DB2 Data - Test Data Generation and Archiving: Two Underappreciated Arts

Susan Lawson, YLA

There are many facets to a DBA's work and responsibilities. We give focus to the most apparent and critical such as backup, recovery and data integrity. But what about the other parts of the job? Tasks such as creating accurate test data for proper application testing and the ability to archive inactive data for better storage are often critical requirements for an applications success, but are too often put on hold, or not done at all. Also, these two tasks sometime get overlooked because maybe they are not necessarily the responsibility of the DBA, but more an application task. This only makes sense because the ability to generate test data for an application run is certainly more the responsibly of the application because they know the scenarios and the type of data they need to test in order to ensure the code is working properly. With archiving, again this often has been a responsibility of the application area, but often not done because it is usually the last thing on the specifications. These days we have more of a need to be able archive and easily query the archived data; and the application area will see most of this responsibility.

There are two products by IBM, the DB2 Test Database Generator (TBDG) and the DB2 Data Archive Expert (DAE), that can help with both of these tasks to help ensure they are done in a timely manner, and are done with accuracy. We may have been able to ignore these tasks in the past, but with the data and transaction growth rates and performance demands we are experiencing today we need to have an automated way to ensure these tasks are done.

Test Data Generation

Let's start with the ability to generate test data. Why is test data generation such a critical task? Why is it so difficult? Whose responsibility is it?

Let's answer these questions in reverse. First, the focus of this paper is both on test data generation and data archiving, which are both more of an application responsibility than one of the database administrator. Test data generation needs to be set up and controlled by the application programmer with a clear understanding of how the users plan to store and use the data. Of course, the DBA needs to be involved in the creation and maintenance of the tables, but in order to properly test the application code, the programmer needs direct involvement in the generation of the test data. Second issue: difficulty in test generation. It is difficult again because often we do not have the proper people involved in determining what data is needed for a true application test and we do not have easy, automated ways of generating and maintaining this type of data. This is of course is where a tool is needed.

The first question of why test data generation is such a critical task can be answered with another very famous question – 'It ran fine in test, so why did the application die in production?' It is a critical task to be able to generate accurate test data in order to really determine how an application is going to act in production. This means that we need to test against representative data both in volume and in data value distribution. In today's environments that requires high performance for extremely large transaction volumes, as well as access to extremely large data stores. But if we are performing our testing on less than accurate data we can get some unpleasant surprises when we get the application into production. We also may need an accurate sampling of data to debug an application problem or test for a particular data problem. And of course one of the main reasons we need to generate accurate test data is to be able to do code validation.

Some of the problems with proper test data generation include the ability to generate proper sampling of data from a variety of sources, ability to generate representative volumes of data, as well as dealing with secured data. In the past we have built homegrown systems to do this, or the task has simply not been done (or done poorly) due to the time and effort needed to do it accurately.

There are also HIPAA regulations that force organizations to ensure all sensitive data is appropriately protected. These organizations need test data for many reasons, one of which is to properly test applications. There needs to be an easy way to help these organizations create good test data while maintaining integrity. The TDBG tool can help create scratch data or manipulate real data so it cannot be attached back to production data. For example, there may be a requirement to test data in a hospital environment where the data cannot be tracked back to the actual patient. In addition there may be the need to further encrypt the data for HIPPA compliance and this can be done with an additional IBM tool called the Data Encryption Tool, which is not discussed in this paper but more information can be found on the IBM tools website.

IBM DB2 Test Database Generator

The DB2 Test Database Generator tool can be used to provide a way to obtain data from thousands of database objects in order to create test data. This can be done through setting up rules and actions either for data extraction or data generation. Through the use of generation rules there are ways to properly change (i.e. transform or randomize) and mask data for the utmost flexibility and security. Various input and output formats can be used for the creation and storage of this data. With this tool we can generate test data on demand or create scripts that can be run at any time, multiple times.

We can use the tool for a variety test data generation activities such as creating test data in new or existing tables, debugging application failures that are data dependent, securing sensitive data for testing (i.e. masking values), creating test data in different output formats or into a restructured database for testing, or copying pieces of data for a specific test.

Relationships between data can also be discovered and used with the DB2 Grouper component. This allows for both DB2 enforced referential relationships as well as application enforced relationships.

Components of the DB2 Test Database Generator Tool

The DB2 Test Database Generator (TDBG) tool supports multiple interfaces (ISPF, GUI, Offline/Batch) and it is supported across multiple platforms (z/OS, Linux, Unix, Windows). Sources for the data can be DB2 table or files (CSV – common delimited), text file with delimiter or fixed length, and in the future XML).

It consists of a client and a server component. The client component can be an ISPF application running on z/OS (figure 1) or a GUI (Java) client running on Linux, Unix or Windows (figure 2).

Figure 1. ISPF Client Interface

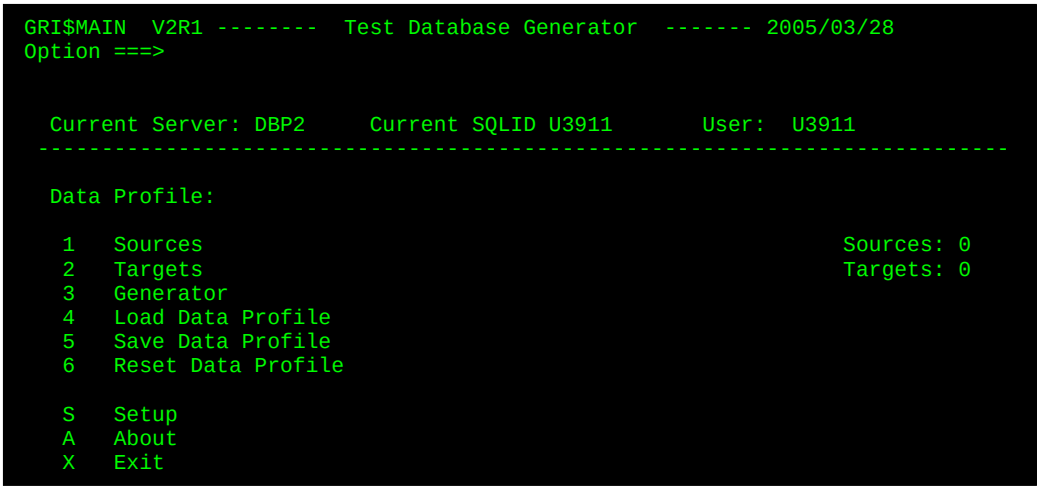
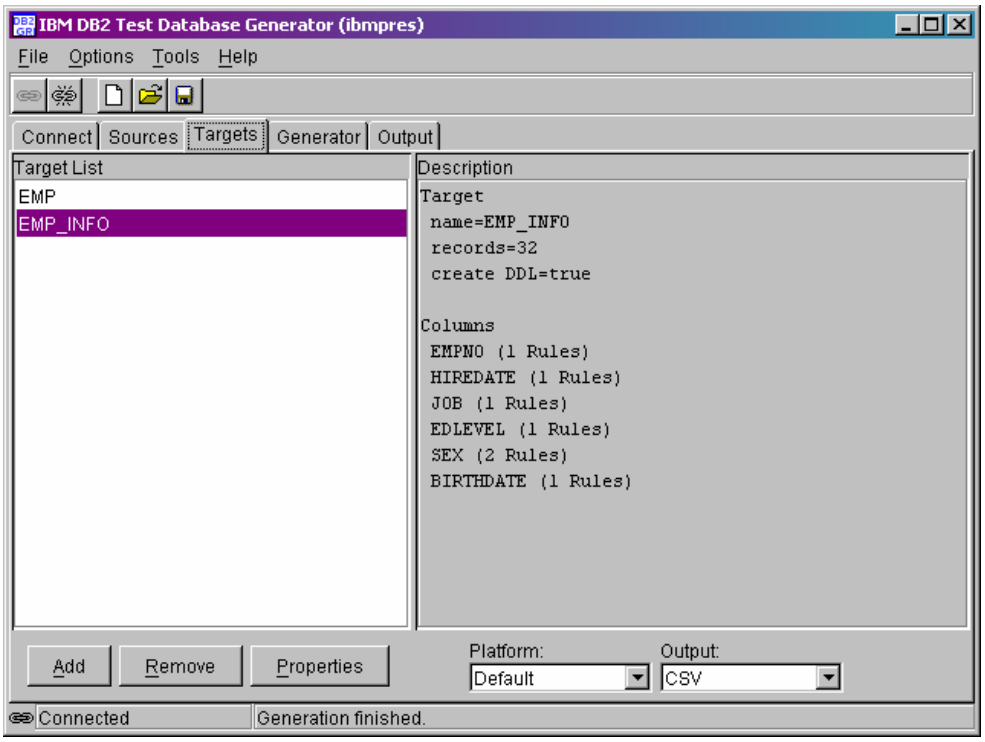
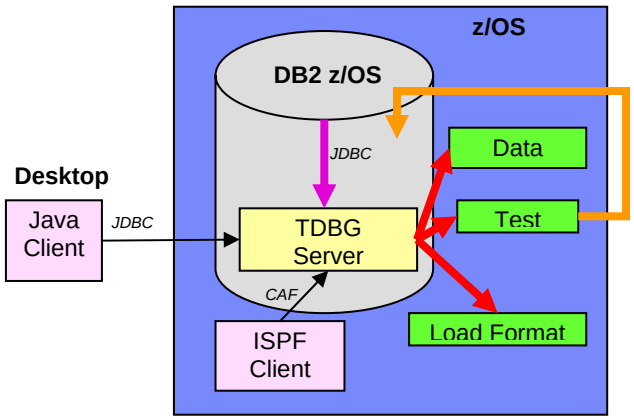


Figure 2. Java GUI Client



This client piece provides the user interface wizard and panels to guide you through the creation of Data Profiles (a component that defines the generation process which is discussed later), and then can connect to the server. The server component is installed on either z/OS, Linux, Unix or Windows as a set of DB2 stored procedures. This component connects to and reads from data sources, and it will read the Data Profiles and generate target test data. It is also used to monitor the progress of the data generation, and can create DB2 tables. Figure 3.

Figure 3 – Operating Environment/Components



The Data Profiles are defined in order to provide the generation process. This includes the source data object used to see the data generation, the definition and relation rules for the targets. They will also be used to describe how the data will be copied, filtered,

masked and transformed. These profiles are stored on the server component and are written in a Test Database Generator Markup Language.

Developing a Method for Test Data Generation

Before test data can be generated it must be determined by the business (users) and the application what is the best test data required. This will include knowledge that test cases that are going to be using the data, as well as understanding the data relationships and rules of any necessary transformation.

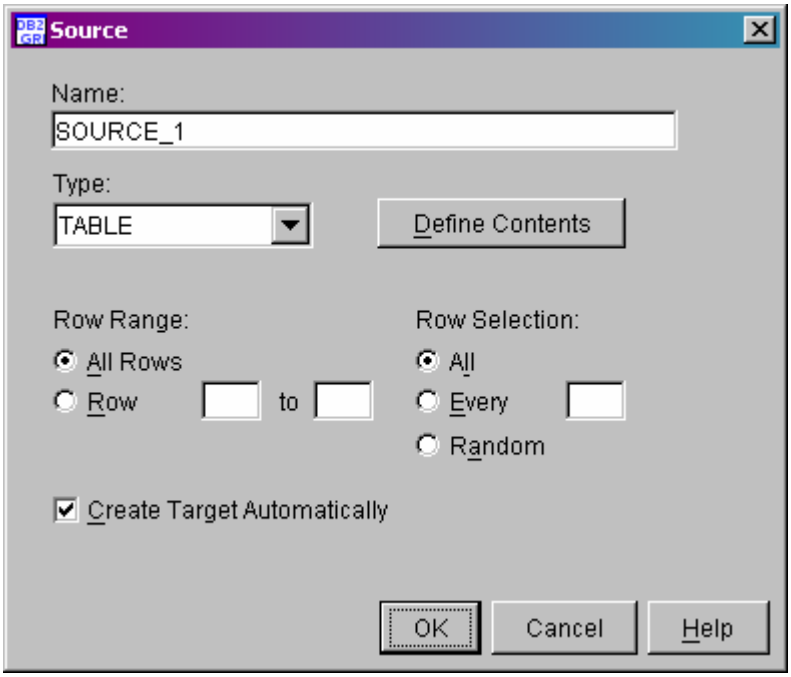
There are two ways to construct data with this tool: new or from an existing data source.

Creating data from existing source

We can create data from an existing source in our organization. We do need to have a good knowledge of this data in order to understand the relationships and representations. Then we can apply any necessary transformation rules to create our test data. This would require the knowledge of both the existing data and the future applications, so it's possible you may need to have users (legacy and new system) and application programmers in the process.

The actual sources can be table or files. DB2 tables are currently supported and soon Informix sources will be supported. Other files types supported are XML(soon), CSV, delimited text and fixed width text. The data can be accessed via various protocols such as JDBC, HTTP, FTP and file. Data can also be specified directly (rows and columns) or copied/pasted from the GUI interface of the tool. There are multiple levels of filtering that can be done. One way is using a WHERE clause when the source is a table. Others include using an all inclusive or range feature, or specific row selection such as every row sequentially, every nth row or rows randomly. In figure 4 below you can see the GUI screen that would allow you to specify this very simply. There is an option also to start with the creation of test data from scratch.

Figure 4 – GUI Screen for Source Definition



We can then choose to include related tables, both those that are RI defined (DB2 enforced) and those that are application enforced. Any number of sources can be defined as related. The product will also discover related tables and these can be edited if necessary. When copying related rows this can be done across the entire set of tables, or a set of specific rows across tables, or rows that are missing relationships (application maintained relationships).

The transformation rules are then established in order to define the target test data. For example, let's say we want to create some data that represents credit card numbers, but we want to mask the first 12 digits with 'x's. We can set up the target column to only have the last 4 digits displayed by setting up a data masking transformation rule to do this. These rules specify how to generate the test data from source or from scratch. There are seven transformation rules currently that can be used:

- Static values – specifying data value
  - Set target column COUNTRY to USA)
- Source column values – generate target based on source
  - Use source column value as-is
- Data lookup – replace data based on table lookup
  - DEPT_NAME = LOOKUP(DEPT in LOC_TABLE)
- Data masking – hide or replace secured data
  - Replace last 4 of social security number with X's
- Expressions - ability to call a database function
  - Use calculation or manipulation – HIRE_DATE + 1 YEAR

- Random values – generate random values
    - Can specify range and are propagated across related tables
- Pattern generation
    - Generates data based on a specific pattern

Generated Data

The output from the Test Database Generation tool can be a comma delimited file, fixed width, text delimited, SQL INSERTs, DB2 for z/OS internal LOAD format, or an XML file. The generated data can also be directly inserted into a DB2 table.

You can also specify the number of rows to generate, the structure of the target table/columns or if the structure is inherited, and if needed, the DDL (which can be modified) for the target tables including all primary and foreign keys.

Implementation Example

Ok, so now we know the components of the TDBG product and how it can be used to generate test data. Now let's look at business example of how we can make use of this in our organizations. Let's say we are a credit card company and we want to be able to test our application against a sizable amount of data and we want to generate this from existing data. Some of our criteria for this test data are that we need to split the existing source table into two new tables and that we will also need to mask the first 12 digits of the credit card number.

Using the GUI screens provided by the tool we would first need to specify the source (CREDIT_CARD table) and the fact that we want all the rows. A query will be generated to perform this task. Then we would specify the targets (ACCOUNT_INFO table and PAYMENT tables) by adding them and removing any unnecessary columns (since we are essentially spitting the source into two tables), and then we create the DDL for the targets. Then we need to specify the specific rules to copy the columns and mask the data. And last, we will then actually generate the data.

This would now map the existing data to our two new target tables that were created and we would have our credit card numbers masked in the new test data. Since we copied all rows from the source we now have a true representation of our volume and a good environment for testing our application with secured data.

Data Archiving

Storing data online has become more of an issue than it ever has been, however we still need, at times, to be able to archive this data and recall it when necessary. There are several manual ways to do this, however many of them never get implemented, as we are always ”on to the next project/problem.”

One of the biggest problems facing us today is that as our very active data stores grow larger so does the amount of data that we must keep as inactive. This inactive data must not go away, such as deleted data does, but can be stored on less expensive devices, but of course not be in the critical path of heavily accessed data. In today's high volume environments we have to separate data stores, not only for our critical processes, but for our most critically accessed data for high volume applications that need to push hundreds of millions of transactions through in a single day. Even with today's volumes in the multi-terabytes and approaching the petabyte range, we still have the need to have less accessed, less active data, somehow "removed" from our production environments.

We also have new requirements dictated by Sarbanes Oxley (SOX) which requires organizations to keep more data than ever before.

The ability to have increased disk capacity is not always the solution because not only are there physical limitations, but there are monetary issues when it comes to the additional hardware cost and maintenance. How many years have you been saying that you cannot throw hardware at a business problem? That has always been true for CPU usage/throughput and it also applies to disk/storage problems.

However, we cannot continue to afford to have 'unwanted data' sitting on our mission critical disk because it can, in the end, elongate our batch processes and due to the fact that we no longer have a "batch" window, this can cause an outage we cannot afford. Even well indexed database applications can become less inefficient as data stores grow larger -- not to mention the cost in terms of unavailability and performance when it comes to maintenance (i.e. REORGs, RECOVER, COPY).

But I do backups, what is the difference? The difference is that backups are made of data that will remain on the database. An archive will selectively move the data to another data store and it will remain there for potential recall at a later point in time. Archiving can actually become part of a backup/recovery strategy even in a low volume/availability application architecture. Archiving can also be set up to be by application, not necessarily by database. This would give us more flexibility to meet the requirements of the application.

The hardest part about data archiving, especially by application, is how to perform the needed selectivity and how to retrieve the data, on demand, in the most efficient manner possible. We need the ability to create a process that can do this when necessary and maintain all necessary relationships. Due to the complexity of these types of requirements, we often never get a process in place to do this and just simply end up dealing with the problems of large data stores.

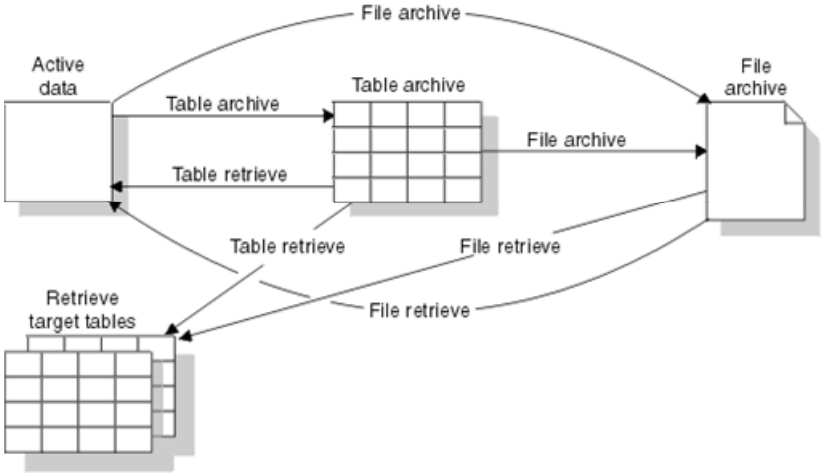
IBM DB2 Data Archive Expert for DB2

The IBM DB2 Data Archive Expert for z/OS is a product that can address this issue. It allows us to easily create a method for performing timely and efficient archives of our

inactive data selectively. Once we effectively set up a method for archiving we can immediately see the benefits through reduced cost and storage needs and possible performance improvements with utilities and processes that scan large amounts of data.

This data can be archived to either DB2 tables or to flat files. The table archives can be accessed in the same manner as the tables that are holding the active data. These tables can sometimes be thought of as historical tables. We have had the need to use history tables in many database applications, and the DB2 Data Archive Expert allows us an easy way to populate and maintain these tables while allowing normal SQL access against them. If storage costs and not immediate access is more important, then archiving to flat files can also provide additional options. The data can be retrieved back into temporary retrieval tables or even back to the source tables. Figure 5 shows how the data can be archived to a table archive and then even further archived to a file archive, or directly to a file archive, with the ability to retrieve the data from either.

Figure 5. Archival and Retrieval Options



The DB2 Data Archive Expert also allows us to maintain relationships among the data. You can define links between the data to ensure related data is archived together. This is supported both for DB2 enforced referential integrity or application enforced referential integrity (which can be defined with another the DB2 Grouper component with a concept called archive units).

The tools can be run with DB2 Version 8, and they support long names as well as long SQL statements.

Components of the DB2 Data Archive Expert Tool

The DB2 Data Archive Expert Tool is comprised of a few, easy to use, components to perform the archiving and retrieving of data. There is an ISPF interface to allow for the setup and execution of the archival and retrieval processes (known as specifications which are discussed later). There is the DB2 Grouper component, which was previously

mentioned, to allow us to link together tables that are related and need to be archived together. Another component is a callable API (Application Programming Interface) using DB2 Java stored procedures, which allows for previously defined specifications to be called from outside the ISPF interface. Finally, there is the metadata component. This allows for information to be stored about the specification definitions, mappings for the source data to the target archives, tracking of archive versions and other necessary information.

Developing a Method for Archiving

In order to use the tool in the best possible manner we must also have a well thought out strategy based on the application/user requirements. This must be done before the technical set up can take place. Since archiving is more an application driven process, it is a bit different than setting up a method for database backup and recovery, therefore you really do need user input and acceptance before the strategy is put into place. By user input and acceptance, I am referring to understanding how and when they will need to retrieve that data. We could agree that archiving the data is the easier part. How much we archive and when we archive is important, but the how and when the archived data will be retrieved is the key to satisfying application users.

Archive Strategy Considerations

When preparing an archive strategy there are a few things to consider in order to ensure success. First, tools aside, let's think about why our archiving strategies in the past have failed, or never happened at all. Too often the process of data arching is at best an afterthought. It never appears in the original specifications of the application requirements, or if it does it is sometimes considered optional. Because of this, it is not considered in the overall design of the database and the applications. We know from experience that any data removal and subsequent access needs have to be considered in the original design or they are near to impossible to implement after the application is in production or even near-production. When we try to implement these strategies after the database design is done and the application is written, we find ourselves looking at very generic database solutions (i.e. REORG UNLOAD or DISCARD), which leave us with an application that cannot easily use the offloaded data. The requirements of our applications today are dictating that we be able to access inactive/offloaded data in a timely manner.

We also need to be sure to have the key stakeholders involved. The end user needs to establish the requirements and then the database administrators, and most importantly the application programmers, need to figure how to best implement and archive strategy to meet those requirements. There is no one right solution for all archive strategies. It may vary be by table(data), time of year, volume, etc. These strategies may also change over time. Data may go from being inactive to active again, or the requirement for retrieval may change. This is another reason why a tool is such a needed part of a success long term strategy.

Data Retrieval

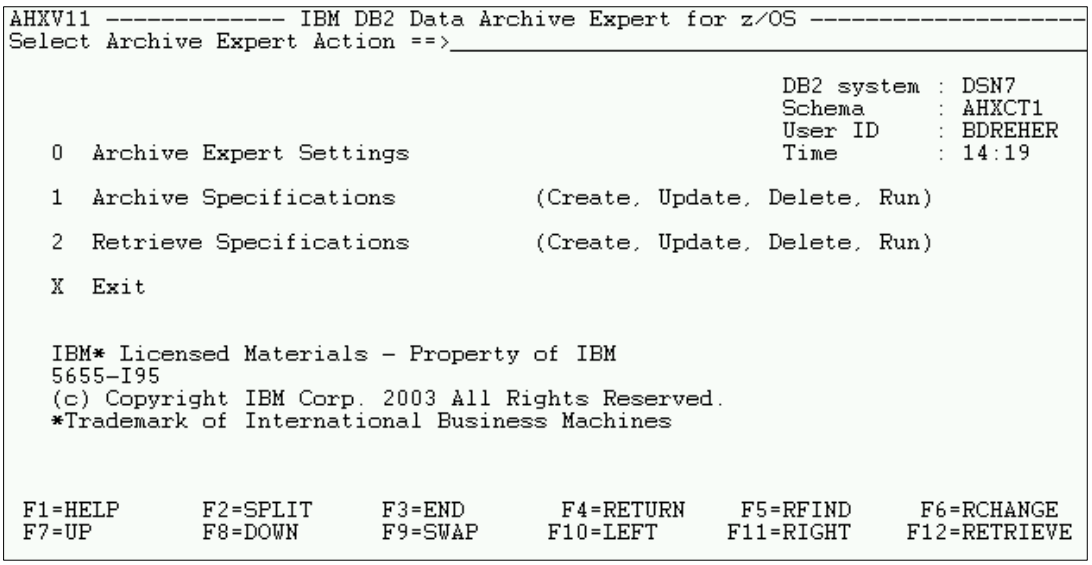
Ok, so now you've got the data archived. Hopefully we have given some relief to the database in terms lessening the amount of active data we have to store and we have given some relief to the application, possibly by not having to scan through inactive data. But now comes the more critical part – retrieval. If we did not want to retrieve it we would have simply deleted it or unloaded it with a REORG. But our applications have a requirement to be able to query the inactive data and in some cases (hopefully rare and should be determined during the archive design to minimize this) even make it active again. This is where a good archival strategy is needed so that retrieval can happen according to the needs of the application.

Ideally, we want to access our archive data in a similar fashion as we do our active data. We do have the ability to do this via SQL, which is preferred, if we are archiving to tables. If we are archiving to files then we have to use a LOAD utility or some other means (i.e. 4GL) to access the file archives. Again, this is something to consider during the development of the archive strategy. Normally we are only going to want to retrieve a subset of data from the archives, and this will also play into how our strategies are developed. We also need to consider if we are retrieving from a single archive, or multiple archives. Retrieving from multiple archives can of course get more complicated, as we could have more variance that needs to be planned for. We also need to plan where the retrieved data will be stored. Should it go to new tables, existing tables or become active again on the production database from which it was archived? Again, this will vary by table and by application needs for accessing the archived data.

Implementing an Archiving Method Using DB2 Data Archive Expert

Once a methodology has been established, the actual archiving and retrieval process using DB2 Data Archive Expert is done via ISPF panels by setup up specifications. Figure 6 shows this ISPF panel.

Figure 6 – ISPF Panel for DB2 Data Archive Expert



There will be two types of specifications defined: archive and retrieve. Archive specifications contain all of the information needed to move data from the source to the archive target. The retrieve specification will have all of the information needed to retrieve the data that was archived either to a table or a file. When archive specifications are run they create new versions of the archive data. Each version will be time stamped at the target.

Implementation Example

To put this into a business context, let's say we have an ACCOUNT table that has accounts that go inactive after a period of time, and we would like to archive them. These inactive accounts could be anywhere in the table. So we need to be able to perform row level archiving based on inactivity for accounts. Then when we archive these accounts we would also like to archive the corresponding demographic data about the account holder from the CUSTOMER table.

We will set up the archive specifications to copy the inactive account rows from the ACCOUNT and then delete them from the source tables. We can then use DB2 Grouper to find the related tables, in particular the CUSTOMER table, and then establish the connection column CUST_ID to link the rows from the two tables. Then we specify the criteria for archiving WHERE ACCT.LAST_ACTIVE_DATE < (CURRENT_DATE – YEAR). We will now archive the accounts that have not had any activity since this date and their associated data in the CUSTOMER table will also be archived, and optionally deleted from the original table or just copied.

Now there may be a situation where we want to access that data because we need a report of all ACCOUNT and CUSTOMER information for our accounts for the past year. This would mean we need to retrieve our archived data. So we set up the retrieval specification to retrieve the current data in the ACCOUNT and CUSTOMER tables and then combine (using UNION in VIEW technique shown below) that data with the data selected from

the archives to complete the request. This appears to be online accessible to the end user possibly without knowledge of what data existed in archives.

```
CREATE VIEW ALL_ACCOUNTS AS
        SELECT * FROM ACCOUNT A, CUSTOMER C
                WHERE A.CUST_ID = C.CUST_ID
                AND ACTIVITY_DATE >= (CURRENT_DATE – 1 YEAR)
UNION ALL
        SELECT * FROM ARC_ACCOUNT A, ARC_CUSTOMER C
                WHERE A.CUST_ID = C.CUST_ID
                AND ACTIVITY_DATE < (CURRENT_DATE –
 1 YEAR)
```

Summary

Let's summarize the two issues at hand: Test data generation and data archival and retrieval.

Test data generation has always presented a challenge for us because it is often a cumbersome task that never gets done. But we do know for proper testing it is essential and necessary. With a tool we can easily do this and make it part of our testing and verification process. The IBM DB2 Test Database Generator is a product that we can use to accomplish this task with ease.

With data archival, the issue is clear that working with large data stores comes with an interesting set of challenges. The problem has always been how to address them. We can try to address the issue upfront with database design options, but often even those techniques can fall short as the requirements continue to increase. We then look at writing our own methods for archival/retrieval but too often those solutions come only from a database perspective and the true application need is not met. As was mentioned in the introduction of this paper the process of data archiving/retrieving is truly more of an application task. With the IBM DB2 Data Archive Expert product, we can address these issues with a tool that helps us set up proper archiving strategies that can be easily maintained and deployed.

Best evaluation of both of these tools will come only with the involvement of the people who write the applications. Too often, we try to solve all problems at the DBA level and give only non-development related (i.e. code debugger) tools to those performing the task of database administration, overlooking the fact that many of our data requirements need to be dealt with in the applications. So with the, easy to use, IBM DB2 Test Database Generator product and the DB2 Data Archive Expert product, these two underappreciated arts - test data generation and data archiving, can be accomplished easily and effectively.

The purpose of this paper was to discuss the business need from an application perspective for tools to assist with test data generation and data archiving and retrieval.

For more details on each of the IBM products that help solve the problems faced by these two needs, please refer to the following documentation.

Resources

IBM DB2 Data Archive Expert

- IBM DB2 Data Archive Expert for z/OS: Put Your Data in Its Place – SC24-7080
- Take a Load Off: Archive Inactive Data – Bryan F. Smith, Thomas Vogel – DB2 Magazine – Q4 2003
- Data Archiving Techniques and Considerations for the DB2 Family – Rajesh Ramachandran – IDUG 2004
- Data Archive Expert – Bryan Smith – IBM Data Mgt Conference - 2004

IBM DB2 Test Database Generator

- IBM DB2 Test Database Generator for z/OS - SC18-7411-01
- Building a Test Database – Dave Schwartz – IBM Data Mgt Conference 2004
- http://www-306.ibm.com/software/data/db2imstools

Author Biography

Susan Lawson is an independent consultant for YL&A, Inc. She has been working in DB2 for 17 years and she specializes in working with large and complex database applications for a variety of clients world-wide. She has authored numerous books and articles on DB2 performance and related topics. She can be reached at susan_lawson@ylassoc.com or via www.ylassoc.com or www.db2expert.com.