WHITE PAPER

databricks

Best Practices Guide

Qlik Sense®

Clever Anjos

Dalton Ruer

David Freriks

Partner Engineering - Qlik

Prasad Kona

Databricks

QLIK.COM

Qlik

Table of Contents

Summary

Qlik and Databricks make a powerful combination when deploying modernized data and analytics pipelines across an enterprise. Due to emerging technologies and urgency of data, there are many architectural avenues available when deploying Qlik and Databricks together. This document will highlight and clarify the options and best practices and discuss high level concepts, practical applications, and help distinguish when to use which approach with Qlik and Databricks. This document will be updated as new capabilities emerge to both Databricks and the Qlik Sense platform. This document also assumes a working knowledge of both Qlik and Databricks technologies by the reader.

Introduction

Qlik Sense sets the benchmark for third-generation analytics platforms, empowering everyone in your organization to make data-driven decisions. Built on our unique Associative Engine, it supports a full range of users and use-cases across the life cycle from data to insight: self-service analytics, interactive dashboards, conversational analytics, custom and embedded analytics, mobile analytics, reporting and alerting. It augments and enhances human intuition with AI-powered insight suggestions, automation, and natural language interaction. And Qlik Sense offers unmatched performance and governance, with the convenience of SaaS or on-premises deployment, or both

Qlik Sense is a complete data and analytics platform enabling users of all levels to explore data with agility and high performance. **Databricks** is a cloud based, massively scalable platform that provides effective management of enterprise class data.

Qlik and Databricks together provide a balance to optimize the "speed of thought" exploration capabilities when Qlik's associative engine with its powerful search and AI capabilities is combined with Databricks's powerful and scalable database engine technology.

Blending the ideas "getting the data you need when you need it" and "getting the data how you need it" with two cutting edge technology platforms creates unique solutions that deliver enterprise analytics, reporting, dashboarding, and data science to the business.

Qlik Architecture

Qlik Sense sets the benchmark for third-generation analytics platforms,empowering everyone in your organization to make data-driven decisions. Built on our unique Associative Engine, it supports a full range of users and use-cases across the life-cycle from data to insight:self-service analytics, interactive dashboards, conversational analytics,custom and embedded analytics, mobile analytics, reporting and alerting. It augments and enhances human intuition with AI-powered insight suggestions, automation, and natural language interaction. Further, Qlik Sense unmatched performance and governance, with the convenience of SaaS or on-premises deployment.

Qlik Sense consists of Qlik-managed cloud-based solutions: *Qlik SenseEnterprise SaaS* & *Qlik Sense Business*, and a customer-managed solution: *Qlik Sense Enterprise Client-Managed*.

THE ASSOCIATIVE DIFFERENCE®

Relational databases and queries were designed in the 1980s for transactional systems, not modern analytics. Query-based tools leave data behind and limit your users to restricted linear exploration, resulting in blind spots and lost opportunities.

Qlik Sense runs on the unique Qlik Associative Engine, enablingusers of all skill levels to explore their data freely without limitations. The Qlik Associative Engine brings together unlimited combinations of data — both big and small — without leaving any data behind. It offers unprecedented freedom of exploration through interactive selection and search, instantly recalculating all analytics and revealing associations to your user in green (selected), white (associated), and gray (unrelated). By keeping all visualizations in context togetherand retaining both associated and unrelated values in the analysis, the Qlik Associative Engine helps your users discover hidden insights that query-based tools would miss.

The Qlik Associative Engine is purpose-built for highly scalable, dynamic calculation and association on massive data volumes for large numbers of users. This unique technology is our primary advantage, with morethan 25 years of innovation and investment.

Associative, in-memory apps

Qlik couples in-memory data caching technology with an Associative Engine that lets you analyze and freely navigate data intuitively. In its second generation, the proven Qlik Associative Engine allows users to easily explore data and create visualizations based on data from multiple data sources simultaneously. These sources range from Excel® and Access® to databases such as Oracle® and SQLServer to big data sources such as Databricks®, Data Lakes with S3, etc.

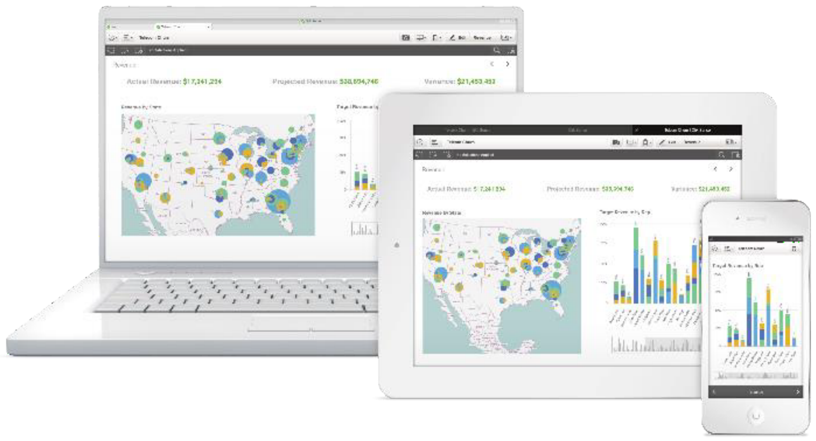
Qlik Sense uses columnar, in-memory storage. Unique entries are only stored once in-memory, and relationships among data elements are represented as pointers.

This allows for significant data compression, more data in RAM,and faster response times for your users.

In some big data scenarios, data should remain at the source, which is why Qlik uses a built-in technique called On-Demand Application Generation. Data sources can be queried based on your users' selections, yet still provide an associative experience to your user. Qlik's Dynamic Views feature expands this capability further for the biggest data sources available.

User Interfaces

Access to the Qlik Sense Enterprise SaaS environment is through a zero-footprint web browser interface (known as the Qlik Sense Hub). The Qlik Sense web browser interface makes all aspects of development, drag-and-drop content creation, and consumption possible. Qlik Sense features a responsive design methodology toautomatically display and resize visualizations with the appropriatelayout and information to fit the device — whether it is a browser on a laptop or desktop, tablet, or smartphone.



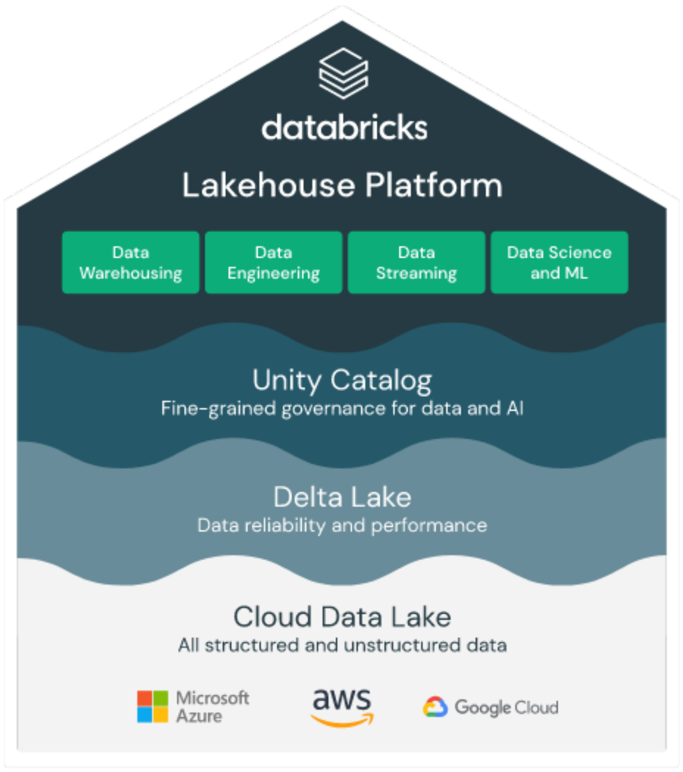
Built with current standards of HTML5, CSS3, JavaScript®, and web sockets, Qlik Sense enables you to build and consume apps on any device.

In addition to the web-based interface, Qlik Sense supports conversational analytics which integrates with major chat platforms such as Slack and MS Teams and data alerting capabilities to allow users tosubscribe to and be notified of key changes to their data.

A quick reference to the entire Qlik Platform including Data Integration capabilities, cataloging, and extending Qlik Sense showcases the power of our integrated suite for Databricks.

Databricks Architecture

The Databricks Lakehouse Platform



The Databricks Lakehouse Platform combines the best elements of data lakes and data warehouses to deliver the reliability, strong governance and performance of data warehouses with the openness, flexibility and machine learning support of data lakes.

This unified approach simplifies your modern data stack by eliminating the data silos that traditionally separate and complicate data engineering, analytics, BI, data science and machine learning. It's built on open source and open standards to maximize flexibility. And, its common approach to data management, security and governance helps you operate more efficiently and innovate faster.

High Level Qlik & Databricks Integration Options

Qlik and Databricks offer a variety of integration capabilities suited for specific scenarios to maximize resources, maintain performance, and fulfill the business requirements across the organization. The number of concurrent users, data refresh frequencies, performance, user experience, and total cost will all contribute to deciding which integration options to take advantage of. Qlik provides several ways to load and consume data from Databricks. The table below describes each integration configuration, feel free to use this as a reference guide.

Integration Option	Method	Use Case
In-Memory	Full / Incremental Load (QVD)	Most common and best performing. Batch reloads leveraging a QVD for incremental updates and optimized for analytics engine.
In-Memory + Databricks	Full / Incremental Load (Change Data Feed)	Databricks optimized reload – especially useful when data is needing to be compared current vs historical state and/or for incremental reloads
On-Demand	ODAG	Structured Drill to Detail for access to large data volumes
On-Demand	Dynamic Views	Supporting details on demand inside existing app as a chart(s)

Summary of Qlik & Databricks Usage Options

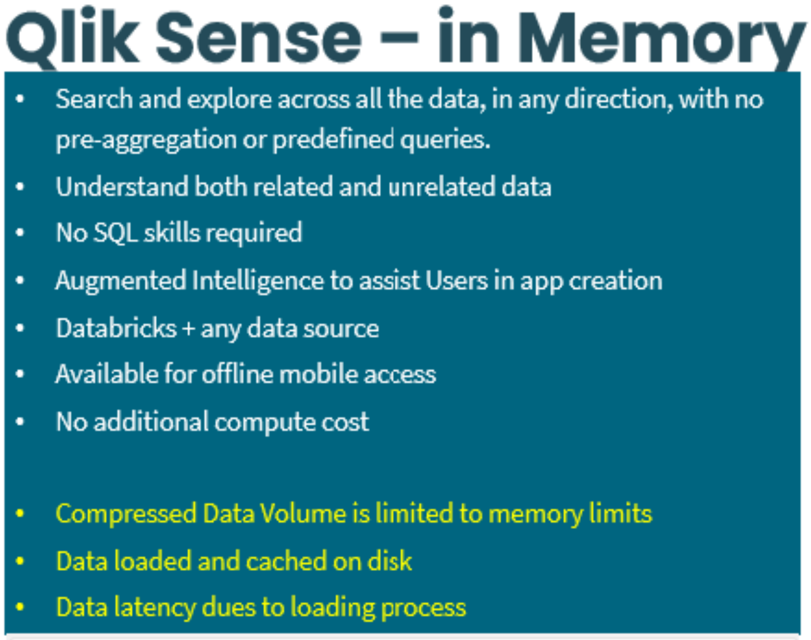
In-Memory options for how to load data into Qlik Sense:

Full reload every time on a schedule:

- Could lead to long load times depending on data volume.
- Recommended if the data is highly volatile or has a high number of changes.

Incrementally load only new data

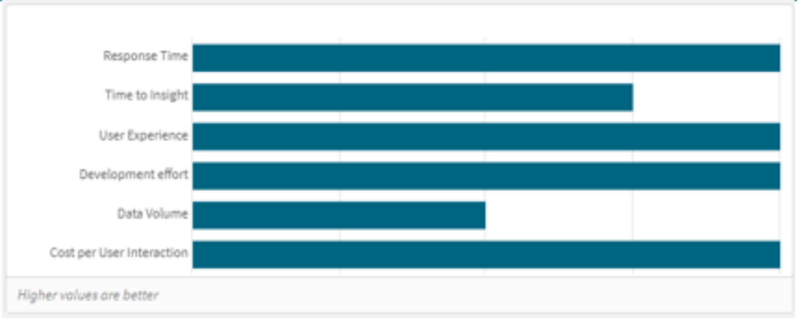
- Reload deltas on a schedule: This can be set up by a Qlik developer in the ELT script. Allows Qlik to only update the changes in the data. This method allows for near-time refresh of data into the Qlik engine and visualization layer.



Qlik Sense – in Memory

- Search and explore across all the data, in any direction, with no pre-aggregation or predefined queries.
- Understand both related and unrelated data
- No SQL skills required
- Augmented Intelligence to assist Users in app creation
- Databricks + any data source
- Available for offline mobile access
- No additional compute cost

- Compressed Data Volume is limited to memory limits
- Data loaded and cached on disk
- Data latency dues to loading process

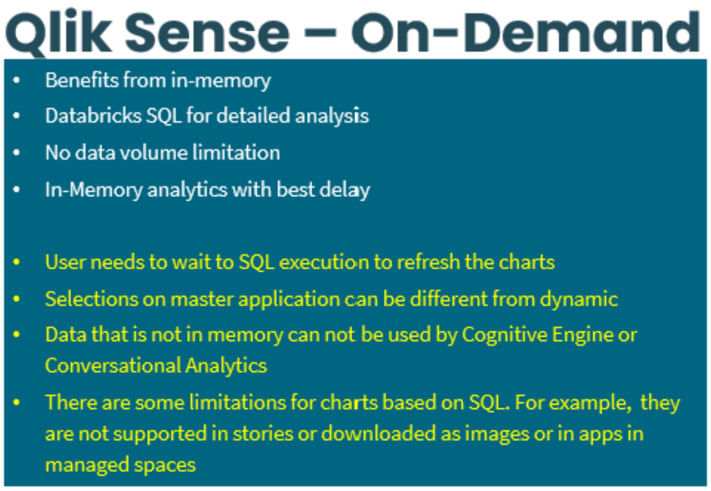


On-demand Apps

On-demand apps help business users and IT departments derive value from big data environments in numerous ways. On-demand apps:

On Demand App Generation (ODAG)

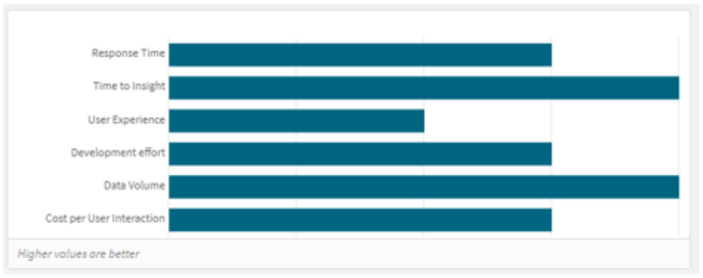
- Detailed Data on Demand: This is a technique typically used in big/huge data scenarios where a Qlik app is built to contain summarized data. There is a Qlik “details” app as a template and that takes parameters passed from the summary app and runs live on demand against Databricks. The user is then presented with the appropriate slice of data based on those selections. This method is good for summary-detail analytics.



Qlik Sense – On-Demand

- Benefits from in-memory
- Databricks SQL for detailed analysis
- No data volume limitation
- In-Memory analytics with best delay

- User needs to wait to SQL execution to refresh the charts
- Selections on master application can be different from dynamic
- Data that is not in memory can not be used by Cognitive Engine or Conversational Analytics
- There are some limitations for charts based on SQL. For example, they are not supported in stories or downloaded as images or in apps in managed spaces



Dynamic Views

- Live Data Visualizations: This option is used when live / near-time data is needed as part of the Qlik app. Using the ODAG framework, live data under certain query volume thresholds can be triggered to update based on user interactions with the application. As users make selections they will be prompted and if they choose to, Qlik will reload data live from Databricks to match their selections.

Direct Query applications

- Direct Query is a capability in Qlik Sense SaaS that enables analytics apps to generate SQL queries directly against cloud databases as the user interacts with data through visualizations and dynamic filtering. There is no need to load any data ahead and there are no limits on the data volumes involved and the performance is all driven by the underlying cloud database

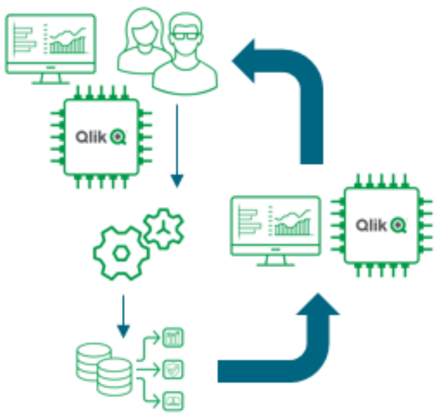
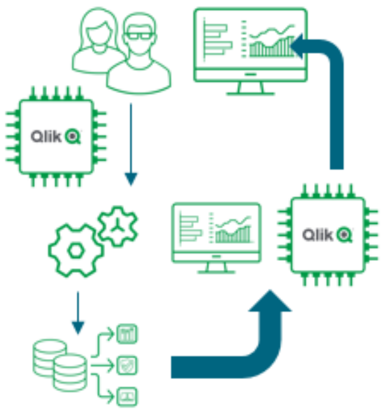
Deep Dive Databricks & Qlik Method Decision Tree:

Many organizations have well established data pipelines and a user base begging for access to data and insights. But between the databases and the dashboards, lay the the integrations and load architectures that make consuming data possible. This next section is to help you understand several integration options available in Qlik Sense and which to use when deploying Qlik and Databricks together.

Some of the main variables when deciding will be the total number of concurrent users, system performance thresholds, costs, and the actual business requirements.

Let's begin with a high level understanding of each of the Qlik integration options before discussing which business use cases they might apply to:

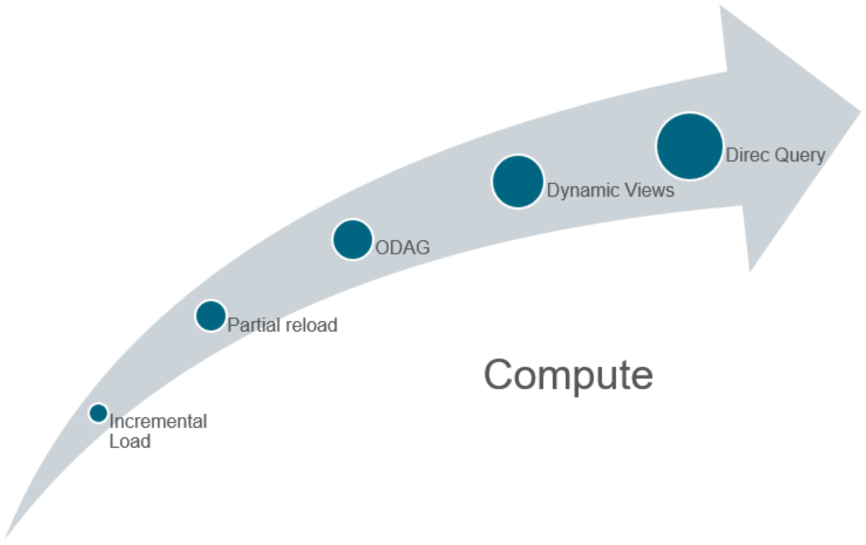
Incremental Load (Scheduled Data Load) Cached In-Memory	An Incremental Load is a scheduled process where Qlik loads the latest data for all of the tables needed, merges the changes to existing QVD's (the previously loaded data/cache), then rewrites those QVD's/Cache. • All applications and users would have access to the new information for those tables. • When dealing with large data sources, this is a great balance between fresh and fast as only the changed data is loaded on a pre-defined schedule after the initial bulk load.
Partial Reload Cached In-Memory + New Data Added In-Memory (On-Demand or Scheduled)	A full reload or incremental load always starts by deleting all tables in the existing data model, and then runs the load script. A partial reload keeps all tables in the data model and then executes only Load and Select statements preceded by an **Add**, **Merge**, or **Replace** prefix. A Partial Reload is where Qlik loads the changed data for a subset of tables and the data is brought into the Associative Engine.

	This technology provides a balance between fresh and fast while also ensuring that the end user is confident that they are seeing the most recent values because they initiate the action rather than it being scheduled. This approach is very helpful when your data source has information about what kind of changes (inserts, updates and deletes)
On Demand Application Generation - ODAG (Live Data Load) Cached In-Memory For selections + Live Data based on Selections in second application when user asks	On Demand Application Generation is a process where an end user passes the selections they have made to a pre-defined template application. Qlik then copies that template, loads data for that specific cohort and presents it to the end user as a new application. • The calling application may be fully in-memory with aggregates, offering speed of thought action, while the spawned application then pulls Live details only if and when they are needed. 
Dynamic Views (Live Data Load) Cached In-Memory For selections + Charts from Live Data based on Selections when user asks	Dynamic Views is a process similar to ODAG but instead of surfacing a new application to the end user, selected charts from a template application are displayed in the application they are currently using. If the user changes their selections they can ask for the Dynamic Views to be refreshed. • This option capitalizes on the functionality of ODAG, while also rendering the "live" cohort of data in the context of the application where the users made their selections. 
Direct Query	You may use this approach when your data volumes are huge or change too often what would make difficult to load the data into the Qlik Engine

So which one is right for YOU?

As a primer to choosing the right solution for the right problem, let's begin with some very high level guidance starting with a focus on how much compute and thus cost is driven by each of the solutions. Because in many cases the difference between "We WANT Live Data" and "1 minute old will be FINE" may come down to the implementation cost. In another use case Live Data is a MUST have regardless of the compute resources and cost associated in solving the problem.

The following chart contains no scale on purpose. It is simply for portraying the concept that each technology will have increasing amounts of computing requirements associated with them.



Asking the question "Why does each solution increase the amount of compute/cost?" Well, doing an Incremental Load means that Qlik only asks for the changes from our source 1 time per scheduled reload. As Partial Re-Load's of data are kicked off by the end user, the same data would need to be consumed by multiple users, thus increasing the compute needed. On Demand Application Generation (ODAG) applications will load all of the data for a defined cohort even if it hasn't changed in years. Dynamic Views require the same data loading as ODAG applications, and as they can be refreshed as often as desired by the end user to keep in synch with their selections they require more computing power. From one usability perspective Dynamic Views tend to be more used than ODAG since the user sees the app as only one datasource. The user not always is aware that SQL queries are execute compared to ODAG when the user hits the “generate new app” button. And using Direct Query all the queries are pushed down do the DeltaLake for every selection the user might use.

Remember though, that was only a general guide. It is entirely possible that one of your business use cases only requires a single SQL for an aggregate in an application that is only used by a small set of end users. Bringing us to the next point: Business driven use cases are often complex and require the answers to many questions.

The following scenarios walks through some hypothetical use case that lines up with each of the architectural solutions.

Scenario 1 – **ODAG**: Drill to Detail Reporting/Analysis

Starting with a summary application of key metrics, a user chooses a selection of criteria which then is passed to a secondary application that is generated on demand. A customer example of when to use ODAG:

> A Human Resources department has an existing application that is used by 1,290 HR administrators, managers and employees through the company who make a lot of selections. The application is currently on a one-hour incremental load schedule. They want to use real time data instead of waiting for reloads to see potential overtime issues. The details are a known subset of content in a specific format that shows potential issues, therefore ODAG provides an interim reload ability in an easily consumable detailed Qlik application.

Scenario 2 – **Dynamic Views**: Transactional Details in Context

This scenario is best suited for when details for specific transactions need to be viewed in context with the original Qlik Sense application as a chart inside the application. A customer example of when to use Dynamic Views:

> A customer has several hundred billion records from a transactional system stored in Databricks which is too much for a single Qlik application. This data is reloaded on a schedule with the aggregates of KPI's and other relevant data but not the transactions themselves. Dynamic Views are used to get the transactions of a cohort of dimensional values that limit the records to a threshold (say under 100k rows) to be analyzed in any chart on demand. The

users of the application need to be able to see the details in the context of the cohort selected.

Scenario 3 – **Partial Reload/Merge**: Closing Books / Financial Reporting

Finance leaders closing the books at the end of the month need access to up to the minute general ledger details. The Qlik application contains very complicated calculations, hierarchies, and transformations not easily replicated with SQL. The application will only be used by Accountants and Sr Executive in the corporation which equates to about 25 people. Data is changing rapidly throughout the close cycle and the users need to see where the company stands at any point in time. End users have a very high level of interactivity are required to analyze the data to find issues.

Personas by Usage: Consumers, Analysts, Designers

Many roles support the data to insight pipeline - from developers to analysts to business casual consumers. Each role though requires certain capabilities within the platform but also levels of access to live data. See the table below to learn which use case fits each persona.

Consumer	Uses prebuilt dashboards, mashups, or pushed content (Reports/Alerts)
Analyst	Deep dives into content, can create own content, understands data literacy
Designer	Builder and deployer of prebuilt content and complex design/data transforms

*This chart assumes a superset of capabilities (i.e. Designer has Analyst and Consumer capabilities, where Consumers only have the singular capability)

Use Case	In-Mem (FL/IL)	In-Mem (TT IL)	In-Mem (TT Merge)	ODAG	Dynamic Views
Business Monitoring	C	C	A		
Business Alerting	C	C			
Business Reporting	C	C			
Embedded / Mashups	C	C	C		
Ad-hoc Analysis	A	A	A	A	A
Insight Suggestions	C	C	A		
Insight Exploration	A	A	A		
Storytelling	A	A	A		
Assisted Data Prep	A				
Data Ingestion (LS)	D	D	D	D	D

Use Advanced Analytics (SSE)[1]	C	C			
Develop Advanced Analytics (SSE)[1]	A	A	A		
Data Modelling and ETL	D	D	D	D	D
Application Design & Development	D	D	D	D	D

[1] Only available on client managed

FL: Full Load

IL: Incremental Load

TT: Databricks Time Travel

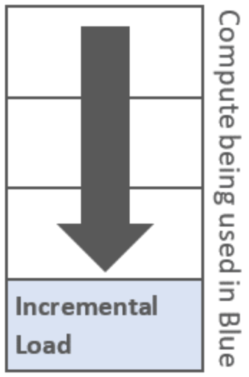
Performance Considerations / Best Practices by Technique

As previewed, Qlik and Databricks together offer a range of ingestion techniques with varying levels of loading data into memory with Qlik's analytics engine and purely loading near/real-time data. There are clear strengths and advantages for each technique in addition to some having limited use cases.

It's necessary to understand the caching process between Databricks and Qlik to distinguish how to optimize each system to support your analytics needs.

Incremental Load Options

The following sections will discuss in detail the above strategies on how to leverage Databricks data depending on use case.



Incremental Loads using QVD's

This is the most common technique used in Qlik environments to only load delta data.

Qlik has a data format called QVD that is binary and optimized for re-ingestion locally. It is a commonly used by Qlik developers as it allows for snapshotting data, reuse of data structures after transformation, combination and pre-calculation of data, and **incremental loads**. Here is the basic process of how it works:

1) Table of data is identified as large enough to not reload entirely every time.
2) Load all the data the first time, store data into QVD file on disk
3) On subsequent loads:
    a. Identify a column in the table used for knowing that it is a new record. It can be an incrementing, numerical ID value in the table, or possibly a date/time field.
    b. After the data is loaded from, get the *highest value* for the aforementioned column and store it in a variable (e.g. "LastDate")
    c. MERGE LOAD the new data from Databricks but selecting from the big table and adding a WHERE clause and including the variable from above; For example, "select * from table where datetimefield > $(LastDate);"
    d. Save the QVD, overwriting the previous QVD. If desired, snapshot by saving an additional QVD with a timestamp or similar. This way you easily can go back to previous loads if desired.
    e. Repeat for any other tables that have a high number of values.

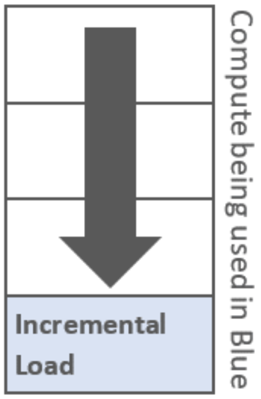
The above is the basic concept; However, there are nuances to account for, so the following information will dive deeper into the variants.

Here is a Qlik Community Post describing the process: https://community.qlik.com/t5/Qlik-Design-Blog/Qlik-Sense-Incremental-Load-using-Merge/ba-p/1944225

QVD-based Incremental Load - Insert Only

This method offers the ability to retrieve only new records. No updates or deletes are considered

Example:

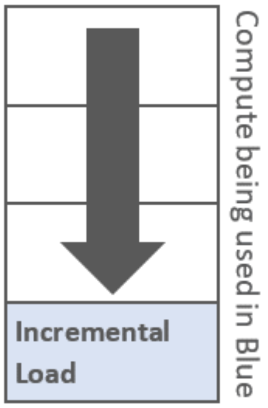


```
QVDFile='lib://DataFiles/NW.Orders_InsertOnly.qvd';
if FileSize('$(QVDFile)') > 0 then //If there is already extracted data
    [NW.Orders]:
    Load
        OrderID,
        CustomerID,
        EmployeeID,
        OrderDate,
        RequiredDate,
        ShippedDate,
        ShipVia,
        Freight,
        ShipName,
        ShipAddress,
        ShipCity,
        ShipRegion,
        ShipPostalCode,
        ShipCountry
    from [$(QVDFile)](qvd);
    t:load Max(OrderID) as MaxOrderID Resident [NW.Orders];
    LastOrderID=Peek('MaxOrderID'); // Retrieves the last OrderID
    Drop Table t;
else
    LastOrderID = '0';
    [NW.Orders]:load null() as OrderID AutoGenerate 0; // fake table
endif

Concatenate([NW.Orders])
SELECT OrderID,
    CustomerID,
    EmployeeID,
    OrderDate,
    RequiredDate,
    ShippedDate,
    ShipVia,
    Freight,
    ShipName,
    ShipAddress,
    ShipCity,
    ShipRegion,
    ShipPostalCode,
    ShipCountry
FROM northwind.orders
Where OrderID > $(LastOrderID);

Store [NW.Orders] into [$(QVDFile)](qvd);
```

QVD-based Incremental Load - Insert and Update

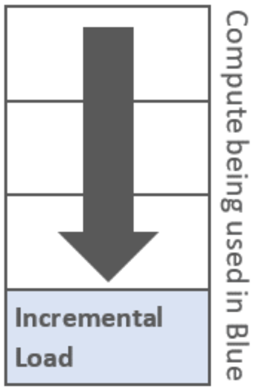
For this approach we rely in two fields:

- A timestamp or sequential value so we can extract only records added/updated since the last execution
- A primary key, so we will keep only the most recent record for every key



```
QVDFile='lib://DataFiles/NW.Orders_InsertUpdate.qvd';
if FileSize('$(QVDFile)') > 0 then //If there is already extracted data
    [NW.Orders]:
    Load
        OrderID,
        CustomerID,
        EmployeeID,
        OrderDate,
        RequiredDate,
        ShippedDate,
        ShipVia,
        Freight,
        ShipName,
        ShipAddress,
        ShipCity,
        ShipRegion,
        ShipPostalCode,
        ShipCountry,
        updated_dt
    from [$(QVDFile)](qvd);
    t:load Max(updated_dt) as updated_dt Resident [NW.Orders];
    updated_dt=Timestamp(Peek('updated_dt'),'YYYY-MM-DD hh:mm:ss.fff');
    // Retrieves the last update
    Drop Table t;
else
    updated_dt = '2000-01-01'; // Retrieves the last update
    [NW.Orders]:load null() as OrderID AutoGenerate 0; // fake table
endif

Merge(updated_dt) on OrderID Concatenate([NW.Orders])
Load If(Exists(OrderID),'U','I') as Operation, *; // New record? based on OrderID
SELECT OrderID,
    CustomerID,
    EmployeeID,
    OrderDate,
    RequiredDate,
    ShippedDate,
    ShipVia,
    Freight,
    ShipName,
    ShipAddress,
    ShipCity,
    ShipRegion,
    ShipPostalCode,
    ShipCountry,
    updated_dt
FROM northwind.orders
Where updated_dt > '$(updated_dt)';

Store [NW.Orders] into [$(QVDFile)](qvd);
```

QVD-based Incremental Load - Insert, Update and Delete



What if records were **deleted** from the source database **between script executions**? In this case we need to:

- Collects all changes since last execution
- Drops all records that don´t exist on source anymore

```
QVDFile='lib://DataFiles/NW.Orders_InsertUpdateDelete.qvd';
if FileSize('$(QVDFile)') > 0 then //If there is already extracted data
    [NW.Orders]:
    Load
        OrderID,
        CustomerID,
        EmployeeID,
        OrderDate,
        RequiredDate,
        ShippedDate,
        ShipVia,
        Freight,
        ShipName,
        ShipAddress,
        ShipCity,
        ShipRegion,
        ShipPostalCode,
        ShipCountry,
        updated_dt
    from [$(QVDFile)](qvd);
    t:load Max(updated_dt) as updated_dt Resident [NW.Orders];
    updated_dt=Timestamp(Peek('updated_dt'),'YYYY-MM-DD hh:mm:ss.fff');
    // Retrieves the last update
    Drop Table t;
else
    updated_dt = '2000-01-01'; // Retrieves the last update
    [NW.Orders]:load null() as OrderID AutoGenerate 0; // fake table
endif

Merge(updated_dt) on OrderID Concatenate([NW.Orders])
Load If(Exists(OrderID),'U','I') as Operation, *; // New record? based on OrderID
```

```
SELECT OrderID,
    CustomerID,
    EmployeeID,
    OrderDate,
    RequiredDate,
    ShippedDate,
    ShipVia,
    Freight,
    ShipName,
    ShipAddress,
    ShipCity,
    ShipRegion,
    ShipPostalCode,
    ShipCountry,
    updated_dt
FROM northwind.orders
Where updated_dt > '$(updated_dt)';

// Keeping only records which OrderID is still in database, this will remove records
// that don´t exists at source anymore
Inner Keep([NW.Orders])
SQL SELECT DISTINCT OrderID
FROM northwind.orders;

Store [NW.Orders] into [$(QVDFile)](qvd);
```

Incremental Loads using "Partial Load"

Partial Load is helpful when you have new data to append to an existing table but do *not* want to load the rest of the tables. This is helpful when you have one table with a lot real time chances and a large data set from many sources in the rest of the data model.

Examples:

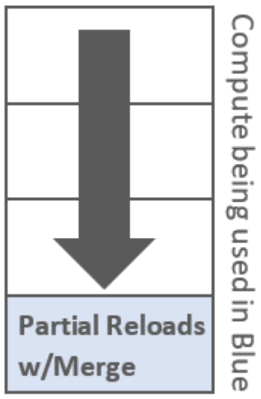
```
ADD LOAD * from Databricks.FactTable where createdatetime > $(lastreloaddatetime);
```

The above will run when the "Partial" option has been set on a reload AND will run when the partial flag is not set.

```
ADD ONLY LOAD * from Databricks.FactTable where createdatetime > $(lastreloaddatetime);
```

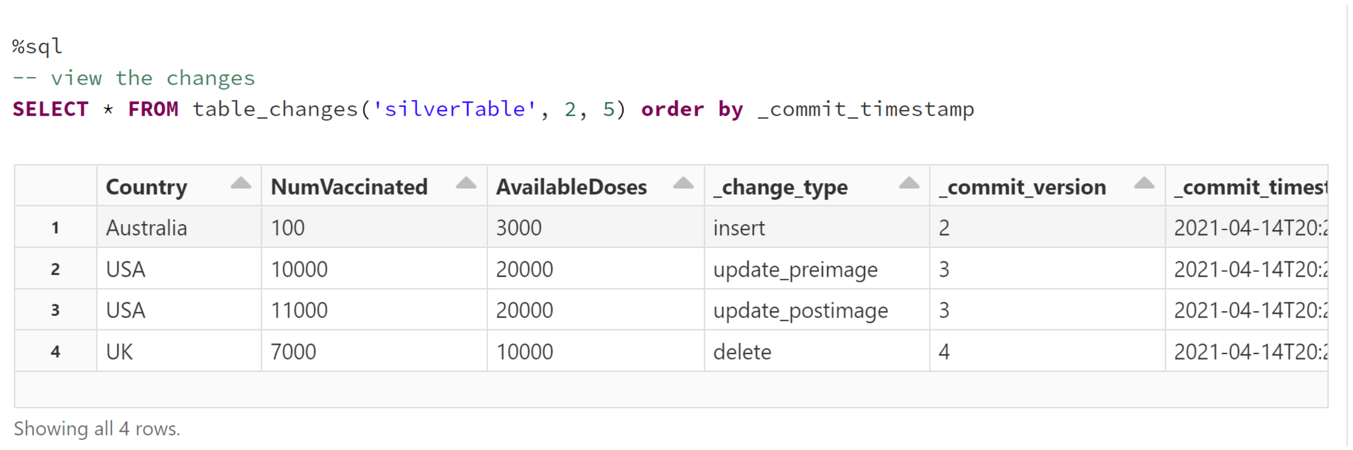
The above will run ONLY when the Partial flag has been set and will *not* be run when partial is set to false in the reload.

Change data feed & Merge



Traditionally Qlik developers have utilized a Last Modified Timestamp field so that they could load only the changed values. However, there are many times that data tables don't have a Last Modified Timestamp so then Full Reloads were the only option.

However, Databricks offers Change data feed. It's the ability to simply get the Change Data Capture information returned in a query. When it is enabled (by table), it adds metadata columns behind the scenes to your tables that allow you to modify your typical SQL Selection using a “where” clause to retrieve only changes since last execution.



```
%sql
-- view the changes
SELECT * FROM table_changes('silverTable', 2, 5) order by _commit_timestamp
```

	Country	NumVaccinated	AvailableDoses	_change_type	_commit_version	_commit_timest
1	Australia	100	3000	insert	2	2021-04-14T20:2
2	USA	10000	20000	update_preimage	3	2021-04-14T20:2
3	USA	11000	20000	update_postimage	3	2021-04-14T20:2
4	UK	7000	10000	delete	4	2021-04-14T20:2

Showing all 4 rows.

We can ignore records where _change_type = ‘update_preimage’ since we are only interested in the new values from updated records.

Combining this CDC capture information from Databricks with the Qlik Merge function offers a highly performant way to do handle Incremental Loads. The Merge function provides a simple way of automatically "merging" changes into an in-memory table. You simply pass the function a few parameters and the results of the Databricks Changes query for the table.

Example where Change Data Feed is enabled in a table

```
[silvertable]: // full load
SELECT
    Country,
     NumVaccinated,
     AvailableDoses
FROM
    `default`.silvertable;
```

```
If IsPartialReload() then
  //Sets a default starting point in time if this is the very first time
  Let Last_commit = Alt(chr(32)&Timestamp(Last_commit,'YYYY-MM-DD hh:mm:ss.ff')&chr(32),1);
  // Incremental load
  Merge only (_commit_timestamp, Last_commit) on Country Concatenate(silvertable)
  SELECT
      left(_change_type,1) as Operation, // We need only the first character
      Country,
      NumVaccinated,
      AvailableDoses,
      _commit_timestamp
  FROM table_changes('silverTable', $(Last_commit))
  WHERE _change_type !='update_preimage'
  ORDER BY _commit_timestamp;

  Drop field _commit_timestamp from [silvertable];//Optional
End if
```

This functionality:

1. Speeds up traditional Incremental Load applications.
2. Can be used to handle Incremental Loads for tables without Last Modified Timestamp columns where you may be doing full reloads.
3. Partial Reloads to allow end users to quickly bring the latest and greatest changes very quickly into memory so that they can analyze them.

Change data feed & Qlik Merge Function Details

Details about Databricks Change Data Feed can be found here:
https://docs.databricks.com/delta/delta-change-data-feed.html

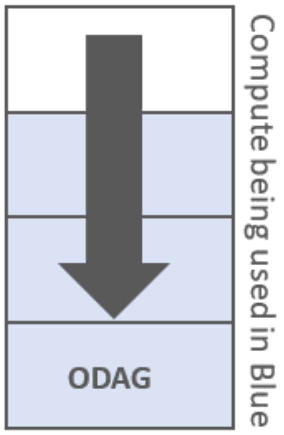
Details about the Qlik Merge function can be found here: https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense_Hub/Scripting/ScriptPrefixes/Merge.htm

On-Demand Options

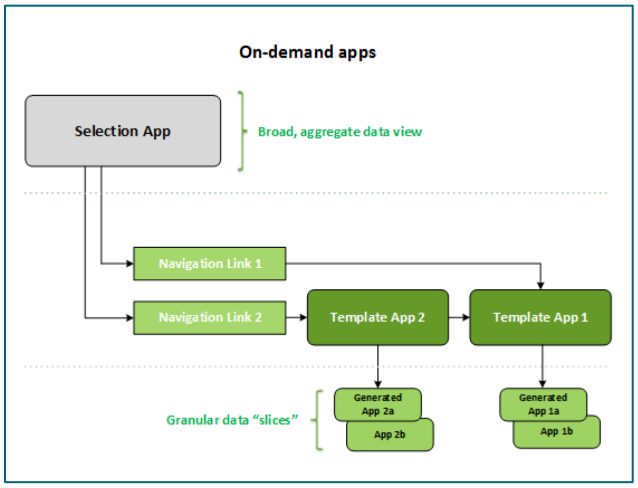
The following sections will discuss in detail the strategies on to leverage real/near time data from Databricks.

On Demand App Generation (ODAG)

Commonly called ODAG, is about bringing in just the right chunk of data you're interested in to do your analysis. It's a technique commonly used in large volumes scenarios where it's just not possible or efficient to load all of the data into their Qlik app. It's common to see this approach when the fact table exceeds 500-800 million rows (purely estimate, depends on rest of data model, how frequent the source is updated and several other scenarios)



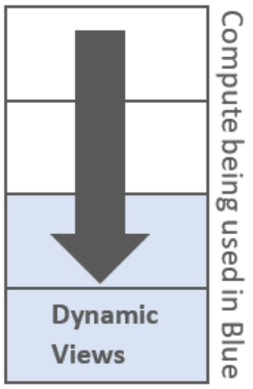
A simple example might be that user comes in to a "shopping cart app" where they see trends and explore high level aggregated data, then make selections to narrow in on the specific dimension values of interest by using the associativity in the engine. When the user has confirmed that the cohort of data is manageable enough of and has the appropriate parameters chosen, then the user can launch the "analysis app" which has all of the chart objects and layout already defined, or after loading that data it can return a blank sheet if desired.



Here is the official help document for creating and managing On Demand Apps: https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense_Hub/DataSource/Manage-big-data.htm

Dynamic Views

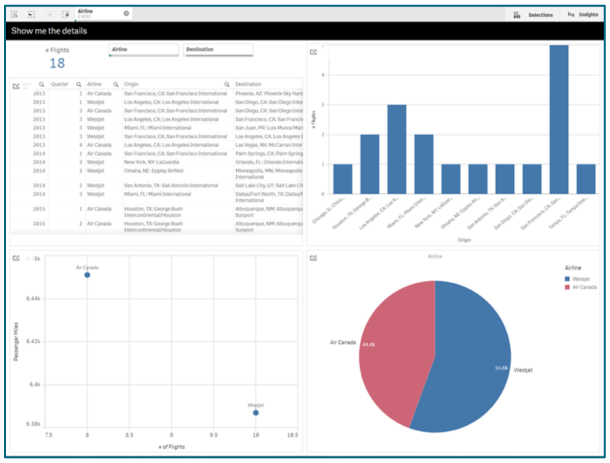
Dynamic views enable you to connect a base app to another app. Master visualizations from that app can then be used in the base app. This enables app creators to use master visualizations from the template app as dynamic charts in other apps. There is no limit to the number of dynamic views you can add to your base app.



Dynamic views are made from three main components:

- Dynamic views: A mechanism added to base apps that connect to template apps and enable app creators to add master visualizations from the template app to the base app.
- Dynamic view template apps: A Qlik Sense app containing connections to data sources, such as cloud databases.
- Dynamic charts: Master visualizations in the dynamic view template app that can be added to base apps and that can be manually refreshed by users.

The template app and the base app do not need to use the same data. If you have a data set covering customer purchases, you could add a dynamic view to a template app containing weather data to look at any correlations.



Year	Quarter	Airline	Origin	Destination
2013	1	Air Canada	San Francisco, CA: San Francisco International	Phoenix, AZ: Phoenix Sky Harb
2013	1	Westjet	Los Angeles, CA: Los Angeles International	San Diego, CA: San Diego Inter
2013	3	Air Canada	San Francisco, CA: San Francisco International	San Diego, CA: San Diego Inter
2013	3	Westjet	Los Angeles, CA: Los Angeles International	San Francisco, CA: San Francis
2013	3	Westjet	Miami, FL: Miami International	San Juan, PR: Luis Munoz Mari
2013	3	Westjet	San Francisco, CA: San Francisco International	Los Angeles, CA: Los Angeles I
2013	4	Air Canada	Los Angeles, CA: Los Angeles International	Las Vegas, NV: McCarran Inter
2014	1	Air Canada	San Francisco, CA: San Francisco International	Palm Springs, CA: Palm Spring
2014	2	Westjet	New York, NY: LaGuardia	Orlando, FL: Orlando Internatio
2014	2	Westjet	Omaha, NE: Eppley Airfield	Minneapolis, MN: Minneapolis International
2014	2	Westjet	San Antonio, TX: San Antonio International	Salt Lake City, UT: Salt Lake Cit
2014	3	Westjet	Miami, FL: Miami International	Dallas/Fort Worth, TX: Dallas/F International
2015	1	Air Canada	Houston, TX: George Bush Intercontinental/Houston	Albuquerque, NM: Albuquerqu Sunport
2015	2	Air Canada	Houston, TX: George Bush Intercontinental/Houston	Albuquerque, NM: Albuquerqu Sunport

If the data queried from the template app's source can be filtered using values in your base app, you can use binding expressions in the template app's script. This enables the dynamic view to only query a subset of data specifically related to the selections in the base app from the data sources of the template app. For example, you could bind the field SalesDate in the base app to the field DailyTemperatureReadingDate in the template app.

Here is the official help document for creating and managing Dynamic Views:

https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense_Hub/DynamicViews/dynamic-views.htm

Direct Query Applications

Direct Query gives the user more options on how they want to access their data in order to suit their individual needs. Accessing the data through Direct Query allows the user to keep the data in its underlying data source. This increases the speed at which users can interact with their data in exchange for some of the flexibility that an in-memory Qlik Cloud app offers.

In general, it is recommended that you import your data into Qlik Cloud whenever possible. In-memory Qlik Cloud apps allow you to better customize your experience and get the most out of your data. However, if you find yourself in a situation where your goals cannot be met by importing data, Direct Query might be the solution for you. Due to their streamlined functionality, Direct Query apps can also help new users take their first step towards creating fully functional and fast in-memory apps.

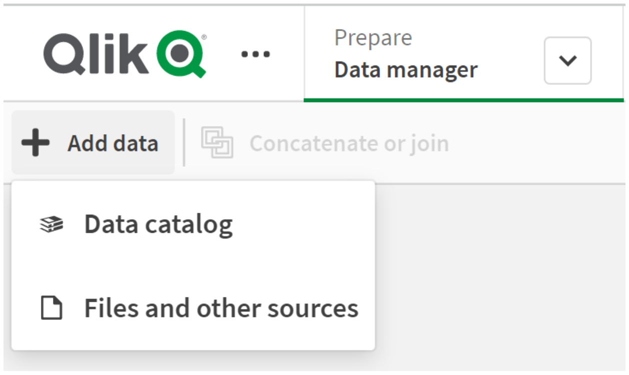
You might consider using Direct Query when:

- Handling big data sources that would not fit in Qlik Engine memory
- Exploring new databases and tables.
- You are in need of near real-time data, for example to see how many orders have come in during the last hour.
- Prototyping your dashboard in the initial phases before production.
- Extracting slices of data into the Qlik Sense engine through ODAG apps.

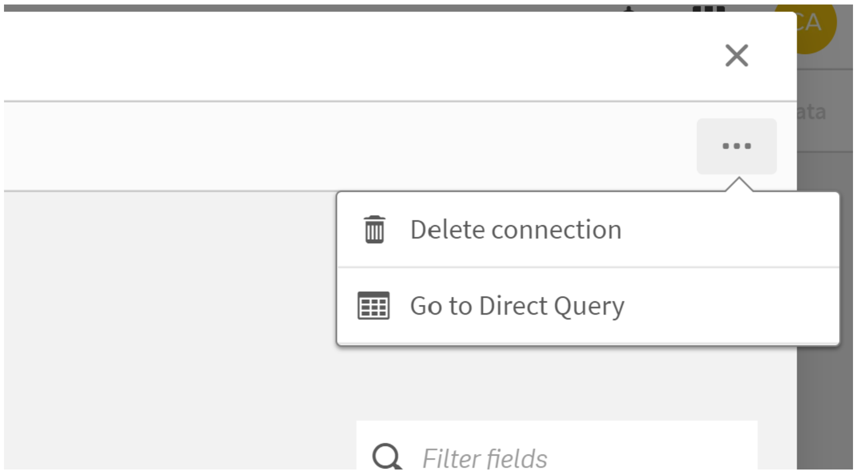
Please refer to Supported capabilities section of Qlik Help to be informed about which features are available when you are creating Direct Query apps.

Creating a Direct Query Application data model

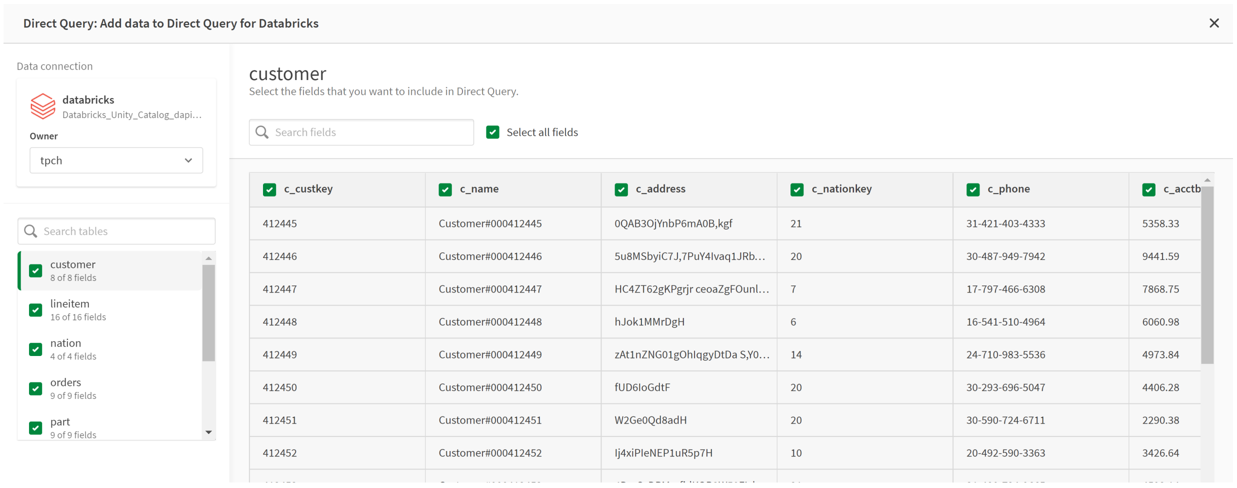
1. Create one Qlik app as usual
2. Go to the Data Manager
3. Use the option **Add data** and then **Files and other sources**



4. Select your Databricks connection on the left panel
5. On your top right menu, you´re going to see the option **Go to Direct Query**

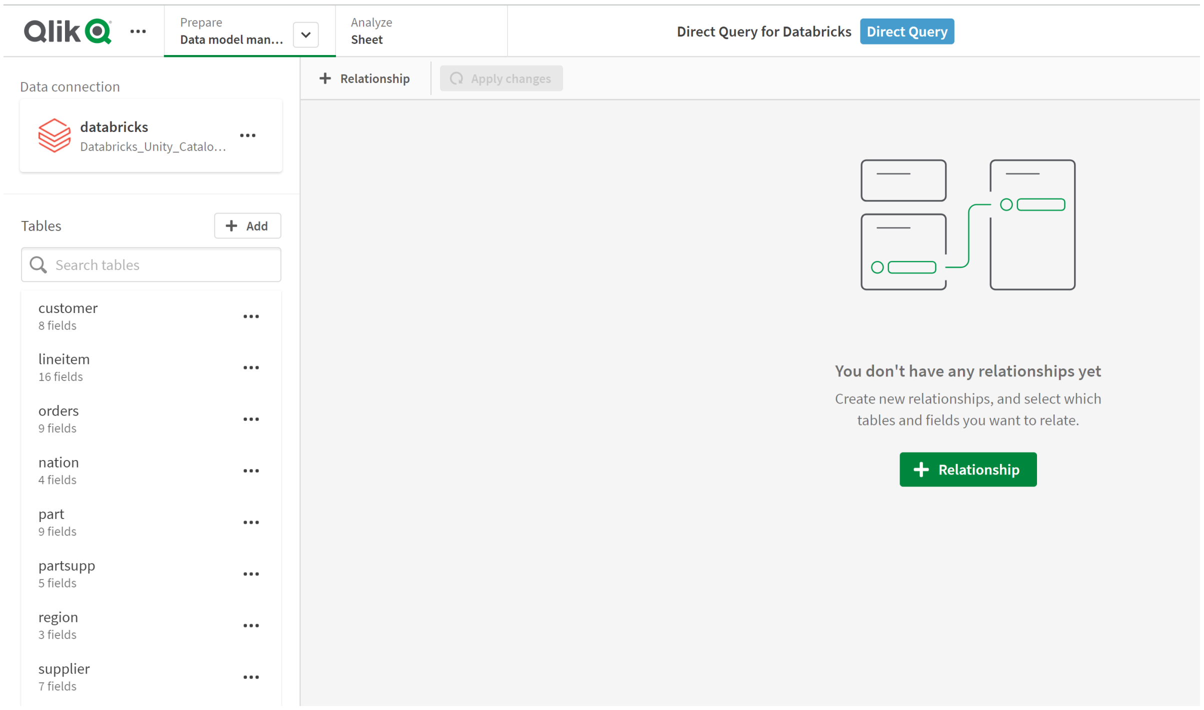


6. In the next screen, select the tables that will make part of your data model and press **Next**

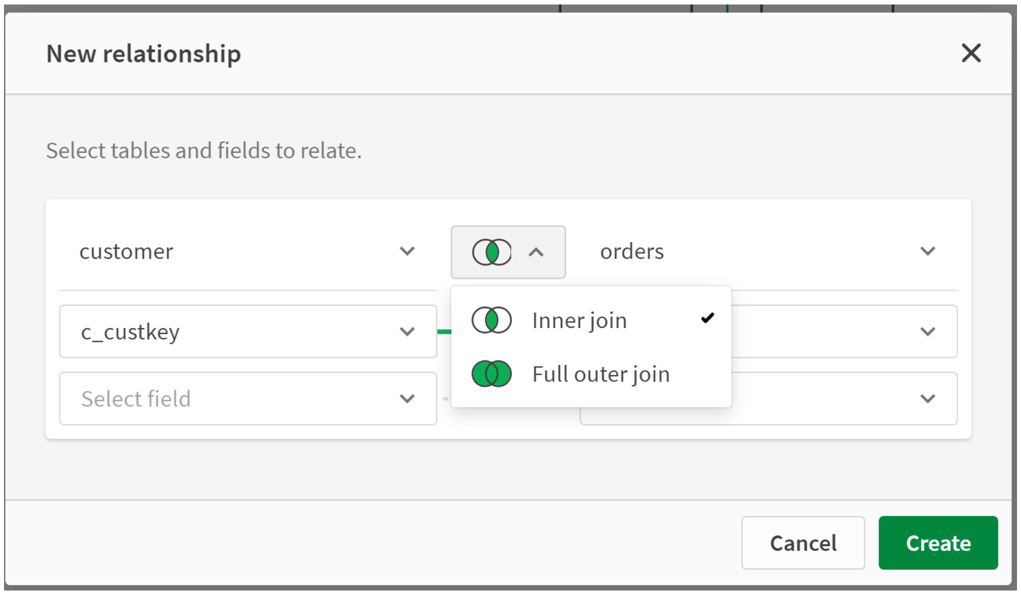


c_custkey	c_name	c_address	c_nationkey	c_phone	c_acctb
412445	Customer#000412445	0QAB3OjYnbP6mA0B,kgf	21	31-421-403-4333	5358.33
412446	Customer#000412446	5u8MSbyiC7J,7PuY4Ivaq1JRb...	20	30-487-949-7942	9441.59
412447	Customer#000412447	HC4ZT62gKPgrjr ceoaZgFOunl...	7	17-797-466-6308	7868.75
412448	Customer#000412448	hJok1MMrDgH	6	16-541-510-4964	6060.98
412449	Customer#000412449	zAt1nZNG01gOhIqgyDtDa S,Y0...	14	24-710-983-5536	4973.84
412450	Customer#000412450	fUD6IoGdtF	20	30-293-696-5047	4406.28
412451	Customer#000412451	W2Ge0Qd8adH	20	30-590-724-6711	2290.38
412452	Customer#000412452	Ij4xiPIeNEP1uR5p7H	10	20-492-590-3363	3426.64

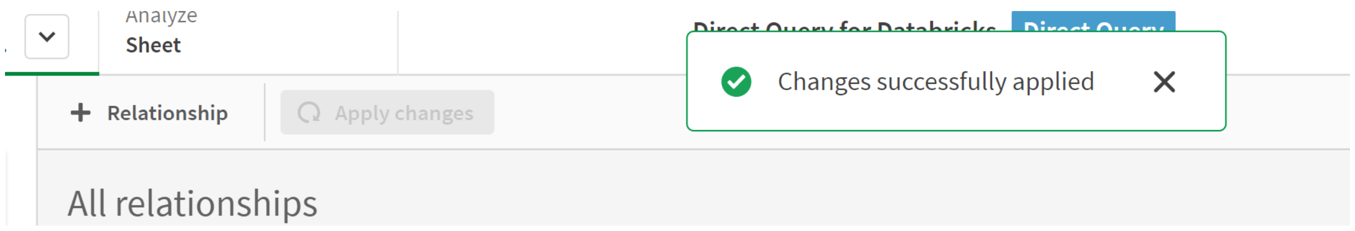
7. In the next screen, you need to define the relationships between your tables. Differently from in-memory applications, Qlik Engine will not suggest the correct way of create the relations between every table



8. For every New Relationship you need to define the left and right table, which fields are the keys and relationship type (Inner Join / Full outer join)



9. Create all your relationships and press **Apply changes** button waiting for the confirmation message



10. Now your model is ready to create your dashboard sheets. Press **Analyze Sheet** button

It is highly recommended you keep your model as simple as possible to avoid complex SQL queries being generated

Creating a Direct Query Application sheet

You can design your sheets as you do in a regular Qlik application with some differences. Keep in mind that some features like “Smart Search”, “Insight Advisor”, “Selections Tool” are disabled, and the user can only filter on a field that is part of an object (filter pane, chart or table).

It is recommended you add as many dimensions as you expect your users to filter to a filter pane to make their analysis easier

Appendix: Connecting Qlik to Databricks (QSE and Qlik SaaS)

The Qlik Sense Databricks Connector connects to Databricks clusters or **Databricks SQL Warehouse** to query data in tables. **Databricks SQL Warehouse** are special Databricks clusters, dedicated to serving BI/ad hoc analytics queries. While these clusters offer access to the same data visible to conventional clusters, they isolate their workloads, allowing greater concurrency. These clusters are provisioned based on small, medium, and large sizes avoiding the need to specify numbers and types of master and worker nodes.

From a cost x performance perspective (especially if you have applications with Dynamic Views and ODAG) it is recommended using **Databricks SQL Warehouse**

Authentication Methods

To connect to Databricks, Qlik Sense currently supports the methods below to authenticate against Databricks. For this method, we are using the “User name / Password” option. For more details about other methods please refer to the https://help.qlik.com documentation

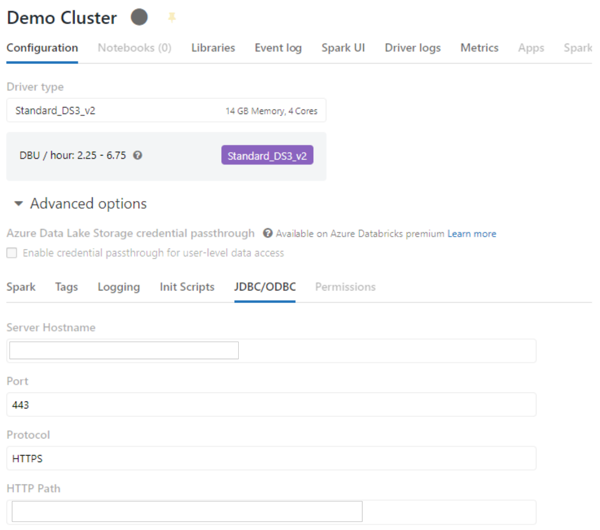
- User name
- User name and password
- No authentication
- Azure AD OAuth

Obtaining Databricks credentials and connection details
To create a data connection in Replicate to a Databricks instance you will need:

- Server Hostname
- Port
- HTTP Path
- Token

- Clusters

  Information 1,2 and 3 can be extracted from Databricks console going to the cluster configuration or your SQL Endpoint configuration and under the **Advanced Options** section you will find the **JDBC/ODBC** tab



Driver type: Standard_DS3_v2 — 14 GB Memory, 4 Cores

DBU / hour: 2.25 - 6.75 — Standard_DS3_v2

Azure Data Lake Storage credential passthrough — Available on Azure Databricks premium Learn more

☐ Enable credential passthrough for user-level data access

Server Hostname:

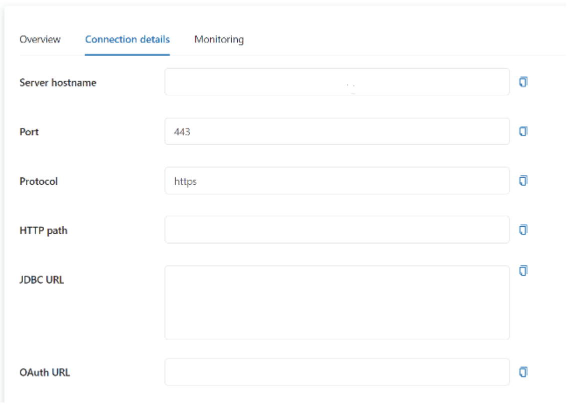
Port: 443

Protocol: HTTPS

HTTP Path:

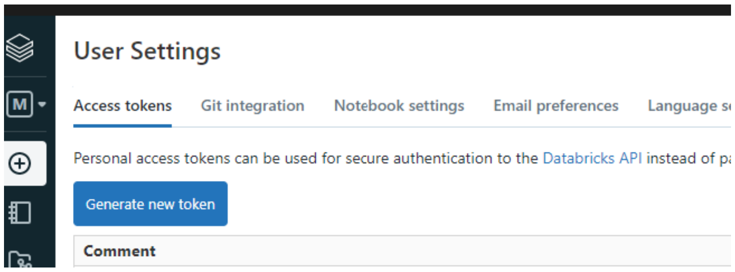
- Databricks SQL Warehouse

  Information 1,2 and 3 can be extracted from Databricks console going to the **SQL Warehouse** section under the "Connection details" tab

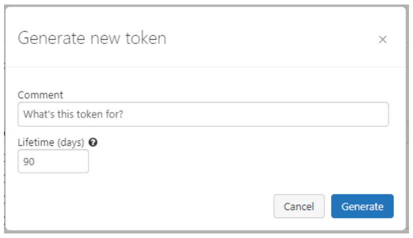


Server hostname	
Port	443
Protocol	https
HTTP path	
JDBC URL	
OAuth URL	

To create an access token, you need to go to **User Settings** section of your Databricks console and use the **Generate new token** button



Personal access tokens can be used for secure authentication to the Databricks API instead of pa



It is important to store the generated token in a safe place, because you cannot retrieve it after you close the screen

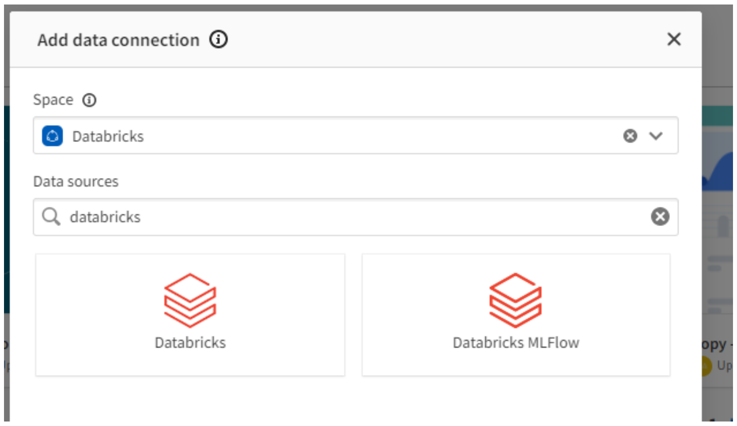
Creating the Data Connection

In your Qlik Cloud Hub there is a button **+ Add new** on the top right of your screen. After clicking that option you are going to see one option **Connect to data** Establish a new data connection .

⚠ *If you cannot see the options above, please contact your tenant admin to check your roles.*

Qlik Cloud will show you an interface to create a Data Connection. In the Data sources you can search for 'databricks'.

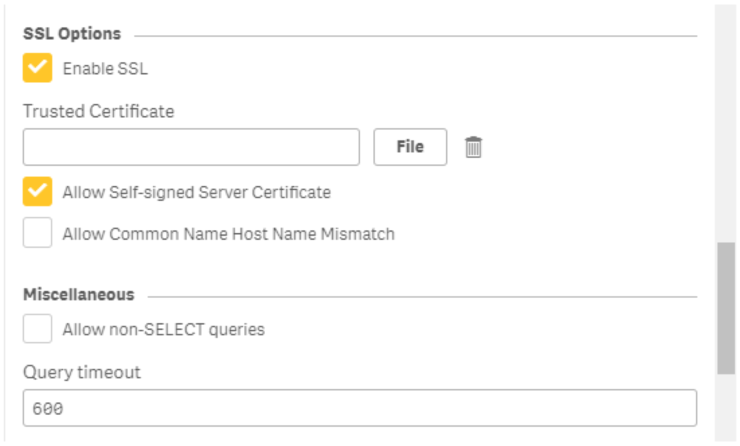
Choose carefully which space you are creating the data connection in, regarding your security plan (available only to yourself, to a dev group, to production, etc)



Now you´re going to see a form to configure your data connection. The first blocks is compound of

Database property	Description	Required
Host name	The IP address or host name of the Databricks server.	Yes
Port	Server port for the Databricks database.	Yes
Database	The name of the Databricks database.	Yes
HTTP Path	Databricks compute resources URL.	Yes

a) Those parameters were collected at step 1 *"Obtaining Databricks credentials and connection details"* with exception of your database name (you need to check your Databricks console so see which databases are available.
b) For **Authentication** use **User Name And Password** option
c) In the **Credentials** section use token as **User Name** and paste your token you created at step 1 as the **Password**
d) For **SSL Options**, check the **Enable SSL** and **Allow Self-signed server Certificate**
e) For **Query Timeout** allow at least 10 min (600 seconds) as timeout. This is to prevent timeout failures when you try to connect to a fully terminated cluster or SQL Endpoint giving time to it to bring back to fully operational status.


SSL Options

✓ Enable SSL

Trusted Certificate

✓ Allow Self-signed Server Certificate

☐ Allow Common Name Host Name Mismatch

Miscellaneous

☐ Allow non-SELECT queries

Query timeout: 600

f) For **Load Optimization** check the option **Enable Bulk Reader**. This will include larger portions of data in the iterations within a load. This may result in faster load times for larger datasets. If not selected, data will be loaded row by row.

Conclusion

This document showcased many integration options and best practices for using the Qlik Sense Analytics Platform with the Databricks. The document discussed high level concepts, practical applications, and most importantly strategies on how to combine Qlik and Databricks best to optimize analytics at your organization. This document will be updated as new capabilities are added to both the Databricks engine and Qlik Sense platform.