



PLATINUM SPONSOR

STRATEGIC PARTNER

TECHNOLOGY
INNOVATION
DATA
KNOWLEDGE



GOLD SPONSORS



CLOUDS ON MARS



SILVER SPONSOR



BRONZE SPONSOR

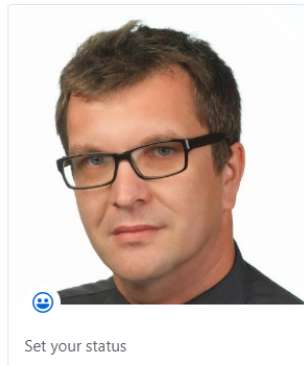


Partycjonowanie danych w rozwiązaniach Big Data

Tomasz Krawczyk

■ Future Processing

About Me



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



<https://github.com/cloud4yourdata>
{CommunityEvents}



Project

Projekt MDM - platforma zarządzania danymi z zaawansowanej infrastruktury pomiarowej



Ogólna koncepcja Platformy MDM

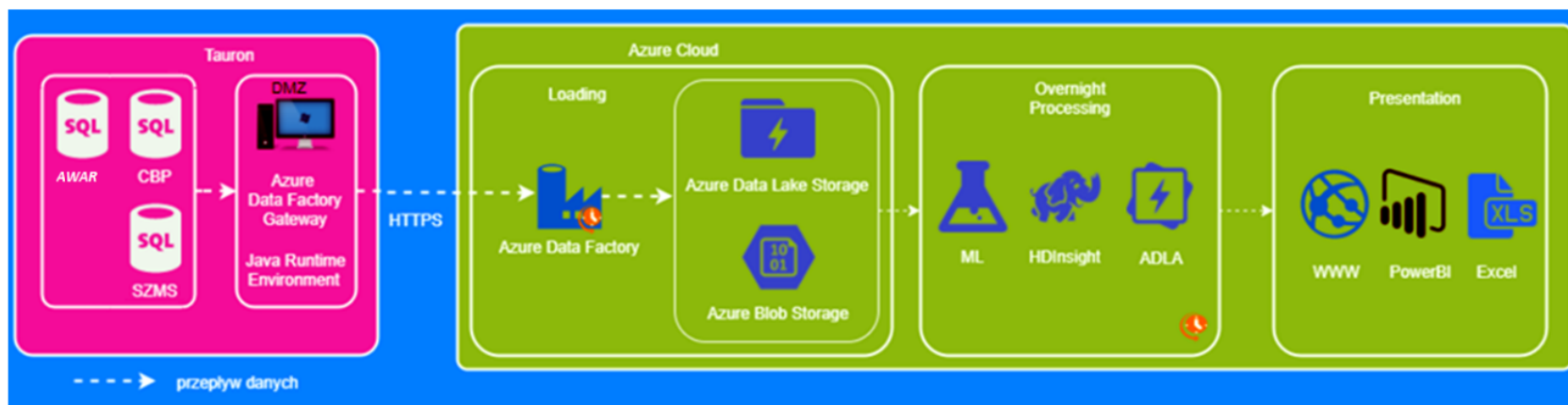
Celem i infrastrukturę obejmującą liczniki. Poniżej przedstawiono schemat ogólnej koncepcji Platformy MDM.

Obszar

Projekt

Klientów

wyznac



<https://www.tauron-dystrybucja.pl/o-spolce/innowacje-tauron/mdm>

Agenda

- Big Data Game
- **Big Data Solutions** (Azure)
- Hive and **Spark** (and Azure Data Lake Analytics)
- External and Manage Tables
- **Partitioning**
- **Bucketing**
- **Delta** and Delta Lake
- Demos

Big Data Game

Is it **Pokemon** or **Big Data** ?

Seahorse

Seahorse is Big Data!



Not to be confused with Horsea. Seahorse lets you create Apache Spark applications in a visual way.

<https://pixelastic.github.io/pokemonorbigdata/>

Big Data (3V)

| | |
|-----------------|----------------------------|
| Byte | One grain of rice |
| Kilobyte | Cup of rice |
| Megabyte | 8 bags of rice |
| Gigabyte | 3 semi trucks |
| Terabyte | 2 container ships |
| Petabyte | Blankets Manhattan |
| Exabyte | Blankets west coast states |
| Zettabyte | Fills the Pacific Ocean |
| Yottabyte | As earth-sized rice ball |



Data Volume

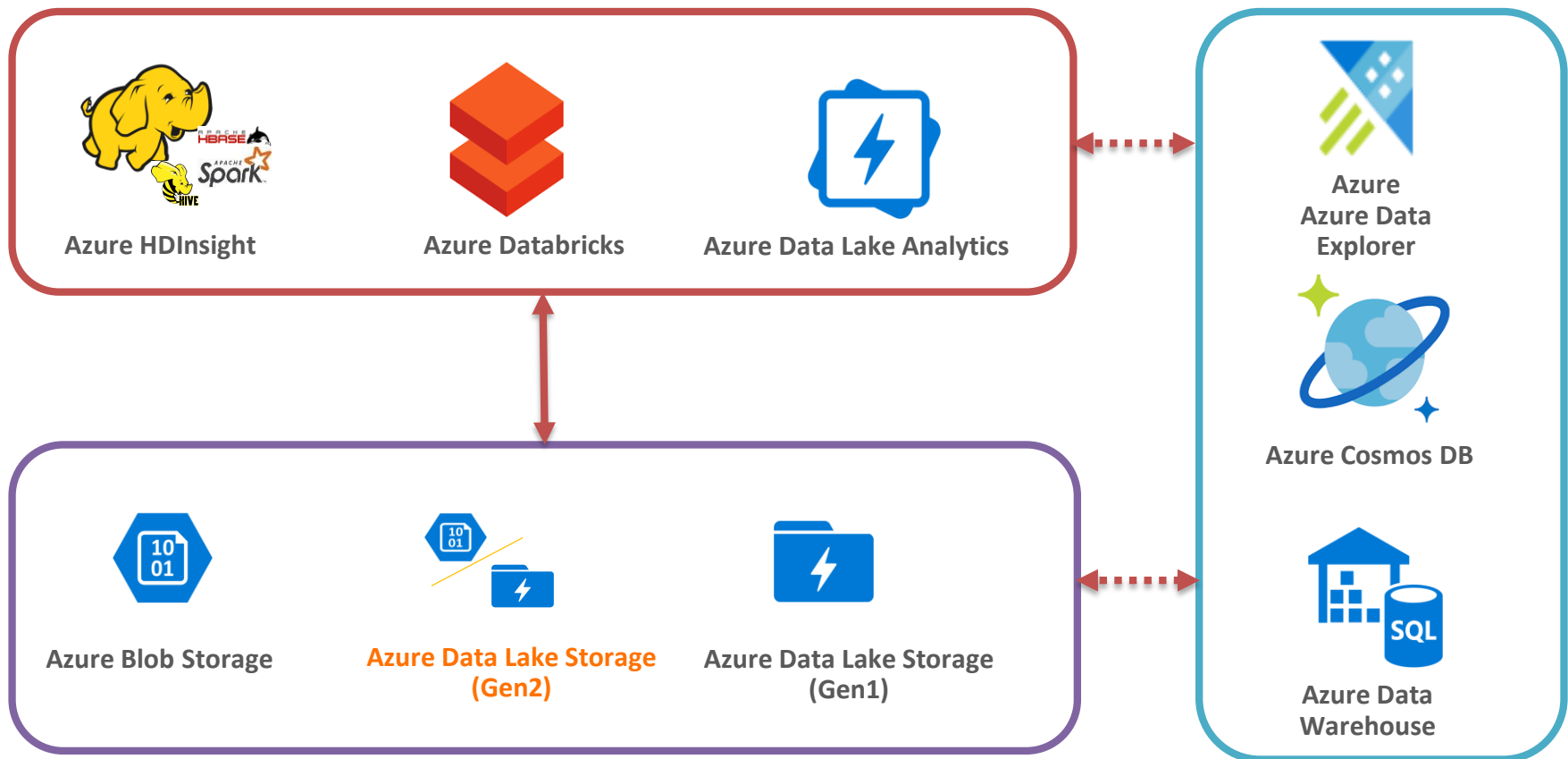


Data Variety



Data Velocity

Big Data Solutions and Azure



Challenge

Data Set:

adlademosadls ► SQLDay2019 ► Data

| NAME | SIZE |
|--|---------|
|  streetcrimes-000.csv | 1.34 GB |
|  streetcrimes-001.csv | 1.53 GB |
|  streetcrimes-002.csv | 1.68 GB |
|  streetcrimes-003.csv | 135 MB |



Azure Data Lake Storage
(Gen1)

```
"CrimeID","DateReported","MonthReported","YearReported","ShortReportedByPoliceForceName","ReportedByPoliceForceName","FallsWithinPoliceForceName","Longitude","Latitude","Location","DistrictCode","DistrictName","CrimeType","Outcome"  
"bd106bd30d971f4f0b0a7266c7e2ea28be38445116399d5f7d58745c7cd733d",2017-01-01T00:00:00.0000000,1,2017,"hertfordshire","Hertfordshire Constabulary","Hertfordshire Constabulary",-0.572545,51.819252,"On or near B4506","E01017701","Aylesbury Vale 009E","Criminal de  
"5f8cd40ab19530450da6780b327f8f5b1eda106545b46f8235e856bf9a2e20",2017-02-01T00:00:00.0000000,2,2017,"hertfordshire","Hertfordshire Constabulary","Hertfordshire Constabulary",-0.224338,51.6575165,"On or near Sports/Recreation Area","E01000253","Barnet 007B","I  
"",2017-03-01T00:00:00.0000000,3,2017,"hertfordshire","Hertfordshire Constabulary","Hertfordshire Constabulary",-0.667245,51.81772,"On or near Parking Area","E01017651","Aylesbury Vale 009A","Anti-social behaviour",""  
"e0e3025d30becb4f11bb00cd892eed23f38c247faeb07335fac1fca1c126bd",2017-04-01T00:00:00.0000000,4,2017,"hertfordshire","Hertfordshire Constabulary","Hertfordshire Constabulary",-0.710269,51.79715,"On or near Dennis Close","E01017629","Aylesbury Vale 021C","Drug  
"4367c5800d422f551457f4b0aab375dd00886c4aee95bee50deb5c0f08c09adi",2017-05-01T00:00:00.0000000,5,2017,"hertfordshire","Hertfordshire Constabulary","Hertfordshire Constabulary",-0.001143,51.77788,"On or near Ryefield Close","E01023316","Broxbourne 001A","Vehicle
```

FILTER (PROCESS):

YearReported = 2017 AND MonthReported = 10

Big Data Solutions



Azure Data Lake Analytics is an on-demand analytics job service that simplifies big data. (Legacy)



The Apache Hive™ data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL.



Apache Spark™ is a unified analytics engine for large-scale data processing

- Scala, Python, R, **SQL and .Net**

Big Data and SQL

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
```

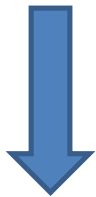
„Phoenix Puts the SQL Back in NoSQL“



Data Processing

Batch mode vs Interactive mode

Batch Mode



- U-SQL Job/Query
- HQL Job
- Spark Job



Azure Data Lake Analytics



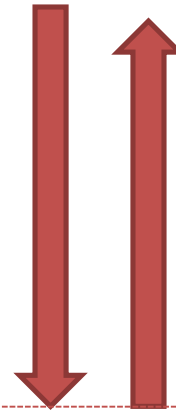
(LLAP)

HIVE



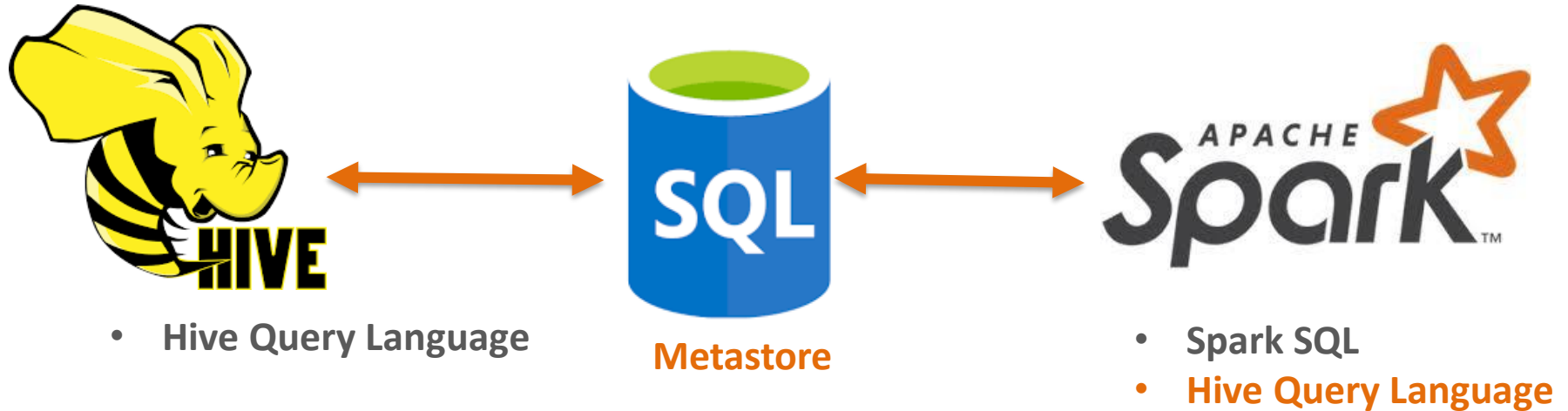
Storage

Interactive Mode



SQL Query
(ODBC/JDBC)

Hive and Spark SQL



Hive **Metastore** allows mapping database structures (database, table, partition, column) to HDFS directories and files

Hive and Spark SQL -Tables

- External (unmanaged) tables
- Managed (internal) tables

%sql

DESCRIBE EXTENDED {TABLENAME}

| | |
|------------|-------------|
| Created By | Spark 2.4.0 |
| Type | EXTERNAL |
| Provider | CSV |

| | |
|------------|-------------|
| Created By | Spark 2.4.0 |
| Type | MANAGED |
| Provider | CSV |

```
CREATE TABLE IF NOT EXISTS  
StreetCrimes
```

```
(  
  CrimeID STRING,  
  DateReported TIMESTAMP,  
  Month INT,  
  Year INT,
```

```
  ...
```

```
)
```

```
  USING CSV
```

```
  OPTIONS(header 'true')
```

```
  LOCATION
```

```
  'dbfs:/mnt/SQLDay2019/'
```

Providers: CSV, ORC, PARQUET, DELTA, **JDBC**

ADLA

- Demo-001.usql

Spark

- 001-ExtAndManagedTables

DEMO

Partitioning and Partition Elimination

Partitioning is a way of dividing a table into related **parts** based on the values of partitioned columns.

▼ PartitionedData

▶ year=2015

▶ year=2016

▶ year=2017

▼ year=2018

▶ month=1

Partition year

Sub partition month

- **Azure Event Hub**
 - (Event data capturing)
- **Azure Stream Analytics**
 - (Output)
- **Azure Data Factory**
 - Load into Azure Storage

{Partition Key}={Partition Value}

ADLA Partitioned Output

[AzureDataLake](#) / [docs](#) / [Release_Notes](#) / [2018](#) / [2018_Spring](#) /

Data-driven Output Partitioning with **OUTPUT** fileset is in Private Preview

We have started a private preview of the data-driven output partitioning with filesets. It enables you to use column values to create parts of a file path and partition the remaining data into different files based on these values.

This feature addresses the following user-voice feedback: [Support 'dynamic' output file names in ADLA](#)

Please [contact us](#) if you want to try it out and provide us with your feedback.

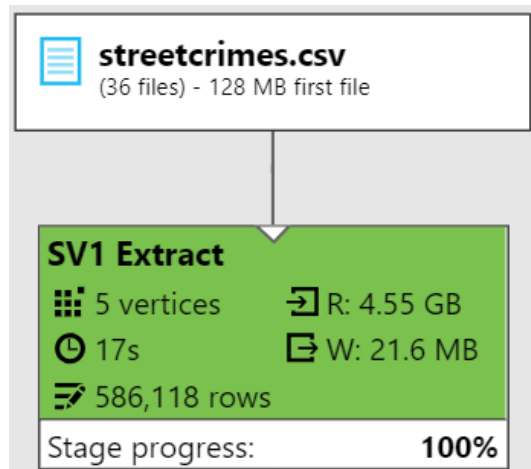
```
SET @@FeaturePreviews = "DataPartitionedOutput:on";
...;
DECLARE @outputPath string =
@" /SQLDay2019/Out/PartitionedData/year={YearReported}/month={MonthReported}/
streetcrimes.csv";
...
OUTPUT @inputData
      TO @outputPath
      USING Outputters.Csv();
```

ADLA- Read Partitioned Data (Partition Elimination)

```
DECLARE @inputPath string = @"../year={Year}/month={Month}/{*}.csv";
```

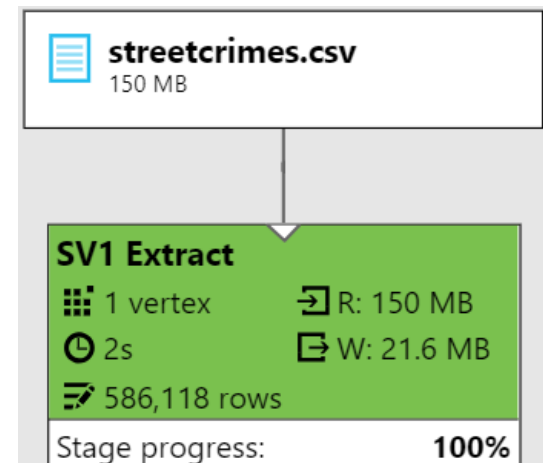
```
@inputData =  
EXTRACT...  
    Year string,  
    Month string;
```

```
@outputData =  
    SELECT * FROM @inputData WHERE  
        Convert.ToInt32(Year) == 2017  
    AND Convert.ToInt32(Month) == 10;
```



```
@inputData =  
EXTRACT...  
    Year int,  
    Month int;
```

```
@outputData =  
    SELECT * FROM @inputData WHERE  
        Year == 2017  
        Month == 10;
```



Hive and Spark - Read Partitioned Data



```
CREATE EXTERNAL TABLE IF NOT EXISTS StreetCrimesPartitioned
(
  CrimeID STRING,
  DateReported TIMESTAMP,
  Month INT,
  Year INT,
  ...
)
USING CSV
OPTIONS(header 'true')
PARTITIONED BY (Year,Month)
...
```



```
CREATE EXTERNAL TABLE IF NOT EXISTS StreetCrimesPartitionedHive
(
  CrimeID STRING,
  DateReported TIMESTAMP,
  ...
)
PARTITIONED BY (year INT, month INT)
...
```

MSCK REPAIR TABLE { TableName }

Data Formats

CSV(TEXT)

```
2017-10-01T00:00:00.0000000,"E01012289","Shoplifting"  
2017-10-01T00:00:00.0000000,"E01010781","Anti-social behaviour"  
2017-10-01T00:00:00.0000000,"","Violence and sexual offences"  
2017-10-01T00:00:00.0000000,"E01009421","Anti-social behaviour"  
2017-10-01T00:00:00.0000000,"E01007502","Anti-social behaviour"  
2017-10-01T00:00:00.0000000,"E01014809","Shoplifting"  
2017-10-01T00:00:00.0000000,"W01000565","Violence and sexual offences"  
2017-10-01T00:00:00.0000000,"E01025306","Anti-social behaviour"  
2017-10-01T00:00:00.0000000,"E01009249","Violence and sexual offences"  
2017-10-01T00:00:00.0000000,"E01013614","Vehicle crime"  
2017-10-01T00:00:00.0000000,"E01012520","Criminal damage and arson"
```



Parquet

Free and open-source column-oriented data storage format of the Apache Hadoop ecosystem.



Apache
orc

The smallest, fastest columnar storage for Hadoop workloads

Hive and Spark SQL – Managed Tables , Static and Dynamic Partition

- **Static Partitions**

```
INSERT INTO
StreetCrimesPartitionedParquet
PARTITION(Year=2018,Month=1)
SELECT CrimeID,
DateReported,
...
FROM StreetCrimesPartitioned
WHERE Year =2018 AND Month = 1
```

- **Dynamic Partitions**

```
INSERT INTO
StreetCrimesPartitionedParquet
PARTITION(Year,Month)
SELECT CrimeID,
DateReported,
...
Year, --Partitioned Column
Month --Partitioned Column
FROM StreetCrimesPartitioned
WHERE Year <2018
```

Key=__HIVE_DEFAULT_PARTITION__

ADLA

- Demo-002.usql
- Demo-003.usql
- Demo-004.usql

Spark

- 002-ExtPartitionedTable
- 003-Partitions

DEMO

Challenge

DATA SET 1 (Crime Location)

adlademosadls ► SQLDay2019 ► Out ► CrimesLocations

| NAME | SIZE |
|--|--------|
|  CrimesLocation_1.csv | 170 MB |
|  CrimesLocation_10.csv | 181 MB |
|  CrimesLocation_11.csv | 175 MB |
|  CrimesLocation_12.csv | 169 MB |

DATA SET 2 (Crime Info)

adlademosadls ► SQLDay2019 ► Out ► Crimes

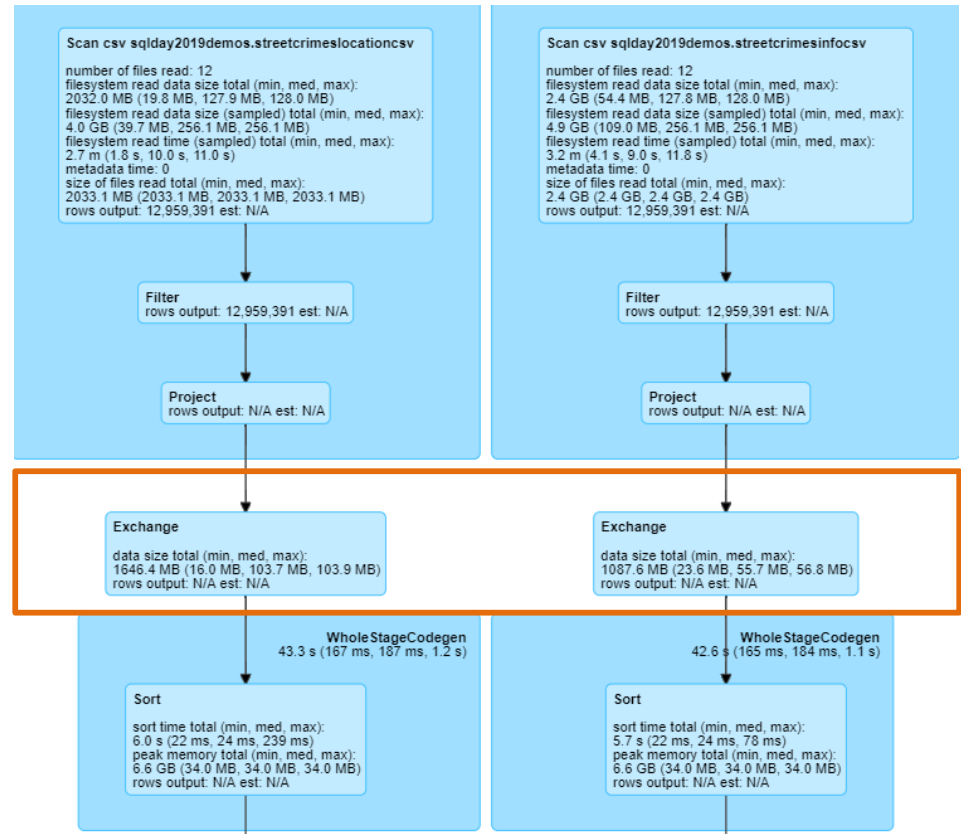
| NAME | SIZE |
|---|--------|
|  Crimes_1.csv | 207 MB |
|  Crimes_10.csv | 221 MB |
|  Crimes_11.csv | 215 MB |
|  Crimes_12.csv | 207 MB |

```
SELECT COUNT(*) AS Total
FROM StreetCrimesLocationCSV c1
JOIN StreetCrimesInfoCSV cd ON cd.CrimeID = c1.CrimeID
```

Challenge

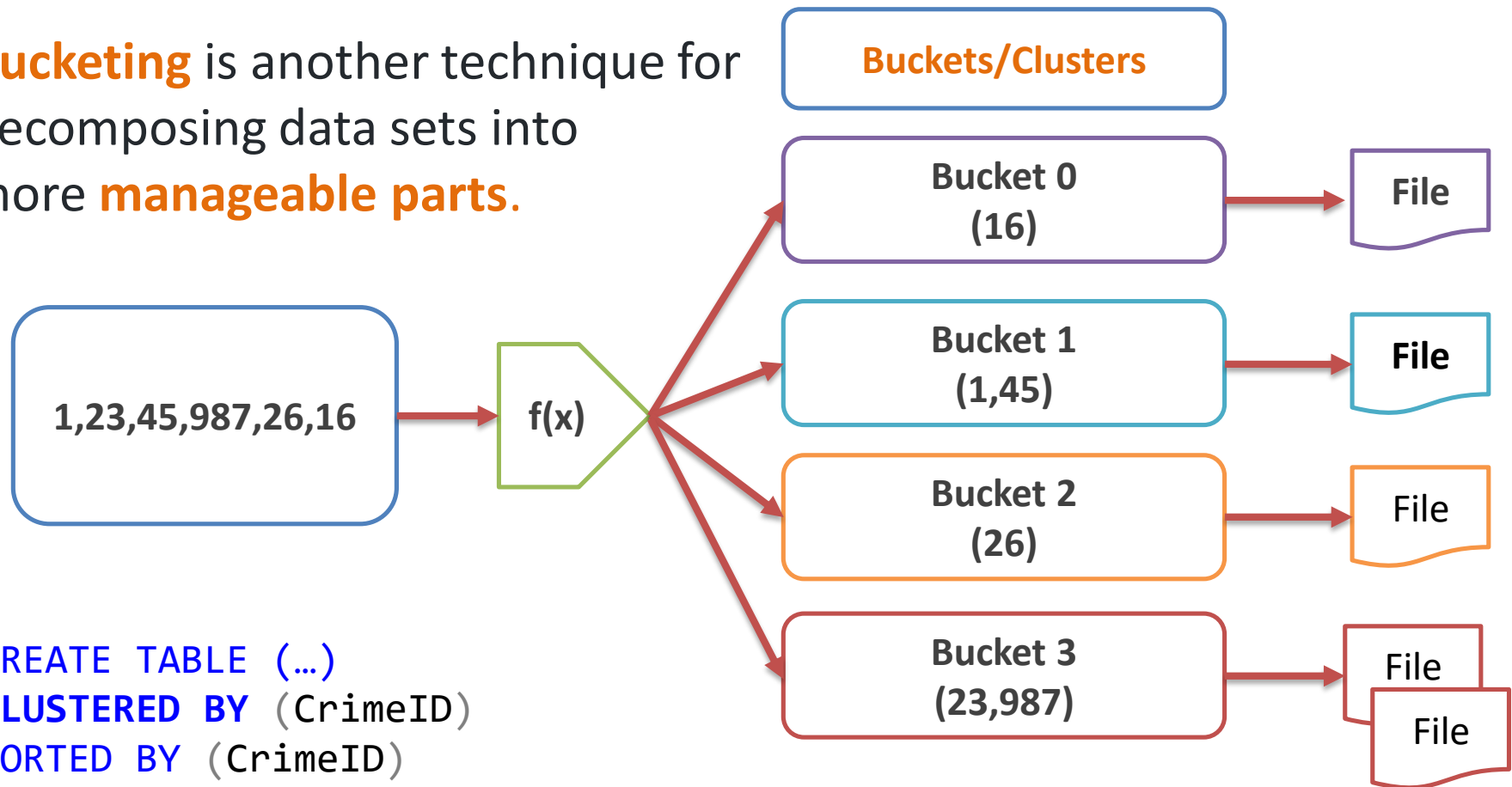
```
SELECT COUNT(*) AS Total FROM
StreetCrimesLocationCSV cl
JOIN StreetCrimesInfoCSV cd ON
cd.CrimeID = cl.CrimeID
```

Shuffle (Exchange) operation is used in **Spark** to re-distribute data across multiple partitions. It is a costly and complex operation.



Bucketing – optimization technique

Bucketing is another technique for decomposing data sets into more **manageable parts**.



```
CREATE TABLE (...)  
CLUSTERED BY (CrimeID)  
SORTED BY (CrimeID)  
INTO 4 BUCKETS;
```

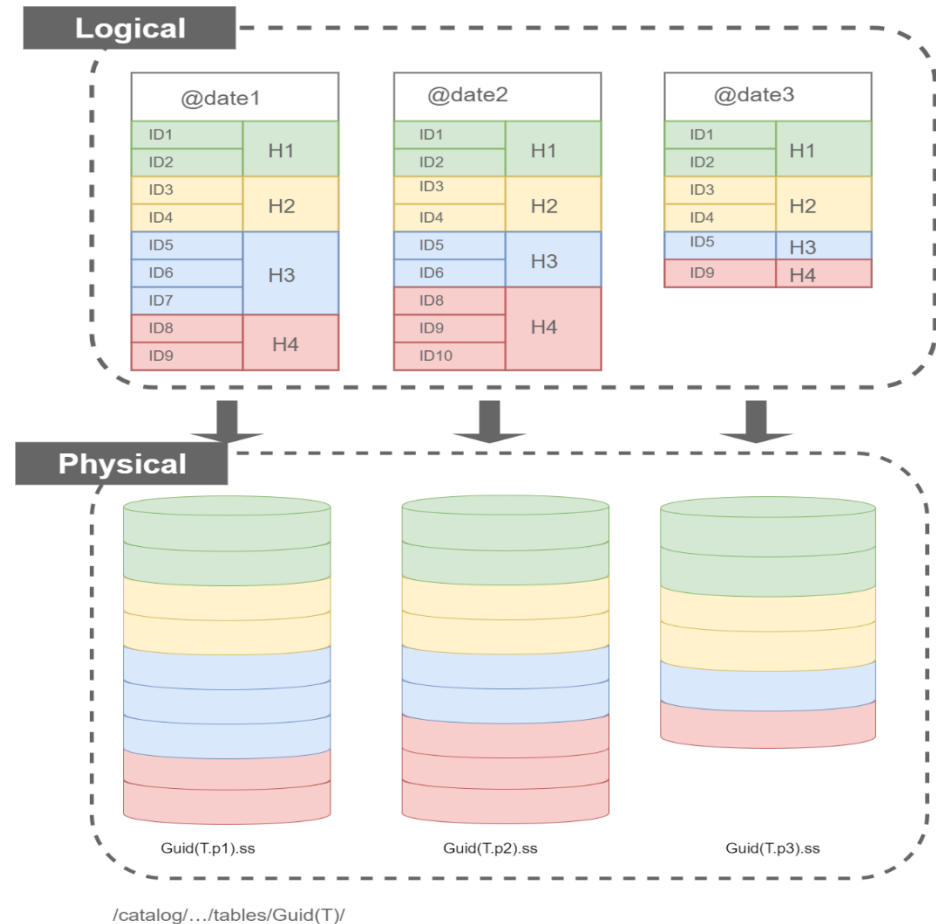
Spark

- 005-BucketingJoin
- 006-HivePartitionedAndBucketedTable
- ODBC/JDBC Connector

DEMO

ADLA Tables, Partitioning and bucketing

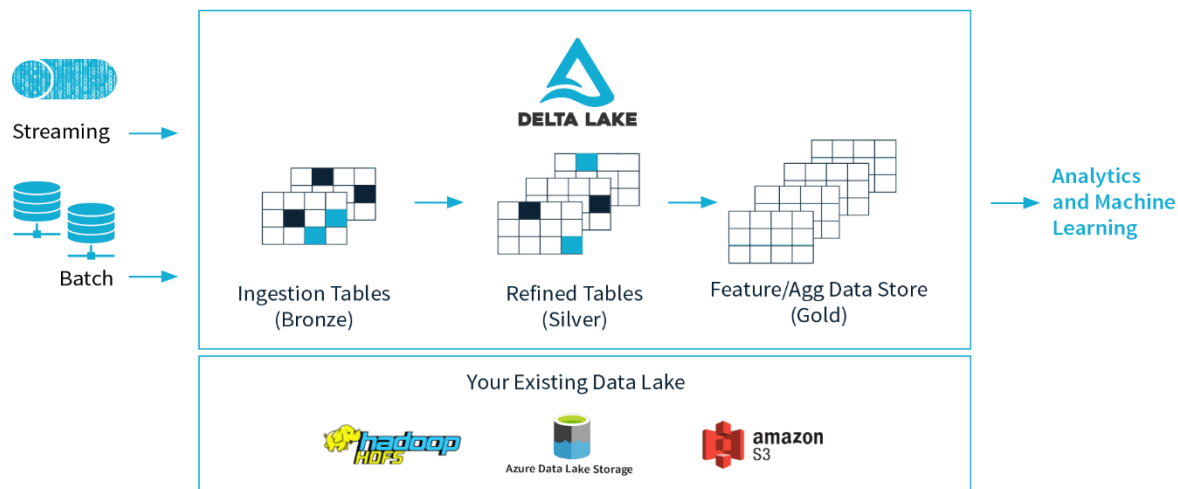
```
CREATE TABLE T  
(  
    id int,  
    date DateTime,  
    INDEX IDX  
    CLUSTERED(id)  
    PARTITIONED BY (date)  
    DISTRIBUTED BY  
    HASH(id)  
    INTO 4  
);
```



Delta Lake

Delta Lake is an open source storage layer that brings reliability to data lakes.

- Full **ACID** Transactions
- Unified Streaming and Batch
- Data versioning
- Native Support for
 - **UPDATE**
 - **DELETE**
 - **MERGE**



<https://delta.io/>

Azure Databricks Delta

```
CREATE TABLE IF NOT EXISTS  
StreetCrimesPartitionedParquet  
(  
  CrimeID STRING,  
  DateReported TIMESTAMP,  
  Month INT,  
  Year INT,  
  ...  
)  
USING Delta  
PARTITIONED BY (Year,Month)
```

Delta

- OPTIMIZE
- Z-ORDER
 - (multi-dimensional clustering)
 - (**WHERE x=1 OR y=2**)
- VACUUM

ADLA

- CreateTableDist.usql
- LoadDataIntoStreetCrimesDistByCrimeType.usql
- PartitionInfo.usql
- Demo-005.usql

Spark

- 010-Delta

DEMO

Summary

- Open and Closed Big Data Solutions
- Storage structure
 - Partitioning vs /and Bucketing (Horizontal and Vertical Partitioning)
- Delta and Delta Lake

Resources

My examples (and demos)

- <https://github.com/cloud4yourdata/usql/tree/develop>
- https://github.com/cloud4yourdata/CommunityEvents/tree/master/DataCommunity201809_InteractiveQueries
- <https://github.com/cloud4yourdata/demos/tree/develop/SQLDay2018>

External resources

- <https://docs.databricks.com/>
- <https://delta.io/>
- <https://spark.apache.org/>
- <https://jaceklaskowski.gitbooks.io/mastering-spark-sql/spark-sql-properties.html>
- <https://hive.apache.org/>
- <https://dotnet.microsoft.com/apps/data/spark>
- <http://usql.io>
- <https://databricks.com/blog/2018/07/31/processing-petabytes-of-data-in-seconds-with-databricks-delta.html>



PLATINUM SPONSOR

STRATEGIC PARTNER

TECHNOLOGY
INNOVATION
DATA
KNOWLEDGE



GOLD SPONSORS



CLOUDS ON MARS



SILVER SPONSOR



BRONZE SPONSOR

