Incremental View Maintenance (IVM)

Table of Contents

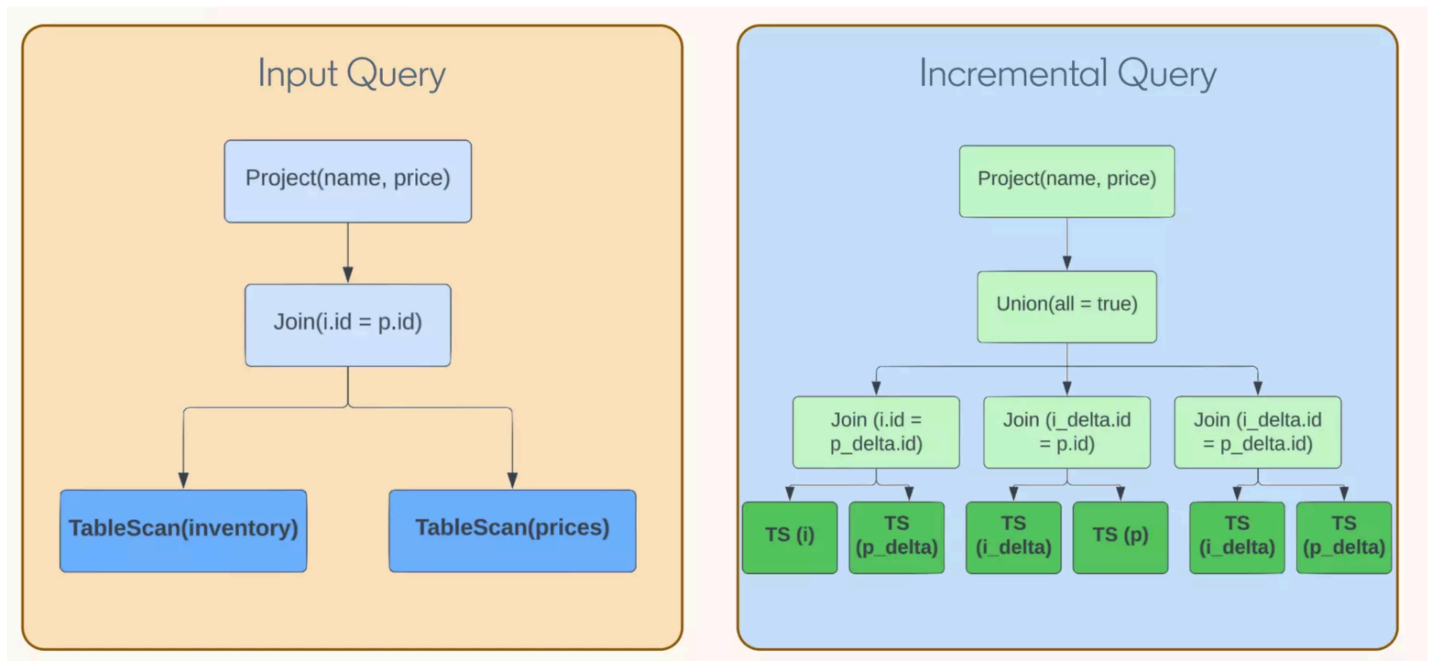
- Context
- Reference code
- Products
- OSS Implementations on top of Spark
- Academic Papers/Blogs/Videos

Context

Incremental View Maintenance is a super clever incremental data processing technique that applies the same differential calculus principles we learnt in school towards processing SQL queries efficiently:

> dy/dt = small approximate change

The idea is, (almost) any SQL query plan can be rewritten to be expressed as a full + diff - the trick is to knowing what the diff is since last run:



Once you figure out how to rewrite the query - e.g. for Spark SQL, you can use `coral-incremental` (see below). Others, like DBSP, parse the AST and implement the processing logic using a novel

Circuit (like electrical circuits) abstraction - crates/dbsp/src/circuit/:

Table 1: Implementation of SQL relational set operators as circuits computing on ℤ-sets.

Operation	SQL example	DBSP circuit	Details
Composition	`SELECT ... FROM (SELECT ... FROM I)`	I → C_I → C_O → O	C_I circuit for inner query, C_O circuit for outer query.
Union	`(SELECT * FROM I1) UNION (SELECT * FROM I2)`	I1, I2 → ⊕ → *distinct* → O	*distinct* eliminates duplicates. An implementation of `UNION ALL` does not need the *distinct*.
Projection	`SELECT DISTINCT I.c FROM I`	I → π_c → *distinct* → O	Project each row with its weight unchanged. Add up weights of identical rows.
Filtering	`SELECT * FROM I WHERE P(...)`	I → σ_P → O	P is a predicate applied to each row. Select each row separately. If the row is selected, preserve the weight, else make the weight 0.
Cartesian product	`SELECT I1.*, I2.* FROM I1, I2`	I1, I2 → × → O	The weight of the pair (a,b) is the product of the the weights of a and b.
Equi-join	`SELECT I1.*, I2.* FROM I1 JOIN I2 ON I1.c1 = I2.c2`	I1, I2 → $\bowtie_{c1=c2}$ → O	Multiply the weights of the rows that appear in the output.
Intersection	`(SELECT * FROM I1) INTERSECT (SELECT * FROM I2)`	I1, I2 → ⋈ → O	Special case of equi-join when both relations have the same schema.
Difference	`SELECT * FROM I1 EXCEPT SELECT * FROM I2`	I1, I2 → ⊖ → ⊕ → *distinct* → O	*distinct* removes rows with negative weights from the result.

With Delta Lake or Iceberg, you basically time-travel and load the `(n-1)th` and `nth` version of the table into a temporary view, which becomes the `dy/dt` you inject into the incrementalized query.

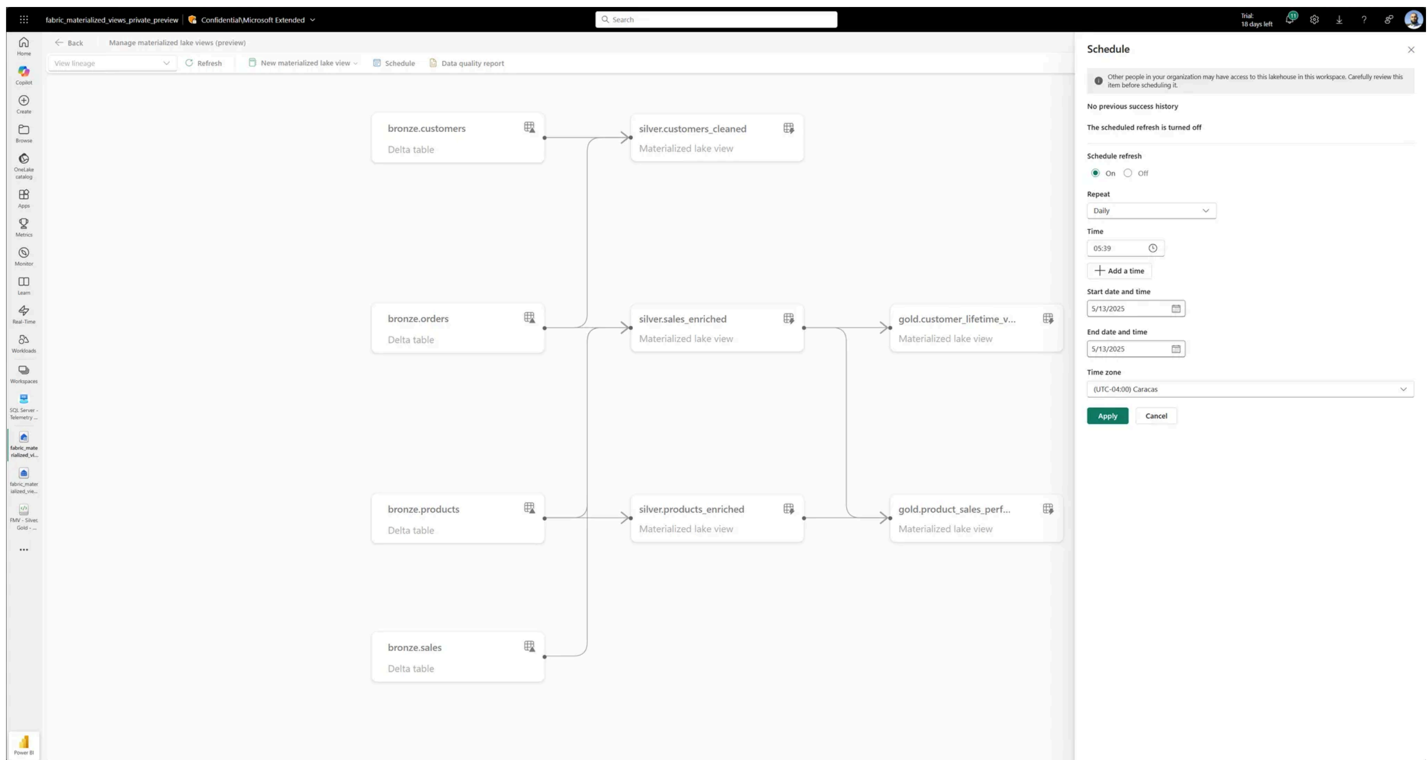
> Or, you can use the [Delta Lake Change Data Feed (CDF)](), which contains the same incremental commit information.

Sometimes it's cheaper to just do a full recomputation, so you need a Cost-based optimizer that makes a decision. If your pipeline runs as a Microbatch Stream continuously, it's probably always cheaper to run incrementally though.

You can implement this pretty easily as a plugin in dbt Labs (see example below with coral) or Tobiko's SQL Mesh (doesn't exist yet) to hook into the Spark engine that does the heavy lifting of time-travel loads and state persistence into stream checkpoints.

Reference code

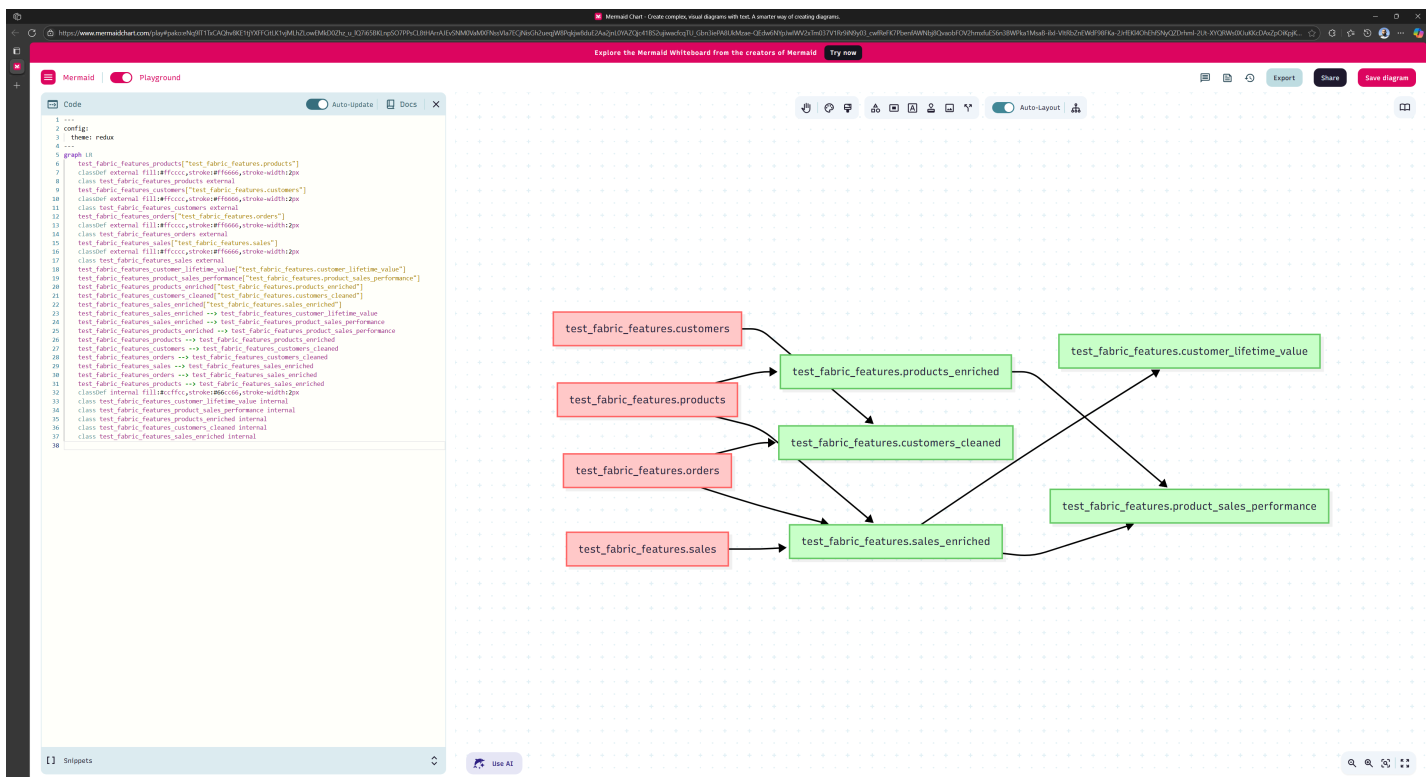
Currently we try to test the Fabric native IVM engine. In Fabric UI, the Directed Acyclic Graph looks like this:



> [A good mermaid visualization website]

See:

- To run an end-to-end demo of ivm capabilities locally: `com.microsoft.azurearcdata.sparkmsit.etl.drivers.demos.DemoIncrementalViewMaintenance`
- Graphviz based DAG creation: `com.microsoft.azurearcdata.sparkmsit.etl.FabricMaterializedViewIntegrationTest`



Products

- BigQuery Materialized View
- Clickhouse Materialized View
- Databricks Enzyme
- Databricks Materialized View
- Fabric Kusto Materialized View
- Fabric Spark Materialized View
- Feldera
- Firebolt Materialized View
- Materialize
- Snowflake Dynamic table
- Snowflake Materialized View
- SQL Server Indexed View

OSS Implementations on top of Spark

- Coral DBT
- Coral Incremental
- OSS Spark Declarative Pipelines (No IVM yet)

Academic Papers/Blogs/Videos

- DBSP: Incremental Computation on Streams and Its Applications to Databases
- DBSP: Automatic Incremental View Maintenance for Rich Query Languages
- DBSP: Taking the short path to streaming on the GPU with DBSP
- DBSP: Python package for any DataFrame
- DBSP: Notebook demo of academic paper
- Everything You Need to Know About Incremental View Maintenance
- Databricks Enzyme
- Coral Deck
- Coral Demo
- SPIP: Declarative Pipeline Framework for Apache Spark