How to Make Test Data Management Work

Every company must do QA and testing. Whether it's in-house applications, business logic in your database, or vendor products, all new code and all changes to existing code must be tested thoroughly and with realistic test cases. To do it right, you need several different environments that have varying requirements as to the amount of test data, the frequency of refreshes, and other things.

Divide and conquer

There is no one silver bullet piece of software that handles all the different requirements for your test and QA databases equally well. An application that is optimized for moving terabytes of data in the fastest possible way will struggle when it comes to copying a single test case consisting of just a few dozen rows from a number or RI-related tables. But both scenarios are important for proper and thorough testing because developers need to run unit tests against a small body of masked data as part of their development process, and the final acceptance test needs a mirror of the entire production database with real data inside.

The key to efficient test data management is to set up different environments and implement processes where data from one environment is used to feed downstream environments. This results in single points of control for several important aspects of your data provisioning processes, including production data access, data consistency, and data masking.

One common mistake it to try and do everything at once. If your requirement is to have a single process that copies a subset of data from an environment with 10,000 tables, and to do it in a consistent way with respect to ongoing transactions and without restricting write access to the source environment, and to mask data in the process consistently across all tables including key columns, then you might be trying to bite off more than you can chew. By splitting the task into multiple smaller pieces that are more manageable, your process will become more robust, reliable and maintainable.

Identify the different environments

Production

There should be only one single test data management process that accesses production data, and it should be the process that creates the so-called pre-production environment (see below). This not only creates a clear separation between the production data and test data, but it also ensures that test data provisioning processes cannot affect the performance or the availability of the production environment in a negative way.

Do not allow random test data extraction processes to access the real production Db2®. When data is extracted by using utilities like UNLOAD, this will consume valuable CPU cycles in your production LPAR. Using SQL is even worse because the queries that are used for selecting test data will differ from your typical production workload. This means that, in addition to the increased CPU usage, both your dynamic statement cache (DSC) and your buffer pools in production will take a hit. As a result, you will see bad response times for your production applications, and it will take some time until everything is back to normal.

Pre-production

A pre-production environment is a full clone of a production environment. Other names for this environment include golden copy or master copy. The purpose of the pre-production environment is to have an identical twin of the production where final acceptance tests can run against the real body of data.

You can also use it to test how to migrate your production environment to the next version of Db2, or how changes in ZPARMs affect production applications.

In addition to that, the pre-production is the environment from which other test data management processes read data. The cloning process that creates the pre-production environment is typically done on volume level, and it includes all the volumes that contain your Db2 table spaces and indexes (including the Db2 catalog and directory), the BSDS, the active log, and the core libraries like SDSNEXIT.

Obviously, the pre-production requires the same amount of space as the real production environment. Some Db2 shops shy away from acquiring more storage, but the benefits of having a pre-production usually outweigh the cost.

A volume level copy works outside of Db2 and therefore does not consider which transactions may be active. To make a consistent clone, you can employ one of two different strategies:

1. Restrict write access to the source for the duration of the volume copy. This is usually done by suspending the log activity of the source Db2, and resuming it after the copy. This option is usually only feasible for smaller shops who have a nightly batch window in which no application requires write access to production.
2. Make a point in time copy of all involved volumes. You need access to hardware-assisted copy technologies from your storage vendor for this purpose. This method has no impact on the availability of the source environment. Even though a point in time copy does not guarantee that there are no ongoing transactions at the chosen point in time, Db2 itself has mechanisms to restore consistency when it is restarted.

The pre-production environment should get a new SSID, and it is good practice to change the volume serials, the first level qualifier of all involved data sets, and DDF parameters. This ensures that the preproduction is not confused with the actual production.

Obviously, you must tightly control who has access to the pre-production environment. This includes authorization on database level, but also on file system level using RACF or a similar product.

Integration environments

Integration environments are smaller environments that only contain a subset of the tables that you have in your pre-production. If your company is divided into several business units, and an application is specific to one of these units, there is no need for a full clone of all the production data to conduct testing. For reasons of space, security, and flexibility, you want to have a copy of your master data and a copy of all data that is relevant to the specific business unit.

Typically, there are many different integration environments. They can share the same Db2 subsystem, as long as they each have their own copy of the data. This ensures that testers don't step on each other's toes, for example by modifying each other's data or by causing congestion issues, when running their tests. In order to have multiple copies of the same set of tables in one Db2, you typically assign distinct schemas to the objects that belong to one environment or one group of testers. Keep in mind that it is not enough to change the object names themselves. You must also change the names in all references to those objects, too. Think of views that refer to other tables by name: not changing the references creates unwanted cross-references to the wrong tables.

You need to have a process that not only moves the data, but can also create new environments quickly – with or without renaming – and check the compatibility of your source and target objects. This becomes particularly important when you employ fast copy tools that move data on file system level. For many environments, that is the only feasible option because traditional ETL tools are not fast enough to handle the sheer amount of data and the desired frequency of refreshes.

Make sure to consider all the little bits and pieces that need to be done before the target environment is usable. You must take care of identity columns and sequence objects to avoid unique key violations. If the process results in any restricted states on one of your target objects – which often happens if you rely on traditional methods like the LOAD utility – then it is necessary to reset this state before you can have full access. If you opt for a data set level copy, make sure you handle table versioning in the target environment properly in order to avoid errors when accessing the target tables. Again, the key is automation because what exactly needs to be adjusted and how may change every time you repeat the copy process.

Local instances for unit testing

Modern development environments include the ability to carry out unit tests on an automated build platform, or even on the developers' machines. Your developers might have a local instance of Db2 for LUW® on their machines, or maybe they use zPDT® and run Db2 for z/OS® in an emulated LPAR. In any case, these developers need specific test data to run unit tests and component tests, but for space and performance reasons it does not make sense to give them a copy of a multi-terabyte database to work with. Instead, these developers need just a few rows from some of the tables, but these rows must be carefully selected so that they constitute a business object, an entity, or in general something that makes sense from the point of view of a unit test. An empty shell of a customer record that has no associated orders and invoices is a good edge case, but you also want something more realistic for your testers.

The real challenge lies in identifying which rows must be copies from which tables after specifying an “entry point”. Referential integrity constraints provide information about the relationships among tables. Some relationships might be hidden in application logic. In this case, you must identify and formally describe them so that an automated process to work properly.

In addition to identifying the required rows, this process must also handle existing rows in the developers’ target tables. Usually, they are not supposed to be deleted. This is the reason why the copy process must be able to deal with duplicate keys and either merge the extracted data into the target tables, or append it by generating new keys. This, of course, can also affect rows from dependent tables over multiple levels.

Make a refresh strategy and determine data masking need

The different environments do not need to be refreshed at the same pace. It is not uncommon to refresh the pre-production on a quarterly basis, the integration environments weekly, and the smaller unit test environments many times per day. By keeping the data in pre-production static for a while, you can ensure that every time one of the smaller environments is refreshed, the process basically “resets” the data back to its old state, which can be desirable if tests should run repeatedly under the same preconditions.

To populate an environment, use data from its immediate neighbor upstream environment. This hierarchy of environments allows you to have one single process that accesses production. It is much easier to schedule a single process and to secure it properly than to take care of dozens of processes that run at different times and pull different data from production.

Similar considerations apply to data masking. If you create one copy process that includes data masking, the resulting environment can be used as a data source for other downstream environments. Copying from the masked environment means that all the data in the target will also be masked without you having to take any extra steps. By dividing your environments into unmasked and masked categories, you will only have to apply masking if a copy process crosses the line between unmasked and masked environments. In an ideal setup, there is exactly one such process, creating a bottleneck – in a positive sense – through which all unmasked data must go before it appears in the environments that are used by the majority of testers and developers. As a bonus, it becomes much easier to properly implement access controls properly.

Implement your strategy

Setting up test data provisioning processes is a challenging task. But there are tools that can do the heavy lifting for you. ESAi / UBS offers highly effective solutions that exploit the characteristics of the Db2 for z/OS database and the system it runs on.

BCV4™ creates full clones of a Db2 for z/OS subsystem or data sharing group. If hardware-assisted copy mechanisms are available, the clone will not impact the availability or performance of the source system. The cloning process can be scheduled and executed periodically without any user intervention. It renames volumes, VCAT prefixes, SSIDs and DDF attributes, thus creating a completely independent new Db2 subsystem or data sharing group that is an exact twin of the source. The target typically becomes ready one to two hours after the cloning process has been started. The required time is almost independent of the amount of data involved.

BCV5™ quickly copies table spaces and indexes within one Db2 for z/OS subsystem or from one subsystem to another. It has intelligent automation that creates missing target objects, automatically checks the compatibility of existing target objects, and then makes file system level copies using a built-in copy program that is 10 times faster than vendor UNLOAD/LOAD based processes, and features parallel processing and automatic workload balancing. Objects can be renamed in the process, and it can copy from the real source page sets or from image copies. If required, it can use information from the Db2 log to make consistent copies of your source objects without restricting access to them.

XDM™ allows you to copy subsets of rows that are consistent with respect to RI constraints, and to modify rows in any imaginable way before they are inserted into your target tables. In addition to Db2 for z/OS, it also supports Db2 for LUW (including AIX®), Oracle®, and MSSQL Server, and – with connectors, IMS® databases and VSAM® data sets. Like BCV5, it also includes the ability to create missing tables for you, check the compatibility of existing target tables, and to rename tables in the process. On top of all that, XDM gives you the functionality of BCV4 and BCV5 for platforms other than Db2 for z/OS.

Summary

Test Data Management is not necessarily a daunting task! By dividing the work into smaller, more manageable pieces, you can set up a robust chain of processes that supply useful data to all the different environments you use for testing. The focus for all processes must be automation, reliability, repeatability, performance, and ease of use. ESAi provides the Test Data Management suite of tools that help you accomplish all your test data provisioning and data masking tasks while avoiding negative impact on your production systems.

Speed, ease of use, and intelligent automation that significantly improves test data provisioning and development timelines.

Contact Us to Learn More and Schedule a Free Product Demo.

Call (866)464-ESAi or email us at info@ESAIGroup.com