SQL Day

14 edycja konferencji SQLDay

9-11 maja 2022, WROCŁAW + ONLINE

Data Community

partner złoty

Future Processing

Infinite DATA

KPMG

partner srebrny

Objectivity

summ-it

UBS

dbWatch DATABASE CONTROL

partner brązowy

elitmind think bright

Tomasz Krawczyk

AZURE DATA LAKEHOUSE
JUŻ ROK NA PRODUKCJI

ABOUT ME

Set your status

tkrawczyk
cloud4yourdata

Tomasz Krawczyk Azure Big Data Architect

Overview | Repositories 7 | Projects 0 | Stars 3 | Followers 2 | Following 0

Pinned — Customize your pins

- usql — C#
- CommunityEvents — CommunityEvents — C# ★1 1
- FP-DataSolutions/AzureDataLake — Azure Data Lake Training — C#
- AzureBigDataWorkshops

FP Data Solutions

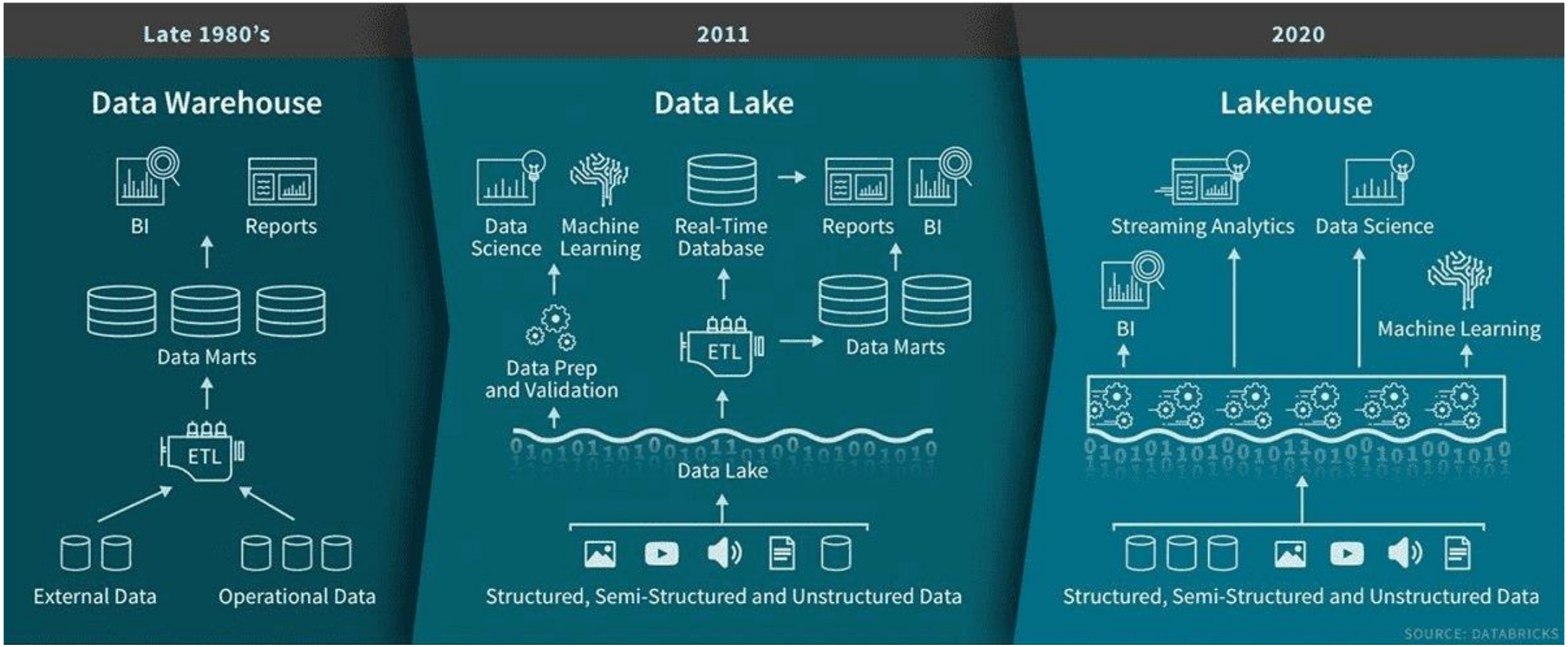
https://github.com/cloud4yourdata/SQLDay2022
https://github.com/cloud4yourdata/
https://github.com/FP-DataSolutions/

Data Community

AGENDA

- Data lakehouse
  - Concept and architecture
- Azure Data lakehouse – Challenges
  - Model materialization
    - Synapse CETAS, Custom Solution, Delta Live Tables
  - Processing optimization
  - Maintenance
  - Serving Layer – Databricks SQL Endpoints
- Demo(s)
- Q&A

WHAT IS A DATA LAKEHOUSE ?



Source: Databricks

WHAT IS A DATA LAKEHOUSE ?

Data Lakehouses on Oracle Cloud Infrastructure

A data lakehouse is a modern, open architecture that enables you to store, understand, and analyze all your data. It combines the power and richness of data warehouses with the breadth and flexibility of the most popular open source data technologies you use today. A data lakehouse can be built from the ground up on Oracle Cloud Infrastructure (OCI) to work with the latest AI frameworks and prebuilt AI services like Oracle's language service.

Easily bring together, analyze, and find new insights from all your data like invoices, forms, text, audio, and video.

BigLake PREVIEW Google Cloud

Built on years of investment in BigQuery, BigLake is a storage engine that allows organizations to unify data warehouses and lakes, and enable them to perform uniform fine-grained access control, and accelerate query performance across multi-cloud storage and open formats.

A data lakehouse is a data solution concept that combines elements of the data warehouse with those of the data lake. Data lakehouses implement data warehouses' data structures and management features for data lakes, which are typically more cost-effective for data storage. Data lakehouses are useful to data scientists as they enable machine learning and business intelligence.

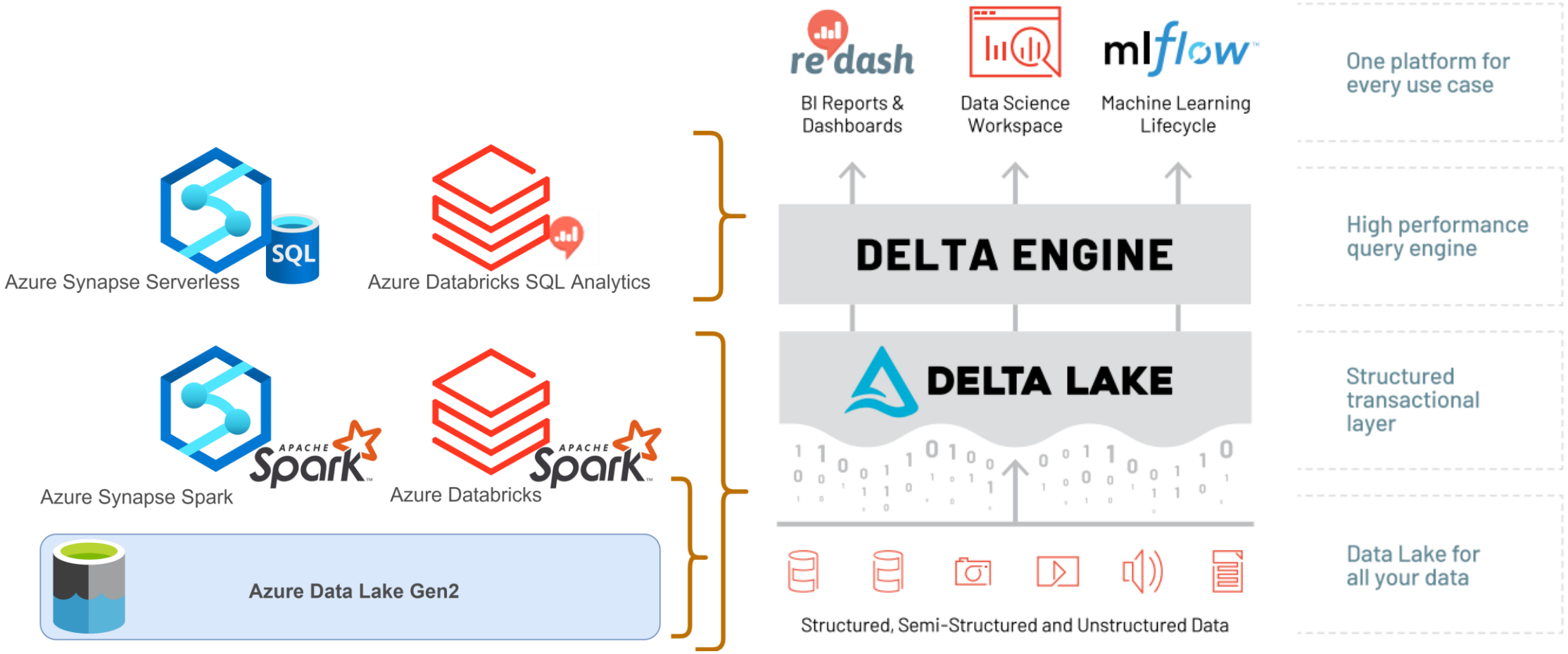
Microsoft

The Data Lakehouse, the Data Warehouse and a Modern Data platform architecture

By Greg Loxton

Published Mar 18 2022 03:20 PM 11.1K Views

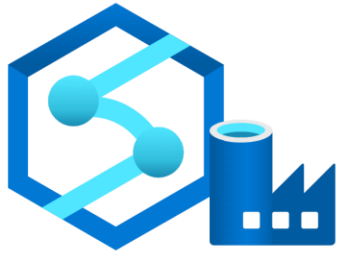
AZURE DATA LAKEHOUSE



AZURE DATA LAKEHOUSE

Azure Synapse Serverless

Azure Databricks SQL Analytics



Azure Synapse Integration Pipelines

Azure Databricks

Azure Key Vault

Azure Data Lake Gen2

DATA LAKEHOUSE - DATA FLOW

Source System(s)

- Relational databases
- Flat files (CSV)
- API (REST)

Landing Zone (Bronze)

- EL Processes:
  - Metadata framework
  - Azure Data Factory
  - Azure Synapse Integration Pipelines
- Data structure
  - Folders
  - Data partitioning
    - By load date
- Data Format
  - **Parquet (**)**
  - Native (Json, Avro, Csv)

Curated Zone (Silver)

- ETL Processes:
  - Metadata framework
  - Spark (Databricks)
- Transformations
  - **Slowly Changing Dimensions Type 2**
- Data Format
  - **Delta**
  - Spark Tables (one to one : source -> curated zone)
- Serving data
  - **Azure Synapse Serverless**
    - **Views on Delta Tables**
  - Azure Databricks SQL Analytics

Serve Zone (Gold)

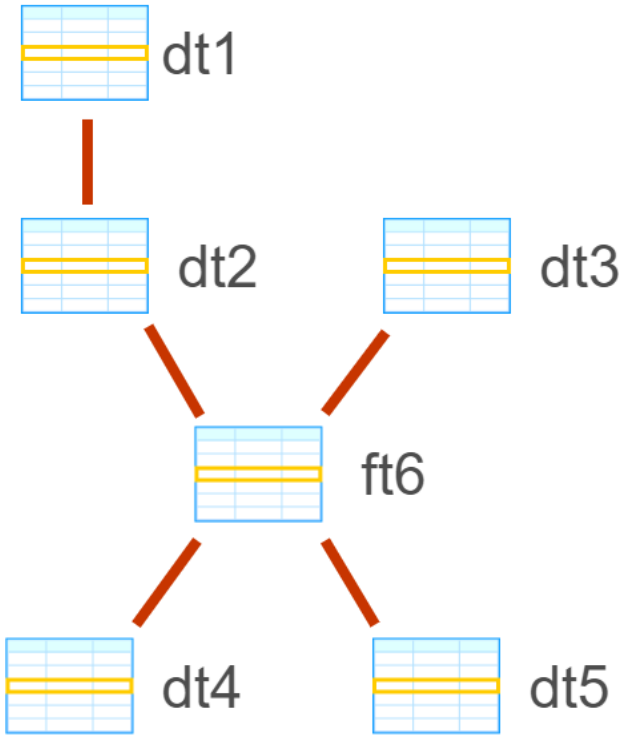
- ETL Processes:
  - Metadata framework
  - Spark (Databricks)
- Transformations
  - **Data Model (Kimball)**
- Data Format
  - **Delta**
  - Spark Tables
- Model
  - ~~Model based on views~~
  - **Materialized tables**
- Serving data
  - **Azure Synapse Serverless**
    - **Views on Delta Tables**
  - Azure Databricks SQL Analytics

DATA LAKEHOUSE – VIRTUAL MODEL

- Virtual Model = Views on Azure Synapse Serverless
- Azure Synapse Serverless
  - Support for Delta Format
    - ~~Parquet + manifest files~~
  - Problems with performance
    - **Query Performance Insights**
      - GitHub - JocaPC/qpi: Query Performance Insight - analyze performance of your SQL Server Database Engine

DATA LAKEHOUSE – MODEL MATERIALIZATION

- Azure Synapse Serverless
  - CTAS – Create Table As Select
    - Limited
- Azure Databricks
  - Custom solution
    - View -> Table (DDL)
    - View -> Merge->Table
  - Delta Live Tables
    - Now GA
    - Limited



DLH- MODEL MATERIALIZATION – CUSTOM SOLUTION

```sql
CREATE VIEW vwDimCustomer AS
SELECT
  c.CustomerID,
  c.PersonId,
  p.Title,
  p.FirstName,
  p.LastName,
  c.AccountNumber,
  c.ModifiedDate,
  c.dczIsDeleted AS IsDeleted
FROM
  DSAdventureWorks2019.Sales_Customer AS c
  LEFT JOIN DSAdventureWorks2019.Person_Person AS p
    ON c.PersonId = p.BusinessEntityID
  AND p.dczIsCurrent = 1
WHERE
  c.dczIsCurrent = True
```

```sql
CREATE TABLE IF NOT EXISTS `DLH`.`dimCustomer`
 USING DELTA PARTITIONED BY (DlhIsCurrent, DlhIsDeleted) AS
SELECT
  CAST(0 AS BIGINT) AS CustomerKey,
  *,
  CAST('2000-01-01' AS TIMESTAMP) AS DlhLoadDate,
  CAST('2000-01-01' AS TIMESTAMP) AS DlhUpdateDate,
  CAST('2000-01-01' AS TIMESTAMP) AS DlhValidFrom,
  CAST('2000-01-01' AS TIMESTAMP) AS DlhValidTo,
  CAST(1 AS BOOLEAN) AS DlhIsCurrent,
  CAST(1 AS BOOLEAN) AS DlhIsDeleted,
  CAST(1 AS INT) AS DlhVersion,
  CAST(1 AS INT) AS DlhDmlAction,
  CAST('' AS STRING) AS DlhRowHash,
  CAST(0 AS BIGINT) AS DlhInsertEtlProcLogId,
  CAST(0 AS BIGINT) AS DlhUpdateEtlProcLogId,
  CAST(0 AS BIGINT) AS DlhEtlProcLogId
FROM
  ModelSource.vwDimCustomer
WHERE
  1 <> 1
```

DLH - MODEL MATERIALIZATION – CUSTOM SOLUTION

Processing Type:

- SCD Type 1
- SCD Type 2
- Truncate/Insert

Configuration:

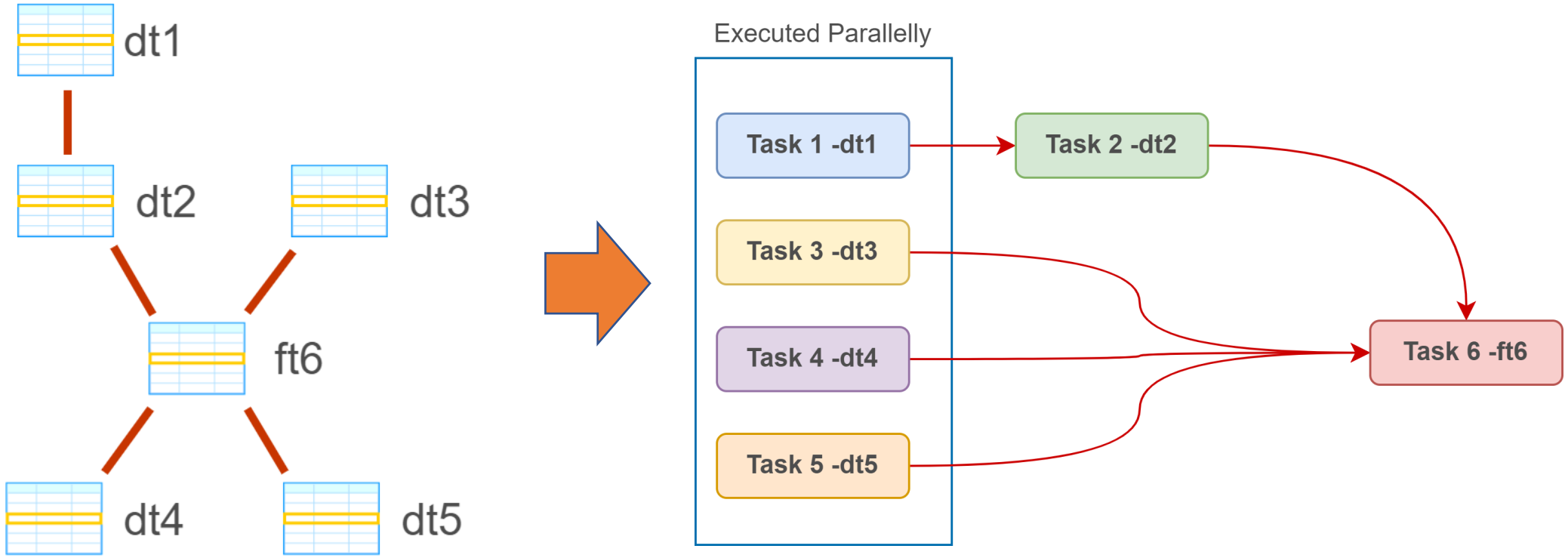
- Hash Columns
- Source Primary Keys

```
MERGE INTO `DLH`.`dimCustomer` AS trg USING (
  WITH CurrentLoad AS (
    SELECT
      (
        SELECT IFNULL(MAX(CustomerKey), 0) FROM `DLH`.`dimCustomer`
      ) + ROW_NUMBER() OVER (ORDER BY CustomerID) AS CustomerKey,
      FirstName,
      LastName,
      AccountNumber,
      ModifiedDate,
      CustomerID,
      PersonId,
      Title,
      IsDeleted,
      CURRENT_TIMESTAMP() AS DlhLoadDate,
      CAST(NULL AS TIMESTAMP) AS DlhUpdateDate,
      CAST('1900-01-01' AS TIMESTAMP) AS DlhValidFrom,
      CAST('9999-12-31' AS TIMESTAMP) AS DlhValidTo,
      TRUE AS DlhIsCurrent,
      IFNULL(IsDeleted, False) AS DlhIsDeleted,
      1 AS DlhVersion,
      NULL AS DlhDmlAction,
      SHA2(CONCAT_WS('||',FirstName,LastName,AccountNumber,ModifiedDate,PersonId,Title,IsDeleted),256) AS DlhRowHash,
      319 AS DlhInsertEtlProcLogId,
      319 AS DlhUpdateEtlProcLogId,
      319 AS DlhEtlProcLogId
    FROM
      ModelSource.vwDimCustomer AS stg
  )
  SELECT
    *
```

DLH - MODEL MATERIALIZATION – CUSTOM SOLUTION –DEPENDENCIES

Azure Databricks processing engine - Job Cluster

The Azure Databricks job scheduler creates a job cluster when you run a job on a new job cluster and terminates the cluster when the job is complete. You cannot restart a job cluster.



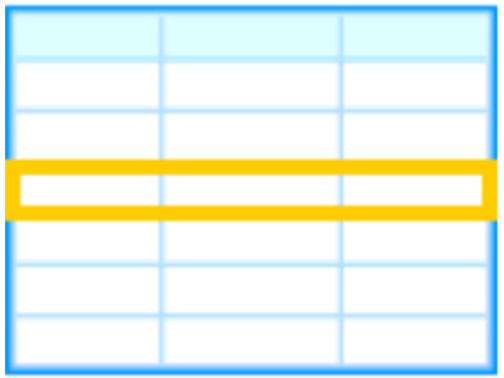
DATA LAKEHOUSE – SURROGATE KEYS

[Ralph Kimbal]
”Actually, a surrogate key in a data warehouse is more than just a substitute for a natural key. In a data warehouse, a surrogate key is a necessary generalization of the natural production key and is one of the basic elements of data warehouse design.”

Source: https://www.kimballgroup.com/1998/05/surrogate-keys/

Strategies:

- monotonically_increasing_id
- row_number() Rank OVER
- ZipWithIndex()**
- ZipWithUniqueIndex()**
- Row Hash with hash()
- **Generated Columns - IDENTITY**



DATA LAKAHOUSE – HASH

"A hash function takes a group of characters (called a key) and maps it to a value of a certain length (called a hash value or hash)"

```
WHEN MATCHED
AND trg.dwh_IsCurrent = true
AND trg.dwh_RowHash <> src.dwh_RowHash THEN
UPDATE
SET
  dwh_IsCurrent = false,
  dwh_ValidTo = src.dwh_ValidFrom
WHEN NOT MATCHED THEN
INSERT
```

hash

hash(expr1, expr2, ...) - Returns a hash value of the arguments. Examples: > SELECT hash('Spark', array(123), 2); -1321691492 Since: 2.0.0

sha

sha(expr) - Returns a sha1 hash value as a hex string of the expr . Examples: > SELECT sha('Spark'); 85f5955f4b27a9a4c2aab6ffe5d7189fc298b92c Since: 1.5.0

sha1

sha1(expr) - Returns a sha1 hash value as a hex string of the expr . Examples: > SELECT sha1('Spark'); 85f5955f4b27a9a4c2aab6ffe5d7189fc298b92c Since: 1.5.0

xxhash64

xxhash64(expr1, expr2, ...) - Returns a 64-bit hash value of the arguments. Examples: > SELECT xxhash64('Spark', array(123), 2); 5602566077635097486 Since: 3.0.0

https://spark.apache.org/docs/latest/api/sql/search.html?q=hash

DATA LAKAHOUSE – SPARK ANSI COMPLIANCE

THE SQL STANDARD - ISO/IEC 9075:2016 (ANSI X3.135) - ANSI BLOG

10/05/2018

SQL (standing for Structured Query Language) is the standard language for relational database ... SQL is used by numerous vendors, and, while most major vendors do modify the language to meet ...

https://www.ansi.org/search#q=sql&sort=relevancy

Using RDD (Spark)

```
data = sc.textFile(...).split("\t")
data.map(lambda x: (x[0], [int(x[1]), 1])) \
   .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
   .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
   .collect()
```

Using SQL (Spark)

```
SELECT name, avg(age)
 FROM people GROUP BY name
```

```
SET spark.sql.ansi.enabled = true
```

When true, Spark attempts to conform to the ANSI SQL specification:
Throws a runtime exception if an overflow occurs in any operation on an integer or decimal field.
Forbids using the reserved keywords of ANSI SQL as identifiers in the SQL parser.

DATA LAKEHOUSE – DELTA LIVE TABLES

- What is a Live Tables?
  - ~~Live Tables are materialized views for the lakehouse.~~
  - **Declarative ETL**
  - A Live Table is:
    - Defined by a SQL query (or Python)
    - Created and kept up-to-date by a pipeline

Create Pipeline

Name	Recent updates	ID
DLT2	⊗ ✓ ✓ ✓	58a56bc2-8f34-4668-a006-bc927e10d0e3
DWH_DLT	⊗ ⊗ ⊗ ✓ ✓	71ba4951-9ebf-4408-af28-990b35c2abbc
SensorRawData	✓ ✓ ✓	87ee21e6-17e6-4777-b0cd-146772a09d48

DATA LAKEHOUSE – DELTA LIVE TABLES

- Change data capture with Delta Live Tables
- MERGE = APPLY CHANGES
- Slowly Changing Dimensions Type2
  - Roadmap

```
APPLY CHANGES INTO LIVE.cities
FROM STREAM(LIVE.city_updates)
KEYS (id)
SEQUENCE BY ts
STORED AS SCD TYPE 2
```

SQL

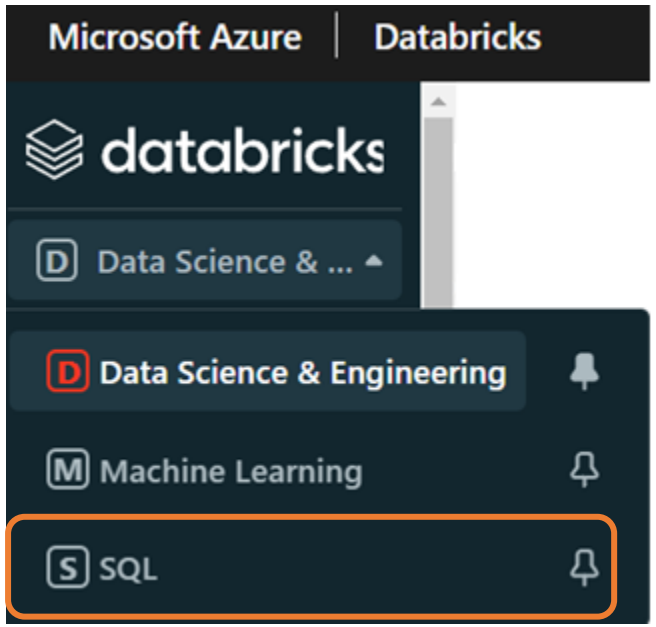
```
APPLY CHANGES INTO LIVE.table_name
FROM source
KEYS (keys)
[WHERE condition]
[IGNORE NULL UPDATES]
[APPLY AS DELETE WHEN condition]
SEQUENCE BY orderByColumn
[COLUMNS {columnList | * EXCEPT (exceptColumnList)}]
```

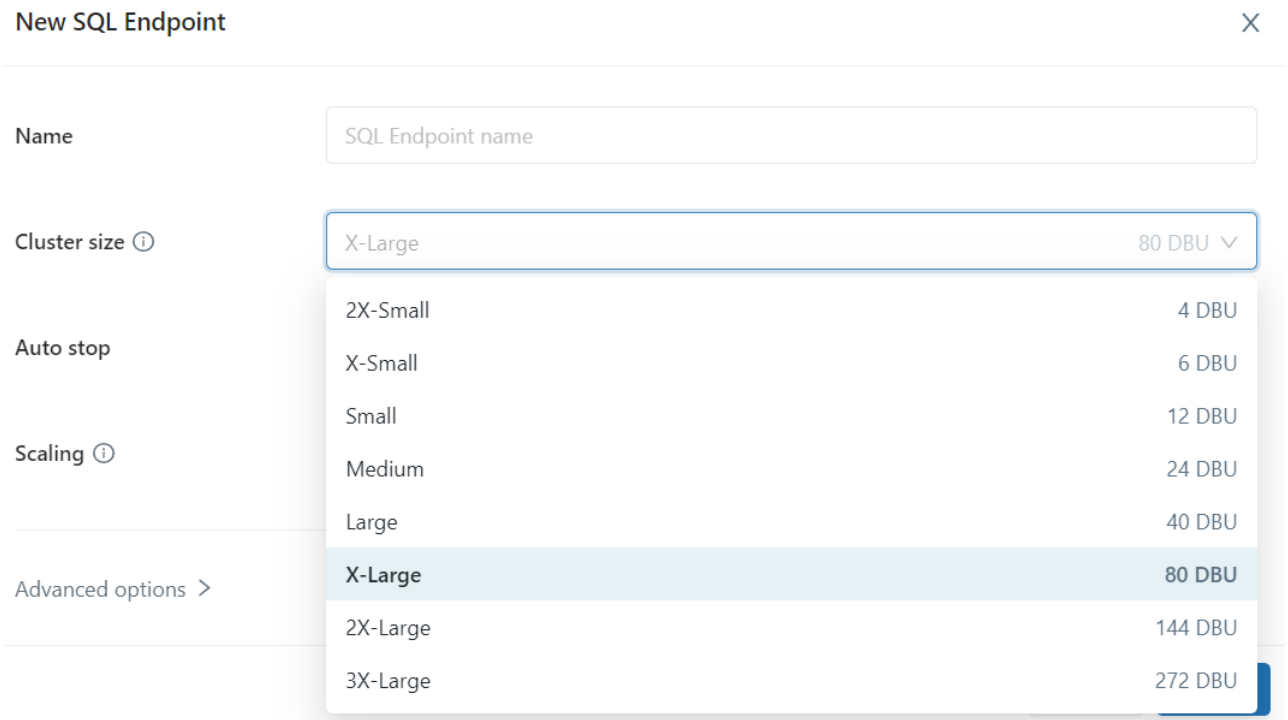
https://docs.microsoft.com/en-us/azure/databricks/data-engineering/delta-live-tables/delta-live-tables-cdc

Enhanced Autoscaling (preview)	●	●	●
Change Data Capture (CDC) - type 1		●	●
Change Data Capture (CDC) - type 2 (preview)		●	●

DATABRICKS SQL ANALYTICS

SQL Analytics allows customers to operate a multi-cloud lakehouse architecture that provides data warehousing performance at data lake economics.

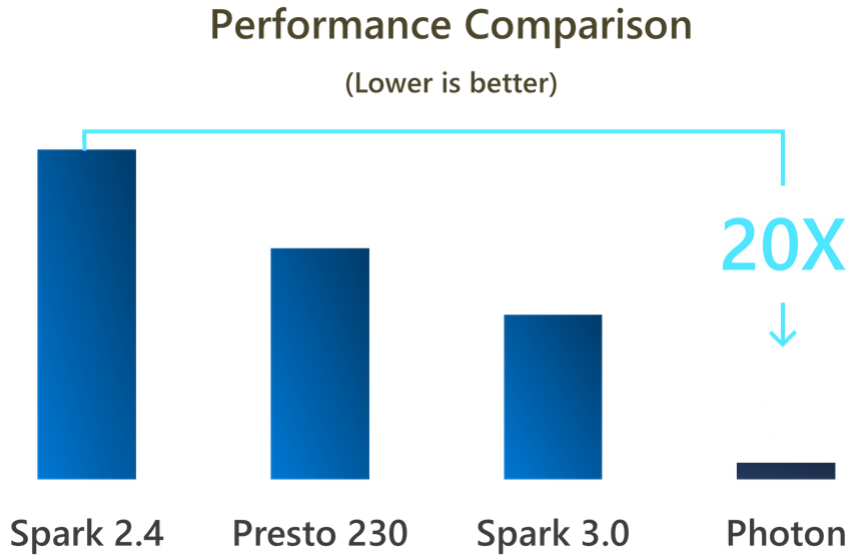




Source: https://databricks.com/product/sql-analytics

DATA LAKAHOUSE – PROCESSING OPTIMIZATION

- PHOTON ENGINE
- SPARK SETTINGS
- DATA PARTITIONING
- MERGE INTO
  - PARTITION PRUNNING
  - LOW SHUFFLE MERGE
- OPTIMIZE
  - Z-ORDER
- BLOOM INDEX
- INSERT OVERWRITE instead of TRUNCATE /INSERT
- SCHEMA EVOLUTION
  - CREATE OR REPLACE AS SELECT

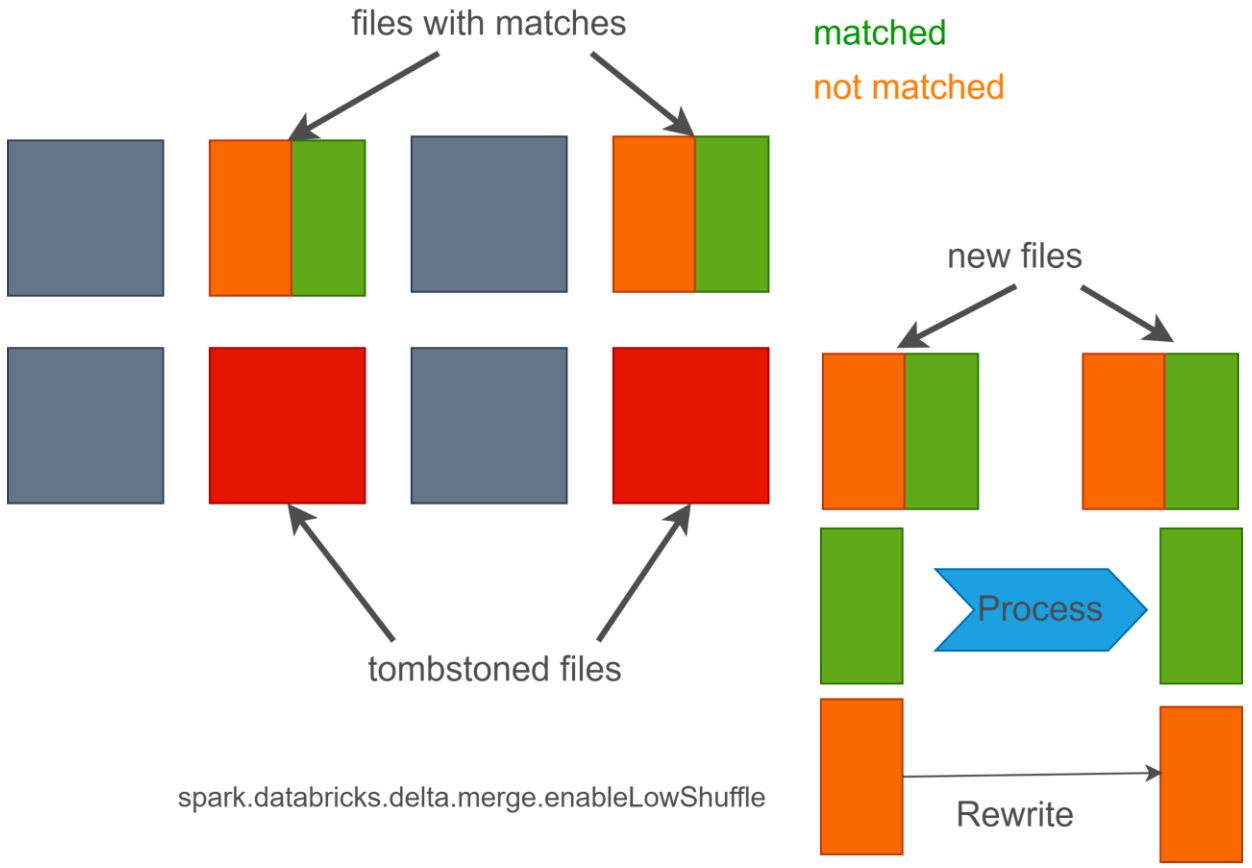


https://techcommunity.microsoft.com/t5/analytics-on-azure-blog/turbocharge-azure-databricks-with-photon-powered-delta-engine/ba-p/1694929

DATA LAKAHOUSE – MAINTENANCE

- DELTA TIME TRAVEL
  - DESCRIBE HISTORY, RESTORE
- OPTIMIZE
  - Z-ORDER
- VACUUM
  - DLT –Auto
  - Auto Vacuum –on the roadmap

MERGE OPERATION



AZURE DATA LAKEHOUSE -SUMMARY

- **Azure Data Lakehouse**
  - **Azure Data Lake Gen2 + Azure Databricks + Azure Synapse**
    - **Databricks –Spark compute Engine + SQL Endpoints**
    - **Synapse –Integration pipelines + SQL Serverless**
  - **Materialized model vs virtual model**
    - **Custom solution** (Spark)
    - **Delta Live Table (?)**
- **Generic Metadata Framework**
  - Generic Metadata Framework – case study. How to use it in a project? | Blog Future Processing (future-processing.com)

DEMOS

Q & A

Contact:

tomasz.k.krawczyk@gmail.com
tkrawczyk@future-processing.com

RESOURCES

My examples (and demos)

- https://github.com/cloud4yourdata/SQLDay2022

Blogs,pages,documentation, articles

- https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/azure-databricks-modern-analytics-architecture
- https://databricks.com/product/delta-live-tables
- https://docs.microsoft.com/en-us/azure/databricks/release-notes/runtime/releases
- https://docs.microsoft.com/en-us/azure/databricks/sql/user/security/access-control/sql-endpoint-acl
- https://www.jamesserra.com/archive/2021/01/data-lakehouse-defined/
- https://docs.microsoft.com/en-us/azure/databricks/delta/optimizations/auto-optimize
- https://pivotalbi.com/local-databricks-development-on-windows/

SQL Day

14 edycja konferencji SQLDay

9-11 maja 2022, WROCŁAW + ONLINE

Data Community

partner złoty

Future Processing

Infinite DATA

KPMG

partner srebrny

Objectivity

summ-it

UBS

dbWatch DATABASE CONTROL

partner brązowy

elitmind think bright