White Paper

Approaches to Cloning DB2 Subsystems

WHAT NEEDS TO BE DUPLICATED?

When we talk about cloning a DB2 system it is important to understand the components of such a system and how they act together. A Host DB2 system consists of:

- Two Bootstrap Datasets (BSDS) which contain information that is critical to DB2, such as the names of the active and archived Logdatasets and other information that describes the Logging.
- Several active Logdatasets. DB2 writes information about changed data and other important events into the Logdatasets. Their contents are basic for any recovery activity in DB2.
- Catalog and Directory Pagesets. These files contain the database tables where a DB2 system describes itself. For example, each Table, Column, View, Index and all their attributes are described here. Usually the DB2 Catalog is read-only for user applications and is only modified by DB2 programs.
- User Pagesets, containing the DB2 objects (tables, indexes, etc.) used by the applications running on DB2.
- Archived Logdatasets are written as soon as an active Logdataset is full and kept for recovery purposes.
- Image Copies—Backup Copies of any Tablespace or Indexspace in the system, including the DB2 Directory and Catalog. Together with the Logs, Image Copies are used for recovery of DB2 objects.
- Libraries, containing JCL procedures and load modules for DB2.

Usually only the first 4 of those component types need to be duplicated for a clone. Libraries can be shared between different subsystems. Archived Logdatasets and Image Copies always exist beneath the specific subsystem they belong to. Descendents (clones) of this subsystem should be able to use them for a historically correct recovery, though. Depending on the cloning approach, it is possible to clone a DB2 system by duplicating even less than the 4 component types.

WHAT NEEDS TO BE CHANGED?

Actually, not very much information needs to be changed. Apart from JCL modifications in the started task procedures of DB2, basically it is only the so-called VCATname. This is the High Level Qualifier where DB2's pagesets are stored. “The” VCATname is the first part of the names of all Catalog and Directory pagesets. There may be other VCATnames for user pagesets.

The VCATnames are stored in the BSDS, in three Catalog tables called SYSSTOGROUP, SYSTABLEPART, and SYSINDEXPART, and in many records of the DB2 Directory database, also called DBD. Therefore during a cloning process there have to be **internal** changes, modifying the names within the DB2 pagesets.

We also need to do **external** changes during the cloning process, because the VCATnames also exist externally as dataset names of the DB2 pagesets and libraries. These external names are to be changed during or after the physical copy operation.

In a non-SMS managed DB2 system, besides the VCATname, the names of the volumes where the DB2 pagesets reside may also need to be changed in the appropriate Catalog table. These volume names are also called VolSers (Volume Serial Numbers). As DB2 subsystems and their data volumes grow continuously, non-SMS managed DB2 subsystems will vanish.

DIFFERENT APPROACHES

There are different approaches for DB2 cloning. They vary in the type and number of duplicated DB2 components, in the method by which they are duplicated, in the method by which names are changed within the DB2 components, how the target DB2 is started after the change processes are finished, and—of course—in the time and efforts consumed for the whole process.

THE RECOVER APPROACH

This approach is based on the fact that DB2 has excellent recovery facilities.

The only components which are duplicated are the BSDS datasets. The target system gets a new set of Logdatasets. All other components are restored from Image Copies and brought to a consistent state using the Archived Log.

In order to achieve this, it is first necessary to copy the source BSDS datasets, using e.g. ADRDSSU. This copy may even be taken during the source system running. After that, new and empty Logdatasets are created. Now the copied BSDSes are changed using DB2 utility DSNJU003. A new VCAT name—i.e. the Highlevel Qualifier where the DB2 datasets reside—and the names of the new logdatasets are written into the target BSDSes. Furthermore, it is necessary to add a so-called CRCR record with an ENDRBA value to the BSDS. This record instructs DB2 to truncate the Log at a specific RBA on the next start. This RBA is the ENDLRSN of the last Archived Log in the source system. When the target DB2 starts, it will therefore backout all changes with a Log-RBA higher than that ENDRBA. In this way, our cloned DB2 will be brought to a consistent state. The administrator may influence this point in time by issuing an ARCHIVE LOG command in the source system immediately before copying the BSDSes. Another required step is to provide a DSNZPARM module with the DEFER ALL option in order to prevent the target system from modifying the pagesets of the source system while the names within the target system's components are still unchanged. Those source system's pagesets have to be RACF-protected in a way that the target system may access them in read-only mode.

After the target DB2 has been started with the temporary DSNZPARM module, the user's task is to recover the DB2 Catalog and Directory following the instructions in IBM documentation. It is essential to take care for a specific sequence. If user-defined Indexes on the DB2 Catalog exist, they have to be rebuilt Index by Index. When the Catalog recovery is finished, the User pagesets may be recovered, then a new STOGROUP with the new VCATname is to be created, and all tablespaces and indexes are to be ALTERed to use the new STOGROUP. For partitioned tablespaces and indexes the ALTER must run for each partition separately.

THE ALTER STOGROUP APPROACH

The second approach is very similar to the first one. The difference consists in the fact that not only the BSDSes, but also the Active Logs, DB2 Catalog and Directory, and the User Pagesets are copied. Therefore the recovery part does not exist in this approach. For the copying part one can use ADRDSSU in a logical copy operation, i.e. all datasets are selected generically based on their names. Another copying method can use ADRDSSU in the physical copy method, where entire Volumes are duplicated. Furthermore and if available, any facility providing Advanced Copy Services can be used, thus resulting in a very fast duplicating process.

The user will issue a SET LOG SUSPEND command in the source system to initiate a state where there is no update activity within DB2. After that the copy process will be started. Depending on the anticipated duration of the copy process, and therefore depending on the copy utility used, it might be required to shutdown the source DB2 instead of suspending it. In this way timeouts in the source DB2's applications can be avoided.

After the copying has finished, there is either a cataloged or an uncataloged copy of the source DB2 system's datasets. The latter is true if instead of a logical copy a physical copy (a volume copy) has been done. In this case the only way to proceed is to use specialized tool which is able to rename and recatalog all those datasets to an ICF Catalog.

As soon as all target datasets have been copied and re-cataloged, the user will proceed with updating the target BSDS. He will change the VCATname, delete the old active Logdataset entries and add new ones with the last copied active logdataset as the current one. Then a modified DSNZPARM with DEFER ALL, LBACKOUT=YES, and

BACKODUR=0 is needed. Again, the source system's pagesets will need to be RACF protected in read-only mode. When DB2 starts, LPL entries will occur on all pagesets. For the DB2 Catalog and Directory they can be removed with START DB commands. For the other pagesets first a new STOGROUP must be created. Then all user pagesets need to be stopped, ALTERed, and restarted. As soon as everything is finished, DB2 may be restarted with the regular DSNZPARM member.

THE CATALOG UPDATE APPROACH

This approach does the copying part exactly like the preceding approach. The same DB2 components are duplicated. After the copy the target BSDSes are changed in the same way as above. The temporary DSNZPARM need one more option to allow Catalog updates. After bringing the target DB2 up, the user will run a series of SQL UPDATE statements, working on the SYSSTOGROUP, SYSTABLEPART, and SYSINDEXPART Catalog tables. These statements will change the VCATname. In the next step all user pagesets are stopped. Then the DB2 utility REPAIR DBD with the REBUILD option is used to change the VCATname within the DB2 Directory. As last steps, the user pagesets are started, and the DB2 system is restarted using the regular DSNZPARM.

THE EDIT APPROACH

All the approaches we’ve talked about up to now were using SQL and DB2 utilities and were run through complicated procedures to change little information—only the VCATname. This is the only information that changes within the pagesets of DB2 when the system is to be copied. All other things like subsystem name and DDF information are changed in the procedures and BSDSes. So in most cases it is really the maximum 8 characters of the VCATname. Why not change them directly in the pagesets of Directory and Catalog?

This approach is called the “edit” approach because it uses editing functions—search and replace—to exchange an “old” VCATname with a “new” VCATname. Some education courses on DB2 cloning teach part of this approach, as far as changing the BSDS and the DB2 Directory is concerned. But also the DB2 Catalog can be treated in this way, thus eliminating every use of SQL or Utilities in this approach.

The method includes unloading the pagesets and changing the names. The most important thing is to change names in the right places. This requires extensive knowledge about the internal structure of DB2's pagesets in order to apply proper plausibility checking. So it is a solution for a software vendor who has access to this kind of documentation and wants to build a product.

Using the Edit approach, the DB2 system cloning becomes a straightforward action. After copying the source volumes and making them usable with a Cataloging Tool, pre-generated jobs run through all DB2 Catalog and Directory tablespaces and change names. Their processing is pretty fast, because they do not use SQL or DB2 Utilities at all, they just use the good old access methods like VSAM and SAM. No temporary DSNZPARM is required. After the change edit job has finished the DB2 starts up and works under its new name.

WHICH ONE IS THE BEST ONE?

As discussed above there are several ways to generate a clone of an entire DB2 subsystem. The question seems to be, what is the most efficient approach; efficient in regard to resource consumption, execution time and minimal manual effort.

DOING THE EXTERNAL RENAMES EFFICIENTLY

An efficient way to copy the data of an entire DB2 system is to copy 'volume wise', i.e. to copy the whole system volume by volume, either with native ADRDSSU or by means of advanced copy services like Flashcopy, ShadowImage or similar. To copy the volumes instead of the datasets is usually faster and requires fewer resources. Another important aspect is the period of time needed to quiesce the original system during the copy process. On the other hand, a volume copy leads to the problem of uncataloged datasets on the copied volumes. These datasets cannot be cataloged under the same Master Catalog without preceding renames. Thus, the performance advantage of 'volume copies' comes along with some additional work after it. All the datasets on all copied volumes must be renamed and then cataloged.

Obviously, the most attractive approach for the copy process would be a programmed procedure that executes mainly automatic, perhaps controlled by a parameter file. This procedure could,

- generate the copy jobs, using copy services available in a given environment,
- execute the copy jobs,
- rename all datasets on copied volumes, adapt VTOC and VVDS, and finally,
- catalog files on copied volumes.

What information does the procedure need for these tasks? First of all it requires the volume names of the source system and the names or addresses of the volumes for the target DB2 system, the clone. These pairs of volumes could be read from a parameter file as well as the "renaming rules" to be used for the datasets. With this basic information and some more peripheral things, like the name of the new user catalog, a programmed procedure can make a "physical" clone of a DB2 subsystem. Physical in the sense that there are still more actionsrequired to provide a DB2 clone ready for start-up. The procedure discussed so far does all the work necessary to make the copied datasets usable from the perspective of the operating system. In this state the DB2 clone would not yet work properly, because its system database and its bootstrap dataset still contain the names of the datasets from the original DB2 system. A further procedure is required to fix this.

Renaming thousands of datasets on hundreds of volumes with standard utilities can become extensive, time- and resource-consuming work. It is more appropriate to load VVDS and VTOC into main storage, do the necessary changes and reload them to the volume. Pretty much the same is true for the subsequent catalog task. The catalog functions are stressed out when thousands of inserts occur within minutes. It is better to read a copy of the exported original user catalog into storage, change it by a dedicated program and construct the user catalog for the clone this way. By this means it is feasible to change a volume within seconds and to construct the user catalog within minutes.

The execution time may be further shortened by parallel processing. Several jobs can work simultaneously on subsets of the volume pool. In order to ease usage of job schedulers to control the schedule, it is advisable to denominate the jobs in a way that a number in their names and member names suggests their execution order. For example, the root job containing number 1 generates the copy jobs which in turn receive the number 2, the jobs which change the volumes carry number 3, number 4 creates and populates user cats, etc. Thus the rule to define requirement/dependency for a job scheduler becomes simple: Jobs having the same number can work simultaneously and execute all n jobs before the n+1 jobs.

THE DB2 PART—INTERNAL RENAMES

After all datasets of the clone 'in preparation' are available under z/OS the DB2 part of the process must be accomplished. Many entries in the tablespaces of the system database of the potential clone are still wrong, they 'point' to the original datasets of the source system. Till now, in the preceding z/OS part, we only changed the names of datasets, we didn't go into them. Programs are required to change those wrong entries within the tablespaces of the system database. These programs can be controlled by the same 'rename rules' which were already used in the make usable part above. Additional control statements are required to govern the VCAT name, the name of the new subsystem itself, and if we deal with a shared environment, group name and member names.

Thus we propose to write dedicated programs for the inevitable changes in the system database and bootstrap datasets. These programs, embedded in a further group of jobs, shall completely adapt the new DB2 instance, the clone of the original, to the naming of itsdatasets. This way a preliminary execution of the new DB2 instance in maintenance mode, to adapt the system database to the requirements of the denomination of its datasets, will be avoided.

The approach suggested leads to a dynamically generated job chain which leaves behind a new DB2 subsystem as a clone of the original.

IN-FLIGHT COPY

Is it possible to use the suggested process to copy a living production system without shutting down the databases or using LOG SUSPEND? Yes, in principle, although it is not advisable to invoke the copy during periods of high activity, just as it would make little sense to copy with slow copy functions.

An advanced copy service, which is able to copy the whole set of volumes consistently to a timestamp, including the Usercatalogs, is one premise. The other is to have a reachable consistency point in the log to where the clone later can be recovered to. Therefore, before the copy/split is invoked an ARCHIVE LOG should be issued. During the subsequent making of the clone there is no significant difference to the usual procedure: Copy, rename, catalog, adapt DB2. The copy/rename includes the active logs. When the clone comes up it will consult its logs and start recovery. This recovery is completely based on DB2's own routines.

WITH A LITTLE HELP

The process outlined so far is a controlfile-driven pure batch process. In complex environments the writing of the control file with all the names (source:target) for volumes, ICF Catalogs, log copies, bootstrap, VCAT, etc. becomes a tedious task. It is conceivable, for simplicity’s sake, to use an ISPF-based program to generate the control file.

This small application basically asks for the subsystem-name of the source system and then gathers all names needed self-acting from the source system and from SMS. It is particularly helpful for the long list of volume names. The user may specify rules for the conversion of names from source to target system, which then will be used by the program to write the control file.

During the cloning of large systems with hundreds of volumes a lot of jobs may be underway to produce the clone. Message and condition code checking may come on strong. It makes sense to enhance the mentioned ISPF-program with a journal feature which collects the messages from the jobs and presents them filtered and sorted as desired.