



CDB Software, Inc.

Data Replication

August 12, 2005

Why Do It?

Users of DB2 will always have the need to replicate data in some way. However, there are as many or more ways of replicating data as there are reasons for replication. Replication, as we will use it in DB2 terms, is broadly defined as copying or moving data from one location to another. For our purposes, our source data will always be on DB2 for z/OS. However, our destination may be the same DB2 system, a different DB2 system on z/OS, UNIX, Windows, some other RDBMS, or even simply flat files for archival purposes. DBAs are required to replicate data to these different platforms for several reasons and in several different ways. In some cases, as in a Data Warehousing system, there is the need to simply add or refresh data at certain intervals and not to replicate a full DB2 system or object.

For Decision Support Systems

A shop may set up history tables for future reference, or perhaps have a decision support subsystem. In this case the destination system would have the same DB2 ID as the source, but be running on a different LPAR. In these cases, there is usually very little update activity, but the queries on this system may be much different than queries in production. In this case the DB2 object structure may be different. There may be different indexes, separate bufferpools etc. This stands opposed to fully cloning data from production to offload the production workload where the DB2 structures would be identical and the queries would be the same from production to production offload.

For Application Testing

Data replication is necessary for application

testing and implementation as well. It is one thing to test on test data, but quite another to test with full production data and volume. The best and most simple way to test an application is to clone an entire production environment to a test DB2 system, for example from DB2P to DB2T. After testing in this easily refreshable environment, the results in production will be well known. Likewise, to migrate a new application to production, a reverse of this clone may be used. If the production data was created in test and the data is verified, a clone to production with a very fast cutover would be ideal. This same structure works well as a hot standby system via a software based mirror. This offers the fastest recovery from a disaster and the data loss can be dictated by the setup of the system. However, this usually comes with a very significant investment attached.

For Distributed Data

Yet another reason for replication has to do with distributed data. It may be necessary for a DBA to consolidate data from several RDBMSs to a single DB2 for z/OS system. Companies often do this to take advantage of large tablespace sizes, SQL parallelism and pure mainframe speed and security. However, the opposite need exists as well. DB2 installations have the need, at times, to sync applications on several different platforms or to copy data from DB2 to a data warehouse on a different platform or RDBMS. These systems are isolated from the production system so as to eliminate any impact on production resources, while maintaining access to the data if needed.

For Backup and Recover

Lastly, the most simple form of data replication is duplicating data to flat files. This takes the form of Image Copies for backup and recovery



purposes, flat files to be used as Load data for other destinations, or archival datasets for data warehousing or simply extended storage in archival format. The need for these can be based on company policy or government regulations depending on the industry. Regulation runs the gamut from audit requirements to SEC compliance.

Issues with Replication

So what makes the topic (and all DB2 topics it seems) interesting is the many issues involved in data replication. If replication were a simple process, there would be fewer options from which to choose, and fewer DBAs looking for help in the matter. Most importantly in dealing with DB2 for z/OS is the amount of data involved. Typically, the data that needs to be replicated is the most important data to a company. It is generally the largest as well. DBAs will run into problems with a tablespace being too big, number of changes too large, too much DASD or CPU needed to perform the replication, or simply too much time required. All DBAs work in confined windows of time allowable in a DB2 shop. Data replication, especially of large objects, almost never fits into these small windows.

Impact on Production

DB2 installations simply cannot afford to have resource contentions on production data because of replication. Managers care little about testing and procedures when a company application is unavailable and the company is losing money. In DB2, unavailability means lost money! Several of the options for data replication involve very intrusive methods that simply can not be allowed in an online production environment. These include system quiesces for system wide points of consistency(SWPoC), or offline utilities that

make the data unavailable while they perform their work.

Complexity

Often times, as explained in some of the examples, the requirements for data replication and output formats are extremely complex. DBAs have to deal with reformatting data taking into account several different platforms, object differences and OBID translation. All this must be done with a limited amount of available automation. There are several manual actions that must be performed as well as knowledge of the process that must be had to achieve the desired outcome. In every step there is ample room for confusion and mistakes. Often, data replication is a slow and tedious process that must be attempted several times before a successful completion.

CDB Solutions

CDB Software understands the needs and the problems associated with data replication, and has worked closely with its customers to provide a robust selection of options and automation for each need of data replication. Each product provides an automated and user-friendly means of performing the necessary tasks in the DB2 for z/OS environment. CDB exploits its technology to provide the fastest means to each of the reasons for replicating DB2 data.

Unload & Load Process

Unload

Unloading and Loading data from one object to another is one of the most basic types of data replication in DB2. This process is sometimes necessary when changes are made to the table



structures from one system to another, or reformatting of the data is required for data warehousing or archiving to another platform. CDB offers several options in this category. CDB/Auto-Online Unload allows for full SHRLEVEL REFERENCE Unloads of DB2 tables while never making the object unavailable. The object remains totally available throughout the entire process. It allows for flexible input files such as DB2 tables, image copies and DSN1COPY and allows for several output formats as well, including ASCIIIDEL, for replication to a different platform. CDB/Auto-Unload also offers powerful output filtering of column selection and order, as well as a standard SQL WHERE CLAUSE row selection that includes subselect support and ORDER BY sorting.

Archiving with CDB/Auto-Reorg

As an Unload requires a sorting of data, it is generally one of the slower options available. For archiving purposes, CDB/Auto-Online Reorg may be used as an alternative. CDB/Auto-Online Reorg also allows for archiving of rows while the tablespace remains available throughout the process. It offers a rich subset of SQL that also includes the subselect capability. As the archiving is performed during the CDB reorg, no sort is required and the process actually executes in less time than just a reorg. CDB/Auto-Reorg with discards takes advantage of CDB's dynamic allocation of output datasets so there is no manual modification of JCL from one reorg to the next. In addition, by taking advantage of CDB's automation capability inside the reorg, the user is able to tailor the process to his needs by, for example, notifying by email another platform of the file and number of rows to be loaded.

Load

The second part of these processes is sometimes the LOAD to another DB2 object. CDB/Auto-Load offers extensive data conversions so as to allow for different source and target table structures without the need for post processing. CDB/Auto-Load uses optimized tablespace allocation based on the input that eliminates manual calculation before each run. CDB/Auto-Load has the option of loading a table while the object remains available in RW or while the object remains available in RO mode. This leaves a data warehouse, for example, available at all times for application queries. In addition, CDB/Auto-Load can update catalog statistics and create a SHRLEVEL REFERENCE Image Copy concurrent with the Load process, eliminating the need for several jobs after the initial load and leaving object in RW immediately after the Load.

While this process is robust in features and abilities, it is generally the most intensive method available. In addition, the process works on a single object at a time. For applications that require an entire database move or simply a large number of tablespaces to be replicated, while CDB/Automation will make the task much easier, using UNLOAD and LOAD is still a tedious and laborious task with several better options available.

CDB/Delta: Replicating transactions for Data Warehouse

When replicating data for a data warehouse, often the changes between production and the DW site are very small compared to the overall object size. In addition, the source system must be available and completely unaffected by the



process used to update the DW site. When this is the case a data extractor or log extractor is a far better option. CDB/Delta provides this capability. Delta allows for a frequent update of the target system on demand or on a scheduled. Beyond simply leaving the source system unaffected, unlike similar products available, Delta does not even require DATA CAPTURE in DB2, thus alleviating a strain on the system. Delta offers a powerful selection criteria using familiar SQL in a WHERE CLAUSE to obtain the changes required for the DW site. CDB/Delta can provide several output formats such as DSNTIAUL, ASCIIDEL or SQL that allow loading to all platforms. In addition, Delta has the capability to eliminate intermediary changes, thus saving multiple updates to the same row and saving CPU resources. This also allows for comparisons from time A to time B of a specific field without the need for home-grown sophisticated applications.

CDB/Clone: Replicating subsystems or subsets of subsystems

When an entire subsystem or a subset of objects must be replicated, CDB/Clone is the preferred solution. When testing new applications on real production data, there is no better way of replicating an entire subsystem to a test system. In a single job with a single statement specifying source and target systems and any VCAT or DASD changes required, the entire manual process of cloning a subsystem is performed automatically. Any subset of a subsystem can be cloned, thus saving resources by only cloning necessary objects for testing purposes. As with Delta, the source subsystem remains completely unaffected since no QUIESCE is required. CDB/Clone will always clone consistent data to any RBA required with out any need to obtain a

System Wide Point of Consistency. Clone only replicates completed units of work, thus negating the need for rollbacks or post processing. CDB/Clone takes advantage of the CDB Advanced Log Processing Technology of preprocessing logs at regular intervals so that the replication CPU usage and Elapsed Time is minimized. In addition, the clone may be run on a remote target site with no CPU impact at all on the production site.

CDB/Active Standby: Mirroring

Data replication can also take the form of a recovery system. A DB2 installation can replicate data through the use of mirroring. However, there is a very large investment to obtain this functionality and once the functionality is obtained the mirrored site is useless, except in the event of a disaster. For a large investment, there is very little return. CDB/Active Standby offers software-based mirroring that leaves the target or mirror system active in Read-Only mode so that it can be used as an alternate production site for offloading applications and queries from the main processor. Active Standby is a great choice when very frequent updates of the target system are required and when the source system must remain totally unaffected. As only completed units of work are processed, the target system is always a consistent image of the production system at a previous point-in-time. This point-in-time is easily dictated by the frequency of updates set in Active Standby. The process is a fully automated solution with no manual intervention in normal operation that saves manpower and eliminates the scope of errors. Since it is a software-based system, the savings in hardware are easily apparent.



CDB/Auto-Restore

One of the fastest ways to replicate data simply and quickly is CDB/Auto-Restore. When moving data from a production environment to a test environment with no changes to the tablespace structure, or when recovering dropped objects to an image copy, CDB/Auto-Restore is the answer. Allowing for flexible inputs from Image Copies to DSN1COPYs to the underlying VSAM datasets, Auto-Restore offers several options for the source of the replication. CDB/Auto-Restore automatically handles internal id translation from source to target using the target catalog. All DBIDs, PSIDs and OBIDs are translated eliminating the need for a SYSXLAT table and all the troubles and errors that are associated with it. During the process of restoring the tablespace, indexes are built in parallel using small key sorts instead of a massive single index sort, thus saving processing time and resources.

CDB/Auto-Restore has several advantages over other options for moving data from one system to another. As stated, indexes are built in line with the tablespace restore so there is no second REBUILD INDEX step. The NPIs(Non-Partitioned Indexes) are built concurrently. Index keys are passed directly to multiple sorts run in parallel to minimize the time and resources needed. CDB/Auto-Restore can make a SHRLEVEL REFERENCE copy of the new object concurrently while restoring the data. The copy is registered in the target subsystem SYSCOPY making the object immediately available for RW access after the Restore. Likewise, CDB/Auto-Restore can collect statistics concurrently with the data restore. The target subsystem catalog is updated, saving an additional RUNSTATS after the process, making the object immediately available for the DB2 Optimizer and any queries

for applications. If a subset of an object is all that is need for testing or recovery, CDB/Auto-Restore can delete rows from the target object as they are restored. Using standard, feature-rich, subsets of SQL in a WHERE clause, the user can specify exactly what data should be kept and which data discarded again saving resources by only loading necessary data.

CDB/Auto-Restore v. DSN1COPY

Even though CDB/Auto-Restore is a replacement for processes like DSN1COPY, there is very little comparison between the two. CDB/Auto-Restore provides a long list of features not available in DSN1COPY. CDB/Auto-Restore rebuilds NPIS and the clustering index where as DSN1COPY does not. DSN1COPY requires a second, very resource expensive and time-consuming REBUILD INDEX. CDB/Auto-Restore collects statistics where as DSN1COPY does not. CDB/Auto-Restore takes a concurrent Image Copy where as DSN1COPY does not. CDB/Auto-Restore checks cross-page structures such as indirect pointers where as DSN1COPY does not. This partial list of features does not even mention the performance gained by running CDB/Auto-Restore. Contact CDB Software for actual benchmarks between the two products.

CDB/Auto-Restore in Action

Example 1: Final Acceptance Testing

For final acceptance testing it is a good practice to use actual production data. This is not normally a simple process, but CDB/Auto-Restore makes it easy. Assume the DDL has been run and therefore the structures exist in the test system as they are in production with no changes. This



should be the norm as the user is attempting to see how the new application acts on production objects. As a production object, there should be standard SHRLEVEL REFERENCE Image Copies being taken on regular intervals. The user simply uses one of these copies as input to a single Restore statement. Restore will lay down the image copy in the test system, automatically translating object ids, while rebuilding the clustering index. The keys for any NPIs are passed in small pieces to sort routines where multiple small sorts are performed and merged to form the new NPIs. As the same time, a new image copy is taken and registered in the test system SYSCOPY table. Likewise, statistics are gathered and updated in the test system catalog. The result is test data completely ready for application testing with no impact on the production system and no manual intervention by the user to translate IDs.

Example 2: Migrating a New Application to Production

A new application has been tested and data maintained in the test environment. It is now time to take the application and all data live. However, there is a need to minimize the outage during cutover. Prior to the cutover, the new object structures are created in the production environment. At cutover time, SHRLEVEL REFERENCE Image Copies are taken. A single Restore step that automatically translates the ids is run with the image copies as input. The tablespace is restored along with the clustering index. At the same time the index keys are passed to small sorts to minimize resources and time and the index keys are merged to form the new NPIs. At the same time, Image Copies are taken concurrently with the restore and registered in the production SYSCOPY. Statistics are

gathered at the same time and updated in the catalog. Binds can be run immediately following the restore job, leaving the object immediately online and the application running. During the process, cutover time was minimized, manual intervention was eliminated, thus eliminating the potential for id translation errors.

Example 3: Production Object Dropped

A DBA mistakenly drops the wrong object while performing maintenance on the system. After panicking and facing scorn and derision from coworkers, and worse, management, under immense pressure the DBA wisely runs the DDL that was saved as part of change control. By simply running a single step CDB/Auto-Restore with the last Image Copy as input, Restore rebuilds the tablespace automatically translating ids and rebuilding indexes as it goes using small efficient sorts. A new image copy is made concurrently during the process and registered in SYSCOPY. Statistics are gathered at the same time and updated in the catalog. The DBA simply runs his BINDS and is now a hero for saving the company immense amounts of money due to his quick thinking and use of CDB/Auto-Restore.

Data Replication with CDB

All of the CDB products that aid in data replication are designed to be simple, automated and robust in capabilities. Each product fits different needs as they arise in DB2 environments. CDB/Auto-Restore, however, fits a wide range of applications with its simplicity and automatic translation of object ids. Its ability to build concurrent image copies and gather statistics only ease the execution and increase the speed of the process. CDB offers the complete range of products to meet any need



with Data Replication that a DB2 Installation may have, from a complete subsystem clone to single object restores to continuous one off testing environments. CDB has the best solutions!

About CDB Software

CDB Software, Inc. is a leader in data management solutions for DB2 z/OS. CDB focuses its business on DB2 for z/OS to provide unique and innovative solutions that enable companies to expand their DB2 system to meet business needs while controlling the overall cost of the mainframe. Founded in 1985, CDB is a privately held corporation based in Houston, Texas with offices worldwide.

For more information visit:

www.cdbsoftware.com



CDB Software, Inc.
11200 Richmond Ave.
Houston, TX 77082