TECHNICAL GUIDE

snowflake®

NETEZZA TO SNOWFLAKE MIGRATION REFERENCE MANUAL

CHAMPION GUIDES

TABLE OF CONTENTS

INTRODUCTION

This document provides the high-level methodology needed to prepare for and execute the migration of an existing Netezza system to Snowflake. This reference manual is not intended to be a comprehensive step-by-step process for migration, but rather a snapshot to help with planning and project scoping. The appendices at the end of this document describe the differences between Netezza and Snowflake you should consider as part of the migration.

The intended audiences of this document are solution architects, program managers, and Snowflake solution partners who need a clearly defined approach for migrating an existing Netezza system to Snowflake.

PREPARING FOR THE MIGRATION

Successful data migration projects start with a well-designed plan. An effective plan accounts for the many components that need to be considered, paying particular attention to architecture and data preparation. This section gives you a checklist of information to gather and decisions to make before you start the actual migration.

DOCUMENT THE EXISTING SOLUTION

KEY OUTCOMES:

- List of Netezza databases and schemas to migrate
- List of Netezza database objects to migrate
- List of processes and tools that populate and pull data from Netezza
- List of security roles, users, and permissions
- List of Snowflake accounts that exist or need creating
- Frequency of security provisioning processes
- Documentation of the existing Netezza solution into an as-is architecture diagram
- Documentation of all third-party and language connectors required, including version and platform information

Begin preparing for the migration from Netezza to Snowflake by determining which Netezza databases and schemas within the Netezza system need migrating and which ones don't. Then, identify and document the database objects within the Netezza databases that need migrating, including the size of the data, to establish the scope of the migration project. Plan to exclude from the migration Netezza specific databases such as the Query History database, the SYSTEM and MASTER_DB databases, and any others not needed in Snowflake. A full list of Netezza databases to exclude from the migration is in Appendix A.

Use the script `nz_db_size` to get a list of all databases and their compressed sizes. Before launching the migration, remove any obsolete tables identified. Avoid migrating unused objects unless you need them for audit or historical purposes.

After you've identified the Netezza databases and database objects that need migrating, begin evaluating the data sources that populate them to determine whether they come from an on-premises or a cloud-based source. This will help determine the methods available for loading the data into Snowflake. Specifically, determine whether you need to load terabytes or petabytes of on-premises data into Snowflake. Depending on the data volume, you may require capabilities such as AWS Snowball or Azure Data Box to move the data as efficiently as possible.

In addition to evaluating the data sources that populate Netezza, document the processes and tools that move data into Netezza and pull data from Netezza (for example, ETL and ELT tools, scripting languages, reporting and visualization tools, data science processes, and machine learning processes). Also document which drivers (for example, ODBC, JDBC, and .NET) are required to connect to these tools and download the equivalent Snowflake drivers. Use this list to evaluate the level of Snowflake support for the tools currently in use, as well as to provide guidance on what migration approach would best serve your needs.

Document the groups and users that currently exist within the Netezza system, and the granted permissions, to prepare for the security implementation in Snowflake. Pay special attention to sensitive data sets and how they're secured within Netezza. Also determine how frequently security provisioning processes run to create similar security within Snowflake. In addition, capture the Snowflake accounts already set up and any Snowflake accounts needed for the migration, because they will have an impact on the security implementation.

If you do not have this information readily available, a Snowflake solution partner can help capture this information.

ESTABLISH A MIGRATION APPROACH

KEY OUTCOMES:

- List of processes to migrate as is
- List of processes that need reengineering
- List of processes that need fixing
- Draft of migration deliverables
- To-be architecture diagram

After you've documented your existing Netezza solution into an as-is architecture diagram, focus on your migration approach. Consider how much reengineering you want to undertake as part of the migration. Organizations typically fall somewhere between wanting to take the existing solution as is or completely reworking the existing solution. Unless your system is broken, we generally recommend minimal reengineering for the first iteration.

The more you migrate the existing solution as is, the more the databases and schemas in Snowflake will look like those in Netezza. This is especially important when pointing reporting and visualization tools from Netezza to Snowflake with minimal rework. The more rework done, the more development and testing required, which could lead to a longer migration project and higher risk.

Also include in your migration plan any issues with your existing implementation that need to be resolved.

Break the migration down into incremental deliverables. In addition to making the transition more manageable, an incremental approach enables your organization to start reaping the benefits of Snowflake sooner.

Use the as-is architecture diagram to create a to-be architecture diagram for communicating the migration approach and ensuring the approach meets the requirements of the business.

CAPTURE THE DEVELOPMENT AND DEPLOYMENT PROCESSES

KEY OUTCOMES:

- List of tools introduced with the migration
- List of tools deprecated after the migration
- List of development environments needed for the migration
- List of deployment processes used for the migration

Depending on your migration approach, you may introduce new tools or deprecate old tools as part of the migration. Since you documented your existing tools and processes in an earlier step, this is the where you should document plans to introduce new tools or deprecate old tools.

Your organization may want to change your development or deployment processes as part of the migration. Regardless of whether these processes change, capture the development environments used for the migration (for example, Pre-Prod/Prod, Dev/QA/Prod), and the deployment processes (for example, source control repository or method for deploying changes from one environment to another) used for the migration. This information is critical to how you will implement development and deployment.

PRIORITIZE DATA SETS FOR MIGRATION

KEY OUTCOMES:

- List of data sets to migrate first
- Method for identifying process dependencies for data sets
- Documentation of process dependencies for data sets

To deliver value as soon as possible, identify which data sets you should migrate first. The ideal candidates provide value to the business with minimal migration effort. Rather than starting with the most complex data sets, begin with a simpler data set that provides a quick win and establishes a foundation for the development and deployment processes from which to build the rest of the migration.

After you've prioritized your list of data sets, leverage it with the above principles in mind. If you don't have a prioritized list, identify those data sets and engage a Snowflake solution partner, if necessary, to help capture this information.

To prioritize data sets for migration, pay careful attention to the process dependencies of the data sets and document those dependencies. By having a solid process to identify dependencies before beginning the migration work, you will experience fewer obstacles during the migration.

Ideally, you can use an automated process that iterates through the existing job schedules and captures the data within Snowflake. This eliminates having to depend on manual work to identify and document changes. Creating an automated process provides value throughout the migration project by more easily identifying ongoing changes throughout the project since the underlying systems are unlikely to remain static during the migration.

IDENTIFY THE MIGRATION TEAM

KEY OUTCOMES:

- List of migration team members and roles
- Contact information for all team members

Document the people involved in the migration, including each team member's name, contact information, and role. Team members may come from your team, Snowflake staff, or a Snowflake solution partner. (A Snowflake solution partner can provide a variety of services, including solution design, requirements gathering, documentation, development, testing, delivery, and training.)

Some of the obvious roles required for a migration are developer, quality assurance engineer, business owner, project manager, program manager, scrum master, and coordinator. The entire team works together to successfully complete the migration and communicate the progress of the migration to stakeholders.

DEFINE THE MIGRATION DEADLINES AND BUDGET

KEY OUTCOMES:

- List of business expectations for the migration deadline
- Documented budget allocated for the migration project
- Template of estimated costs to run Snowflake

Consider deadlines, budget, availability of resources, and amount of reengineering required for your migration. By gathering all of this information, you can establish and communicate achievable deadlines, even if the deadlines differ from what the business expects.

It's common to set migration deadlines before evaluating the scope of the project. This is often done to determine whether the deadlines are achievable, especially if the business is trying to deprecate a system before a renewal date. In situations in which you can't move the deadline and the migration requires more time than is available, work with your stakeholders to agree on a path forward.

It's also critical to understand the budget allocated to complete the migration. Compare the amount of work and the costs associated with the migration to the available budget to ensure you have sufficient funds. Pausing in the middle of a migration, or stopping it altogether, is a bad outcome for everyone.

When considering the budget, be sure to consider the Snowflake virtual warehouses (compute cluster) required to support the migration and the continuing operation of the data warehouse. A Snowflake representative can provide a template and work with you to select the virtual warehouses needed to do the work (for example, ETL/ELT, reporting and visualization, etc.). The template calculates the number of minutes a warehouse is expected to run each day and the number of days a warehouse is expected to run each week. After you complete the template, you will get an estimated annual cost.

DETERMINE THE MIGRATION OUTCOMES

KEY OUTCOMES:

- List of high level outcomes after you complete the migration
- Documented plan for communicating the migration project wins to stakeholders

The final step of preparing for your migration is to capture the high-level outcomes and assumptions the migration has achieved and the benefits they provide the business. For example, turning off a Netezza system could be one of your desired outcomes.

The documentation can also include benchmarks that compare process execution on Netezza and Snowflake. Use this information to communicate the wins of the migration project to stakeholders.

ADDITIONAL RESOURCES AND DOCUMENTATION

KEY OUTCOMES:

- Understanding Snowflake data warehouse service

To get familiar with Snowflake and start using Snowflake, read the Snowflake documentation.

EXECUTING THE MIGRATION

After gathering the information and decisions needed to prepare for the migration, it's time to execute the migration. This section guides you through the steps required to complete the migration.

If you need assistance with any part of executing the migration from Netezza to Snowflake, check with your Snowflake representative for recommendations on tools or Snowflake solution partners.

DEVELOP A TEST PLAN

Determine and execute the appropriate level and scope of testing for each environment. For example, you might want to test schedules only in QA and production, or you might want to compare Netezza and Snowflake data only in production. Automate testing as much as possible, so it's repeatable and provides results for evaluating any issues. Define, document, and get agreement on acceptance criteria for the tests.

ESTABLISH SECURITY

When first setting up a Snowflake account, you can manually create users and roles. But quickly move to an automated process that creates users, assigns them to roles, and removes them when they are no longer applicable. Depending on your security auditing requirements, create processes to create, delete, and modify users and roles.

Your existing Netezza system security can be a good starting point for setting up security in Snowflake. Netezza does not implement a concept of roles, but uses a similar object type of groups. The difference is that roles can be nested within roles, but groups are a single level of hierarchy to hold a list of users. Assigning privileges to the Snowflake role is analogous to assigning privileges to the Netezza group. Determine whether any Netezza groups and users are no longer needed or should be implemented differently as part of your migration to Snowflake.

Capture a list of Netezza groups that contain at least one user. For each group, create a Snowflake role. Within Netezza, groups are at the global system level. If the plan is to consolidate multiple Netezza systems into one Snowflake account, consider a group-naming standard to account for duplicates. For example, you can assign a suffix to the group name to include the platform (i.e., DEV, TEST, QA, PROD).

```
--Run this on Netezza to create a file to feed
into Snowflake
SELECT DISTINCT 'CREATE ROLE ' || groupname ||
'_<platform>' || ';' FROM _v_groupusers ORDER BY
1;
```

Capture a list of Netezza users and create a corresponding Snowflake user for each one. You can assign passwords and other user parameters later.

```
--Run this on Netezza to create a file to feed
into Snowflake
SELECT 'CREATE USER IF NOT EXISTS '||username||';'
FROM _v_user;
```

Capture the Netezza user to group mapping. Grant Snowflake roles to users.

```
--Run this on Netezza to create a file to feed
into Snowflake
SELECT 'grant role ' || groupname || '_<platform>'
|| ' to ' || username || ';' FROM _v_groupusers
ORDER BY groupname, username;
```

Capture a list of all users not assigned to a group. In Netezza, you can assign privileges directly to a user rather than inherited from a group. This is often the case for Service IDs. Evaluate these to determine how to migrate them.

```
For this select statement it does not need
the alias gu;

SELECT u.username FROM _v_user u WHERE u.username
NOT IN (SELECT username FROM _V_groupusers);
```

CREATE THE DATABASES

Because data is isolated between environments (Dev/Test, QA, Production), a Netezza system could have the same database name in both the test and production systems (i.e., SALES). Moving these to a single Snowflake account, in which the database name must be unique, means that one or both of these names would need to change. You have three options to consider:

- **Move each environment to its own Snowflake account, so the database names remain the same.** This preserves the separation between non-production and production environments. However, you would need to manage different sets of user IDs, and you would lose the ability to clone a test database from production.
- **Use schema names to partition the database by environment.** For instance, using the <database>.<schema> convention, you would have SALES.TEST, SALES.QA, SALES.PROD.
- **Rename the non-production databases to unique names:** SALES_TEST, SALES_QA, while leaving production the same: SALES.

Use the third option, unless you have a good reason to use one of the first two.

CREATE THE VIRTUAL WAREHOUSES

Create one virtual warehouse (compute cluster) for each environment (e.g., Dev/QA/Prod) and function or combinations of functions (e.g., ETL/ELT, reporting/visualization, etc.). A simple rule of thumb is to align each virtual warehouse to each Netezza resource group.

The following is example SQL to capture a list of resource groups:

```
SELECT groupname FROM _v_group WHERE
grorsgpercent > 0;
create or replace warehouse <groupname_platform>
  with warehouse_size = <based on sizing
estimates>
       min_cluster_count = 1
       max_cluster_count = 3;
```

See the Netezza to Snowflake sizing directions in Appendix G.

Consider this as a starting topology. Adjust as necessary based on information captured during your migration preparation and any performance behavior you observed during the migration.

Set up resource monitors to track usage and take appropriate action when thresholds and limits are reached. Preferably, establish one resource monitor per virtual warehouse to measure whether initial definitions are accurate.

CREATE THE DATABASE OBJECTS

After creating the appropriate databases in Snowflake, create the database objects. To capture the data definition language (DDL) from your Netezza system, use the nz_ddl_* scripts from the Netezza Software Support Toolkit (/nz/support/bin) for each Netezza database you will migrate.

Sequences

The following script will produce output containing DDL to create all sequences within the specified database on Snowflake.

```
/nz/support/bin/nz_ddl_sequence <database>
```

The starting value for the sequence will be the next value assigned at the time of this run.

Tables

This script will produce output containing DDL to create all tables within the specified database on Snowflake.

```
/nz/support/bin/nz_ddl_table <database>
```

You'll also need to evaluate the Netezza DDL for compatibility. The main change will be to remove the `DISTRIBUTE ON` clause for all database tables, since Snowflake does not use table distributions. You can change the Netezza `ORGANIZE ON` clause to the Snowflake `CLUSTER BY` clause since the desired outcome (reducing the amount of data read to satisfy a query) is the same for both systems. Note that Snowflake will automate the clustering of data via the Clustering Service, so there will be no need to script this re-clustering via `GROOM` or any other commands.

NOTE: When creating DDL objects, avoid creating tables with undefined VARCHAR lengths. Snowflake by default uses the MAX field length (16 MB). When fetching from these fields using ODBC or other connectors, Snowflake will allocate 16 MB during the fetch, which could create performance degradation depending on the size of the result set.

Synonyms

This script will produce output containing DDL to create all synonyms within the specified database on Snowflake.

```
/nz/support/bin/nz_ddl_synonym <database>
```

Snowflake does not offer the object synonym. You can use views in the place of synonyms.

Views

This script will produce output containing DDL to create all views within the specified database that can be replayed on Snowflake.

```
/nz/support/bin/nz_ddl_view <database>
```

NOTE: This will consist of the view definition as modified and stored by Netezza. It will not be the original SQL used for the initial CREATE VIEW statement. Ordinarily, this would not present a problem. But there are times where the modified version of the view does not replay as expected. For example, the nz_ddl_view output may cast literal string values to VARCHAR without a length, which defaults to 16 MB in Snowflake. This should be changed to include the actual max length of the string.

Materialized Views

The following script will produce output containing DDL to create all materialized views within the specified database on Snowflake.

```
/nz/support/bin/nz_ddl_mview <database>
```

Netezza materialized views, though rarely used (in favor of ORGANIZE BY capabilities), provided physical copies of the table data, which had the following purposes:

- Enable a different sort order to better exploit zone maps for a certain query set.
- Provide a subset of columns as a way to minimize disk I/O (Netezza is row organized).

Snowflake Materialized Views can support the same use cases as those used for Netezza. Like Netezza, they support operations on a single table, but unlike Netezza, they also support certain aggregations within the materialized view. Also, these views are automatically maintained via the materialized view service, so you do not need to manually refresh the views. See Working with Materialized Views for more info on this feature.

User Defined Functions

The following script will produce output containing DDL to create all user-defined functions within the specified database on Snowflake.

```
/nz/support/bin/nz_ddl_function <database>
```

This will provide the DDL required to register the UDF but it will not include the C++ source code, which you can rewrite using JavaScript or SQL.

Stored Procedures

The following script will produce output containing DDL to create all stored procedures within the specified database.

```
/nz/support/bin/nz_ddl_procedure <database>
```

Convert these stored procedures to use Snowflake's JavaScript stored procedure language. See Working with Stored Procedures and Stored Procedures API for more details.

GRANT AUTHORIZATIONS TO THE DATABASE OBJECTS

After you've created the database objects and virtual warehouses, assign the appropriate privileges to the various roles. In Netezza, you can grant SELECT or other privileges to all tables and views at the database level, which all current and future tables and views within that database will inherit. In Snowflake, bulk granting (GRANT SELECT ON ALL TABLES) will address all existing tables and views within a database but will not account for future tables and views. For future tables and views, use a separate statement to grant future privileges (GRANT SELECT ON FUTURE TABLES). See Configuring Access Control for more information.

Build a file to contain Netezza grant statements for each group. You'll have to update this to account for Snowflake comparable privileges and to eliminate privileges no longer necessary (for example, there is no GROOM privilege in Snowflake). See Appendix F for a Netezza-to-Snowflake privilege mapping.

```
/nz/support/bin/nz_my_grants <groupname>
```

LOAD INITIAL DATA SETS

To load data into Snowflake, you must stage the data in a cloud storage service such as AWS S3 or Azure Blob Storage. First, identify a method for moving the Netezza data and any other data intended for Snowflake to cloud storage. Take into account the amount of data and expected network transit time, including any potential firewall or security issues.

You may require an AWS Snowball or Azure Data Box if you need to move terabytes or petabytes of data. Add an appropriate amount of time to the migration schedule to provision these boxes, load them with data, transport them to the cloud data center, and offload the data into the cloud servers.

However you choose to move your data, you will need to extract the data for each table to one or more delimited flat files in text format using either external tables to get single files, or NZ_UNLOAD or NZ_BACKUP to create several files in parallel. With any option, generating compressed output files will save both storage space and will reduce network transfer time. You can then upload these files using the Snowflake PUT command into an Amazon S3 or Azure Blob Storage staging bucket, either internal or external, or use standard AWS or Microsoft tools for this data movement. We recommend these files be between 100 MB and 1 GB to take advantage of Snowflake's parallel bulk loading.

You can load data into Snowflake after you've extracted the data from Netezza and moved to the cloud. Use this data loading to test the configuration of the databases, database objects, and virtual warehouses, as well as the security that you have implemented. You can learn more about loading data in the Snowflake documentation.

You may be able to use cloning to move data within Snowflake from one database to another, depending on which Netezza environment the data came from and which Snowflake database you populate. This will require fewer resources than loading the same data multiple times into different Snowflake databases.

Plan to extract data from Netezza into Snowflake more than once. Sometimes issues arise during extraction. Also the ETL and ELT processes may not be ready to update the data when you load the initial data sets. You should begin with a subset of the data from Netezza, rather than trying to load the entire contents of the Netezza system at the beginning of the migration.

In addition, you might not extract or load the data in the same order that it exists in Netezza. Therefore, analyze large tables to ensure they are clustered properly in Snowflake. If not, reorder them by the appropriate clustering columns (usually a date and timestamp and possibly one or more other columns). You can accomplish this either through a CTAS or INSERT with an ORDER BY, or through the Snowflake CLUSTER BY command.

KEEP DATA UP TO DATE

You can implement processes to keep the data current after you load the historical data sets from Netezza, so that a complete history is available in Snowflake.

Set up the appropriate data loading schedules to reflect the existing Netezza loading processes or new processes to load Snowflake. This is another opportunity to evaluate whether changes to the schedule would benefit the migration.

To ensure that data is populated in the correct order, create the appropriate schedules based on a clear understanding of the process dependencies captured as part of preparing for the migration.

Along with scheduling the processes to run, monitor those processes so you understand and communicate the state of the data loading: in progress, completed successfully, and any failures that you need to address. Use the monitoring to compare execution results to verify your meeting SLAs within Snowflake or identify performance and process issues.

REMOVE OR IMPLEMENT ADMISTRATION TASKS

Snowflake handles many routine Netezza maintenance processes automatically as a service. Therefore, remove them from any scripts. They include:

- `GENERATE STATISTICS`
- `GROOM`
- All processes related to backup and restore
- All processes related to monitoring and alerting for hardware or other infrastructure issues
- Weekly or monthly maintenance (nz_manual_vacuum, nzstop/nzstart)
- Dual feel ETL or nz_migrate jobs in support of load balancing or disaster recovery

The few Snowflake administrative tasks to consider implementing as part of the migration include:

- Reclustering of very large tables if cluster ratio/depth leads to query slowdown
- Cloning to periodically refresh test data
- Cloning to provide for data backups greater than 90 days old

Because ETL and other data loading processes can run on their own isolated compute cluster, you may also choose to remove restrictions on when these processes can run, as well as any restrictions on user access while these processes are running.

In addition, because storage space is not an issue in Snowflake, you may choose to remove or change processes that purge older data. For example, instead of purging data older than six months, you may choose to keep data for three years or choose to eliminate the purge processes completely.

Transient data, such as intermediate or stage tables, should not consume storage for the purpose of time travel or fail safe. Consider declaring these types of tables as temporary if they are needed only within a particular session, or transient if they need to be globally visible.

IMPLEMENT THE TEST PLAN

Begin the Snowflake implementation with the initial data sets loaded and processes running so the data remains current. Then, start testing your Snowflake implementation. Be sure to engage the identified team members and additional groups to test their data sets and applications against Snowflake. Engage these additional groups after you complete initial testing, confirming that the data is ready for further scrutiny.

Compare data between the Netezza and Snowflake environments throughout the migration to confirm its successful progress. If differences exist, investigate to determine the cause and how to resolve the issue.

If your migration includes fixing incorrect Netezza processes, you can expect differences between Netezza and Snowflake. When this happens, use other methods to check whether the data is correct in Snowflake. Document and share any reasons why data may not match between Netezza and Snowflake with groups performing the testing, so they don't spend time researching previously identified issues.

Also, compare the performance of the processes that load and consume data to ensure Snowflake performs as expected. Capture and measure a sample workload from the Netezza environment that sufficiently represents the various processes such as ETL or analytics from key applications. Ideally, reproduce workloads using existing processes. If that option isn't available and you have to capture SQL, run the following query against the active query history database.

```
create temporary table QH as
select npsid, npsinstanceid, opid, 0 as
sequenceid, query as querytext
from "$v_hist_successful_queries"
where substr(upper(querytext),1,5) in ('SELEC',
'WITH ', 'UPDAT', 'INSER', 'DELET', 'CREAT', 'TRUNC',
'DROP ', 'ALTER')
  and submittime between '2018-07-01 08:00:00'
and '2018-07-01 17:00:00'
  and dbname in ('SALES', 'MARKETING')
;

select querytext from
( select npsid, npsinstanceid, opid, sequenceid,
querytext from QH
    union all
  select npsid, npsinstanceid, opid, sequenceid,
querytext
  from "$hist_query_overflow_3"
  where (npsid, npsinstanceid, opid) in (select
npsid, npsinstanceid, opid from QH)
    union all
  select npsid, npsinstanceid, opid, 999999 as
sequenceid, ';' from QH
) sub
order by npsid, npsinstanceid, opid, sequenceid
;
```

- Choose a representative data range
- Choose the appropriate databases for each sample set
- Determine the correct version of the Query History database (version number is a suffix to the history tables)

Use `nzsql` with these specified flags to produce replayable SQL, where HISTDB is the active Query History database.

```
nzsql HISTDB -t -A -R '' -f file_with_above_SQL  >
fully_qualified_output_filename
```

Measure and compare these on Snowflake and look for areas to address and performance gains to highlight. Share these comparisons with stakeholders to generate confidence for the migration.

RUN NETEZZA AND SNOWFLAKE IN PARALLEL

During the migration, run the Netezza and Snowflake systems in parallel just long enough to confirm a successful migration before shutting down Netezza.

While running them in parallel, consider how to best run Netezza and Snowflake to compare data and performance. For example, you may need to create hashes as you extract data from Netezza in order to compare data at the row level between Netezza and Snowflake. This approach is explained further in Appendix E. Perform these comparisons in Snowflake by provisioning resources without negatively impacting your Netezza system.

REDIRECT TOOLS TO SNOWFLAKE

Review the list of tools you gathered while preparing for the migration and the information on the level of support each tool has for Snowflake. Then, after you've migrated a sufficient amount of data to Snowflake for use by each tool, update the tool connections to redirect to Snowflake.

Redirecting tools to Snowflake usually involves creating copies of the existing solution that points to Netezza within the tool and updating the solution to point to Snowflake instead. Compare the output of the tools to ensure the results are the same between Netezza and Snowflake. In addition, evaluate the performance of the tool to verify it's performing as expected in Snowflake.

FINAL CUTOVER

The cutover from Netezza to Snowflake can occur only after you've migrated the initial data, enabled processes to keep the data current, completed testing that verifies you've successfully migrated the data, and redirected the tools from Netezza to Snowflake.

Make sure you've planned and communicated the cutover date in advance to your Netezza users. They should have the ability to log into Snowflake and run the tools they depend on.

To complete the cutover, turn off data processes that populate Netezza. In addition, revoke access to Netezza so users and tools interact solely with Snowflake.

MIGRATION SUCCESS FACTORS

Paying attention to certain success factors will help you to successfully complete the migration. This section provides insight into how to increase the speed of delivery of a successful migration from Netezza to Snowflake.

IDENTIFY AND MITIGATE DIFFERENCES BETWEEN NETEZZA AND SNOWFLAKE

Use Appendix D to identify issues early in the migration process.

After identifying these issues, present them to the business along with available mitigation strategies. Then, confirm that your proposed approach will meet their requirements.

RESOLVE MIGRATION ISSUES

Issues inevitably arise during and after migration. Establish processes to document and escalate migration issues, so you can resolve them as quickly as possible.

The escalation process needs to document each issue, who is responsible for working on the issue, who is responsible for communicating the progress on the issue, and a list of contacts from the business, Snowflake, and any other parties involved in resolving the issue. Be sure everyone logs a support ticket, ask questions, and find resources in the Snowflake Lodge Community.

With the list of issues documented, establish a regular cadence for reviewing the issues and getting an updated status on resolving each issue. Depending on how severe the issue is, you may require daily meetings to review the progress.

You may also identify issues during the migration that don't need resolving during the migration. Document and prioritize them, so you can work on them post-migration.

COMMUNICATE MIGRATION BENEFITS

Using the high-level outcomes captured while preparing for the migration, determine which of the outcomes have tangible evidence of success and document the actual, corresponding benefits that occurred.

Choose tangible evidence meaningful to the business that you can confirm with data (for example, benchmarks that compare performance, or cost savings). Publish these results to stakeholders, so they clearly understand the benefits of the migration and that the time and money required was well spent.

NEED HELP MIGRATING?

Snowflake is available to accelerate your migration, structure and optimize your planning and implementation activities, and apply customer best practices to meet your technology and business objectives. Snowflake's Engagement, Delivery, and Advisory Services Team deploys a powerful combination of data architecture expertise and advanced technical knowledge of the platform to deliver high performing data strategies, proofs of concept, and migration projects.

Our global and regional solution partners also have extensive experience performing proofs of concept and platform migrations. They offer services ranging from high-level architectural recommendations to manual code conversions. Many Snowflake partners have also built tools to automate and accelerate the migration process.

Whether your organization is fully staffed for a platform migration or you need additional expertise, Snowflake and our solution partners have the skills and tools to accelerate your journey to cloud-built data analytics, so you can reap the full benefits of Snowflake quickly. To find out more, contact the Snowflake sales team or visit Snowflake's Customer Community Lodge.

APPENDICES

APPENDIX A
NETEZZA DATABASES TO EXCLUDE WHEN MIGRATING TO SNOWFLAKE

The following list of databases are needed for Netezza only but should be excluded from the migration to Snowflake:

- SYSTEM
- MASTER_DB
- Query History database
    - SHOW HISTORY CONFIGURATION ALL; -> Look for CONFIG_DBNAME values
- Databases that may exist for the sole purpose of establishing user defined functions from the Netezza SQL Extensions Toolkit. These are not required to follow a standard naming convention.
- “Admin” databases that hold homegrown defined metrics related to the system.

APPENDIX B
CONVERTING NETEZZA DATA TYPES TO SNOWFLAKE DATA TYPES

NETEZZA DATA TYPE	NOTES	SNOWFLAKE DATA TYPE	NOTES
BOOL/BOOLEAN		BOOLEAN	
CHAR	Max 64,000	CHAR	Stored as VARCHAR Default 1 MB, Max 16 MB
VARCHAR	Max 64,000	VARCHAR	Max/default is 16 MB
NCHAR	Max 16,000	CHAR	CHAR handles UTF-8 by default, so NCHAR type not needed
NVARCHAR	Max 16,000	VARCHAR	VARCHAR handles UTF-8 by default, so NVARCHAR type not needed
DATE		DATE	
TIMESTAMP		TIMESTAMP TIMESTAMP_LTZ TIMESTAMP_NTZ TIMESTAMP_TZ	Snowflake supports timestamps stored with and without timezone info
TIME		TIME	
INTERVAL		N/A	INTERVAL data types aren't supported in Snowflake, but INTERVAL constants can be used for date arithmetic.
TIME WITH TIME ZONE		TIMESTAMP_TZ	Use timestamp with time zone when time zone info is needed
REAL		REAL	Stored as FLOAT
DOUBLE		DOUBLE	Stored as FLOAT
FLOAT		FLOAT	Stored as FLOAT
DECIMAL(p,s) NUMERIC(p,s)		DECIMAL(p,s) NUMERIC(p,s) NUMBER(p,s)	DECIMAL, NUMERIC and NUMBER are synonyms
BYTEINT		BYTEINT	All integer types are stored as NUMBER(38,0)
SMALLINT		SMALLINT	All integer types are stored as NUMBER(38,0)
INTEGER		INTEGER	All integer types are stored as NUMBER(38,0)
BIGINT		BIGINT	All integer types are stored as NUMBER(38,0)
Internal data types rowid, datasliceid, createxid, deletexid		Not applicable in Snowflake	
XML, SPATIAL, VARBINARY, ST_GEOMETRY other binary types	Max 64 KB. Not strictly data types but stored in VARCHAR by SQL Extensions Toolkit functions	VARBINARY VARIANT	Data can be stored in binary form or in native semi-structured form in Snowflake

APPENDIX C
NETEZZA CLI COMMANDS TO SNOWFLAKE

NETEZZA CLI COMMAND	SNOWFLAKE EQUIVALENT	NOTES
nzbackup	CREATE ... CLONE	Necessary only if backup needed for longer period than Time Travel
nzhw	N/A	
nzload	COPY	
nzrestore	CREATE... CLONE... AT | BEFORE	Using Time Travel, can create new clone or use INSERT or UPDATE statements to modify existing table
nzrev	SELECT current_version();	
nzsession	SHOW TRANSACTIONS	
nzsession -abort	Abort from Console	
nzsql	snowsql	
nzstart	ALTER WAREHOUSE RESUME	
nzstate	SHOW WAREHOUSES	
nzstats	SHOW RESOURCE MONITORS	
nzstop	ALTER WAREHOUSE SUSPEND	

APPENDIX D
OTHER KNOWN MIGRATION ISSUES FROM NETEZZA TO SNOWFLAKE

Date subtraction

Netezza will subtract one date from another date (for example, SELECT '2018-12-31' – '2018-12-01'). In Snowflake, to do this same type of date comparison, use the DATEDIFF function (for example, SELECT DATEDIFF(day, '2018-12-01', '2018-12-31')).

No changes should be needed for all other date arithmetic expressions, since Snowflake supports the use of INTERVAL constants in date arithmetic. See Interval Constants for more information.

SUBSTR (start pos < 1)

Netezza: if start is less than 1, it will start counting to the left of position 1.

Snowflake: if start position is 0, then assume 1. If start position is negative, its start position is calculated from the last position of the string.

```
select SUBSTR('abcd',-2,2) as A, SUBSTR('abcd',0,2)
as B, SUBSTR('abcd',2,2) as C;
```

	A	B	C
Netezza		a	bc
Snowflake	cd	ab	bc

Netezza-specific syntax

AGE – Retrieves the interval between two timestamps/dates and provides precision to the same degree as the data. Use the DATEDIFF function in Snowflake to achieve this functionality.

Netezza	Snowflake
Select age('10-22-2003', '7-6-2002');	SELECT DATEDIFF(day, '2003-10-22', '2002-7-6');

STRPOS(s,b) – Retrieves the starting position of string "b" in string "s". Can be achieved via the POSITION function in Snowflake:

Netezza	Snowflake
SELECT STRPOS("fuzzy wuzzy","uzz");	SELECT POSITION("uzz","fuzzy wuzzy");

CURRENT_PATH – Retrieves the PATH session variables for the current Netezza session. Similar functionality can be achieved in Snowflake by taking substrings from CURRENT_SCHEMAS.

Netezza	Snowflake
SELECT CURRENT_PATH;	SELECT substr(current_schemas(), 3, length(current_schemas()) - 4) as CURRENT_PATH;

Concurrency Considerations

Snowflake does not support short query bias (SQB) or priorities at a session level. Instead, it uses a single FIFO queue for all queries in a single virtual warehouse cluster. You can take either of the following two approaches:

- Using a multi-cluster warehouse (MCW) will enable additional virtual warehouses (up to 10) to be automatically provisioned to handle concurrency demands and automatically suspended when concurrency demands subside.
- Assuming it's possible to delineate small and large queries, run small queries on a smaller warehouse and run large queries on a larger (possibly MCW) warehouse. This will prevent fast running queries from getting queued behind longer running queries.

APPENDIX E
COMPARING DATA FROM NETEZZA WITH SNOWFLAKE

Use row counts and sums of numeric data to validate data matches between Netezza and Snowflake. Another way to confirm you've successfully loaded all data into Snowflake is to get unique values from columns in Netezza and compare those to corresponding values in Snowflake.

For use cases that require more in-depth validation, add an MD5 hash to the data extracted from Netezza. Construct this MD5 hash using columns that won't change when the data is loaded into Snowflake (for example, include key columns and attributes in the hash, but exclude insert and update dates or timestamps that can change based on when the data is loaded into Snowflake). As you load data into Snowflake, generate another MD5 hash across the same set of columns, so you can compared it with the MD5 hash from Netezza. This enables you to compare the contents of the row on the MD5 hash from Netezza with the MD5 hash from Snowflake, rather than comparing each column individually.

APPENDIX F
NETEZZA TO SNOWFLAKE PRIVILEGES

NETEZZA PRIVILEGE	SNOWFLAKE PRIVILEGE	NOTES
LIST	USAGE	Used to view object
SELECT	SELECT	
INSERT	INSERT	
UPDATE	UPDATE	
DELETE	DELETE	
TRUNCATE	TRUNCATE	
LOCK	N/A	
ALTER		Role needs Object Ownership
DROP		Role needs Object Ownership
ABORT	ALTER user ABORT ALL QUERIES	
LOAD	READ on STAGE	COPY command
GENSTATS		N/A
GROOM		N/A
EXECUTE		Authority to execute Stored Procedure
LABEL ACCESS	N/A	Used for Row Secure Tables
LABEL RESTRICT	N/A	Used for Row Secure Tables
LABEL EXPAND	N/A	Used for Row Secure Tables

APPENDIX G
SNOWFLAKE SIZING BASED ON NETEZZA METRICS

Use the following scripts and procedure to capture Netezza utilization data to estimate sizing on the Snowflake platform.

Gathering Netezza Resource Data

Do the following procedure once for each Netezza appliance.

Capture the Netezza model (i.e., N1001-010, N3001-080, etc.). This provides critical metrics such as processor speed, number of cores per blade, and number of blades.

```
/nz/support/bin/nz_get_model
```

NOTE: Netezza stores guaranteed resource allocation (GRA) utilization data in a virtual table called _VT_SCHED_GRA. This table holds approximately seven days of data and will be truncated on database stop and start. Because seven days would likely not be sufficient for accurate analysis, archive this data by copying it periodically (once an hour). Allow history to build for enough time to cover a representative workload.

```
/nz/support/bin/nz_gra_history.ddl – one time
script to create persistent table for this data
/nz/support/bin/nz_gra_history –  periodic copy
of data from virtual table to persistent table
```

See support documentation for more detail on this persisting process.

When enough GRA historical data is available, you can begin analysis. You can run the queries described in this document against the GRA data on the Netezza appliance. Or, you can bring the necessary data into Snowflake for analysis. Use the following statements to capture required data for analysis on Snowflake.

```
CREATE EXTERNAL TABLE '/nzscratch/nzgra.sizing.
dat' USING (DELIMITER '|') AS
SELECT end_time
     , groupname
     , actual_resource_pct
     , rsg_horizon_us
     , busy_secs
     , plans_waiting_long
     , plans_waiting_short
     , plans_running_long
     , plans_running_short
FROM nz_gra_history;

CREATE EXTERNAL TABLE '/nzscratch/dslice.sizing.
dat' USING (DELIMITER '|') AS
SELECT tblid, used_bytes FROM
_v_sys_object_dslice_info;

CREATE EXTERNAL TABLE '/nzscratch/object.sizing.
dat' USING (DELIMITER '|') AS
SELECT objname, objid, objdb, objclass FROM
_t_object;

gzip /nzscratch/*.sizing.dat
```

Send the file 3 gzip files and the Netezza appliance model to Snowflake.

ABOUT SNOWFLAKE

Snowflake is the only data warehouse built for the cloud, enabling the data-driven enterprise with instant elasticity, secure data sharing and per-second pricing, across multiple clouds. Snowflake combines the power of data warehousing, the flexibility of big data platforms and the elasticity of the cloud at a fraction of the cost of traditional solutions. Snowflake: Your data, no limits. Find out more at **snowflake.com**