Netezza to Actian Avalanche Cloud Data Warehouse Migration Guide

Version 1.0

July 30, 2019

Contents

Introduction

The TwinFin and Striper models (N1001 and N2001) of IBM Netezza PureData have officially reached end of support. As you consider your migration plans for your Netezza implementation, make sure that your exit strategy dovetails into your strategy for the future.

Cloud data warehouses have emerged to the fore for insight-driven businesses who want to take advantage of new data types to engage with audiences in real time. Their elastic scale and cloud economics make them well-suited for the digital landscape.

Actian Avalanche is a third generation, fully-managed cloud data warehouse, designed from the ground up to deliver and maintain high performance as you rapidly increase concurrent use with very large data sets. Its hybrid cloud architecture minimizes the risk of migrating from Netezza by enabling you to move to the cloud—AWS, Azure, your own private cloud—at your own pace. You can modernize your data warehouse by incorporating Hadoop, streaming, mobile and IoT data and running AI and ML algorithms at scale.

This guide provides items to consider when migrating your existing Netezza environment to the Avalanche service.

Planning Your Migration

Thoughtful planning is an essential first step to executing a successful migration project. Migrating a large data warehouse in a "lift and shift" approach is usually unrealistic and fraught with risk. Rather, taking the iterative approach allows your on-premises data warehouse to remain operational as the cloud data warehouse is gradually deployed.

Incremental migration is done in steps: from design, implementation, testing/verification, to maintenance. While maintaining both environments, data must be synchronized between the on-premise and cloud instances.

Iterative Plan

Create Your Migration Team

You will need to identify the people in your organization that will participate in the migration effort. Some of the key roles you will need include:

- Tables Program Manager
- Migration Architect
- Application Architect
- Data Architect
- Quality Assurance Manager
- Information Security Officer (infosec)

One person may act in one or more of the roles mentioned above, but someone should be assigned to each of the roles.

Develop Your Statement of Work

Here is a high-level list of topics to consider in planning your SOW.

- Applications: Identify the applications that will migrate to the Avalanche environment. Determine how these applications interact with their respective databases. Most "Analytic" applications are primarily read-only (although they may persist temporary or intermediate data). "Operational" applications (e.g. call center apps, manufacturing control, or delivery logistics) have a mixture of analytical operations coupled with record-keeping operations. Identify any application processes that require transactional semantics.
- Databases: Identify the database instance(s) and database(s) that support these applications. These databases will be the focal point of the migration to Avalanche. For any given data set in the database, are there any special security requirements (e.g. restricted access, need for encryption, data obfuscation, etc.)? If you are a multi-national organization, are there any special regulatory consideration for data that

might leave the region?

- Database Objects: Identify the database objects that will be migrated (e.g. tables, views, stored-procedures, etc.).
- Data Sources: Identify the data source(s) that feed these databases. You should consider the geographical location of these databases and the network bandwidth available between these systems and Avalanche.
- Data Integration: Identify the data integration tools that are currently in use. Do you perform ETL, ELT, special data cleansing processes, etc.? How often is this data acquired and loaded into the database? Are there any SLAs regarding when a given data load job begins, and needs to complete?
- Users: Identify the users who will interact with the applications being migrated. In many cases, an end user interacts with the database via a "system-of-engagement" (e.g. internal systems such as finance, marketing, CRM, etc., as well as BI tools such as QLIK, Looker, Tableau, Cognos, etc.). How will these applications access Avalanche? That is to say, what user-id(s) will the applications use? Will the applications use session-pooling? Will certain applications have restricted access to certain data items in Avalanche? Also, you need to identify those human-users that will access Avalanche directly.
- Security: Identify user-id(s) and their associated roles and permissions. Determine which methods will be used for authentication, authorization, accounting, and logging.
- Network: How will your applications and end users connect to the Avalanche service? Based on your organization's internal security policies, changes may be required to firewall settings, port access, etc. in order to facilitate access to cloud-based services.
- Operational Procedures: Operational procedures can sometimes be hard to quantify, but this is precisely why you need to examine them. Will any Operational Procedures need to be modified in order to execute in a cloud-based environment? To ensure that your migration completes with minimum impact to the enterprise, walk through these procedures with an eye for adapting to a cloud-based environment.
- Operational Environments: Most organizations have a Development Environment, Test/Certification Environment, and a Production Environment. While these environments will be very similar in nature, they will each have a few unique characteristics. Thus, there should be a separate migration plan for each environment.

Migrating your Netezza System

Set up Cloud Storage

On AWS, create an S3 bucket. It is important to locate your S3 bucket in the same AWS region as your Avalanche service. S3 is used to stage your table data before it is loaded from S3 into Avalanche.

On Azure, create an Azure Data Lake Storage (ADLS) Gen2 storage account with a hierarchical namespace which provides a native directory-based file system. ADLS is used to stage your table data in Azure before it is loaded into Avalanche.

Migrating Databases

Netezza namespace follows the following format to refer to objects in a database: *DataBaseName.SchemaName.ObjectName*

In a single-schema Netezza environment, you can also use the "dot dot" notation such as *DataBaseName..ObjectName*, and the database will default to the only schema of the database.

You cannot use CREATE, ALTER, or DROP commands to change objects in a database that is not your current connected database.

You cannot use a cross-database INSERT, UPDATE, or DELETE statements to change a table that is not your current connected database.

You can, however, perform cross-database joins to tables that is not your current connected database.

Netezza also has the concept of an object owner. A user *user1* can own a table *table1* in schema *schema1*. Referencing a table does not refer to the owner name, but uses the schema name.

Avalanche uses the namespace *SchemaName.ObjectName.*

In Avalanche, users and schemas are essentially the same thing. You can consider that a user is the account you use to connect to a database, and a schema is the set of objects (tables, views, etc.) that belong to that account.

Databases from Netezza maps to schemas in Avalanche. Typically, production schemas will belong to secure user accounts where access is controlled.

Migrating Users and Groups

Before you can create any tables or views, you need to create the schemas which will own the databases defined in Netezza.

In Avalanche, groups and roles can simplify control of database access. Groups are used to apply permissions to a list of users, while roles are used to associate subject privileges and permissions with an application.

To create a group, use the SQL statement: CREATE GROUP inside_sales ;

Users can be added to a group by specifying:

```
ALTER GROUP inside_sales ADD USERS (dannyh, helent) ;
```

Users can be assigned a default group:

```
CREATE USER bspring WITH PASSWORD='secret', GROUP = engineering ;
```

or

```
ALTER USER bspring WITH GROUP = engineering ;
```

A user can be a member of more than one group. After a group is created, you can associate permissions with it. When you grant permission to a group, you are, in effect, granting that same permission to each user in the group.

Migrating Tables

Data Types

Avalanche Data Types:

Class	Data Type	Description
Character	CHAR	CHAR(size) Fixed-length character string having maximum length *size* bytes.
	VARCHAR	VARCHAR(size) Variable-length character string having maximum length *size* bytes.
	NCHAR	NCHAR(size) Fixed length Unicode string having maximum length *size* characters.
	NVARCHAR	NVARCHAR(size) Variable-length Unicode string having maximum length *size* characters.
Numeric	INTEGER1	INTEGER1 TINYINT Exact number that requires one byte.
	INTEGER2	INTEGER2 SMALLINT Exact number that requires two bytes.
	INTEGER4	INTEGER4 INTEGER Exact number that requires four bytes.
	INTEGER8	INTEGER8 BIGINT Exact number that requires eight bytes.
	DECIMAL	DECIMAL(p,s) Exact numeric type with precision *p* and scale *s*.
	FLOAT	FLOAT[(n)] FLOAT8[(n)] Approximate number that requires eight bytes with optional minimum required binary precision of *n* (24 to 53).
	FLOAT4	FLOAT4[(n)] Approximate number that requires four bytes with optional minimum required binary precision of *n* (0 to 23).

Class	Data Type	Description
Date and time	ANSIDATE	ANSIDATE Year, month, day in the format yyyy-mm-dd.
	TIME WITHOUT TIME ZONE	TIME[(n)] Hour, minutes, seconds with *n* digits of precision in the fractions of seconds.
	TIME WITH TIME ZONE	TIME[(n)] WITH TIME ZONE Hour, minutes, seconds with *n* digits of precision in the fractions of seconds, at the specified time zone.
	TIME WITH LOCAL TIME ZONE	TIME[(n)] WITH LOCAL TIME ZONE Hour, minutes, seconds with *n* digits of precision in the fractions of seconds, in local time zone.
	TIMESTAMP WITHOUT TIME ZONE	TIMESTAMP[(n)] Year, month, day, time with *n* digits of precision in the fractions of seconds.
	TIMESTAMP WITH TIME ZONE	TIMESTAMP[(n)] WITH TIME ZONE Hour, minutes, seconds with *n* digits of precision in the fractions of seconds, at the specified time zone.
	TIMESTAMP WITH LOCAL TIMEZONE	TIMESTAMP[(n)] WITH LOCAL TIME ZONE Hour, minutes, seconds with *n* digits of precision in the fractions of seconds, in local time zone.
	INTERVAL YEAR TO MONTH	INTERVAL YEAR TO MONTH An interval of years and months.
	INTERVAL DAY TO SECOND	INTERVAL DAY TO SECOND[(n)] An interval of days and seconds with *n* digits of precision in the fractions of seconds.
Abstract	MONEY	MONEY Monetary amount in the range -999,999,999,999.99 to 999,999,999,999.99
	IPV4	IPV4 An IPv4 host address. 4-byte, network address in dotted-decimal notation (four decimal numbers, each ranging from 0 to 255, separated by dots). For example: 172.16.254.1
	IPV6	IPV6 An IPv6 host address. 16-byte, network address in eight groups of four hexadecimal digits separated by colons. For example: 2001:0db8:85a3:0042:1000:8a2e:0370:7334
	UUID	UUID A unique 128-bit integer.
Boolean	BOOLEAN	BOOLEAN Literal values TRUE or FALSE, strings 'TRUE' or 'FALSE'

All Netezza data types are accepted according to ANSI standards with the exception of translating BYTEINT to TINYINT.

Avalanche stores all character data in UTF8 so the NVARCHAR data type is equivalent to the VARCHAR data type.

Varchar strings are variable-length UTF8 strings. Varchar strings can contain any character, including non-printing characters, except the ASCII null character ('\0').

DOUBLE PRECISION is a valid data type in Avalanche that is equivalent to FLOAT8.

Data Distribution

Intelligent data distribution is one of the key factors that contributes to the performance of Netezza.

Netezza system follows two distribution methods, hash and random.

Avalanche tables are either partitioned or not. Unpartitioned tables are read into cache by each node and are sometimes referred to as “replicated tables”. Unpartitioned tables are generally the smaller tables by number of rows in the schema (for example, dimension tables). Partitioned tables (sometimes referred to as “sharded” tables) are managed as multiple “chunks” in which a given chunk (partition) resides on a given node. Medium-to-large tables (for example, large dimension and fact tables) are more efficiently processed as partitioned tables because each node needs to only deal with the partitions of the table that are stored on that node.

Following is the syntax for partitioning:

```
CREATE TABLE employee (
  emp_no INTEGER NOT NULL NOT DEFAULT,
  emp_name CHAR(32) NOT NULL NOT DEFAULT,
  dept_no INTEGER,
  emp_rating INTEGER)
WITH
PARTITION = (HASH ON emp_no DEFAULT PARTITIONS);
```

The DEFAULT keyword denotes the pre-configured number of partitions by cluster based on Avalanche Units size for high performance. To specify no partition, use the phrase WITH NOPARTITION.

Constraints

You can specify constraints to ensure that the contents of columns fulfill your database requirements.

Ordinary (enforced) constraints are always checked at the end of every statement that modifies the table. If the constraint is violated, an error is returned and the statement is aborted. If the statement is within a multi-statement transaction, the

transaction is not aborted.

Constraints can also be declared as NOT ENFORCED. This allows the database designer to describe a constraint (such as a referential relationship) without the overhead of checking the constraint. The assumption is that the constraint will be enforced externally in some way, and so the DBMS does not have to do it. The constraint description is available to the query optimizer and to external query generators, allowing better query plans or better queries to be generated. If the actual data violates the non-enforced constraint, incorrect results are possible.

Constraints can be specified for individual columns or for the entire table. For details, see Table-level and Column-level Constraints.

The types of constraints are:

- Unique constraint–Ensures that a value appears in a column only once. Unique constraints are specified using the UNIQUE option.
- Referential constraint–Ensures that a value assigned to a column appears in a corresponding column in another table. Referential constraints are specified using the REFERENCES option.
- Primary key constraint–Declares one or more columns for use in referential constraints in other tables. Primary keys must be unique.

Additional column-level constraints for masking or encrypting sensitive PI data is available in Avalanche.

Here are some examples:

Create a table in which the social security number is encrypted using AES 256-bit encryption:

```
CREATE TABLE socsectab (
   fname CHAR(10),
   lname CHAR(20),
   socsec CHAR(11) ENCRYPT )
   WITH ENCRYPTION=AES256, PASSPHRASE='decoder ring', NOPARTITION;
```

Create a table with the address and salary columns masked:

```
CREATE TABLE employee(
   name VARCHAR(20),
   address VARCHAR(20) MASKED,
   salary FLOAT MASKED AS 0)
WITH ENCRYPTION=AES256,  NOPARTITION;
```

Migrating Stored Procedures

Stored procedure are implemented as a convenience. Stored procedure architecture has serious downsides in a big data SQL data warehouse environment.

Avalanche is a columnar storage data warehouse. Stored procedures, in general, do not result in a high performing data access layer. Typically, stored procedures will read more columns from the underlying database than needed. Resulting in more I/O causing queries to run slower simply because they perform more I/O.

Additionally, stored procedures implement business logic. Typically, they do row-by-row execution in a loop. Avalanche is highly optimized to do set-wise, vectorized processing using single instruction multiple data (SIMD) operations in a highly parallelized environment. Implementing for each row logic will seriously damage the performance of the system.

Guidelines for stored procedures are to use set-based operations that do not inhibit the performance of Avalanche.

Below is an example of a Netezza stored procedure.

This procedure selects the next value from a sequence named seq_snap. It then inserts a new row into table sales_snapshot with the new sequence number, the snapshot name, date, timestamp with other relevant metadata.

```
CREATE or REPLACE PROCEDURE  CREATE_SALES_SNAPSHOT(CHARACTER VARYING(ANY), DATE, TIMESTAMP)
   RETURNS BIGINT
   LANGUAGE NZPLSQL
   EXECUTE AS CALLER
AS BEGIN_PROC
DECLARE snap_name ALIAS FOR $1;
      snap_date ALIAS FOR $2;
      snap_timestmp ALIAS FOR $3;
DECLARE next_snapshot_id INT8;
BEGIN
next_snapshot_id := SELECT NEXT VALUE FOR "seq_snap";
IF EXISTS (SELECT 1 FROM sales_snapshot
         WHERE snap_name = snap_name
          and snap_date = snap_date
          and snap_time = snap_timestmp )
  THEN RAISE NOTICE 'Snapshot already exists. Please enter a new Snapshot name';
ELSE
      INSERT INTO sales_snapshot
               VALUES ( next_snapshot_id, snap_name, snap_date, now(), snap_timestmp, now(), 'COMPLETE' );
END IF;
END;
END_PROC;
```

The converted procedure in Avalanche looks like the following:

```
CREATE or REPLACE PROCEDURE CREATE_SALES_SNAPSHOT
   (snap_name VARCHAR(50) NOT NULL, snap_date DATE NOT NULL,
    snap_timestmp TIMESTAMP) AS
DECLARE
   next_snapshot_id INTEGER;
   err INTEGER;
BEGIN
      SELECT NEXT VALUE FOR "actian"."seq_snap" into :next_snapshot_id;
      SELECT count(*) as cnt into :err FROM SALES_SNAPSHOT
         WHERE snap_name = :snap_name
           and SNAP_DATE = :snap_date
           and snap_time = :snap_timestmp;
       IF (:err != 0)
       THEN RAISE ERROR 1234 'Snapshot already exists. Please enter a new Snapshot name';
           MESSAGE 999 'Duplicate Snapshot';
           RETURN 1;
        ELSE
           INSERT INTO SALES_SNAPSHOT
                   VALUES ( :next_snapshot_id, :snap_name, :snap_date, :snap_timestmp, sysdate, 'COMPLETE' );
           MESSAGE IIROWCOUNT ' Inserted';
           RETURN 0;
ENDIF;
END;
```

Migrating Data

Moving data from Netezza to Avalanche is comprised of multiple steps:

1. Identify and unload required tables
2. Perform charset/codepage conversion, if needed
3. Stage the data in cloud storage and load the data into Avalanche

Unloading Data

Use Netezza external tables to unload your tables for staging to the cloud.

Avalanche use external tables to load data from S3 or ADLS.

For AWS, depending on the size of your data that will be staged in the cloud, we recommend the use of AWS Snowball. Snowball is a petabyte-scale data transport solution that uses secure devices to transfer large amounts of data into and out of S3. Using Snowball addresses common challenges with large-scale data transfers, including high network costs, long transfer times and security concerns.

For Azure, use Data Box offline data transfer products to move large amounts of data to Azure when you are limited by time, network availability or costs.

Both products provide secure encryption of your data.

If your data is of a manageable size, files can be uploaded to cloud storage using AWS Console, Azure portal, or a command line CLI provided by AWS or Azure.

To ensure your data is cloud compatible, provide an external table definition that includes headers or column names. Additionally, we recommend that you unload tables using multiple files for large tables. Avalanche will load multiple files in a directory in parallel, dramatically speeding up the load process.

Below is an example of creating an external table in Netezza for unloading data:

To unload a larger table into different CSV files in a directory, use the Netezza pseudocolumn DATASLICEID.

```
CREATE EXTERNAL TABLE '/mnt/data/mytable/part_000.csv'
USING
(
   IncludeHeader
   DELIMITER ','
   ENCODING 'internal'
   REMOTESOURCE 'ODBC'
   NULLVALUE ''
)
AS
SELECT *
  FROM MYTABLE WHERE DATASLICEID BETWEEN 0 and 6
```

Smaller tables can be unloaded as a single CSV file. Use the directory format in the external table above. Best practice is to create a directory for each table and unload the table data to one or more files in that directory.

ENCODING

Netezza SQL treats char and varchar as Latin-9 encoding for western European character data.

Latin-9 covers most Western European written languages such as French, Spanish, Catalan, Galician, Basque, Portuguese, Italian, Albanian, Afrikaans, Dutch, German, Danish, Swedish, Norwegian, Finnish, Faroese, Icelandic, Irish, Scottish and English, but none of the central European languages such as Polish, Czech, Hungarian and Romanian.

Avalanche uses UTF-8 encoding. For loading data from delimited files (CSV, TSV, etc.), UTF-8 is the default. UTF-8 is the World Wide Web's most common character encoding.

Netezza extends its character set support in two ways:

- The char/varchar data type uses the Latin-9 code set
- The nchar/nvarchar data type supports Unicode by using the UTF-8 encoding

Code points 1 to 127 are the same in both Latin-9 (ISO-8859-15) and UTF-8.

This feature in Netezza results in unloaded text files that may contain both encodings and text files that may contain invalid byte sequences in UTF-8. Code points greater than 127 in Latin-9 results in invalid UTF-8 characters.

To covert internationalized data to UTF-8 encoding:

- Use an ETL tool that can understand and convert code pages, or
- For files containing a single encoding such as Latin-9, use a Linux conversion utility like iconv to convert the file from Latin-9 to UTF-8
  eg. iconv -f LATIN-9 -t UTF8 < inputfile > outputfile
- For tables containing both Latin-9 and UTF-8, unload the data into different files using a table primary key and convert the Latin-9 file to UTF-8.

Staging Data in the Cloud

Transferring large amounts of data may be very time-consuming and expensive. Use an AWS Snowball or Azure Data Box to easily and securely move your unloaded and converted UTF-8 files to the cloud.

Alternatively, use AWS Console or Azure Portal in a browser to upload the directories and files to an S3 bucket or ADLS directory.

Optionally, both AWS and Azure provide command-line interface (CLI) that can be downloaded and installed to move data to cloud storage.

Choose your preferred transfer method and upload your data to cloud storage.

Loading Data into Avalanche

Avalanche use external tables to load data from S3 or ADLS.

To create the external table for data with a header row:

```
CREATE EXTERNAL TABLE parts_s3 (
   p_partkey INTEGER NOT NULL,
   p_name VARCHAR(55) NOT NULL,
   p_mfgr CHAR(25) NOT NULL,
   p_brand CHAR(10) NOT NULL,
   p_type VARCHAR(25) NOT NULL,
   p_size INTEGER NOT NULL,
   p_container CHAR(10) NOT NULL,
   p_retailprice DECIMAL(18,2) NOT NULL,
   p_comment VARCHAR(23) NOT NULL
) USING SPARK WITH
REFERENCE='s3a://<bucket>/part.tbl*',FORMAT='csv',OPTIONS=('header'='true','delimiter'='|');
```

Important! Ensure that your external table column names match the names used for columns in your source data header row.

You can now manipulate the data using standard SQL against the external table.

To load data into your table, use standard SQL. To load the parts table from external table parts_s3, use an INSERT statement:

INSERT INTO parts SELECT * FROM parts_s3;

This statement reads the data from the external table “parts_s3”, which points to the data on S3 and inserts it into the native Avalanche table “parts”.

Migrating Queries and Applications

Avalanche is extremely easy to adopt due to its use of industry-standard ANSI SQL:2016 and its extensive support for data formats.

Most standard SQL queries will execute without requiring any changes. There are specific extensions and functions native to Netezza.

A shortcut to cast data types, the :: notation is broadly used by the Netezza community. For example, to cast a number as a BIGINT, you can specify SELECT 12345::bigint;

Avalanche makes use of the standard CAST operator that is also available in Netezza.

The CAST operator takes the form: CAST (<int2-column> AS NUMERIC(12,3)).

Another Netezza extension is the NOW() function that returns the current date and time. In Avalanche, you can use the SYSDATE built-in function that returns the current date and time. Alternatively, use CURRENT_TIMESTAMP.

E.g. SELECT SYSDATE; or SELECT CURRENT_TIMESTAMP;

Avalanche has been certified for use with popular BI tools like Looker, Tableau, and Qlik. Avalanche supports ODBC, JDBC and .Net and comes with a set of connectors to popular sources, including Salesforce, NetSuite, Workday and ServiceNow. With these standard connectivity tools, you should be able to seamlessly connect your favorite applications to Avalanche.

About Actian – Activate your Data™

Actian, the hybrid data management, analytics and integration company, delivers data as a competitive advantage to thousands of customers worldwide. Through the deployment of innovative hybrid data technologies and solutions Actian ensures that business critical systems can transact and integrate at their very best – on premise, in the cloud or both. Thousands of forward-thinking organizations around the globe trust Actian to help them solve the toughest data challenges to transform how they run their businesses, today and in the future. For more, visit https://www.actian.com.

2300 Geng Rd, Suite 150, Palo Alto, CA 94303
+1 888 446 4737 [Toll Free] | +1 650 587 5500