



MIGRATING ORACLE DATABASE TO SNOWFLAKE: REFERENCE MANUAL



TABLE OF CONTENTS

3 INTRODUCTION

4 PREPARING FOR THE MIGRATION

- 4 Document the existing solution
- 5 Establish a migration approach
- 5 Capture the development and deployment processes
- 6 Prioritize data sets for migration
- 6 Identify the migration team
- 7 Define the migration deadlines and budget
- 7 Determine the migration outcomes

8 EXECUTING THE MIGRATION

- 8 Establish security
- 9 Develop a test plan
- 9 Prepare Snowflake for loading
- 11 Load initial datasets
- 12 Keep data up-to-date
- 12 Implement the test plan
- 12 Run Oracle and Snowflake in parallel
- 13 Redirect tools to Snowflake
- 13 Cut over to Snowflake

14 ENSURING MIGRATION SUCCESS

- 14 Identify and mitigate differences between Oracle and Snowflake
- 14 Resolve migration issues
- 14 Communicate migration benefits
- 15 Need help migrating?

16 APPENDICES

- 16 Appendix A - Oracle Schemas to exclude when migrating to Snowflake
- 17 Appendix B - Converting Oracle data types to Snowflake data types
- 18 Appendix C - Query to evaluate Oracle data type usage

- 19 Appendix D - Other known issues for Oracle to Snowflake migration

- 19 *Enforcement of primary keys & foreign keys*

- 19 *DATE vs. TO_DATE()*

- 19 *Date subtraction*

- 19 *Updating Data Through a View*

- 19 *Oracle Syntax*

- 19 *Stored Procedure*

- 19 *Synonyms*

- 20 Appendix E - Comparing data from Oracle with Snowflake

- 21 About Snowflake

INTRODUCTION

This document provides the high-level process needed to prepare for and execute a migration from an Oracle Database implementation to Snowflake. This reference manual is not intended to be a comprehensive step-by-step method 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 Oracle and Snowflake that you should consider as part of the migration.

The intended audience for this document are data engineers, solution architects, and Snowflake solution partners who need guidance on the scope and process for migrating from Oracle database 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 Oracle databases that need to be migrated
- List of Oracle database objects that need to be migrated
- List of processes and tools that populate and pull data from the Oracle databases
- List of security roles, users, and permissions
- List of Snowflake accounts that exist or need creating
- List of identified issues with design, datasets, or processes that may impact migration
- List detailing the frequency of security provisioning processes
- Documentation of the existing Oracle solution into an as-is architecture diagram

Begin preparing for the migration from Oracle to Snowflake by determining which Oracle databases within the Oracle system need to be migrated. Then, identify and document the database objects within the Oracle databases that need to be migrated, including the size of the data, to establish the migration project scope. Plan to exclude schemas specific to Oracle, such as SYS, SYSAUX, ANONYMOUS, CTXSYS, DBSNMP that aren't needed in Snowflake. Appendix A provides a full list of Oracle schemas that you should exclude from the migration.

When you are unsure which databases and database objects to migrate from Oracle database 12c/18c, you can query the newly introduced ORACLE_MAINTAINED column in the DBA_USERS dictionary view. This column contains a value of "Y" for any schema that was created via Oracle scripts. Avoid migrating unused objects unless they are needed for audit or historical purposes.

After you have identified the Oracle databases and database objects for migration, evaluate the data sources that populate them to determine whether they come from



an on-premises or cloud-based source. This will help you determine the methods available for loading the data into Snowflake. Specifically, look for the number of terabytes or petabytes of on-premises data loaded into Snowflake. If you need to load a large amount of data, you may need to use AWS Snowball or Azure Data Box to move the data as efficiently as possible.

Evaluate the data sources that populate the Oracle databases. Identify and document the processes and tools that move data into the Oracle databases and pull data from the Oracle databases (for example, ETL/ELT tools, scripting languages, reporting and visualization tools, data science processes, and machine learning processes). Use that information 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 fit your needs. As these are critical processes, be sure to document areas that could present issues in the migration.

Document the roles and users that exist within the Oracle system, and their granted permissions, to prepare for the security implementation in Snowflake. Pay special attention to sensitive data sets and how they are secured within the Oracle system. Also determine how frequently security provisioning processes run to create similar security within Snowflake. In addition, document the Snowflake accounts set up and any Snowflake accounts necessary for the migration, since they will affect 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
- Draft of migration deliverables
- Future state architecture diagram

After you have documented your existing Oracle system into an as-is architecture diagram, focus on your migration approach. Document the areas that could present issues in the migration. Consider how much reengineering you want to undertake as part of the migration. Organizations fall somewhere on a

spectrum from wanting to take the existing solution as-is to completely reworking the existing solution. More reengineering requires more development and testing, which extends the length of a migration project. Therefore, unless your system is broken, we generally recommend minimal reengineering for the first iteration.

In addition, as part of your migration, you may need to resolve challenges that exist with your Oracle implementation, so include these in your migration plan.

Break the migration into incremental deliverables that enable your organization to make the transition to Snowflake and provide value to stakeholders sooner.

Use the as-is architecture diagram to create a future state architecture diagram for communicating the migration approach to stakeholders and ensuring that the approach meets their requirements.

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 approach, you might introduce new tools and deprecate old tools as part of the migration. Document plans to introduce new tools or deprecate old tools.

Your organization may want to change its development or deployment processes as part of the migration. Whether these processes change or not, capture the development environments used for the migration (for example, preproduction/production and development/QA/production) and the deployment processes used for the migration (for example, source control repository and method for deploying changes from one environment to another). This information is critical to how you implement the 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 build momentum for the project, identify which data sets to migrate first. Consider high priority data sets that have few dependencies. Begin with a simple data set that provides a quick win and establishes a foundation of development and deployment processes from which to build the rest of the migration.

To prioritize data sets for migration, pay careful attention to the process dependencies of the data sets and document those dependencies. By identifying dependencies before beginning the migration work, you will experience fewer challenges during the migration. When you have a prioritized 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.

Ideally, capture this documentation using an automated process that iterates through the existing job schedules. This will minimize the need to manually identify and document changes. Creating an automated process provides value throughout the migration

project by more easily identifying the ongoing changes that occur throughout the migration project. This is important since the underlying systems are unlikely to be static during the migration.

IDENTIFY THE MIGRATION TEAM

KEY OUTCOMES:

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

To complete the migration plan, document the people involved in the migration and the roles they will play. Team members may come from your organization, Snowflake or a Snowflake solution partner.

The roles required on the migration team include database administrator, quality assurance engineer, business owner, project manager, program manager, scrum master, and communications specialist.

A Snowflake solution partner can fulfill multiple needs including solution design, requirements gathering, documentation, development, testing, delivery, project management, and training. 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
- Completed template for estimating Snowflake virtual warehouse costs

Business expectations for migration deadlines are an important input to your plan. In addition, consider other information such as the budget, availability of resources, and the amount of required reengineering. By gathering all of this information, you can establish and communicate achievable deadlines, even if the deadlines differ from the business expectations.

Often businesses have migration deadlines based on events like needing to deprecate a system before a removal date. Sometimes these deadlines are not achievable. When this happens, work with stakeholders on more realistic migration scenarios.

Be sure to understand the migration budget. Compare the amount of migration work and the associated costs to the available budget to ensure that there are sufficient funds to complete the work.

A key consideration for budget planning is the basic data warehouse sizing, such as the number of compute clusters required to support the migration and the post migration data warehouse. A Snowflake representative can provide a template and work with you to determine the virtual warehouses necessary to do the work (for example, ETL/ELT and reporting and visualization). 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 assumptions and high-level outcomes for the completion of the migration
- Documented plan for communicating the success of the migration project to stakeholders

As the final step of preparing for the migration, capture the assumptions that will determine whether the migration is successful, the high-level outcomes that should be achieved by the migration, and the benefits those outcomes provide for stakeholders. Use this documentation to validate that the migration project provides the overall benefits stakeholders expect to achieve from the migration. For example, if turning off an Oracle system is one of the desired outcomes, the migration plan should include that outcome.

You can express this information as success or failure criteria for the migration project. You might also include benchmarks that compare process execution on Oracle to process execution on Snowflake. After you compile this information, use it to communicate the success of the migration project to stakeholders.

EXECUTING THE MIGRATION

After gathering information and making decisions needed to prepare for the migration, it's time to execute the migration. This section provides guidance on the steps required to complete the migration. If you need assistance with any part of executing the migration from Oracle to Snowflake, check with your Snowflake representative for recommendations on tools or Snowflake solution partners.

ESTABLISH SECURITY

You can manually create roles and users when you initially set up a Snowflake account; however, as soon as possible, you should automate a process for creating roles and users. You should also establish an automated process for users to request system access. Depending on the security requirements, for auditing purposes, you might need to establish and document role creation, user creation, and the assignment of users to roles.

Although the existing Oracle system security can be a good starting point for setting up security within Snowflake, evaluate the Oracle security configuration to determine if there are roles and users that you no longer need or that you should implement differently as part of the migration to Snowflake.

Start by creating roles for at least the first data sets that you will migrate. Assign users to these roles based on the work they will do for the migration. When you complete this setup, users can log into Snowflake to see their roles in preparation for the creation of databases and warehouses in the next step.

You can establish common roles for developer access to non production databases, read-only access, read and write access, and administrative access. You might need additional roles for restricting access to sensitive data.



DEVELOP A TEST PLAN

Determine and execute the appropriate level and scope of testing for each environment (for example, schedules aren't executed in development, but they are in QA and production, and data comparisons between the Oracle system and Snowflake occur only for production). Automate testing as much as possible so that it is repeatable and provides results that you can evaluate. Ensure that acceptance criteria for the tests are defined, agreed to, and documented.

PREPARE SNOWFLAKE FOR DATA LOADING

Create a Snowflake database for each Oracle database that you need to migrate (for example, databases for development, QA, and production). Within the Snowflake databases you create for each Oracle database, create corresponding schemas.

As shown in the following two figures, Oracle Database supports two general configurations:

- Single-instance configuration, where all database functionality and resource management are self-contained. (Figure 1)
- Multitenant configuration, which is a two-part architecture consisting of a container database (CDB) and pluggable databases (PDBs). (Figure 2)

A container database provides a layer of common resources shared across the environment and each pluggable database operates as a set of schemas, objects, and non-schema objects plugged and unplugged from the container database. From the user perspective, a PDB appears as a single database, but it is actually managed within a container that may have one or more PDBs.

FIGURE 1: SINGLE-INSTANCE ORACLE DATABASE TO SNOWFLAKE DATABASE

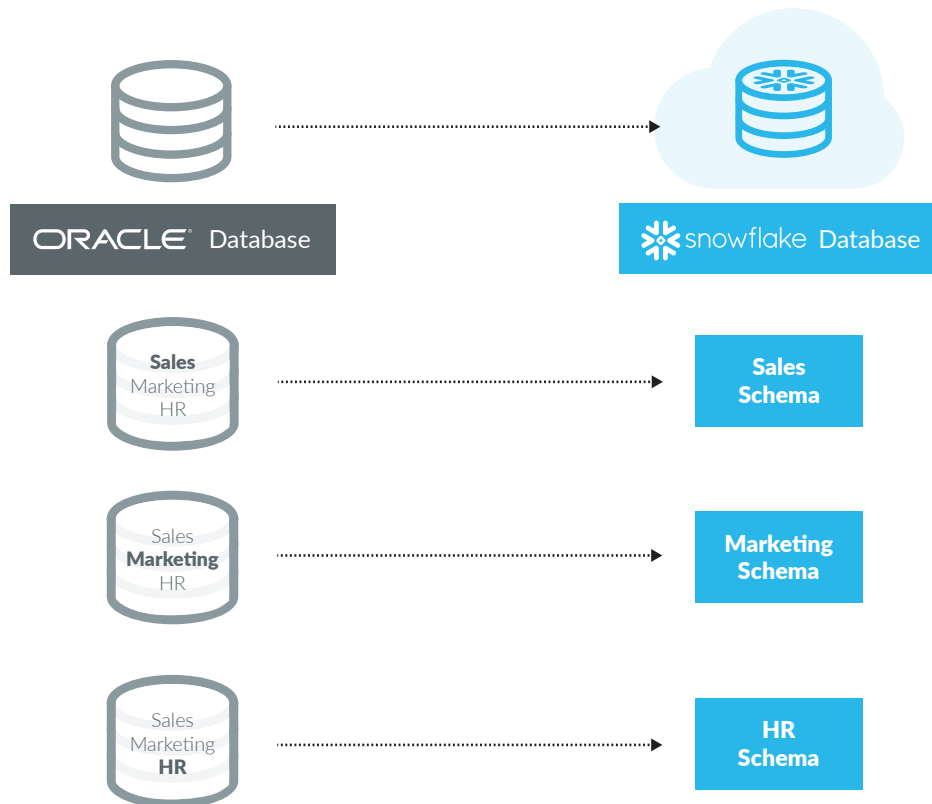
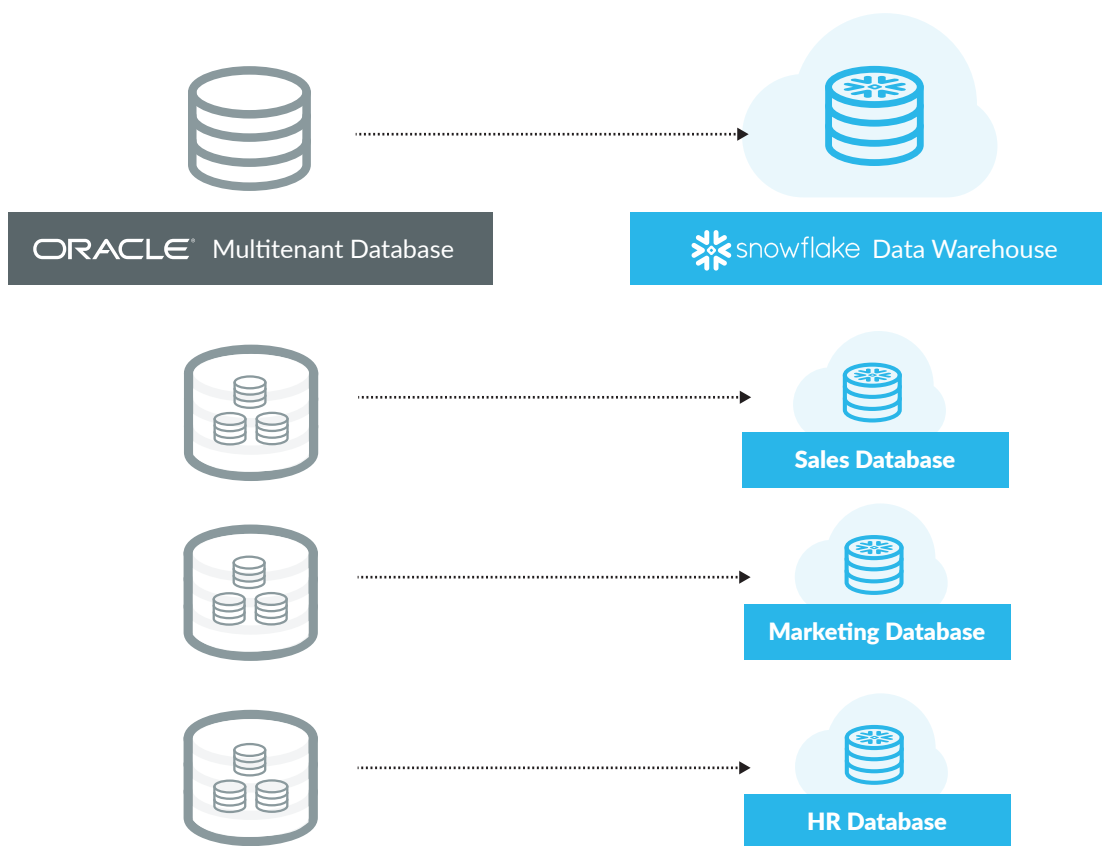


FIGURE 2: MULTITENANT ORACLE DATABASE TO SNOWFLAKE DATABASE



Using the approach described in Figure 2 clearly identifies the environment and uses schemas to contain the tables and views, so that tools can more easily be redirected from the Oracle system to Snowflake.

The Snowflake database warehouse specifies the environment (for example, production or QA) in the view name. Therefore, when you migrate a view from one environment to another, be sure to update the name. For example, migrating a database named QA to one named PROD, using a fully qualified schema object, has the following form:

```
<database_name>.<schema_name>.<object_name>
```

Therefore, the following SQL statement:

```
create or replace view db_view as select *
from QA.public.db_table;
```

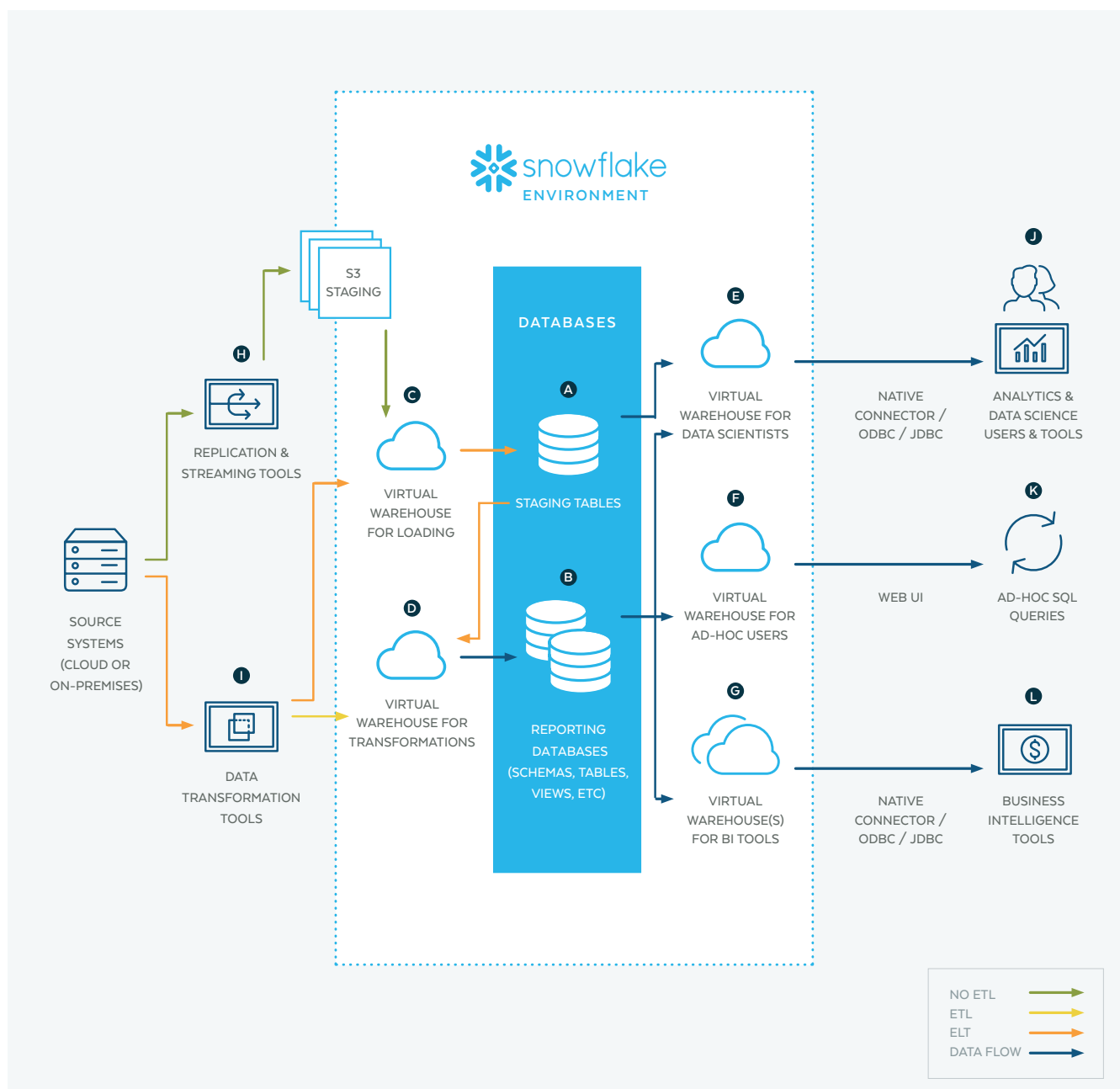
would need to be modified as follows:

```
create or replace view db_view as select * from
PROD.public.db_table;
```

After you create the databases and schemas in Snowflake, you can execute the DDL for creating the database objects in Snowflake.

Create the virtual warehouses based on the information you captured during the migration preparation. As shown in the following figure, there should be a separate virtual warehouse for each function in the environment. The virtual warehouse will support activities such as data science, ad-hoc user queries, and BI tools.

The diagram on the following page contains a reference architecture for using virtual warehouses for different workloads.



Base the initial sizing of the virtual warehouses on the estimates created while preparing for the migration. Then adjust them as needed throughout the migration. Also, set up resource monitors to track usage, and take appropriate action when limits are reached. See the [“Managing Resource Monitors”](#) section in the Snowflake documentation for detailed information.

As you create the databases, database objects, and virtual warehouses, assign them to the appropriate security roles.

LOAD INITIAL DATA SETS

To begin migrating data from the Oracle system to Snowflake, you'll need to extract data from the Oracle system. If the Oracle system is on-premises, and you need to move terabytes or petabytes of data into the cloud, you might need to use [AWS Snowball](#) or [Azure Data Box](#). 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.

After you have moved the data to the cloud, load the data into Snowflake.

See the [“Overview of Data Loading”](#) section in the Snowflake documentation. Use this data loading to test the configuration of the databases, database objects, virtual warehouses, and the security you’ve implemented.

Depending on which Oracle environment the data came from and which Snowflake data warehouse is populated, you could use cloning to move data within Snowflake from one data warehouse to another. Cloning requires fewer resources than loading the same data multiple times into different Snowflake data warehouses.

Due to issues such as failed data type or column mappings that occur with extraction and loading, plan to extract and load data more than once. Also plan for time between when the initial data sets are loaded and when the ETL/ELT processes are ready to keep the data up to date. Begin with a subset of the data from the Oracle system, rather than trying to load the entire contents of the Oracle system at the beginning of the migration. Appendix B and Appendix C of this guide provide additional data type conversion and usage information that can further clarify the data extraction and loading process.

KEEP DATA UP TO DATE

To ensure a complete history is available in Snowflake, wait until after you load the historical data sets from Oracle to implement the processes for keeping the data up to date.

Whether you load data into Snowflake by updating existing Oracle processes or by creating new processes, set up the processes on appropriate schedules. Usually, this means using the same schedules you used for loading the Oracle database. This is another opportunity to evaluate whether changes to the schedule should be part of the migration.

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

In addition to scheduling the processes to run, monitor the processes so you understand and can communicate the state of the data (for example, loading is in progress, loading completed successfully, or loading failures occurred that need addressing). Use monitoring to compare execution results with established SLAs to

verify that SLAs are being met within Snowflake and to identify performance and process issues.

IMPLEMENT THE TEST PLAN

Test the Snowflake implementation after you’ve loaded the initial data sets, and make sure processes are running to keep the data up to date. Engage team members to test their data sets and applications against Snowflake. Then, engage the additional groups after you complete initial testing to perform tests and validate the data.

To make sure the migration has completed successfully, compare data between the Oracle and Snowflake environments throughout the migration. Investigate differences to determine the cause and resolve any issues.

If part of the migration includes fixing processes that were incorrect in the Oracle system, the test results may not match Snowflake. In such a case, use other methods to make sure the data is correct in Snowflake. Document reasons that data won’t match between the Oracle and Snowflake environments and share the documentation with groups who are performing 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 is performing as expected. Share these comparisons with stakeholders to highlight the benefits of migrating from Oracle to Snowflake.

RUN THE ORACLE AND SNOWFLAKE SYSTEMS IN PARALLEL

During the migration, the Oracle and Snowflake systems will run in parallel for a period of time. Minimize the amount of time both systems are running but validate the migration has completed successfully before shutting down the Oracle system.

When both systems are running in parallel, determine how to best compare data and performance. For example, you may need to create hashes as data is extracted from the Oracle system, you can then use to compare data at the row level between the Oracle and Snowflake systems (Appendix E explains this approach further). Perform these comparisons in Snowflake with resources provisioned to compare data without negatively impacting the Oracle system.

REDIRECT TOOLS TO SNOWFLAKE

Find 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 update the tool connections to redirect the tools to Snowflake after you've migrated a sufficient amount of data to Snowflake for use by each tool.

Redirecting tools to Snowflake usually involves creating copies of the existing solution that point to the Oracle database, and updating them to point to Snowflake instead. Compare the output of the tools to ensure the results are the same between the Oracle and Snowflake systems. Also evaluate the performance of the tools to verify they are performing as expected in Snowflake.

CUT OVER TO SNOWFLAKE

After you migrate the initial data from Oracle to Snowflake, implement processes to keep the data up to date, confirm you've migrated the data successfully, and redirect the tools from Oracle to Snowflake that you are ready to cut over to Snowflake.

Plan the cut over date and communicate it to Oracle users so they are aware of when the switch will take place. Make sure they can log into Snowflake and run the redirected tools they depend on.

To complete the cut over, turn off the data processes that populate Oracle and revoke access to the Oracle system.

ENSURING MIGRATION SUCCESS

Paying attention to certain success factors will increase the likelihood of successfully completing the migration. This section provides insight into how to increase the speed of delivery of a successful migration from Oracle to Snowflake.

IDENTIFY AND MITIGATE DIFFERENCES BETWEEN ORACLE AND SNOWFLAKE

Using the information you gathered while preparing for the migration, identify areas that could present issues based on the current usage of Oracle or other tools. [Appendix D](#) lists known issues for migrating from Oracle to Snowflake to help you identify issues early in the migration process.

Identify these issues to set expectations from the beginning of the migration. Then, present them to your stakeholders with available mitigation strategies and the proposed approach to meet their requirements.

RESOLVE MIGRATION ISSUES

Issues are inevitable during and after the migration. Establish processes to document and escalate migration issues so you can resolve them quickly.

The escalation process should document each issue, who is responsible for working on the issue, and who is responsible for communicating progress on the issue. Include a list of contacts from your team, Snowflake, and any other parties involved in resolving the issue. Be sure everyone involved has access to, and knows how to, log a support ticket in the [Snowflake portal](#).

Likewise, they should know how to ask questions and find resources in the [Snowflake community forums](#).

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 an issue is, you may require daily meetings to review the progress.

After documenting the issues, establish a regular cadence for reviewing and getting updates on resolving each issue. Depending on how severe an issue is, you may need to hold daily meetings to review the progress.

You might 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

Use the high-level outcomes you captured while preparing for the migration to document the actual benefits. Publish these results to stakeholders, so they clearly understand the benefits of the migration.

NEED HELP MIGRATING?

Snowflake expert resources are 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 Professional Services 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, please contact the Snowflake sales team or visit [Snowflake's Customer Community Lodge](#).

APPENDICES— ORACLE TO SNOWFLAKE MIGRATION GUIDANCE

APPENDIX A ORACLE SCHEMAS TO EXCLUDE WHEN MIGRATING TO SNOWFLAKE

The following list schemas are needed for Oracle only and shouldn't be migrated to Snowflake:

- ANONYMOUS
- APEX_XXXXXX
- CTXSYS
- DBSNMP
- EXFSYS
- LBACSYS
- MDSYS
- MGMT_VIEW
- OLAPSYS
- ORDDATA
- OWBSYS
- ORDPLUGINS
- ORDSYS
- OUTLN
- SI_INFORMTN_SCHEMA
- SYS
- SYSMAN
- SYSTEM
- WK_TEST
- WKSYS
- WKPROXY
- WMSYS
- XDB
- APEX_PUBLIC_USER
- DIP
- FLOWS_040100
- FLOWS_FILES
- MDDATA
- ORACLE_OCM
- SPATIAL_CSW_ADMIN_USR
- SPATIAL_WFS_ADMIN_USR
- XS\$NULL
- B'
- HR
- OE
- PM
- IX
- SH

An update to the Oracle 12c/18c dba_users view includes a column called ORACLE_MAINTAINED that identifies schemas managed by the database. These schemas are not required for migration.

APPENDIX B

CONVERTING ORACLE DATA TYPES TO SNOWFLAKE DATA TYPES

Snowflake supports most basic SQL data types (with some restrictions) for use in columns, local variables, expressions, parameters, and any other appropriate/suitable locations. Data types are automatically coerced whenever necessary and possible.

| ORACLE | SNOWFLAKE | NOTES |
|--------------------------------|---------------|--|
| VARCHAR2 | VARCHAR | Max 16MB |
| NVARCHAR2 | VARCHAR | Max 8MB (2 byte characters), 4MB (4 byte characters) |
| NUMBER | NUMBER | Numbers up to 38 digits, with a specified precision and scale. NUMBER specified without precision and scale will round loaded data values to a scale of 0. Precision and scale will need to be consistent with that of the data values being loaded. |
| FLOAT | NUMBER | Snowflake FLOAT is a 64 bit floating point number. FLOAT within Oracle is a subtype of NUMBER. Precision and scale will need to be consistent with that of the data values being loaded. |
| DATE | TIMESTAMP_NTZ | Snowflake DATE does not store time. |
| BINARY_FLOAT | FLOAT | Snowflake FLOAT is a 64 bit floating point number. |
| BINARY_DOUBLE | FLOAT | Snowflake FLOAT is a 64 bit floating point number. |
| TIMESTAMP | TIMESTAMP_NTZ | All operations are performed without taking any time zone into account. |
| TIMESTAMP WITH TIME ZONE | TIMESTAMP_TZ | All operations are performed with the time zone offset specified. |
| TIMESTAMP WITH LOCAL TIME ZONE | TIMESTAMP_LTZ | All operations are performed in the current session's time zone, controlled by the TIMEZONE session parameter. |
| INTERVAL YEAR () TO MONTH | n/a | Alternative: Use the TIME_SLICE function to calculate the start and end times of fixed-width "buckets" into which data can be categorized. |
| INTERVAL DAY () TO SECOND | n/a | Alternative: Use the TIME_SLICE function to calculate the start and end times of fixed-width "buckets" into which data can be categorized. |
| RAW | BINARY | Max 8MB |
| LONG RAW | BINARY | Max 8MB |
| ROWID | n/a | Snowflake VARCHAR can be used if ROWID values are migrated to Snowflake. |
| UROWID | n/a | Snowflake VARCHAR can be used if UROWID values are migrated to Snowflake. |
| CHAR | CHAR | Synonymous with VARCHAR in Snowflake. Max 16MB |
| NCHAR | CHAR | Synonymous with VARCHAR in Snowflake Max 8MB (2 byte characters) and 4MB (4 byte characters) |
| CLOB | VARCHAR | Max 16MB |
| NCLOB | VARCHAR | Max 8MB for 2 byte and 4MB for 4 byte |
| BLOB | BINARY | Max 8MB |
| BFILE | n/a | Alternative: Store file location as a VARCHAR and process programmatically. |

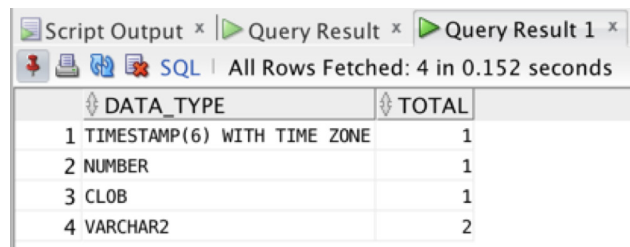
APPENDIX C

QUERY TO EVALUATE ORACLE DATA TYPE USAGE

The query below provides counts for each Oracle data type used in the database schema. You can update the WHERE clause of the query to include the specific list of schemas that need migrating from Oracle to Snowflake. Use the results of this query, along with the list of Oracle and Snowflake data types, to identify the prevalence of data types that may pose challenges during the migration.

```
SET Heading off
SELECT '----- Distinct Column Data Types and
their Count in the Schema: '
FROM dual;
SET Heading on
SELECT data_type, count(*) total
FROM all_tab_columns
--exclude all Oracle Maintained Schemas
WHERE OWNER NOT IN ( 'ANONYMOUS', 'APEX_
XXXXXX', 'CTXSYS', 'DBSNMP', 'EXFSYS', 'LBACSYS',
'MDSYS', 'MGMT_VIEW', 'OLAPSYS', 'ORDDATA', 'OWBSYS',
'ORDPLUGINS', 'ORDSYS', 'OUTLN', 'SI_INFORMTN_
SCHEMA', 'SYS', 'SYSMAN', 'SYSTEM', 'WK_TEST',
'WKSYS', 'WKPROXY', 'WMSYS', 'XDB', 'APEX_PUBLIC_USER',
'DIP', 'FLOWS_040100', 'FLOWS_FILES', 'MDDATA', 'ORACLE_
OCM', 'SPATIAL_CSW_ADMIN_USR', 'SPATIAL_WFS_ADMIN_
USR', 'XS$NULL', 'BI', 'HR', 'OE', 'PM', 'IX', 'SH') ---
&schema_name
GROUP
```

The following figure shows example output from the script:



The screenshot shows a database interface with a 'Query Result' tab. It displays a table with two columns: 'DATA_TYPE' and 'TOTAL'. The table contains four rows of data, representing the results of the SQL query. The status bar indicates 'All Rows Fetched: 4 in 0.152 seconds'.

| DATA_TYPE | TOTAL |
|-------------------------------|-------|
| 1 TIMESTAMP(6) WITH TIME ZONE | 1 |
| 2 NUMBER | 1 |
| 3 CLOB | 1 |
| 4 VARCHAR2 | 2 |

APPENDIX D

OTHER KNOWN ISSUES FOR ORACLE TO SNOWFLAKE MIGRATION

ENFORCEMENT OF PRIMARY KEYS AND FOREIGN KEYS

Oracle enforces Primary Keys and Foreign Keys constraints. While Snowflake supports the syntax to define Primary Keys and Foreign Keys, they are not enforced within Snowflake. This means you need to reengineer load processes that depend on constraints to prevent duplicate entries and orphaned records from being entered into Snowflake.

DATES AND TIMES

DATE vs. TO_DATE()

Oracle has the capability to put DATE in front of a string to treat the string as a date value (for example, SELECT DATE '2018-12-31' from dual;). In Snowflake,

the syntax is TO_DATE() (for example, TO_DATE('2018-12-31')). It is not necessary to use DATE or TO_DATE() in many situations, since both Oracle and Snowflake can interpret the date values stored in a string, but when you migrate SQL from Oracle to Snowflake, replace DATE with TO_DATE() rather than dropping DATE altogether. A notable date/time difference between Oracle and Snowflake is the DATE datatype. DATE within Snowflake holds a date value without time. Migrating Oracle DATE values to Snowflake will require the use of TIMESTAMP_NTZ to persist the time component of those values. Examples of other considerations are below:

| ORACLE | SNOWFLAKE | NOTES |
|------------------------|----------------------------|--|
| Data Type DATE | Data Type TIMESTAMP_NTZ | The Snowflake DATE data type records date without time. |
| TRUNC | TRUNC | Snowflake supports TRUNC as an alias for DATE_TRUNC. |
| SQL Model Format RR | SQL Model Format YY | Snowflake model format YY assumes the first two digits of the full year value as follows: 00-69 will translate to 2000 - 2069 70-99 will translate to 1970 - 1999 |
| TO_DATE | TO_DATE | Behavior within Snowflake will be similar to Oracle. |
| DATEDIFF | DATEDIFF | Behavior within Snowflake will be similar to Oracle. |
| DATEADD | DATEADD | Behavior within Snowflake will be similar to Oracle. |
| TO_CHAR(<date>, 'J') | n/a | The Julian date SQL format model is not currently supported within Snowflake. An alternative would be to create a user defined function to calculate the Julian value from a specified date. |
| <date> - <date> | n/a | Use DATEDIFF instead. The direct subtraction of two dates is not currently supported within Snowflake. |

APPENDIX D (CONT'D)

OTHER KNOWN ISSUES FOR ORACLE TO SNOWFLAKE MIGRATION

Date subtraction

Oracle will subtract one date from another date (for example, `SELECT date '2018-12-31' - date '2018-12-01' from dual;`). 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')`).

Oracle can also subtract integers from dates, for example, `SELECT hire_date, (hire_date-1) FROM employees.` This syntax is not supported in Snowflake. The `DATEADD` and `TIMESTAMP` add functions are used to add or subtract units from dates or timestamps in Snowflake.

Updating data through a view

Oracle allows inserts, updates, and deletes to be executed against a view, which will then update the underlying table. In Snowflake, you must execute inserts, updates, and deletes against a table and not executed against a view. Again, you may need to reengineer load processes to account for this.

ORACLE SYNTAX

Snowflake does not support the following Oracle SQL syntax for creating tables (DDL):

- `SEGMENT`
- `PCTFREE`
- `PCTUSED`
- `INITRANS`
- `MAXTRANS`
- `NOCOMPRESS`
- `LOGGING`
- `STORAGE`
- `TABLESPACE`
- `PARTITION`

STORED PROCEDURES

Snowflake does not support PL/SQL stored procedures. You can rewrite most stored procedures in Python, which is a more portable and extensible language.

SYNONYMS

Snowflake does not support synonyms.

APPENDIX E

COMPARING DATA FROM ORACLE WITH SNOWFLAKE

Use row counts and sums of numeric data to verify data from Oracle matches the data loaded into Snowflake. Or, get unique values from columns in Oracle and compare those unique values with Snowflake to ensure you've loaded all your data successfully.

For use cases where you need more in-depth validation add an MD5 hash to the data extracted from Oracle. Construct this MD5 hash using columns that won't change when you load data into Snowflake (for example, include key columns and attributes in

the hash, but exclude insert and update dates and 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 and compare it with the MD5 hash from Oracle. This allows you to compare the contents of the row based on the MD5 hash rather than comparing each column individually.

The table below shows example hash queries and their results:

| DATABASE | HASH QUERY | RESULTS |
|---------------------|--|----------------------------------|
| Oracle ¹ | select standard_hash ('Snowflake' 'v2','MD5') from dual; | 85004C51CBD03E81EC558E2EBD05A3C1 |
| | select lower(standard_hash ('Snowflake' 'v2','MD5')) from dual; | 85004c51cbd03e81ec558e2ebd05a3c1 |
| Snowflake | select standard_hash ('Snowflake' 'v2','MD5') from dual; | 85004c51cbd03e81ec558e2ebd05a3c1 |

¹ Oracle returns hash values in uppercase. MD5 hash values should always be lowercase. Use the Oracle lower function to correct for this.

APPENDIX F

ORACLE INSTANCE VS. SNOWFLAKE ACCOUNT

Similar to an Oracle instance, a Snowflake account encapsulates users, roles and databases.

| ORACLE | SNOWFLAKE | NOTES |
|------------|-------------|--|
| Instance | Account | A Snowflake account is created within a single cloud providers region, defines the Snowflake edition, controls authenticating user connections and encapsulates all costs associated with the platform. |
| User | User | Snowflake users are created and managed at the account level and are independent of database schema objects. |
| Role | Role | Object ownership and object access control are managed at the role level using a combination of discretionary access control (DAC) and Role-Based Access Control (RBAC). Access privileges are not granted directly to a user. |
| Database | Database(s) | A single Snowflake account supports the creation of an unlimited number of logical databases. |
| Tablespace | n/a | Snowflake's unique architecture eliminates the need to manage tablespaces as well as database files and block and extent sizing. |

APPENDIX F (CONT'D) ORACLE INSTANCE VS. SNOWFLAKE ACCOUNT

DDL & Procedural Languages

Snowflake SQL reserves all ANSI keywords (with the exception of type keywords such as CHAR, DATE, DECIMAL, etc.), as well as some additional keywords that are reserved by Oracle and other popular databases. DDL operations such as CREATE, DROP and ALTER within Snowflake will, in many cases, be the same or very similar to their ANSI counterparts within Oracle.

| ORACLE | SNOWFLAKE | NOTES |
|-------------------------|------------------------|--|
| Schemas | Schemas | Snowflake schema objects are created and managed independent of a user login. |
| Tables | Tables | Snowflake supports permanent, transient, temporary and clustered tables. External tables are not currently supported within Snowflake. |
| Table Partitions | n/a | Snowflake's unique architecture eliminates the need to manage physical table partitions. |
| Constraints | Constraints | Snowflake provides support for constraints as defined in the ANSI SQL standard, as well as some extensions for compatibility with other databases, such as Oracle. Snowflake supports defining and maintaining constraints, but does not enforce them, except for NOT NULL constraints, which are always enforced. |
| Indexes | n/a | Snowflake's unique architecture eliminates the need to manage indexes. |
| Views | Views | A Snowflake view can be created for any valid SELECT statement. |
| Materialized Views | Materialized Views | Materialized views are supported in Enterprise Edition and above. |
| Transactions | Transactions | A Snowflake transaction is a set of SQL statements, both reads and writes, that are processed as a unit and guarantee ACID properties. |
| PL/SQL and Java | JavaScript | Stored procedures and user defined functions (UDF) within Snowflake utilize JavaScript as their procedural language. |
| PL/SQL Anonymous Blocks | n/a | Anonymous block JavaScript is not currently supported within Snowflake. |
| Packages | n/a | Packages are not currently supported within Snowflake. |
| Stored Procedures | Stored Procedures | Snowflake stored procedures utilize JavaScript as the procedural language. |
| User-Defined Functions | User-Defined Functions | A UDF can contain either a SQL expression or JavaScript code, and can return either scalar or tabular results (i.e., table functions). |
| Sequences | Sequences | Sequences can be used for generating sequential, unique numbers. |
| Synonyms | n/a | Synonyms are not currently supported within Snowflake. |

APPENDIX F (CONT'D)

ORACLE INSTANCE VS. SNOWFLAKE ACCOUNT

DML

Snowflake supports standard SQL, including a subset of ANSI SQL:1999 and the SQL:2003 analytic extensions. Snowflake also supports common variations for a number of commands where those variations do not conflict with each other.

Snowflake SQL reserves all ANSI keywords (with the exception of type keywords such as CHAR, DATE,

DECIMAL, etc.), as well as some additional keywords (ASC, DESC, MINUS, etc.) that are reserved by Oracle and other popular databases. Additionally, Snowflake reserves keywords REGEXP and RLIKE (which function like the ANSI reserved keyword LIKE) and SOME (which is a synonym for the ANSI reserved keyword ANY).

| ORACLE | SNOWFLAKE | NOTES |
|-------------------|-------------------|---|
| ANSI SQL | ANSI SQL | ANSI compliant SQL:1999 and SQL:2003 will transfer to Snowflake with little to no modification if schema, table and column names remain the same. |
| SQL Functions | SQL Functions | Snowflake supports a wide range of scalar, aggregate and window functions. |
| SQL Format Models | SQL Format Models | Snowflake supports a wide range of standard SQL format models for converting numeric and date values to text and vice versa. |
| Hints | n/a | Snowflake's unique architecture is optimized for data warehousing and eliminates the need for fined-grained query plan tuning via hints. |
| CONNECT BY | n/a | A CONNECT BY sub-clause in a FROM clause can join a table to itself as many times as necessary to process hierarchical data in that table. |
| MODEL | n/a | MODEL is not currently supported within Snowflake. |

ABOUT SNOWFLAKE

The Snowflake Cloud Data Platform shatters the barriers that prevent organizations from unleashing the true value from their data. Thousands of customers deploy Snowflake to advance their businesses beyond what was once possible by deriving all the insights from all their data by all their business users. Snowflake equips organizations with a single, integrated platform that offers the only data warehouse built for any cloud; instant, secure, and governed access to their entire network of data; and a core architecture to enable many other types of data workloads, including a single platform for developing modern data applications. Snowflake: Data without limits. Find out more at [Snowflake.com](https://www.snowflake.com).

