: Depending on the application and its nature, maintaining 24x7 availability or having an amount of time for planned outages to create backups may be required.
- Application requirements: This is a broad topic that can go from taking copies on a delimited time frame to having the copies validated by support teams.

- Auditing requirements: At any given time, be ready to provide reports that can list the objects copied, their frequency, locations, and names of copy procedures.

Consider that the application may not have immutable backup requirements, which means that the backup strategy may be insufficient after some period as the application evolves. Prepare to adapt the strategy as required.

3.2 Know The Applications

Consider the applications the business provides to customers, as these programs are the primary consumers of the data. It is not so much the data itself that must be accessible, but rather it is the business applications accessing the data which must be up-and-running for business to operate smoothly. In this sense, it is important to determine which databases support which business applications, and how important each business application is to the business. This ensures a consistent recovery for all affected objects and that the recovery Service Level Agreement (SLA) can be met.

The environment can be complex—but this can be simplified. This approach can be summarized as follows:

- Group business applications into categories classified by the business needs.
- Determine the recovery SLA.
- Map the database objects to application groups.
- Implement and test the plan.

3.2.1 Grouping Business Applications

This task should be done first, prior to defining backup and recovery plans. Interact with application development teams and business leaders to determine and classify the business applications that support the business. This may vary depending on the applications.

Try grouping them into categories, such as critical, high, medium, and low. As an example, the DBA for a bank may use a classification similar to the following:

Table 1: Classification of Business Applications

Category	Description	Business Applications	Associates Programs (or Plans)
Critical	Trading and inter-banking applications	Instant trading, currency exchange valuation, critical inter-bank transfers	PGMABC*, PGMDEF*, PGMIIJK*
High	Customers online operations	ATM withdrawals, immediate transfers, new customers registration	ATM*, PGMLMN*, PGMOPQ*, PGMIRST*
Medium	Customers offline operations	Processing of checks, overnight transfers	CHECK*, PGMUVW*
Low	Internal administrative processing	Payroll, HR processing, hiring	PAYROLL*, PGMXYZ*, PGMZZZ*

3.2.2 Determine the Recovery SLA (Recovery Time Objective, RTO)

After classifying business applications, determine the recovery SLA (also known as Recovery Time Objective, the time it takes to recover an object to a determined point). When speaking to business leaders, it is important to remind them that the tighter the time requirement for the SLA, the more expensive the SLA will be to implement (taking copies more often means higher CPU and storage costs). The goal of this exercise is to come up with figures (timings) like the following:

Table 2: SLA Classification

Category	Time
Critical	30 minutes
High	2 hours
Medium	4 hours
Low	10 hours

3.2.3 Map the Database Objects to Application Groups

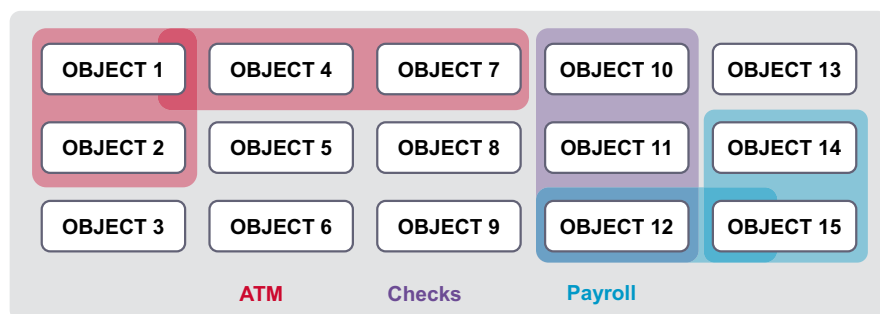
Determine which Db2 objects (in perhaps a Db2 table) support which business applications. The complexity of this task depends on how well the coding standards are respected by application development. Typically, expect that each program interacts with Db2 using a specific PLAN/PACKAGE name (or PLAN/PACKAGE naming convention). Sometimes, however, application developers do not follow a simple PLAN/PACKAGE naming convention. Instead, for instance, each development team may use a specific USERID to bind their plans to a specific business application. Therefore, this task could end up being simple or very complex. Either way, the ultimate outcome is to determine which Db2 table supports which program in the organization. As shown in [Table 3](#), it is likely that a specific Db2 table supports various application programs. For instance, OBJECT_12 contains data that is accessed by both CHECKS* and PAYROLL*.

Table 3: Mapping of Applications

Associated Programs (or Plan/Package) names	Object Name
ATM*	OBJECT_1, OBJECT_2, OBJECT_4, OBJECT_7
CHECKS*	OBJECT_10, OBJECT_11, OBJECT_12
PAYROLL*	OBJECT_12, OBJECT_14, OBJECT_15
PGMABC*	OBJECT_3, OBJECT_5
PGMDEF*	OBJECT_5, OBJECT_8, OBJECT_9
PGMXYZ*	OBJECT_13, OBJECT_14,
...	...

3.2.4 Review

Combining the information collected in the three previous tables, a picture begins to develop of the Db2 tables grouped by the business applications using them and based on each Recovery SLA, as shown in [Figure 1](#). Some Db2 Tables belong to two different Recovery SLA categories: OBJECT_12 is in both the critical group and the high group. When implementing the Db2 backup for this table, opt for the most critical Recovery SLA.

Figure 1: Grouping of Database Objects

For the sake of keeping this section readable, the example is simplistic. More often than not, the Db2 tables and the objects associated with the tables must be recovered for the data to be available for the business application. A strong example of this is the associated indexes that must be recovered (or rebuilt) with the base table. Within each group, it is also necessary to consider the Db2 built-in relationships, such as RIs or triggers, and include the respective target Db2 object in the group.

3.3 Introduction to Recovery

When a failure occurs, the DBA typically uses image copy backups and the Db2 log to recover the data. Note that the recovery of a database can be a very complex task that is difficult to manage and prone to errors. Recovery involves much more than simply restoring an image of the data from the last image copy. Often a recovery process involves restoring data from backups and then re-applying the correct changes that occurred to that data, in the correct sequence, to the state at (or before) the time of the failure. If recovery is required, determine which recovery options are possible and which resources are available. There are numerous recovery options and scenarios that must be taken into consideration. Below is a sampling:

- Recover a DASD volume
- Recover a storage group
- Recover a bootstrap data set
- Recover log data sets
- Recover Db2 subsystems
- Recover SYSTEM (based on IBM BACKUP SYSTEM)
- Recover (accidentally dropped) table
- Recover a page (or range)
- Recover Db2 catalog and directory
- Recover to current
- Point-in-time recovery
- Recover due to an erroneous transaction

The implementation and regular simulation of a backup and recovery plan significantly helps businesses react quickly when a failure occurs and decide quickly on what to do in the event a recovery is necessary. To create a solid backup and recovery strategy, and to be able to perform test scenarios, it is essential to understand the key components of backup and recovery in Db2:

- Image copy
- Bootstrap data set (BSDS)
- Db2 log
- Catalog and directory

Chapter 4: Db2 Logging and Creating Data Backups

To help determine recoverability points for Db2 objects and executing recoveries, Db2 contains specific structures to identify consistency points and whether a point in time is a valid recovery point. Knowledge of the contents of these structures affects the recovery alternatives, while proper management provides improved response times for transactions.

4.1 The Db2 Logging Environment

Db2 records all changes to the data in the DB2 log. Inserts, updates, and deletes are logged, as well as internal data to ensure the consistency and durability of the data. The logging environment is therefore a crucial component that maintains the data consistency and durability to ensure compliance with the ACID rules.

The initial logging environment is established during the installation of Db2. Once the initial set up is complete, monitor and change options using the commands and utilities. The key components of the logging environment are checkpoints, log buffers, active logs, archive logs, and the bootstrap data set (BSDS). These components are discussed further below.

4.1.1 Bootstrap Data Set

The bootstrap data set (BSDS) is a VSAM key-sequenced data set that registers information about the Db2 logs of a Db2 subsystem member. It holds the names of active and archive log data sets together with the time range specification relevant to each log. Db2 automatically allocates two copies of BSDS during installation. The total space required is about 30 MB.

When the active log is successfully offloaded, a new archive log data set is recorded in the BSDS and a copy of BSDS is created. BSDS is used during a recovery process to find all available archive logs for a specific log range. Db2 also writes a checkpoint record to the BSDS data set. If the Db2 subsystem is a member of a data-sharing group, BSDS also contains information about other BSDS data sets in the group.

The change log inventory utility (DSNJU003) can be used to update the BSDS data set, add or delete active or archive log data sets, add or delete checkpoint records, and more. However, Db2 must be stopped to execute this utility. To add additional log data sets dynamically, use the SET LOG NEWLOG command.

The DSNJU004 utility enables the ability to display all parts of the BSDS.

4.1.2 Active and Archive Log Data Sets

The Db2 log is contained in active and archive log data sets. Log records are externalized from log buffers to active log data sets first. When full, the active log data set is offloaded to an archive log data set so it can be re-used again. Both processes, externalizing log buffers and active log data set offloads, may affect the whole Db2 subsystem. If either of the processes fail for any reason, Db2 stops processing. To mitigate this risk, use dual logging—especially on production subsystems.

When dual logging is used, Db2 writes the same log record to two active log data sets and produces two copies of the archive log during offloading. This removes a single point of failure. If one log data set is inaccessible, the other can be used. Dual logging for active log and archive log data sets can be set by TWOACTV and TWOARCH DSNZPARMS.

Db2 may need to read the log during certain situations such as rollback, restart, or recovery processing. Note that reading the active log is much faster than reading the archived log. The size of the active log data set significantly affects Db2 performance. Keep at least one day of activity in the active log data sets. To control how often active log data sets are copied to the archive log, use the ARCHIVE LOG FREQ field on the DSNTIPL installation panel. The UPDATE RATE field provides an estimate of how many database changes (inserts, updates, and deletes) are expected per hour. Both fields together determine the size of the active logs. The quantity of active log data sets is determined by the NUMBER OF LOGS field on the same DSNTIPL panel.

To control how much of the log data is kept archived, use the DSNZPARM MAXARCH. This parameter determines the maximum number of archive log data sets that are recorded in the BSDS data set. If applications require recovery, ensure the recovery process can go back to a point in time sufficiently in the past. Be sure to define an adequate size and number of archive log data sets.

NOTE: Although a high number of archive log data sets recorded in BSDS can be kept, a site's retention policies will erase them over time. Ensure that your retention policies match the business needs so that, in the case of a long recovery, Db2 can find all related archive logs required to satisfy your recoveries.

Tips and recommendations:

- Use dual logging and keep active logs in separate volumes.
- Keep archive logs on separate volumes (physical disks) from the active logs so the active log I/O does not need to share resources when Db2 offloads to the archive log.
- Keep at least one day of activity in active log data sets.
- Do not use tape compression for archive log tapes.

The print log map utility (DSNJU004) can be used to obtain information about log data sets such as log data set name, log RBA association, log LRSN for both copy 1 and copy 2 of all active and archive log data sets, active log data sets that are available for new log data, contents of the checkpoint queue, archive log command history, and more.

4.1.3 Log Buffers

It has already been mentioned that Db2, as part of the transaction processing, writes log records to the Db2 log. Log records are written into log buffers first and then the whole log buffer is externalized to the Db2 active log. More precisely, there are two types of log writes: asynchronous and synchronous.

- Asynchronous writes occur when a transaction updates data. The images before and after the update are written to the log buffer. If there is no log buffer available, the application must wait for one to become available.
- Synchronous writes occur at commit time. This write causes the log buffers to get externalized to the Db2 log. If the log data set is busy the request is queued until it is processed. The application must wait until this write is complete.

Pages for log buffers are permanently fixed in real storage, so make the log buffers as large as feasible. A large size may decrease the number of forced externalizations of the log buffer to the Db2 log because additional buffers are unavailable, and it can also reduce the number of times an application has to wait for a log buffer during an asynchronous write. The log buffer's size defaults to 4 MB and can be changed by OUTBUFF DSNZPARM to a maximum size of 400 MB.

IFCID 0001 trace records may be collected to determine the number of times the log buffer was full and caused a log record to wait for an I/O. The number of waits is indicated in the QJSTWTB field. When planning to use any data replication that reads IFCID 306 trace records, consider increasing the size of the log buffer. The log write is part of the transaction process and log buffers that are too small might significantly decrease the performance of the whole Db2 subsystem.

NOTE: The OUTBUFF DSNZPARM value cannot be changed online. To change its value, follow the site's procedures for DSNZPARM changes that require a Db2 restart.

4.1.4 Checkpoints

Checkpoint log records contain information about all incomplete units of recovery and the page set summary. At startup, Db2 uses this information to reconstruct what was occurring at checkpoint time and to reduce the restart time. Checkpoints are used not just by Db2, but also by many vendor tools, so frequent checkpoints are important.

Checkpoints can be triggered by elapsed time, the number of log records written, or both. This can be set up by the CHKTYPE, CHKFREQ, CHKLOGR, and CHKMINS DSNZPARM parameters. It can also be dynamically changed by the SET LOG command and its LOGLOAD and CHKTIME parameters. Specify a checkpoint interval of 3 minutes during busy hours of the day and/or night.

Regardless of the defined checkpoint frequency, Db2 creates checkpoints at other events, for instance when an active log is offloaded to the archive log, or after a successful restart of Db2. To display the configuration for checkpoint intervals, use the DISPLAY LOG command.

NOTE: To trigger a checkpoint at any desired time, execute one of the following options:

- SET LOG command with option CHKTIME(0)
- SET LOG command with option LOGLOAD(0)

Upon execution, the subsystem confirms that a checkpoint has been initiated and also displays the current settings for checkpoint intervals. A sample output can be observed next.

```
DSNJ333I  <ssid> DSNJC009 SYSTEM CHECKPOINT INITIATED
DSNJ339I  <ssid> DSNJC009 SET LOG COMMAND COMPLETED, LOGLOAD (500000)
DSN9022I  <ssid> DSNJC001 '-SET LOG' NORMAL COMPLETION
```

If the storage administrator is executing a global LPAR backup on a disk level and Db2 activity cannot be suspended for this task, consider triggering a checkpoint before the backup begins. A checkpoint RBA is considered as a valid recovery point when used as a target for a RECOVER utility.

A complete record of all checkpoints taken in the past can be retrieved by the print log map utility (DSNJU004), where the following information regarding checkpoints can be found:

- Time and date of checkpoint
- Begin RBA
- End RBA

Below is a sample of how this information is reported from utility DSNJU004.

```

                CHECKPOINT QUEUE
                08:48:12 JANUARY 16, 2019
TIME OF CHECKPOINT      03:37:35 JANUARY 13, 2019
BEGIN CHECKPOINT RBA    0000000000575DE8FC5C
END CHECKPOINT RBA      0000000000575DF19CC2
END CHECKPOINT LRSN     00E6525CCE1F19000000
```

4.1.5 Logging in a Data Sharing Environment

In a data sharing environment, each member manages its own active and archive log data sets and the BSDS data set. In addition, each member's BSDS data set contains information about other members' BSDS and log data sets. The Shared Communication Area (SCA) in the coupling facility contains information about all members' BSDSs and log data sets.

In the event of a recovery, Db2 accesses the logs of other Db2 subsystems in the group and merges them in a sequence. To determine this sequence, each log record of a data-sharing member is identified by a timestamp. This timestamp is called the log record sequence number (LRSN) and is based on the store clock (STCK) value. As LRSN is basically a timestamp, it is possible that multiple log records that change the same data page may be assigned duplicate LRSN values. To determine whether the log record should be applied, its LRSN is compared to the LRSN value of the data page.

ATTENTION: Consider the impact of archiving logs in a data sharing environment. During recovery, at least one tape unit is required for each member whose archived log records are to be merged. The archived logs are recalled one at a time.

4.1.6 SYSLGRNX Directory Table

Located in the Db2 directory database, the table SYSLGRNX plays a crucial role for the general process of a recovery. It contains the log ranges that represent the time a tablespace was open for updates and it also records the same for indexes defined with COPY YES. This way, in the event of a recovery, Db2 avoids reading the entire recovery log and only has to read the specific log records required for the object to be recovered.

NOTE: If the site uses indexes defined as COPY YES or if a Data Sharing environment is in use, be mindful of the size of this table as it will require a considerable amount of space.

4.1.7 Closing and Opening Data Sets

An important factor to consider for performance is the quantity of open data sets for read/write access. This quantity also impacts the cost of checkpoints and logging. In normal operation, Db2 defers closing data sets until necessary to avoid additional I/O operations and provide opportunity for other applications to reuse objects without having to reopen them.

It is possible to influence how data sets are closed when reaching the limit of open data sets. For this task, leverage the use of parameter CLOSE at tablespace creation. The possible values are as follows:

- CLOSE YES: Tablespaces defined with value YES are eligible for closing.
- CLOSE NO: Tablespaces defined with value NO are only eligible for closing after all objects defined with CLOSE YES are closed.

4.1.8 Reducing Log Records by Converting to Read-Only Page Sets

With more precise SYSLGRNX entries, RECOVER utilities require fewer log records to process and the business benefits from more effective and therefore faster recoveries. There are two subsystem parameters that control when an infrequently updated object can transition to read-only state, which closes the corresponding entry in SYSLGNX and creates more accurate records:

- RO SWITCH CHKPTS or PCLOSEN: This parameter counts the consecutive checkpoints after a page set was last updated.
- RO SWITCH TIME or PCLOSET: This parameter counts the elapsed time after a page set was last updated.

Their respective default values suffice, unless a performance problem is detected by the amount of transitions to read-only (and this, in return, increases the updates to SYSLGRNX), in which case parameter PCLOSET must be increased.

Switching from read-write to read-only externalizes updated pages from buffer pool to disk.

NOTE: Objects defined as NOT LOGGED that are not in use transition to read-only after a Db2 checkpoint or one minute after the last update/commit.

4.1.9 Summary

The Db2 log ensures database consistency, plays a vital role in Db2 recovery, and stores the necessary information to restart Db2 after termination. The logging environment has a high impact on the operation of the whole Db2 subsystem.

The Db2 log is not only used by Db2, it is leveraged by many tools that use the information stored in the Db2 log for the following:

- Reporting on past Db2 activity
- Auditing data and object changes
- Recovering Db2 data
- Replicating Db2 activity

4.2 Image Copy Considerations

Image copies are a fundamental element of backup and recovery. This section discusses the types of image copy that exist and the options available upon creation. This section also describes some hints and tips for using image copies.

4.2.1 Availability of Copied Objects

When creating image copies, decide whether the application programs can access the database object being copied.

The decision is based on the importance of the database object for your business needs. Does the database object need to be up and running 24x7? Can a short downtime be afforded? Can a maintenance window with limited availability be tolerated? How important is the time needed for recovery of the database object? After considering these questions, choose one of the SHRLEVELs shown in [Table 4](#):

Table 4: Image Copy SHRLEVELs

SHRLEVEL	Impact on Applications	Impact on Copy Creation	Impact on Recovery
NONE	Applications have no access. Executed queries and statements will time-out due to unavailability of the database object.	Fastest creation. Image copy will not contain any uncommitted data.	No log records are required to ensure consistency of the data when recovering to the copy.
REFERENCE	Applications can read data from the database object, but cannot change the data (INSERT, UPDATE, DELETE). SQL statements will time-out due to read-only availability of the database object.	Almost as fast as SHRLEVEL NONE. Image copy will not contain any uncommitted data.	No log records are required to ensure consistency of the data when recovering to the copy.
CHANGE	Applications have full access (INSERT, UPDATE, DELETE).	Slightly slower than SHRLEVEL REFERENCE. Image copy can contain uncommitted data.	Log records are required to ensure consistency of the data when recovering to the copy. This requires additional time when recovering the database object.

4.2.1.1 Availability Recommendations and Considerations

- If uncompromised 24x7 availability is required, choose SHRLEVEL CHANGE.
- If a maintenance window with limited availability is possible, choose SHRLEVEL REFERENCE.
- When a database object is in the COPY PENDING status, create a SHRLEVEL REFERENCE image copy to re-establish recoverability of the object.
- Creating a SHRLEVEL CHANGE image copy of objects defined with the LOG NO option is not allowed.
- Consider using SHRLEVEL REFERENCE FlashCopy image copies as an alternative to traditional SHRLEVEL REFERENCE image copies. Thanks to the nature and benefits of FlashCopy technology, the portion of time when the object is non-updatable is reduced and the resulting outage lasts only a few seconds.

4.2.1.2 Choosing Between SHRLEVEL REFERENCE and CHANGE Image Copies

Generally, the key factor when choosing the appropriate SHRLEVEL mode comes from the availability of the data while the copy is generated. However, this decision can drastically change the time to recover (RTO) and, in many cases, the estimated time for RTO can take a higher priority.

An important fact to consider: A backup process is a timed and planned process whereas a recovery scenario is most likely an emergency scenario. For instance, in some applications and environments, the response time in an emergency will determine the overall end-user satisfaction and will outweigh the unavailability of data during backups. Therefore, it is important to keep a clear written agreement of the priorities on backup and recovery operations.

As mentioned in the table from the previous section, SHRLEVEL CHANGE image copies require the use of log records when used for a RECOVER utility. However, an image copy made with SHRLEVEL REFERENCE does not have this condition since a copy made with SHRLEVEL REFERENCE does not include information on uncommitted data.

4.2.1.3 DD Statements

For allocation of the image copy data sets, use JCL DD statements and instruct the copy utility to use them. For each image copy data set, a JCL DD statement is needed. Although DD statements are straightforward, they also lack flexibility when copying multiple database objects, multiple partitions, or if multi-tasking is involved.

Example: Create daily copies of all three segmented tablespaces in the DALPHA database, with the date as December 3rd. Syntax example:

```
//COPY1 DD DISP=(,CATLG),DSN=DALPHA.REPORTS.P0000.T1203
//COPY2 DD DISP=(,CATLG),DSN=DALPHA.FILES.P0000.T1203
//COPY2 DD DISP=(,CATLG),DSN=DALPHA.DEPTS.P0000.T1203
//COPY EXEC PGM=COPY
//SYSIN DD *
COPY TABLESPACE DALPHA.REPORTS COPYDDN(COPY1)
COPY TABLESPACE DALPHA.FILES COPYDDN(COPY2)
COPY TABLESPACE DALPHA.DEPTS COPYDDN(COPY3)
```

4.2.1.4 Dynamic Allocation

Dynamic allocation of the image copy data sets provides more flexibility than DD statements when copying multiple database objects. Hundreds of image copy data sets can be created using a single dynamic allocation definition.

Dynamic allocation typically allows the following:

- Data set name expressions with substitution symbols
- Space definitions
- Data set attribute definition
- Switching (different definition for small and large data sets resolved at runtime)

4.2.1.5 TEMPLATE Utility

The TEMPLATE utility is a standard tool for dynamic allocation for Db2 for z/OS. When constructing a job step that creates image copies, proceed as follows:

1. Create the syntax for the TEMPLATE utility to define templates.
2. Create the syntax for the copy utility where you specify template names instead of JCL DD names.

Example: Create daily copies of all three segmented tablespaces in the DALPHA database. Example syntax:

```
//COPY EXEC PGM=COPY
//SYSIN DD *
TEMPLATE TMPLT1 DSN(&DBNAME..&TSNAME..&PA..&MO..&DA) .
COPY TABLESPACE DALPHA.% COPYDDN(TMPLT1)
```

4.2.2 Types of Image Copies

With Db2, the following image copy types can be created to back up data:

- Sequential image copies to create a full, incremental, or differential backup using basic access methods.
- Concurrent image copies to create a full backup with enhanced performance and availability.
- FlashCopy image copies to create a full backup using the advanced capabilities of contemporary storage devices.

4.2.2.1 Sequential Image Copies

Sequential image copies are the foundation of Db2 backup and recovery. A sequential copy is a data set that holds the same data as the database object (or its changed parts) at the time the image copy is created. Sequential image copies are easy to create, do not require special hardware or setup for storage, can be stored directly to tapes, and can be processed with basic access methods such as BSAM or QSAM.

The copies are called sequential for their sequential nature; they are meant to be fully processed sequentially from beginning to end. It is difficult (though not entirely impossible) to locate specific data in the image copy using direct access.

4.2.2.2 Concurrent Image Copies

With concurrent image copies, the extended functions of your storage subsystem can be leveraged to create SHRLEVEL REFERENCE image copies that require only a few seconds of read-only availability. SHRLEVEL CHANGE copies can also be created with the concurrent copy.

A COPY utility creates a SHRLEVEL REFERENCE concurrent image copy as follows:

1. Establishes a quiet point.
2. Changes the database object status to read-only (RO).
3. Calls DFSMSdss to initialize the concurrent copy. The DUMP command with the CONCURRENT keyword is executed by DFSMSdss.
4. Restores full access to the database object (RW).
5. Registers the image copy to the SYSIBM.SYSCOPY table as a concurrent image copy (ICTYPE='F', STYPE='C', or 'J').

The resulting image copy data set is in the format that the DFSMSdss DUMP command creates.

When using concurrent image copies, consider the following:

- Hardware is required that supports concurrent copy.
- Multiple database objects can be copied to a single concurrent image copy. This allows creation of compact multi-data set and multi-object backups.
- For page sizes greater than 4 KB, SHRLEVEL CHANGE concurrent copies cannot be created if the CI size of the Db2 VSAM data set does not match the page size.

Concurrent image copies are suitable for creation of consistent and compact multi-data set or multi-object backups with minimum down-time.

NOTE: Data from a concurrent image copy cannot be unloaded and the COPYTOCOPY utility cannot be executed on this type of copy to create an unloadable copy.

4.2.2.3 FlashCopy Image Copies

FlashCopy technology is a storage system function that is invoked by software tools such as DFSMSdss COPY. To use FlashCopy technology, the hardware must support FlashCopy—such as IBM DS 8xxx Enterprise Storage Server or any other vendor hardware with the licensed microcode.

The basic idea behind FlashCopy technology is to create a point-in-time instant snapshot of the data on the disk. The hardware function establishes a relationship between the source and target volumes that together form a FlashCopy pair. Once this pair is established, the hardware becomes a background process, but the call is returned to the FlashCopy caller and the source objects are available for read/write access for both target and source. Except for the tiny fraction of time spent establishing the relationship, this allows for instantaneous copies with full data set availability.

Below is an example of FlashCopy image copy syntax:

```
COPY TABLESPACE dbname.tsname FLASHCOPY YES
```

4.2.2.4 FlashCopy and SHRLEVEL CHANGE

There are two options of SHRLEVEL CHANGE FlashCopy image copies:

- **Not consistent:** By specifying SHRLEVEL CHANGE and FLASHCOPY YES, a fuzzy FlashCopy image copy is created. The creation time for the copy is very fast because there is no drain and no background processing. The trade-off compared to SHRLEVEL REFERENCE copies is that since the copy is fuzzy, log apply must be performed during the recovery. On the other hand, there is almost no outage and, except for the small fraction of time when the relationship between the target and source is being established, the object is available at all times. This option is likely the solution if consistent copies are not required. Syntax example:

```
COPY TABLESPACE dbname.tsname FLASHCOPY YES SHRLEVEL CHANGE
```

- **Consistent:** Specifying SHRLEVEL CHANGE and FLASHCOPY CONSISTENT tells the Db2 COPY utility to perform processing in the background to make the copy consistent. Any uncommitted data is removed from that copy. This involves the COPY utility taking three additional steps (LOGAPPLY, LOGSCR, and LOGUNDO) after the DFSMSdss finishes its processing and returns back. The final image copy includes only the committed data, but this process can be very slow depending on the number of transactions running during the image copy processing. If this type of consistent copy is used by the RECOVER utility, extra phases must be performed as well—basically, RECOVER must first re-apply all the removed transactions.

NOTE: Given that FLASHCOPY CONSISTENT requires the log to back out uncommitted work, the use of FLASHCOPY CONSISTENT is not compatible with objects defined as NOT LOGGED.

4.2.2.5 FlashCopy Processing

After the FlashCopy pair is established, the copy process begins in the background. This process copies all tracks to the target as they appear in time zero—meaning when the pair was established. If there are any changes to the source data, the block is first copied to the target and only then can the source data be updated. This is all driven by the storage server microcode.

Once all the tracks are copied, the relationship between the source and target is withdrawn.

There are two main options for FlashCopy copies - COPY and NOCOPY.

- The COPY option copies all physical blocks from the source and is the default option. This is also called a physical copy because by the end of the copy process there is a whole new copy. The COPY utility uses this option.
- The NOCOPY option copies only the parts that have changed in the source since FlashCopy established the relationship. For the NOCOPY option, FCNOCOPY must be specified in DFSMSdss. The CHECK SHRLEVEL CHANGE utility uses the NOCOPY option.

4.2.2.6 FlashCopy Image Copies Considerations

The following considerations must be made when implementing FlashCopy:

- FlashCopy image copies are compatible with Db2 utilities and are recorded in the SYSIBM.SYSCOPY catalog table together with traditional image copies.
- FlashCopy image copies are based on the data set level and cannot be incremental: even if FULL NO is specified, only full image copies are created. FlashCopy image copies are written to a VSAM data set and are always logged. Due to their nature, the copies are per each partition or data set.
- When a single partition or piece of an object is copied, a single row is added to the SYSCOPY table and the DSNUM=0. If there are multiple partitions or pieces of objects, a special row with a corresponding DSNUM value is added for each partition or piece, while the whole object is represented with a row that has the DSNUM=0.
- The DDNAME for a FlashCopy image copy is specified in FCCOPYDDN. Because of the data set level nature of the copies, the uniqueness of the copy data sets must be guaranteed (meaning by using the &DSNUM variable in the TEMPLATE specification).
- FlashCopy image copies can be of the following types:
 - SHRLEVEL REFERENCE
 - SHRLEVEL CHANGE (not transaction consistent)
 - SHRLEVEL CHANGE (transaction consistent)
- The following options are ignored for FlashCopy image copies: PARALLEL, TAPEUNIT, CHECKPAGE, and SYSTEMPAGES.
- The following Db2 utilities may benefit from FlashCopy: COPY, REORG, LOAD, REBUILD INDEX, REORG INDEX, and CHECK.
- The operational behavior of FlashCopy can be configured in the following Db2 ZPARMs:
 - FLASHCOPY_COPY: This specifies whether the FLASHCOPY option for the COPY utility is used by default. Possible values are YES or NO, with NO being the default. Note that there are similar ZPARMs for other utilities as well, like FLASHCOPY_LOAD and FLASHCOPY_REORG_TS.
 - FCCOPYDDN: This is the default template for the image copy if one is not provided in the utility control statements. Observe that the copies are created on a data set level and must be unique for each data set. The default value for the ZPARM is HLQ.&DB..&SN..N&DSNUM..&UQ.
 - COPY_FASTREPLICATION: This defines whether the COPY uses fast replication (meaning FlashCopy) or the traditional copy process. The values are PREFERRED and REQUIRED, with PREFERRED being the default. With PREFERRED, the COPY utility attempts to use fast replication whenever possible. If it is not possible, the utility falls back to traditional methods like IDCAMS REPRO. If REQUIRED is specified, the COPY utility fails when fast replication cannot be used. The NONE option means that the COPY utility only uses traditional copy methods. Similarly, the CHECK_FASTREPLICATION option exists for the CHECK utility.

4.2.2.7 FlashCopy Requirements and Consideration

FlashCopy 2 is required to fully utilize FlashCopy technology. Without FlashCopy 2, Db2 uses traditional methods like IDCAMS REPRO, and the elapsed time may be significantly higher. From a security perspective, a user that is about to create a FlashCopy image copy must have the authority to execute the DFSMSdss COPY command that drives the copy processing.

FlashCopy also requires that all data sets be managed by the storage management subsystem. In addition, there are some restrictions that can result in traditional and slower I/O operations, such as:

- FlashCopy 2 disk volumes are not available.
- The source tracks are already the target of a FlashCopy operation or the target tracks are the source of a FlashCopy operation (meaning there can be only one utility running FlashCopy on a source or target object).
- The maximum number of relationships between source and target tracks is exceeded for the FlashCopy operation.
- The input data set is less than one cylinder. In this case, if FlashCopy is not used, the copy is performed by IDCAMS.

Verify that FlashCopy (or so-called fast replication) has been used by reviewing the output of the DFSMSdss COPY operation. Search for message ADR806I, for example:

```
ADR806I (001)-T0MI (03), DATA SET EMC7.CPS.D2017.S2017A.I0001.A001.FC COPIED USING A FAST
REPLICATION FUNCTION
```

4.2.3 Tips and Recommendations for Creating Image Copies

Db2 offers many tools to assist in the task of creating and managing image copies of Db2 objects. Become familiarized with these tools to gain flexibility in the ways to execute a backup strategy for better maintainability and faster execution of backup jobs. A solid backup management plan can help determine the recovery options that can be offered to users.

4.2.4 Copying Multiple Objects with LISTDEF

A copy utility allows copying of multiple objects. Specify multiple objects with the following:

- Enumerating the database objects in the syntax
- Specifying a predefined list of objects created with the LISTDEF utility

Some vendor utilities can provide other means for copying multiple objects (examples include wildcarding or special syntax).

4.2.4.1 LISTDEF

LISTDEF is a utility shipped with Db2. This utility allows the specification of a predefined list of objects based on selection criteria. LISTDEF allows for wildcarding, automatic inclusion of associated objects, and so on. A list is created and valid for one job step.

Below is an example of the syntax for copying multiple objects with the LISTDEF utility:

```
LISTDEF ASSETS INCLUDE TABLESPACE DASSETS.*
EXCLUDE TABLESPACE DASSETS.TMP*
INCLUDE INDEXSPACE DASSETS.*IX
EXCLUDE INDEXSPACE DASSETS.TMPIX*

COPY LIST ASSETS COPYDDN(TMPLT1)
```

Creating copies of multiple objects is best combined with dynamic allocation of the image copy data sets.

4.2.4.2 Using LISTDEF for Partition Selection

Leverage the options for LISTDEF to work on a partition level. Several options are available:

- Select an individual partition.
- Exclude selected partitions.
- Select a specific partition number and apply this criteria for all tablespaces that match a given name mask.

Below are a few examples:

- To include all partitions from all matching tablespaces, where each partition creates a separate match entry, and also include all non-partitioned tablespaces:

```
LISTDEF SEL1 INCLUDE TABLESPACE MYDB.* PARTLEVEL
```

- To include partitions 2, 3, and 4 of tablespaces within database MYDB.

```
LISTDEF SEL1 INCLUDE TABLESPACE MYDB.* PARTLEVEL(2:4)
```


- To include all partitions from tablespace MYDB.TS01 except partition 8.

```
LISTDEF SEL1 INCLUDE TABLESPACE MYDB.TS01 PARTLEVEL
          EXCLUDE TABLESPACE MYDB.TS01 PARTLEVEL(8)
```

- To include all non-partitioned tablespaces within database MYDB.

```
LISTDEF SEL1 INCLUDE TABLESPACE MYDB.* PARTLEVEL(0)
```

NOTE: LISTDEF matching is only done once per job execution. If a new partition is created during a job execution but after this job evaluated its LISTDEF, the new partition will not be included in the job even if it matches the LISTDEF filter.

Implications for RBAs In SHRLEVEL REFERENCE and SHRLEVEL CHANGE Copies

It is a common practice to use LISTDEF when creating copy JCLs. In this case, the list contains a group of objects or an entire database and is copied as a unit or single group. Using SHRLEVEL REFERENCE and SHRLEVEL CHANGE image copies requires special consideration due to the effect on copy RBAs, as shown below:

	SHRLEVEL REFERENCE Copy	SHRLEVEL CHANGE Copy
Copy RBA meaning	Copy RBA represents the end of the COPY phase.	Copy RBA represents the beginning of the copy processing.
Effect on the list of objects	All objects have the same RBA for this copy invocation.	All objects have different RBAs for this copy invocation.
Usage effect for RECOVER using LISTDEF	The copy RBA can be used for all objects in the list.	Each element in the list has a different RBA.
Final state of the recovered tablespaces	Read Write	Possibility of the following states depending on your data types: CHKP, ACHKP, AUXW.

NOTE: When recovering an object using an image copy older than the most recent copy, identify it by its data set name or by the RBA recorded in SYSCOPY table.

If copies are identified by their RBA, observe that if a LISTDEF is used to execute a SHRLEVEL CHANGE image copy, each object must have a different copy RBA. This particularity becomes especially relevant for recovering from old copies. For this situation, consider taking SHRLEVEL REFERENCE copies at strategic points for the application.

4.2.5 Parallel Processing Best Practice

When creating image copies of multiple objects or multiple partitions, parallel processing can improve the performance of the copy process. Parallel processing is limited to sequential image copies only.

The following guidelines are best practice for parallel processing:

- Parallel processing provides best performance when database objects or partitions are stored on different volumes, so the copy utility is not competing for one DASD.
- When the copies are stored on tapes, keep the number of tape units that are dynamically allocated for the image copies under control. The copy utility allows you to limit the number of allocated tape units to a specific number.

4.2.6 Stacking Multiple Image Copies on One Tape

When creating image copies, it is possible to stack multiple copy data sets on one tape. Data set stacking can improve performance during data recovery by decreasing the number of tapes that must be accessed to retrieve the image copies.

When using JCL DD statements to allocate image copy data sets, specify REF and RETAIN in the volume parameter. When using the TEMPLATE utility, specify STACK YES when defining a template.

When using stacking, parallel processing is limited because one task must write all data sets stacked to one tape. Below is an example of the syntax:

```
//SYSA1101 DD DSN=TEST.DB22.TS22.A101,DISP=(NEW,CATLG,KEEP) ,
// UNIT=VTAPE,VOL=(,RETAIN),LABEL=(1,SL)
//SYSB1101 DD DSN=TEST.DB22.TS22.B102,DISP=(NEW,CATLG,KEEP) ,
// UNIT=VTAPE,VOL=REF=*.SYSA1101,LABEL=(2,SL)
//*
//SYSIN DD *
COPY
    TABLESPACE DB22.TS22
    COPYDDN(SYSA1101)
    TABLESPACE DB22.TS22
    COPYDDN(SYSB1101)
    SHRLEVEL CHANGE
```

4.2.7 Deleting Old Copies from SYSCOPY

Creating image copies and using them for recovery is only one part of an overall backup and recovery strategy. Because storage devices do not have infinite capacity, decide what image copies to keep and for how long. This decision depends on the nature of the business, and the legislation that regulates the business.

To delete no-longer-needed image copies, a SYSIBM.SYSCOPY maintenance utility (MODIFY utility) is required that deletes any copies that meet the specified criteria. Some examples of the criteria:

- Delete copies older than *m* days.
- Keep the *n* most recent image copies.
- Keep only copies needed for recovery of a database object.
- Delete SYSCOPY records that refer to image copy data sets that no longer exist.
- Other complex criteria specified using a custom SQL.

Additionally, set up MODIFY utilities to delete image copy data sets together with SYSIBM.SYSCOPY records, delete obsolete SYSIBM.SYSLGRNX records, or create an image copy when all copies have been deleted.

Leave deletion of no-longer-needed copies in the hands of the Storage Management Subsystem (SMS). SMS can migrate old data sets to tapes or delete them and the MODIFY utility can delete the SYSCOPY records that refer to the deleted data sets.

4.2.8 Copying Partitioned Objects

4.2.8.1 Sequential and Concurrent Image Copies

Backup and recovery of partitioned objects requires deciding whether the backup and recovery will be performed on the object level or partition level.

Creating an image copy on partition level allows quick, surgical recovery of a single partition that reduces the down-time of the database object. However, a full backup of the whole tablespace can consist of up to 4096 separate data sets. To keep the image copy data sets under control, first establish a naming convention and use dynamic allocation. Creating copies on partition level also allows process tasks to be performed in parallel, which increases performance.

Creating image copies on an object level provides a compact, single data set backup. It is a viable solution when the object has only a few partitions. Obviously, multi-tasking is limited, because there is only one output data set that must be written sequentially.

Some copy utilities allow finer control over the process. For example, store four partitions per data set and keep the right balance between the number and size of the image copy data sets, multi-tasking, speed of recovery, and availability of the partitions during the recovery.

4.2.8.2 FlashCopy Image Copies

When creating flash copies, create the backup on partition level due to the nature of these image copies. A FlashCopy image copy is a snapshot of the underlying VSAM data set.

4.2.9 Considerations for Image Copies of Indexes

If an index must be recovered, it can either be restored from a full image copy or rebuilt from data stored in the associated tablespace.

When creating image copies of indexes, consider the following:

- Define the index with the COPY YES option.
- Only full image copies of indexes can be created. Creation of incremental image copies is not supported by Db2. However, there are vendor utilities that allow incremental image copies of indexes.
- Always copy indexes together with the associated tablespaces to have a common recovery point.

4.2.10 Copying Db2 Catalog and Directory

For copying all Db2 catalog and directory tablespaces, IBM Db2 defines a specific procedure that prescribes copying the objects in a specific order and using a specific grouping of the objects.

To copy the Db2 catalog, use JCL DSNTIJIC located in the <prefix>.NEW.SDSNSAMP installation data set. This JCL is different for every Db2 release (the catalog and directory changes with each release). Copy utilities supplied by vendors provide a special keyword for copying the Db2 catalog and directory and prepare a list of objects for the Db2 release of the subsystem on which they execute.

Because the catalog tables update constantly, create SHRLEVEL CHANGE image copies of the Db2 catalog to preserve full availability of the Db2 catalog objects. SHRLEVEL CHANGE has been the default setting since Db2 release 10.

4.2.11 Multiple Image Copies

A copy utility allows multiple sequential image copies to be created in one pass. Each copy can have different attributes (as an example, one copy is stored to a DASD and another copy is stored to a tape). Db2 supports registration of up to four copies. Two copies are intended to be stored at the local site and two copies at the recovery site.

- Local primary copy (registered with ICBACKUP=' ').
- Local backup copy (ICBACKP='LB').
- Recovery site primary copy (ICBACKUP='RP').
- Recovery site backup copy (ICBACKUP='RB').

It is also possible to create sequential copies from FlashCopy image copies. The Db2 COPY utility allows the creation of up to four sequential image copies based on a single FlashCopy image copy. SEQCOPY is the special phase for creating the sequential copies. Create two copies for the local site (COPYDDN option) and two for the recovery site (RECOVERYDDN option).

4.2.12 Deciding Between Full and Incremental Image Copies

When creating sequential image copies, choose between full and incremental image copies.

- Full image copies contain all pages of the database object.
- Incremental image copies contain only pages that have changed since the last image copy.

The decision whether to create full or incremental image copies is driven by the following elements:

- The amount of external storage required to store the image copies.
- The time required to create the image copy.
- The time required to recover database objects from the image copies.
- The number of changes of the database object since the last image copy.

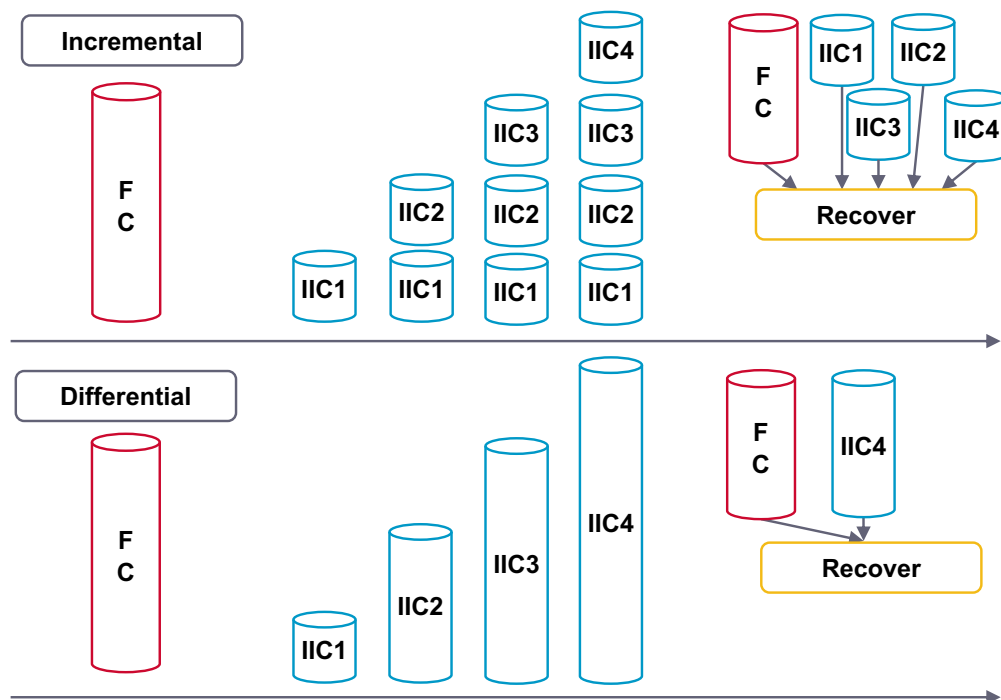
4.2.12.1 Incremental and Differential Backup

Incremental and differential backup differ in the following ways:

- True incremental image copies contain only pages that change after the most recent full or incremental image copy. To recover from true incremental image copies, a recovery utility must read the most recent full image copy and all subsequent incremental image copies. These copies provide incremental backup.
- Merged incremental image copies always contain all changes since the most recent full image copy. To recover from merged incremental image copies, a recovery utility reads only the most recent full image copy and the most recent incremental image copy. These copies provide differential backup.

Figure 2 shows a comparison of incremental and differential backup:

Figure 2: Incremental vs. Differential Image Copy



4.2.12.2 Storage Implications for Full Image Copies

Full image copies always require as much external storage as the whole database object, regardless of the number of changes. The size of the incremental image copies heavily depends on the number of changed pages. The smaller the number of the pages changed, the smaller the size of the incremental image copy.

4.2.12.3 Time Required to Create an Image Copy

When creating an image copy, a copy utility spends most of the time reading and writing data from and to external storage. If the number of changed pages is limited, an incremental image copy is created quickly and the database object is available sooner. As the number of changed pages becomes closer to the total number of pages in the database objects, creation of an incremental copy takes as much time as a full copy.

4.2.12.4 Time Required to Recover Database Objects from Image Copies

When deciding whether to create full or incremental image copies, consider the impact on the recovery process:

- Restoration from a full image copy requires reading that image copy.
- Restoration from an incremental image copy requires reading of the most recent full image copy and all subsequent incremental copies.

Incremental image copies require less time to create, but they require additional work for the recovery utility.

4.2.12.5 Automatic Switching to Full Copy

A copy utility can automatically decide whether to create a full or incremental image copy based on the status of the database object and the number of changed pages using real time statistics (RTS).

A full copy is created in these instances:

- For objects that are in the copy pending or non-recoverable state.
- When the number of changed pages is above a certain threshold.

In other cases, an incremental copy is created.

4.2.13 Creating Inline Copies

Inline copies are a special type of sequential copy created by data reorganization and load utilities.

When performing a reorganization of a tablespace, create a full image copy to maintain the recoverability of the database object. An obvious solution is to create a full image copy after the reorganization has finished.

Another possibility is to exploit the fact that the reorganization utility writes all pages of the tablespace to the Db2 VSAM data sets. A full image copy can be created in-line with that process by simultaneously writing the pages to both the VSAM data set and full image copy.

Considerations for Inline Copies

Some pages of the tablespace are changed during the log apply phase of the online reorganization. These pages are simply appended to the full image copy. This has the following consequences:

- Pages in the inline copy are not in strict sequential order and multiple versions of one page can be stored in the image copy.
- A recovery utility processing an inline image copy must first fully read the copy and restore Db2 VSAM data sets and then start rebuilding the indexes. This has an impact on the speed of the recovery.

4.2.14 Copying Image Copies

Even though a copy utility can create multiple image copies in one pass, it may be beneficial to create a copy of an existing image copy. When copying an existing image copy, the database object is not touched. Use one of the following methods to create an image copy:

- When creating multiple SHRLEVEL REFERENCE image copies, create just one image copy first, and then copy it. This reduces the read-only time of the database object for your applications.
- Copy an existing FlashCopy image copy and convert it to a sequential copy in the process.
- If the number of tape units is limited and additional image copies are needed on tapes, copy existing image copies.

4.2.15 DSN1COPY Considerations

The DSN1COPY standalone utility enables special processing of image copies or VSAM data sets. The greatest difference compared with the standard copy utility is that DSN1COPY processes only data sets (not database objects) and does not access the Db2 catalog.

With the DSN1COPY utility, the following can be performed:

- Copy a Db2 VSAM data set to a sequential data set and vice versa.
- Copy a sequential image copy to a Db2 VSAM data set.
- Copy a Db2 VSAM data set to a Db2 VSAM data set.
- Translate the OBID of data rows.
- Reset log RBAs in all pages.
- Check the integrity of the pages.

4.2.15.1 Typical Usage of DSN1COPY

- Problem diagnosis and determination. DSN1COPY is executed to check the integrity of the pages and print the contents.
- Restoring a database object that was dropped. After executing the DDL that recreates the database object, DSN1COPY is used to read the image copies and translate the OBIDs of all data rows. OBID translation is required because the dropped object is recreated with new OBIDs.
- Transferring data from one Db2 subsystem to a different one. This involves OBID translation (the database object will have different OBIDs on the destination Db2 subsystem) and resetting the log RBAs in all pages. Resetting RBAs is required when the destination Db2 subsystem uses different logs.

4.3 Determine Image Copies

Db2 registers the image copies in the SYSIBM.SYSCOPY catalog table. Find additional details about the image copies by querying this table or using the REPORT RECOVERY utility, which extracts the information from the SYSCOPY table and adds additional recovery related information such as log ranges from the SYSLGRNX directory table and archive logs from BSDS.

4.3.1 Registration of Image Copies and the COPY Utility

The COPY utility is an online utility that creates image copies. Use the COPY utility shipped with Db2 or choose a vendor product. Every created image copy is tracked in a dedicated catalog table named SYSIBM.SYSCOPY. This table holds information on image copies and information on every event related to backup and recovery.

Registration of the image copies allows the following:

- Easy location of image copies required for recovery
- Consolidation of image copies
- Easy removal of obsolete and unneeded image copies

There are many columns in the SYSIBM.SYSCOPY table. The most important information on the image copies is the following:

- Database object (database, space, data set number): DBNAME, TSNAME, DSNUM.
- Date and time of creation of the copy: RBA, LRSN, date, time, timestamp (START_RBA, PIT_RBA, TIMESTAMP).
- Type of copy: ICTYPE, STYPE, TTYPE.
- Information on the image copy data set (name, volume, file sequence number): DSNAME, DSVOLSER, FILESEQNO.
- SHRLEVEL parameter: SHRLEVEL.
- Number of pages copied (size of the image copy): COPYPAGESF.

NOTE: For standard image copies, the meaning of the START_RBA column varies depending on the SHRLEVEL of the copy during execution. This subtle difference has a significant impact between the recovery of objects or a LISTDEF.

The SYSIBM.SYSCOPY catalog table holds much more than just information on the image copies. All events that can have an impact on backup and recovery are stored in this table. These events include the following:

- Utility executions (LOAD, REORG, REBUILD, RECOVER, CHECK DATA, REPAIR).
- Database object attribute changes (for example, DSSIZE changes, ALTERs).
- Quiet points—no activity on the object (for example, executions of the QUIESCE utility).
- SQL activity (for example, mass delete).

If the information on image copies and other events is combined, it provides the backup and recovery utilities with a full picture. It enables the recovery utility to determine the most desirable image copies for recovery, to detect non-recoverable situations, or to outline a fallback plan when the most desirable image copies are not available.

The following section demonstrates how to identify different types of copies from the SYSCOPY table as well as from the REPORT RECOVERY utility. Filter the SYSCOPY rows per objects: tablespaces with databases. WHERE clause can be WHERE DBNAME='database' AND TSNAME='tablespace'. The DSNAME column shows the data set name.

4.3.1.1 Full SHRLEVEL REFERENCE Sequential Image Copy

- ICTYPE = F
- SHRLEVEL = R
- STYPE = blank

4.3.1.2 Full SHRLEVEL CHANGE Sequential Image Copy

- ICTYPE = F
- SHRLEVEL = C
- STYPE = blank

4.3.1.3 Incremental Sequential Image Copy

- ICTYPE = I

4.3.1.4 SHRLEVEL REFERENCE FlashCopy Image Copy

- ICTYPE = F
- ICBACKUP = FC
- SHRLEVEL = R
- STYPE = T
- PIT_RBA = RBA or LRSN when FlashCopy completes
- START_RBA = Corresponds to a value when the pages were last externalized to disk.

4.3.1.5 SHRLEVEL CHANGE: Inconsistent FlashCopy Image Copy

- ICTYPE = F
- ICBACKUP = FC
- SHRLEVEL = C
- STYPE = N
- PIT_RBA = RBA or LRSN when FlashCopy completes
- START_RBA = Start point for the updates since the image copy was taken

4.3.1.6 SHRLEVEL CHANGE: Consistent FlashCopy Image Copy

- ICTYPE = F
- ICBACKUP = FC
- SHRLEVEL = C
- STYPE = T
- PIT_RBA = RBA or LRSN when FlashCopy completes
- START_RBA = If there are any active units of work, then this value represents the oldest unit of work that has been backed out. Otherwise, the RBA or LRSN of the last externalization is recorded here.
- DSVOLSER = If there are any uncommitted units of work, this column lists the checkpoints for each member. The RECOVER utility uses this information for its PRELOGC phase.

4.3.1.7 Concurrent Image Copy

- ICTYPE = F
- STYPE = C or J?

4.4 System-Level Backup

System-Level Backup (SLB) is a mechanism to consider in the Db2 recovery strategy. The ability to make copies of the whole environment makes this approach suitable for ERP environments that have hordes of database objects with many relations defined.

The online BACKUP SYSTEM utility invokes DFSMSHsm to create a fast replication copy of the volumes on which the Db2 data and log information reside. The output is called a system-level backup, which is an SHRLEVEL CHANGE copy of the volumes. The BACKUP SYSTEM history is recorded in the bootstrap data sets (BSDSs) and is visible in the REPORT utility in the event of a recovery.

Run the RESTORE SYSTEM utility at a later time to recover the subsystem or data sharing group. This utility also enables the use of the regular RECOVER utility to recover a single data set (object) from a system-level backup.

4.4.1 Important Notes and Considerations

When opting for system-level backups there are important notes to observe:

- BACKUP SYSTEM does not reset the COPY-pending status. A regular full image copy is still required to switch the status off.
- During BACKUP SYSTEM processing, the following real-time statistics tables are invalidated:
 - COPYUPDATEDPAGES (pages updated since the last image copy)
 - COPYCHANGES (number of update DML operations, including rows loaded with LOAD RESUME)
 - COPYUPDATELRSN (LRSN/RBA of first data change)
 - COPYUPDATETIME (timestamp of first data change)
- SYSCTRL or SYSADM authority is required to use the BACKUP SYSTEM utility
- All data sets to be copied must be SMS-managed data sets.
- In a data sharing environment, there should not be any quiesced members that terminated abnormally. Otherwise, the BACKUP SYSTEM request fails.
- BACKUP SYSTEM does not cause any workload disruption.
- BACKUP SYSTEM can be part of a disaster recovery scenario.

4.4.2 Can System-Level Backup Be the Only Backup?

No, traditional image copies cannot be abandoned. Observe the following:

- System-level backup is not as mature as traditional COPY/RECOVER: Restrictions can prevent such recovery.
- System-level backup is a SHRLEVEL CHANGE copy.
- System-level backup cannot be used for recovery of an application or subset of data within the subsystem.
- BACKUP SYSTEM does not reset the COPY-pending status.

4.4.3 Setup Considerations

Preparing for a system-level backup is not a trivial task. It typically demands work from the storage team to separate the Db2 data from the log data in different storage groups and move non-Db2 data sets out of the storage groups. The Db2 staff must create another ICF catalog for the logs, copy them from the old volumes, and update the BSDS.

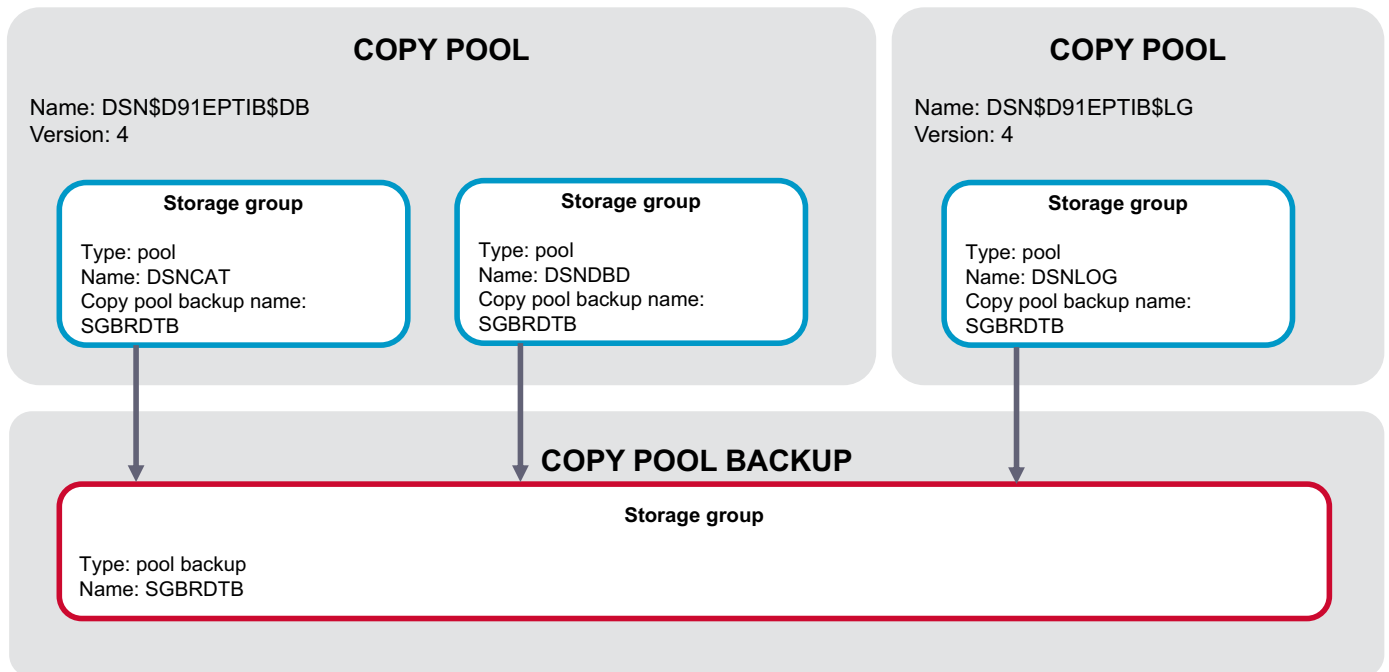
In summary, the tool is easy to use but the setup is quite cumbersome for the storage team. Before using the BACKUP SYSTEM utility, define the copy pools using the following Db2 naming convention:

`DSN$locn-name$cp-type`

- DSN is the unique Db2 product identifier.
- \$ is a delimiter. Use the dollar sign character (\$).
- locn-name is the Db2 location name.
- cp-type is the copy pool type. Use DB for database (including Db2 catalog) and LG for log.

It is possible to define alternate copy pools, which can be shared among different Db2 subsystems.

Figure 3: System-Level-Backup Copy Pools



As shown in [Figure 3](#), databases (together with logs) can be copied using either BACKUP SYSTEM FULL or DATA ONLY.

At times it is preferable to not make a full copy due to a lack of storage, or because some objects have not changed since the last copy. In this case, if a full copy has been executed in the past, establish an incremental FlashCopy image copy by specifying ESTABLISH FCINCREMENTAL. Subsequent copies are incremental until the command/keyword END FCINCREMENTAL is invoked.

Chapter 5: Db2 Recovery

Occasionally, and especially in emergency scenarios, a DBA is required to provide all recovery alternatives available so that the application teams can decide which is the most convenient recovery point for the business given that they may need to process additional data after their Db2 objects have been recovered. For this reason, it is imperative that all backup strategies are aligned with the business processes so that the recovery alternatives are meaningful for the applications.

5.1 Recovery Scenario Key Points

Executing a backup strategy is a timed, planned event. While recovery procedures should be tested so they can be controlled, documented, and measured, a real recovery scenario will occur and be reported as an emergency where timing will have a significant impact on the overall satisfaction of application support teams and end-users, making this step critical. Below is a list of the points to consider when designing and evaluating backup strategies:

- Db2 objects will not always recover to the last image copy. Be prepared to offer and recover from any previous copy that the application team requires.
- It is likely that a group of objects must be recovered to the same point. In the case of ERP applications and databases that serve as the data repository for vendor products, entire databases will be recovered with hundreds of tables. Some of them may include XML and LOB tables that require additional considerations depending on the point of recovery.
- After recovering Db2 objects, maintenance utilities such as RUNSTATS, REORGs, and REBIND commands must be executed.

5.2 Recovery Point Objective

Plan what and where to recover. Recovery Point Objective (RPO) determines the point to which an object must be recovered. Identify the following:

- The database object or objects to be recovered: tablespaces and/or indexes.
- The recovery point: the RBA/LRSN to which the data will be recovered.

The RBA/LRSN can be unspecified (recovery to the current point), specified by value (X'00112233445566000000'), or specified indirectly (recovery to last image copy, to last full image copy, to last quiet point, to specific image copy, and so on). The next element to consider is how to locate the recovery points and alternatives for any given object. For this task, execute the REPORT utility described next.

5.3 Report on Recovery

The REPORT utility reports on the recovery points for the object specified. The output report includes information on the image copies, the log records used for recovery of the database objects, the indexes that were rebuilt, and other important information about the recovery.

Tips on Executing the REPORT Utility

Commonly, a table to be reported on will not be a standalone table. This means that a table can be involved in any of the following situations:

- A referential constraint.
- The table contains LOB or XML columns.
- The table is an archived enabled table.
- The table is a system-period temporal table.
- The table has been cloned at some point in the past.

In any of these cases, executing the REPORT utility reports on the specific table mentioned but does not include dependent objects. For these cases and for more accurate reporting, use the REPORT utility with the option TABLESPACESET, as this option looks for dependent objects and includes them in the output report.

See below for a sample output of REPORT RECOVERY showing the main parts of the report: SYSCOPY rows, log range rows, and log data sets. The types of image copies in the report use codes that identify each type of image copy. These codes are listed in the official IBM documentation on the description for the SYSIBM.SYSCOPY catalog table.

```

DSNU000I      327 07:14:21.80 DSNUGUTC - OUTPUT START FOR UTILITY, UTILID = <utility-id>
DSNU1044I     327 07:14:21.81 DSNUGTIS - PROCESSING SYSIN AS EBCDIC
DSNU050I      327 07:14:21.82 DSNUGUTC - REPORT RECOVERY TABLESPACE DSNDB06.SYSTSTAB
DSNU581I      !<ssid> 327 07:14:21.82 DSNUPREC - REPORT RECOVERY TABLESPACE DSNDB06.SYSTSTAB
DSNU593I      !<ssid> 327 07:14:22.08 DSNUPREC - REPORT RECOVERY ENVIRONMENT RECORD:
                MINIMUM RBA: 00000000000000000000
                MAXIMUM RBA: FFFFFFFFFFFFFFFFFF
                MIGRATING RBA: 00000000000000000000
DSNU582I !<ssid>32707:14:22.08DSNUPPCP-REPORTRECOVERYTABLESPACEDSNDB06.SYSTSTABSYSCOPYROWSANDSYSTEMLEVELBACKUPS

TIMESTAMP = 2018-10-23-08.05.48.331184, IC TYPE = F, SHR LVL = C, DSNUM = 0000, STARTLRSN=0000000002CBCDFCFAD2
DEV TYPE = 3490, IC BACK = , STYPE = , FILE SEQ = 0017, PIT LRSN= 0000000002CBCE034EA0
LOW DSNUM = 0001, HIGH DSNUM = 0001, OLDEST VERSION = 0002, LOGICAL PART = 0000, LOGGED = Y, TTYPE =
JOBNAME = IS<ssid>I1, AUTHID = DCBATCH, COPYPAGESF = 1.222E+03
NPAGESF = 1.26E+03, CPAGESF = 1.57E+02, MODE CREATED = N
DSNAME = <ssid>BKUP.SYSTSTAB.GVS4GBOI, MEMBER NAME = , INSTANCE = 01, RELCREATED = Q

TIMESTAMP = 2018-10-25-08.16.36.132846, IC TYPE = F, SHR LVL = C, DSNUM = 0000, STARTLRSN=0000000002CF519369AD
DEV TYPE = 3490, IC BACK = , STYPE = , FILE SEQ = 0018, PIT LRSN= 0000000002CF519B3013
LOW DSNUM = 0001, HIGH DSNUM = 0001, OLDEST VERSION = 0002, LOGICAL PART = 0000, LOGGED = Y, TTYPE =
JOBNAME = IS<ssid>I1, AUTHID = DCBATCH, COPYPAGESF = 1.222E+03
NPAGESF = 1.26E+03, CPAGESF = 1.68E+02, MODE CREATED = N
DSNAME = <ssid>BKUP.SYSTSTAB.GVWDBVVS4, MEMBER NAME = , INSTANCE = 01, RELCREATED = Q

DSNU583I !<ssid> 327 07:14:22.08 DSNUPPLR - SYSLGRNX ROWS FROM REPORT RECOVERY FOR TABLESPACE DSNDB06.SYSTSTAB
UCDATE UCTIME START RBA STOP RBA START LRSN STOP LRSN PART MEMBER ID
102118 21200300 0000000002C88CA4E56B 0000000002C894206BB2 00D51D757CB702000000 00D51D7C902B75000000 0001 0000
102118 22382451 0000000002C8948CB04A 0000000002C9161E3246 00D51D87006C5F000000 00D51D9B6CF35D000000 0001 0000
102218 02210526 0000000002C91D004629 0000000002C926F75000 00D51DB8C63956000000 00D51DBC13FF0F000000 0001 0000
102218 02373121 0000000002C92DF3947D 0000000002C92FD1EA81 00D51DBC728031000000 00D51DC627DFBE000000 0001 0000
102218 03214836 0000000002C9300D2062 0000000002C953A46A6D 00D51DC6588DE8000000 00D51DCDF581B0000000 0001 0000

DSNU584I !<ssid> 327 07:14:22.08 DSNUPPBS - REPORT RECOVERY TABLESPACE DSNDB06.SYSTSTAB ARCHLOG1 BSDS VOLUMES
START TIME END TIME START RBA END RBA START LRSN END LRSN
20182922107595 20182950140041 0000000002C865EE3000 0000000002C891E02FFF 00D51AB95130C3743600 00D51D79DC7B78681C00
DSN=<ssid>.ARCHLOG1.A0005582 VOLSER=LOG016 UNIT=SYSDA
20182950140041 20182950312248 0000000002C891E03000 0000000002C8BDD22FFF 00D51D79DC7B78681C00 00D51D8E80828ECC6C00
DSN=<ssid>.ARCHLOG1.A0005583 VOLSER=LOG004 UNIT=SYSDA
20182950312248 20182950337174 0000000002C8BDD23000 0000000002C8E9C42FFF 00D51D8E80828ECC6C00 00D51D940FF5A530D400
DSN=<ssid>.ARCHLOG1.A0005584 VOLSER=LOG002 UNIT=SYSDA
DSNU586I !<ssid> 327 07:14:22.08 DSNUPSUM - REPORT RECOVERY TABLESPACE DSNDB06.SYSTSTAB SUMMARY
ARCHLOG1 BSDS VOLSER(S) LOG016
LOG004
LOG002
LOG018
LOG021
LOG005
LOG003
LOG019
LOG001
LOG017
LOG006
LOG008
LOG007

DSNU589I !<ssid> 327 07:14:22.08 DSNUPREC - REPORT RECOVERY TABLESPACE DSNDB06.SYSTSTAB COMPLETE
DSNU598I !<ssid> 327 07:14:22.08 DSNUPPBK - REPORT RECOVERY SYSTEM LEVEL BACKUPS
DSNU588I !<ssid> 327 07:14:22.08 DSNUPPBK - NO DATA TO BE REPORTED

DSNU580I 327 07:14:22.08 DSNUPORT - REPORT UTILITY COMPLETE - ELAPSED TIME=00:00:00

```

5.4 Recovery for NOT LOGGED Objects

As the name implies, objects defined as NOT LOGGED do not record information in the log to be used for recovery. However, this does not mean that the object is unrecoverable. Db2 detects the following events as recoverable points for these objects:

- Creation of new image copies.
- Transition from LOGGED to NOT LOGGED attribute.
- Transition from NOT LOGGED to LOGGED attribute. When a NOT LOGGED object is altered to make it LOGGED, the object is placed in COPY pending status to enforce a recoverable point before resuming the logging activity.
- Addition of new partitions for NOT LOGGED tablespaces. This point applies for a new partition on a PBG tablespace or an explicitly created partition by executing ALTER TABLE.

Normally, tablespaces defined as NOT LOGGED include LOB tablespaces which, by their nature, tend to have a considerable size and are defined as NOT LOGGED to avoid impacting the size of the log and improve response times.

NOTE: Since SHRLEVEL CHANGE image copies rely on the log, SHRLEVEL CHANGE image copies cannot be executed over NOT LOGGED objects.

NOTE: System-level backups include NOT LOGGED objects. However, they cannot be used for object-level recovery over NOT LOGGED objects.

5.5 Recovery from Image Copy

The standard method of forward recovery is based on previously created image copies and the Db2 log. This method may be the best option for any of the following:

- Recovering to a strategic point in a production cycle: The COPY utility is executed in a meaningful moment and you decide to restore to that point.
- Db2 VSAM data sets are lost or corrupted.
- Recovering a dropped object.

However, when data must be recovered due to application or human error, a log analysis tool may be more suitable.

Recovery from image copies can be performed by the Db2 RECOVER utility or any equivalent vendor utility (hereinafter collectively referred to as a recovery utility).

5.5.1 Requirements for Image Copies

The recovery utility queries the SYSIBM.SYSCOPY table and identifies candidate image copies that can be used for recovery. A candidate image copy must meet the following criteria:

- It must have been created before the recovery point.
- The image copy data set must be available.
- The image copy must not be in a PIT range. PIT ranges are defined by the ICTYPE='P' rows in the SYSIBM.SYSCOPY table inserted after PIT recoveries.

The recovery utility verifies the candidate image copies. The most recent suitable image copies are selected first. If the selected image copy is not available, the recovery utility can fall back to an older image copy. When creating full and incremental image copies, the recovery utility selects a full image copy and the subsequent incremental image copies for recovery.

5.5.1.1 Recovery from Sequential Image Copies

Restoration of Db2 VSAM data sets from sequential image copies is performed at the page level. The recovery utility reads data from the sequential image copy data sets and writes the data to the Db2 VSAM data sets.

The actual processing is more complicated: The recovery utility must create and extend the Db2 VSAM data sets, create zero pages for pages that are not stored in the image copy, process duplicated dictionary and system pages, ensure that a page already present in the Db2 VSAM data set is not overwritten with an older version of the page, and so on.

5.5.1.2 Recovery from Concurrent Image Copies

Recovery is performed at data set level. The recovery utility creates and executes DFSMSdss commands that restore the Db2 VSAM data sets from the concurrent image copies.

5.5.1.3 Recovery from FlashCopy Image Copies

Recovery is performed at data set level. Because the flash copies are Db2 VSAM data sets, the recovery utility restores the Db2 VSAM data sets by simply copying the FlashCopy image copy. Fast replication can be used for restoration (if possible). Fast replication significantly reduces the downtime to recover a database object. The recovery utility can also restore the Db2 VSAM data sets at page level.

5.5.2 Non-Recoverable Scenarios

When processing records from the SYSIBM.SYSCOPY table, the recovery utility also checks for any non-recoverable scenarios. A typical non-recoverable scenario is a recovery to a point between a non-recoverable event and a full image copy. A non-recoverable scenario is reported, and the recovery is terminated.

5.5.3 Recovery Process and Db2 Logs

The Db2 logs play an important part in the recovery process. Depending on the options used while executing the COPY utility, the logs may need to be read to ensure data consistency by resolving URs that were in progress at the time of the COPY.

5.5.4 Log Data Sets and Log Ranges

When the image copies are identified and verified, the recovery utility determines which part of the Db2 log is read, to ensure that changes from the identified image copy with the most recent RBA/LRSN are applied to the recovery point. The recovery utility reads the BSDS to identify the Db2 log data sets and it also identifies log ranges to speed up the recovery process by skipping the parts of the Db2 log that are not relevant for the recovery.

5.5.4.1 Applying Log Records

After the Db2 VSAM data sets are restored, the recovery utility must apply all log records since the START_RBA of the most recent image copy is used for the recovery. The recovery utility reads the log records in sequence and changes the pages in the restored Db2 VSAM data sets accordingly.

Processing of the Db2 log record is more complicated when recovering from a consistent snapshot image copy or when performing a point-in-time recovery.

- When recovering from a consistent snapshot image copy, the recovery utility must identify uncommitted work that was removed from the consistent snapshot image copy.
- For a PIT recovery, all uncommitted work at the recovery RBA/LRSN is removed from the recovered Db2 VSAM data sets.

In the case of FlashCopy consistent image copies, there is a pre-log apply phase that reapplies the inflight transactions that were removed from the copy to make it consistent.

5.5.5 Recovering vs. Rebuilding Indexes

Indexes that were not recovered from image copies must be rebuilt. However, it is possible to recover them if they are defined with COPY YES and if they were explicitly copied. Otherwise, execute REBUILD INDEX. The recovery utility reads pages from the restored tablespaces, extracts keys from the rows in the tablespaces, sorts the extracted keys, and builds pages of the indexes.

5.5.6 Recovery Registration to SYSCOPY

After recovering the Db2 VSAM data sets and rebuilding the indexes, the recovery utility updates the Db2 catalog tables.

The following rows are inserted to the SYSIBM.SYSCOPY table:

- ICTYPE='E' rows for recovery to the current point. The START_RBA is the current RBA/LRSN at the time of recovery.
- ICTYPE='P' row for point-in-time recovery. The START_RBA is the current RBA/LRSN at the time of recovery. The PIT_RBA is the recovery RBA/LRSN.
- ICTYPE='B' rows for index rebuilds. The START_RBA is the current RBA/LRSN at the time of the rebuild.

5.5.7 Recovering Partitioned Objects

Recover partitioned objects on a partition level, or as a single entity. The recovery processing depends on how the image copies of the partitioned object were created. If the image copies were created at partition level, it is possible to recover a single partition, a subset of partitions, or all partitions and use parallel processing to increase the performance of the recovery. If a single image copy was created for the whole tablespace, it is possible to recover a single partition, a subset of partitions, or all partitions. However, parallel processing is not possible, because there is only one input image copy data set.

5.5.7.1 Migrating Data from One Db2 Subsystem to Another

Assume data is migrating from one Db2 subsystem (source subsystem) to another Db2 subsystem (target subsystem). Create an image copy on the source subsystem and perform OBID translation on the target subsystem to migrate the data. The recovery utility can read an image copy of an object created on a *source* Db2 subsystem and restore the data to Db2 VSAM data sets on a *target* Db2 subsystem.

Migration with OBID translation can be faster than a combination of UNLOAD and RELOAD, but it also has additional restrictions:

- The schema of the objects must be the same on both sides of the OBID translation
- The edit procedures must be the same on both sides of the OBID translation. This is important when the edit procedures are used for compression and encryption.

After recovery from the image copy, the Db2 log records can be applied only if the Db2 log data sets and BSDS of the source Db2 subsystem are accessible from the target Db2 subsystem.

5.5.8 OBID Translation

OBID translation is a special type of recovery from an image copy provided by a vendor recovery utility. The recovery utility changes the OBIDs (internal identification numbers of database objects) in the pages of the recovered Db2 VSAM data sets.

OBID translation is required for the following events:

- Recovering a previously dropped object.
- Migrating data from one Db2 subsystem to another.

5.5.9 Fast Log Apply

Fast Log Apply is a feature of the IBM RECOVER utility that accelerates the recovery. Fast Log apply is based on principles very similar to SORTLOG processing. The recovery is accelerated by the grouping of log records by data set number and page number. Grouping of the log records reduces I/O overhead, because it minimizes repeated access to the same page of a Db2 VSAM data set being restored.

5.5.10 Recovering the Db2 Catalog

Recovery of a Db2 catalog from image copies requires special treatment. Db2 catalog objects must be recovered in a specific order. Details on the procedure are described in the documentation for the RECOVER utility shipped with Db2.

IBM provides the installation job DSNTIJIC that can be used to create backup copies of catalog and directory objects. The copying of objects must be done in the sequence defined in the job DSNTIJIC. For example, SYSIBM.SYSCOPY, SYSIBM.SYSUTILX, and SYSIBM.SYSLGRNX should be the last tables copied.

In the same way as executing COPY, Db2 catalog and directory tablespaces must be recovered and the related indexes rebuilt in a specific order. A full recover of the catalog and directory is recommended to avoid creating a situation where the two are out of sync. The procedures to recover the catalog and directory differs between releases, so be sure to always use the correct guidelines.

The point-in-time recovery of catalog and directory objects involves recovering all the system objects (catalog and directory) together with the user objects to the same point in time. It is highly recommended that you recover to a point of consistency. Creating a point of consistency is not an easy task and involves the QUIESCE utility.

To check the consistency of the catalog and directory, perform the following procedure:

1. Execute DSN1COPY utility with CHECK option against the catalog underlying VSAMs.
2. Check the consistency of the catalog tables using the sample SQL SELECT statements provided by IBM in the DSNTESQ member of the sample library.
3. Check the consistency between the catalog and a physical tablespace by running the REPAIR CATALOG TEST utility.
4. Check the consistency between the catalog and the directory by running the REPAIR DBD TEST utility.

5.5.10.1 Recovery from System-Level Backup

Instead of an image copy, the recovery utility can use a system-level backup, if one is available, to recover a database object or multiple database objects.

Recovery from a system-level backup is very similar to recovery from a FlashCopy image copy. The recovery utility executes HSM commands to restore the Db2 VSAM data sets. The rest of the recovery process is the same as when recovering from image copies. See [System-Level Backup](#) for additional information.

NOTE: There is a limitation when choosing a system-level backup to recover individual objects. If any of these utilities were run after the execution of a system-level backup, the usage of a system-level backup is not allowed:

- REORG TABLESPACE
- REORG INDEX
- REBUILD INDEX
- LOAD REPLACE
- RECOVER using an image copy or concurrent copy

5.5.10.2 Backout Recovery

Backout recovery is suitable for quick recovery from application or human errors. Instead of identifying the most recent image copy and applying the Db2 log records forward, the recovery utility uses the current Db2 VSAM as the recovery base. Then it applies the Db2 log record backwards to the specified recovery point. Obviously, backout recovery cannot be used if the Db2 VSAM data sets have been lost.

Backout recovery provides better performance when a full image copy is available but contains data that is too old (several days to a week), meaning forward recovery would require days of log records to be applied.

5.5.10.3 Recovery with a Log Analysis Tool

A log analysis tool can generate SQL statements that undo selected changes in database objects. On one hand, executing the SQL is slower than directly updating the Db2 VSAM data set pages, but on the other hand, the database object is available during the process and the changes that are undone can be restricted to specific applications, users, units of recovery, and so on.

5.5.10.4 Recovery with LOAD RESUME

Some objects could be recovered by a LOAD rather than a regular recovery. Consider, for example, a multi-table tablespace where one table must be recovered. It may not be possible to do this from an image copy. Instead, use UNLOAD from the image copy and then use LOAD RESUME.

5.6 Point-In-Time Recovery

The ability to recover data to a prior time is a point-in-time (PIT) recovery. Use this approach to recover data to a specific point in time using one of the various options. The logic behind this approach is that data stays consistent, meaning that for every running transaction, uncommitted data for the given PIT is removed. Recognize a point-in-time based on:

- A log point represented by RBA/LRSN (TOLOGPOINT).
- Image copies (FULL, INCREMENTAL, SYSTEM LEVEL BACKUP), which can be accompanied by TOCOPY, TOLASTCOPY, and TOLASTFULLCOPY.

By selecting a full image copy created with SHRLEVEL REFERENCE as a recovery target, Db2 only uses the given data set without applying Db2 logs. When incremental copy is specified, Db2 also requires a full image copy to recover the data and Db2 logs to establish a consistent point.

As mentioned earlier, recover tablespaces to an earlier point-in-time (partial recovery) using the TORBA or TOCOPY keywords. Exercise caution when making a partial recovery, since data may be left inconsistent in the following ways:

- If one partition of a tablespace is recovered, that data may be inconsistent with the rest. The tablespace partition is placed into copy pending status.
- If the tablespace and indexes are not consistent with each other after a partial recovery, the indexes are placed into recover pending status.
- Referentially related tablespaces may be inconsistent with a tablespace that is partially recovered. Tablespaces with dependent tables are placed into check pending status if a referentially related tablespace is partially recovered. In this case, use CHECK DATA or CHECK INDEX.
- When using TORBA, it is possible to restore to the middle of a unit of recovery. Determine which event the RBA marks before using a TORBA recovery point. Quiesce points are good recovery points.

5.6.1 Selecting an RBA for a PIT Recovery

As discussed previously, a quiesce point or the `START_RBA` value of an image copy made with `SHRLEVEL REFERENCE` both serve as excellent recovery points since each naturally ensures consistency. This means that executing `RECOVER` to such points guarantees consistency and each requires no additional log processing. However, a PIT recovery is not limited to only such points. Use `RECOVER` with options `TORBA` or `TOLOGPOINT` to any point and the `RECOVER` utility can process uncommitted units of work to place the object in a consistent state.

Even in very busy Db2 environments, there are points in time (or rather ranges) with no activity (no data updates) on the related group of objects. Those ranges are called quiet points. A log analysis tool helps identify those ranges from the Db2 log and allows registration of the lower RBA of the range as a recovery point in `SYSIBM.SYSCOPY`. Those points are consistent and can be used in recovery.

5.6.2 PIT Recovery Considerations

Apart from the list above, recovering data to a specific point in time can be influenced by the following:

- Compressed data.
- Physical model changes, including pending definition changes and extended log format.
- Pending statuses. When recovering data to `TOCOPY`, `TOLASTCOPY`, or `TOLASTFULLCOPY`, associated indexes are placed in rebuild-pending status. `REBUILD` index is preferred.
- Referential integrity. All related objects should be at the same consistency point: `-REORG TABLESPACE` with `LOG(NO)`.
- Not logged objects. When applying the `NOT LOGGED` attribute to a tablespace and making updates to it, the updates are not recoverable because they are not recorded in the log. In this case, choose an image copy as the recovery point.
- Some PIT recoveries may not be possible due to `SCHEMA` changes or pending changes, so a system-wide DR may be required. Save the `SYSCOPY` information frequently as DDL changes are stored here that may be useful to analyze in a recovery scenario.
- Another good approach can be regularly establishing a consistent point by running `FULL` image copy with `SHRLEVEL REFERENCE` or `QUIESCE` utility. When using this approach on a group of objects defined as an application, it is possible to easily recover all objects to that point. When the `QUIESCE` is not affordable, consider using the log analysis tool to identify quiet points from the Db2 log, if any are present in the environment.
- Point-in-time recovery can be a solid alternative during data corruption, but observe the limitations. Existence of consistent points is crucial, but it may be hard to identify them for a related group of objects. Application-defined referential integrity makes it harder to identify those relations, so they are good candidates for recovery points.

5.6.3 Recovery Avoidance for Unchanged Objects

Starting with Db2 12, the use of the options `TOLOGPOINT` or `TORBA` try to avoid recovering objects that are unchanged. Db2 makes this decision by determining whether the object to be recovered has changed after the specified log point. The result of this enhancement is a decrease in recovery times. If an object is not recovered due to this analysis, the object remains accessible to users.

To use this feature, the parameter `SCOPE` is provided for the `RECOVER` utility. With the combination of a `LISTDEF` that evaluates the objects to be processed, the following options are available:

- `SCOPE UPDATED`: This option recovers only the objects that have changed since the given log point.
- `SCOPE ALL`: Recovers all objects in the given list.

NOTE:

- When using `SCOPE UPDATED`, if an object is in `COPY` pending or `RECOVER` pending it will be recovered even if the object has not changed.
- If a partitioned tablespace contains a single partition that has changed after the given recovery point, a `RECOVER` on a tablespace level recovers all partitions.

5.7 Recovering Erroneous Db2 Transactions

Db2 application developers design and implement their programs well and do extensive testing in the development environment. Software quality engineers test those new or updated programs, and retest them in the test environment. When all is validated, the changes are implemented in the production environment. Even when all the correct procedures are in place and followed properly, running the application on a production subsystem may cause the unexpected to happen. The application may fail, use an incorrect input, or execute in the wrong order—any of which may result in unwanted changes to data. Alternatively, a Db2 user may accidentally perform updates to the Db2 data in the production environment, believing that they are working in the test environment.

These types of erroneous Db2 transactions usually affect only a small number of Db2 objects and result in a limited amount of data being accidentally corrupted. However, the situation must be corrected. As a DBA, avoid executing a regular RECOVER job because of the following:

- **Availability:** The Db2 objects will not be accessible for some time.
- **Data Loss:** Other valid transactions may be performed (by other applications/users) in the meantime.

Db2 recovery is not a trivial task, and human errors can happen when defining/executing RECOVER jobs, especially if there is unfamiliarity with RECOVER syntax or if the recovery strategy has not been tested recently. Hopefully, there is another way to recover from such a situation: a soft recovery from the Db2 log that allows the erroneous Db2 transactions to be reverted (and not the other valid transactions) while keeping all the Db2 objects online and accessible. This is also much less risky, as the DBA can easily review what will be changed prior to running this soft recovery. However, this option is not available using just Db2 base utilities and the aid of vendor tooling is required for such a recovery.

5.7.1 Reconstructing the Transaction

Reconstruct the SQL statements issued by the transaction from the Db2 log. The data stored within the log records (undo/redo) depends on the type of data manipulation performed. For a delete operation, Db2 records the entire after-image (redo) and the entire before-image (undo). For an update operation, only the byte stream of changes in the data page is logged, unless the object is defined with DATA CAPTURE CHANGES.

Review the simplified process of generating UNDO SQL for the example transaction:

1. The BSDS data set and SYSLGRNX directory table are read to get the necessary log ranges and log data sets that must be scanned based on the time when the transaction happened. Starting with Db2 for z/OS version 10, the SYSLGRNX table is now accessible by SQL. To read the BSDS, the DSNJU004 utility can be used.
2. The unit of recovery that was created as part of the transaction processing must be identified in the log. It can be done by reading the Begin_UR log record of the unit of recovery that contains information about who executed the transaction.
3. Once the UR is identified, the undo/redo log records within the UR are scanned. Those records contain the data changes as well as the IDs of objects for which the data change happened. Since the transaction performed inserts, the undo/redo log records contain the entire after-image of the row. To read the Db2 log, use the DSN1LOGP utility. However, DSN1LOGP runs on archive data sets, but not on active data sets when Db2 is running.

Remember, the image of the data in the log contains just the raw data without any description of the data. To get this information, determine which objects were changed. The IDs of the objects from the log records are used to identify the objects in the Db2 catalog. Reading the Db2 catalog gives the necessary information to understand what has been inserted into which table. Combining this information enables the reconstruction of the original INSERT statements. Since the data was inserted in error, the opposite operation to INSERT is required. So, the proper DELETES are generated and finally executed to completely remove all changes that were made by the transaction.

Even in this very simplistic scenario, such reconstruction requires a great deal of user effort. It is far too time consuming and error-prone. Moreover, the real-world scenarios are much worse. Since the update operation log record does not contain the whole before and after-image, it must be reconstructed using either the current image from the tablespace or the last image from the image copy, depending which point is closer to the transaction execution time. In both cases, additional log records must be scanned to collect all subsequent changes of the data. Additionally, there is much more that must be considered.

5.7.2 Summary

Fortunately, there are many tools available that makes recovering from erroneous transactions much easier. The most common features are:

- Reports to identify and audit erroneous transactions.
- Automatic generation of UNDO SQL statements issued by the transaction to reverse its changes.
- Automatic generation of REDO SQL statements issued by the transaction to apply valid transaction changes after a point-in-time recovery.

5.8 Dropped Object Recovery

One of the most painful experiences in the work life of a DBA may be the accidental drop of a database object on a production subsystem. Most organizations have procedures to prevent this occurrence, yet it still happens. When it does happen, it is crucial to recover the object into its original state without any loss of data as soon as possible. Every minute your applications cannot access data harms your business. These situations are very stressful for a DBA and the recovery process is heavily exposed to human error.

5.8.1 Implications of Dropping a Table

Observe what happens when the DROP TABLE MYTBL statement is issued. Besides applications that start failing, and the loss of all data stored in the dropped table, Db2 may also drop additional objects. Db2 holds information about all objects in the Db2 catalog. Upon issuing a DROP statement, Db2 starts deleting rows from the catalog that belonged to the dropped table and to any other object that depends on the deleted table. The following table summarizes the affected objects:

Table 5: Implications of Dropping a Table

Affected Object	Consequences
Table	The row in SYSIBM.SYSTABLES that contains information about the table and rows from SYSIBM.SYSCOLUMNS for all columns of the dropped table are deleted. If the table had an identity column, the sequence attributes are removed from SYSIBM.SYSSEQUENCES.
Trigger	If triggers are defined on the table, they are dropped and the corresponding rows are removed from SYSIBM.SYSTRIGGERS and SYSIBM.SYSPACKAGES.
View	Views defined on the table are dropped from SYSIBM.SYSTABLES.
Alias	Alias defined for the table becomes orphaned.
Package	Packages that involve the use of the table are invalidated.
Synonym	Synonyms from the table are dropped from SYSIBM.SYSSYNONYMS.
Index	Indexes created on any columns of the table are dropped and rows deleted from SYSIBM.SYSINDEXES.
Statistics	Access path statistics and space statistics for the table are deleted from the catalog tables: SYSTABLES, SYSTABLESPACE, SYSTABSTATS, SYSTABLEPART, SYSLOBSTATS, SYSCOLUMNS, SYSCOLSTATS, SYSCOLDIST, SYSCOLDISTSTATS, SYSINDEXES, SYSINDEXSTATS, and SYSINDEXPART.
Authorizations	Users who were authorized to use the table or views lose those privileges because rows from SYSTABAUTH, SYSCOLAUTH, and SYSSEQUENCEAUTH catalog tables are deleted.
Tablespace	If the tablespace containing the table is implicitly created (using the CREATE TABLE statement without the TABLESPACE clause), the tablespace is also dropped.
LOB objects	If the table contains a LOB column, the auxiliary table and the index on the auxiliary table are dropped. The LOB tablespace is dropped if it was created with SQLRULES (STD).

Table 5: Implications of Dropping a Table

Affected Object	Consequences
XML objects	If the table contains an XML column, the auxiliary table, the index on the auxiliary table, and XML tablespace are dropped.

5.8.2 Dropped Table Recovery Considerations

In general, Db2 recovery is a complex task. However, dropped object recovery is nearly impossible using only the base utilities provided with Db2. It often takes hours to collect the necessary information and set up all the jobs. It is also near impossible to recover from such a situation using the base utilities provided with Db2. Even with the help of more advanced utilities, a manual approach is complicated, even more so when a database with many objects must be recovered.

There many physical factors that affect the speed of a recovery process. The speed of the disks where image copies and log data sets reside, how many archived logs are required to reconstruct data changes, and many more. The factor that has the most significant effect on the speed of a log-based recovery is the DATA CAPTURE parameter.

There are several vendor tools on the market that automate dropped-object recovery to some degree. However, no tool can save a situation that is lacking a prepared solution for accidental situations and has not prepared possible recovery scenarios. If backup and recovery procedures are correctly set and proper tooling exists, outage time and the loss of critical data can be kept to a minimum. With the proper preparation, the time needed to set up recovery jobs to recover a dropped object can also be reduced from hours to minutes.

Chapter 6: Recommendations and Guidelines

This user guide has examined various elements of the theory that different copies that can be created, along with some recommendations, special considerations, particularities, and reminders to be used when recovering from these copies. This section provides general guidelines to help in evaluating or creating actual backup and recovery strategies.

This section is a collection of tips, best practices, and recommendations from subject matter experts with expertise in all aspects of backup and recovery in Db2.

6.1 Determine Db2 Objects Critical to Business

At a minimum, determine which objects are critical to the business and must be recovered quickly in the event of a disaster recovery scenario.

6.2 Plan for Evolving Requirements

An obvious point for a DBA may not be obvious to business leaders and application departments. Entire backup strategies must be redesigned from scratch if they do not provide the most useful recovery points to the application, and their critical points may change as the application evolves. In some scenarios, a predefined set of rules must be observed. Below is a list of a few:

- Maintain duplicate copies in different storage devices with different expiration policies.
- Keep a specific number of copies for Db2 objects of the same application over a predefined time frame.
- Maintain a predetermined time between subsequent copies, regardless of the application's individual needs and critical times. This case likely complies with audit requirements.

Frequently, a strategy must be designed that is flexible enough to satisfy requirements from different business roles:

- Application departments specify the critical times for base backups.
- Auditors provide specific guidelines regardless of the logic or nature of the application.
- DBAs create contingency plans that cover many recovery options in the least amount of time possible.

6.3 Leverage the Contents of the Db2 Catalog and Real-Time Statistics

To comply with the requirements that exist over an application's lifetime, leverage the contents of SYSIBM.SYSCOPY to answer questions centering around backups. Query the contents of the catalog to decide which objects to copy, where, and how often. This formatted output can then be processed by a REXX script that generates and submits the resulting JCL that must be invoked. Additionally, the RTS table SYSIBM.SYSTABLESPACESTATS provides enough information for a much more detailed logic that can decide if it is the proper time for a copy (use these contents to also decide on other utilities such as a REORG or RUNSTATS). If the backup guidelines and requirements demand it, it is common to find scripts and applications to handle this logic and provide quick responses to user demands.

6.4 Consider the Recovery SLA

Classify Db2 objects based on the business applications they serve, and how important these are to the business. Seek feedback from business leaders to define recovery SLAs and perform recovery estimation benchmarks. Adjust the backup frequency accordingly.

Table 6: Recover SLA

Business Importance	Recovery SLA	Backup Strategy
CRITICAL	30 minutes	Weekly FIC + daily IIR
HIGH	2 hours	Weekly FIC + Weekly IIR
MEDIUM	4 hours	Monthly FIC + Weekly IIR
LOW	10 hours	Monthly FIC

6.5 Review, Practice, and Document Db2 Recovery Procedures

While a data loss situation is less likely to happen now than in the past, a Db2 recovery may still be required one day. When it comes to business data, even if the risk is low, assume that a major recovery scenario could arise. Review and practice the Db2 data backup and recovery processes and procedures on a quarterly basis (or annually, at least). When documenting, keep in mind that someone else may be performing the recovery process. Remember to cover the following actions:

- Register the object grouping.
- Document each step in all aspects.
- Include all scripts that are used.
- Document the environment specifics.
- Write down the names of all key people and contacts.
- Keep the recovery plan up-to-date (review and update after every change of objects/environment).

6.6 Ensure Existing Procedures and Jobs Take Advantage of New Features

IBM Db2 and the associated Database Management Solutions for Db2 for z/OS are regularly enhanced. Ensure the Db2 recovery procedures and processes are adjusted to take advantage of the new functions that are available. For instance:

- Backward log recovery.
- A new SET LOG command to add active logs on the fly, without stopping Db2.
- Db2 11 introduced less restrictive recoveries when pending DDL-materializing.
- Db2 12 introduced a new DSNZPARM that enables stacking changes on columns, where changes can be immediate (with AREO* state on the tablespace and index) or deferred (with AREOR state on the tablespace).

6.7 Read-Only Db2 Objects

Your system may consist of read-only Db2 objects. These do not require a high backup frequency to speed up recovery, since no log scanning is necessary to apply logs. If this is the case, consider reducing the frequency of backups (saving CPU and storage) without affecting recovery time.

6.8 Keep (at Least) the Latest Full Image Copy on DASD

Recovery jobs run faster when the image copy is on a DASD, which allows a higher level of parallelism. When image copies are on tape, a recovery is slower and tape stacking severely limits parallelism. Use an automated mechanism (such DFSMSHsm or CA Disk™ Backup and Restore) to move older copies from DASD to tape.

6.9 Use Incremental Image Copies

An Incremental Image Copies (IIR) strategy helps speed up recovery of Db2 objects. IIR can be merged with a Full Image Copy (FIC) to produce an updated FIC, further reducing the recovery time of Db2 objects (Db2 does not need to perform the merging when performing a recovery).

6.10 Index Backups

To avoid CPU costs, refrain from taking image copies of the indexes. Understand however that, in the event of a recovery scenario, indexes can be rebuilt only after the recovery of the tablespace is complete. Hence, for especially large objects or for objects critical to business ensure that the related indexes are copied: recovering an index is likely to be faster than rebuilding it, and indexes can be recovered while the tablespace is being recovered.

6.11 Keep Db2 Log Information

Keep at least one day of active logs on DASD. Consider keeping a couple of days of archive logs on DASD and using an automated mechanism (such as DFSMSHsm or CA Disk) to move older logs to tape. The other recommendations are as follows:

- Use dual logging.
- Keep archive logs on separate volumes (physical disks) from the active logs so the active log I/O does not need to share resources when Db2 offloads to the archive log. The same rule applies to log copies when dual logging is used.
- Keep at least one day of activity in active log data sets.
- For archive logs on tapes, do not use tape compression.

6.12 Consider Upgrading Segmented Tablespaces to Partition-by-Growth (UTS) Tablespaces

When a tablespace is partitioned, objects can be copied at the partitioned level, which can speed up single-partition recovery and allows a higher level of parallelism when recovering the entire tablespace.

6.13 Consider Partition By Date

Depending on the business applications interacting with Db2 data, partitioning by a continuously-ascending key may lead to updates being applied only to the data in the current partition, which leaves the other partitions read-only. In such a case, there is no need to frequently backup the other older partitions.

6.14 Regularly Use the MODIFY RECOVERY Utility

Use MODIFY RECOVERY to clear obsolete backup information from SYSIBM.SYSCOPY. However, ensure that RETAIN is used on the three most recent Full Image Copies for every object to ensure recoverability of y data in case a backup happens to be unusable for recovery.

Also, ensure compliance with the auditing regulations for the business. For instance, in the Financial Industry in some countries, auditing regulations require retention of (at least) a monthly backup of your business data for an extended period (such as 7 years).

6.15 Accidental Dropping of Objects

To avoid an accidental DROP of objects in production, consider altering them using RESTRICT ON DROP syntax in the ALTER statement.

6.16 Automated Techniques to Adjust Backup Strategy

Consider using techniques provided by IBM Db2 to automate some tasks. For example:

- DSNACCOX uses real-time statistics to make backup recommendations for Db2 objects.
- A simple SQL against the Db2 catalog enables the ability to locate read-only Db2 objects (COPYUPDATEDPAGES information in SYSIBM.SYSTABLESPACESTATS indicates the number of pages updated since the last COPY).

6.17 Optimize Recovery with IBM Db2 zPARMs

The following zPARMs affect the performance of Db2 recovery. Review and adjust these system parameters according to the specifics of the systems and Db2 recovery requirements.

- CHKFREQ specifies the checkpoint frequency. Control the frequency by the number of minutes, the number of Db2 log records written, or both. Use the SET LOG command to dynamically change the checkpoint frequency. The recommendation is to specify a checkpoint interval of 2 to 5 minutes during busy hours of the day and/or night.
- DLDFFREQ [ON|OFF] specifies whether the level ID of a page set or partition is to be updated at Db2-determined checkpoint intervals.
- LOGAPSTG specifies the maximum DBM1 storage that can be used by the fast log apply process. The default value is 0 MB, which means that fast log apply is disabled except during Db2 restart, when fast log apply is always enabled.
- PCLOSEN/PCLOSET specifies the number of consecutive Db2 checkpoints/number of minutes that a page set or partition can remain in read-write mode since it was last updated. When this limit or the RO SWITCH CHKPTS/RO SWITCH TIME is reached, Db2 changes the page set or partition to read only. Tuning this parameter can improve performance for recovery, logging, and data-sharing processing.
- OUTBUFF specifies the size of the log buffer. To improve log write performance, consider increasing the size of the log buffer. The default is 4 MB. The maximum size of the output log buffer is 400 MB.
- MAXARCH determines the maximum number of archive log data sets that are recorded in the BSDS data set. If applications require recovery, be sure that the recovery process can go back to a point of time far enough in the past.

Appendix A: Understanding Database Objects

To build recovery processes, there must be an understanding of database objects. Applications and users work with Db2 tables that contain data traditionally organized into rows with defined columns. The tables are physically stored in tablespaces, which are grouped in databases. These are the objects that contain the most valuable data.

All information about the database objects is stored in the Db2 catalog, which can be considered metadata storage. The catalog itself also contains tables and can be queried with the proper authorities.

The following subsections also contain a few sample queries to help instruct on the various database objects.

A.1 Obtain Database Information

To list all database names, use the following query:

```
SELECT DB.NAME FROM SYSIBM.SYSDATABASE;
```

To list all databases and their tablespaces, use the following query:

```
SELECT DBNAME, NAME FROM SYSIBM.SYSTABLESPACE;
```

A.2 Obtain Table Information

If the table schema and name are known, obtain information about a table from SYSIBM.SYSTABLES with the following query:

```
SELECT *
  FROM SYSIBM.SYSTABLES
 WHERE NAME = 'tablename'
    AND CREATOR = 'tablecreator';
```

Obtain information from SYSIBM.SYSCOLUMNS about the columns in a table with the following query:

```
SELECT NAME, TBNAME, COLTYPE, LENGTH, NULLS, DEFAULT
  FROM SYSIBM.SYSCOLUMNS
 WHERE TBNAME = 'tablename'
    AND TBCREATOR = 'tablecreator';
```

To list the columns for a table, join the SYSIBM.SYSTABLES and SYSIBM.SYSCOLUMNS tables on the CREATOR = TBCREATOR and NAME = TBNAME columns.

Table authorizations are stored in the SYSIBM.SYSTABAUTH table. This table contains information about who can access data. To list all the users that have been granted access to a table, use a query like the one in the following example:

```
SELECT GRANTEE
  FROM SYSIBM.SYSTABAUTH
 WHERE TTNAME = 'tablename'
    AND TCREATOR = 'tablecreator'
    AND GRANTEETYPE <> 'P';
```

The GRANTEETYPE <> 'P' guarantees a listing of only the users and not the packages and plans.

Obtain DBID, OBID for table:

```
SELECT NAME DBID, OBID FROM SYSIBM.SYSTABLES WHERE NAME='MYDB.MYTBL' AND CREATOR='USERID';
```

A.3 Obtain Tablespace Information

The SYSIBM.SYSTABLESPACE table contains one row per tablespace. To see the tablespaces defined in a database, join the SYSIBM.SYSTABLESPACE and SYSIBM.SYSDATABASE tables on the DBNAME = NAME column.

To display information about the physical tablespace partitions that are stored in the SYSIBM.SYSTABLEPART table to, for example, check the logical partition number, use a query like the one in the following example:

```
SELECT LOGICAL_PART, LIMITKEY, PARTITION,
       FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'dbname'
      AND TSNAME = 'tsname'
ORDER BY LOGICAL_PART;
```

To see the number of partitions for a tablespace, use the following query:

```
SELECT COUNT(*) FROM SYSIBM.SYSTABLEPART
WHERE DBNAME = 'dbname'
      AND TSNAME = 'tsname';
```

To select tablespaces from a given database that has 8K pages, use the following query:

```
SELECT * FROM SYSIBM.SYSTABLESPACE
WHERE DBNAME = 'dbname'
      AND BPOOL LIKE 'BP8K%';
```

To list tablespaces where the amount of extents is greater than *n*, use the following query:

```
SELECT NAME, DBNAME, PARTITION
       FROM SYSIBM.TABLESPACESTATS
WHERE EXTENTS > n;
```

To see compressed tablespaces where the compression has no savings, use the following query:

```
SELECT A.TSNAME, A.DBNAME, A.PARTITION
       FROM SYSIBM.SYSTABLEPART A
WHERE A.COMPRESS = 'Y'
      AND A.PAGESAVE = 0
      AND A.CARD > 0;
```

To obtain DBID, PSID for a tablespace, use the following query:

```
SELECT NAME, DBID, PSID FROM SYSIBM.SYSTABLESPACE WHERE NAME='MYDB.MYTS' AND CREATOR='USERID';
```

A.4 Obtain Index Information

Below are two tables that contain the data for indexes. The SYSIBM.SYSINDEXES table contains a row per index. The SYSIBM.SYSINDEXPART table contains data for each physical index partition and even non-partitioned indexes.

To obtain all the data about a specific index, use the following query:

```
SELECT *
  FROM SYSIBM.SYSINDEXES
 WHERE NAME = 'indexname'
    AND CREATOR = 'creator';
```

To see all the indexes defined for a table, use the following query:

```
SELECT *
  FROM SYSIBM.SYSINDEXES
 WHERE TBNAME = 'tablename'
    AND TBCREATOR = 'tablecreator';
```

To see the amount of keys in an index, use the following query:

```
SELECT COUNT(*)
  FROM SYSIBM.SYSKEYS
 WHERE IXCREATOR = 'schema'
    AND IXNAME = 'ixname';
```

To list index spaces where the amount of extents is greater than *n*, use the following query:

```
SELECT INDEXSPACE, DBNAME, PARTITION
  FROM SYSIBM.INDEXSPACESTATS
 WHERE EXTENTS > 40;
```

To list indexes that have more than 10% of their entries pseudo-deleted, use the following query:

```
SELECT A.INDEXSPACE, A.DBNAME, A.PARTITION
  FROM SYSIBM.INDEXSPACESTATS A, SYSIBM.SYSINDEXPART B, SYSIBM.SYSINDEXES C
 WHERE DECIMAL((REORGPEUDODELETES)*1)/DECIMAL(CARDF,31,0) > 0.1
    AND A.DBNAME = C.DBNAME
    AND A.INDEXSPACE = C.INDEXSPACE
    AND A.PARTITION = B.PARTITION
    AND C.NAME = B.IXNAME
    AND C.CREATOR = B.IXCREATOR
    AND DECIMAL(CARDF,31,0) > 0;
```

NOTE: Starting with Db2 11, Db2 can automatically clean up pseudo-deleted index entries.

To list indexes where the insert value exceeds the maximum key value by more than 10% since the last REORG, use the following query:

```
SELECT INDEXSPACE, DBNAME, PARTITION
  FROM SYSIBM.INDEXSPACESTATS
 WHERE REORGAPPENDINSERT > 0
    AND REORGAPPENDINSERT IS NOT NULL
    AND REORGINSERTS IS NOT NULL
    AND REORGINSERTS > 0
    AND ((REORGAPPENDINSERT*10)/REORGINSERTS) > 1
    AND REORGNUMLEVELS > 0;
```

To find indexes with a low cluster ratio, use the following query:

```
SELECT DISTINCT A.NAME, A.CREATOR, A.CLUSTERRATIO, B.NAME
  FROM SYSIBM.SYSINDEXES A, SYSIBM.SYSTABLES B, SYSIBM.SYSTABLESPACE C
 WHERE A.CLUSTERRATIO < 96 AND A.CLUSTERRATIO > 0
    AND A.CLUSTERING = 'Y'
    AND A.TBNAME = B.NAME
    AND A.CREATOR = B.CREATOR
    AND C.PARTITIONS > 0
    AND A.DBNAME = B.DBNAME
    AND A.DBNAME = C.DBNAME
    AND C.NAME = B.TSNAME;
```

A.5 Obtain Views Information

Information about views is spread throughout several tables.

- SYSIBM.SYSTABLES has a row for each view, where TYPE='V'.
- SYSIBM.SYSTABAUTH contains authorizations for views.
- For each column in a view, there is a row in the SYSIBM.SYSCOLUMNS table.
- Additionally, there are two specific tables for views:
 - SYSIBM.SYSVIEWS contains the definitions for a view. Locate the view definition in the STATEMENT CLOB column.
 - Db2 table dependencies for a view are saved in the SYSIBM.SYSVIEWDEP table.

A.6 Obtain Storage Group Information

Storage group information is found in the SYSIBM.SYSSTOGROUP catalog table. However, to obtain information about the volumes that are in the storage group, query the SYSIBM.SYSVOLUMES table. It is possible to join these two tables on the SGNAME column.

To show the storage group, how much space is used, and when the space was last calculated, use the following query:

```
SELECT SGNAME, VOLID, SPACE, SPCDATE
  FROM SYSIBM.SYSVOLUMES, SYSIBM.SYSSTOGROUP
 WHERE SGNAME = NAME
 ORDER BY SGNAME;
```

A.7 Obtain Primary and Foreign Key Information

The KEYSEQ column in the SYSIBM.SYSCOLUMNS identifies whether the column is part of the primary key. If the value is non-zero, the column is part of a primary key.

To retrieve the columns in a primary key for a specific table, use the following query:

```
SELECT TBcreator, TBNAME, NAME, KEYSEQ
  FROM SYSIBM.SYSCOLUMNS
 WHERE TBcreator = 'tablecreator'
    AND TBNAME = 'tablename'
    AND KEYSEQ > 0
 ORDER BY KEYSEQ;
```

The UNIQUERULE column of the SYSIBM.SYSINDEXES table identifies the primary index of a table if the value is 'P'. To find information about the primary index, use the following query:

```
SELECT TBCREATOR, TBNAME, NAME, CREATOR, DBNAME, INDEXSPACE
FROM SYSIBM.SYSINDEXES
WHERE TBCREATOR = 'tablecreator'
      AND TBNAME = 'tablename'
      AND UNIQUERULE = 'P';
```

The SYSIBM.SYSRELS and SYSIBM.SYSFOREIGNKEYS contain information about referential constraints. The SYSRELS table includes one row per referential constraint and the SYSFOREIGNKEYS table has one row for every column in a referential constraint. Each constraint is uniquely identified by the schema, the dependent name, and the constraint name (RELNAME).

To retrieve information about referential constraints and the parent table for every relationship, use the following query:

```
SELECT A.CREATOR, A.TBNAME, A.RELNAME, B.COLNAME, B.COLSEQ,
       A.REFTBCREATOR, A.REFTBNAME
FROM SYSIBM.SYSRELS A, SYSIBM.SYSFOREIGNKEYS B
WHERE A.CREATOR = 'tbschema'
      AND B.CREATOR = 'tbschema'
      AND A.TBNAME = 'tbname'
      AND B.TBNAME = 'tbname'
      AND A.RELNAME = B.RELNAME
ORDER BY A.RELNAME, B.COLSEQ;
```

To find all the child tables in the referential relationship, use the following query:

```
SELECT A.RELNAME, A.CREATOR, A.TBNAME, B.COLNAME, B.COLNO
FROM SYSIBM.SYSRELS A, SYSIBM.SYSFOREIGNKEYS B
WHERE A.REFTBCREATOR = 'parentschema'
      AND A.REFTBNAME = 'parentname'
      AND A.RELNAME = B.RELNAME
ORDER BY A.RELNAME, B.COLNO;
```

To find the quantity of child tables, use either of the following queries:

```
SELECT CHILDREN
FROM SYSIBM.SYSTABLES
WHERE TYPE='T'
      AND CREATOR='creator'
      AND NAME='name';

SELECT COUNT(*)
FROM SYSIBM.SYSRELS
WHERE REFTBCREATOR = 'creator';
      AND REFTBNAME='name';
```

A.8 Obtain Check Constraints Information

The SYSIBM.SYSCHECKS table contains one row for each check constraint defined on a table. The SYSIBM.SYSCHECKDEP table contains one row for each reference to a column in the check constraint.

To list the check constraints on a table, use the following query:

```
SELECT A.TBOWNER, A.TBNAME, B.COLNAME, A.CHECKNAME, A.CREATOR,
       A.CHECKCONDITION
FROM SYSIBM.SYSCHECKS A, SYSIBM.SYSCHECKDEP B
WHERE A.TBOWNER = B.TBOWNER
      AND A.TBNAME = B.TBNAME
      AND B.TBNAME = 'tablename'
      AND A.CHECKNAME = B.CHECKNAME
ORDER BY TBOWNER, TBNAME, COLNAME;
```

A.9 Obtain LOB Information

The relationship between base and auxiliary tables is tracked in the SYSIBM.SYSAUXRELS table. To list a table and its related auxiliary table, use the following query:

```
SELECT COLNAME, PARTITION, AUXTBOWNER, AUXTBNAME
FROM SYSIBM.SYSAUXRELS
WHERE TBNAME = 'tablename'
      AND TBOWNER = 'tablecreator';
```

Information about LOB columns can be found in the SYSIBM.SYSCOLUMNS table. For example, to find BLOB columns, use the following query:

```
SELECT NAME, TBNAME, COLTYPE, LENGTH2, NULLS, DEFAULT
FROM SYSIBM.SYSCOLUMNS
WHERE TBNAME= 'tablename'
      AND TBCREATOR = 'tableschema'
      AND COLTYPE = 'BLOB';
```

A.10 Obtain Routine and User-Defined Function Information

The information about procedures and functions is saved in the SYSIBM.SYSROUTINES table, which contains a row for each routine. The ROUTINETYPE column distinguishes between procedures and functions. To list all functions, use the following query:

```
SELECT SCHEME, NAME, FUNCTION_TYPE, PARM_COUNT
FROM SYSIBM.SYSROUTINES
WHERE ROUTINETYPE= 'F';
```

The routine parameters are saved in the SYSIBM.SYSPARMS table.

A.11 Obtain Trigger Information

Information on a user-defined trigger is stored in the SYSIBM.SYSTRIGGERS table.

To list all the triggers defined for a table, use the following query:

```
SELECT DISTINCT SCHEMA, NAME, TRIGTIME, TRIGEVENT, GRANULARITY, CREADEDTS
FROM SYSIBM.SYSTRIGGERS
WHERE TBNAME = 'tablename'
AND TOWNER = 'tableschema';
```

The STATEMENT column contains the trigger text. Retrieve the text with the following query:

```
SELECT STATEMENT, CREATEDTS
FROM SYSIBM.SYSTRIGGERS
WHERE SCHEMA = 'schema_name'
AND NAME = 'trigger_name'
ORDER BY CREATEDTS;
```

To view the triggers that must be rebound because they were invalidated, use the following query:

```
SELECT COLLID, NAME
FROM SYSIBM.SYSPACKAGE
WHERE TYPE = 'T'
AND (VALID = 'N' OR OPERATIVE = 'N');
```

A.12 Obtain Sequence Information

The SYSIBM.SYSSEQUENCES table contains information about sequences. The SYSIBM.SYSSEQUENCEAUTH table contains information about the authorizations on a sequence.

To retrieve information about a sequence, use the following query:

```
SELECT *
FROM SYSIBM.SYSSEQUENCES
WHERE NAME = 'name'
AND SCHEMA = 'schema';
```

To list all the privileges a given user has on sequences, use the following query:

```
SELECT GRANTOR, NAME, DATEGRANTED, ALTERAUTH, USEAUTH
FROM SYSIBM.SEQUENCEAUTH
WHERE GRANTEE = 'user';
```


A.13 Obtain Recoverability Information

Information relevant to recovery is stored in the SYSIBM.SYSCOPY table. To find objects that do not have a full image copy, use the following query:

```
SELECT X.DBNAME, X.NAME, X.CREATOR, X.NTABLES, X.PARTITIONS
FROM SYSIBM.SYSTABLESPACE X
WHERE NOT EXISTS
  (SELECT *
   FROM SYSIBM.SYSCOPY Y
   WHERE X.NAME = Y.TSNAME
        AND X.DBNAME = Y.DBNAME
        AND Y.ICTYPE = 'F')
ORDER BY 1, 3, 2;
```

To convert the START_RBA into a printable form from the SYSCOPY table, use the following query, which uses the HEX built-in function:

```
SELECT DBNAME, TSNAME, DSNUM, ICTYPE, TIMESTAMP, HEX(START_RBA)
FROM SYSIBM.SYSCOPY
ORDER BY DBNAME, TSNAME, DSNUM, TIMESTAMP;
```

A.14 Identify Log Ranges for a Tablespace

The most useful table to query in the Db2 directory is likely the SYSLGRNX table. To see the RBA ranges where an object was opened for updates, use the following query:

```
SELECT HEX(LGRSRBA), HEX(LGRSPBA), LGRMEMB, LGRPART
FROM SYSIBM.SYSLGRNX
WHERE LGRDBID = X'dbid'
AND LGRPSID = X'psid';
```

A.15 Retrieve All Grantees with Granted Privileges

Query all authorization tables to retrieve all grantees, which will include all grantee types.

```
SELECT GRANTEE, 'PACKAGE ' FROM SYSIBM.SYSPACKAUTH
UNION ALL
SELECT GRANTEE, 'TABLE ' FROM SYSIBM.SYSTABAUTH
UNION ALL
SELECT GRANTEE, 'COLUMN ' FROM SYSIBM.SYSCOLAUTH
UNION ALL
SELECT GRANTEE, 'ROUTINE ' FROM SYSIBM.SYSROUTINEAUTH
UNION ALL
SELECT GRANTEE, 'PLAN ' FROM SYSIBM.SYSPLANAUTH
UNION ALL
SELECT GRANTEE, 'SYSTEM ' FROM SYSIBM.SYSUSERAUTH
UNION ALL
SELECT GRANTEE, 'DATABASE' FROM SYSIBM.SYSDBAUTH
UNION ALL
SELECT GRANTEE, 'SCHEMA ' FROM SYSIBM.SYSSCHEMAAUTH
UNION ALL
SELECT GRANTEE, 'USER ' FROM SYSIBM.SYSRESAUTH
UNION ALL
SELECT GRANTEE, 'SEQUENCE ' FROM SYSIBM.SYSSEQUENCEAUTH
;
```

For additional filtering, use the GRANTEETYPE column of the corresponding tables.

A.16 Retrieve All Plans and Packages with Access to a Specified Table

To retrieve all packages that have access to a table, for example SCHEMA1.TABLE1, use the following query:

```
SELECT DISTINCT GRANTEE FROM SYSIBM.SYSTABAUTH
WHERE GRANTEETYPE = 'P' AND
TCREATOR = 'SCHEMA1' AND
TTNAME = 'TABLE1';
```

Appendix B: Database Management Solutions for Db2 for z/OS

Db2 recovery is a very complex task. The Database Management Solutions for Db2 for z/OS from Broadcom provide a powerful and flexible set of tools that are designed to ensure optimal database and SQL performance, efficient administration, and reliable backup and recovery of Db2 databases, systems, and applications. With these interoperable solution suites, an enterprise can also improve service levels, data availability, and application responsiveness—helping to reduce costs.

Database management solutions are flexible and provide choice without giving up control. Utilize full automation where appropriate, powerful manual control is desired or use expert software-guided assistance. Reduce the cost and risk associated with Db2 version upgrades and streamline workflows, use available skill sets, make more precise updates, and improve responsiveness across complex database and application environments.

This section provides additional details about the complete set of Database Management Solutions for Db2 for z/OS from Broadcom.

B.1 Database Administration Suite

Database Administration Suite for Db2 for z/OS automates routine administrative tasks while keeping complete control of the database environment in the DBA's hands. It helps reduce costs, simplifies Db2 object schema and data management, and helps provide for the integrity of the Db2 environment. This solution increases database availability through automation and helps easily perform complex database administration tasks. Database Administration Suite for Db2 for z/OS includes:

- **RC/Query[®]** for Db2 for z/OS: Provides catalog management capabilities and alleviates the time-consuming task of manually developing and testing specialized queries. RC/Query enables you to take immediate action—whether that means executing a command, invoking a utility or moving to another product from the Database Management Solutions for Db2 for z/OS product suite.
- **RC/Update[™]** for Db2 for z/OS: Automates labor-intensive tasks related to changing Db2 objects and data. It provides a development environment for the application developer, an editor and data copy feature for the end user, and sophisticated object management facilities for the DBA. These features result in simplified database administration tasks, reduced database downtime, and increased DBA productivity.
- **RC/Migrator[™]** for Db2 for z/OS: Provides automation that increases database availability and helps reduce the time and risk of human error involved with complex database administration tasks.
- **RC/Compare[™]** for Db2 for z/OS: Helps compare and synchronize Db2 structures that reside on different subsystems.
- **Fast Unload[®]** for Db2 for z/OS: Helps reduce the time required to unload Db2 data, thus improving the availability of corporate data.
- **Fast Load[™]** for Db2 for z/OS: A high-speed utility that helps load large amounts of data into Db2 tables.
- **RC/Secure[™]** for Db2 for z/OS: A comprehensive online security administration tool for Db2 for z/OS that helps streamline and automate the tasks of Db2 security administration.
- **Quick Copy** for Db2 for z/OS: Provides backups of Db2 data by making single or multiple image copies at high speed.
- **Fast Recover[™]** for Db2 for z/OS: Provides the ability to quickly recover Db2 data (tablespaces and indexes) after a disaster or system failure while minimizing CPU cycles, service units, and EXCPs. Fast Recover offers significant performance advantages over other utilities, incorporating multiple advanced features that differentiate it from competing products.

B.2 Database Backup and Recovery Suite

Database Backup and Recovery Suite for DB2 for z/OS enables organizations to back out a single update, support auditing of data changes, and replicate data changes to remote subsystems. Additionally, it is possible to quickly backup the data and consolidate incremental backups or accumulate log changes to produce a new backup to streamline the processing and improve the overall efficiency of the recovery jobs. These backup and recovery solutions also help to reduce the risk of a prolonged outage due to manual creation of recover jobs. Database Backup and Recovery Suite for Db2 for z/OS includes:

- Log Analyzer™ for Db2 for z/OS: A powerful product that analyzes Db2 log and SMF records to aid in auditing data changes.
- Recovery Analyzer™ for Db2 for z/OS: Simplifies complex Db2 recovery situations by automating the generation of efficient recovery jobs.
- Fast Recover for Db2 for z/OS: Provides the ability to quickly recover Db2 data (tablespaces and indexes) after a disaster or system failure, while minimizing CPU cycles, service units and EXCPs.
- Merge/Modify™ for Db2 for z/OS: Accelerates image copy and recovery processes to help reduce the drain on valuable CPU resources and streamline backup and recovery procedures.
- Quick Copy for Db2 for z/OS: Provides backups of Db2 data by making single or multiple image copies at high speed.

B.3 Database Performance Suite

Database Performance Suite for Db2 for z/OS helps increase database administrator (DBA) productivity by leveraging existing Db2 Real-Time Statistics (RTS), collecting new statistics on Db2 objects, and using them to automate Db2 housekeeping, automate utility scheduling and execution, and perform space management following user-defined thresholds. The included high-speed reorganization utility is designed to increase data availability and performance and save resources. Database Performance Suite for Db2 for z/OS includes:

- Database Analyzer™ for Db2 for z/OS: A sophisticated analysis engine that uses real-time statistics to provide graphs, trending, and forecasting information as well as a flexible scripting language to proactively take corrective actions.
- Rapid Reorg® for Db2 for z/OS: Performs quick and effective Db2 data reorganizations to help increase data availability and performance and save resources.

B.4 SQL Performance Suite

SQL Performance Suite for Db2 for z/OS helps quickly identify poorly running SQL statements and recommends changes to these statements based on expert rules. The access path, SQL performance, and Db2 object usage history can be stored for future trending and analysis. Application change control procedures can automatically detect and stop an inefficient SQL statement from moving into production, saving the company money due to high CPU usage and improving customer experience by reducing the risk of slow performing applications. SQL Performance Suite for Db2 for z/OS includes:

- Detector for Db2 for z/OS: Provides in-depth analysis capabilities that enables the identification of programs and SQL statements that most significantly affect Db2 system performance.
- Subsystem Analyzer for Db2 for z/OS: Designed to save database administrators (DBAs) time by consolidating information into a clear, concise view.
- Plan Analyzer for Db2 for z/OS: Designed to improve Db2 performance by efficiently analyzing SQL and utilizing expert rules to offer SQL performance improvement recommendations.

B.5 Report Facility

Report Facility is a full-function Db2 for z/OS reporting environment that provides users with easy access to business information. The product offers a robust yet easy-to-use reporting environment that combines an SQL creation system with multiple diverse and flexible report formatting styles. It provides users with intuitive access to business data, builds the queries designed to obtain that data, and formats the data into business information documents designed to meet operational and management requirements without forfeiting any measure of control.

Appendix C: Acronyms and Abbreviations

The table below lists the acronyms and abbreviations used in this document.

Table 7: Acronyms and Abbreviations

Term	Definition
BPAM	Basic partitioned access method
BSAM	Basic sequential access method
BSDS	Bootstrap data set
DASD	Direct access storage device
DBA	Database administrator
Db2	IBM Db2 for z/OS
DML	Data manipulation language
FIC	Full image copy
IIR	Incremental image copy
JCL	Job control language
LOB	Large object
LRSN	Log record sequence number
PIT	Point-in-time
RO	Read-only
RTS	Real time statistics
RTO	Recovery time objective
RBA	Relative byte address
QSAM	Queued sequential access method
STCK	Store clock
SLA	Service level agreement
VSAM	Virtual storage access method

Revision History

Db2-zOS-RM100; April 29, 2022

Updated product names.

Db2-zOS-RM100; May 14, 2019

Initial release.

