Best Practices

Meridian Technologies
5210 Belfort Rd, Suite 400
Jacksonville, FL 32257
Meridiantechnologies.net

Contents

Overview

Meridian Technologies AcceleTest provides a facility to create and maintain test data from live production data. As a result, comprehensive and robust datasets can be created and maintained that represent the complex data found in production environments. This data can be created in order to replicate the naturally occurring combinations of data conditions found in raw production data, which would otherwise be difficult if not impossible to create from scratch.

When using production as your source for test data, provisions must be taken In order to prevent PHI and or PII from being present in the test data. This is facilitated by AcceleTest's many transformation methods that can be used to de-identify the raw data. The goal of this text is to describe the functionality of AcceleTest as well as provide some pest practices around the use of this tool.

AcceleTest also contains a facility that can compare datasets. This can be used to perform regression, unit and other kinds of tests or even verify that all PHI/PII have been cleansed from an environment. Meridian has developed a concept called "Data Testing Data"; we will describe this concept later in this document.

A Word on Data Profiling

Data profiling is essentially a way of obtaining element and dataset characteristics that provide a clear picture of what a dataset actually contains and how the elements relate to the other elements either in the same dataset or other datasets. In terms of test data creation data profiling is important in that it provides an understanding of the data that can help in the definition of how a data can be subset, or what pattern to follow when de-identifying or masking the data, or if a value generally falls within a certain range, or domain of values.

Before any subsetting or de-identification of data is undertaken, it is recommended that at least an element analysis is performed against the datasets being operated upon. This step generally does not take long and can save hours of time down the road, just by simply accurately describing the data.

Extract

The extract functionality allows for two activities to be performed against a dataset or collection of datasets; sub-setting, and transformation which includes de-identification. Subsetting provides the ability to reduce the size of a dataset while still preserving the complex and naturally occurring relationships present in the raw dataset. Transformation includes de-identification which is more focused around removing or altering information that can identify an individual and should be talked about separately. Transformations can also be defined to alter the format of a data field so that the output matches a business rule or expected format of a field. A couple of examples would be, removing punctuation from a phone number or combining last and first names to form a full name field.

AcceleTest separates the definition of an extract into eight activities:

- Data Source Definition
- Data Source Identification
- Field Mapping
- Field Linking
- Field Grouping
- Transformation
- Subsetting
- Execution

This section of the document will focus on Subsetting and the de-identification topics of the Transformation activity. For detailed instructions on all of the activities, please see the AcceleTest User Manual.

De-Identification

It is currently common practice in the IT industry to use production data in lower environments for use in software development, testing and training. The reasons for this are generally due to the complexity of production data and the difficulty in generating representative test data that meets all of the naturally occurring combinations of data conditions found in raw production data. Unfortunately, this practice dramatically increases the exposure an organization has to the possibility of private customer data being leaked outside of the organization either accidentally or deliberately. This increased risk of a data breach exposes the organization to legal, financial and reputational risk, including damage to the company's brand, fines, and even imprisonment.

PHI

According to the Health and Human Services Website, the article entitled, "Guidance Regarding Methods for De-identification of Protected Health Information in Accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule" (http://www.hhs.gov/), the HIPAA Privacy Rule provides protection for "individually identifiable health information" (IIHI) and calls this information "protected health information" or, more commonly, PHI. PHI is information, including demographic information, which relates to:

- the individual's past, present, or future physical or mental health or condition,
- the provision of health care to the individual, or
- the past, present, or future payment for the provision of health care to the individual, and that identifies the individual or for which there is a reasonable basis to believe can be used to identify the individual. Protected health information includes many common identifiers (e.g., name, address, birth date, Social Security Number) when they can be associated with the health information listed above.

- The Privacy Rule provides two methods by which health information can be designated as de-identified. The first is the "Expert Determination" method. This method provides data to be classified as not individually identifiable only if:

  "(1) A person with appropriate knowledge of and experience with generally accepted statistical and scientific principles and methods for rendering information not individually identifiable:

  (i) Applying such principles and methods, determines that the risk is very small that the information could be used, alone or in combination with other reasonably available information, by an anticipated recipient to identify an individual who is a subject of the information; and

  (ii) Documents the methods and results of the analysis that justify such determination"

This requires an organization to employ an individual that can meet the qualifications stated above and the individual needs to perform an analysis and certify each data set created. This creates large overhead in terms of cost and time, and since a "person with appropriate knowledge" is not exactly defined in the Privacy Act, we will focus on the second method, the "Safe Harbor" method.

The "Safe Harbor" method identifies 18 fields that need to be removed from a dataset in order to be deemed not individually identifiable. Below, we will discuss the way AcceleTest should be configured for each of these fields in order to meet these requirements.

Names
Names are easily dealt with in AcceleTest; the goal here is to either place nonsense data such as a "Random String" transform or to populate the name fields with randomly chosen names selected from a file or AcceleTest can randomly choose names from the source dataset, these are known as domain transforms. There are two types of domain transforms; "Domain from File" and "Domain from Input."

A "Random String" can be chosen if the downstream application or test does not require realistic looking data. Generally speaking, it is always preferable to provide realistic looking data for testing, so this method should only be used if an extract is needed in a very short timeframe.

The "Domain from File" and "Domain from Input" transforms essentially do the same thing; the only difference is the source from where the domain of values is read. There are positives and negatives to both methods and these should be taken into consideration when choosing between them. The "Domain from Input" transform takes less time to set up, however, it will perform a table scan of the input table specified in order to create the domain. If this table is very large, over a million records or so, this selection will add significant time to the extract process, so the "Domain from File" transform should be used in that case. Regardless of which domain transformation is chosen, the "Replacement record cannot match input" should always be selected.

The "Domain from File" transform takes a little more time to set up, however, it will perform significantly better when the extract is executed. To set up the "Domain from File" transform, a file that contains names needs to be generated and loaded into AcceleTest via the GUI. To generate the file a simple SQL statement can be run against any data set that contains names such as:

```
SELECT DISTINCT TABLE.MBR_FIRST_NAME
FROM TABLE;
```

The output should be put into a text file that contains a header that has the exact field name as the field that will be receiving the names. This file should be large enough to provide a good size domain of names 1000- 2000 entries should be sufficient. An example file intended to load a field called MBR_FIRST_NAME will look like the following:

```
MBR_FIRST_NAME
Mary
John
Joe
Jane
…
Frank
Sue
```

The long term solution is to build a names database that is specified as an input to the extract. This database contains a good representative sample of first, last and middle names along with gender classified name prefixes and suffixes that enables the definition of the relationship to gender, prefix and title. This will ensure that the name fields populated in a row makes sense; for example, Mrs. Karen Smith, Dr. Susan Jacobs, MD or Mr. Bob Brown III. Meridian has a name database available; please contact your Meridian representative for more information.

An alternative to building an address database would be to use the "Multi-Column Domain" transformations. This is a set of two transformations one being the "Base" transform and the other being the "Extension". The base transform defines the driving characteristic such as gender and the extension transforms will select random domain values based on the input value in the base transform. This is accomplished by creating a comma delimited text file that contains Cartesian products of the valid combinations of values. An example file looks like this:

```
Name,Gender,Prefix,Suffix
Katherine,F,Mrs,PhD
Katherine,F,Mrs,JD
Katherine,F,Mrs,DO
Katherine,F,Mrs,MD
Katherine,F,Dr,PhD
```

Katherine,F,Dr,JD
Katherine,F,Dr,DO
…
Kaden,M,Prof,PhD
Kaden,M,Mr,Jr
Kaden,M,Mr,Sr
Kaden,M,Mr,JD
Kaden,M,Mr,III
Kaden,M,Mr,MD
Kaden,M,Mr,DO
Kaden,M,Mr,PhD

In this example, the MBR_GENDER field would use the "Multi-Column Domain (Base)" transform with the Gender column selected and the remaining fields of MBR_FIRST_NAME, MBR_PREFIX, MBR_SUFFIX would use the "Multi-Column Domain (Extension)" transformation with the Name, Prefix, Suffix columns selected respectively.

Geographic subdivisions smaller than a state
This field pertains to any reference to a geographic area. Examples of the fields that this refers to are:

- Street Address
- City
- County
- Zip

As with names, the option to replace the data with a "Random String" transformation is always an option, but will produce data that is not representative of its original form, so there are better transforms available. Also, as with names, we can address each field individually using "Domain from File" and "Domain from Input" in the same way and with the same caveats and justifications. This will result in a "not individually identifiable" result; however the relationship between city, state and zip will be lost. This may be fine if the expected consumer of this dataset does not need to perform state to zip verification, but more robust options are available.

One such option is to use the "Multi-Column Domain" pair of transformations. In the case of addresses, a file that contains relationally correct values for city, county, state and zip may be used as an input. In this case though, the "Base" column would be zip and the "Extension" columns would be city, state and county. Since street address does not have a relationship to a zip code, the street address column may be dealt with using a "Domain from File" or "Domain from Input" transformation.

The second option is to build an address database that would be specified as one of the inputs to the extract. Like the names database, this database would contain a representative sample of street addresses, but will also contain correct listings of cities, states and counties classified by the correct zip code. Meridian has an address database available; please contact your Meridian representative for more information.

Dates

Dates that qualify as PHI according to the Privacy rule are:

- Birth Date
- Admission Date
- Discharge Date
- Death Date

AcceleTest has two types of transforms that may be applied to dates, they are: “Date Variance” and “Random Date”. Either method is suitable for the de-identification of PHI dates as long as the “Never copy input date” option is selected. The main decision here is whether the data needs to be representative of the volume for a specific date range, or if the distribution of age needs to be preserved in the dataset. If either of these are a requirement, then it is a matter of determining the level of variance that is tolerable and setting up the transform accordingly.

Telephone Numbers/ Fax Numbers

Unformatted telephone/fax numbers are relatively straight forward and very easy to de-identify by using a “Random String with Keep Mask” transform. If the phone number is a 10 digit domestic number that does not contain punctuation, the Mask String should be 9999999999, for a standard 10 digit US phone number. If the phone number contains punctuation, the Mask String should look like *999*+999*9999. This will preserve the parentheses, spaces and dash for a standard US formatted phone number if the input is formatted in a standard way. The exact pattern to use for the Mask String will need to be determined by analyzing the output of the data profile of this field. If the format of this number varies in the input field then the Mask String 9999999999 can be used to remove the varying punctuation from the field.

Vehicle Identification Numbers

Vehicle identification numbers can be de-identified as a Random String unless the format of the number is critical to a business process or test case. If this is true the data profile of this field should be studied and the pattern of the field should be replicated using the “Random String with Keep Mask” transform.

Device Identifiers

Device identifiers can be de-identified as a Random String unless the format of the number is critical to a business process or test case. If this is true the data profile of this field should be studied and the pattern of the field should be replicated using the “Random String with Keep Mask” transform.

Email Addresses

Email addresses can be dealt with using a few different approaches. These approaches vary in the complexity required to fit a test case. One very simple approach is to just substitute a default email address such as “person@domain.com” into each record. This can be sufficient if there are no test cases that require the email to be tied back to another record such as a customer entry. To do this a “Random String with Overlay Mask” transform would be used with the default address entered into the Mask String. If a reasonably unique email is needed for the verification of a join rule a simple script may be used that will place the de-identified first name at a default domain address. For example, the script: return transformed_MEMBER_FIRST_NAME.."@testdomain.com" would populate the email field with <first_name>@testdomain.com where <first name> would be the de-identified field value of MEMBER_FIRST_NAME.

Web URLs

Web URLs, if encountered, are more likely informational than a requirement for a business rule, so these fields should either be populated with a “Random String” or if the format needs to be preserved replaced with a default value with a “Random String with Overlay Mask” transform with the default address entered into the Mask String.

Social Security Numbers

AcceleTest provides a transform that randomly generates an SSN value conformant to the US Social Security administration rules but within a range known not to be currently assigned to an existing individual. This transform should always be used for Social Security Numbers.

IP Addresses

IP addresses, if encountered, are more likely informational than a requirement for a business rule, so these fields should either be populated with a “Random String” or if the format needs to be preserved replaced with a default value with a “Random String with Overlay Mask” transform with the default address entered into the Mask String.

Medical Record Numbers/Health Plan Beneficiary Numbers/Account Numbers

These identification fields are core to any healthcare based organization and will most likely be a part of a natural key and therefore need to be placed in a Field Group so that AcceleTest will de-identify them the same way for each instance of the data in a dataset. For more on Field Groups, please see the AcceleTest User Manual. Also, the format of these identifiers may contain an alphanumeric pattern that needs to be preserved, so a data profile of this field should be studied and the pattern of the field should be replicated using the “Random String with Keep Mask” transform.

Biometric Identifiers

This type of field refers to binary data that describes personal biological identifiers such as a fingerprint or voice print. This type of data does not lend itself to de-identification, so any data that fits this category should be dropped from the dataset.

Full Face Photographs
This type of data does not lend itself to de-identification, so any data that fits this category should be dropped from the dataset.

Any Other Unique Identifying Number/ Certificate/License Numbers
The format of these identifiers may contain an alphanumeric pattern that needs to be preserved, so a data profile of this field should be studied and the pattern of the field should be replicated using the "Random String with Keep Mask" transform.

Subsets

Testing applications is a very important component of the software development lifecycle and in order to test an application thoroughly, a dataset needs to be used that represents the data found on the production environment and exercises all of the test cases that have been developed for the application. The most robust data in any organization is always found in production and as such, production is a great place to start when creating a test environment. One problem with using production data is its size; especially for large transactional systems.

AcceleTest provides a facility that enables the iterative creation of test data. When configuring the subsetting activities of the extract process, the Test Cases that have been defined should be used to drive the subsetting effort. For example, each test case should be defined as a separate subset on the Subsetting tab in the extract definition. By using the test cases to build subsets the user can be assured that the data needed for each case will be present during the test run. Also by using this approach, any new test cases can be added to the existing extract definition without affecting any other test case. If there is any overlap of records between the subsets, only one copy of that record will be present in the output.

Compare

The Compare Utility provides a generalized mechanism for comparing data in two different locations. This capability allows for the quick and easy comparison of individual elements all the way up to full schema comparisons. This functionality is core to Meridian's concept of "Data Testing Data" and is can be used accelerate very complex tasks like environment remediation or determining if PHI has inadvertently been introduced into a lower environment.

Data Testing Data

The reason why Data Testing Data is valuable to an organization can be explained in the following scenario: When performing a regression test on an ETL application, typically a test dataset is run through the application and the result set is compared to the output of some set of SQL queries that mimic the business rules present in the application. When there is a discrepancy between the two, the test fails for that specific business rule. The problem with this approach is that it is not known whether or not the issue lies within the application, or if the issue lies within the SQL that was developed to test the application. As a result, reverification of the application code and testing SQL has to take place. This is a very expensive approach because not only does the application code need to be developed, but the testing code needs to be developed, duplicating the development effort. Then, after the test process, one or both sets of code needs to be re-evaluated and/or changed before a failed test may be resolved. Using AcceleTest's Compare Utility, along with the concept of Data Testing Data, eliminates the duplication of code and accelerates the testing process by providing a very simple report that informs the tester which rows of data have been added, deleted, are the same, or have changed as compared to a baseline run. Below we will discuss three use cases that explain how to use AcceleTest's Compare Utility for accelerating the testing effort.

Existing Application

The most simple use case is the testing of an existing application. We will use the following diagram to describe the activities in this scenario.

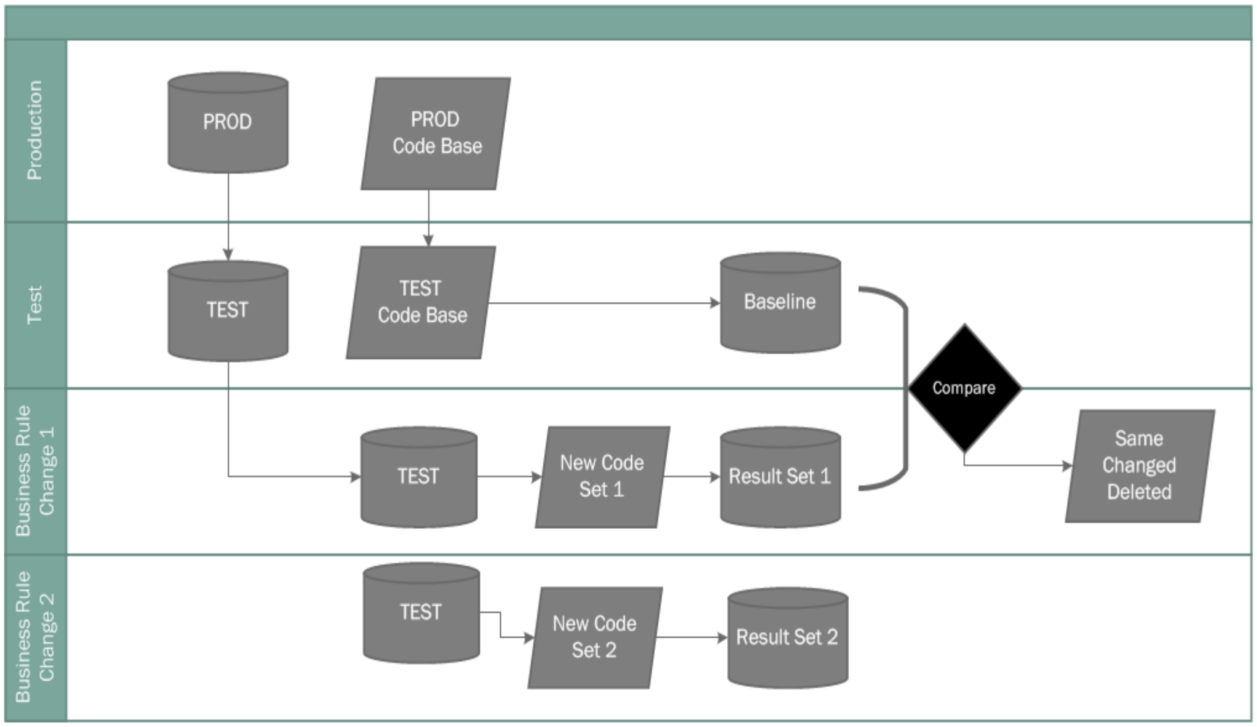

Figure 1 - Data Testing Data - Existing Application

Since this is an existing application that is currently running in production, the certified code that is in production will be used to create the baseline run for the testing of any modification of that code going forward. First a test data subset will be created and placed in to the test environment, using AcceleTest. Then, the current production code base will be copied into the same test environment and executed against the test dataset. Since this code base is certified good, the output of this run can be considered a certified baseline for future tests.

When the code for this application is changed, the new code will be copied into the test environment and run against the same test dataset as the baseline run. The resulting output is then compared to the baseline output of the previous run via the Compare Utility and the resulting comparison report should be able to be explained by all of the code changes that were programmed in this development cycle, and any items that cannot be explained are a result of a coding error that can be identified and remediated very quickly. Once any offending code is fixed, the new code is then copied to the test environment and run against the same set to test data; this is done until all of the output of the compare is correct.

Once a set of code is run successfully in the test environment and is certified to be migrated to production, the last result set in test becomes the new baseline for any future enhancement and the above process is repeated.

New Business Rule or New Data

Occasionally, a new business rule is created that requires data that does not exist in the current test dataset or a new source of data needs to be incorporated into an application. This scenario is the same as the previous one, with one minor modification.

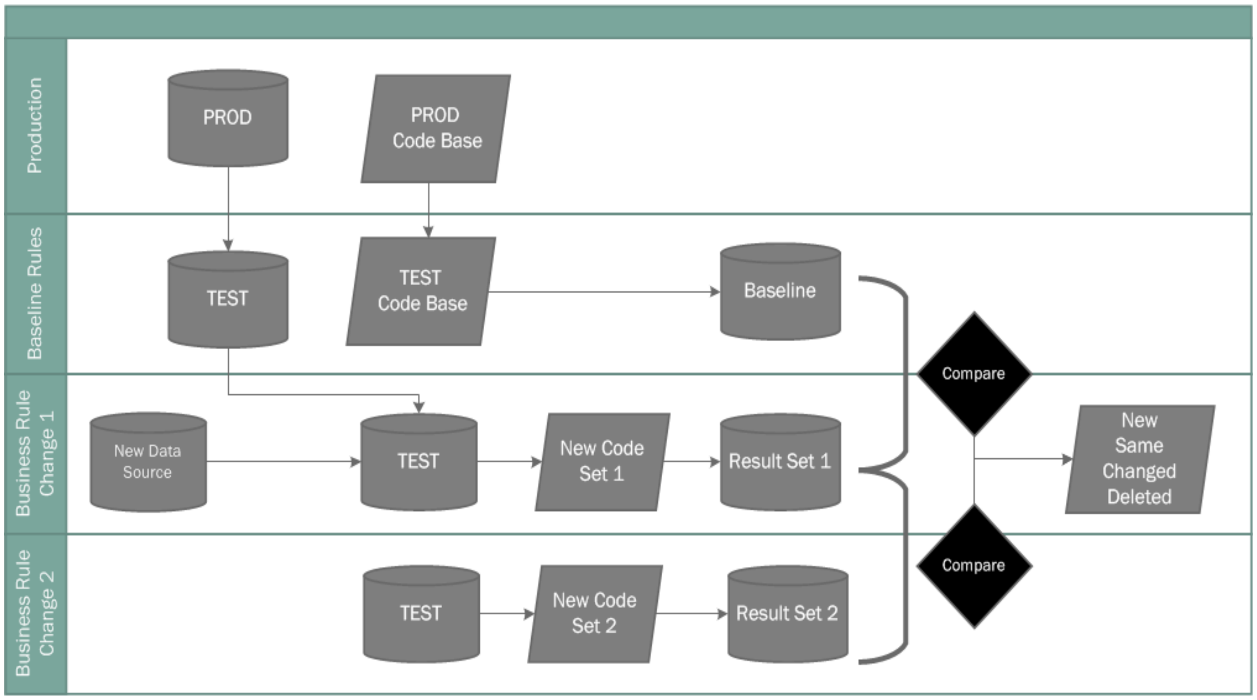

Figure 2 - Data Testing Data - New Business Rule and/or Data Source

In this case we will start with the existing application scenario explored in the previous section and we will assume that the baseline result set has been created. Since a new subset of data needs to be brought into the application, the AcceleTest Subset will need to be modified in order to bring in this new data. This subset will then be run to repopulate the test environment.

The new code will be copied into the test environment and run against this newly created test dataset. From this point on, the process is the same; the resulting output is then compared to the baseline output of the previous run via the Compare Utility and the resulting comparison report should be able to be explained by all of the code changes that were programmed in this development cycle, and any items that cannot be explained are a result of a coding error that can be identified and remediated very quickly. Once any offending code is fixed, the new code is then copied to the test environment and run against the same set to test data; this is done until all of the output of the compare is correct.

And once again, when the code is run successfully in the test environment and is certified to be migrated to production, the last result set in the test environment becomes the new baseline for any future enhancement and the above process is repeated.

New Application
For a brand new application, the end of the process is the same as the previous two cases; the difference is how we create the baseline.

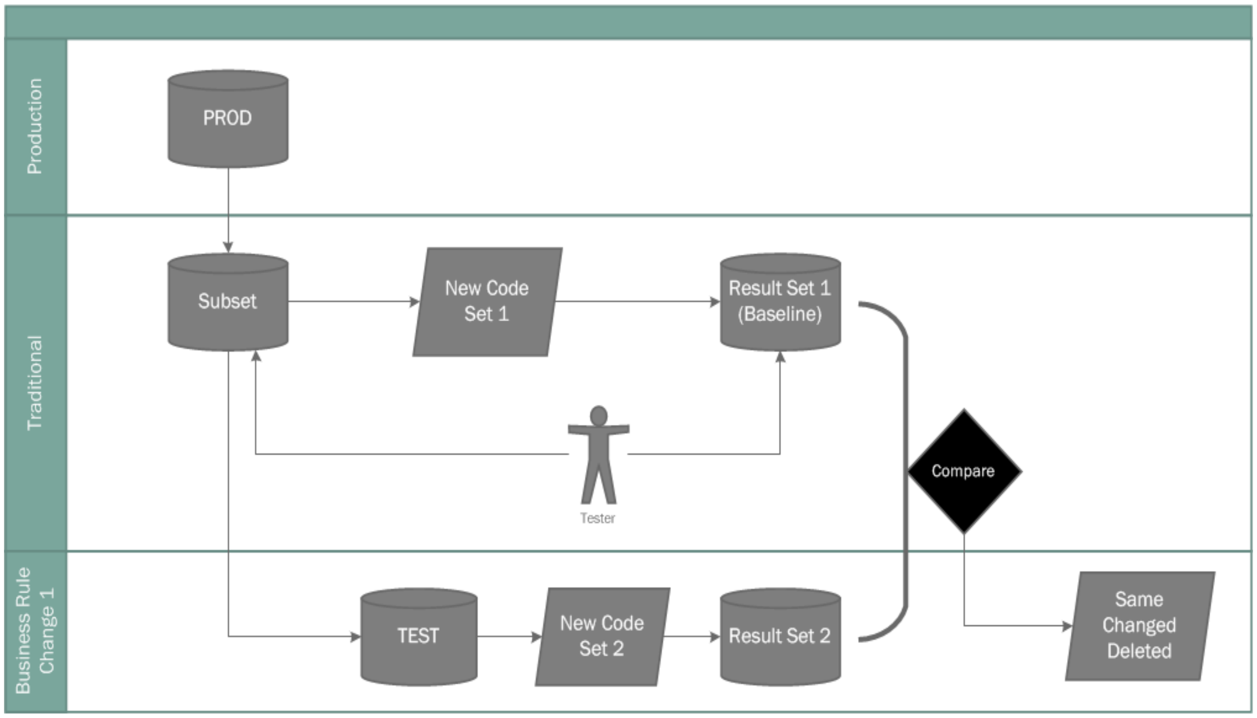

Figure 3 - Data Testing Data - New Application

In order to test a new application, we first need to create a test data set for the application to be run against. AcceleTest is used to create a new subset of production data using the test cases and/or business with which the application is to be in compliance. The new code application is then copied to the test environment and run against this new test set creating the first result set. Since there are no previous runs of the application to certify against, this first result set needs to be certified using traditional testing methods, however once this first set of code is certified the previous two scenarios may be used going forward to ensure quick and economical test cycles going forward.

Lower Environment Audit

Another use of the AcceleTest Compare Utility is to identify instances of PHI in the lower environments that need to be de-identified. Also, once the de-identification of the lower environments is established, the Compare Utility may be used in an automated process that may be run at an appropriate frequency, to ensure that no PHI is inadvertently migrated to a lower environment without being detected in a timely manner.

This is accomplished by first identifying all PHI in a given dataset, and then setting up AcceleTest to compare the PHI fields in the production dataset against the PHI fields in the test and development subsets. In this case, since only PHI fields have been identified in the compare, every field in every row should be marked as changed. Any deviation from this result means that PHI has made it to the lower environment and needs to be cleansed. If the above process is repeated for all production datasets and their corresponding test subsets, PHI should never exist undetected in a lower environment for more than the duration of time between the scheduled automated runs.