Dremio Software

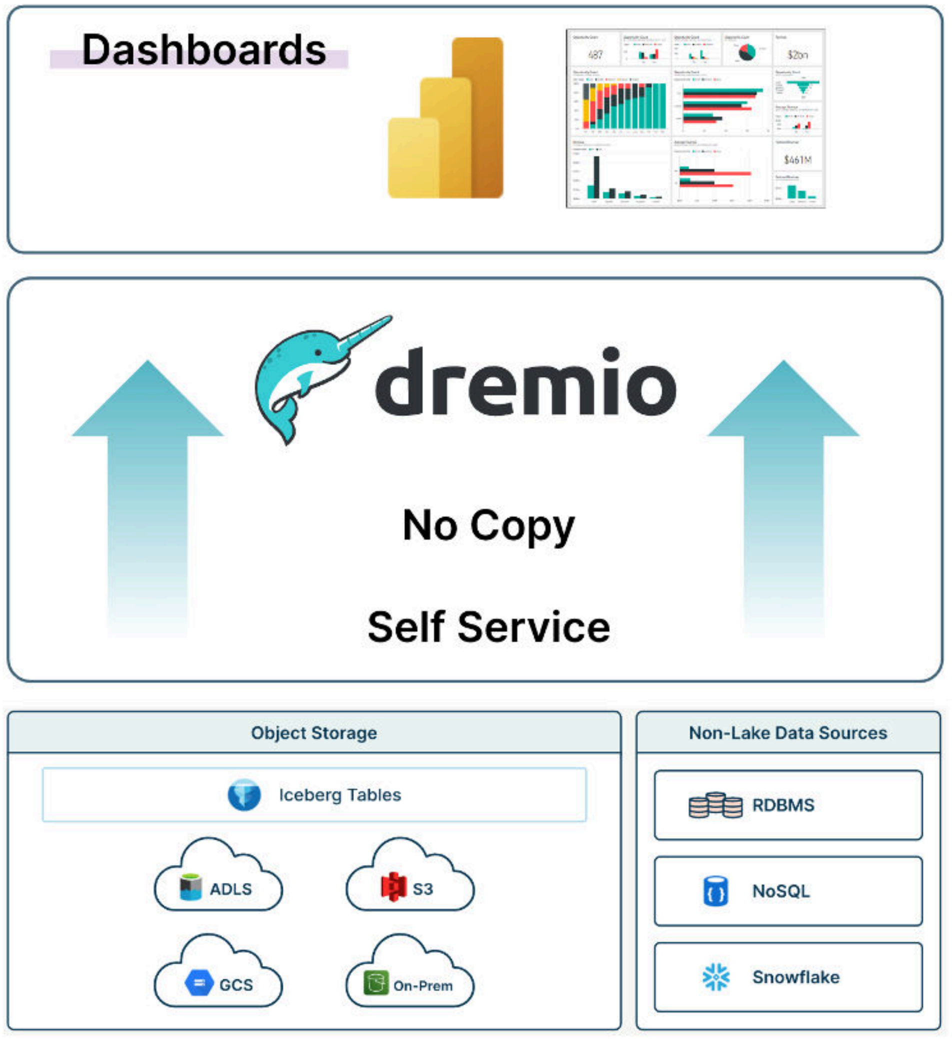
Best Practices for Using Dremio with BI Tools

Introduction

This document provides best practices for business and technical users on how to model and build performant analyses and dashboards using BI tools on Dremio.

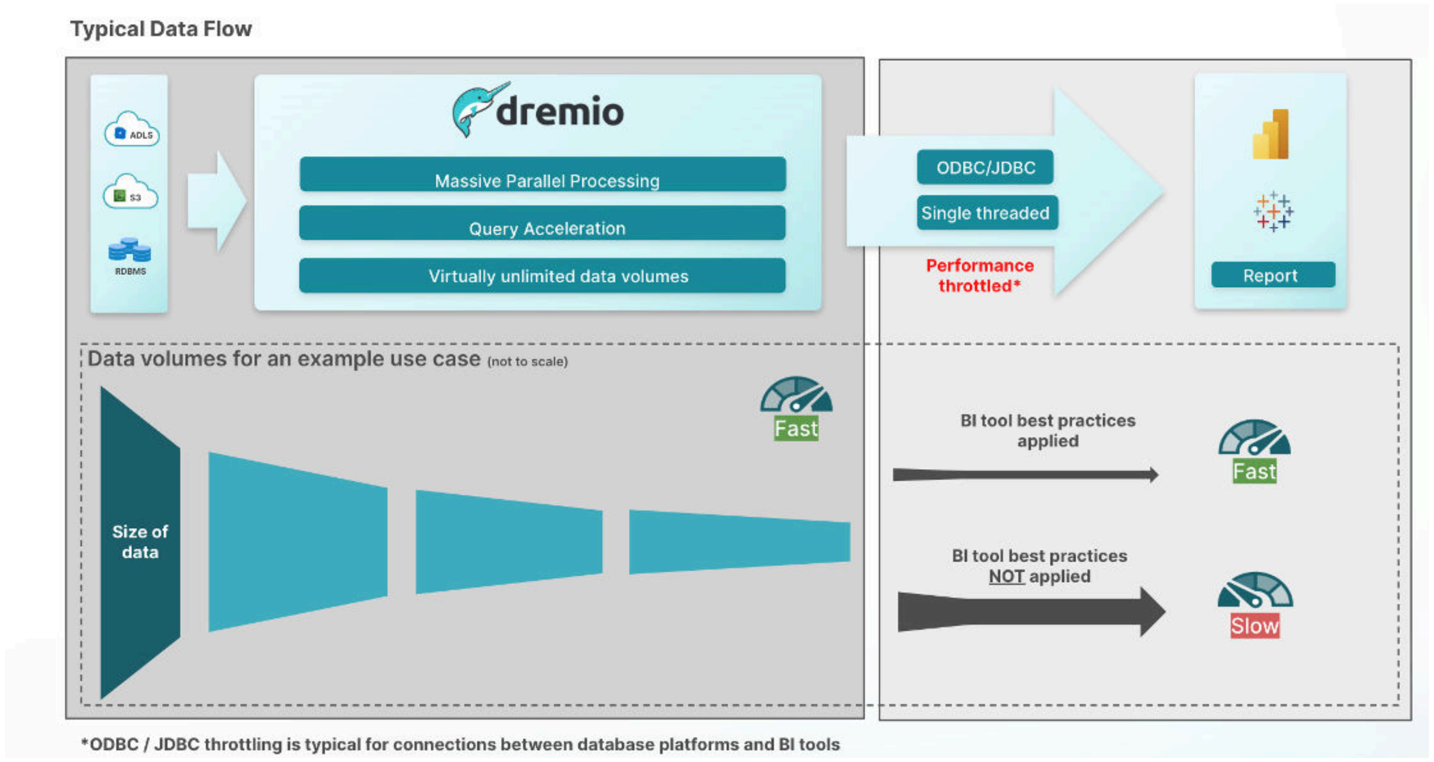
Architecture overview

Dremio acts as the query engine, which connects to all your data lakes or databases and provides BI dashboards with fast and governed access to data.



The Importance of Dremio BI Tool Best Practices

In emerging data lake architectures, data volume has increased significantly compared to traditional databases. At the same time, the connection protocol of BI tools like Power BI (single-threaded ODBC/JDBC) has remained constant. Thus, it is essential to leverage Dremio's query engine and semantic layer before loading the data into dashboards to preserve end users' performance SLAs.


Typical Data Flow
ADLS
S3
RDBMS
dremio
Massive Parallel Processing
Query Acceleration
Virtually unlimited data volumes
ODBC/JDBC
Single threaded
Performance throttled*
Report
Data volumes for an example use case (not to scale)
Size of data
Fast
BI tool best practices applied
Fast
BI tool best practices NOT applied
Slow
*ODBC / JDBC throttling is typical for connections between database platforms and BI tools

General best practices for connecting BI tools to Dremio

1. **Be aware of the size of the underlying data** - Trying to consume billions of records in a dashboard will be either slow or impossible. Be sure to pre-aggregate or limit your data to a granularity that your dashboard can handle.

2. **Be aware of the complexity of your data model and the required operations** - If your dashboards do not meet your performance SLAs, try to simplify your operations or leverage Dremio's reflections to accelerate your most expensive queries.

3. **Review and understand the query patterns of your dashboards** - BI tools will generate SQL queries that are sent to Dremio. These queries can be reviewed in the Dremio job history tab to understand the causes of performance bottlenecks or lack of reflection matching, sometimes introduced by unconventional SQL code created by the BI tool.

4. **Use direct connection (DirectQuery) over imported data (Import Mode), wherever possible** - Direct queries avoid creating stale copies and ensure that your dashboards are always up-to-date. Additionally, by using direct queries, your users benefit from the power of your Dremio cluster instead of relying on their local machines or Power BI service. If you need to materialize data, Dremio reflections are recommended over dashboard imports since you can leverage reflections across all of your queries, dashboards, and use cases.

5. **Verify your underlying data model assumptions on the actual data** - For example, when modeling foreign key relationships between tables, your BI tool will use optimization techniques, like join-pruning, when sending queries to the data lake via Dremio. Your BI tool may return incorrect results if the underlying data quality does not align with these assumptions and constraints.

6. **Enforce data governance via access controls and row-/column-level security** - Dremio's role-based access control (RBAC) allows limiting access to sources, folders, datasets, and even individual rows or columns based on the user's identity or group membership. When users connect via single sign-on in their BI tool, their identity is known to Dremio and appropriate controls are applied.

7. **Make a conscious decision on where you build and maintain your data model** - Having Dremio's semantic layer to access all your data allows you to define a single definition of truth for important business datasets that propagate to all downstream BI dashboards. On the other hand, not every dashboard may need all tables to be pre-joined in the semantic layer and Dremio allows for the flexibility of defining data models inside of the BI tool as well.

8. **Write business logic in SQL rather than using vendor-proprietary query languages (like Power BI's DAX)** - SQL has been the language of data for almost 50 years. It allows logic to be transferred to hundreds of technologies. Vendor-proprietary languages, like DAX, move core business logic into a client application (Power BI) that makes it difficult or impossible to re-use across use cases and technologies.

9. **Monitor your dashboard performance over time** - Both the size of your data and the overall cluster workload will change over time. Be sure you understand the volume and concurrency of users and react with the appropriate workload management and cluster sizing. For example, large dashboards containing many widgets may send dozens of smaller queries. High concurrency limits in Dremio will eliminate the unnecessary need for those queries to wait in the queue.

Power BI-specific best practices for connecting to Dremio

Power BI Connection Types (Dataset Modes)

There are several means of connecting to Dremio from Power BI:

- DirectQuery (DQ): Power BI connects directly to Dremio data on demand
- Import: Power BI captures a snapshot of data from Dremio and caches it
- Composite: Power BI uses both Import and DQ within the same data model

➔ Read More about Connection Types (Dataset Modes).

Ideally, Power BI should always connect to Dremio using DirectQuery (DQ); this offers the lowest data latency and prevents the need to store costly and redundant data copies within Power BI reports and Power BI data models.

➔ Read more about DirectQuery.

For a given data model, the choice of connection type depends on several factors:

- Granularity of data
- Volume of data
- Freshness of data
- Utilization of standard SQL versus custom DAX expressions
- Level of report interactivity
- Application of access controls

Import mode offers the best performance for report interaction, particularly where DAX is used. However, high data volumes can result in increased Power BI Premium Capacity storage costs, and access controls to data must be applied inside Power BI. In contrast, DQ enables reduced data latency and reduced costs but at the expense of interactive performance. Given that both modes offer advantages and disadvantages, the typical best practice is to employ a composite connection strategy that leverages both modes.

Hybrid Tables

This kind of table can leverage both Import and DQ modes. We can configure a partition within Power BI to divide "hot" data (recent) and "cold" data (historical). As a result, the most commonly used hot data is imported into Power BI and less frequently used cold data is accessed dynamically via DQ. This has the advantage of providing fast interaction times for the most used data whilst limiting the amount of data imported into Power BI, reducing Premium Capacity Storage costs. This approach can also facilitate near real-time reporting by switching the hot data to DQ and the cold data to Import. When enabled, this is done automatically by Power BI rather than requiring a partition.

➔ Read more about Hybrid Tables

User-defined Aggregations

⚠ IMPORTANT NOTE
Since August 2022, due to changes in functionality, Power BI will ignore import mode aggregation tables with SSO-enabled data sources (like Dremio) because Power BI cannot keep track of permissions when the granularity of data is reduced via aggregations.

As discussed previously, aggregation is at the core of the composite connection strategy. Aggregations can be defined in either Power BI as part of the Data Model or within Dremio as a view, table or aggregate reflection and consumed by the Data Model. Regardless of the method chosen, it's essential that the aggregation is pushed down to Dremio as much as possible to avoid massive volumes of data needing to be consumed and computed by Power BI.

Power BI Aggregate Tables offer developers the ability to create aggregations without the need to interact with Dremio. Essentially, transformations in Power Query Editor (PQE) are used to define group bys and filters, which are then pushed down to Dremio by a mechanism known as Query Folding. This can have the advantage of making SQL less relevant for Power BI developers, as these aggregations are defined exclusively within the Power BI interface. However, building aggregations in Dremio would make those computations reusable in multiple Power BI data models, reducing the need to build Power BI Aggregate tables and bringing more consistency to models across the enterprise. In practice, commonly used aggregations should always be built in Dremio, whereas dashboard-specific analyses can also be built using Power BI Aggregate tables.

Regardless of whether Power BI Aggregate tables or Dremio aggregated data is used, it is possible to build or consume multiple Aggregations that align with common aggregation patterns within a given Power BI Data Model. These aggregations can be configured within

Power BI such that the optimum table will be selected automatically based on the dimensions selected within a given Power BI report.

➔ Read more about Power BI Aggregate Tables

Limitations of Power BI's proprietary DAX language

Frequently, Power BI reports use Power BI's formula language known as Data Analysis Expressions (DAX). When using DirectQuery with DAX, Power BI will typically render and push down each DAX formula to the data source in serial, which can severely limit the performance benefits of massively parallel processing engines, like Dremio. Parallelization of these queries in DirectQuery mode is only available in Power BI Premium.

Essentially, this means that what appears to be a relatively simple Power BI report may push down many SQL queries in series, so no matter how well the data has been accelerated in Dremio, it will still perform poorly during both report refresh and report interactions. This is particularly notable for report interactions such as drill down and filtering, which will resubmit all the queries to Dremio, meaning that sub-second or even sub-five-second response times are generally unachievable.

For this reason, Dremio recommends writing most business logic in SQL as part of the Semantic Layer rather than using a vendor-proprietary language (DAX) in a client application (Power BI) that makes it difficult or impossible to re-use across use cases and technologies.

Limitations of ODBC and BI tools

Regardless of whether Import or DQ is used, Power BI's connection to data sources (like Dremio) will be handled through the ODBC interface. The nature of ODBC dictates that the data passed from Dremio to Power BI will be transported via a single thread and will require serialization and deserialization into individual rows. This structure is not ideally suited to analytical queries, which tend to aggregate over columns. Ultimately, this means that large volumes of data will take significant time if they are directly ingested by a BI client tool, like Power BI.

To mitigate these challenges when interacting with potentially huge datasets in your data lake via Dremio, it is recommended to:

- Select an appropriate level of granularity
- Apply sufficient filtering
- Push necessary aggregations into Dremio's semantic layer, whenever possible
- Limit the number of columns

Row Level Security

When Row Level Security (RLS) is required, the Connection strategy will determine the nature of its implementation. Since Dremio supports Power BI single sign-on, user permissions can and should be applied in the central data access layer in Dremio instead of relying on multiple, possibly inconsistent, access control implementations in various client tools. Whenever the data is loaded into Power BI using Import or Composite mode, RLS must be applied dynamically inside Power BI.

➔ Read more about Power BI RLS.

Data type and data model definitions

As a general rule, it is advised to push as much business logic, data filtering, and data type conversions as possible into the central data layer inside of Dremio so that all client tools and dashboards can leverage data quality improvements and agree on a single source of truth for business KPIs.

On a technical level, applying column transformations, like data type changes, inside of Power BI can also create performance limitations since Power BI frequently creates unconventional SQL code, which throws off Dremio's upstream reflection matching algorithm. To identify this behavior, it is recommended to review the queries that Power BI sends inside Dremio's job history and validate whether these conform to the expected style.

Similarly, defining a data model with relationships between tables inside Power BI can create data consistency issues if these relationships are not verified. For example, setting a one-to-many relationship between two tables in Power BI assumes that the left-hand table's column has unique entries. The BI tool will inevitably produce incorrect query results if that condition is not met.