IBM® DB2® for Linux®, UNIX®, and Windows®

Best Practices

Minimizing Planned Outages

Matthew Huras
DB2 LUW Kernel Chief Architect

Chris Eaton
DB2 Technical Evangelist

Jens Seifert
DB2 / SAP Development

Rob Causley
DB2 Information Development

Executive Summary

This paper describes several best practices for minimizing or even potentially eliminating planned database outages. An outage is any situation in which the database is unable to serve user requests, either completely as in the case of an offline database or partially as in the case of unacceptable performance. Planned outages can include activities such as routine database maintenance activities or upgrades to your database environment.

Outages have a direct impact on the availability of your database environment. Availability is a measure of a system's ability to serve user application requests. Some database environments can tolerate occasional interruptions in availability while others must be available around the clock. The availability strategy you choose, and the resources you spend to achieve your availability goals, should be driven by the needs of your business.

DB2 data server contains features that can help you eliminate some types of planned outages and reduce the impact of others. This paper will discuss those features as well as when and how to use them to help achieve a high level of availability.

Introduction

This paper focuses on strategies for managing planned database outages, particularly when performing maintenance activities, with the goal of achieving a level of database availability that meets the needs of your business.

Areas of specific focus in this paper include:

- Table and index maintenance
- Data loading or data ingestion
- Upgrades
- Backups

This paper will also provide an overview of strategies for improving availability through:

- Tuning features and configuration
- Storage management

Availability

In the context of your business' process and your underlying database solution, availability refers to the measure of the database's ability to process transactions for user applications in a timely fashion.

The availability requirements of a database solution are determined by the needs of your business. For example, if a database solution serves a single storefront business that is open from 9:00am to 5:00pm, then the database could potentially be off-line from 5:01pm to 8:59am every night and still be considered highly available. On the other hand, if that same database solution serves a chain of stores that span multiple time zones, the window of possible down time becomes smaller. If the database solution serves an on-line business that needs to serve customers 24 hours every day, the database cannot be taken off-line at all without affecting customers.

Outages

An outage is any disruption in the ability of the database solution to serve user applications. Outages can be complete, as in the case of a database being offline, or partial, as in the case of unacceptably slow performance due to a high demand on the system's resources.

Outages can be classified in two groups: planned outages, which are the focus of this best practices paper, and unplanned outages.

Unplanned outages

Examples of unplanned outages include:

- The failure of one or more key components of the system, including hardware or software failure.
- Inadvertent administrative or user application actions such accidentally dropping a table that is needed for business-critical transactions.
- Poor performance due to suboptimal database configuration, or inadequate hardware or software.

Planned outages

Examples of planned outages include:

- Maintenance activities that require you to take the database offline, or maintenance activities that can be performed without stopping the database but that can adversely affect performance. These are the most common types of planned outages.
- Upgrades to your software or hardware, such as the installation of a DB2 Fix Pack, that may require you to take a partial or complete database outage.

Availability strategies

The availability strategy you choose should be based on the impact that planned outages will have on your business. In some cases, investing significant resources in a strategy that guarantees your database is highly available may be appropriate; in other cases you may not need a highly-available environment, if at all.

A crucial factor in determining your availability strategy is to ask how tolerant your business, or a specific system in your business, is to the occurrence of an outage. For example, a restaurant that has a website that contains menu information may be able to tolerate the occasional planned outage. On the other hand, any outage (planned or unplanned) on a stock exchange server that handles trade-related transactions would be catastrophic. Investing a significant amount of resources to ensure the restaurant's server is 99.99% available may not be cost-effective, but it certainly would be in the case of the stock exchange. Quantifying the impact of outages, be it in terms of lost revenue, lost productivity, or even lost customer confidence and goodwill, will help you make decisions about the level of investment you should make in your availability strategy.

Of course, you should try to avoid planned outages whenever possible by performing routine maintenance tasks, like database backups, while the database is online. You can also greatly reduce the duration of planned outages when performing upgrades.

Some database maintenance tasks may always require full or partial planned outages but you may be able to take steps to reduce the impact of these outages in terms of the

amount of time it takes to complete the maintenance task, or by minimizing the amount of system resources the task consumes. For example, excessive reorganization of tables or indexes can needlessly consume system resources to the point where your end users experience unacceptably slow response times to their queries. The database may still be theoretically online but suffering symptoms of a “brown out”, perhaps causing applications to time out before transactions can be committed. When it comes to upgrades, you can dramatically reduce the duration (and therefore the impact) of the outage by using DB2 features like High Availability Disaster Recovery (HADR) or by taking advantage of separate DB2 installations on the same system.

The following sections will discuss best practices for minimizing or potentially eliminating the effects of planned outages.

Improving availability during table and index maintenance

Reorganization operations (reorgs) on tables or indexes tend to be resource intensive activities which limit the availability of the tables and reduce concurrency. However, if you follow these general guidelines, you should be able to reduce the impact of reorg-related outages:

- Perform reorgs only if necessary
- Use only type-2 indexes
- Reduce the need to perform reorgs
- Minimize the impact of your reorg

Perform index and table reorgs only if necessary

Reorgs should not be performed periodically just as a matter of routine. In fact, many DB2 indexes and tables will seldom ever need to be reorganized. For additional information on this topic, see Determining when to reorganize tables and indexes.

If you are using the REORGCHK command to determine if you need to do a reorg, ensure that the threshold formulas are relevant to your workload. If you use the following rules of thumb to interpret the formulas, and you should be able to limit the frequency and impact of reorgs dramatically:

F1: Set PCTFREE to a moderate value

This formula is related to the percentage of overflow records. Consider adjusting the percentage of free space on each page (PCTFREE) to reduce the need for future reorgs. Setting PCTFREE to a moderate value such as 5% has shown good results. Overflow records increase processing time because the overflow area needs to be accessed as a part of any actions against the table. If overflow records are not accessed frequently in your workload, you can ignore this formula. You can determine how often overflow records are accessed using the overflow_accesses monitor element.

F2, F3: Ignore them if you don't need the space

These formulas are related to the free space to be gained by reorganizing the table. If you don't need to use the space elsewhere and don't need to free space for other tables, or if more data is going to be added to the table anyway, you can ignore these two formulas.

F4: Use an MDC table or a clustering index

This formula is related to the clustering ratio of an index. If you require clustering for good performance, consider using a multidimensional clustering (MDC) table or a clustering index. If your workload contains no or few important statements that would benefit from table clustering it, ignore this formula. For a detailed discussion of the benefits and design of MDC tables, see the "Physical Database Design" Best Practices paper.

F5, F6: Ignore them if you don't need the space, or use the CLEANUP option

These formulas are related to whether you should rebuild the indexes. If you don't have the need to free space for use elsewhere, you can ignore these two formulas. If you do want to perform an index-rebuilding reorg, you should first try a reorg with CLEANUP ONLY ALL first, to compact the index. The reorg with CLEANUP ONLY ALL is faster, consumes fewer resources, and has less impact because there is no object switch phase involved, which requires table-level locks.

F7: Ignore if pseudo-deleted keys don't have an impact

This formula is related to the number of pseudo-deleted RIDs on non-pseudo-empty pages. This describes the need for the lowest impact index reorg with CLEANUP ONLY ALL as the best. Even though this formula may suggest that you perform a reorg, you may still be able to ignore it. You don't need to reorg if, for example, your online workload is not penalized by the presence of the pseudo-deleted keys and if you don't need the extra space occupied by the pseudo-deleted keys.

F8: Ignore if pseudo-empty leaf pages don't have an impact

This formula is related to the number of pseudo-empty leaf pages. It describes the need for the lowest impact index reorg with CLEANUP PAGES as the best match. As with F7, even though this formula may suggest that you perform a reorg, you may still be able to ignore it.

Convert any type-1 indexes to type-2 indexes

Prior to DB2 Universal Database™ Version 8.1, all indexes were type-1 indexes. Even though all new indexes are type-2 indexes (except those built on tables already containing a type-1 index), type-1 indexes can still be present in your database. Because type-2 indexes have several advantages, such as increased concurrency characteristics and enabling additional function, such as online table reorgs, you should convert any remaining type-1 indexes. Specify the CONVERT option of the REORG command to perform this conversion.

Reduce the need to perform reorgs

Ask yourself if the reorg is even necessary and if there are alternative ways of achieving the desired outcome of the reorg.

You should not be performing a reorg without first understanding what is to be gained by it. For example, common wisdom in database systems may indicate that index reorgs are required to do a number of things, including (not exhaustively):

- Reducing the size and number of levels in the index
- Reducing the number of pseudo-deleted keys in the index
- Increasing the sequentiality of the index pages, thereby improving the I/O performance of index range scans

While these reasons are valid reasons for a reorg, remember that DB2 index management algorithms work towards eliminating the need for explicit reorgs in many cases. For example, pseudo-deleted keys are removed automatically during data manipulation language (DML) processing.

Similarly, there are a number of reasons why you might perform table reorgs, including:

- Reclustering data rows with respect to index keys, thereby improving the I/O performance of some access plans.
- Reclaiming embedded free space
- Eliminating overflow records
- Compressing a table

There are a number of strategies you can employ to achieve these outcomes that don't require a reorg.

- **Use clustering technology**. Multi-dimensional clustering (MDC) tables and clustering indexes help you avoid the need for reclustering reorgs. MDC tables keep the table clustered with respect to all specified dimensions, at all times. If you decide to use MDC tables, you will need to choose your table dimensions carefully to minimize the increase in storage consumption they entail. Use the Design Advisor to examine your data and workload, and to provide a recommended set of dimensions that optimize performance and space consumption.

  Clustering indexes attempt to keep the table clustered with respect to the specified index, on a best-effort basis, which reduces the need for reorgs just to keep the cluster ratio high. Although this should significantly reduce the need for reorganizing the table, clustering indexes can result in slightly reduced insert and update speeds.

- **Exploit the DB2MAXFSCRSEARCH registry variable**. In some cases, this can reduce the need for space-reclaiming reorgs. This registry variable controls the tradeoff between insert speed and space reclamation during insert processing. Low values favor faster inserts over space, by limiting the number of pages that

are searched for enough space to accommodate the row being inserted before a new page is added to the table. High values favor better space utilization, by searching more of the table. The default value of 5 is typically a good tradeoff. However, if you have very large tables with frequent deletes, you should consider a higher value for DB2MAXFSCRSEARCH. The value -1 ensures that if there is enough contiguous free space available, it will be used before a page is added to the table.

Note that large values, and the -1 setting, do not necessarily mean the entire table is searched on every insert. Instead, information about the free space in the table is cached and may be used to bypass the space search. For example, when all pages in the table are full, that information is cached, and in a subsequent insert the search is bypassed, and a page is immediately added to the table, even if the registry variable is set to -1.

- **Keep your transactions as short as possible**. Shorter-running transactions reduce the need for space reclaiming reorgs, so take actions such as committing as frequently as possible. The presence of long-lived transactions delays the reuse of embedded deleted space within tables.

- **Reduce or eliminate the frequency of UPDATEs which enlarge rows**. This can reduce the need to perform table reorgs to remove overflow records. Updates that cause a row to no longer fit on its current page result in the creation of an overflow record. For example, you could use CHAR(8) instead of VARCHAR(8) for a column that is frequently updated, if the potential for extra space consumption is acceptable. Updates to the VARCHAR(8) column may change the row length, whereas updates to the CHAR(8) column would not.

- **Exploit the PCTFREE parameter**. This can reduce the need to perform table reorgs as frequently to remove overflow records. This free space can then be used to accommodate row growth without the creation of overflow records. Typically, you should use PCTFREE if you know that the rows will be enlarged later, for example, by a subsequent update. You should not use it if you know that the rows won't be enlarged later, for example, if they are fixed-length rows.

- **Use automatic compression**. This can reduce or eliminate the need to reorganize for the purposes of data compression. With automatic compression, tables that grow to a size where compression starts to make sense are automatically compressed.

In addition, note that you should take special care if you use ALTER TABLE APPEND ON for tables in which you do a large number of inserts and deletes. When you use append mode, delete operations leave holes in the middle of the table and inserts will not use up those holes because the add data to the end of the table only. Therefore, using APPEND mode for tables can cause more table fragmentation and lead to the need to reorg more frequently

Minimize the impact of your reorgs

If you do have to perform a reorg of your tables or indexes, avoid doing an offline reorg whenever possible. Use either an online (also known as inplace) reorg or a cleanup reorg to achieve similar results while still maximizing availability.

- **Use an online table reorg**. An online reorg is often the ideal way to maintain data availability during a reorg operation. Full read and write access is uninterrupted except during the truncate phase, during which all write access is suspended but read access is maintained. Use the INPLACE...ALLOW WRITE ACCESS options with the REORG command to specify an online reorg. One tradeoff to be aware of here is that online reorgs experience slower performance and increased log space consumption. However, you can modify your reorg to mitigate this. For example, if you are reorganizing for space reclamation only and clustering is not important in your workload, perform an online reorg with no index specification (and ensure no clustering index is defined on the table) to maximize availability during the reorg.

- **Perform a cleanup instead of a full index reorg**. For instance, if a bulk deletion job just deleted large number of the rows in a table, this may leave index keys pseudo-deleted. The index manager will typically remove these pseudo-deleted keys automatically at some later time. However, if you want to be sure the next accesses through the index aren't penalized by the presence of the pseudo-deleted keys, use the ALLOW WRITE ACCESS … CLEANUP ONLY options of the REORG INDEXES command to efficiently remove the pseudo-deleted keys, while still maintaining read and write access to the table and index. If you don't need the space and your workload doesn't hit these keys, you are better off ignoring them and letting the index manager handle their removal.

Improving availability during data loading or data ingestion

When you are moving data into a table, there are several strategies that you can use to maximize the availability of the unaffected data in the target table. These methods are presented in order of the level of availability they can provide, from the highest to lowest:

- Use SQL INSERTs
- Roll in data using an ATTACH operation
- Use online loads

Use INSERTs to insert data into a table

While the high speed load utility can load data at even faster speeds, SQL INSERT transfer rates are very high and will often be sufficient for most needs. However, the advantage with INSERT processing is that it keeps your table fully available for read and write access. In cases where the base SQL INSERT may not meet your performance needs, consider employing other insert methods such as buffered inserts and array inserts, or coding your application to drive inserts concurrently from multiple connections.

Use ATTACH or ADD/LOAD to roll in data to range-partitioned tables

Table partitioning allows you to efficiently associate, or attach, a new or unused table as new partition of a range-partitioned table. Alternatively, you add a table as a new data partition and load the data into it instead.

You can use the ATTACH clause to streamline data ingestion by loading the new partition's data into the new or unused table and then performing any data cleansing activities needs through SQL, while the data remains in this table. These activities have minimal impact on the online workload accessing the range-partitioned table, because the new or unused table is not yet associated with the range-partitioned table. Once the data preparation activities are complete, this table can be attached as a new partition of the range-partitioned table, using the ATTACH clause of the ALTER statement. This is an efficient operation because it simply associates the existing table as a partition of the range-partitioned table, without moving any data.

After this transaction is committed, the new partition is not visible to online accesses until you issue the SET INTEGRITY statement with the ALLOW WRITE ACCESS clause. This will reflect the new partition's data in any indexes and MQTs that are defined on the table, while allowing read and write access against the table. However, if you are concerned about the amount of SET INTEGRITY logging required by an attach operation,

this may not be the best method to use. For more detailed information on rolling in data, see the "Data Life Cycle Management" Best Practices paper.

Another option is to add a table as a new data partition to the rest of the partitioned table and then load data into it. This avoids the need to issue the SET INTEGRITY statement, if the table has no constraints or MQTs defined on it. Use this option if you only require read access to the table during the operation.

Use online loads

During a standard load operation, the load utility locks the target table with a super-exclusive lock, whereas online loads allow read access. Use online loads for situations in which you require the fastest data loading possible as well as some degree of availability on the target table. To specify an online load operation, include the ALLOW READ ACCESS option with your LOAD command.

Improving availability during upgrades

Use the High Availability Disaster Recovery (HADR) rolling upgrade feature

You can use HADR to upgrade to a higher DB2 Fix Pack level while incurring only a minor interruption.

The procedure is as follows:

1. Deactivate the standby database.
2. Upgrade the standby database.
3. Reactivate the standby database.
4. Once the standby database is in peer state, issue the TAKEOVER HADR command on the standby database.
5. Direct clients to the new primary
6. Upgrade the former primary data as in steps 1-3 above.

Install new DB2 software in a different location on the same system

Since DB2 Version 9, you have been able to install multiple DB2 server and client copies on the same system. You can use this capability to install a new Fix Pack on a different path and roll over your production instances to the new installation path. This can be an effective method of speeding up the upgrade process if you are not using HADR.

1. Install your new DB2 product in a different path from your existing DB2 database production installation.
2. Shut down your production database instance.
3. Call the db2iupdt command from the new installation location to upgrade the instance.
4. Start the new database instance.
5. Perform any necessary post-upgrade actions like calling db2updv8 or db2updv9, or rebinding packages as directed by the Fix Pack installation topics in the DB2 Information Center.
6. Optionally, uninstall the earlier version of your DB2 product when you no longer need it.

Improving availability during backups

Backup operations can be performed when the database is online or offline. An online backup allows applications to have full read and write access the database while the backup is taking place. Although an online backup can take significantly longer than the default offline backup, there are steps you can take to shorten them.

There are two methods that allow you to reduce online backup times**: table space-level backups** and **incremental backups**. A table space-level backup backs up a specified set of table spaces only instead of the entire database. An incremental backup only reads pages that have been updated since the last backup.

Online backups cannot be run concurrently with a few other operations, as described in Compatibility of online backup and other utilities. However, there are two registry variables that you can use to prevent some of these incompatibilities: **DB2_OBJECT_TABLE_ENTRIES** and **DB2_TRUNCATE_REUSESTORAGE**. If your table spaces contain large numbers of objects (tables, indexes, lob columns), setting DB2_OBJECT_TABLE_ENTRIES to 65532 before you create DMS or automatic storage table spaces will help you avoid incompatibilities with online create index and online index reorg during the backup. If you plan to use the import utility with the REPLACE option during an online backup, set the DB2_OBJECT_TABLE_ENTRIES registry variable to IMPORT. Import replace operations are commonly used to truncate table content, and by using this registry variable you can improve availability during online backups.

A detailed discussion of database recovery may be covered in a future best practices paper on unplanned outages, as a restore operation is typically only performed due to an unplanned outage of the primary system.

Improving availability during performance tuning

Another maintenance task that can potentially affect availability is making changes to the database configuration. Fortunately, DB2 has automatic tuning capabilities, dynamic (online) configuration abilities, and registry variable settings that are intended to allow you to tune performance without incurring an outage.

Exploit DB2's automatic tuning capability

Several DB2 configuration parameters can be tuned automatically, which allows you to optimize performance without taking an outage. Although the automatic tuning algorithms can be left on permanently, a common best practice is to enable automatic tuning temporarily, allowing the self-tuning algorithms determine the optimal settings for the workload. Then, once these optimal settings have been determined, you can disable automatic tuning to ensure no further automatic changes take place. This strategy is well suited for environments with relatively static workloads. If you are using the self-tuning memory manager (STMM) in a partitioned database environment, enable it temporarily on a representative data partition.

Another key advantage of setting a number of these parameters to AUTOMATIC is that it prevents outage-like error conditions. For example, the AUTOMATIC setting eliminates the need to specify an upper bound. An upper bound could lead to an error message because a heap is exhausted even if there is enough free memory on the system. Although the system remains up and running, these types of errors may appear to end users as an application error. From a user perspective, these errors may indicate that the system is not available when that may not really be the case.

The following parameters can now be set to AUTOMATIC:

- applheapsz: The application heap can grow now until appl_memory or instance_memory is reached.
- database_memory: Starting in DB2 Version 9.5, STMM can tune the memory allocated to the database based on its requirements, within the limits of instance_memory.
- dbheap: The database heap can now grow as necessary inside the limits of database_memory and instance_memory. This is very important if you have a very large number of tables active in the database (like on SAP databases), that require some space from the dbheap for the TCB (Table Control Block).
- instance_memory: Starting in DB2 Version 9.5, this parameter now limits the overall memory consumption of applications and databases together. You need to be careful in setting this parameter to automatic if other applications are contending for memory. The database may release too much memory if other applications consume more and more memory, and this may reduce the ability of the database to handle large volumes of transactions or complex queries.

- mon_heap_sz: This prevents avoidable error messages because of missing memory for event monitors.

- stat_heap_sz: RUNSTATS and the real-time statistics require memory for the statistics collection. Because it can be difficult to estimate an upper limit, it is better to set it to automatic to avoid failing runstats operations.

- stmtheap: Complex statements require a large amount of statement heap to compile. Setting this to automatic will help avoid unnecessary errors during statement compilation.

Set DB2_OPT_MAX_TEMP_SIZE to 10240. Setting the DB2_OPT_MAX_TEMP_SIZE variable enables the DB2 optimizer to avoid large temporary space usage for sorts, if other access patterns (for example, sort by an index) are available. This reduces the likelihood of file system full conditions on the temporary table spaces and therefore decreases the possibility of SQL errors, helping to increase the overall availability of the system.

Use DB2's dynamic configuration capability

Many DB2 configuration parameters can be changed online, while the database is available. Online changes to database (or database manager) configuration parameters must take place within a database connection (or database manager instance attachment).

Set the DB2 registry variable DB2_OPTPROFILE to YES. If this variable is set before you start the DB2 instance, you can work around DB2 optimizer problems without the need for restarting the database instance. Many optimizer problems can be fixed by changing registry variables like DB2_REDUCED_OPTIMIZATION or by applying a new DB2 Fix Pack, but both operations require a short outage.

Improving availability through storage management

Use automatic storage or enable the auto-resize capability of DMS table spaces

These two technologies automatically add storage to table spaces when needed by automatically extending existing containers, or automatically adding new containers. This allows the database manager to handle full file system conditions automatically. In both cases full read and write capability is preserved and minimal I/O bandwidth is consumed.

Use automatic storage with a single table per table space

When many tables exist within a DMS table space, it can be inconvenient to reclaim space associated with individual dropped tables. With one table per table space, reclaiming the space is as simple as dropping the table space. Prior to the advent of automatic storage, such a recommendation may have introduced an excessive amount of management overhead, because each table space required its own storage management attention. With automatic storage, this is no longer the case because automatic storage table spaces are logical entities where tables can be created. Storage management for all automatic storage table spaces is performed in a single place, at the database level.

Keep in mind that table space page sizes must be one of 4, 8, 16 or 32 kilobytes. These values determine the maximum allowable row length for tables in a given table space.

Use the REDUCE clause of the ALTER TABLESPACE statement

The REDUCE clause can be used to reclaim unused space above a table space's high water mark (that is, the highest address that was ever in use in the table space). Under certain conditions, this ALTER statement can also reduce the high water mark without requiring the database to be offline.

Best Practices

Table and index maintenance

- Although reorgs can be performed online, they can be resource intensive. In many cases reorgs are simply unnecessary, so avoid them when possible.
- Use DB2 clustering technology such as MDC tables or clustering indexes to prevent or reduce the need for table reorgs.
- Set the DB2MAXFSCRSEARCH registry variable or keep your transactions as short as possible to reduce the need for space claiming table reorgs.
- Use the PCTFREE parameter or reduce the number of row-enlarging UPDATEs to reduce the need to remove overflow records.
- Use automatic compression instead of a table reorg to compress your tables.
- Convert any type-1 indexes to type-2 indexes
- Use an online reorg if you do need to perform a reorg.
- Use the CLEANUP option to reduce the scope of your index reorgs.

Data movement and data ingestion

- Use SQL INSERTs instead of a load operation to keep the target table available for read and write access.
- Attach a data partition or add (and load into) a table as a data partition when using range-partitioned tables.
- Use an online load if applications only require read access to the

target table.

Upgrade operations

- Use the rolling upgrade feature, if you have HADR.
- Install the new DB2 software parallel to your existing database.

Backup operations

- Use online backups to keep your database available.
- Use table space-level or incremental backups to shorten the length of your backup operations.
- Create all data, index and long table spaces as DMS or automatic storage table spaces, and set the DB2_OBJECT_TABLE_ENTRIES registry variable to the maximum setting to reduce incompatibilities with other concurrent operations.
- Set DB2_TRUNCATE_REUSESTORAGE to IMPORT if you are performing an import replace operation during an online backup.

Tuning and configuration

- Use DB2's dynamic configuration ability to keep the database online during configuration changes.
- Use DB2's automatic tuning capability to improve performance, while keeping the database available
- Set certain configuration parameters to AUTOMATIC to prevent error messages related to lack of resources

Storage management

- Use automatic storage or enable the auto-resize capability of DMS table spaces to have the database manager prevent full file system conditions automatically, while still maintaining full read and write access.
- Use automatic storage with a single table per table space or the REDUCE clause of the ALTER TABLESPACE statement to

simplify reclaiming space.

Conclusion

The database availability strategies you choose, and the resources you should invest to implement them, should reflect how tolerant your business is with regard to database outages.

DB2 provides several features and capabilities that are designed to help you to reduce or potentially eliminate the impact of some types of outages related to routine maintenance activities such as reorgs, loads, backups, upgrades, database configuration and tuning exercises, and storage management. Reducing the duration or frequency of outages will increase the time your database solution is available to serve the needs of your business.

A final note on developing strategies for avoiding planned outages: No matter how well designed a plan or process is, its successful implementation depends on a sound understanding of the technologies mentioned in this paper, along with rigorous testing. You can learn more about the high-availability capabilities of DB2 by exploring the resources listed below.

Further reading

- DB2 Version 9.5 for Linux, UNIX and Windows manuals
  http://www.ibm.com/support/docview.wss?rs=71&uid=swg27009727
- DB2 9.5 Version 9.5 for Linux, UNIX, and Windows Information Center at
  http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp
- DB2 Best Practices website
  http://www.ibm.com/developerworks/db2/bestpractices/

Contributors

Bill Minor
DB2 LUW Kernel Development

Sarah Packowski
DB2 Information Development

Dwaine Snow
Senior DB2 Technical Evangelist

Tim Vincent
Chief Architect DB2 LUW

Mike Winer
DB2 LUW Kernel Architect

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

Without limiting the above disclaimers, IBM provides no representations or warranties regarding the accuracy, reliability or serviceability of any information or recommendations provided in this publication, or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any recommendations or techniques herein is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Anyone attempting to adapt these techniques to their own environment do so at their own risk.

This document and the information contained herein may be used solely in connection with the IBM products discussed in this document.

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

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual

results may vary. Users of this document should verify the applicable data for their specific environment.

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

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

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

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