Discover the secrets to supercharging your analytics with Microsoft Fabric Warehouse

Brian Bønk Rueløkke

Principal & Enterprise architect, Data & AI

Fellowmind

https://linkedin.com/in/brianbonk
https://brianbonk.dk
https://github.com/brianbonk

Microsoft Most Valuable Professional

Microsoft
FastTrack Recognized
Solution Architect
Power BI
2022 >>

Microsoft
Certified Trainer
Data Platform

2018 >>

AN OVERVIEW OF FABRIC AFTER BUILD

Intelligent data foundation

Data Factory	Data Engineering	Data Warehouse	Data Science	Real-Time Intelligence	Power BI	Industry Solutions

Powered by AI with Copilot in Microsoft Fabric

Catalog for data in motion
Real-Time Hub

Unified data foundation
OneLake

Intelligent data foundation

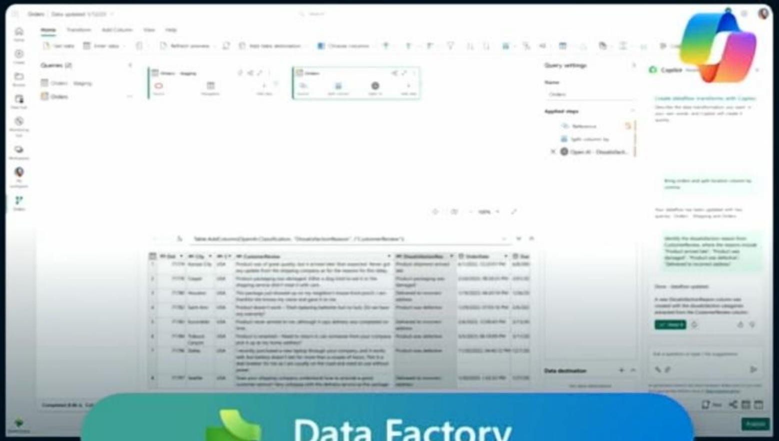
- Data Factory
- Data Engineering
- Data Warehouse
- Data Science
- Real-Time Intelligence
- Power BI
- Industry Solutions

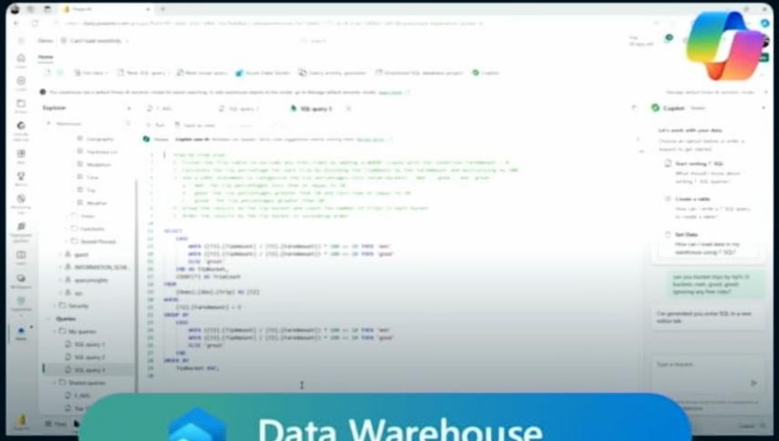
Powered by AI with Copilot in Microsoft Fabric

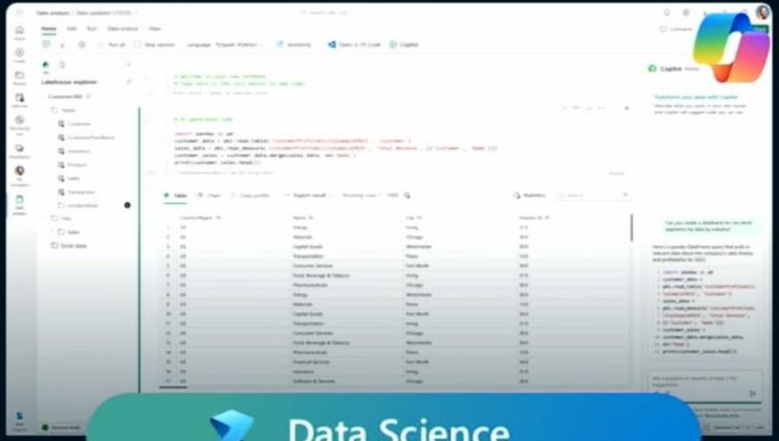
Catalog for data in motion
Real-Time Hub

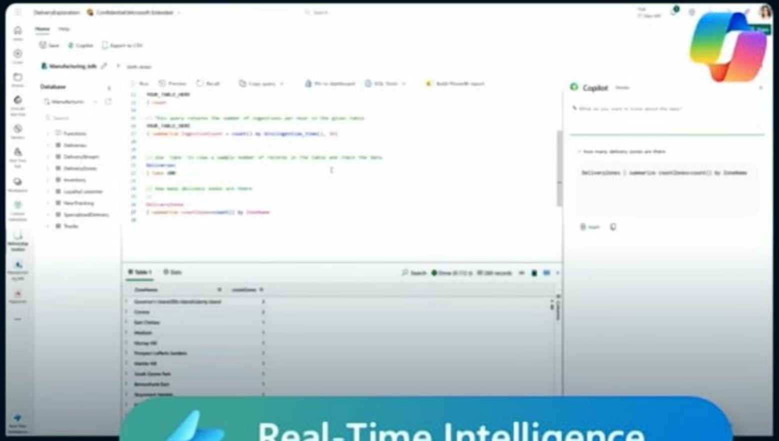
Unified data foundation
OneLake

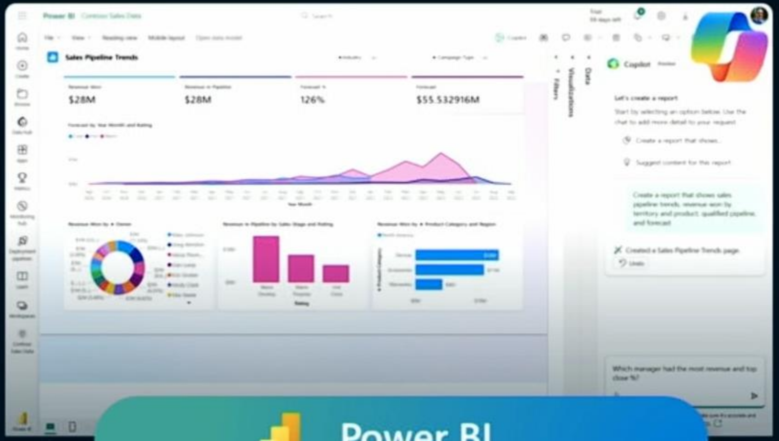
Copilot Integrated in every Microsoft Fabric Experience


Data Factory


Data Warehouse

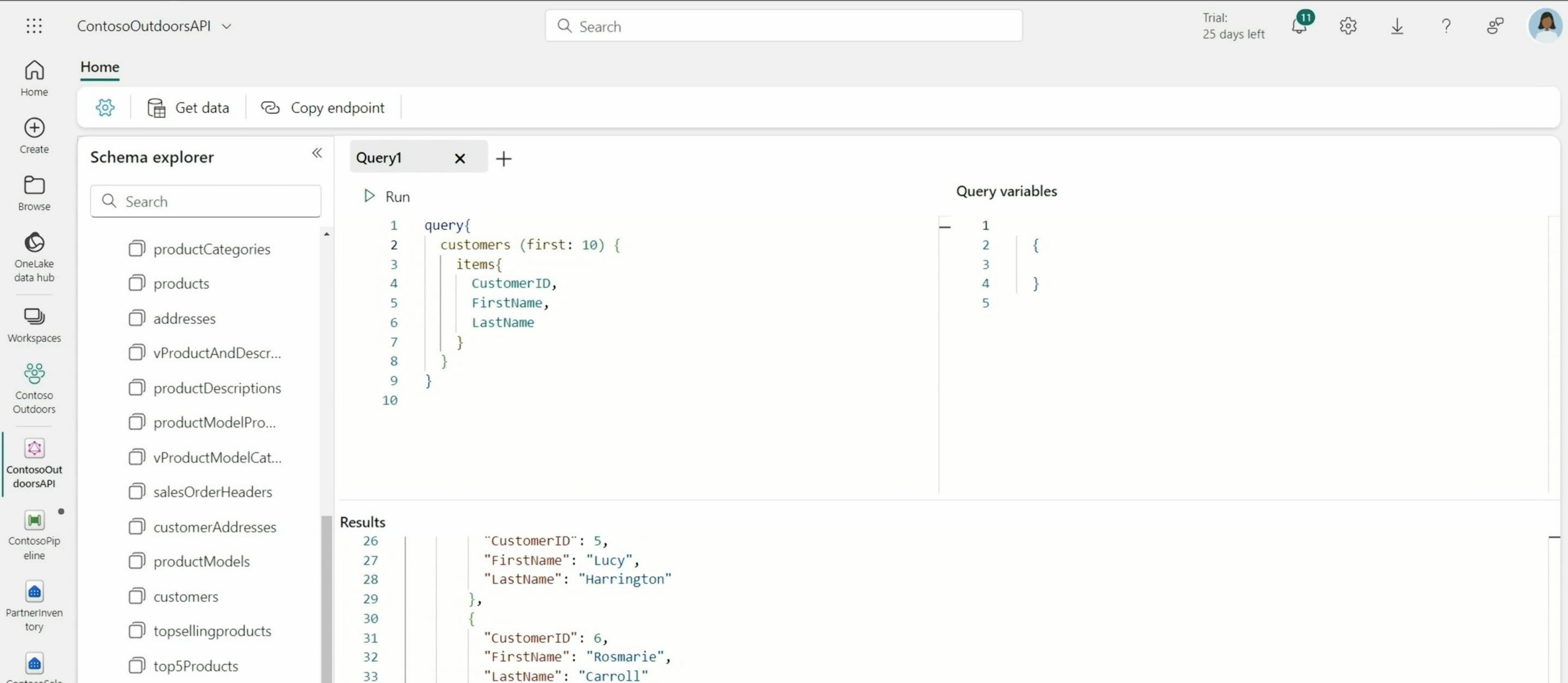

Data Science


Real-Time Intelligence



Power BI

API for GraphQL

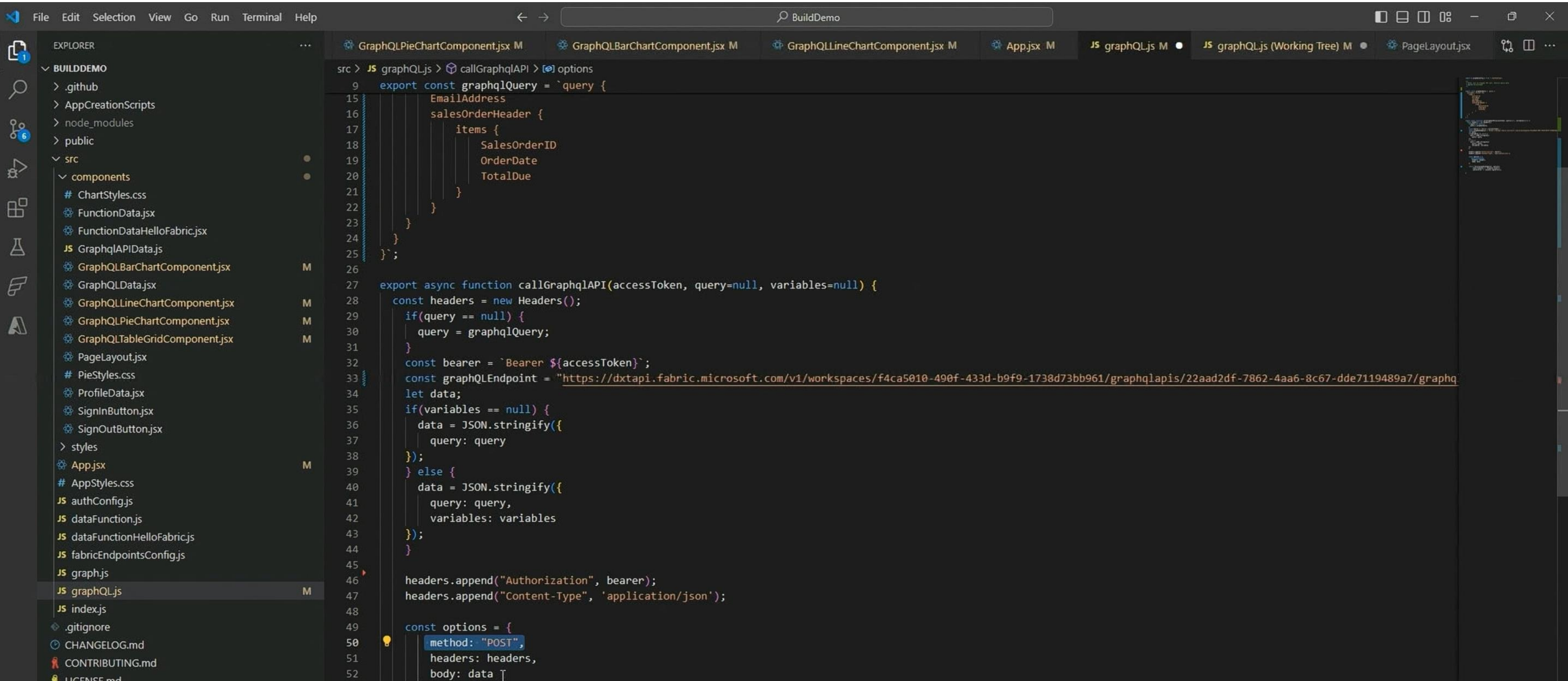


```
query{
  customers (first: 10) {
    items{
      CustomerID,
      FirstName,
      LastName
    }
  }
}
```

```
{

}
```

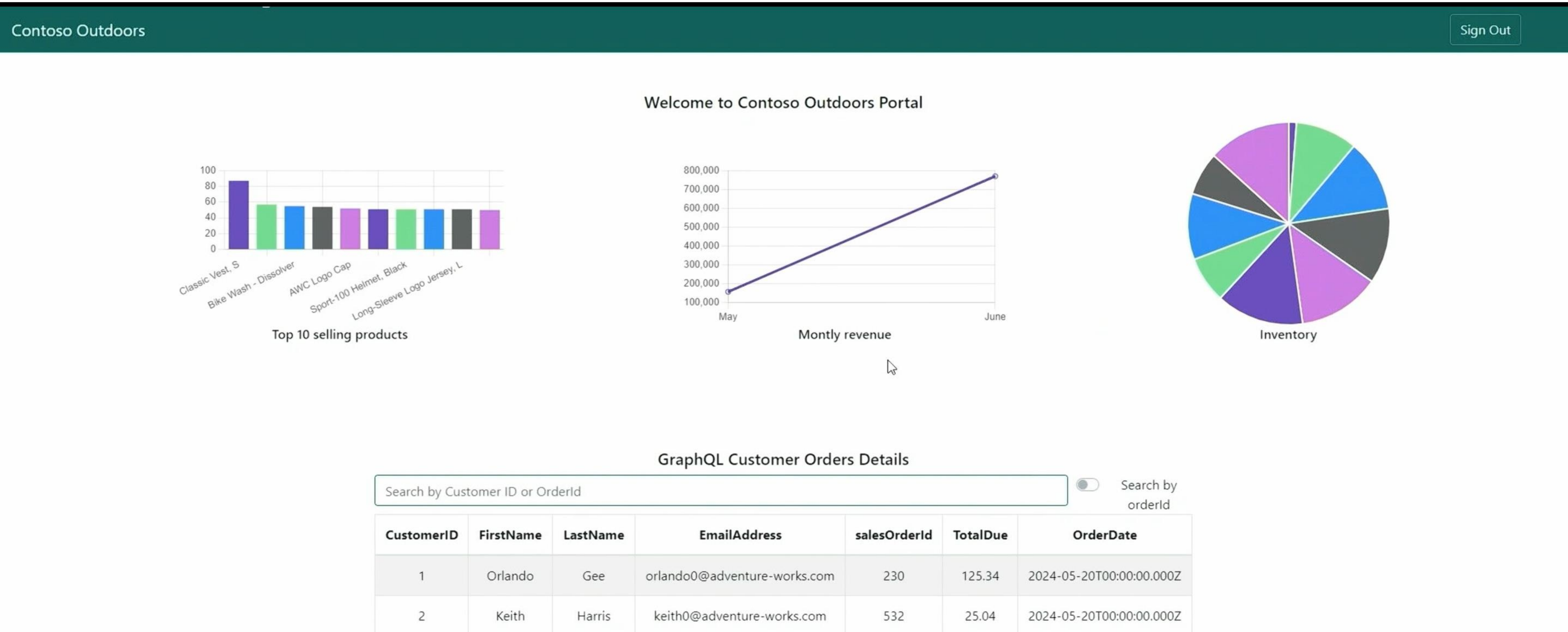
```
        "CustomerID": 5,
        "FirstName": "Lucy",
        "LastName": "Harrington"
      },
      {
        "CustomerID": 6,
        "FirstName": "Rosmarie",
        "LastName": "Carroll"
```

API for GraphQL



```
export const graphqlQuery = `query {
        EmailAddress
        salesOrderHeader {
          items {
            SalesOrderID
            OrderDate
            TotalDue
          }
        }
      }
    }
  }`;

export async function callGraphqlAPI(accessToken, query=null, variables=null) {
  const headers = new Headers();
    if(query == null) {
      query = graphqlQuery;
    }
    const bearer = `Bearer ${accessToken}`;
    const graphQLEndpoint = "https://dxtapi.fabric.microsoft.com/v1/workspaces/f4ca5010-490f-433d-b9f9-1738d73bb961/graphqlapis/22aad2df-7862-4aa6-8c67-dde7119489a7/graphq
    let data;
    if(variables == null) {
      data = JSON.stringify({
        query: query
    });
    } else {
      data = JSON.stringify({
        query: query,
        variables: variables
    });
    }

    headers.append("Authorization", bearer);
    headers.append("Content-Type", 'application/json');

    const options = {
      method: "POST",
      headers: headers,
      body: data
```

API for GraphQL



Contoso Outdoors

Sign Out

Welcome to Contoso Outdoors Portal

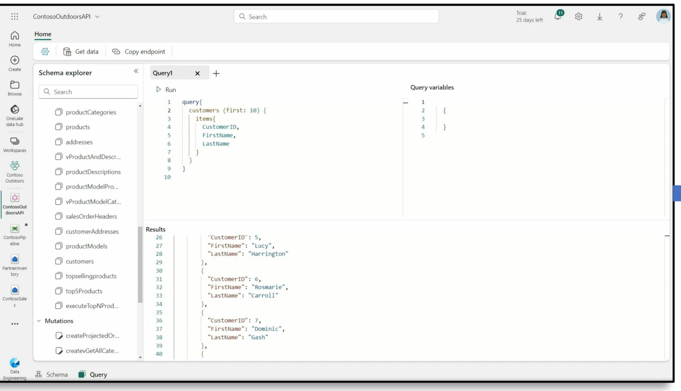
GraphQL Customer Orders Details

Search by Customer ID or OrderId

Search by orderId

CustomerID	FirstName	LastName	EmailAddress	salesOrderId	TotalDue	OrderDate
1	Orlando	Gee	orlando0@adventure-works.com	230	125.34	2024-05-20T00:00:00.000Z
2	Keith	Harris	keith0@adventure-works.com	532	25.04	2024-05-20T00:00:00.000Z

API for GraphQL

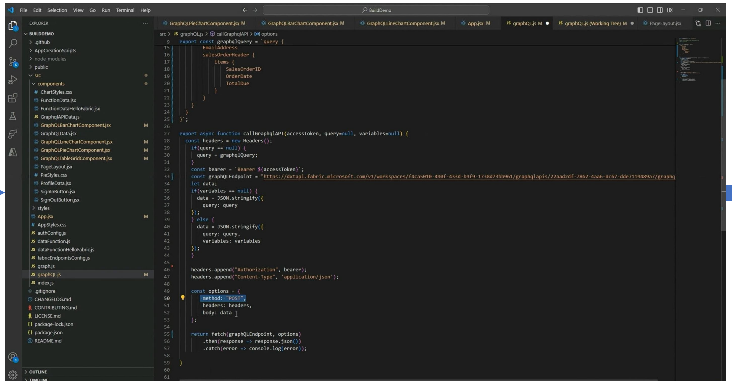


```
query{
  customers (first: 10) {
    items{
      CustomerID,
      FirstName,
      LastName
    }
  }
}
```

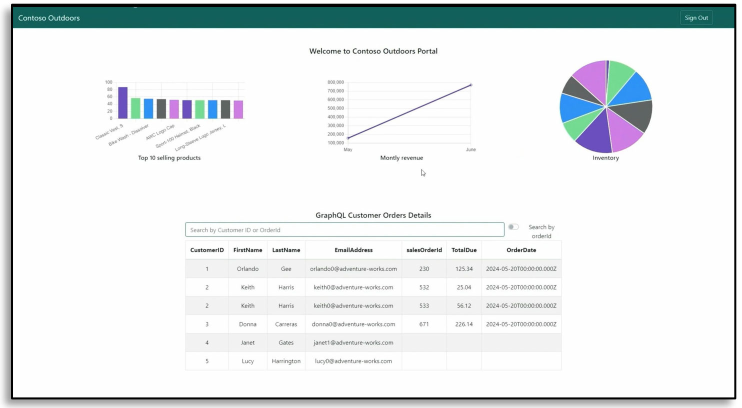
```
{

}
```

Results

```
      "CustomerID": 5,
      "FirstName": "Lucy",
      "LastName": "Harrington"
    },
    {
      "CustomerID": 6,
      "FirstName": "Rosmarie",
      "LastName": "Carroll"
    },
    {
      "CustomerID": 7,
      "FirstName": "Dominic",
      "LastName": "Gash"
    },
    {
```


BuildDemo
GraphQLPieChartComponent.jsx M
GraphQLBarChartComponent.jsx M
GraphQLLineChartComponent.jsx M
App.jsx M
graphQL.js M
graphQL.js (Working Tree) M
PageLayout.jsx
src > graphQL.js > callGraphqlAPI > options

```
export const graphqlQuery = `query {
        EmailAddress
        salesOrderHeader {
          items {
            SalesOrderID
            OrderDate
            TotalDue
          }
        }
      }
    }
}`;

export async function callGraphqlAPI(accessToken, query=null, variables=null) {
  const headers = new Headers();
    if(query == null) {
      query = graphqlQuery;
    }
    const bearer = `Bearer ${accessToken}`;
    const graphQLEndpoint = "https://dxtapi.fabric.microsoft.com/v1/workspaces/f4ca5010-490f-433d-b9f9-1738d73bb961/graphqlapis/22aad2df-7862-4aa6-8c67-dde7119489a7/graphq
    let data;
    if(variables == null) {
      data = JSON.stringify({
        query: query
      });
    } else {
      data = JSON.stringify({
        query: query,
        variables: variables
      });
    }

    headers.append("Authorization", bearer);
    headers.append("Content-Type", 'application/json');

    const options = {
      method: "POST",
      headers: headers,
      body: data
    };

    return fetch(graphQLEndpoint, options)
      .then(response => response.json())
      .catch(error => console.log(error));

}
```

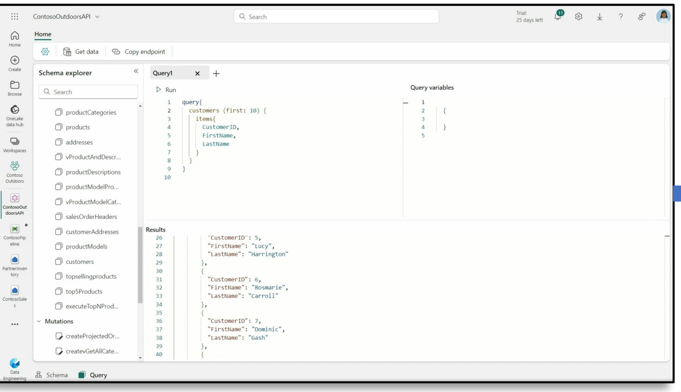


Welcome to Contoso Outdoors Portal

GraphQL Customer Orders Details

CustomerID	FirstName	LastName	EmailAddress	salesOrderId	TotalDue	OrderDate
1	Orlando	Gee	orlando0@adventure-works.com	230	125.34	2024-05-20T00:00:00.000Z
2	Keith	Harris	keith0@adventure-works.com	532	25.04	2024-05-20T00:00:00.000Z
2	Keith	Harris	keith0@adventure-works.com	533	56.12	2024-05-20T00:00:00.000Z
3	Donna	Carreras	donna0@adventure-works.com	671	226.14	2024-05-20T00:00:00.000Z
4	Janet	Gates	janet1@adventure-works.com			
5	Lucy	Harrington	lucy0@adventure-works.com			

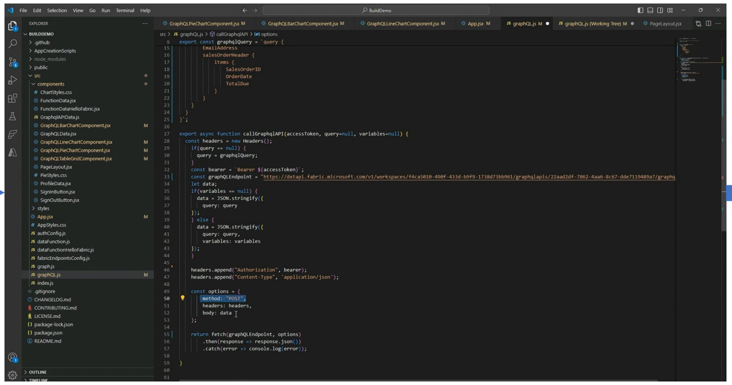
API for GraphQL



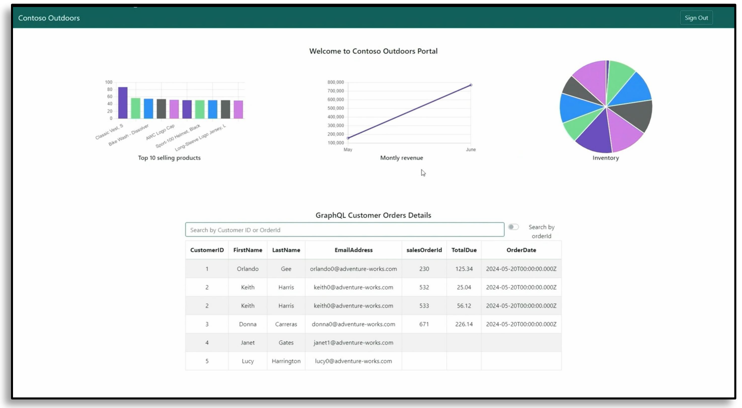
```
query{
  customers (first: 10) {
    items{
      CustomerID,
      FirstName,
      LastName
    }
  }
}
```

Results

```
      "CustomerID": 5,
      "FirstName": "Lucy",
      "LastName": "Harrington"
    },
    {
      "CustomerID": 6,
      "FirstName": "Rosmarie",
      "LastName": "Carroll"
    },
    {
      "CustomerID": 7,
      "FirstName": "Dominic",
      "LastName": "Gash"
    },
    {
```


BuildDemo
GraphQLPieChartComponent.jsx M
GraphQLBarChartComponent.jsx M
GraphQLLineChartComponent.jsx M
App.jsx M
src > graphQL.js > callGraphqlAPI > options

```
export const graphqlQuery = `query {
          EmailAddress
          salesOrderHeader {
            items {
              SalesOrderID
              OrderDate
              TotalDue
            }
          }
        }
      }
}`;

export async function callGraphqlAPI(accessToken, query=null, variables=null) {
  const headers = new Headers();
    if(query == null) {
      query = graphqlQuery;
    }
    const bearer = `Bearer ${accessToken}`;
    const graphQLEndpoint = "https://dxtapi.fabric.microsoft.com/v1/workspaces/f4ca5010-490f-433d-b9f9-1738d73b
    let data;
    if(variables == null) {
      data = JSON.stringify({
        query: query
    });
    } else {
      data = JSON.stringify({
        query: query,
        variables: variables
    });
    }

    headers.append("Authorization", bearer);
    headers.append("Content-Type", 'application/json');

    const options = {
      method: "POST",
      headers: headers,
      body: data
    };

    return fetch(graphQLEndpoint, options)
      .then(response => response.json())
      .catch(error => console.log(error));

}
```



GraphQL Customer Orders Details

Search by Customer ID or OrderId — Search by orderId

CustomerID	FirstName	LastName	EmailAddress	salesOrderId	TotalDue	OrderDate
1	Orlando	Gee	orlando0@adventure-works.com	230	125.34	2024-05-20T00:00:00.000Z
2	Keith	Harris	keith0@adventure-works.com	532	25.04	2024-05-20T00:00:00.000Z
2	Keith	Harris	keith0@adventure-works.com	533	56.12	2024-05-20T00:00:00.000Z
3	Donna	Carreras	donna0@adventure-works.com	671	226.14	2024-05-20T00:00:00.000Z
4	Janet	Gates	janet1@adventure-works.com			
5	Lucy	Harrington	lucy0@adventure-works.com			

Takeaways

API for GraphQL in Fabric

Bridging the gap between data and applications

VS VS VS

Synapse Data Warehouse

Synapse Serverless SQL

Synapse Dedicated pool

Databricks

Synapse Data Warehouse VS Synapse Serverless SQL VS Synapse Dedicated pool VS Databricks

Storage: Fabric focuses on a single storage solution, the Unified Data Location, built on Azure Data Lake Storage Gen2. This eliminates the need for role-based access control and simplifies data management for organizations.

Integration: Seamless integration with multiple Azure services, including Azure Data Factory, Azure Synapse Analytics, Synapse Real-Time Analytics, Synapse Data Warehousing, Power BI and Data Explorer, with automatic provisioning of the underlying hardware.

Collaboration: Dedicated workspaces enable diverse developers – including data engineers and data scientists – to collaborate effortlessly.

Synapse Data Warehouse VS Synapse Serverless SQL VS Synapse Dedicated pool VS Databricks

A unified workspace: Synapse Analytics provides a unified workspace where data engineers and data scientists can collaborate on big data and SQL-based analytics tasks.

Real-time analysis: The integration of Apache Spark and dedicated SQL pool enables users to perform real-time analysis - on both structured and unstructured data.

Security and Governance: Synapse Analytics provides robust security features and fine-grained access controls to ensure data is protected.

Synapse Data Warehouse VS Synapse Serverless SQL VS Synapse Dedicated pool VS Databricks

Scalability: Databricks can easily handle large data tasks as it has the ability to scale horizontally.

Notebooks: Interactive notebooks allow users to execute and visualize code in real time, facilitating data exploration and model training.

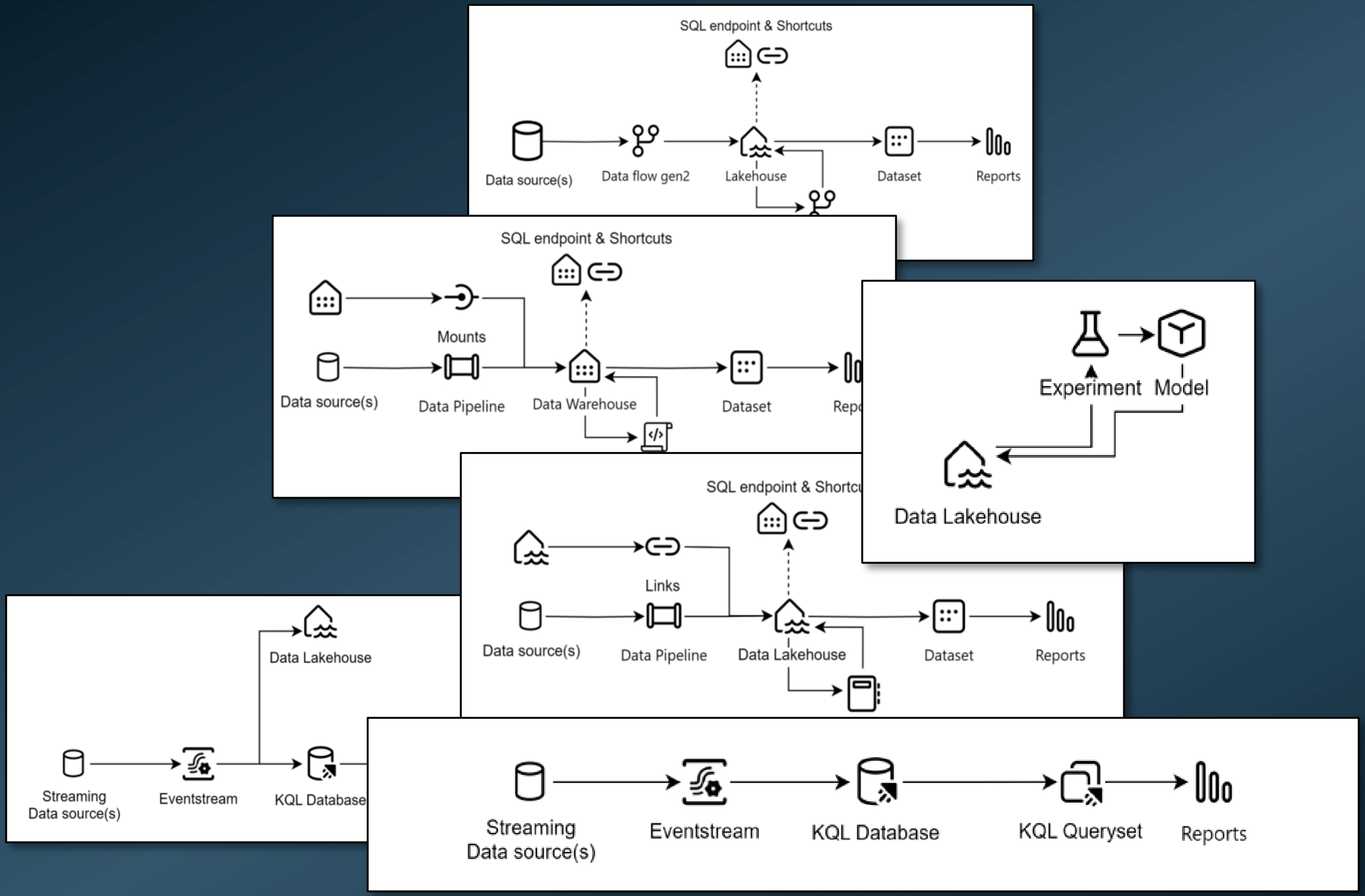
Integration: Seamless integration with other Azure services such as Azure Storage and Azure Data Lake Storage simplifies the process of data import and extraction.

Data Warehouse

Lakehouse

Warehouse	Lakehouse
Structured	Structured, Semi structured, Unstructured
SQL language	Notebooks
Schemas, Tables	Folders, Files and tables
OLS, RLS, CLS	RLS, TLS
T-SQL	Spark (Scala, PySpark, SparkSQL)

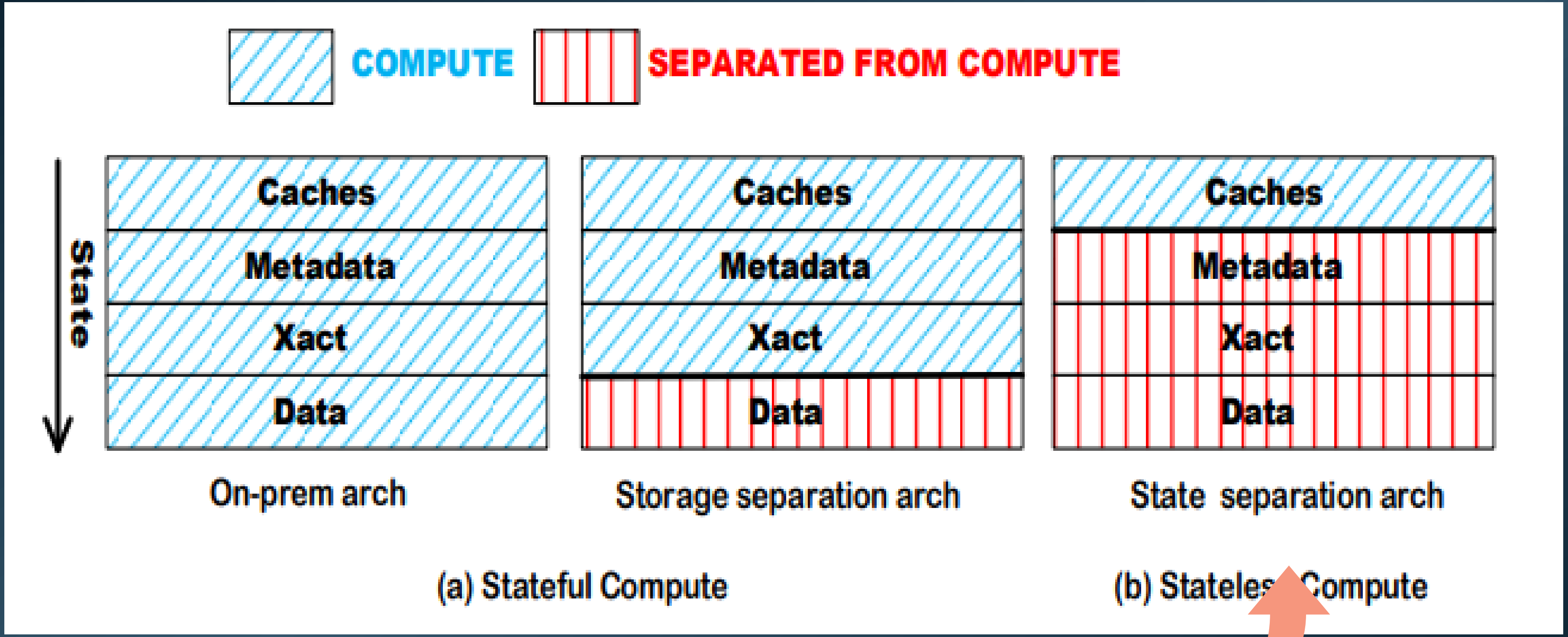
Data Warehouse



Data
Warehouse

The Polaris engine

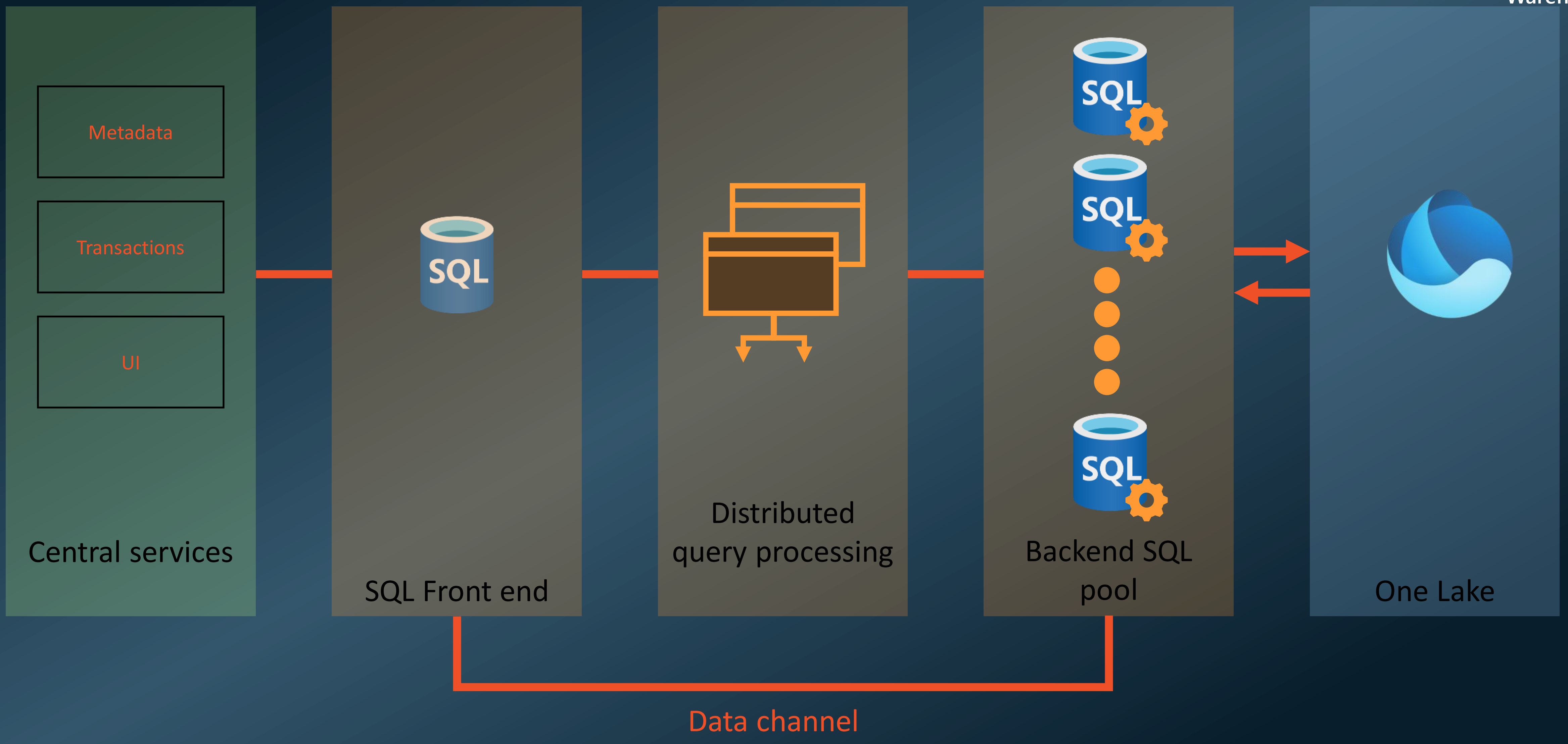
Stateless compute



Synapse and Fabric

The Polaris engine

Service architecture



Data Warehouse

T-SQL Surface Area

The T-SQL Surface area covers the supported syntax within both the Warehouse & the Lakehouse SQL Endpoint

Currently unsupported syntax (and the potential impact...)

🔴	🟡	🟢
ALTER TABLE... ...<ADD/ALTER/DROP>	CREATE ROLE/USER	BULK LOAD
Identity Columns	MERGE	SP_SHOWSPACEUSED
TRUNCATE	Recursive Queries	SET TRANSACTION ISOLATION LEVEL

Transactions & Isolation Levels

Transactions are included (including multi-table)

Only **Snapshot Isolation** supported (Optimistic Concurrency)

Locks are at the **table level**

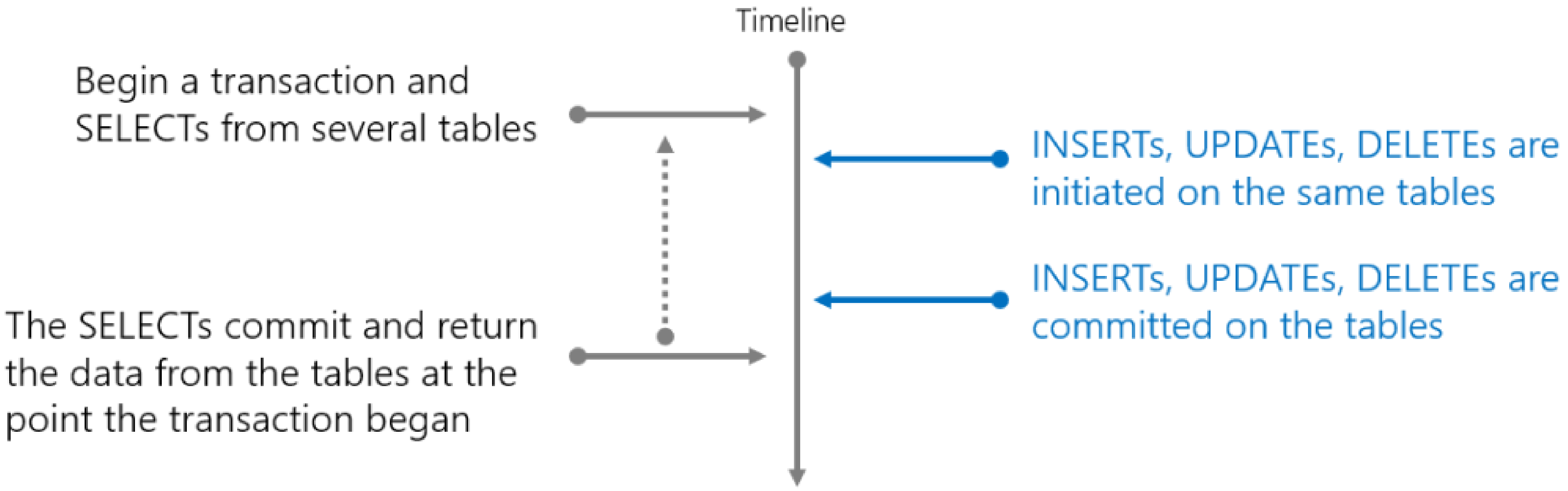
Use **sys.dm_tran_locks** to show current locks

Schema Stability
Sch-S
→ **SELECT**

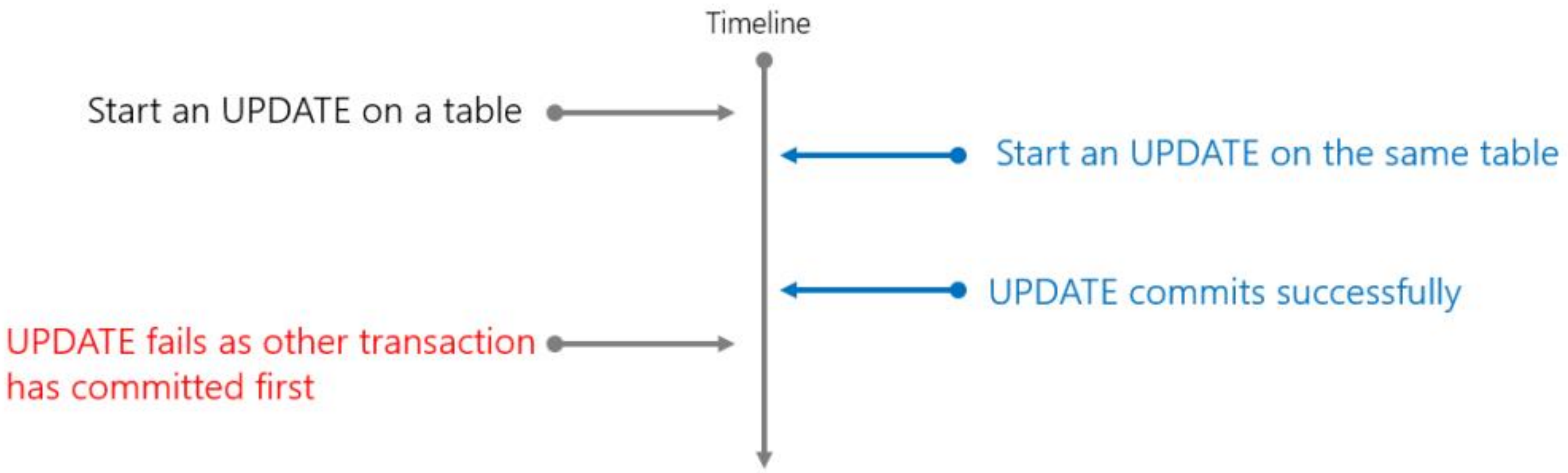
Intent Exclusive
IX
→ **INSERT**
DELETE
UPDATE
COPY INTO

Schema Modification
Sch-M
→ **DDL**

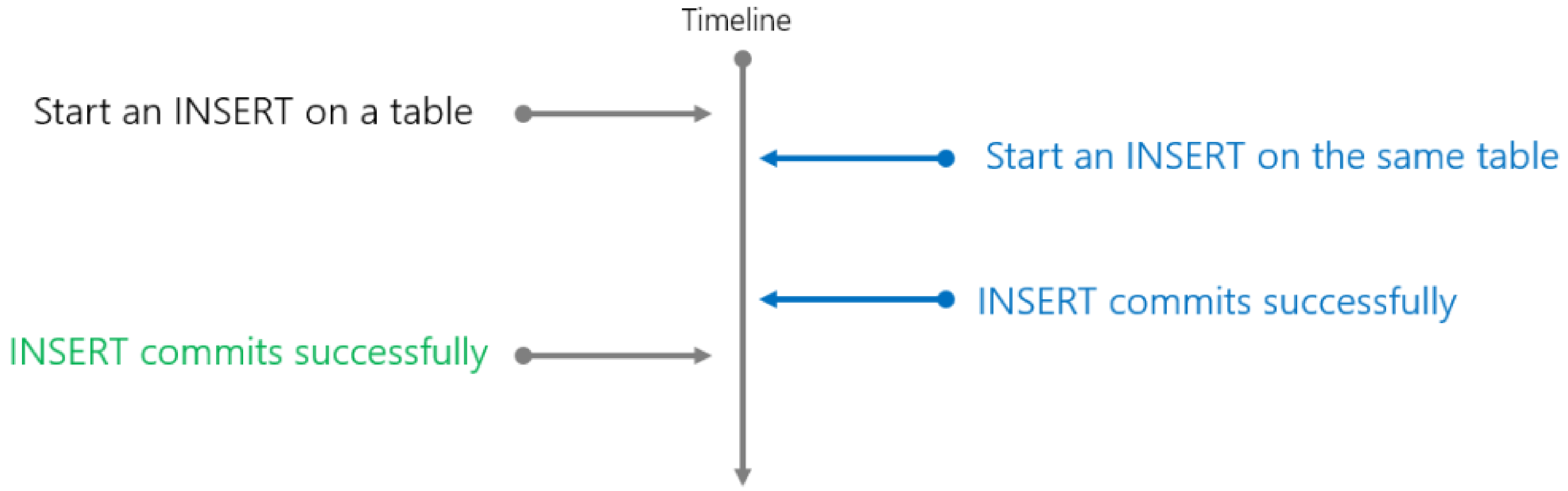
Transactions & Isolation Levels


Timeline
Begin a transaction and SELECTs from several tables
The SELECTs commit and return the data from the tables at the point the transaction began
INSERTs, UPDATEs, DELETEs are initiated on the same tables
INSERTs, UPDATEs, DELETEs are committed on the tables

Transactions & Isolation Levels


Timeline
Start an UPDATE on a table
Start an UPDATE on the same table
UPDATE commits successfully
UPDATE fails as other transaction has committed first

Transactions & Isolation Levels


Timeline
Start an INSERT on a table
Start an INSERT on the same table
INSERT commits successfully
INSERT commits successfully

Ingestion: Loading the Warehouse

Data Warehouse

Code

SQL **COPY INTO…** <from Azure storage>

SQL **CREATE TABLE AS…SELECT**

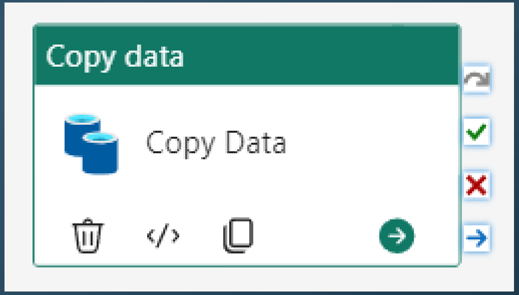
SQL **INSERT INTO…VALUES / SELECT**

Pyodbc/JDBC (from Notebooks)

Low/No-Code

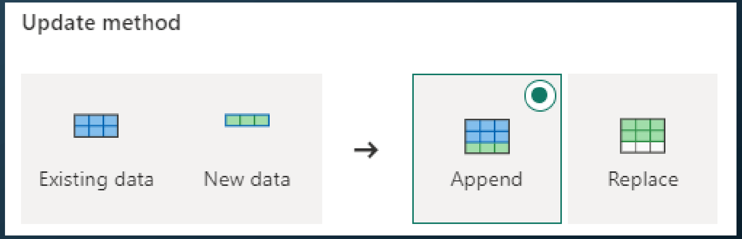
Pipelines

"Brute force"



Dataflows Gen2

(Power Query)

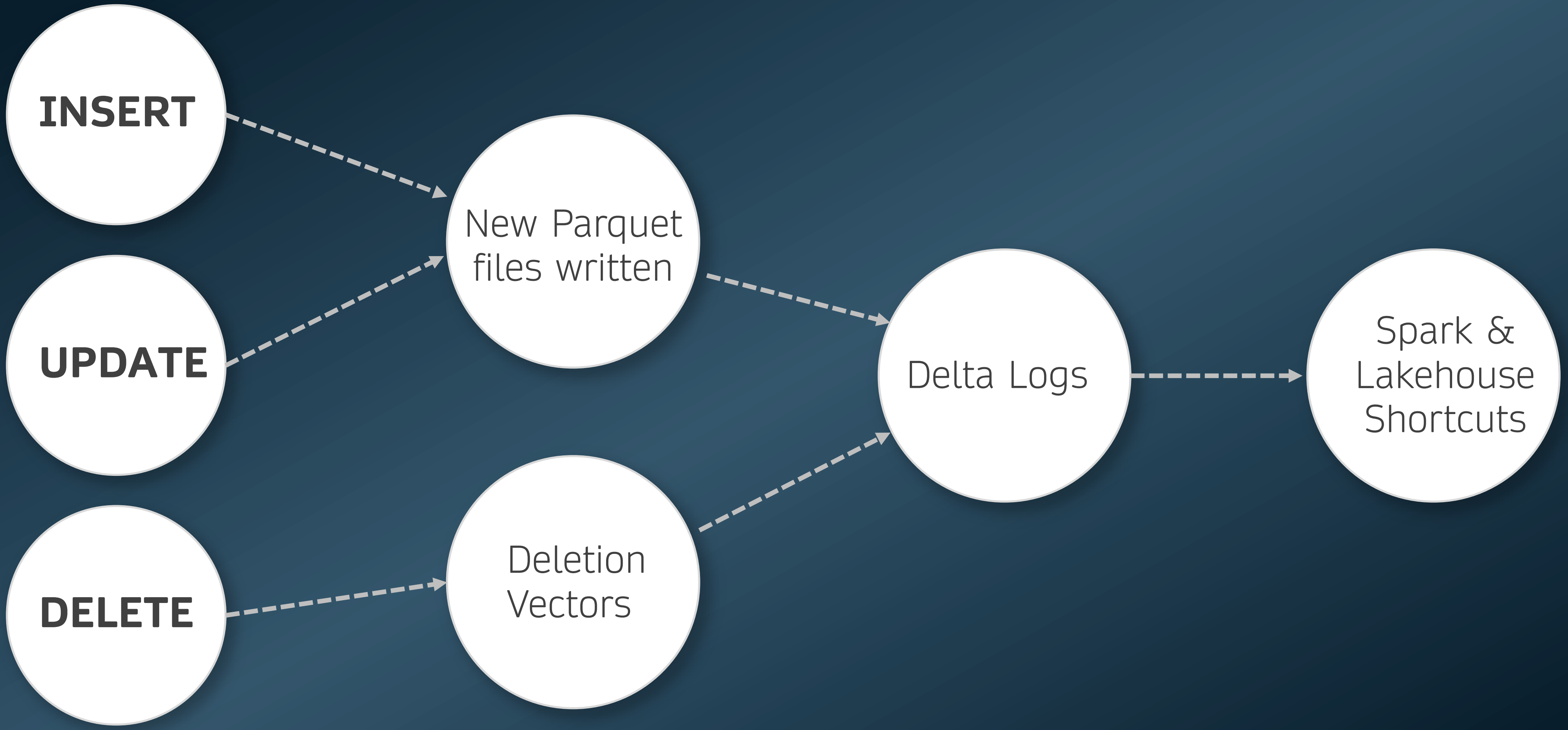

Update method

Existing data · New data → [✓ Append] · Replace

Ingestion: Loading the Warehouse

Data Warehouse

All SQL operations are performed on Parquet files in OneLake



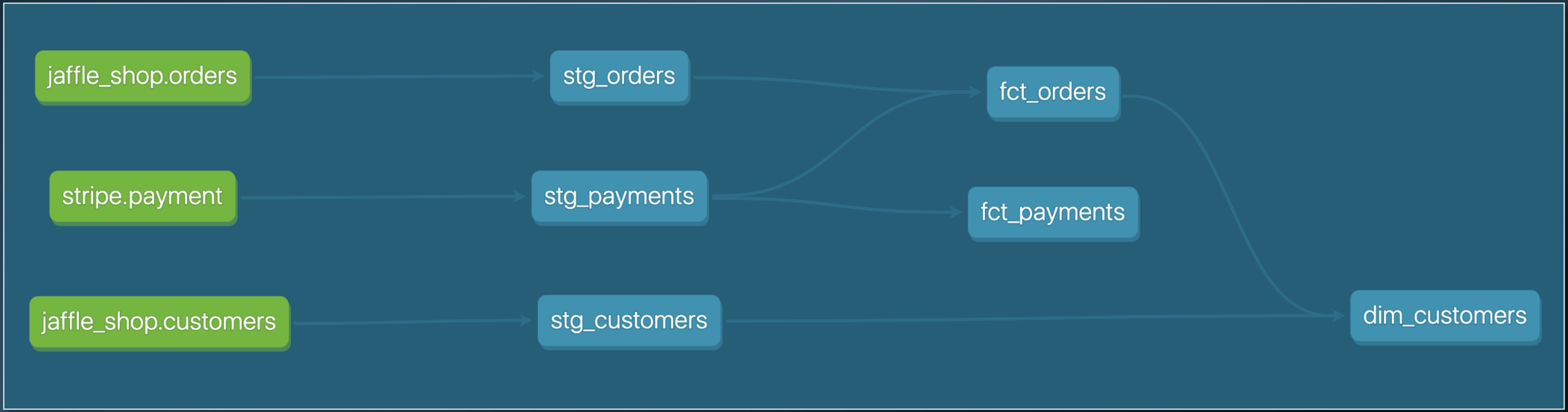
Ingestion: Loading the Warehouse

Data Warehouse

dbt (including cloud) support in Warehouse

Declare your SQL loading processes

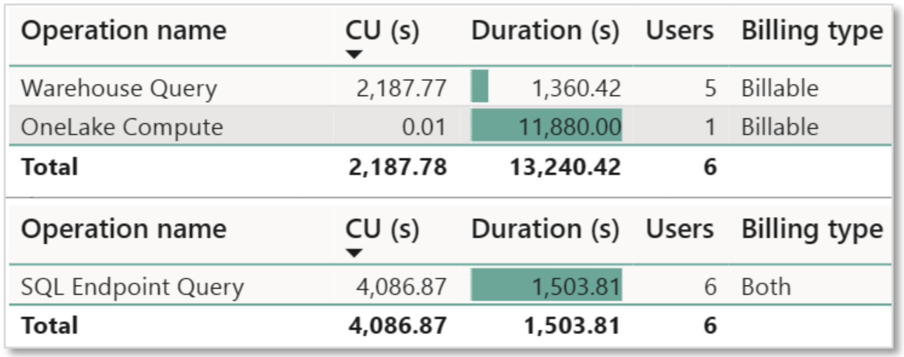
Create DAG for loading tables based on dependencies



Warehouse Utilization

Usage is tracked by Capacity Unit seconds (consumed by read and write activity against the Warehouse and reads against Lakehouse SQL Endpoint)

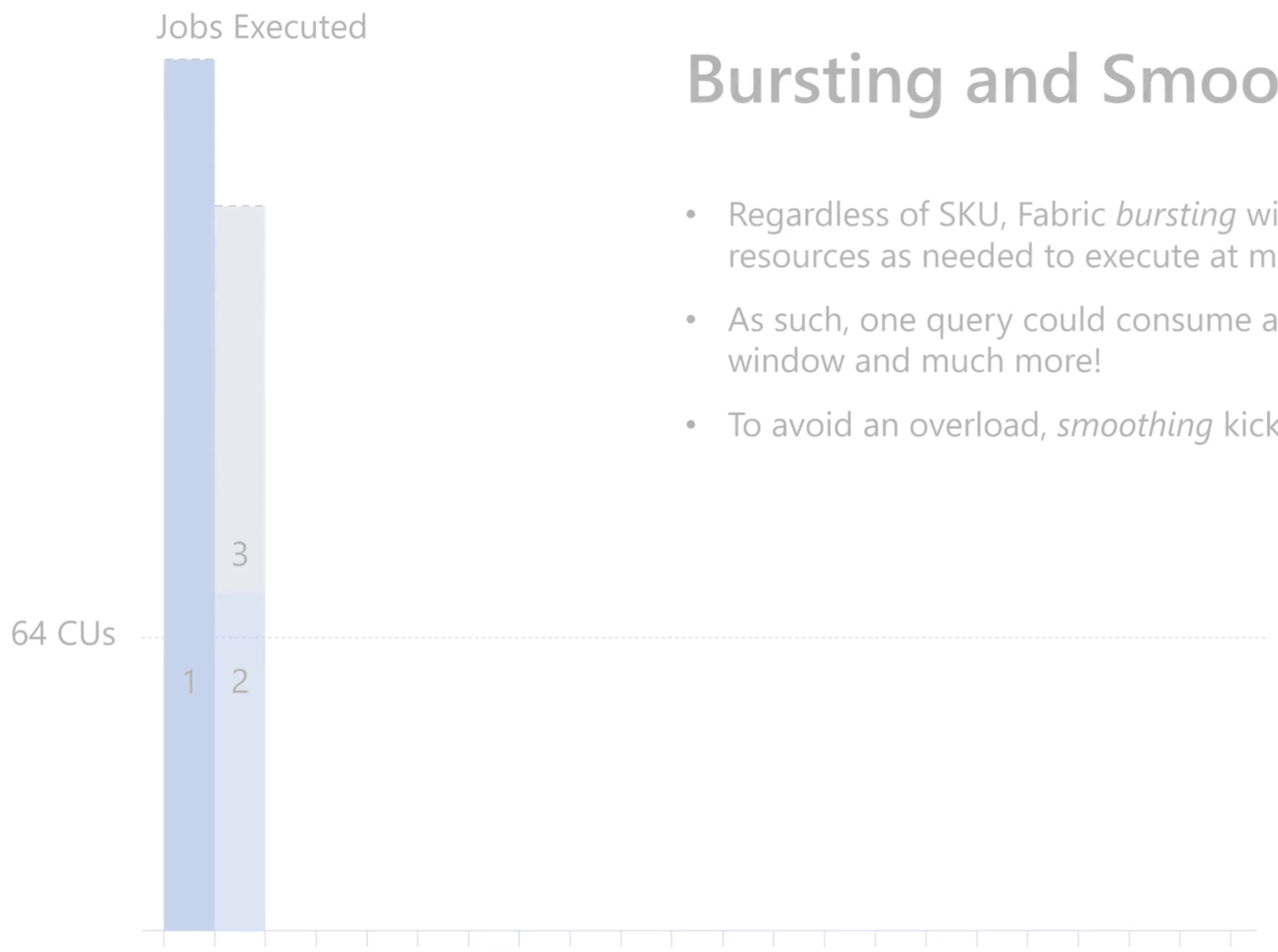
- Warehouse Query: Compute charged for Warehouse (includes user generated and system tasks)
- SQL Endpoint Query: Compute charged for Lakehouse SQL Endpoint (includes user generated and system tasks)
- One Lake compute: Compute charged for all reads and writes for data stored in One Lake



Operation name	CU (s)	Duration (s)	Users	Billing type
Warehouse Query	2,187.77	1,360.42	5	Billable
OneLake Compute	0.01	11,880.00	1	Billable
Total	**2,187.78**	**13,240.42**	**6**	

Operation name	CU (s)	Duration (s)	Users	Billing type
SQL Endpoint Query	4,086.87	1,503.81	6	Both
Total	**4,086.87**	**1,503.81**	**6**	

Warehouse Utilization



Bursting and Smoothing

- Regardless of SKU, Fabric *bursting* will automatically allocate resources as needed to execute at maximum performance
- As such, one query could consume all the quota of a single time window and much more!
- To avoid an overload, *smoothing* kicks in

Dynamics Management Views

Monitor Connection, Session, and Request Status in SQL Analytics Endpoint and Warehouses

sys.dm_exec_connections

- Provides comprehensive information about active connections to the Fabric Warehouse SQL Engine
- Includes details such as session ID, client address, client port, protocol, and authentication method.

sys.dm_exec_sessions

- shows information about all active user connections
- Offers insights into resource utilization by each session, such as the number of active requests

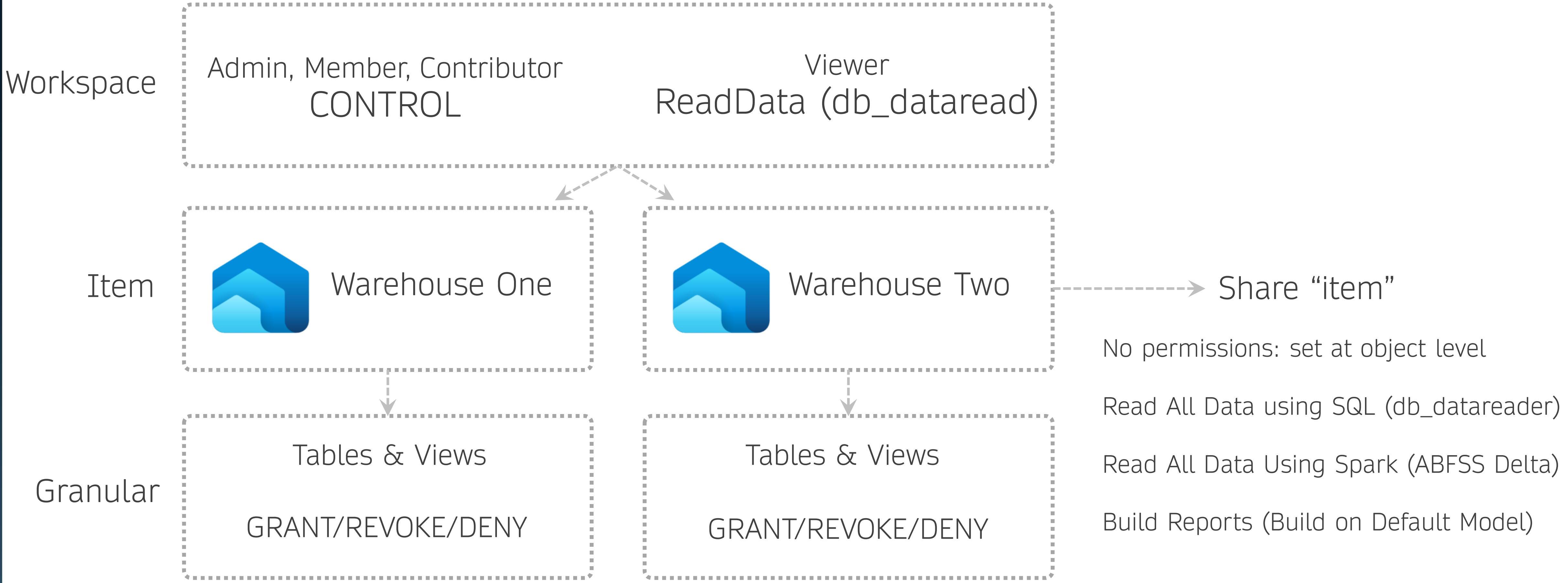
sys.dm_exec_requests

- Provides detailed information about currently executing or waiting queries
- Includes details such as session ID, query text, start time, status

Member, Contributor, and Viewer can execute sys.dm_exec_sessions and sys.dm_exec_requests

Security

Security can be applied at Workspace, Warehouse, and Object level



No permissions: set at object level

Read All Data using SQL (db_datareader)

Read All Data Using Spark (ABFSS Delta)

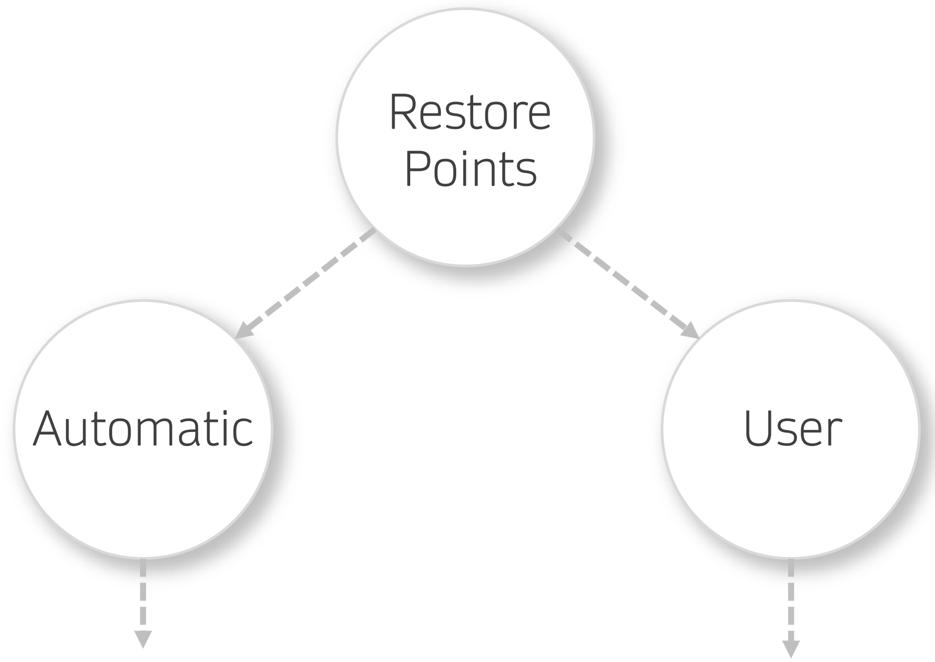
Build Reports (Build on Default Model)

Disaster Recovery

Restore-In-Place

- Restore to a known “good” state
- Used when any corruption occurs
- Restore back after a failed deployment
- Restore back to a version for dev/test

Table Cloning (see next section)



7 Days Availability

Up to 42 restore points

8 hour RPO

7 Days Availability

Use before pausing

Use before significant data modifications

Table Cloning

Clone an existing table into new table

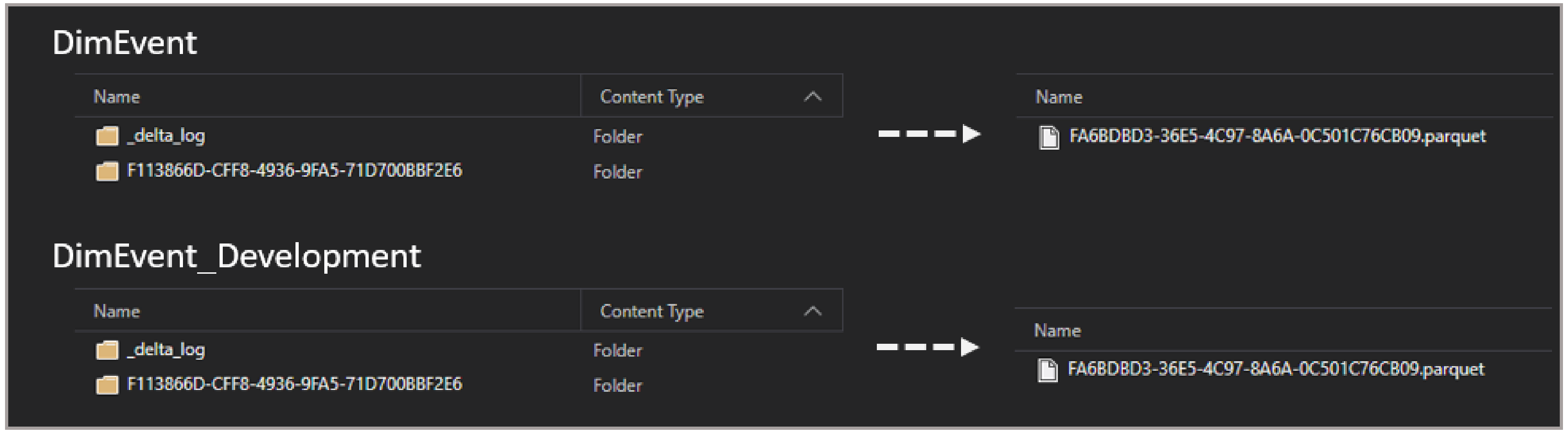
“Shallow” clone as only the metadata is cloned

Cloned table is separate from base table

```
--create table clone
CREATE TABLE <schema>.<new_table_name>
AS CLONE OF <schema>.<existing_table_name>
```

Used for:

- Snapshotting at point-in-time
- Backups for recovery
- Testing data changes before applying to main table



DimEvent

Name	Content Type
_delta_log	Folder
F113866D-CFF8-4936-9FA5-71D700BBF2E6	Folder

Name
FA6BDBD3-36E5-4C97-8A6A-0C501C76CB09.parquet

DimEvent_Development

Name	Content Type
_delta_log	Folder
F113866D-CFF8-4936-9FA5-71D700BBF2E6	Folder

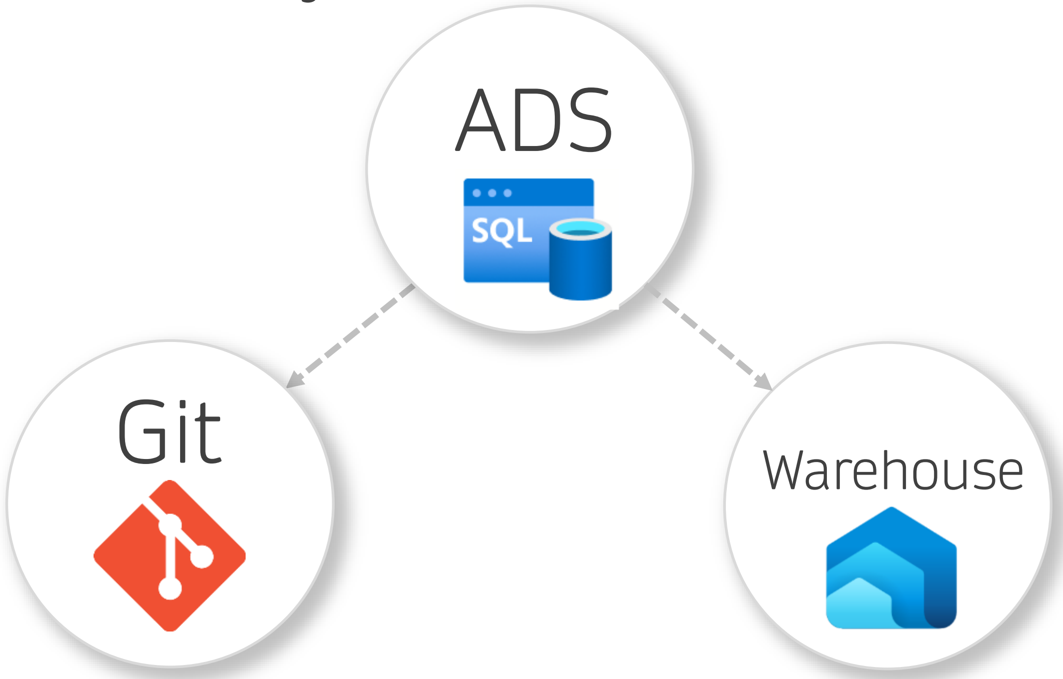
Name
FA6BDBD3-36E5-4C97-8A6A-0C501C76CB09.parquet

Source Control & Deployment

“Treat your database like code” Grant Fritchey

Azure Data Studio Database Projects

- Current version 1.48.0 (DB: 1.4.2)
- Support for Warehouse and Lakehouse SQL Endpoint
- Uses dacpac for build and deploy
- Cannot ALTER!

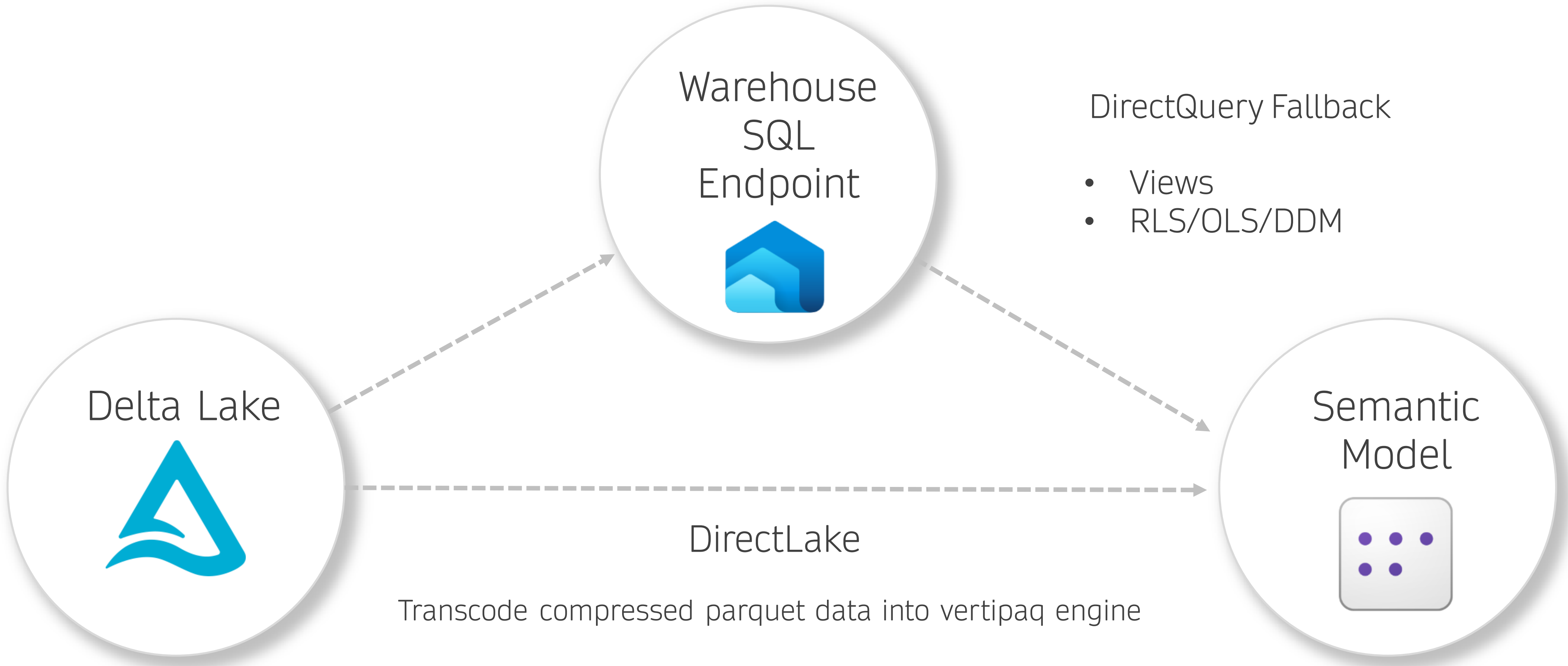


Connect ADS to Git to allow source control

Semantic Models – Direct Lake

Direct Lake is the new connectivity method from Semantic Models to Warehouse & Lakehouse

No requirement to Import the data into the Semantic model



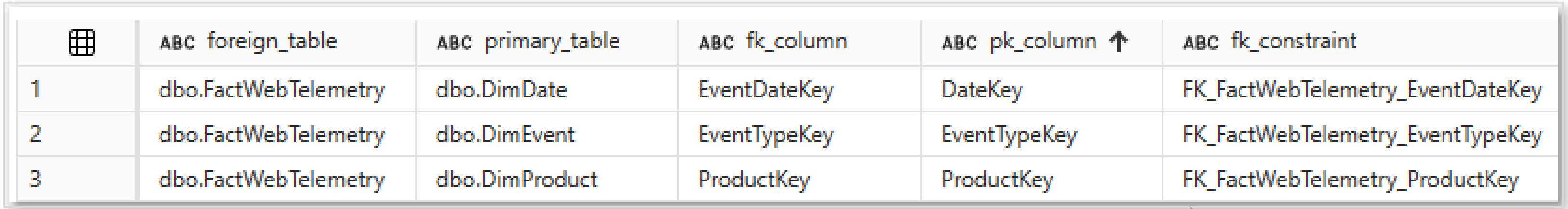
DirectQuery Fallback

- Views
- RLS/OLS/DDM

Transcode compressed parquet data into vertipaq engine

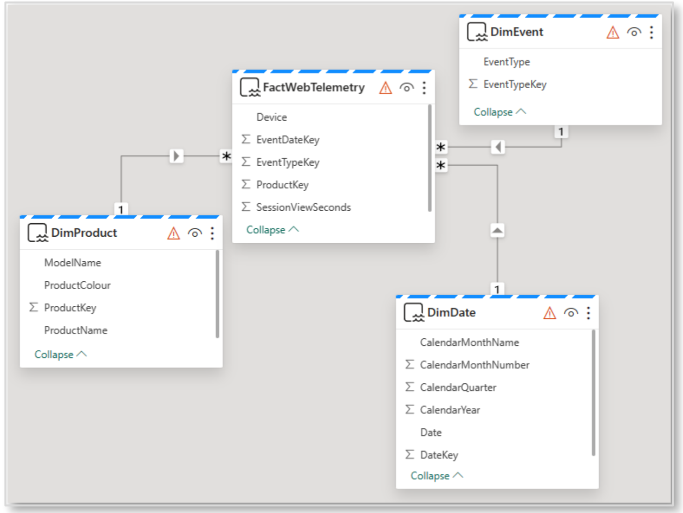
Constraints created in Warehouse appear in Default Semantic Model

Constraints created in Default Semantic Model appear in Warehouse



	foreign_table	primary_table	fk_column	pk_column	fk_constraint
1	dbo.FactWebTelemetry	dbo.DimDate	EventDateKey	DateKey	FK_FactWebTelemetry_EventDateKey
2	dbo.FactWebTelemetry	dbo.DimEvent	EventTypeKey	EventTypeKey	FK_FactWebTelemetry_EventTypeKey
3	dbo.FactWebTelemetry	dbo.DimProduct	ProductKey	ProductKey	FK_FactWebTelemetry_ProductKey

Default Semantic Model has a bi-directional sync with it's underlying Warehouse



Custom Semantic Models are unaffected by Warehouse constraints

Thank you

Connect with me at:

https://linkedin.com/in/brianbonk
https://brianbonk.dk
https://github.com/brianbonk

Connect