NO.

Title

Enjoy the most efficient handwriting experience! Taking notes on the go, whether for inspiration, ideas, knowledge learning, business insights, or even sketches...

NOTE IN

Creative notes

By reading we enrich the mind;
by writing we polish it.

In General

Data warehouse

Characterstics:

1) Data Integration: Ingest data from multiple sources into a unified store/Repository

2) Schema-on-write: Data is transformed & structured before loading, adhering to a predifined schema for optimized quering

3) Security & governance: provide Role-based access control (RBAC), data encsiption, and Compliance features for enterprise security.

4) ACID Compliance 5) High performance 6) Historical data managment

Architecture

1) bottom layer: where data flow into DWH Through ETL.
- In many DWHs, data is stored in RDBMS or similar systen

2) The middle layer: build around an analytics engine (OLAP)
3) The Top layer: Includes user interfaces & reporting tools.

Data lake

- low-cost data storage solutions.
- handle massive volumes of data
- uses a schema-on-read approach → They don't apply standard format to incoming data but on accessing it
- Data lakes stores data on its native format; structured, unstructured, semi-structured

Architecture

- Early ones are built on HDFS, Modern ones often use a cloud object store (Microsoft blob storage)
- Data lakes separate data storage from compute resources, which make them more cost-effective and scalable than DWH
- to process data in datalake: users can connect external data processing tools such as spark
- used: to maintain backups & archive old unused data | store datasets for ML, AI and big data analytics workloads

Data lake house

- Schema-on-Read & schema-on-write
- Diverse data types :
- unified storage. Combines the scalability of data lakes with the performance of data ware house
- Cost in governance & security | Cost efficiency

Fabric lakehouse

* is a data architecture platform for storing, managing, and analyzing structured and unstructured data in a single location

* flexibl & scalable solution that allows organizations to handle large volums of data using various tools and frameworks to process and analyze the data.

* lakehouse combine → scalability of a data lake
  with → performance and structure of a data warehouse
  providing a unified platform for data storage, management, and analytics

* you can perform sql-analytics directly or top of the Delta tables in the lake to provides a frictionless and performant experiance (Ingestion → Reporting)
  Through sql analytics endpoint (read-only)

note only the tables in delta format are available in sql analytics endpoint

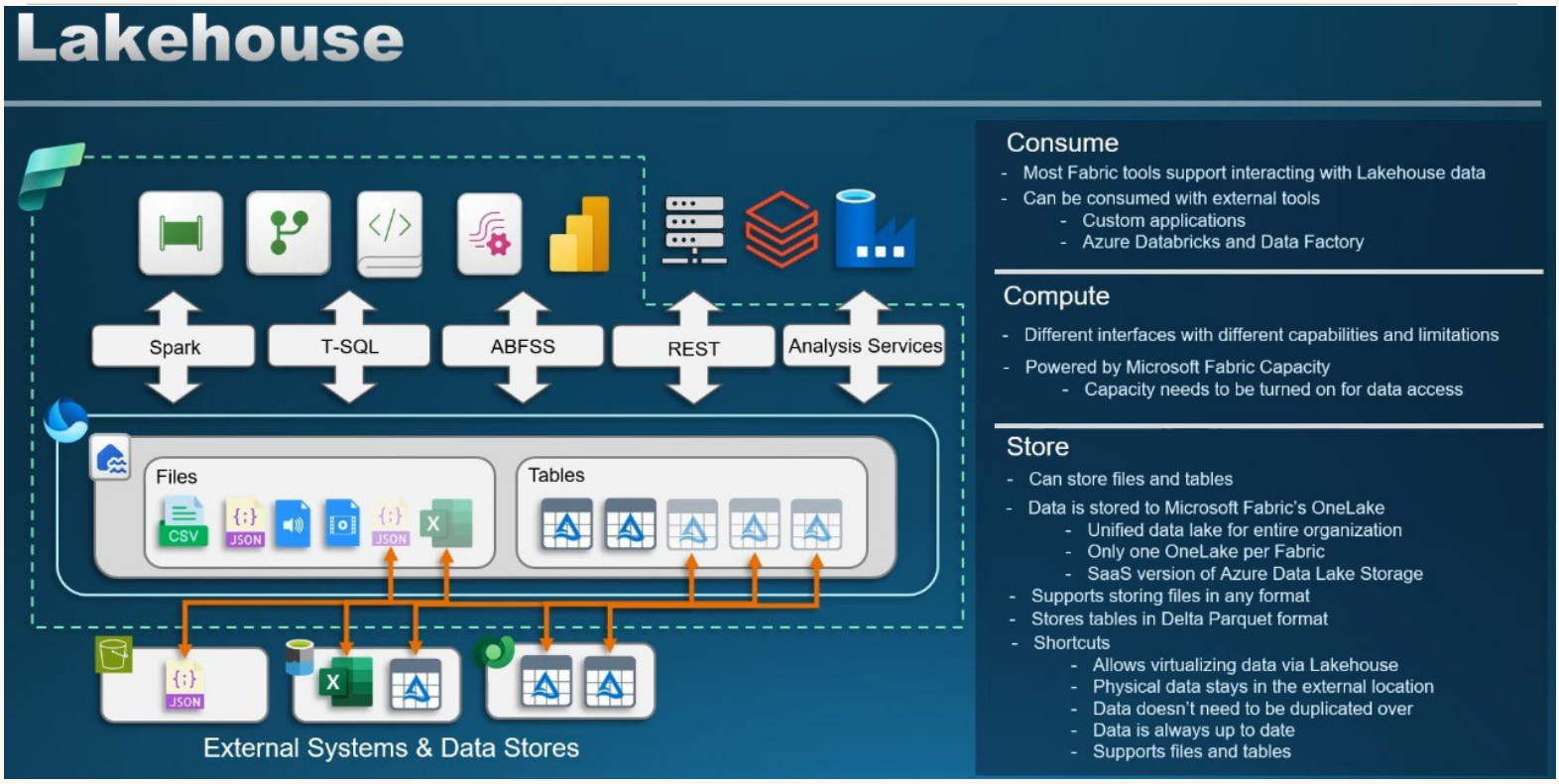
* lakehouse provide a fully managed file to table experiance
  Through Automatic table discovery and registeration

1- you drop a file into the managed area of lakehouse
2- system auto. validates it for the structured formats (only support Delta table) and register it into the metastore with necessary metadata such as Columns names.

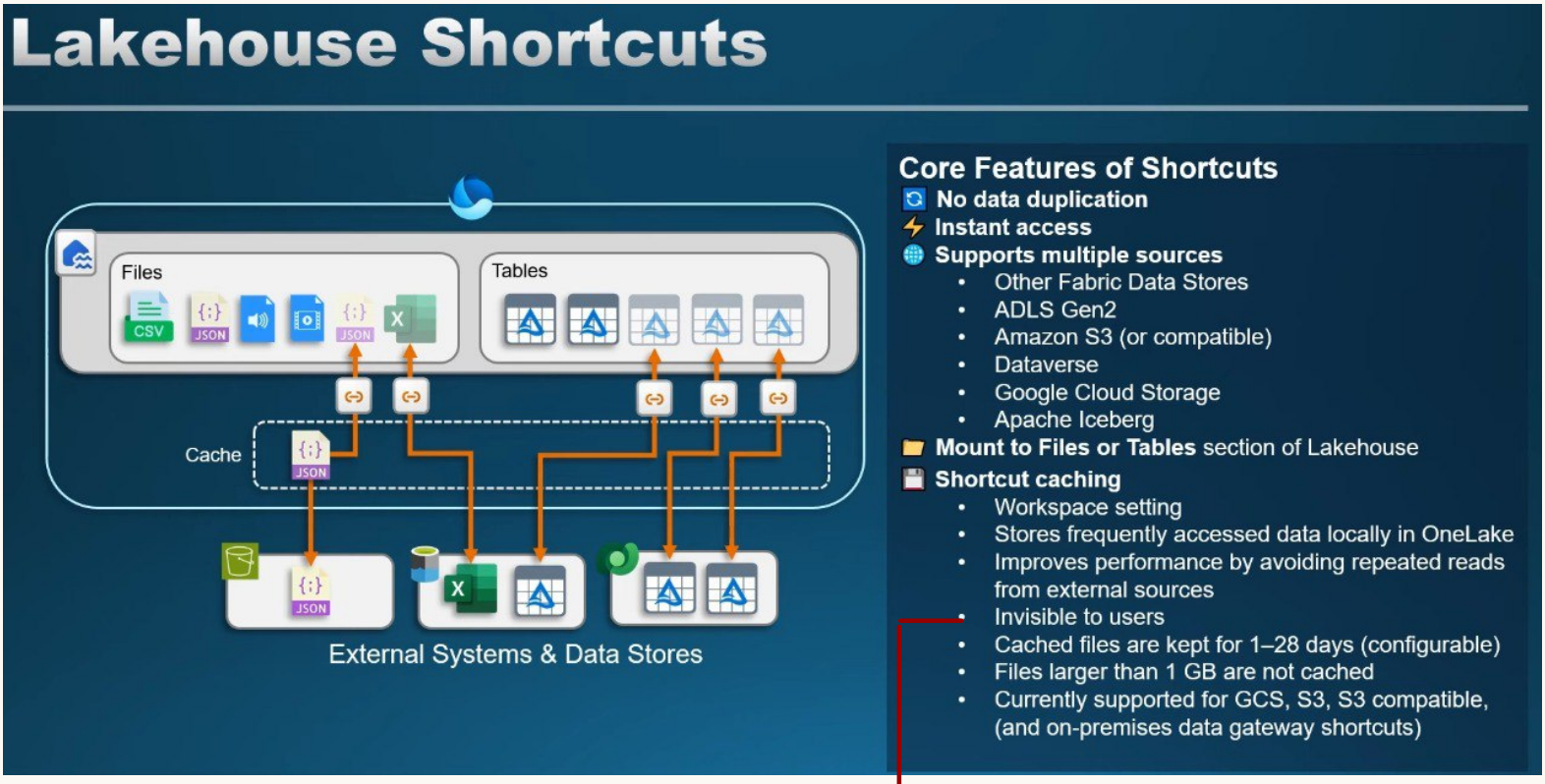
formats, Compression and more.

3- You can after That refrence The file as a table and use like spark to interact

* You can interact with lakehouse through pipelines, Notebooks lakehouse explor, Data flow Gen2, Apace spark jobs



Shortcuts in OneLake allow you quickly & easily source data from external cloud providers and use it across all fabric workloads



Is behind the scenes Copy of data managed entirely by Fabric

However:

each time these workloads read data from cross-cloud sources charges additional egress fees on the data.

Thankfully, Shortcut Caching allows the data to be sourced once and then used across all fabric workloads without additional egress fees.

stores local copy of the data inside OneLake after the first need.

retention period

defines: How long data is stored and available before it's automatically removed

Basics of Delta Tables

dbo.person

OneLake file structure

```
.../dbo/person/
  _delta_log
  0001.parquet
  0002.parquet
  0003.parquet
  0004.parquet
```

Table Representation

first_name	last_name
Aleksi	Partanen
John	Smith
Jane	Doe
Alex	Wang
Lisa	Bush
Jack	Doe

OneLake file structure
- Delta log keeps track of the parquet files
- Parquet files store the actual data

Operations
- INSERT
  - Adds new files
  - Updates delta log
- UPDATE / DELETE / MERGE
  - Doesn't modify files directly
  - Marks old files as invalid
  - Writes new files
  - Updates delta log

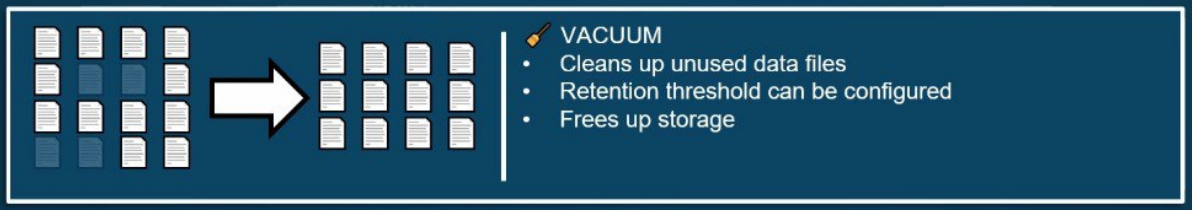
Some of the special features
- Time travel
- ACID transactions
- Schema evolution

schema evolution

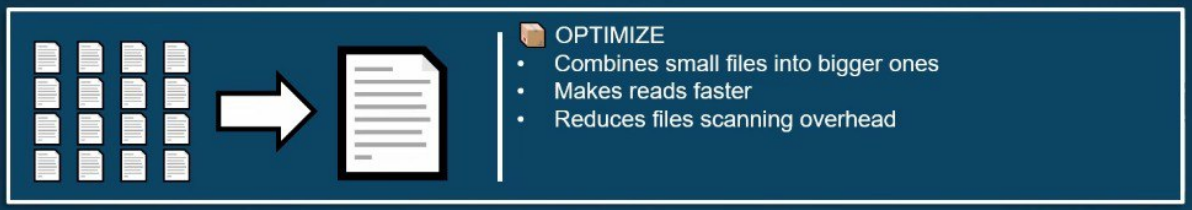
- ability of data system to handle changes in data structure over time without breaking existing pipelines, queries, or reports

- allow systems to adapt safely to changes in schema:
  new fields added, data type changes, columns are removed

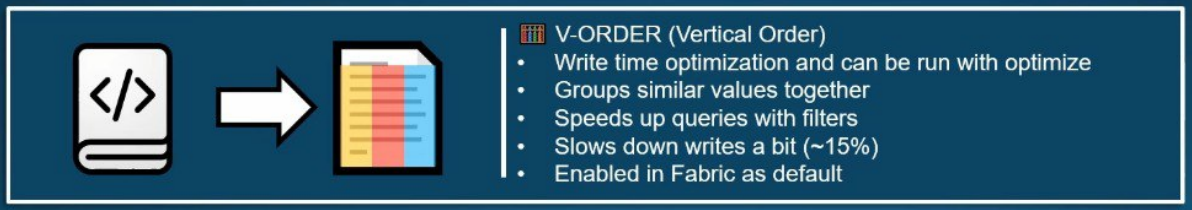
Maintenance & Optimization



VACUUM
- Cleans up unused data files
- Retention threshold can be configured
- Frees up storage



OPTIMIZE
- Combines small files into bigger ones
- Makes reads faster
- Reduces files scanning overhead



V-ORDER (Vertical Order)
- Write time optimization and can be run with optimize
- Groups similar values together
- Speeds up queries with filters
- Slows down writes a bit (~15%)
- Enabled in Fabric as default

Optimize Command

- Consolidate multiple smaller parquet files into fewer larger files

- Reading lots of small files leads to more overhead & reduced performance, while larger files have better data distribution & compression

- we can add appling V-order beside it Optimize in UI

- optimize Command takes time to run as it's potentially quite heavy on resources

- after appling optimize Command:

  1) in delta_log, we find new added json file state That Previous Parquet files are removed and packed in N file(s)

     and a metric column with json data is returned, where it show what happened during optimization (How many files added, remov

  2) Previous Parquet files are not actually removed to use time travel feature on need

- optimize Command is Idempotent → Nothing happen on executing it a second time.

- you can add a predicate with The Command to optimize only Part of table. However, The predicate needs to be applied to a Partitioning Coloumn.

Vacuum Command

- As you see using optimize command, the old files are not removed and essentially total storage increased
- we can then use Vacuum to remove old unneeded files
- Vacuum can break the "time travel" feature.
- we can use retention threshold to control the deletion based on it
  - specify age of files that should be deleted based on it (maximum 7 days)

V-Order

- write time optimization technique
- It applies special sorting, encoding, row group distribution, and compression to achieve about 50% more compression at the cost of some write overhead (15%)
- V-order is compatible with open parquet format, so any engine can read parquet files, can read it with V-order.
- disabled by default

- table property reduces number of parquet files written when delta table is created & increases file size

  - comes with extra processing cost on writting, but improves reading performance as fewer files must be read

  - boosts Vacuum & Optimize Commands
  - advised to be used on partitioned table
  - enabled by default in Microsof fabric Runtime for Apache spark

Merge optimization

- use Merge statement to update/modify delta tables with advanced options
- It update data from a source table, view or dataframe into a target table by using Merge Command
- The open-source distribution of delta is not fully optimized for handling unmodified data (even matched rows are shuffeled across spark nodes)

  with this feature, an optimization called "Low Shuffle Merge" is done to execlude unmodified rows from an expensive shuffling operation

- enabled by default.

Fablic Datawarehouse

- lake centric warehouse ⟶ Fabric datawarehouse stores data directly in the data lake using open formats like parquet which enables:
  - Easily handles massive data volumes
  - Different engineers can access the same data with different tools
  - Data doesn't need to be copied (No data duplication)
- The datawarehouse use a high-performance distributed processing engine.
- Data stored in Delta-parquet format
- Datawarehousing Items:

Fabric Datawarehouse is a lake warehouse that supports 2 distincts warehousing items: . The Fabric data warehouse &
. The SQL analytics endpoint

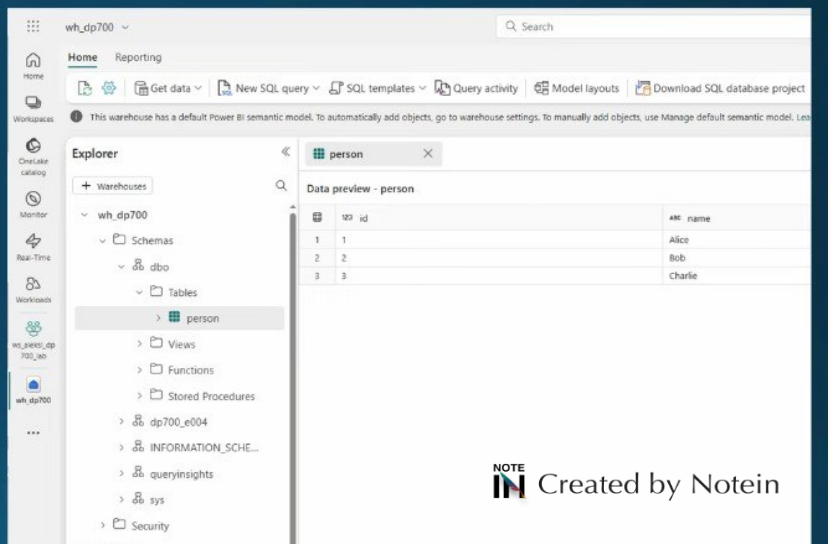
Warehouse

Key Points

- Feels like a **traditional relational database**
- **Stores data in Delta Parquet format** and stores data to Fabric OneLake under the hood
- Built for **structured, tabular data**
- Not designed for **raw or unstructured files**
- Ideal for the **Silver and Gold layers** of the **Medallion architecture**
- Best suited for **cleaned and transformed data**, not raw source files
- Familiar to anyone who's used **SQL Server** or other relational databases
- Powered by the **cloud** and an **open storage format**

Objects

- **Schemas**: Group related objects (like folders)
- **Tables**: Store actual data
- **Views**: Saved SQL queries that behave like virtual tables
- **Functions**: Return values from input (e.g., fiscal quarter logic)



- **Stored Procedures:** Encapsulate SQL logic for tasks and automation

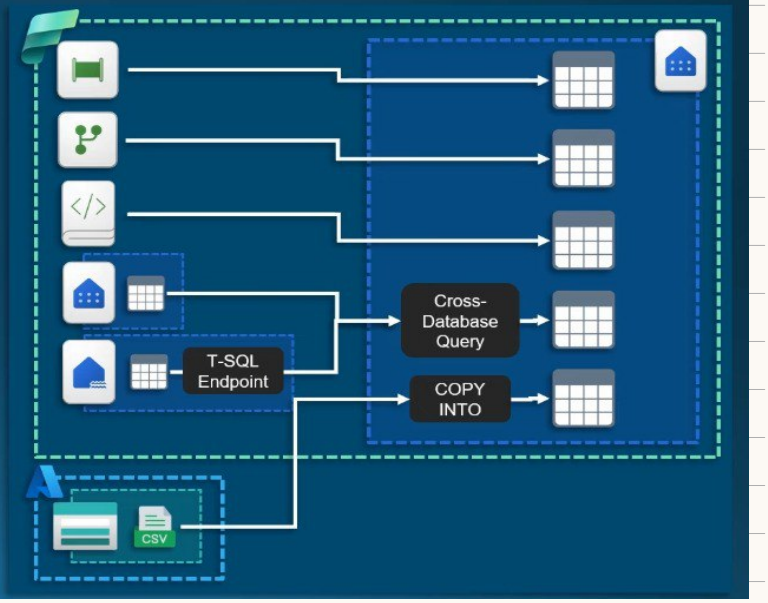
- The same feature in lakehouse:

A Warehouse, SQL database, and Fabric One lake all automatically provision a SQL analytics endpoint when created.

with it, T-SQL commands can define and query data objects only (~~Read only~~)

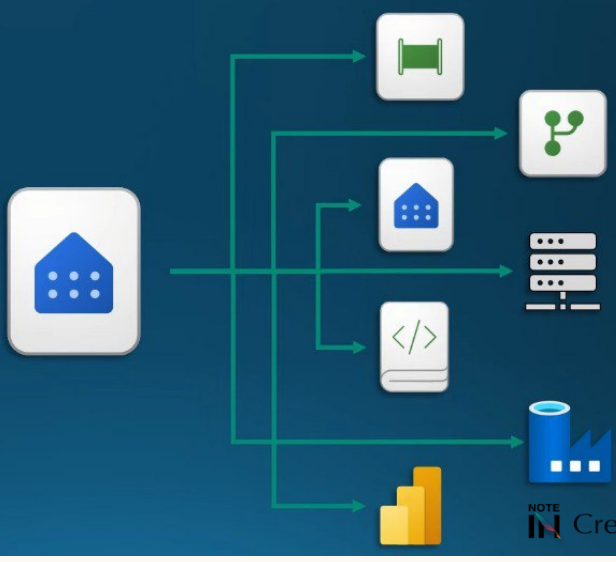
Ingesting Data to Warehouse

- Data pipelines
- Dataflow Gen2
- Notebooks
  - T-SQL Notebooks
  - Spark Connector
  - ODBC and other more generic interfaces
- T-SQL
  - Cross-Database Queries
  - COPY INTO statement
- Other ways also that are not in the scope of DP-700
  - Fabric Copy Job
  - Azure Data Factory
  - ODBC, JDBC etc.



Reading Data from Warehouse

- Familiar T-SQL experience
- Power BI
- SQL Server Management Studio (SSMS)
- Data Pipelines
- Dataflows Gen2
- Notebooks
- Cross-Database Queries from another Warehouse
- Many other tools, services and apps
  - Copy Job
  - Azure Data Factory
  - ODBC, JDBC etc.



Time Travel & Table Cloning

Time Travel

- View table as it existed previously
- Recover data after accidental DELETE or TRUNCATE
- Less need for manual backups or versioning
- 30-day history window
- Great for audits, comparisons, or rollbacks

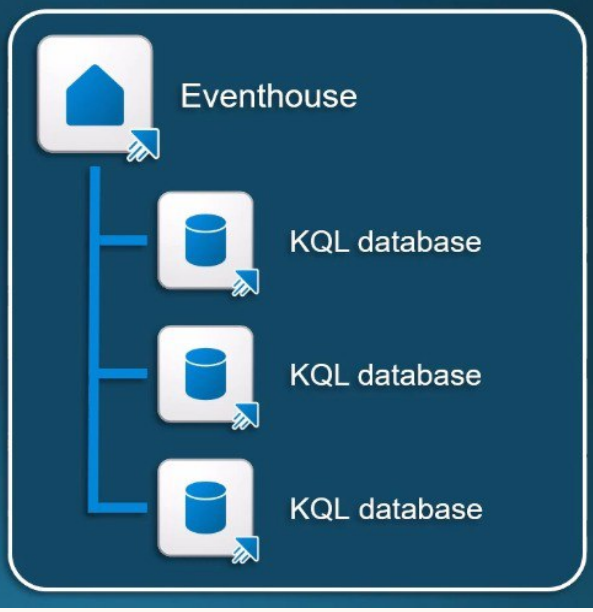
Table Clones

- Instantly create a copy of a table
- Can clone current or past version (uses time travel)
- Zero-copy clone -> Only metadata is copied
- Inherits properties like RLS, dynamic data masking etc.
- Currently only within the same Warehouse

Event House

Eventhouse Overview

- Eventhouse is a data store optimized for a real-time data and analytics
- Eventhouse is a logical grouping of KQL databases
- Each KQL database is optimized for fast event ingestion and high-speed analytics
- Ideal for working with streaming data, telemetry, clickstreams, and IoT events
- Kusto Query Language (KQL) is the primary way to query KQL databases
- T-SQL can be also used
- Data can be made available in OneLake for open access across Fabric services
- Supports flexible data ingestion from many sources, including OneLake storage, Eventstream, cloud storage like Azure Blob and Amazon S3, and Fabric tools like Pipelines and Dataflows.



Eventhouse Strengths and Weaknesses

Strengths

- Optimized for real-time streaming and telemetry data ingestion
- Designed for low-latency analytics with fast query responses
- Ideal for semi-structured or event-based data (e.g., JSON logs, IoT data)
- Highly scalable to handle massive data volumes (millions of events/second)
- Supports real-time dashboards, anomaly detection, and operational monitoring

Weaknesses

- Not designed for traditional relational data modeling
- Transaction support is limited compared to relational engines
- For more shorter-term data than multi-year archives
- Limited support for complex multi-table joins
- Mastering Kusto Query Language (KQL) is necessary to unlock full capabilities

Key Objects and Concepts

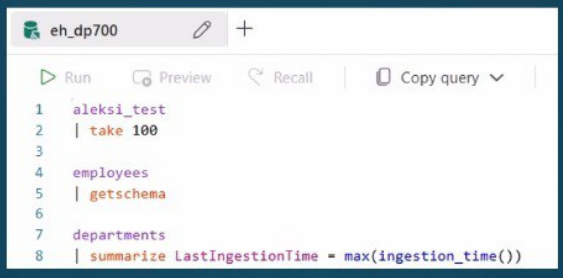
KQL Queryset
- A Fabric item that organizes multiple saved KQL queries

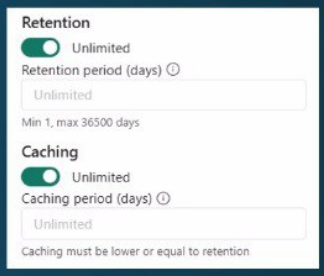
Tables
- Core storage unit where data is ingested and queried

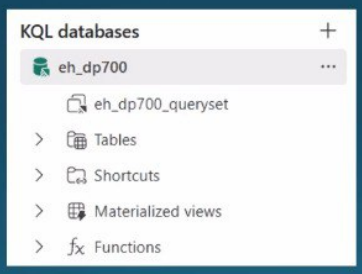
Materialized Views
- Precomputed query results to boost performance for dashboards and reports

Functions
- Saved reusable KQL query logic for simplifying complex queries

Shortcuts
- Query external data without duplicating it inside Eventhouse







Retention Policies
- Automatically manage how long data is kept before deletion

Caching Policies
- Improve query performance by caching frequently accessed data

OneLake Availability
- Make Eventhouse data accessible in OneLake

Materialized Views

- Precomputed query results stored as managed tables
- Automatically updated as new data is ingested
- Improve performance for dashboards and recurring queries
- Ideal for aggregation, filtering, or light transformations on streaming data
- Important properties to remember
  - backfill (boolean)
    - Include historical data
    - Defaults to False
  - effectiveDateTime (datetime)
    - Starting point for historical backfill

```
.create async materialized-view with (
    backfill = true,
    effectiveDateTime = datetime(2024-12-01T00:00:00Z),
    docString = "Daily website visits with backfill"
) e015_mv_daily_visits_with_backfill
on table e015_website_visits
{
    e015_website_visits
    | extend visit_day = bin(visit_timestamp, 1d)
    | summarize visit_count = count() by visit_day
}
```

OneLake Availability

- Default Eventhouse storage is optimized for fast querying inside Eventhouse, not automatically accessible by other Fabric items without using the KQL database engine
- OneLake availability enables Eventhouse data to be stored in OneLake as Delta Lake tables
- Can be enabled at the database level or individual table level
- Allows querying the data in your KQL database in Delta Lake format via other Fabric engines such as Direct Lake mode in Power BI, Warehouse, Lakehouse, Notebooks, and more
- The Data retention policy of your KQL database is also applied to the data in OneLake
- When OneLake availability is enabled, you cannot rename tables, alter schemas, apply Row-Level Security, or delete, truncate, or purge data