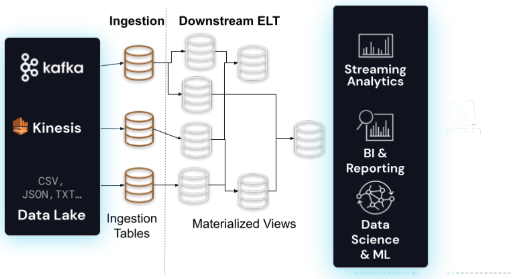
Making Data Engineering Declarative

Michael Armbrust, Bilal Aslam, Yingyi Bu, Sourav Chatterji, Yuhong Chen, Yijia Cui, Vuk Ercegovac, Ali Ghodsi, Rahul Govind, Aakash Japi, Kiavash Kianfar, Eun-Gyu Kim, Xi Liang, Paul Lappas, Jon Mio, Mukul Murthy, Supun Nakandala, Andreas Neumann, Yannis Papakonstantinou, Nitin Sharma, Yannis Sismanis, Justin Tang, Joseph Torres, Reynold Xin, Min Yang, Li Zhang
{firstname.lastname}@databricks.com

ABSTRACT[1]

ETL and ELT are required to prepare comprehensive, clean, and correct derived data that can fuel successful analytics and ML. Based on our observations from thousands of customers processing data in the cloud at Databricks, the preparation of derived data typically involves a complex DAG of transformations, which are split into two activities:



(a) *Ingestion:* At the sources of the DAG, raw data are fetched from streaming platforms, like Apache Kafka™ and Amazon Kinesis, and from cloud storage that stages incoming data. This data is typically in blob stores such as AWS S3. The majority of our customers store it in Delta Lake tables, the data format that enables transaction processing on data lakes [1,2].
(b) *Downstream ELT:* Multiple transformations clean, enrich, and aggregate the ingested data. The downstream ELT typically leads to many transformations, using popular frameworks such as DBT, or individual jobs that transform the data. In the former case, the transformations are typically in SQL as CREATE-TABLE-AS (CTAS) statements. In all cases, the underlying processing engine sees a series of seemingly independent SQL queries.
Delta Live Tables (DLT) [5] takes a different approach by introducing the concept of a *declaratively specified pipeline*. These pipelines, which are often quite complex, have a series of incrementally maintained SQL Materialized Views (MVs), instead of CTAS, to produce derived data. (See Figure.) They open up opportunities for a wide spectrum of novel optimizations and functionality:
(1) The large scope of needed transformations requires MVs that have the full SQL expressiveness (and even more). At the same time, the large volume of base data requires that the large SQL scope is combined with automated incrementalization (see survey [3]). The incrementalization planner builds upon the strength of the Catalyst query rewriter [4] to solve the problem.
(2) The large number of operations (ingestions and derivations) are efficiently orchestrated and auto-scaled. Conventional optimizers are not privy to the pipeline and merely optimize the steps one-at-a-time. In contrast, DLT holistically optimizes the full pipeline, reducing latency and maximizing infrastructure utilization. In the case of fully incremental downstream ELT, the pipeline-aware parallelism and autoscaling deliver >50% cost reduction in comparison to the same incrementalization operations executing as independent statements.
(3) The typical pipeline has a large number of lines of code and often more than one engineers collaborate on this code.[2] This requires the enablement of software engineering practices such as version control and Continuous Integration /Continuous Deployment (CI/CD). For CICD purposes, DLT treats the pipeline as one declarative specification. When a few lines are modified it falls on DLT's optimization to find which MVs are the same, which ones have changed, and how to evolve the prior MVs into the new ones.
(4) Dirty data are frequent in the sources. The user achieves high data quality by providing (i) declarative correctness expectations, which are automatically and efficiently monitored, and (ii) policies for the management of data that violate the correctness expectations.
In conclusion, similar to how SQL did for query processing, declarative pipelines open up multiple research opportunities in enabling optimizations and improving productivity.

https://doi.org/10.1145/1234567890

[2] The average pipeline DAG has >12 nodes, while the top 10% has >26 nodes.

REFERENCES

[1] Michael Armbrust, Ali Ghodsi, Reynold Xin and Matei Zaharia, 2021. Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. In CIDR 2021 http://cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

[2] M. Armbrust et al,2020, Delta Lake: High-Performance {ACID} Table Storage over Cloud Object. VLDB 2020, pg 3411-24. http://www.vldb.org/pvldb/vol13/p3411-armbrust.pdf. DOI 10.14778/3415478.3415560

[3] R. Chirkova and J. Yang, 2012, Materialized Views. In Foundations and Trends in Databases 4(4) pg 295-405. DOI 10.1561/1900000020

[4] M. Armbrust et al, 2015, Spark SQL: Relational Data Processing in Spark. In ACM SIGMOD 2015, pg 1383-94. DOI 10.1145/2723372.2742797

[5] Delta Live Tables https://docs.databricks.com/workflows/delta-live-tables/