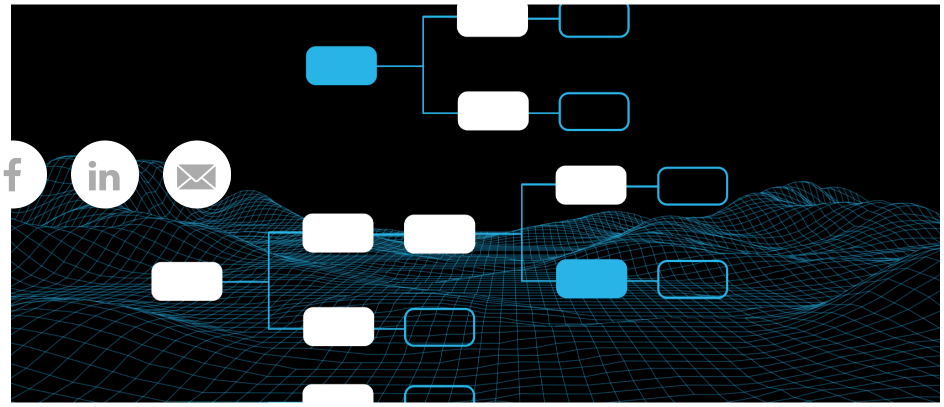
JUL 16, 2024

AUTHOR

Jakub Puchalski

Snowflake Brings Seamless PostgreSQL and MySQL Integration with New Connectors

SHARE



Using transactional data from OLTP databases, like PostgreSQL and MySQL, for machine learning and generative AI is becoming hypercritical. These databases store rich, real-time data that can be leveraged to train sophisticated AI models, enabling the extraction of valuable patterns and trends. PostgreSQL, for one, with its appealing advanced features and open source nature, has grown significantly in popularity. By applying AI solutions to this data, organizations can uncover deeper insights, optimize operations, predict future trends and enhance decision-making, ultimately leading to more data-driven business outcomes.

Snowflake's AI Data Cloud is a leading platform for storing, processing and analyzing data with cutting-edge AI capabilities. This raises the question of what is the most efficient way to ingest data from PostgreSQL and MySQL into Snowflake in order to mash it up with all other enterprise data and unlock its full potential. Well, the good news is that Snowflake has delivered a solution. Now it has never been easier to make your transactional data available for Enterprise AI in a low-latency streaming fashion.

The OLTP database connectors

AUTHOR

Jakub Puchalski

Building on the success of Snowflake Native Connectors — Snowflake Connector for Kafka and connectors for SaaS applications, like ServiceNow and Google Analytics — we have just announced a public preview of connectors for two of the leading open source relational databases, PostgreSQL and MySQL. These new database connectors are built on top of Snowpipe Streaming, which means they also provide more cost-effective and lower-latency pipelines for customers. They further our commitment to offering simple native connectors for change data capture (CDC) from the top online transaction processing (OLTP) database systems. We soon expect to expand the connectors roster to integrate with leading proprietary databases as well.

SHARE

At a high level, the Snowflake database connectors consist of two components:

- The **Universal Agent**: A standalone application distributed as a docker image, available on Docker Hub, which is deployed in the customer's infrastructure. It is responsible for reading the data from the source database. The Universal Agent's two main responsibilities are:
  - sending the initial snapshot load
  - reading incremental data changes from the source database CDC stream
- The **Snowflake Native App:** An object that resides in the customer's Snowflake account and is the brain behind the connector. It is primarily responsible for managing the replication process, controlling the agent state and creating all database objects, including the target database.

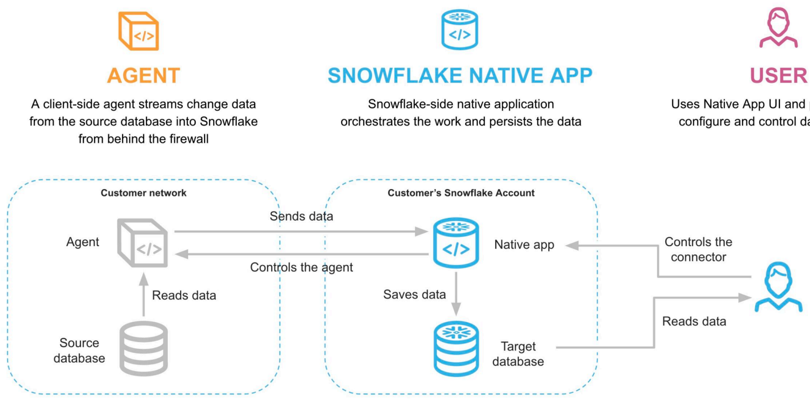

AGENT
A client-side agent streams change data from the source database into Snowflake from behind the firewall
SNOWFLAKE NATIVE APP
Snowflake-side native application orchestrates the work and persists the data
USER
Uses Native App UI and procedures to configure and control data ingestion
Customer network
Customer's Snowflake Account
Agent
Sends data
Controls the agent
Native app
Controls the connector
Reads data
Source database
Saves data
Target database
Reads data

Figure 1. Main components of the database connector

AUTHOR

Jakub Puchalski

Consider the example database connector configuration presented in Figure 2 (below). First, a Snowflake user installs two native Snowflake connector applications, found in the Snowflake Marketplace: one for MySQL and one for PostgreSQL.The two native apps are then configured in the user's Snowflake account, where they are pointed to their respective target databases. Then,

SHARE

[illegible] two copies of the Universal Agent inside their company network (or in the cloud — because the agent runs inside a docker container, it can be installed wherever the user wants). In this example, each agent will read data from two data sources, so it needs to receive two sets of source database credentials — one for each OLTP database server. Lastly, each agent must be connected to its respective Snowflake native app with an additional set of credentials (unique for each app). As a result, each source database (four in this example: two MySQL and two PostgreSQL) is mirrored with a target database in the user's Snowflake account. Now the connectors can begin replicating data for the selected source tables.

To summarize:

> A single agent can be connected to multiple source databases, but they have to be of one kind — either MySQL or PostgreSQL.
>
> A single native app can be connected to a single agent. There is a dedicated connector app for MySQL and a separate dedicated app for PostgreSQL.

A single native app can write to multiple Snowflake target databases — each represents a connected source database.

A user can choose which tables and columns are going to be replicated and at what interval — continuously or according to a predefined schedule (e.g., every six hours or every Thursday at 2 p.m.).

AUTHOR

Jakub Puchalski

(Coming soon) Multiple instances of the same native app type can be installed on a single Snowflake account, which is helpful in case you want to split the data pipeline for separate business units or different infrastructures (e.g., network-isolated PostgreSQL database instances).

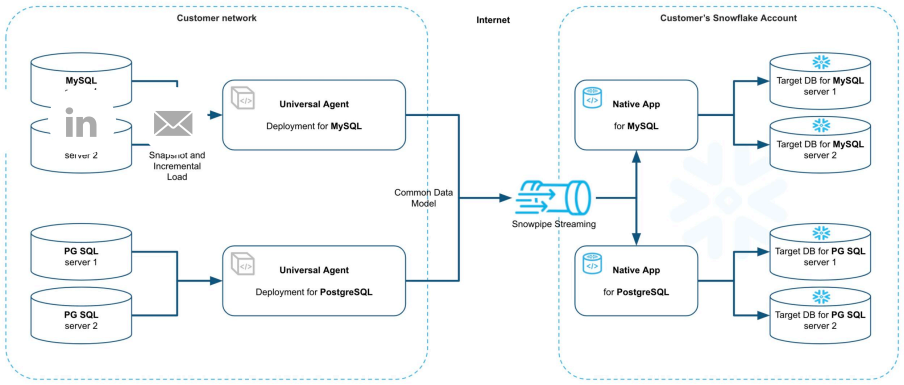

Customer network
Internet
Customer's Snowflake Account
MySQL
server 2
Snapshot and Incremental Load
Universal Agent
Deployment for MySQL
PG SQL server 1
PG SQL server 2
Universal Agent
Deployment for PostgreSQL
Common Data Model
Snowpipe Streaming
Native App for MySQL
Native App for PostgreSQL
Target DB for MySQL server 1
Target DB for MySQL server 2
Target DB for PG SQL server 1
Target DB for PG SQL server 2

Figure 2. Example database connector configuration

Splitting the solution into two — a native app and an agent — brings several benefits:

The agent can be deployed inside the customer's network and act as a gateway that ensures a secure connection between the source database, hidden behind the firewall, and the customer's Snowflake account.

The agent can be distributed as a single docker image and used for various source databases. Currently, it has extractors for PostgreSQL and MySQL, but soon it can be extended to other popular OLTP databases.

The agent is very easy to set up: You just pull it from Docker Hub; mount a database driver and configuration files, with source and target database credentials; and run it. It's portable across environments and operating systems.

The agent is stateless, so in case of failures like VM outages or network issues, the docker container can be restarted or even reinstalled in a new environment, and it will pick up the work where it left off because the native app would always know the state of the replication and instruct the agent what it needs to do.

AUTHOR

Jakub Puchalski

The native app stores the replication state inside the customer's Snowflake account, which ensures the security and reliability of the data pipeline.

SHARE

As a native application, the connector leverages Snowflake's sc[illegible]ilit[illegible] compute resources. Based on the data volume, number of tables and sync frequency, you can select different virtual warehouse sizes on demand to balance cost vs. throughput.

Moreover, on the security front, the data is transferred directly from your OLTP database instance to your Snowflake account and encrypted in transit. Because you control where the agent runs, no third-party infrastructure or processor is used in between. Once your data reaches its final destination, access to the ingested data is governed by Snowflake's built-in access controls.

Change Data Capture (CDC) and Snowpipe Streaming

Data ingestion is controlled inside the Snowflake native application. A user logs into their Snowflake account, opens the native app's UI or the Worksheets, and defines what tables, with what columns, they want to replicate and at what frequency (continuously or according to a schedule).

When a table is added for the first time, the native app instructs the connected agent to do several things (see Figure 3). First, the agent introspects the source table's schema so that the connector knows the data type of each column and how to map it to the destination table in Snowflake. Then it kicks off the initial load of the entire table's snapshot, which can take some time to complete. In parallel, it also kicks off the incremental sync. Due to the fact that the snapshot load and incremental sync of new changes are performed in parallel, the moment the initial load is completed, you already have the most up-to-date data in your destination table.

AUTHOR

Jakub Puchalski

The ongoing incremental updates use the Change Data Capture (CDC) technique to capture changes performed on the source database. The changes include INSERT, UPDATE and DELETE operations, along with DDL changes, which are automatically replicated on the destination database in Snowflake. The agent reads MySQL's binary log (binlog) and PostgreSQL's logical replication (Write-Ahead Log or WAL). Then it decodes data from database-specific format (which can be a low-level, binary format) to a Common Data Model. This is essentially type conversion and not data processing. Source-database SQL types and binary types retrieved from a CDC stream are converted to universal, JSON-friendly types.

SHARE

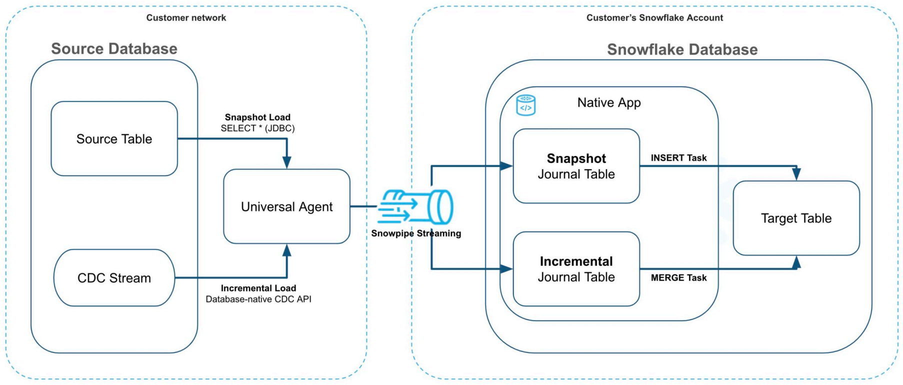


Figure 3. Database connector data flow

This is the moment when Snowpipe Streaming kicks in — the highly efficient, low-latency continuous data ingestion feature. The agent sends source data, converted to the Common Data Model events, to Snowflake using Snowpipe Streaming. On the other side of the pipe (inside your Snowflake account), the data is written to so-called journal tables. The journal tables are event-based and append-only. They store a sequence of row-level events (insert, update and delete) that happen in the source tables. The connector native app creates two journal tables for each source table — one for the snapshot load and another for the incremental load. The remaining step is to replicate the latest state of the source tables into their respective destination tables. For that, the native app uses two Tasks:

AUTHOR

Jakub Puchalski

The Insert Task is a stored procedure executed once by the native app, when the agent reports that it's done with the snapshot load. The procedure copies data from the snapshot journal table to the target table.

The Merge Task is a stored procedure executed periodically by the native app. In the incremental sync process, the procedure reads incoming events from the incremental journal table and merges them into the target table.

SHARE

ch a anges

The connector supports changes in a source table schema. If a column is removed or renamed, or a new column is added, the connector will reflect these changes in a destination table:

Adding a new column: The new column will be visible in the destination table, just like any other column that existed at the start of the replication.

Deleting an existing column: If a column is deleted in the source table it will not be deleted in the destination table. Instead, a soft-delete approach is followed, and the column will be renamed to '<previous name>__SNOWFLAKE_DELETED' so that historical values can still be queried. All rows replicated after the column is deleted will have a NULL value in this column. For example, if column A is deleted, it will be renamed 'A__SNOWFLAKE_DELETED' in the destination table, but its contents will remain unchanged.

Renaming a column: Renaming a column is equivalent to deleting it and creating a new one with a new name. The deletion follows the aforementioned soft-delete approach. For example, if column A was renamed to B in the source table, it will be renamed to ‘A__SNOWFLAKE_DELETED,’ and a new column B will be created in the destination table.

Monitoring and alerting

AUTHOR

Jakub Puchalski

Under the hood, the connector logic is built around a Hierarchical Finite State Machine. This brings a number of benefits, such as scalability (to larger workloads and volumes), extensibility (to new data sources and features), improved maintainability (testing and debugging), and — probably most importantly — excellent observability. Since all connector and data flow state transitions are clearly represented, it is easy to monitor the data ingestion process with a rich set of events and metrics.

SHARE

Out of the box, the connector native app offers a set of views that provide users with both aggregated and granular information about data replication at a data source or a table level. For example, for a given table, you can check its current replication phase (schema introspection, initial load or incremental sync) and the replication statistics (each micro batch start and finish time; duration per module, whether agent, Snowpipe or app; number of ingested rows; throughput; and latency). In case of ingestion errors, it is easy to understand at what stage things got stuck (e.g., source database connection lost). Optionally, you can enable email alerts, which will notify you in case an error occurs and requires your attention.

The savings of simplicity

Customers who bring their transactional data from the OLTP databases to Snowflake unlock a plethora of use cases, such as Customer Analytics (segmentation, Customer Lifetime Value, churn); Sales & Revenue Analysis (performance, forecasting); Inventory & Supply Chain Analysis (optimization, forecasting); Financial Analysis (profitability, cash flow); Fraud Detection & Security (anomaly detection); Marketing Analytics (campaign effectiveness, Customer Acquisition Cost); Compliance & Reporting (regulatory compliance) and many more.

StreetMetrics — a company specializing in the measurement and analytics of out-of-home (OOH) and transit advertising — was an early adopter of this tool. Josh Worthington, Senior Big Data Engineer at StreetMetrics, lauds the connector's ease of use. "The connector was a great replacement to a set of other clunky infrastructure," he says. "The basic functionality of replicating data was nice, but the added bonuses of extra monitoring and visibility made the tool even better and saved tons of engineering time. Beyond just being a valuable, easy-to-use tool, the team that supported it had always been responsive and had quickly added many features over the last few months."

AUTHOR

Jakub Puchalski

The connector is already driving value for its users, but this is just the beginning. We have a rich roadmap of feature enhancements, performance and scalability improvements in front of us, which I plan to reveal along the way, so stay tuned for more articles on the topic.

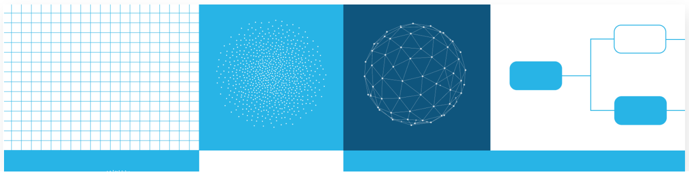
How to get started

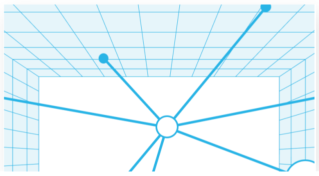
SHARE

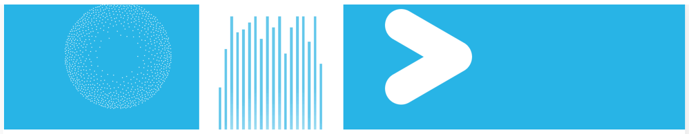
Install the Snowflake Connectors for PostgreSQL and MySQL from the Snowflake Marketplace and download the agent from the Docker Hub. They are available with no additional access fees set by Snowflake and use Snowflake credits for compute and storage. You can also explore the product documentation (PostgreSQL Connector, MySQL Connector) and go through our quick tutorial, which provides an easy, step-by-step exercise on getting started with Snowflake's database connectors. Watch this demo video or go to Snowflake.com to learn more.

SHARE

RELATED CONTENT





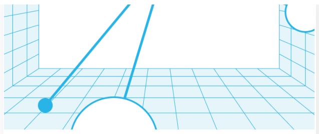


JUL 11, 2024

Snowflake Arctic Cookbook Series: A Deep Dive into LLM Evaluation Standards

What level of astronomy knowledge should an accountant have? Today's evaluations of LLMs assess their performance on a wide range of academic benchmarks and trivia knowledge, with the regular introduction...

Here's How



JUN 27, 2024

Time-Series Forecasting: Comparing Transform Techniques for Tree-Based Models

A key question in data analytics is: What happens next? At Snowflake, this time-series forecasting...

Learn More

JUN 17, 2024

Snowflake Arctic Cookbook Series: Instruction-Tuning Arctic

On April 24, we released Snowflake Arctic with a key goal in mind: to be...

Full Details

START YOUR
30-DAY FREE TRIAL

START NOW

Snowflake Inc.

PLATFORM	SOLUTIONS	RESOURCES	EXPLORE	ABOUT
Cloud Data Platform	Snowflake for Financial Services	Resource Library	News	About Snowflake
Pricing		Webinars	Blog	

Marketplace

Security & Trust

Snowflake for Advertising, Media, & Entertainment

Snowflake for Retail & CPG

Healthcare & Life Sciences Data Cloud

Snowflake for Marketing Analytics

Documentation

Community

Procurement

Legal

Trending

Guides

Developers

Investor Relations

Leadership & Board

Snowflake Ventures

Careers

Contact

AUTHOR

Jakub Puchalski

Sign up for Snowflake Communications

diana.shaw@sn

United States

By submitting this form, I understand Snowflake will process my personal information in accordance with their **Privacy Notice**. Additionally, I consent to my information being shared with Event Partners in accordance with Snowflake's **Event Privacy Notice**. I understand I may withdraw my consent or update my preferences **here** at any time.

SHARE

SUBSCRIBE NOW