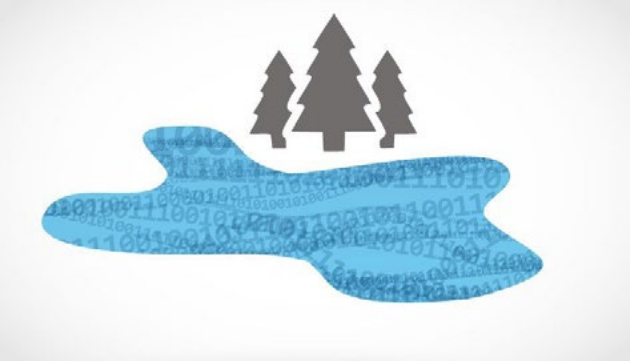
CASE STUDY: TESTING DATA LAKE APPLICATIONS IN FINANCIAL SERVICES

A multi-national banking and financial services corporation required comprehensive test data automation for testing its data lake applications. The company offers financial products for retail banking, direct banking, commercial banking, investment banking, wholesale banking, private banking, asset management, and insurance services. They operate in more than 40 countries and rank as one the world's largest banks.



Data lakes are repositories for large amounts of data collected from multiple sources in a raw and native format. They eliminate information silos by combining data from diverse sources such as electronic banking systems, IoT devices, social-media sites, and internal-collaboration systems. Data may be stored in structured, semi-structured or unstructured formats.

Increasingly, banks are using data lakes to turn big data into actionable business intelligence to drive profitable business outcomes. Electronic data is growing at a phenomenal rate due to the rise in online banking and digital transformation of the customer experience. The challenge and opportunity presented by data lakes is how to accelerate the cycle time from raw data inputs to actionable business intelligence through the use of data mining and analysis software. The mandate for QA organizations is to ensure that quality data is used for business intelligence purposes by performing comprehensive testing of its data lake applications.

THE TEST DATA CHALLENGE

The Bank issued a Request For Information (RFI) to evaluate a test data solution that would enable multiple teams to perform a complete range of testing operations in a highly efficient and scalable manner. They focused on synthetic test data generation because of the ability to produce highly controlled data variations in multiple data formats and its inherent data security. They were looking for a solution that would meet their needs for automated unit testing, exhaustive functional testing and performance testing procedures.

GenRocket responded to the RFI with a complete *Test Data Automation* solution and participated in a rigorous Proof of Concept (POC). The combined RFI/POC process included several test data challenges. They were incorporated into four use cases that reflect their application testing requirements. They also wanted to evaluate the management and scalability of the system. The POC requirements are briefly outlined below.

System Setup and User Account Management

The Bank required a platform to provide control over access to the system and its resources. They also required reporting on various aspects of system operations.

- Install software and create admin user
- Create projects to perform various test cases
- Create a user account with read-only privileges
- Demonstrate effectiveness of security controls
- Demonstrate configuration process and audit trails
- Demonstrate deployment scalability and change management
- Demonstrate reporting and supportability options

Use Case 1: Generation of Insert Statements Based on DDL and DML

This use case evaluates an ability to generate synthetic test data based on a supplied data model (DDL) and using Data Manipulation Language (DML) to test database interactions. The data structures must maintain referential integrity. Data must be formatted with variable length values based on pre-defined rules and separated by a special delimiter. Here is a small sample of their data requirements.

- Create test data tables preserving primary/foreign key relationships
- Create random data using multiple value ranges as specified
- Create data within a value range and concatenated with unique ID
- Generate unique data within a pre-defined value range
- Generate data within a value range and apply HASH8 on column values
- Perform a data insert and lookup function to a SQL database

Use Case 2: Generation of Test File Based on Metadata

Generate test data by importing a CSV file containing metadata that specifies the format the test data must follow. Create a patterned file that includes a pre-defined label and a date stamp in the format: YYYYMMDDHHMMSS.

- Generate 1-character string where one code is 95% of values with other codes random
- Generate 10-digit number with foreign key relationship to a table column in Use Case 1
- Demonstrate creation of insert statements based on the relationship of key fields

Use Case 3: Generation of Test File Based on Metadata with Business Rules

Generate test data by importing a CSV file containing metadata that specifies the format the test data must follow and apply a number of business rules such as:

- Generate test data based on pre-defined allowed values
- Generate patterned test data to vary possible data ranges
- Generate variable length integers that are empty 50% of the time
- Generate 10-digit numbers with a foreign key relationship to column in separate table
- Generate account numbers with defined character positions and check digit remainder

Use Case 4: Generate 3 Files with Creation Time Stamp at Different Volumes

This use case is designed to show how the number of records for a test case can be controlled. Generate 3 test files and demonstrate control over record count, file size, and elapsed time.

- File 1 = 1,000 records
- File 2 = 100,000 records
- File 3= 1,000,000 records
- Concatenate a fixed file identifier with formatted time stamp

THE GENROCKET SOLUTION

GenRocket worked closely with one of its global IT services partners to jointly conduct the POC with the Bank. The process showed how GenRocket can combine speed of provisioning with full control over data quality in a way no other test data solution can match. This successful POC resulted in the selection of GenRocket as the best solution for the Bank's Data Lake testing requirements. Several GenRocket capabilities combined to make this evaluation a success.

Modular Architecture

GenRocket's **component based architecture** provides the flexibility to design any variation or volume of test data with assured referential integrity. Powerful data generators and receivers allow the Bank's QA team to conduct exhaustive testing with extensive control over data combinations, patterns and permutations. The ability to query external data sources allowed testers to combine real-world production data with controlled synthetic data. The use of synthetic data provided total security and compliance with privacy laws.

Model-Based Test Data

Because GenRocket's test data is based on the customer's data model, any database schema, DDL file, or a metadata contained in a CSV file can be used to structure test data that accurately reproduces the original database or file format. Data models can be imported and immediately used to define test data scenarios for generating real-time test data on-demand.

Self Service Provisioning

The Bank POC called for extensive use of ***GSelf-Service***, GenRocket's self-service provisioning capabilities. *GSelf-Service* includes *Test Data Rules*, a feature that allows the configuration of test data based on business rules and workflow logic. *Test Data Queries* is a feature used to query production data values from databases or files and dynamically blend that data with synthetically generated data. *Test Data Cases* makes it easy to quickly and easily modify *Test Data Scenarios* to meet the requirements of any given test case. *Test Data Cases* can be defined for unit, functional, integration, API, performance, regression and other types of tests.

Enterprise Scalability and User Management

In addition to its technology capabilities, GenRocket also provided the scalability and management features that were required by this large multi-national banking operation. The *Test Data Scenarios* configured for specific test procedures can be managed and versioned within the context of a GenRocket *Project*. *Team Permissions* ensure users are only able to access the *Projects* appropriate for them and controls access to information such as the *Domains* or *Scenarios* for those Projects. The *Organization Admin* status is the highest level of authority and is required to manage users, team permissions and access to GenRocket system resources.

Test Automation Integration

GenRocket has numerous ways to integrate into test automation frameworks including, two ***GenRocket Real-Time Engines***, the *GenRocket Socket Engine*, the *GenRocket REST Engine* and *the* ***GMUS (GenRocket Multi User Server)*** which allows many users to generate data via a central client application. These engines allow external applications to launch test data scenarios in real-time during test operations. They also allow the platform to be seamlessly integrated with the Bank's CI/CD pipeline in addition to its many test automation tools. The result is Test Data Automation that is fully integrated with the Bank's test automation environment.