snowflake®

MICROSOFT SQL SERVER TO SNOWFLAKE
MIGRATION REFERENCE MANUAL

CHAMPION GUIDES

TABLE OF CONTENTS

2 Introduction

3 Preparing for the migration

8 Executing the migration

13 Migration success factors

14 Need help migrating?

15 Appendix A—Microsoft SQL Server to Snowflake Feature Mapping

21 Appendix B—Other Known Migration Issues

22 Appendix C—Comparing Data from Microsoft SQL Server to Snowflake

22 Appendix D—References

23 About Snowflake

INTRODUCTION

This document provides the high-level methodology needed to prepare for and execute the migration of an existing Microsoft SQL Server deployment to Snowflake. The appendices at the end of this document list differences between Microsoft SQL Server and Snowflake that you should consider as part of the migration.

The intended audience of this document includes the solution architects, program managers, and Snowflake solution partners that need a clearly defined approach for migrating an existing Microsoft SQL Server 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.

IDENTIFYING APPROPRIATE USE

Snowflake, at the core of the platform, is an ANSI SQL relational database built for analytical queries. Therefore, OLAP workloads are ideal for migrating to Snowflake. As an example, Snowflake works best as an operational data store (ODS) and data warehouse. Its unique architecture enables processing of huge quantities of data, but it creates challenges when organizations migrate highly transactional, high capacity workloads. Snowflake is an ever-evolving platform with enhanced capabilities for high concurrency, low latency workloads and point read searches through its search optimization service. However, for optimal success and to ensure you realize the value that Snowflake enables, be sure to identify the appropriate workload to migrate. Contact your local Snowflake representative to discuss appropriate workloads.

DOCUMENT THE EXISTING SOLUTION

Key outcomes:

- List of Microsoft SQL Server databases to migrate
- List of Microsoft SQL Server database objects to migrate
- List of Microsoft SQL Server schemas to migrate
- List of processes and tools that populate and pull data from Microsoft SQL Server
- List of security roles, users, and permissions
- List of Snowflake accounts that exist or need to be created
- Frequency of security provisioning processes
- Documentation of the existing Microsoft SQL Server solution into an as-is architecture diagram

Begin preparing for your migration from Microsoft SQL Server to Snowflake by determining which databases within the Microsoft SQL Server need to be migrated and which ones don't. Then determine which schemas to migrate.

Identify and document the objects within the Microsoft SQL Server databases to migrate. Include the size of the data to establish the scope of the migration project. Plan not to migrate Microsoft SQL Server catalog tables and views such as sys.tables (tables that have sys as a prefix) since they are catalog (metadata or directory) objects and aren't needed in Snowflake.

If you aren't sure which databases and database objects to migrate, reference the Microsoft documentation, here. Avoid moving unused objects, unless you need them for audit or historical purposes.

After you've identified the Microsoft SQL Server databases and database objects to migrate, evaluate each data source to determine whether the data comes from on-premises or a cloud-based source. This will help determine the methods available for loading the data into Snowflake. Specifically, will you need to load terabytes or even petabytes of on-premises data into Snowflake? If so, you may require capabilities such as AWS Snowball, Azure Data Box, or Google Transfer Appliance to move the data as efficiently as possible.

In addition to evaluating the data sources that populate Microsoft SQL Server, identify and document the processes and tools that move data in and out of Microsoft SQL Server.

Here are some examples:

- ETL/ELT tools
- Scripting languages
- Reporting/visualization tools
- Data science processes
- Machine learning processes

Use this list to evaluate the level of Snowflake support for the tools you currently use, and to help determine the migration approach that would best fit your needs.

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

If you do not have this information readily available, Snowflake Professional Services and/or 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

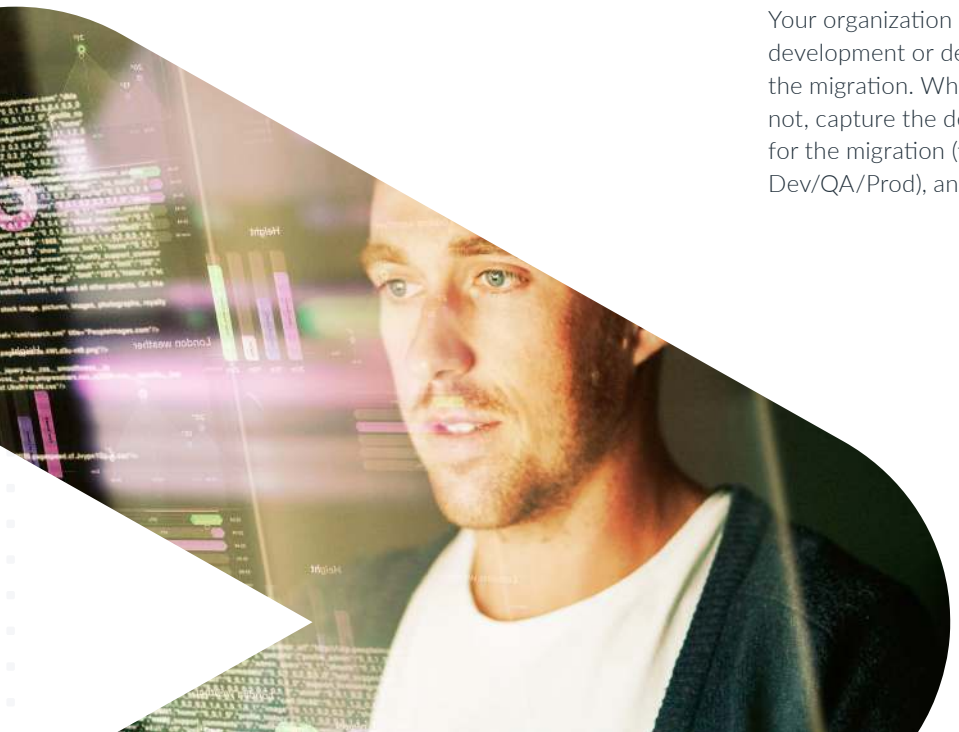
After you've documented your existing Microsoft SQL Server solution into an as-is architecture diagram, focus on your migration approach. Carefully consider how much reengineering you want to undertake as part of the migration. Organizations usually fall somewhere between wanting to migrate the existing solution as is and completely reworking the existing solution.

Snowflake recommends minimal reengineering for the first iteration unless your current system is broken. When you decide what you will reengineer, remember that changes to the existing data structures will impact downstream reporting and visualization tools. Also, more reengineering requires more development and testing, which extends the length of a migration project.

If you want to resolve issues with your existing implementation as part of the migration, include that information in your migration plan.

Break the migration into incremental deliverables that enable your organization to start making the transition to Snowflake faster. This will also provide value to your organization 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 organization.



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

Your organization may want to change your 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, Pre-Prod/Prod or Dev/QA/Prod), and the deployment processes used for the migration (such as the source control repository and the method for deploying changes from one environment to another). 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 for starting the migration provide value to the organization 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 of development and deployment processes from which to build the rest of the migration.

To prioritize data sets for migration, pay careful attention to understanding the process dependencies of the data sets. Document those dependencies. By identifying dependencies through a solid process before beginning the migration work, you will experience fewer challenges during the migration. If needed, you can engage Snowflake Professional Services and/or a Snowflake solution partner to help capture these dependencies.

Ideally, you can create this dependency documentation using an automated process that iterates through the existing job schedules and captures the data within Snowflake. This eliminates having to depend on manual investigation. Using an automated process pays dividends throughout the migration project by updating dependency documentation as changes take place. This is important since the underlying systems will likely change 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 and the roles they will play. The documentation should include each team member's name, contact information, and role. Team members may come from your team, Snowflake staff, or a Snowflake solution partner.

Some of the obvious roles required for a migration are developer, quality assurance engineer, business owner, project manager, program manager, scrum master, and communication specialist.

Snowflake Professional Services and/or a Snowflake solution partner can fill multiple needs for a migration, including solution design, requirements gathering, documentation, development, testing, delivery, 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 migration plan and budget required for the migration project
- Template of estimated costs to run Snowflake

Organizational expectations for migration deadlines are important planning inputs. In addition, consider other information such as the budget, resource availability, and amount of reengineering required. By gathering all of this information, you can establish and communicate achievable deadlines, even if the deadlines differ from what was originally expected.

It's critical to create a migration plan to understand the budget required to complete the migration. Snowflake Professional Services and/or a Snowflake solution partner can help create an end-to-end plan that includes an estimate of the migration costs and timeline. In addition, they can provide code conversion services to help accelerate the migration and reduce the overall costs. Compare the scope and costs of the migration to the available budget to ensure you have sufficient resources to complete the migration work.

A key input into the budget is the number of Snowflake compute clusters (also called virtual warehouses) required to support the migration and the number of compute clusters needed after the migration is completed. A Snowflake representative can provide a template and work with you to

determine how many compute clusters are needed. 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. Once you complete the template, you will get an estimated annual cost.

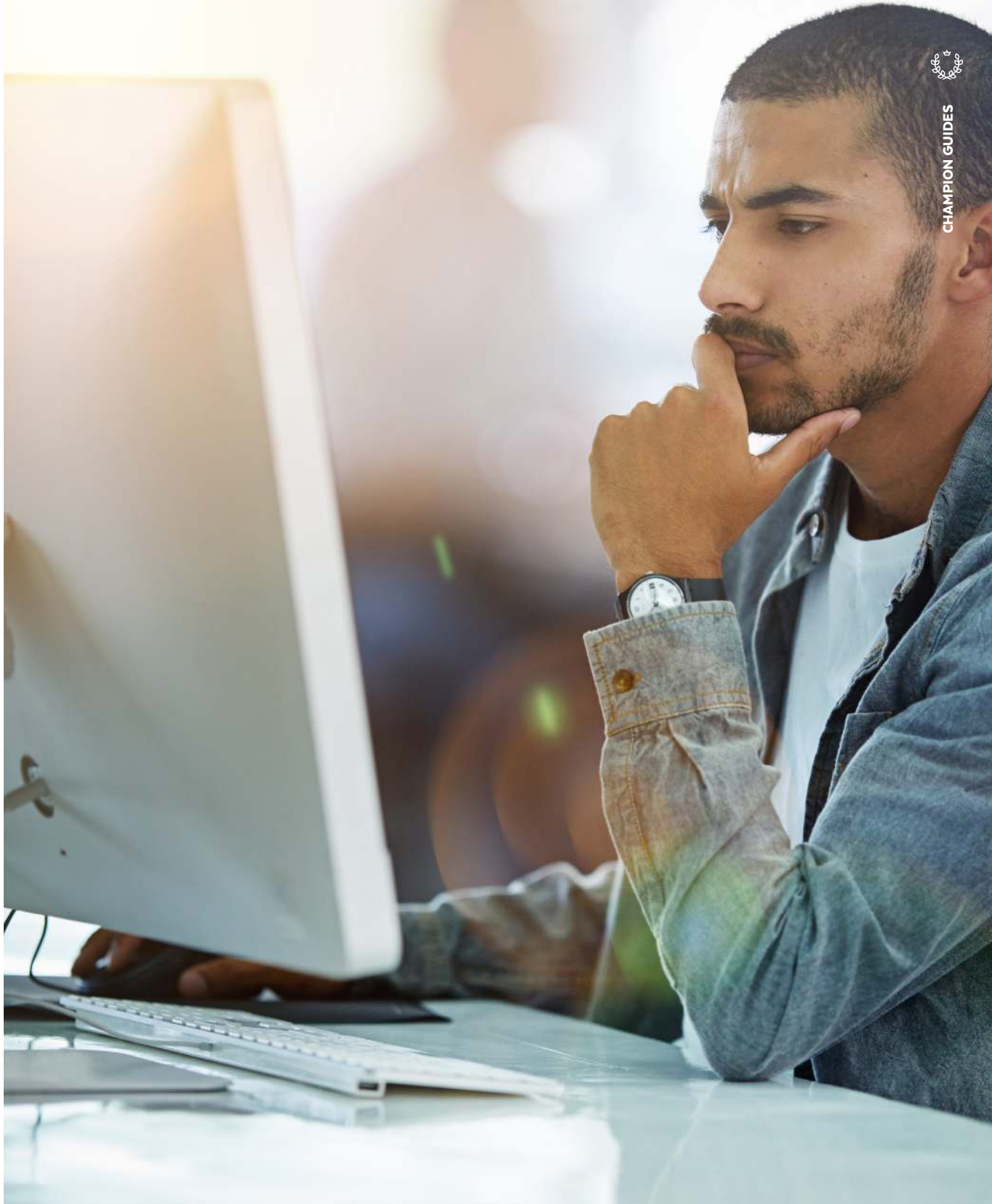
DETERMINE THE MIGRATION OUTCOMES

Key outcomes:

- List of high-level outcomes once the migration is completed
- Documented plan for communicating the migration project wins 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. (For example, turning off a Microsoft SQL Server could be one of your desired outcomes.) Use this documentation to validate that the migration project provides the overall benefits stakeholders expect to achieve from the migration.

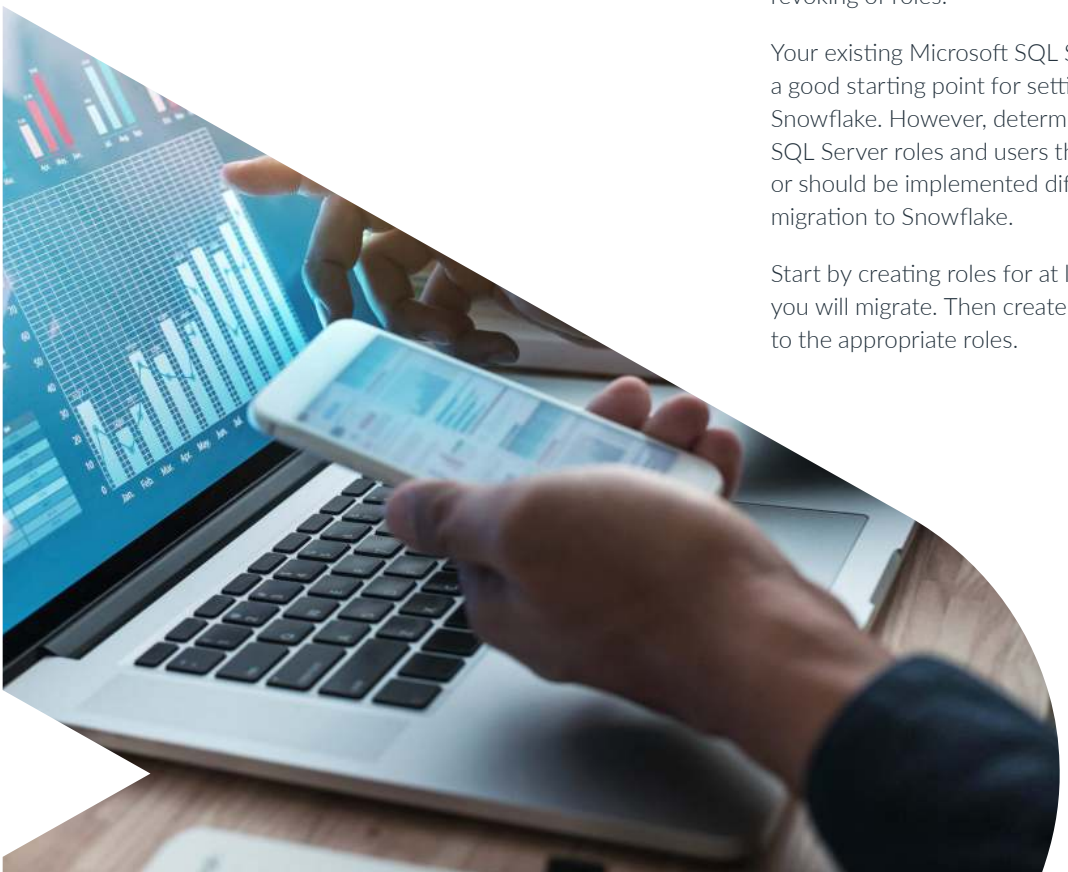
You can express this information as success or failure criteria for the migration project. The documentation can also include benchmarks that compare process execution on Microsoft SQL Server and Snowflake. After you compile this information, use it to communicate the success of the migration project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



EXECUTING THE MIGRATION

After you gather the information and decisions needed to prepare for the migration, you can execute the migration. This section guides you through the steps required.

If you need assistance with any part of executing the migration from Microsoft SQL Server to Snowflake, check with your Snowflake representative for Snowflake Professional Services and/or recommended Snowflake solution partners.



ESTABLISH SECURITY

When first setting up a Snowflake account, you can manually create users and roles to get started. From there, make it a priority to move to an automated process that creates users and assigns them to roles, and removes users when they are no longer applicable. Depending on your security auditing requirements, create processes to capture role and user creation and deletion, as well as the granting and revoking of roles.

Your existing Microsoft SQL Server security can be a good starting point for setting up security within Snowflake. However, determine if there are Microsoft SQL Server roles and users that are no longer needed or should be implemented differently as part of your migration to Snowflake.

Start by creating roles for at least the first data sets you will migrate. Then create users and assign them to the appropriate roles.

You can establish common roles for developer access, including read-only access, read and write access, and administrative access, for nonproduction databases. You may require 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, you might set it up so schedules are executed in QA and Prod but not in Dev, or data comparisons between Microsoft SQL Server and Snowflake occur only for Prod.) Automate testing as much as possible, so it's repeatable and enables you to identify any issues. Define, document, and get agreement on acceptance criteria for the tests.

PREPARE SNOWFLAKE FOR LOADING

There are a couple of options for setting up your Snowflake implementation, depending on the number of Snowflake accounts you have.

- When you have one Snowflake account, "follow these steps:

  1. Create a Snowflake database for the combination of the Microsoft SQL Server environment and database that you need to migrate (for example, Dev_Sales/QA_Sales/Prod_Sales).
  2. Create schemas in Snowflake that match the schemas from Microsoft SQL Server for each schema you intend to migrate.
  3. Create a Snowflake database for each Microsoft SQL Server environment (such as Dev/QA/Prod) that you need to migrate.
  4. Create a Snowflake database for each Microsoft SQL Server database.
  5. Create schemas for each of the Microsoft SQL Server databases you intend to migrate to Snowflake.

  This approach clearly identifies the environment and database in the database name and uses schemas to contain the tables and views, so it's easier to point tools from Microsoft SQL Server to Snowflake. Be aware that since the Snowflake

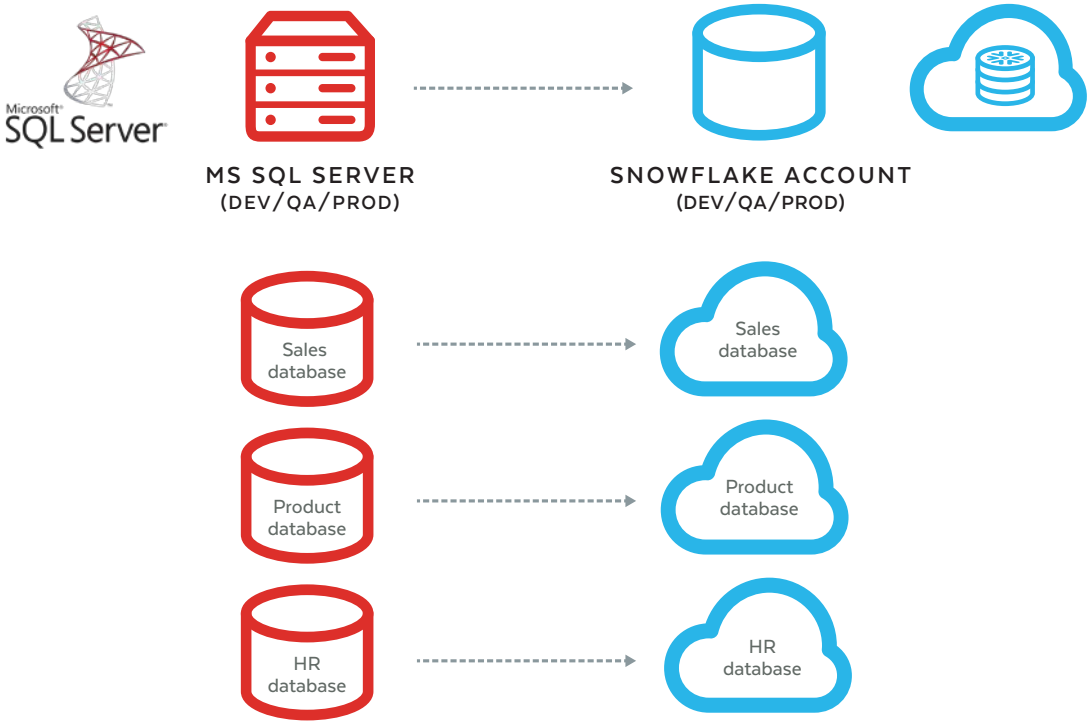


Figure 1: Migrating a Microsoft SQL Server environment and database to Snowflake

database name contains the environment in its name, you will need to update any views that reference a database as the view is deployed from one environment to another (for example, deploying from QA to Prod).

- When you have multiple Snowflake accounts, you can create the Snowflake databases and schemas to match the Microsoft SQL Server databases and schemas for a single environment. Figure 1 provides examples of migrating a Microsoft SQL Server environment to a Snowflake account database and Microsoft SQL Server databases to Snowflake databases.

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

Create the compute clusters (virtual warehouses) based on the information captured during the migration preparation. There should be separate compute clusters for each environment (such as Dev/QA/Prod) and for each function that the compute clusters will support (such as ETL/ELT or reporting and visualization). Figure 2 contains a reference architecture for using compute clusters for different workloads.

Base the initial sizing of the compute clusters (virtual warehouses) on the estimates you made while preparing for the migration. Then adjust the size as needed throughout the migration. Also, set up resource monitors to track usage and take appropriate action when limits are reached.

As you create the databases, database objects, and compute clusters, assign the appropriate security roles.

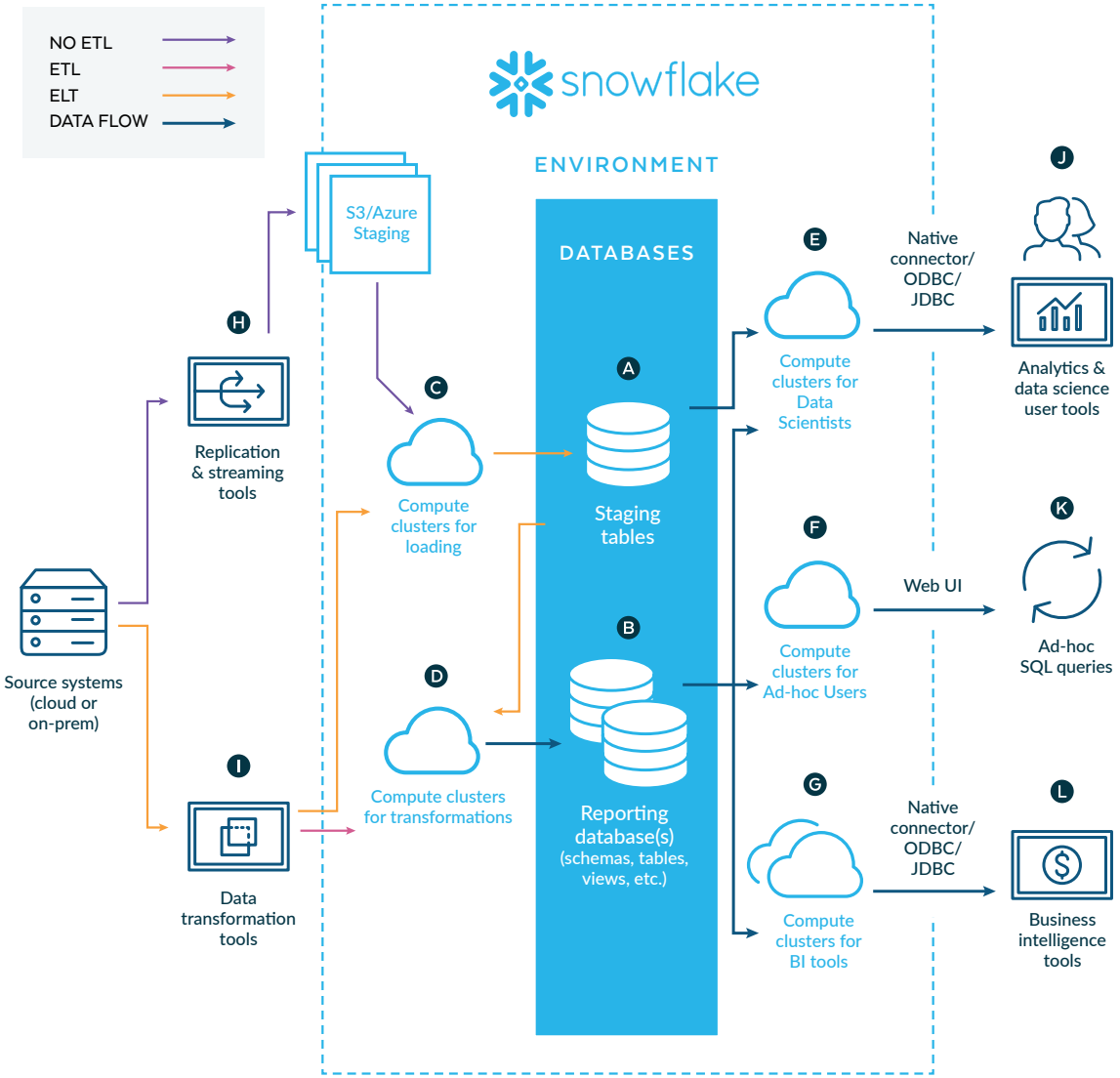


Figure 2: Using compute clusters for different workloads

LOAD INITIAL DATA SETS

You may require AWS Snowball, Azure Data Box, or Google Transfer Appliance if your Microsoft SQL Server is on-premises and 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.

You can load data into Snowflake after you've extracted the data from Microsoft SQL Server and moved the data to the cloud. Use this data loading to test the configuration of the databases and database objects, compute clusters, and the security you have implemented. See the Snowflake documentation for more information.

Depending on which Microsoft SQL Server environment the data came from and which Snowflake database you populate, you could use cloning to move data within Snowflake from one database to another. Cloning in Snowflake doesn't require additional storage. Therefore, you avoid the challenges and costs of loading the same data multiple times into different Snowflake databases.

Plan to extract data from Microsoft SQL Server and load it into Snowflake more than once. Also, begin with a subset of the data rather than trying to load the entire contents of the Microsoft SQL Server at the beginning of the migration.

KEEP DATA CURRENT

Implement processes to keep the data current after you load the historical data sets from Microsoft SQL Server into Snowflake.

You can set up data loading schedules that parallel the existing Microsoft SQL Server loading processes, or you can create new processes for loading data into Snowflake. This is another opportunity to evaluate whether changes to the schedule would be beneficial and should be part of the migration.

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

Along with scheduling the processes to run, monitor those processes so you can clearly understand and communicate the state of the data (for example, loading is in progress, loading completed successfully, or loading failures occurred). You can verify whether SLAs are being met within Snowflake, or identify and resolve process issues.

IMPLEMENT THE TEST PLAN

Begin the Snowflake implementation with the initial data sets loaded and processes running to keep the data current. Then you can start testing your Snowflake implementation. Be sure to engage the team members you identified when you prepared for the migration. After you complete initial testing and validate the data is ready for further scrutiny, engage additional groups to test their data sets and applications against Snowflake.

Compare data between the Microsoft SQL Server and Snowflake environments throughout the migration. If there are data differences, investigate to determine the cause and resolution.

Part of your migration may include fixing processes known to be incorrect in Microsoft SQL Server. Therefore, the test results may not match Snowflake, so use other methods to validate that the data is correct in Snowflake. Document why data won't match and share the information with the groups performing the testing so that 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 Microsoft SQL Server to Snowflake.

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 Teradata to Snowflake.

RUN MICROSOFT SQL SERVER AND SNOWFLAKE IN PARALLEL

During the migration, run the Microsoft SQL Server and Snowflake systems in parallel. Minimize the time you have both systems running, but make sure it's long enough to validate you've completed the migration successfully before shutting down Microsoft SQL Server.

Consider how to best run Microsoft SQL Server and Snowflake in parallel to compare data and performance. For example, you may need to create hashes as you extract data from Microsoft SQL Server in order to compare data at the row level between Microsoft SQL Server and Snowflake (See Appendix C). Perform these comparisons in Snowflake to keep from negatively impacting your Microsoft SQL Server.

REDIRECT TOOLS TO SNOWFLAKE

After you've migrated a sufficient amount of data for each tool you identified while preparing for the migration, redirect the tool connections to Snowflake.

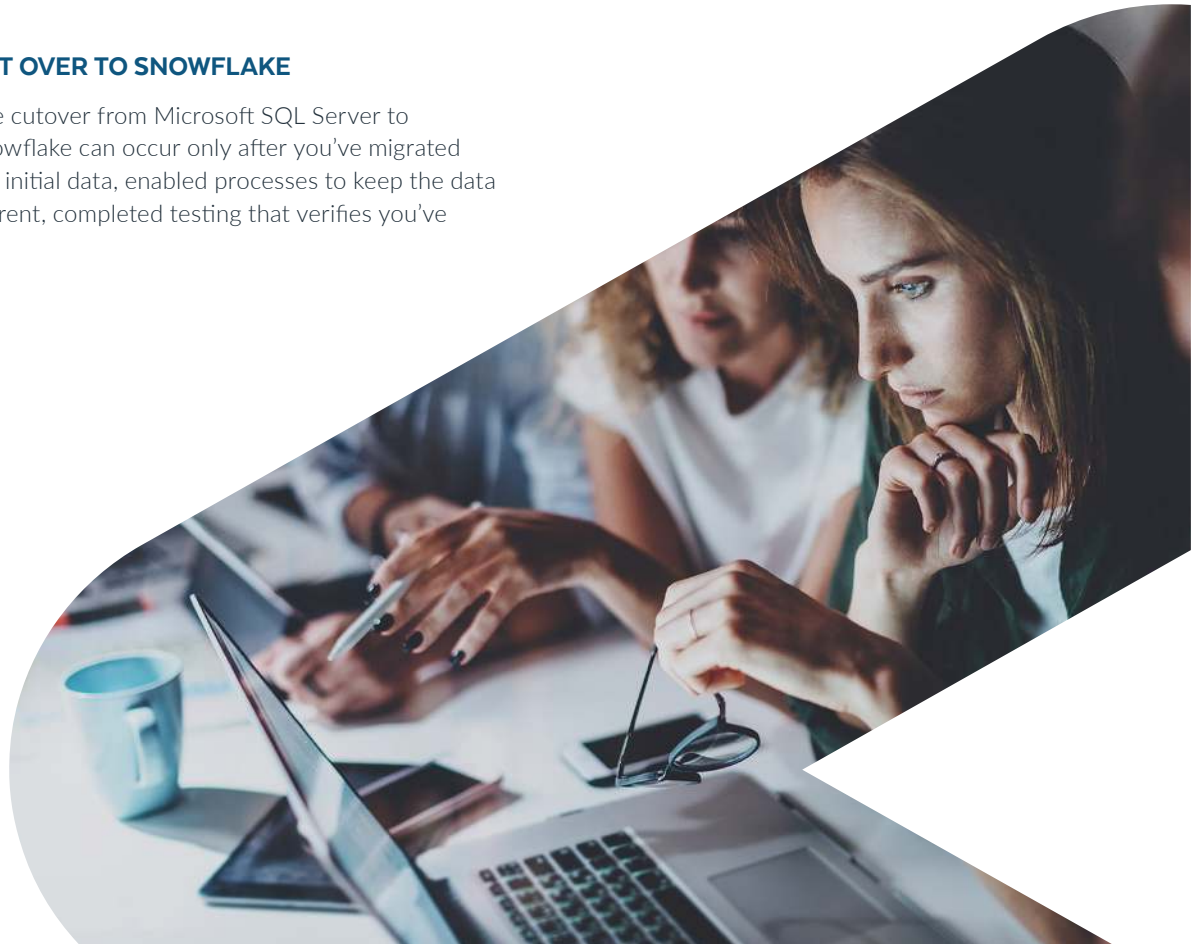
This usually involves copying and updating the existing solution to connect to Snowflake instead of Microsoft SQL Server. Compare the tools' output to ensure the results are the same between the two systems. In addition, evaluate the performance of the tools to verify they are performing as expected in Snowflake.

CUT OVER TO SNOWFLAKE

The cutover from Microsoft SQL Server 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 Microsoft SQL Server to Snowflake.

Make sure you've planned and communicated the cutover date in advance to your Microsoft SQL Server users. In addition, make sure they can log into Snowflake and run the redirected tools they depend on.

To complete the cutover, turn off data processes that populate Microsoft SQL Server. In addition, revoke access to Microsoft SQL Server so users and tools no longer have access.



MIGRATION SUCCESS FACTORS

Paying attention to certain success factors will reduce risk and enable you to successfully complete the migration. This section provides insight into how to increase the speed of migration and ensure the migration from Microsoft SQL Server to Snowflake is successful.

IDENTIFY AND MITIGATE DIFFERENCES BETWEEN MICROSOFT SQL SERVER AND SNOWFLAKE

Use Appendix A early in the migration process to identify differences between Microsoft SQL Server and Snowflake. Present these differences to the organization along with strategies for handling the differences. Then, confirm that your proposed approach will meet their requirements.

RESOLVE MIGRATION ISSUES

There will inevitably be issues that occur during and after the migration. Establish processes to document and escalate migration issues so you can resolve them as quickly as possible.

For each escalation, document the issue, who is responsible for working on the issue, who is responsible for communicating progress, and a list of contacts (include contacts from your organization, from Snowflake, and from any other parties involved). Be sure everyone involved can log a support ticket in the Snowflake Community portal. Likewise, they should know how to ask questions and find resources in the Snowflake Community forum and on Stack Overflow.

Establish a regular cadence for reviewing documented issues and getting an updated status on resolving each issue.

You may also identify issues during the migration that can be resolved after the migration. Document and prioritize them, so you can work on them post-migration.

COMMUNICATE MIGRATION BENEFITS

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

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 Professional Services Team deploys a powerful combination of data architecture experience and advanced technical knowledge of the platform to deliver high performing data strategies, proofs of concept, and migration project planning and implementation.

Snowflake's 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 Professional Services and solution partners have the skills and tools to accelerate your journey to cloud-built data analytics, so you can quickly reap the full benefits of Snowflake.

To find out more, contact the Snowflake sales team or visit the Snowflake Community.

APPENDIX A: MICROSOFT SQL SERVER TO SNOWFLAKE FEATURE MAPPING

UTILITIES

SQL SERVER	SNOWFLAKE	COMMENTS
MSSQL-CLI	SnowSQL	SnowSQL is the next-generation command-line client for connecting to Snowflake to execute SQL queries and perform all DDL and DML operations, including loading data into and unloading data out of database tables.
BCP (Bulk Copy Program)	COPY <INTO>	COPY INTO is a command executed within SnowSQL or any other client connected to Snowflake and can ingest AVRO, Parquet, ORC, JSON, XML, and flat delimited text files.
MS	INTERVAL MINUTE TO SECOND	INTERVAL data types aren't supported in Snowflake but date calculations can be done with the date comparison functions (e.g. DATEDIFF and DATEADD).

DATA TYPES

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

SQL SERVER (TSQL)	SNOWFLAKE	COMMENTS
BIGINT	NUMBER	Precision and scale not to be specified when using Numeric.
BIT	BOOLEAN	Recommended: Use NUMBER if migrating value-to-value to capture the actual BIT value. Use Boolean in Snowflake if the desired outcome is to restrict to Ternary Logic (three valued): TRUE, FALSE, or NULL (UNKNOWN).
DECIMAL	NUMBER	Default precision and scale are (38,0).
INT	NUMBER	Precision and scale not to be specified when using Numeric.
MONEY	NUMBER	Money has a range of 19 digits with a scale of 4 digits, so NUMBER(19,4) can be used.
NUMERIC	NUMBER	Default precision and scale are (38,0).

APPENDIX A: MICROSOFT SQL SERVER TO SNOWFLAKE FEATURE MAPPING

DATA TYPES *(cont'd)*

SQL SERVER (TSQL)	SNOWFLAKE	COMMENTS
SMALLINT	NUMBER	Default precision and scale are (38,0).
SMALLMONEY	NUMBER	NUMBER with precision of 10 digits, with a scale of 4, so NUMBER(10,4) can be used.
TINYINT	NUMBER	Default precision and scale are (38,0).
FLOAT	FLOAT	Snowflake uses double-precision (64-bit) IEEE 754 floating point numbers.
REAL	FLOAT	The ISO synonym for REAL is FLOAT(24).
DATE	DATE	Default in SQL Server is YYYY-MM-DD.
DATETIME2	TIMESTAMP_NTZ	Snowflake: TIMESTAMP with no time zone, time zone is not stored. DATETIME2 has a default precision of up to 7 digits, Snowflake has TIMESTAMP_NTZ with the precision of 9 digits.
DATETIME	DATETIME	SQL Server datetime is not ANSI or ISO 8501 compliant. Storage size is 8 bytes. Accuracy is rounded to increments of .000, .003, or .007 seconds.
DATETIMEOFFSET	TIMESTAMP_LTZ	Up to 34,7 in precision, scale.
SMALLDATETIME	DATETIME	SMALLDATETIME is not ANSI or ISO 8601 compliant. It has a fixed 4 bytes storage space.
TIMESTAMP (Unsupported)	TIMESTAMP_NTZ	Use DATETIME2 or CURRENT_TIMESTAMP function.
TIME	TIME	SQL Server has a precision of 7 nanoseconds. Snowflake has precision of 9 nanoseconds.

APPENDIX A: MICROSOFT SQL SERVER TO SNOWFLAKE FEATURE MAPPING

DATA TYPES *(cont'd)*

SQL SERVER (TSQL)	SNOWFLAKE	COMMENTS
CHAR	VARCHAR(1)	Any set of strings that is shorter than the maximum length is not space-padded at the end.
TEXT	VARCHAR	This data type will be discontinued on SQL Server. Use NVARCHAR, VARCHAR, or VARBINARY instead.
VARCHAR	VARCHAR	Any set of strings that are shorter than the maximum length is not space-padded at the end.
NCHAR	VARCHAR	NCHAR is used on fixed-length-string data.
NTEXT	VARCHAR	This data type will be discontinued on SQL Server. Use NVARCHAR, VARCHAR, or VARBINARY instead.
NVARCHAR	VARCHAR	NVARCHAR's string length can range from 1–4000.
BINARY	BINARY	Snowflake: maximum length is 8 MB.
IMAGE	N/A	This data type will be discontinued on SQL Server. Use NVARCHAR, VARCHAR, or VARBINARY instead.
VARBINARY	BINARY	Snowflake: maximum length is 8 MB.
UNIQUEIDENTIFIER	N/A	Not Supported.

APPENDIX A: MICROSOFT SQL SERVER TO SNOWFLAKE FEATURE MAPPING

SQL SERVER INSTANCE VERSUS SNOWFLAKE ACCOUNT

Similar to SQL Server instance, a Snowflake account encapsulates users, roles, and databases..

SQL SERVER	SNOWFLAKE	COMMENTS
Instance	Account	A Snowflake account is created within a single cloud provider region, and it 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. SQL Server offers a similar model with a user setup and is assigned a login but the user may not be able to use or access the databases.
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. SQL Server has a similar model of having set role permissions with the user being assigned to a role. Access privileges are not granted directly to a user, even for logging into SQL Server.
Databases	Databases	A single Snowflake account supports the creation of a soft limit of 10,000 logical databases.

DDL AND PROCEDURAL LANGUAGES

Snowflake SQL reserves all ANSI keywords (with the exception of type keywords such as CHAR, DATE, and DECIMAL), as well as some additional keywords that are reserved by SQL Server 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 SQL Server.

SQL SERVER	SNOWFLAKE	COMMENTS
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. Snowflake enforces only the NOT NULL constraints. SQL Server enforces NOT NULL constraints as well as referential integrity constraints.
Indexes/ Partitioned Indexes	n/a	Snowflake's unique architecture eliminates the need to manage indexes.

APPENDIX A: MICROSOFT SQL SERVER TO SNOWFLAKE FEATURE MAPPING

DDL AND PROCEDURAL LANGUAGES *(cont'd)*

SQL SERVER	SNOWFLAKE	COMMENTS
Views	Views	A Snowflake view can be created for any valid SELECT statement.
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.
CLR/TSQL	JavaScript	Stored procedures and user defined functions (UDF) within Snowflake utilize JavaScript as their procedural language.
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 (such as table functions).
Unsupported	Sequences	Sequences can be used for generating sequential, unique numbers.
Indexed Views	Materialized Views	A materialized view is a precomputed data set derived from a query specification (the SELECT list in the view definition) and stored for later reuse. In SQL Server, Indexed Views uses the syntax SCHEMABINDING.

DML

Snowflake supports standard SQL, including a subset of ANSI SQL:1999 and the SQL:2003 analytic extensions. Snowflake also supports common variations for several commands where those variations do not conflict with each other. Snowflake SQL reserves all ANSI keywords (with the exception of data type keywords such as CHAR, DATE, and DECIMAL). Snowflake supports some additional keywords (such as ASC, DESC, or MINUS) that are reserved by SQL Server 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).

SQL SERVER	SNOWFLAKE	COMMENTS
ANSI SQL	ANSI SQL	ANSI-compliant SQL 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 (such as Lead/Lag).
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 fine-grained query plan tuning via hints.

APPENDIX A: MICROSOFT SQL SERVER TO SNOWFLAKE FEATURE MAPPING

DATE CONSIDERATIONS

Unlike SQL Server, in Snowflake, there is no GETDATE() function, so you cannot insert the value of CURRENT_TIMESTAMP() into a column with DATETIME datatype. This is because, in Snowflake, DATETIME is an alias for TIMESTAMP_NTZ but CURRENT_TIMESTAMP returns TIMESTAMP_LTZ, which is incompatible, as detailed below:

SQL SERVER	SNOWFLAKE	COMMENTS
GETDATE()	TIMESTAMP_NTZ	Snowflake's TIMESTAMP_NTZ is similar to TSQL Datetime. Use ALTER SESSION SET TIMESTAMP_TYPE_MAPPING=TIMESTAMP_LTZ;
TO_DATE	TO_DATE	Behavior within Snowflake will be similar to SQL Server.
DATEDIFF	DATEDIFF	Behavior within Snowflake will be similar to SQL Server.
DATEADD	DATEADD	Behavior within Snowflake will be similar to SQL Server.

APPENDIX B: OTHER KNOWN MIGRATION ISSUES

CASE SENSITIVITY

Since Snowflake is case sensitive (for example, Glass, GLASS, and glass are three different values), during the migration, check for comparison issues in queries. Microsoft SQL Server is case insensitive by default, however, it is possible to create a case-sensitive SQL Server database and even to make specific table columns case sensitive. Determine if a database or database object is case sensitive by checking its COLLATION property and look for CI or CS in the result. Run SELECT name, description FROM sys.fn_helpcollations(), to illustrate all collations supported by the SQL Server installation. Another simple solution to this issue is to use UPPER on both sides of a comparison, for example, WHERE UPPER(COLUMNNAME)=UPPER(COLUMNNAME), if you want to ignore any differences in case.

CONSTRAINTS

Microsoft SQL Server enforces Primary Key and Foreign Key constraints. While Snowflake supports the syntax to define Primary Keys and Foreign Keys, they aren't enforced within Snowflake. This means you'll need to reengineer load processes that depend on constraints to prevent duplicate entries and orphaned records from being entered into the data warehouse.

DATE VERSUS TO_DATE()

Microsoft SQL Server has the capability to put DATE in front of a string in order to treat it as a date value (for example, DATE '2018-12-31'). In Snowflake, the syntax is TO_DATE(), for example, TO_DATE('2018-12-31'). It isn't necessary to use DATE or TO_DATE() in many situations, since both Microsoft SQL Server and Snowflake can interpret the date values stored in a string. When migrating SQL from Microsoft SQL Server to Snowflake, it may be more desirable to replace DATE with TO_DATE() rather than dropping DATE altogether.

UPDATING DATA THROUGH A VIEW

Microsoft SQL Server allows inserts, updates, and deletes to be executed against a view, which then updates the underlying table. In Snowflake, inserts, updates, and deletes must be executed against a table and can't be executed against a view. Again, load processes may need to be reengineered to account for this.

SYNTAX SPECIFIC TO SQL SERVER

SQL Server has SQL syntax that is not used in Snowflake:

- PARTITION BY
- COMPRESS/DECOMPRESS
- FORMAT
- INDEXES
- CURSOR
- HINTS
- FREETEXT
- BULK INSERT
- DISABLE/ENABLE TRIGGER
- COLLATE
- UPDATE STATISTICS (not needed in Snowflake as this is carried out as a service)
- ROLLBACK TRANSACTION (within Snowflake, it is ROLLBACK)

OTHER CONSIDERATIONS:

In Snowflake, variables can be initialized in SQL using the SET command. The data type of the variable is derived from the date type of the result of the evaluated expression. In Azure SQL Data Warehouse, DECLARE <variable> <date type> is used followed by setting the value to the variable.

SQL Server was built primarily for OLTP. Snowflake was built for OLAP.

Note that you do not need to define a schema in advance when loading JSON data into Snowflake.

For SQL Server Stored Procedures, use table functions if you are returning result sets from SQL stored procedures and Python if just executing business logic when using Snowflake.

Integration tables are not required as Snowflake is an ELT platform. Staging tables are not required.

APPENDIX C: COMPARING DATA FROM MICROSOFT SQL SERVER TO SNOWFLAKE

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

For use cases where you require more in-depth validation, add an MD5 hash to the data extracted from Microsoft SQL Server. 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/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 compare it with the MD5 hash from Microsoft SQL Server. This allows you to compare the contents of the row on the MD5 hash from Microsoft SQL Server with the MD5 hash from Snowflake, rather than comparing each column individually.

APPENDIX D: REFERENCES

SQL SERVER DOCUMENTATION

- Data types (Transact-SQL) - SQL Server
- Maximum capacity specifications for SQL Server - SQL Server

SNOWFLAKE DOCUMENTATION

- General Reference

ABOUT SNOWFLAKE

Snowflake's cloud data platform shatters the barriers that have prevented organizations of all sizes from unleashing the true value from their data. Thousands of customers deploy Snowflake to advance their organizations beyond what was 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 the cloud; instant, secure, and governed access to their entire network of data; and a core architecture to enable many types of data workloads, including a single platform for developing modern data applications. Snowflake: Data without limits. Find out more at **snowflake.com**.

Snowflake is FedRAMP Authorized