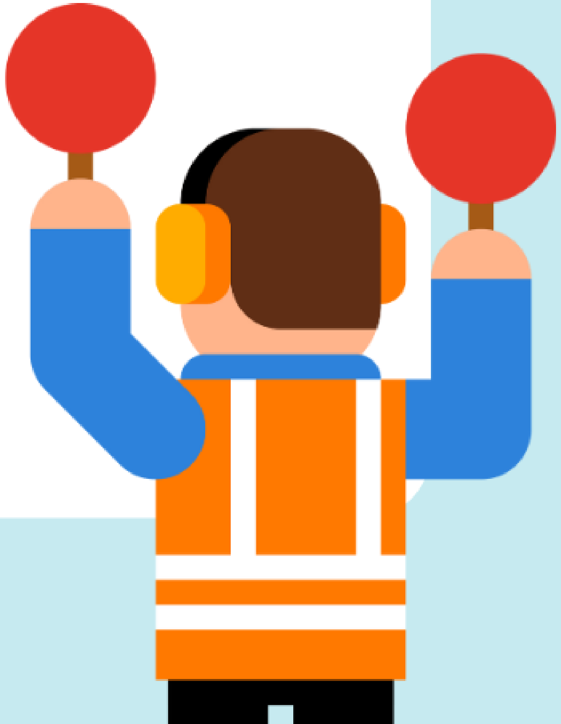
Flying High Amongst the Clouds with Fabric Warehouses

Andy Cutler & Brian Bønk

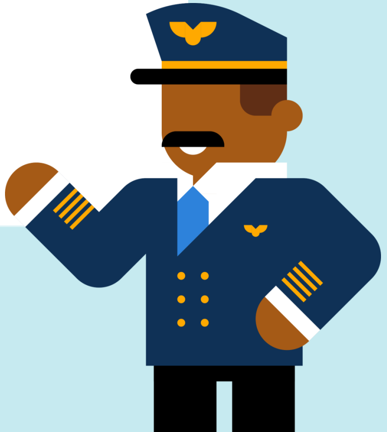


sqlbits

2024

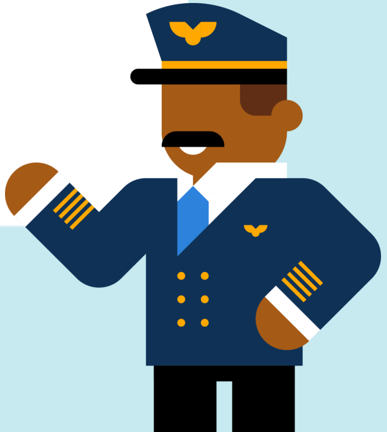
Agenda – part 1

- Welcome and who are we
- What is Fabric an introduction
- Fabric Data warehouse vs other data services
- Architectures and architectural patterns
- The Polaris engine
- OneLake
- The T-SQL surface area
- Isolation levels
- Ingestion – load patterns from external sources



Agenda – part 2

- CRUD
- Table cloning
- Meta-data
- Security
- Disaster recovery and business continuity
- Source control and deployment
- Semantic Models – both default and user defined
- Direct Lake for DWH vs import
- Round up and Q/A



Who are we

Andy Cutler

Brian Bønk

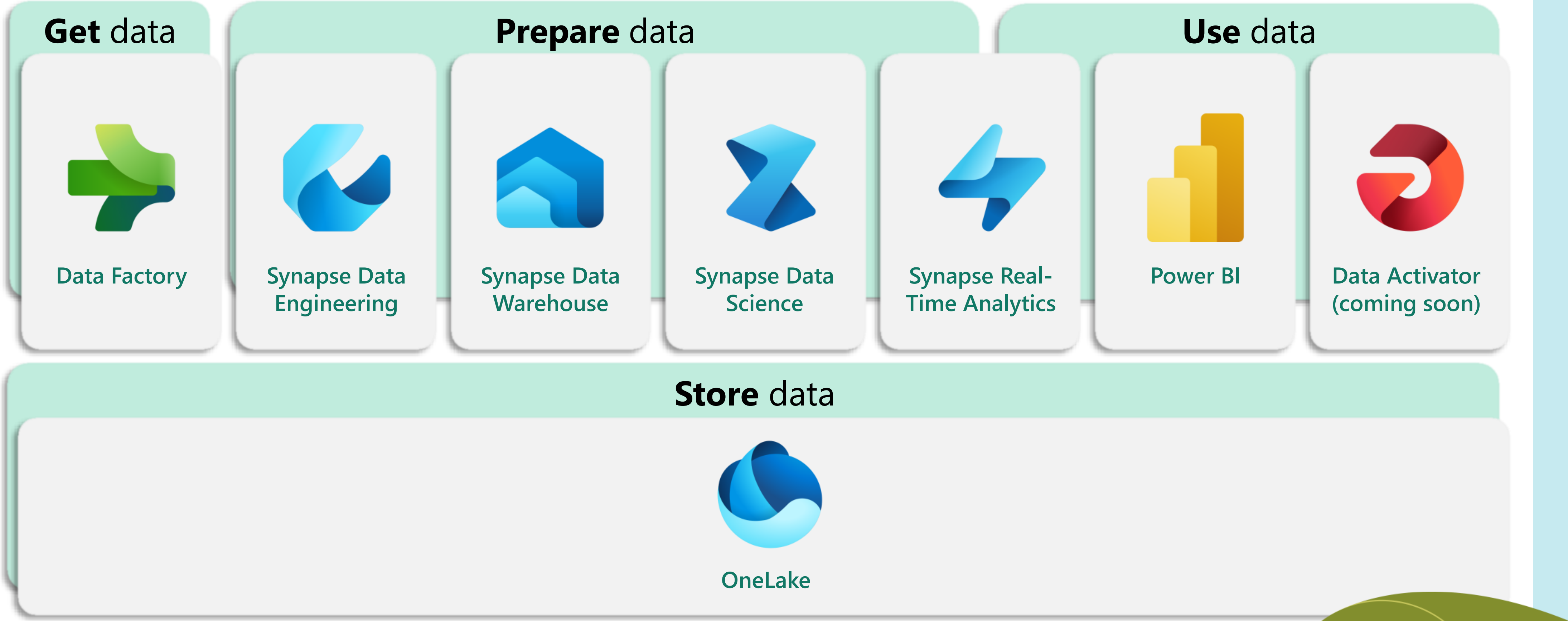
MVP
Microsoft
Most Valuable Professional

MVP
Microsoft
Most Valuable Professional

sqlbits
2024

sqlbits

2024



Get data

- Data Factory

Prepare data

- Synapse Data Engineering
- Synapse Data Warehouse
- Synapse Data Science
- Synapse Real-Time Analytics

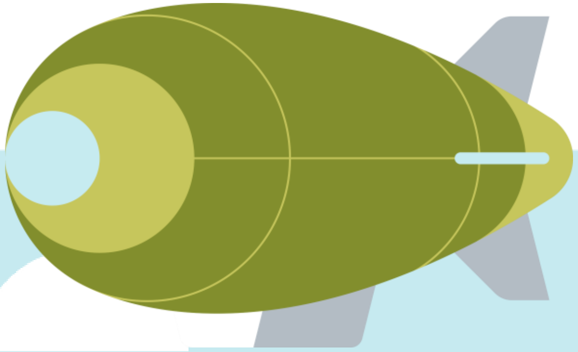
Use data

- Power BI
- Data Activator (coming soon)

Store data

- OneLake

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

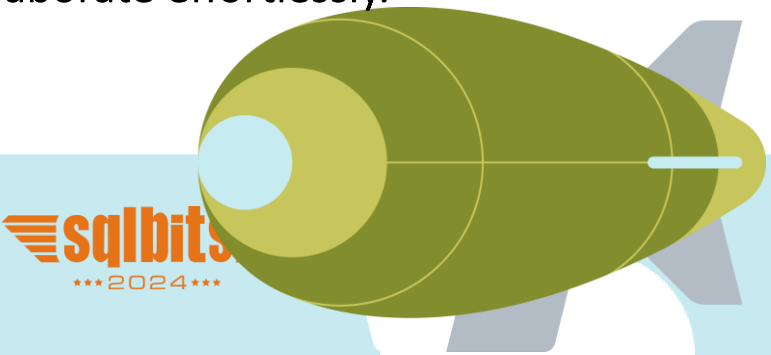


Synapse Data Warehouse VS Synapse Serverless SQL VS Synapse Dedicated pool VS Databricks

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.

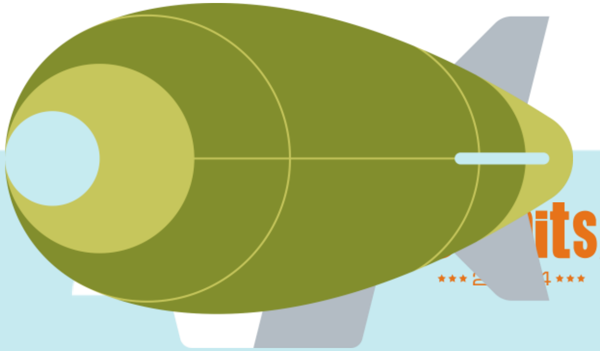
Integration: Seamless integration with multiple Azure services, including Azure Data Factory, Azure Synapse Analytics, Synapse Real-Time Analytics, Synapse Data Warehousing, Power BI and Data Explorer, with automatic provisioning of the underlying hardware.

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



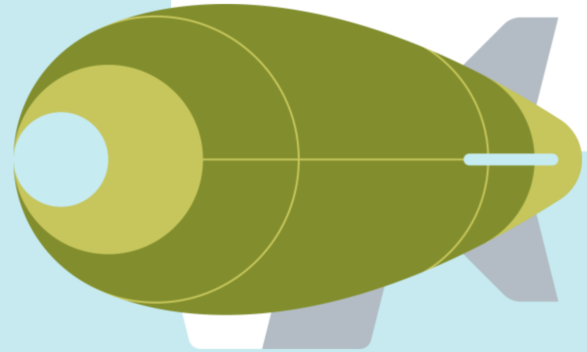
Synapse Data Warehouse VS Synapse Serverless SQL VS Synapse Dedicated pool VS Databricks

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.

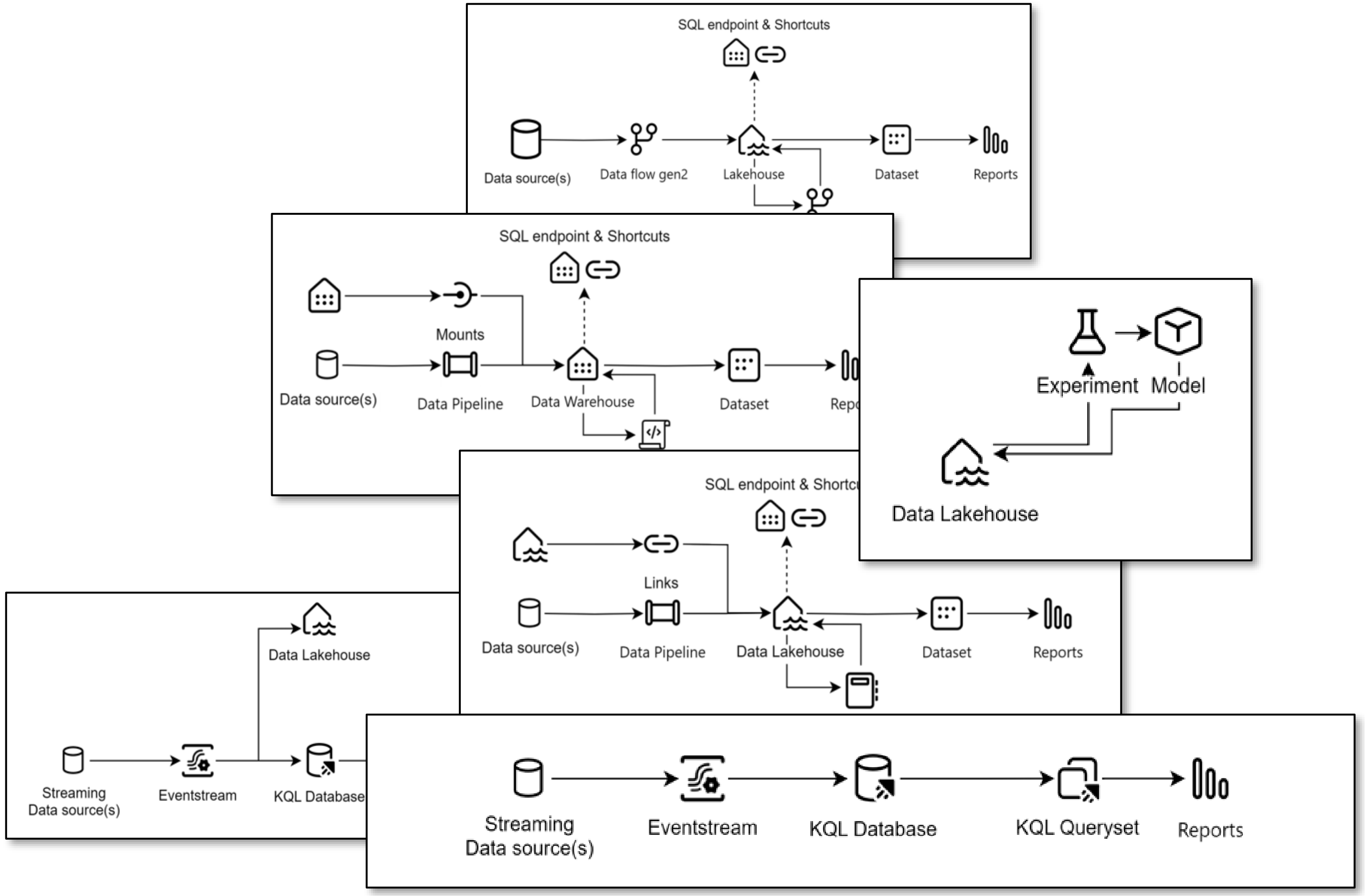


Data Warehouse

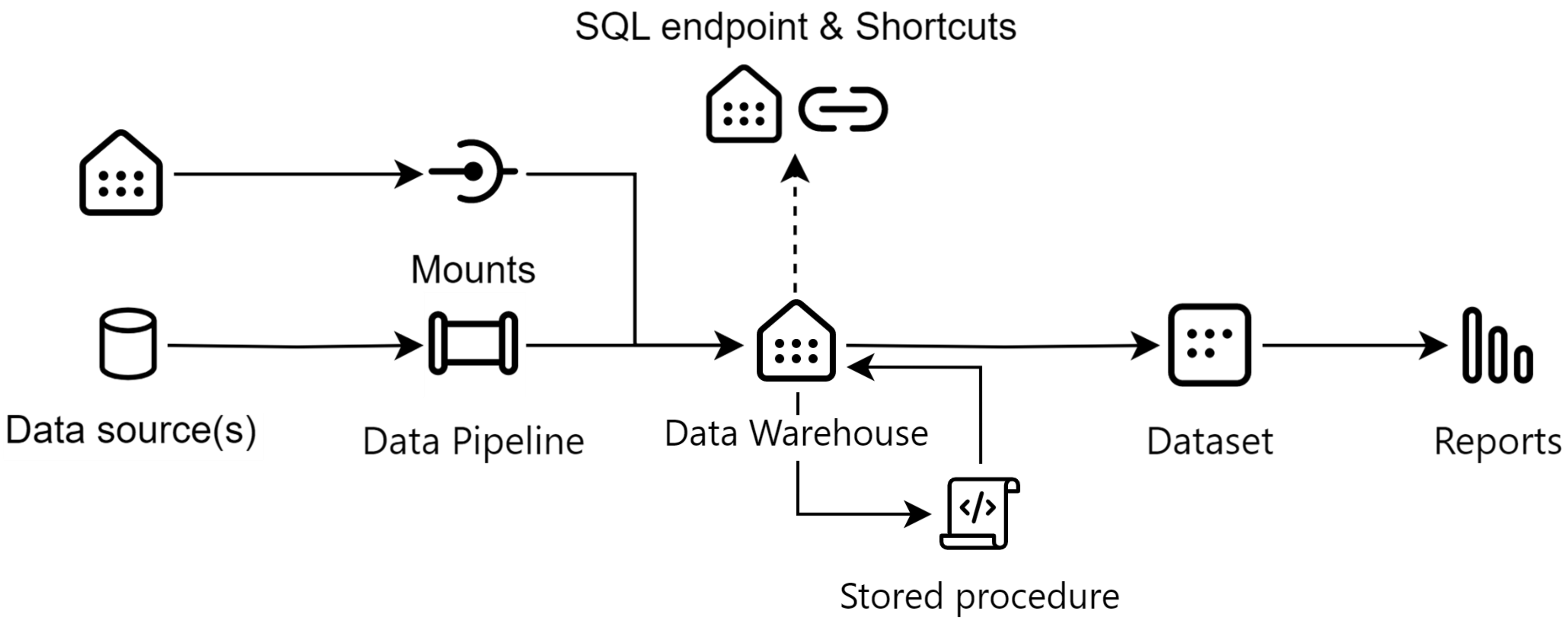
Lakehouse

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)

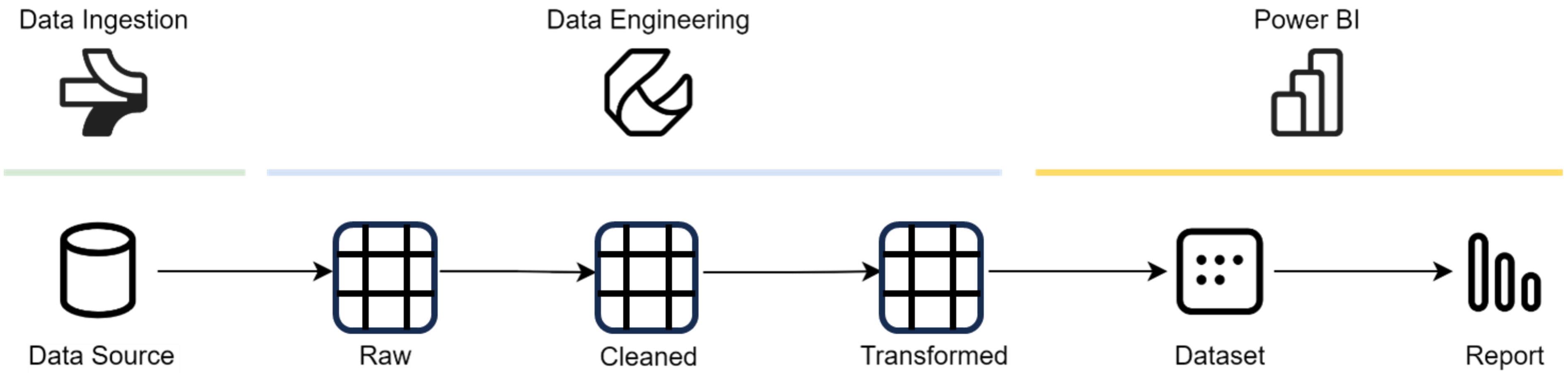
Data Warehouse



Data
Warehouse

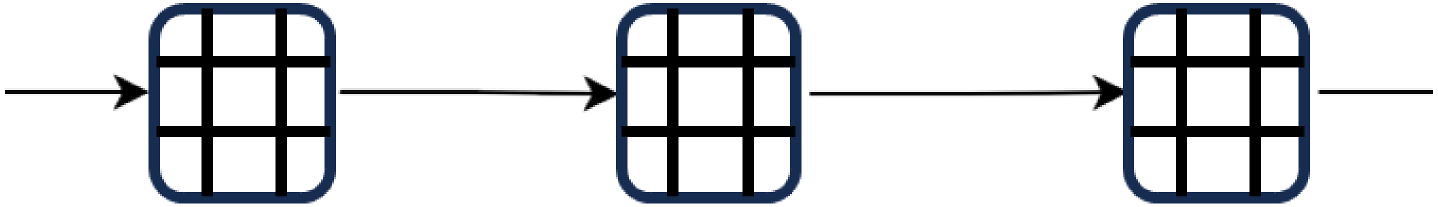


Data Warehouse



Logical Warehouse

Viewhouse



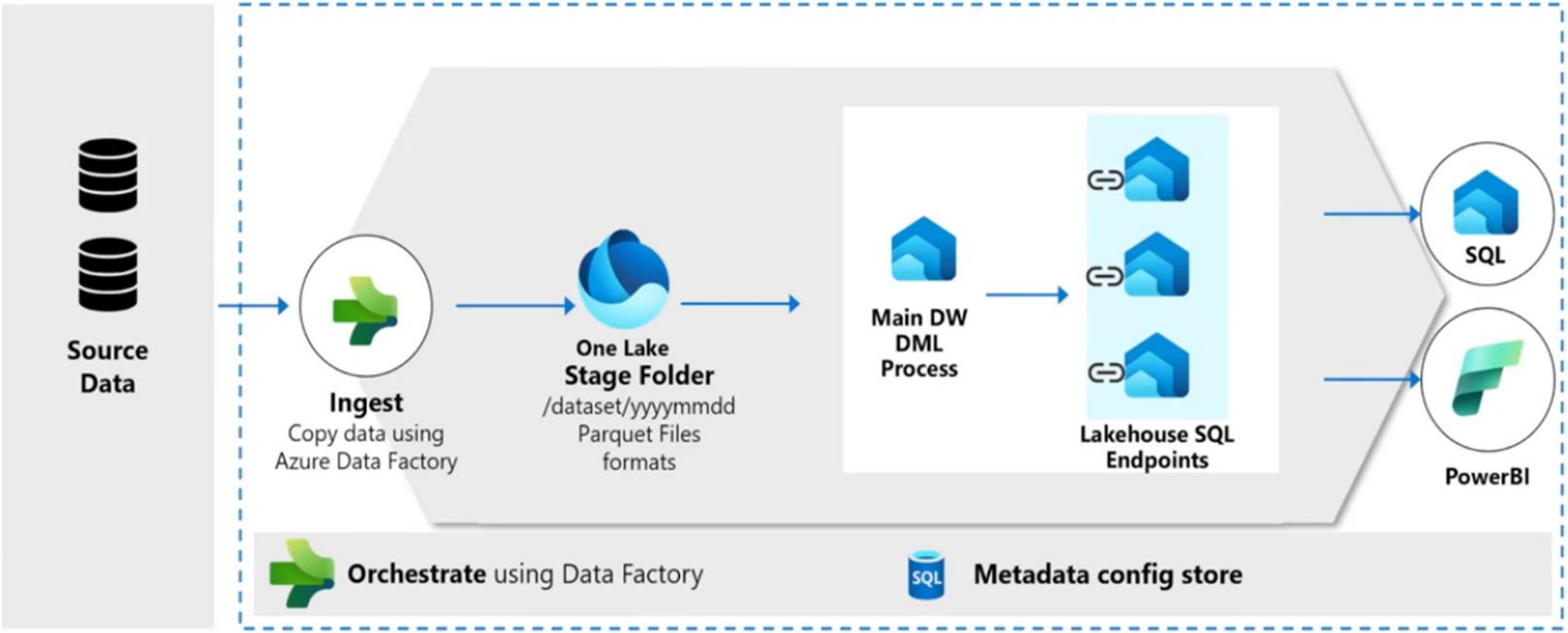
SQL VIEW

SQL VIEW

SQL VIEW

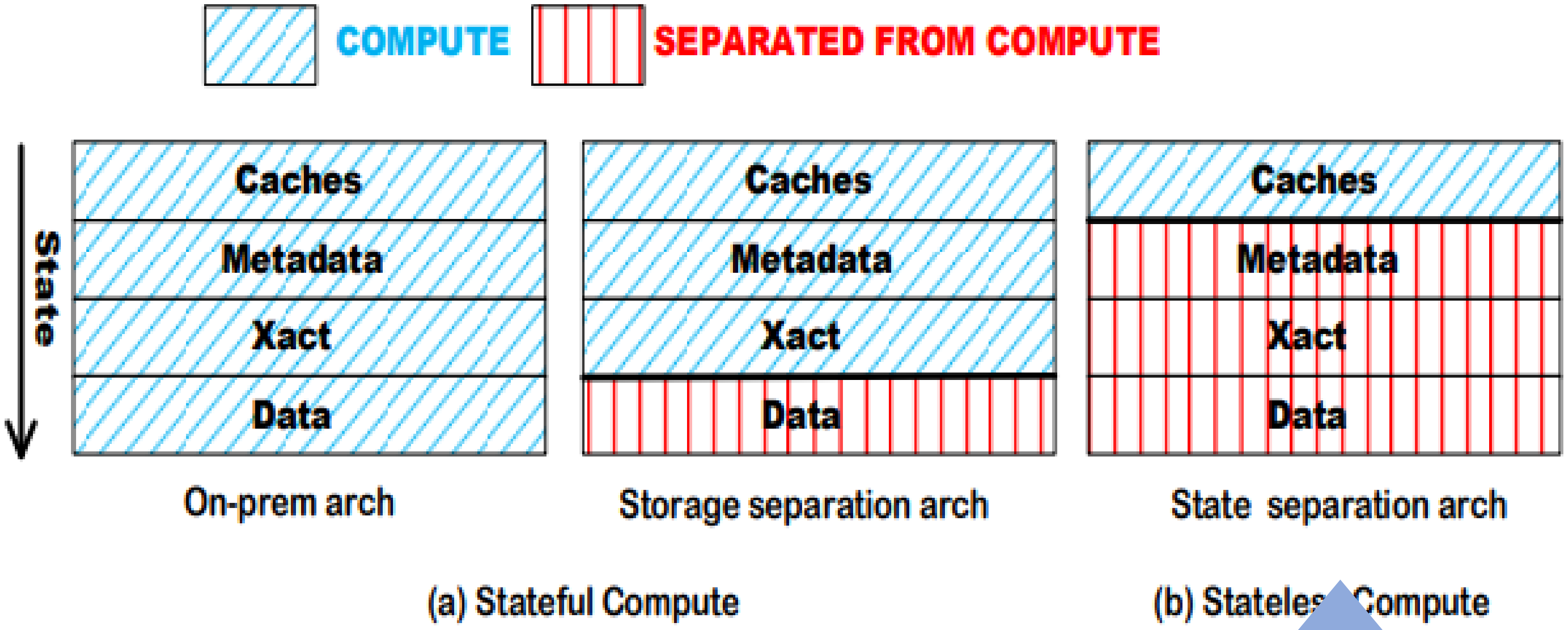
Data Warehouse

Modern Data Warehouse Pattern in Microsoft Fabric



The Polaris engine

Stateless compute



▨ COMPUTE ▥ SEPARATED FROM COMPUTE

State

	On-prem arch	Storage separation arch	State separation arch
Caches	▨	▨	▨
Metadata	▨	▨	▥
Xact	▨	▨	▥
Data	▨	▥	▥

(a) Stateful Compute

(b) Stateless Compute

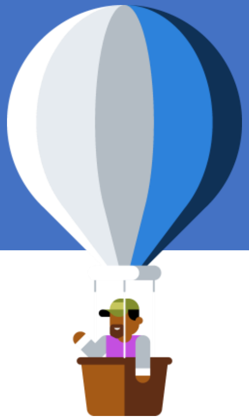
Synapse and Fabric

The Polaris engine

Data abstraction

Data
Warehouse

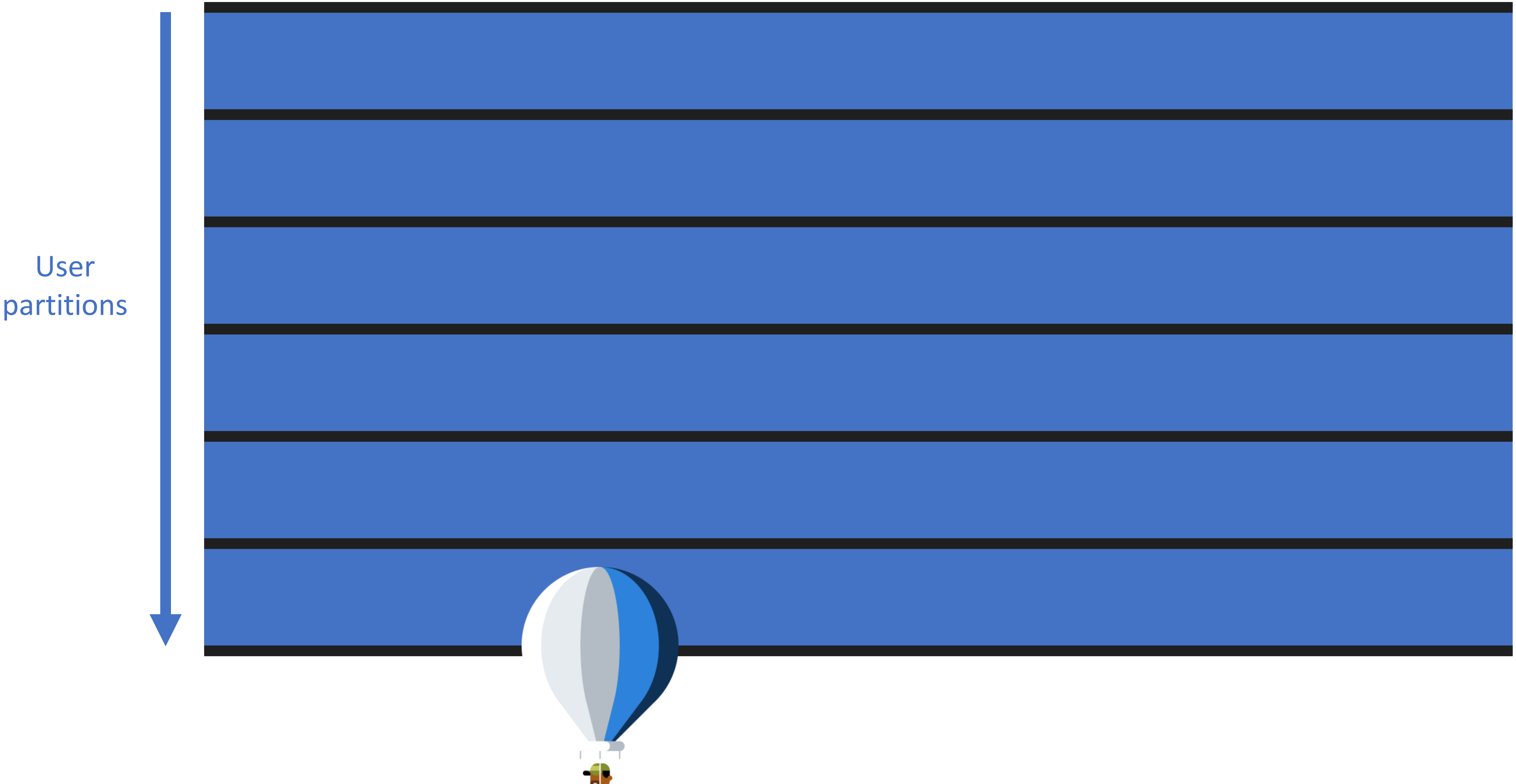
Data



The Polaris engine

Data abstraction

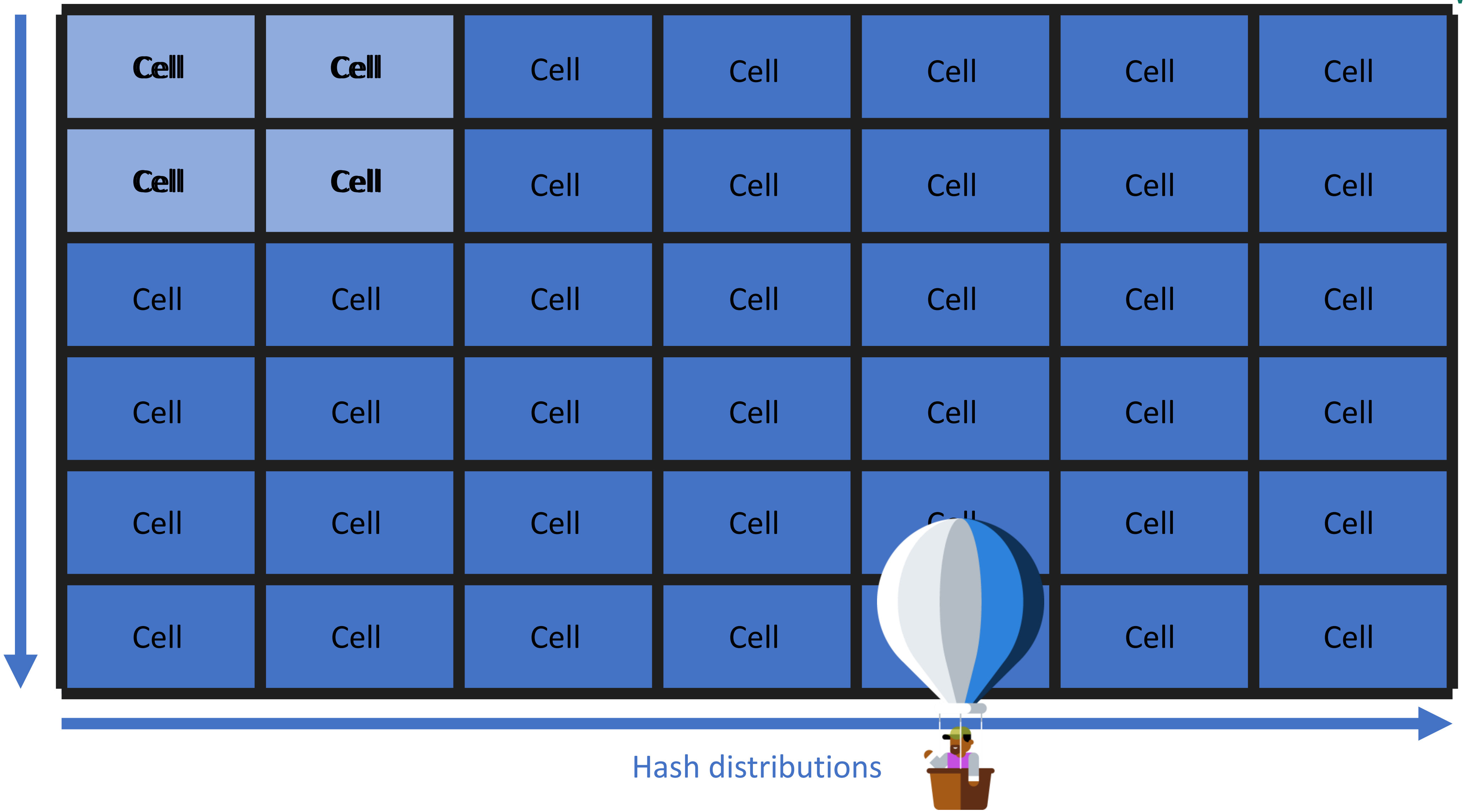
Data Warehouse



The Polaris engine

Data abstraction

Data Warehouse



Cell	Cell	Cell	Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell	Cell	Cell	Cell
Cell	Cell	Cell	Cell		Cell	Cell

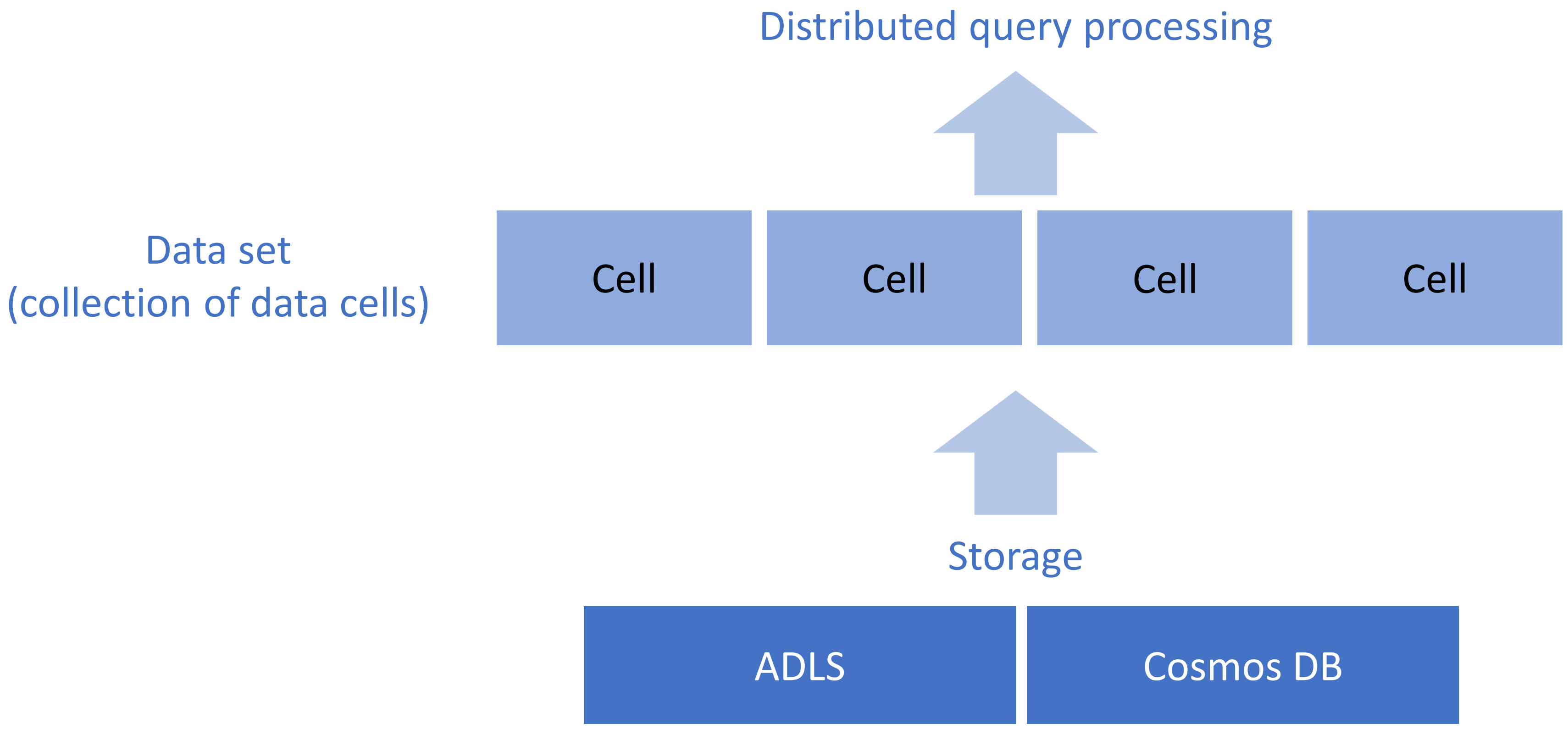
User partitions

Hash distributions

The Polaris engine

Data abstraction

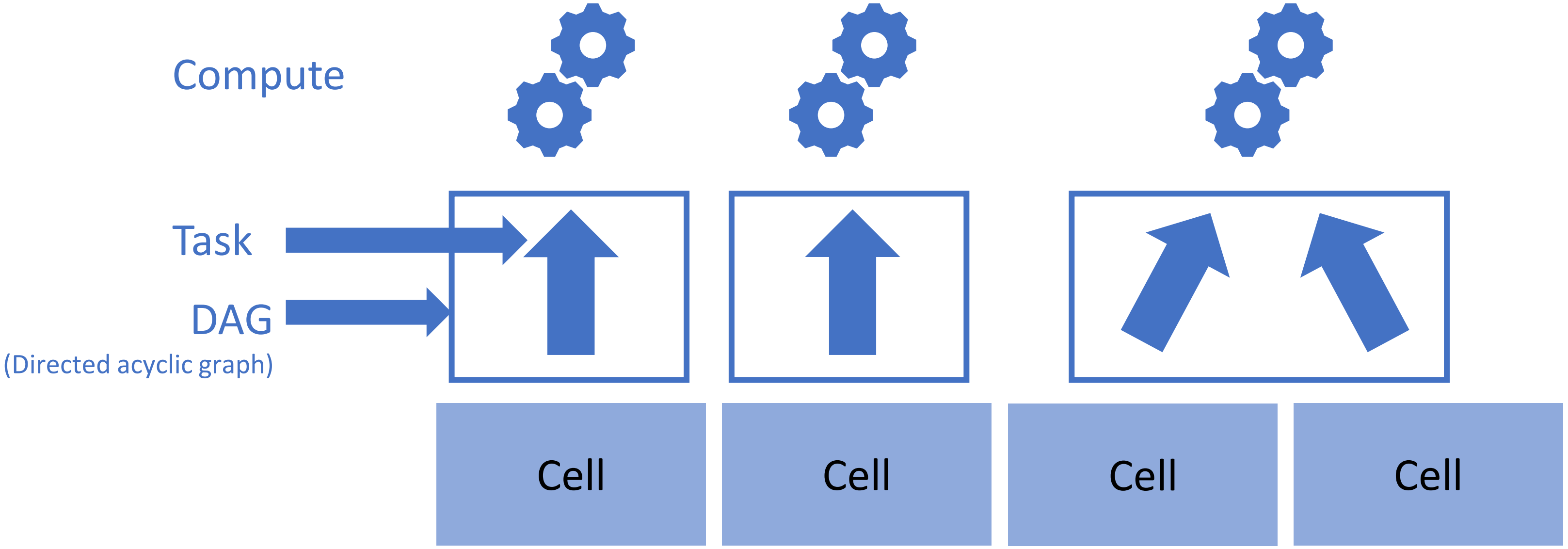
Data Warehouse



The Polaris engine

Data abstraction

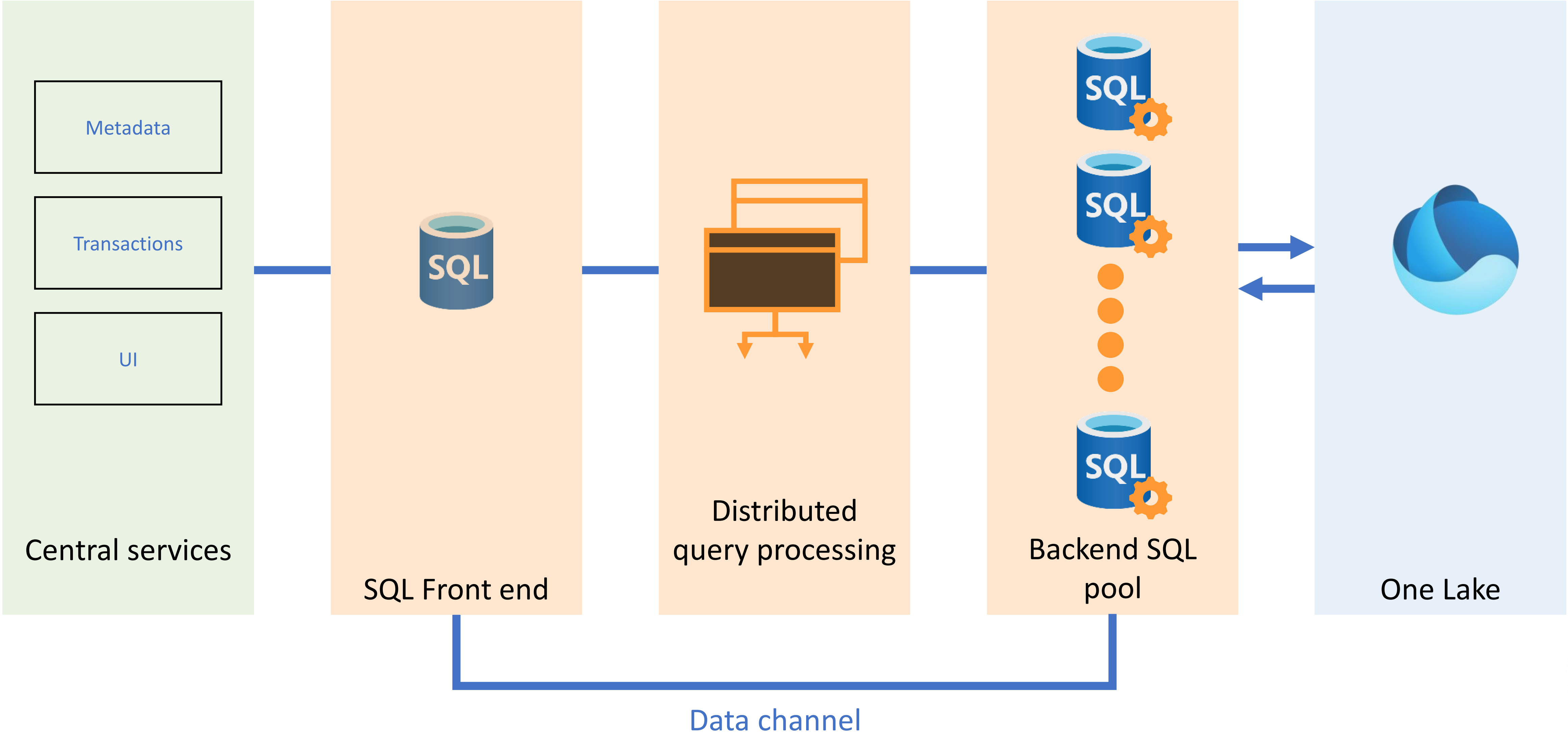
Data
Warehouse



The Polaris engine

Service architecture

Data Warehouse



The Polaris engine

Read more here

Data
Warehouse

Whitepaper on Polaris:
aka.fm/polaris

Broken down to a blogpost:
aka.fm/polaris-blog

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>

Identity Columns

TRUNCATE

CREATE ROLE/USER

MERGE

Recursive Queries

BULK LOAD

SP_SHOWSPACEUSED

SET TRANSACTION ISOLATION LEVEL

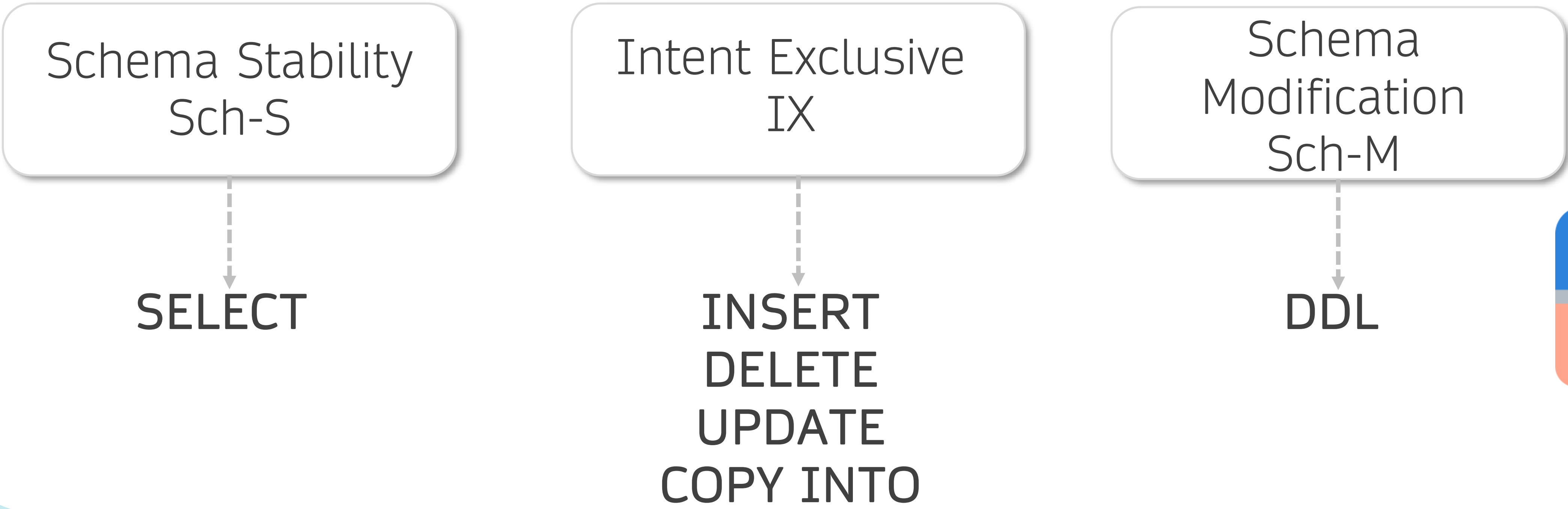
Transactions & Isolation Levels

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

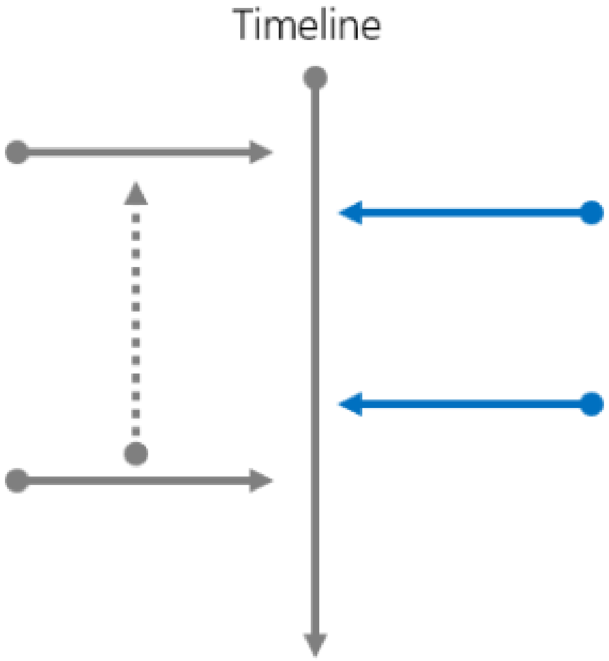
Intent Exclusive
IX

INSERT
DELETE
UPDATE
COPY INTO

Schema
Modification
Sch-M

DDL

Transactions & Isolation Levels



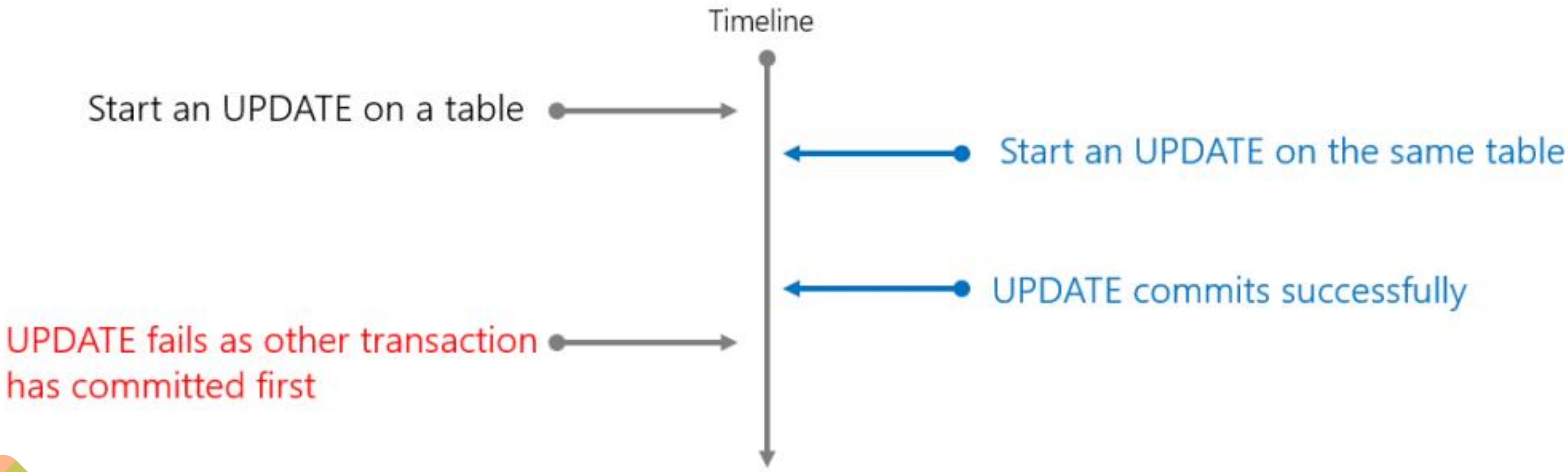
Begin a transaction and SELECTs from several tables

INSERTs, UPDATEs, DELETEs are initiated on the same tables

INSERTs, UPDATEs, DELETEs are committed on the tables

The SELECTs commit and return the data from the tables at the point the transaction began

Transactions & Isolation Levels



Timeline

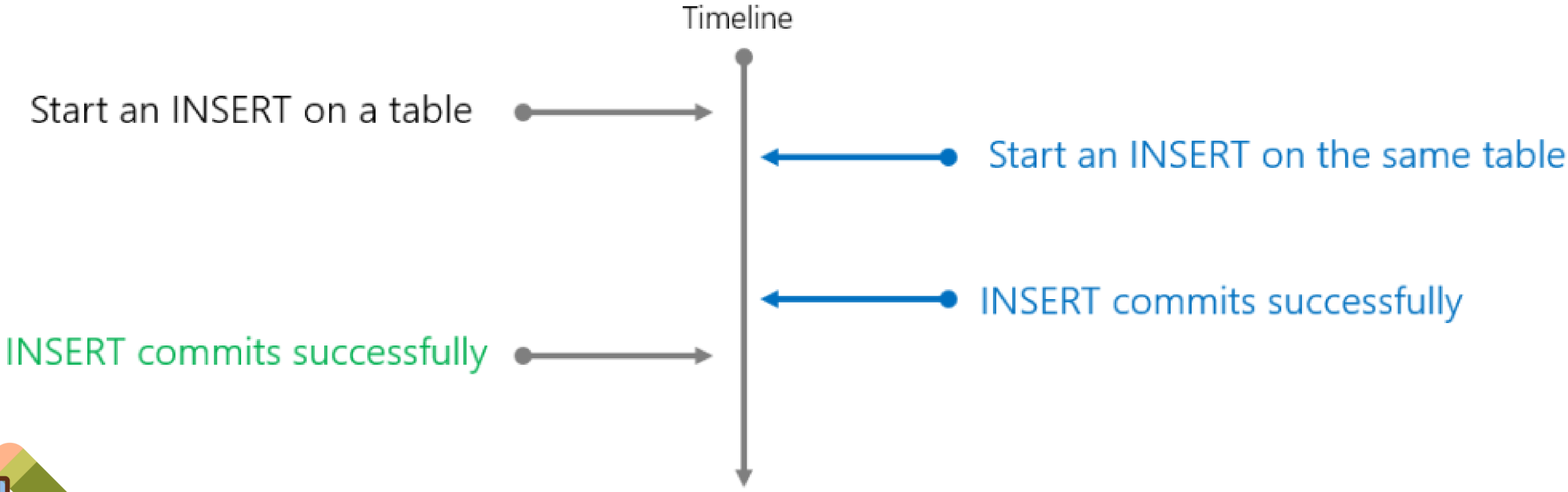
Start an UPDATE on a table

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**

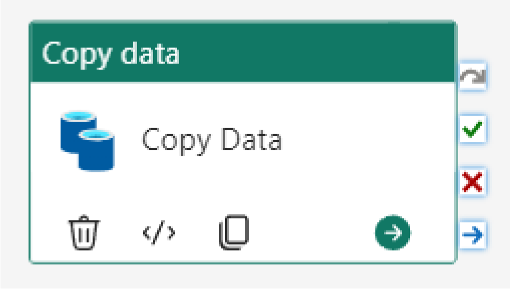
SQL **INSERT INTO…VALUES / SELECT**

Pyodbc/JDBC (from Notebooks)

Low/No-Code

Pipelines

“Brute force”

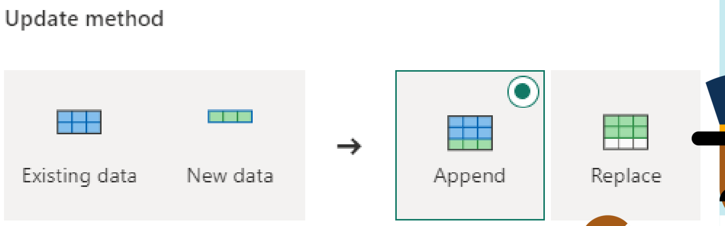


Copy data

Copy Data

Dataflows Gen2

(Power Query)



Update method

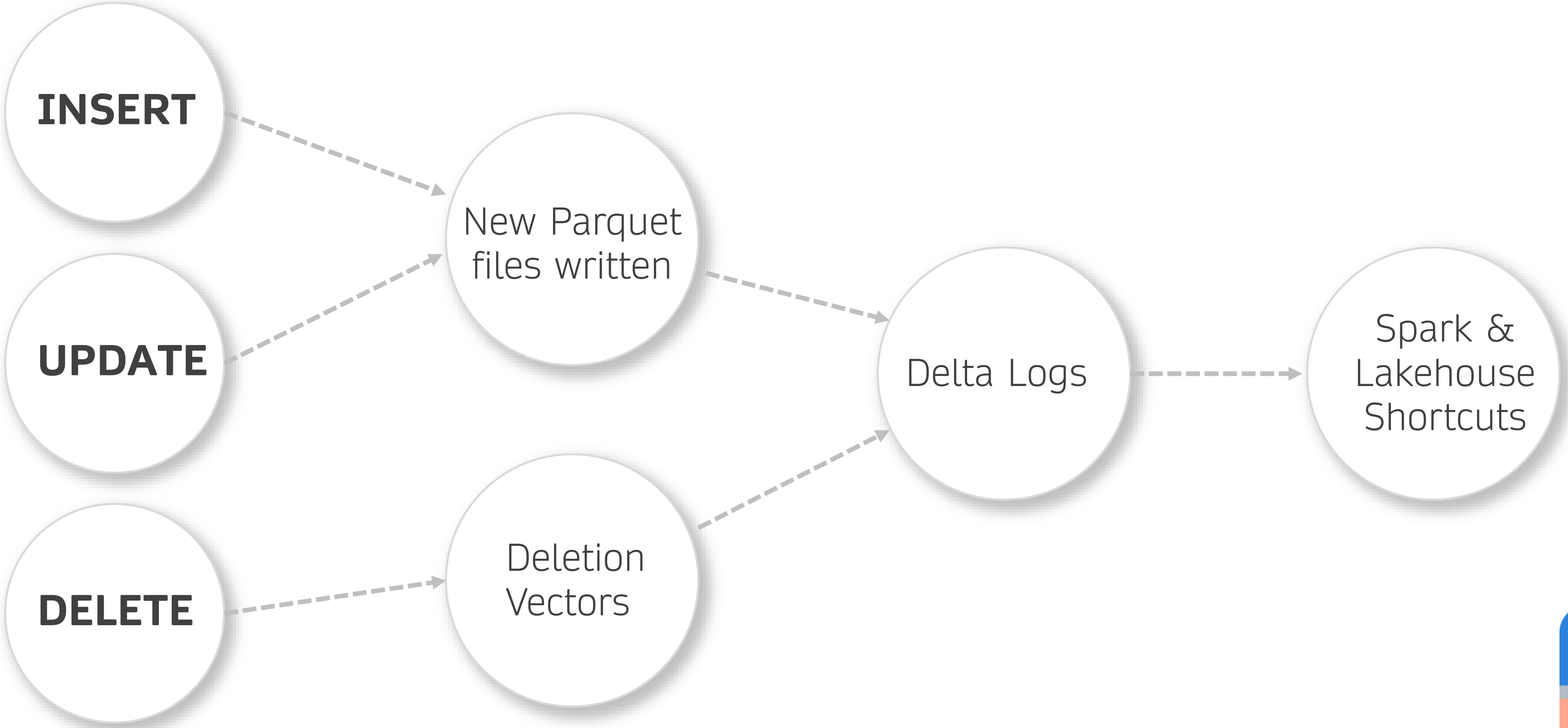
Existing data New data → ● Append ○ Replace

sqlbits 2024

Ingestion: Loading the Warehouse

Data Warehouse

All SQL operations are performed on Parquet files in OneLake



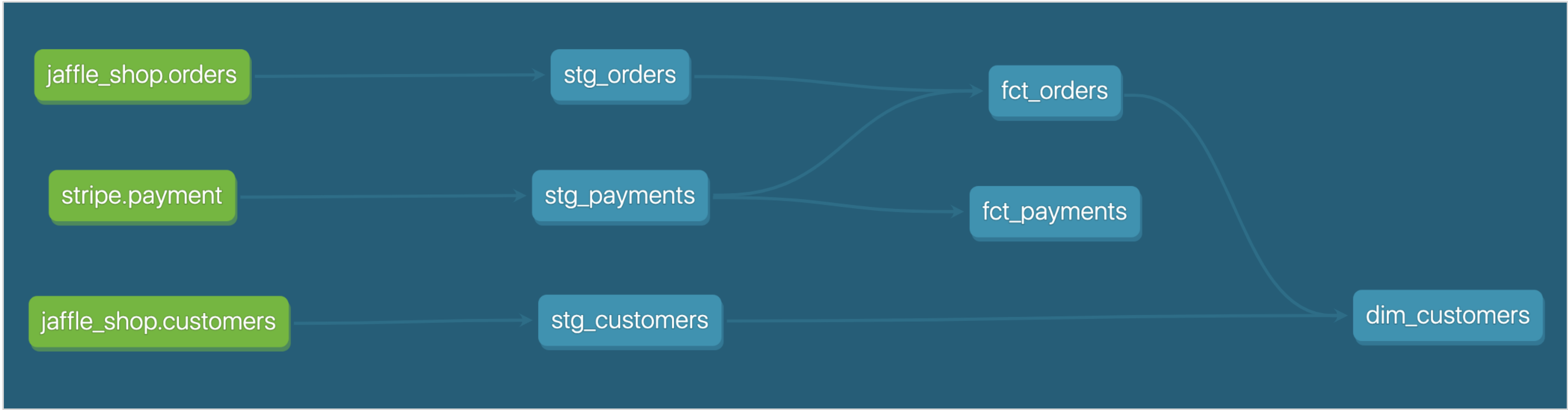
Ingestion: Loading the Warehouse

Data Warehouse

dbt (including cloud) support in Warehouse

Declare your SQL loading processes

Create DAG for loading tables based on dependencies

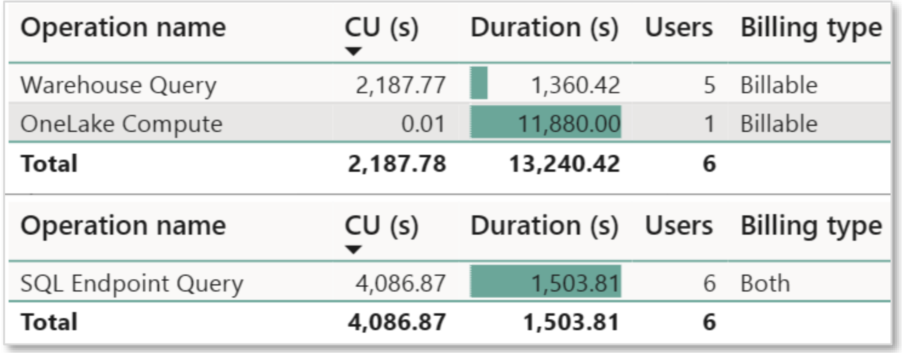


Warehouse Utilization

Data Warehouse

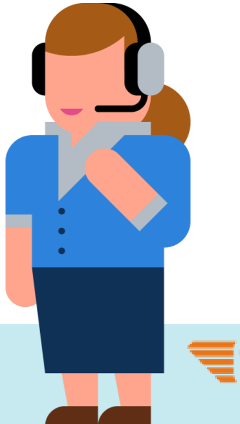
Usage is tracked by Capacity Unit seconds (consumed by read and write activity against the Warehouse and reads against Lakehouse SQL Endpoint)

- 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



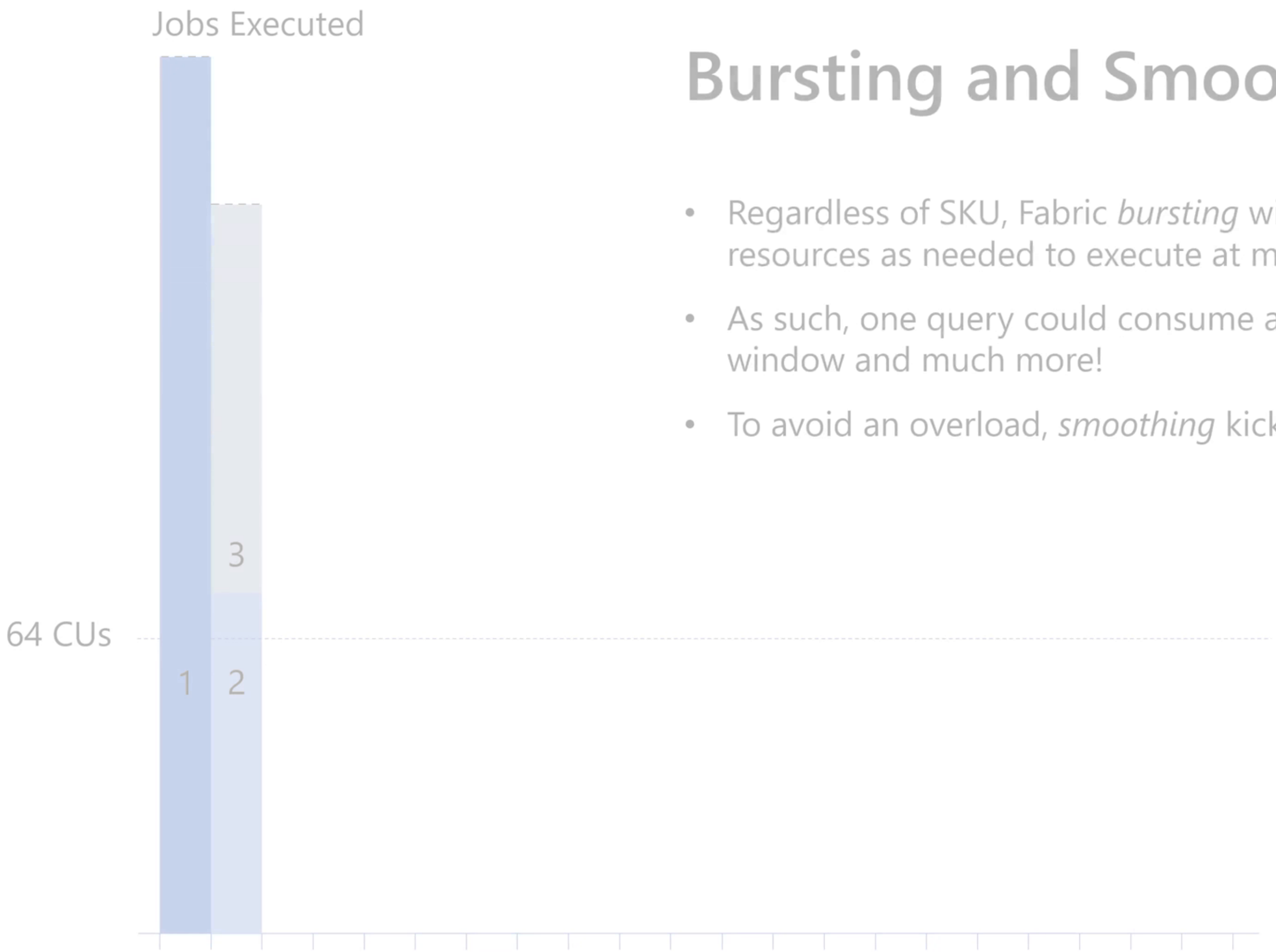
Operation name	CU (s)	Duration (s)	Users	Billing type
Warehouse Query	2,187.77	1,360.42	5	Billable
OneLake Compute	0.01	11,880.00	1	Billable
Total	**2,187.78**	**13,240.42**	**6**	

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

Data Warehouse



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

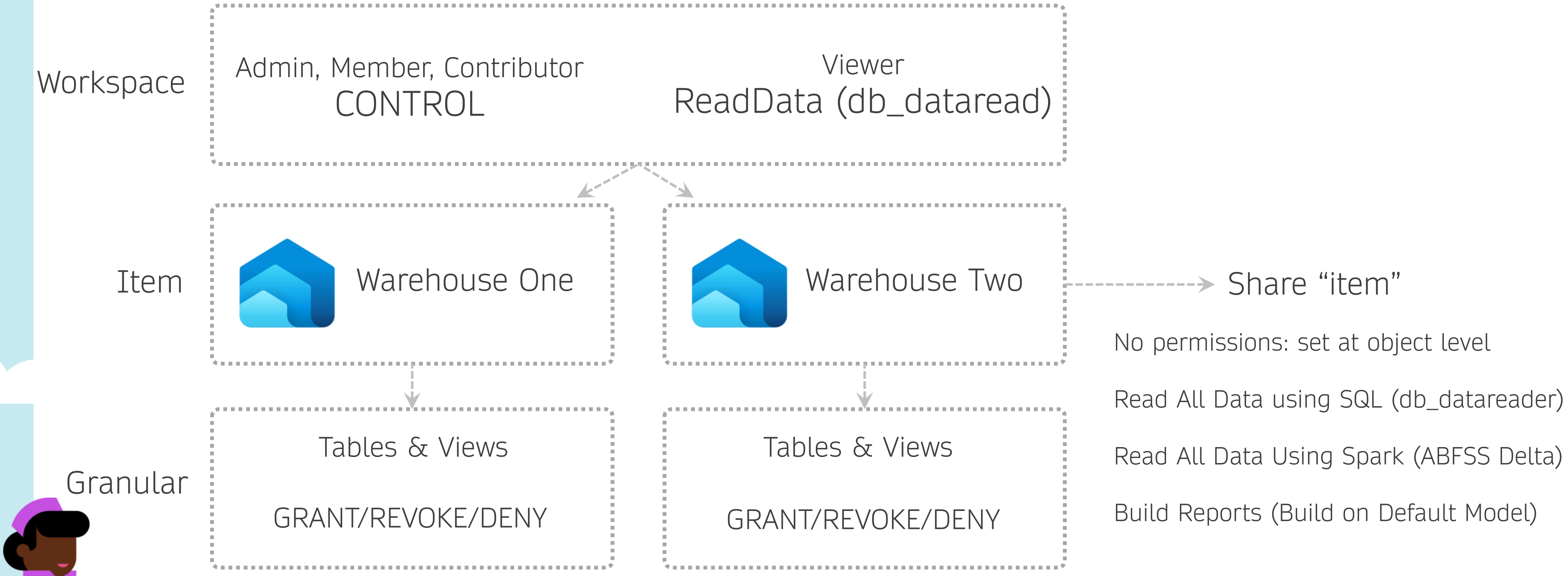
- Provides detailed information about currently executing or waiting queries
- Includes details such as session ID, query text, start time, status

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

Security

Data Warehouse

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



Workspace

Admin, Member, Contributor
CONTROL

Viewer
ReadData (db_dataread)

Item

Warehouse One

Warehouse Two

Share “item”

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)

Granular

Tables & Views

GRANT/REVOKE/DENY

Tables & Views

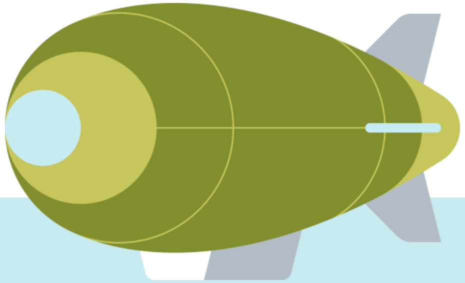
GRANT/REVOKE/DENY

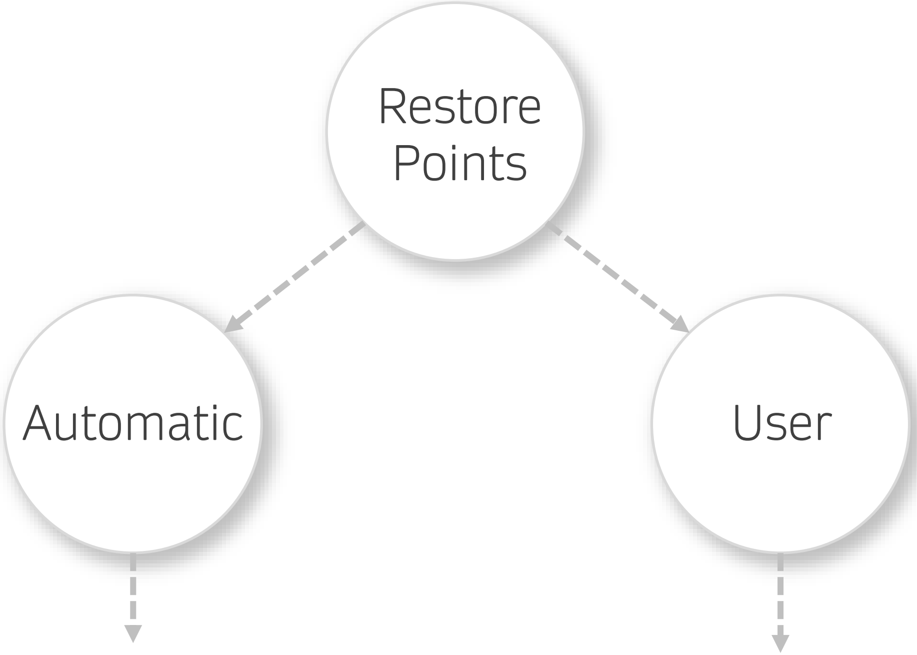
Disaster Recovery

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)





Restore Points

Automatic

User

7 Days Availability

Up to 42 restore points

8 hour RPO

7 Days Availability

Use before pausing

Use before significant data modifications

Table Cloning

Data Warehouse

Clone an existing table into new table

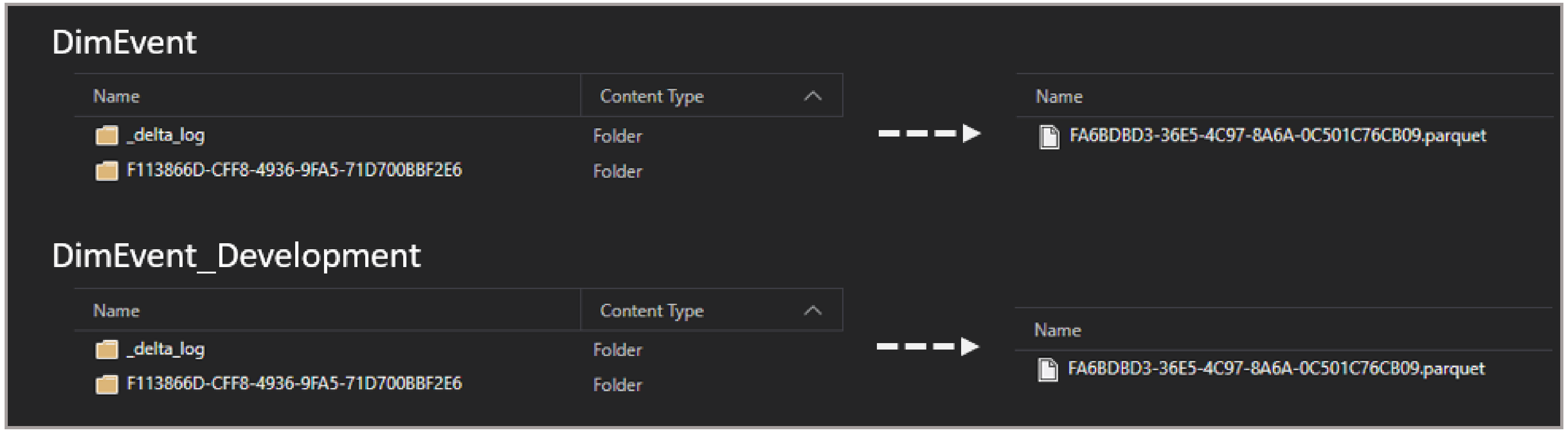
“Shallow” clone as only the metadata is cloned

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

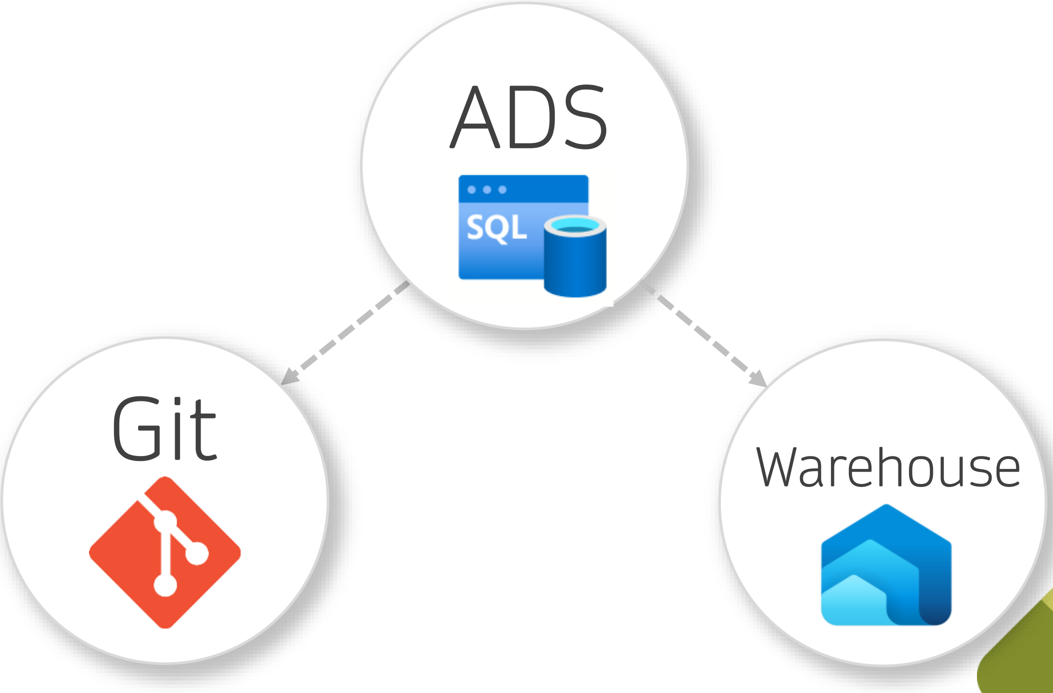
Source Control & Deployment

Data Warehouse

“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!



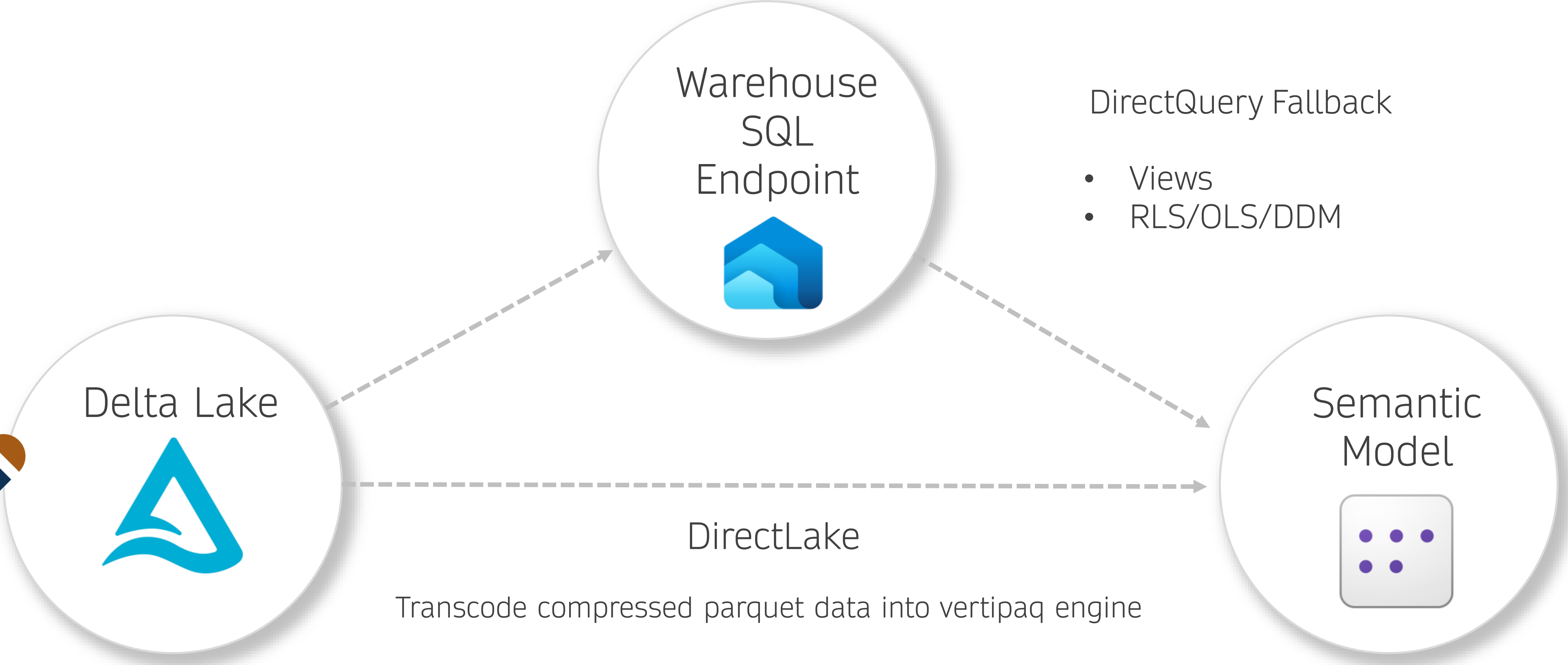
Connect ADS to Git to allow source control

Semantic Models – Direct Lake

Data Warehouse

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

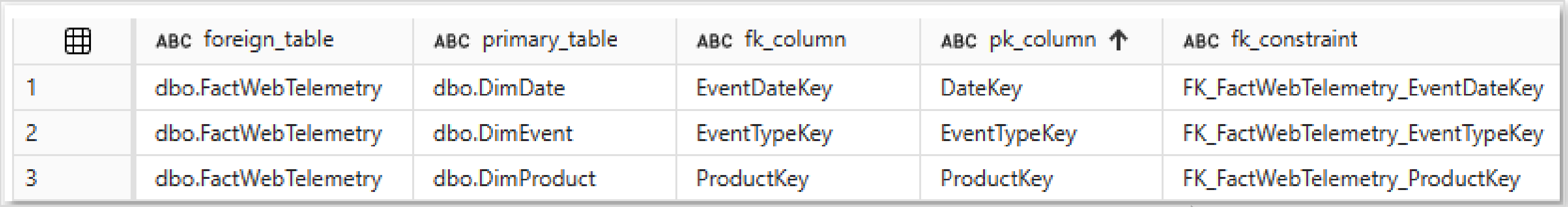
Transcode compressed parquet data into vertipaq engine

Semantic Models – Relationships

Data Warehouse

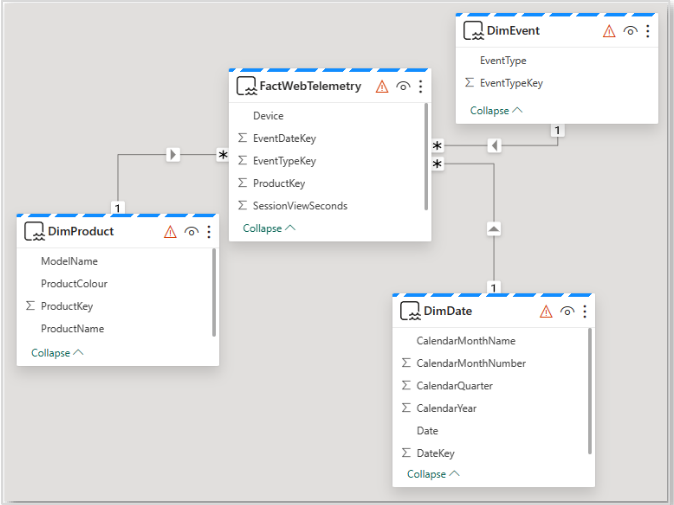
Constraints created in Warehouse appear in Default Semantic Model

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

Questions and dialogue

sqlbits
★★★2024★★★