COLUMBUS DAY 4.0

BIG DATA FOR .NET AND SQL

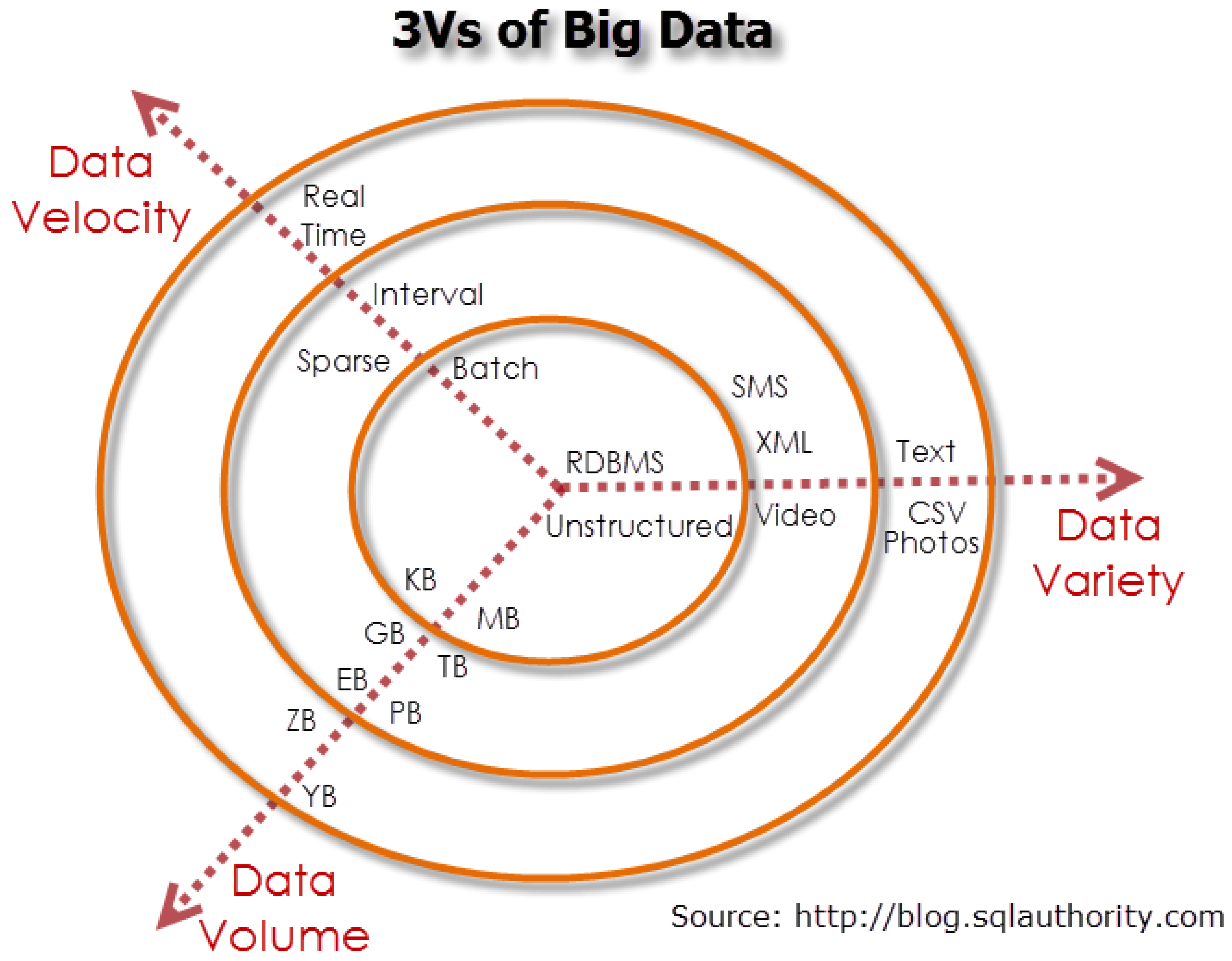
(AZURE DATA LAKE)

PLAN

- BIG DATA
- Azure Cloud
- Azure Data Lake
  - Azure Data Lake Store
  - Azure Data Lake Analytics
- U-SQL
- Azure Data Lake Live Azure Demo

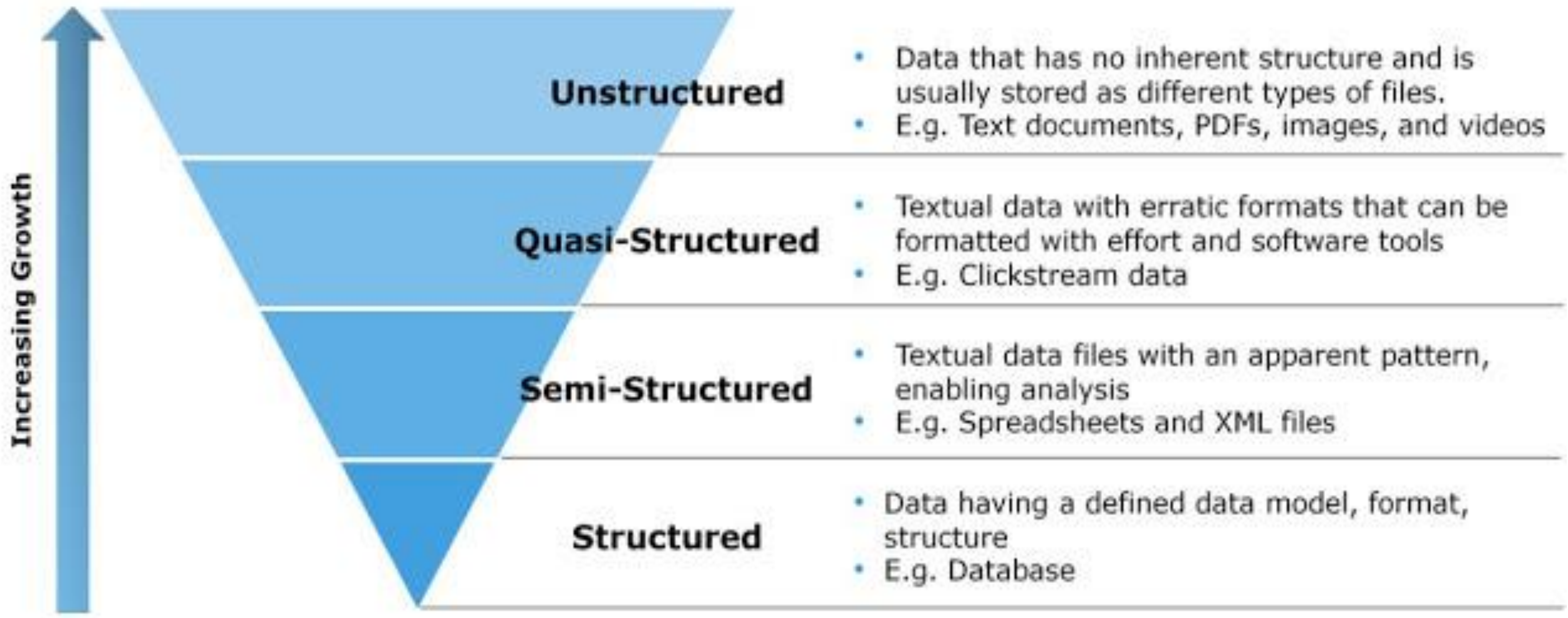
BIG DATA

- 3V
- 4V (Veracity)



Source: http://blog.sqlauthority.com

The Structure of Big Data



Unstructured

- Data that has no inherent structure and is usually stored as different types of files.
- E.g. Text documents, PDFs, images, and videos

Quasi-Structured

- Textual data with erratic formats that can be formatted with effort and software tools
- E.g. Clickstream data

Semi-Structured

- Textual data files with an apparent pattern, enabling analysis
- E.g. Spreadsheets and XML files

Structured

- Data having a defined data model, format, structure
- E.g. Database

Source: http://www.tsmtutorials.com/2016/06/data-and-information-basics.html

Schema-on-Read vs Schema-on-Write

SCHEMA-ON-WRITE (RDBMS):

- Create static DB schema
- Transform data into RDBMS
- Query data in RDBMS format

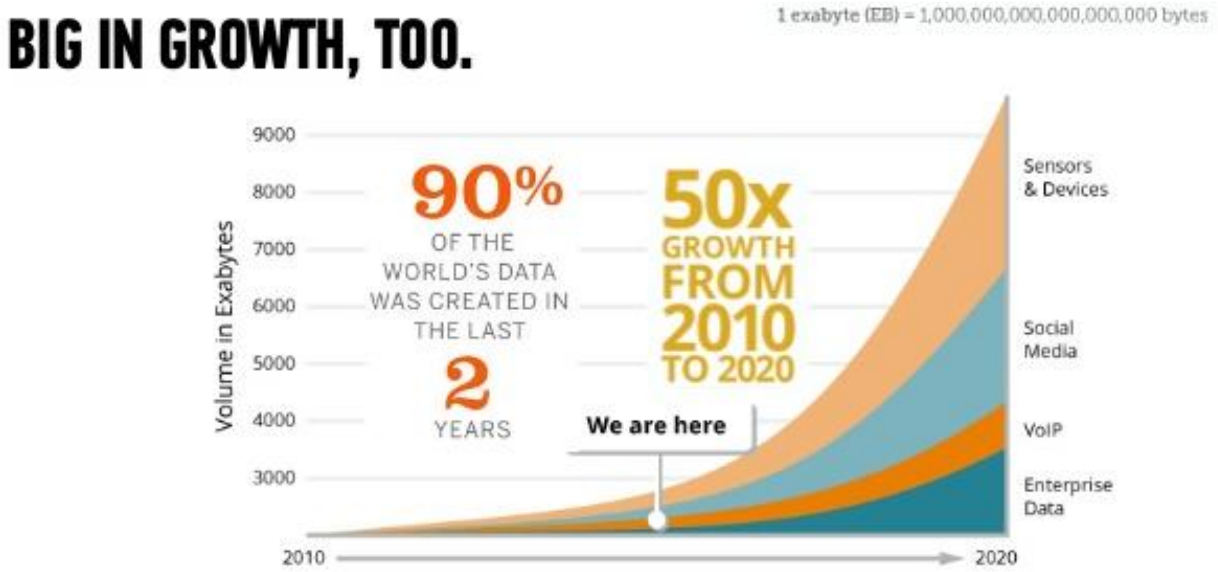
New columns must be added explicitly before new data can propagate into the system.

SCHEMA-ON-READ (HADOOP OR ADLS):

- Copy data in its native format
- Create schema + parser
- Query Data in its native format
  (does ETL on the fly)

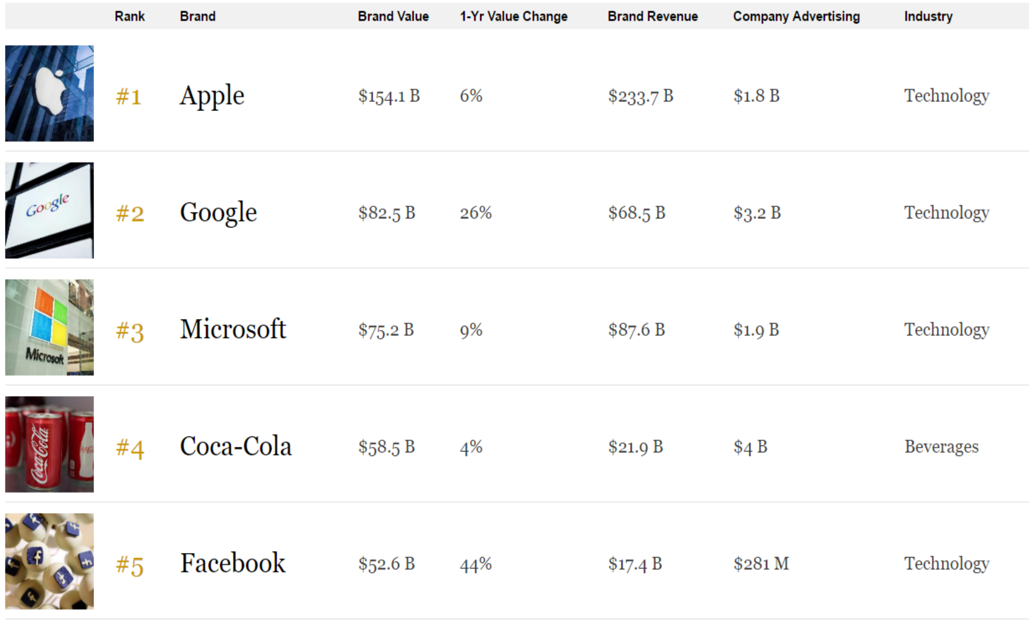
New data can start flowing any time and will appear retroactively once the schema/parser properly describes it.

Why Big Data?


BIG IN GROWTH, TOO.
1 exabyte (EB) = 1,000,000,000,000,000,000 bytes
Volume in Exabytes
9000
8000
7000
6000
5000
4000
3000
90%
OF THE
WORLD'S DATA
WAS CREATED IN
THE LAST
2
YEARS
50x
GROWTH
FROM
2010
TO 2020
We are here
Sensors
& Devices
Social
Media
VoIP
Enterprise
Data
2010
2020

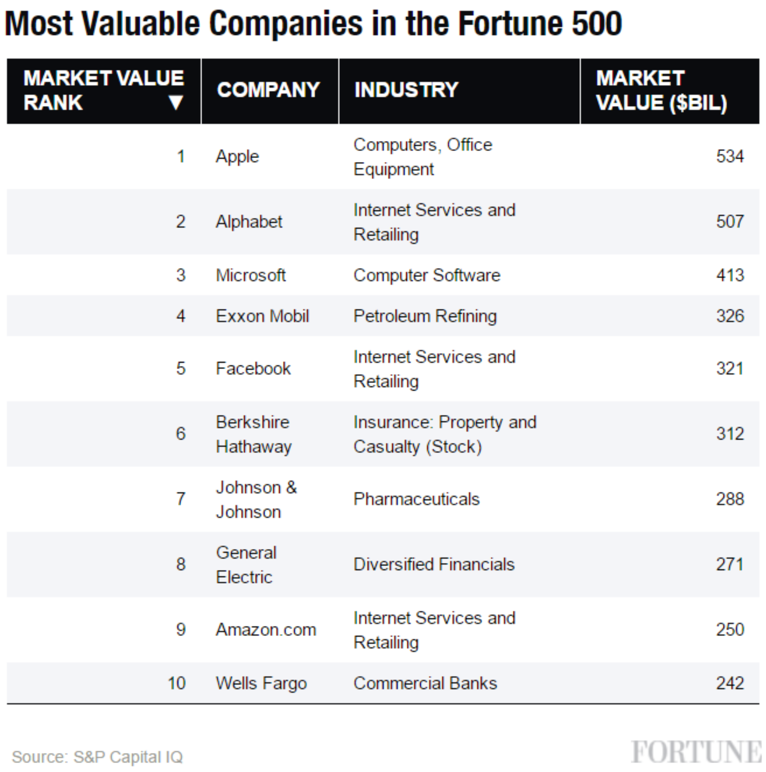
[illegible] [illegible]

Why Big Data?



	Rank	Brand	Brand Value	1-Yr Value Change	Brand Revenue	Company Advertising	Industry
	#1	Apple	$154.1 B	6%	$233.7 B	$1.8 B	Technology
	#2	Google	$82.5 B	26%	$68.5 B	$3.2 B	Technology
	#3	Microsoft	$75.2 B	9%	$87.6 B	$1.9 B	Technology
	#4	Coca-Cola	$58.5 B	4%	$21.9 B	$4 B	Beverages
	#5	Facebook	$52.6 B	44%	$17.4 B	$281 M	Technology

Source: http://www.forbes.com/powerful-brands/list/



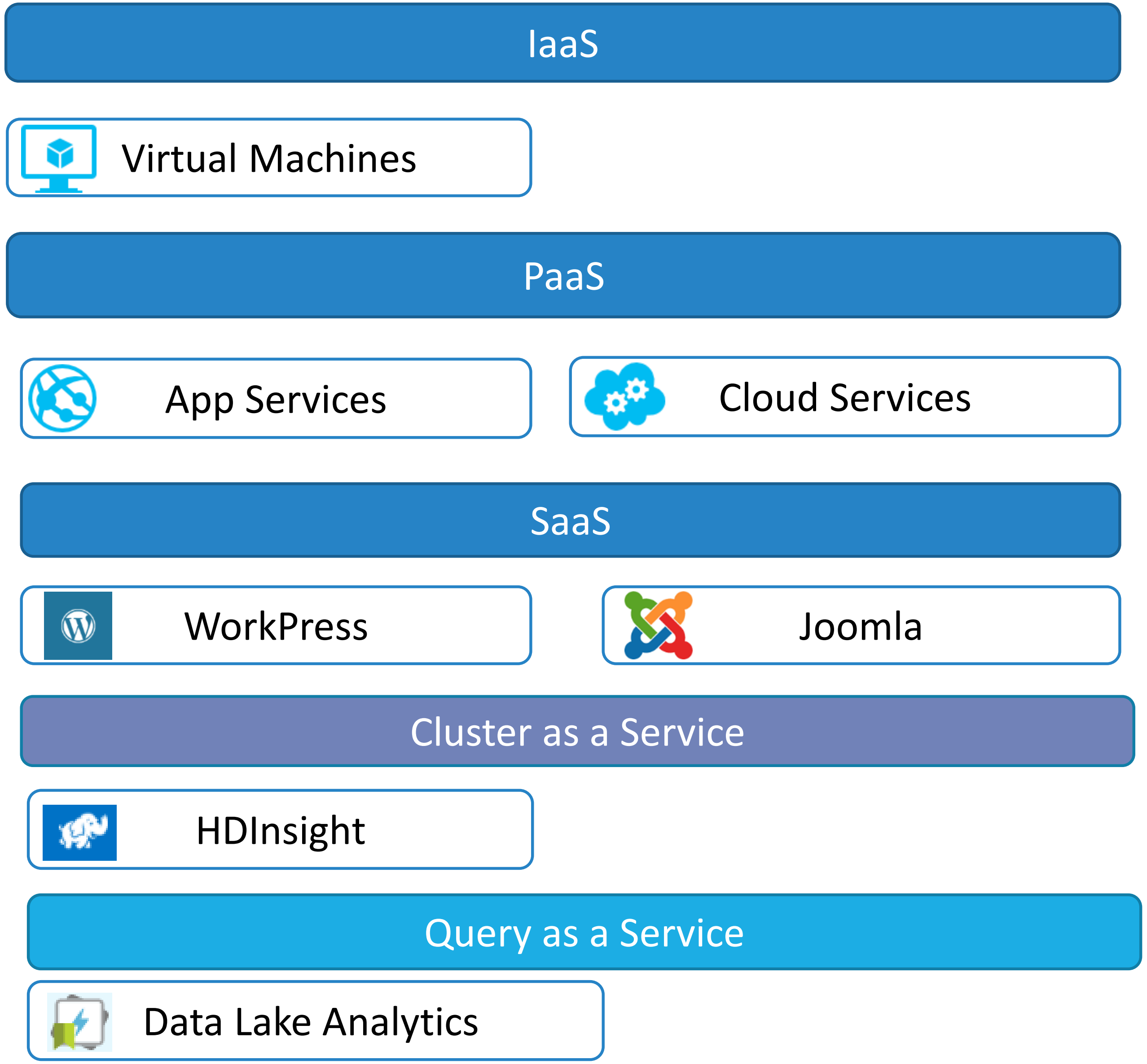
Most Valuable Companies in the Fortune 500

MARKET VALUE RANK ▼	COMPANY	INDUSTRY	MARKET VALUE ($BIL)
1	Apple	Computers, Office Equipment	534
2	Alphabet	Internet Services and Retailing	507
3	Microsoft	Computer Software	413
4	Exxon Mobil	Petroleum Refining	326
5	Facebook	Internet Services and Retailing	321
6	Berkshire Hathaway	Insurance: Property and Casualty (Stock)	312
7	Johnson & Johnson	Pharmaceuticals	288
8	General Electric	Diversified Financials	271
9	Amazon.com	Internet Services and Retailing	250
10	Wells Fargo	Commercial Banks	242

Source: S&P Capital IQ

FORTUNE

Cloud Service Models and Azure



IaaS

Virtual Machines

PaaS

App Services

Cloud Services

SaaS

WorkPress

Joomla

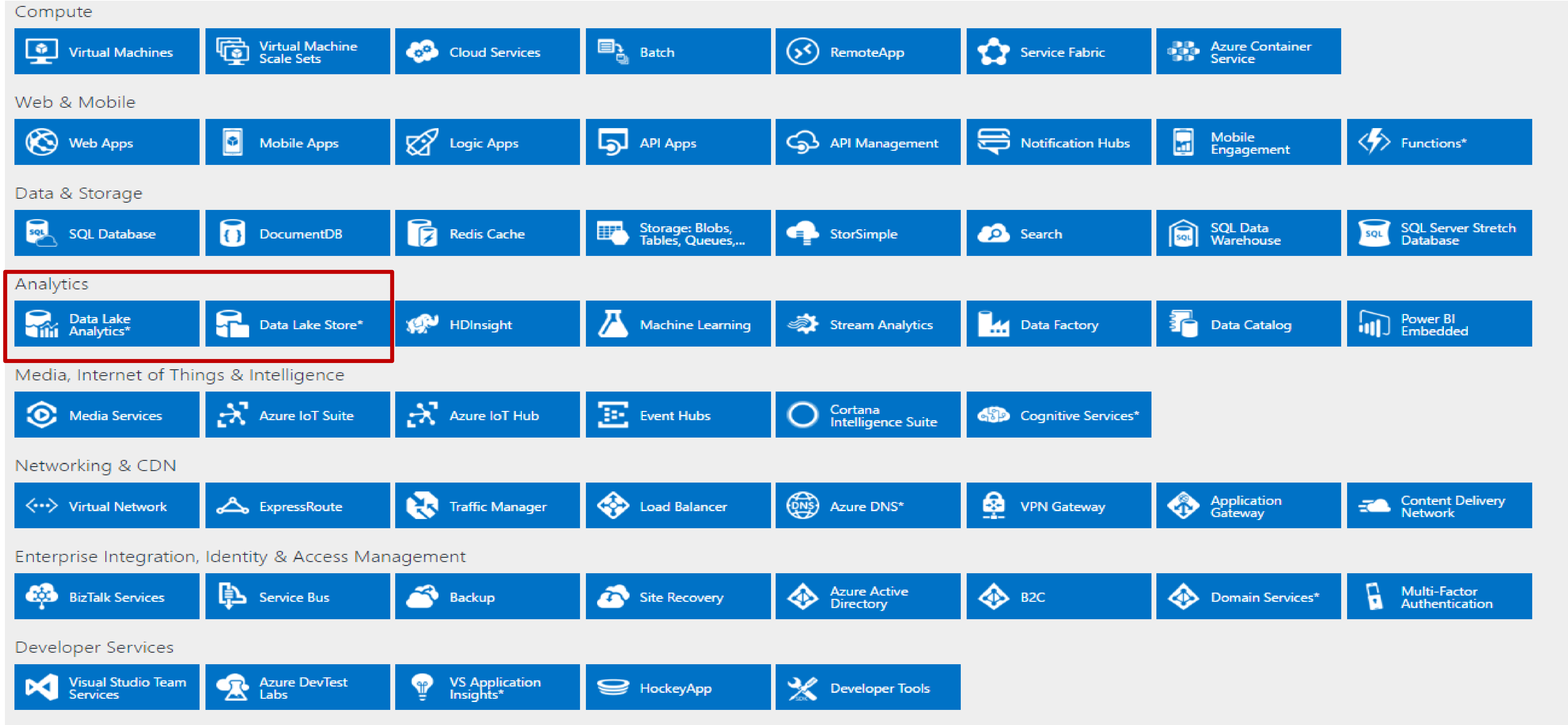
Cluster as a Service

HDInsight

Query as a Service

Data Lake Analytics

Azure Services



Compute

- Virtual Machines
- Virtual Machine Scale Sets
- Cloud Services
- Batch
- RemoteApp
- Service Fabric
- Azure Container Service

Web & Mobile

- Web Apps
- Mobile Apps
- Logic Apps
- API Apps
- API Management
- Notification Hubs
- Mobile Engagement
- Functions*

Data & Storage

- SQL Database
- DocumentDB
- Redis Cache
- Storage: Blobs, Tables, Queues,...
- StorSimple
- Search
- SQL Data Warehouse
- SQL Server Stretch Database

Analytics

- Data Lake Analytics*
- Data Lake Store*
- HDInsight
- Machine Learning
- Stream Analytics
- Data Factory
- Data Catalog
- Power BI Embedded

Media, Internet of Things & Intelligence

- Media Services
- Azure IoT Suite
- Azure IoT Hub
- Event Hubs
- Cortana Intelligence Suite
- Cognitive Services*

Networking & CDN

- Virtual Network
- ExpressRoute
- Traffic Manager
- Load Balancer
- Azure DNS*
- VPN Gateway
- Application Gateway
- Content Delivery Network

Enterprise Integration, Identity & Access Management

- BizTalk Services
- Service Bus
- Backup
- Site Recovery
- Azure Active Directory
- B2C
- Domain Services*
- Multi-Factor Authentication

Developer Services

- Visual Studio Team Services
- Azure DevTest Labs
- VS Application Insights*
- HockeyApp
- Developer Tools

http://azureplatform.azurewebsites.net/en-us/

Azure Data Lake

Mike Rys

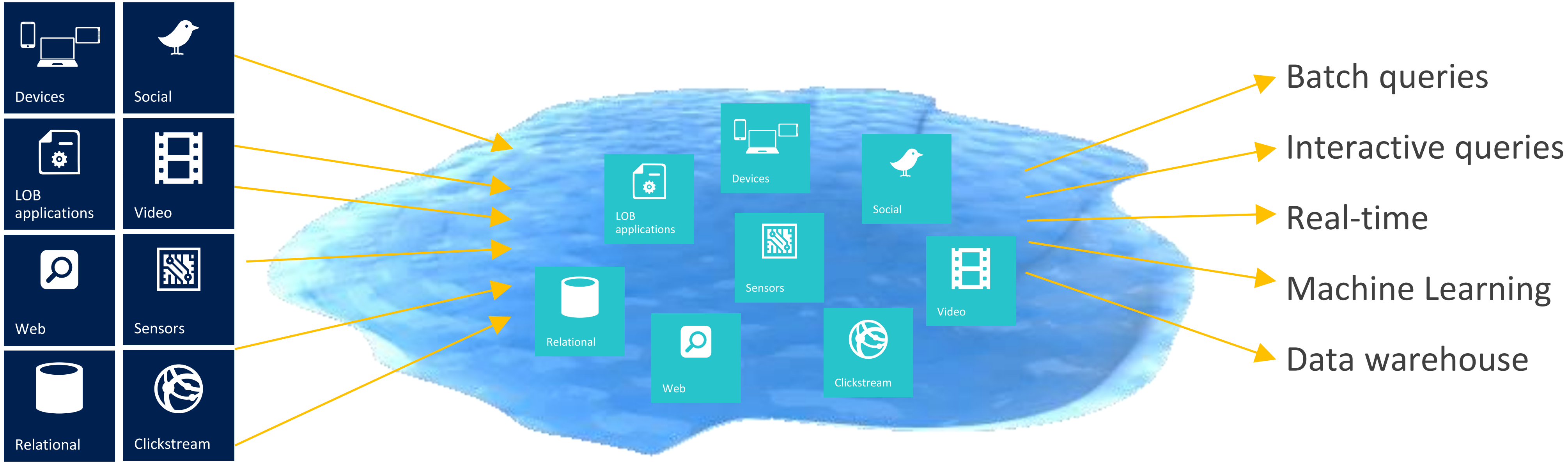
Saveen Reddy

Data Lake Appoach

Ingest all data
regardless of requirements

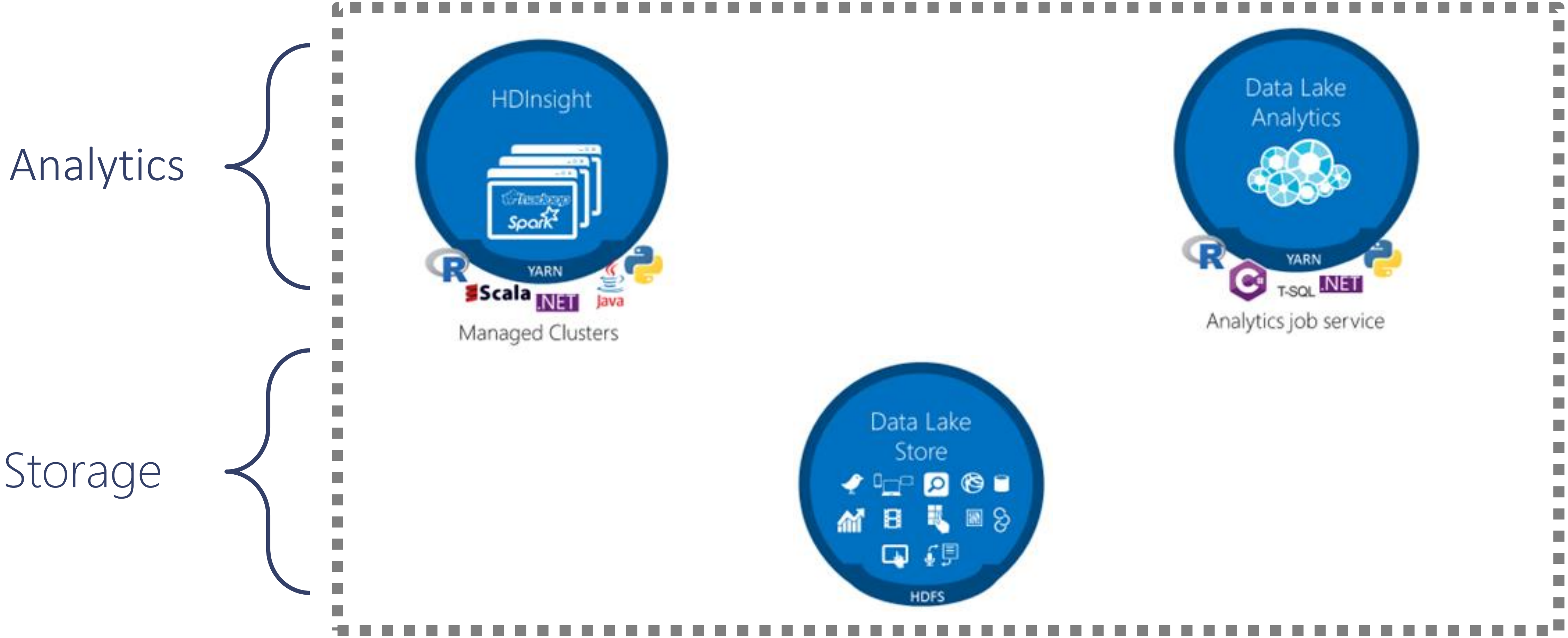
Store all data
in native format without schema definition

Do analysis
Using analytic engines like Hadoop

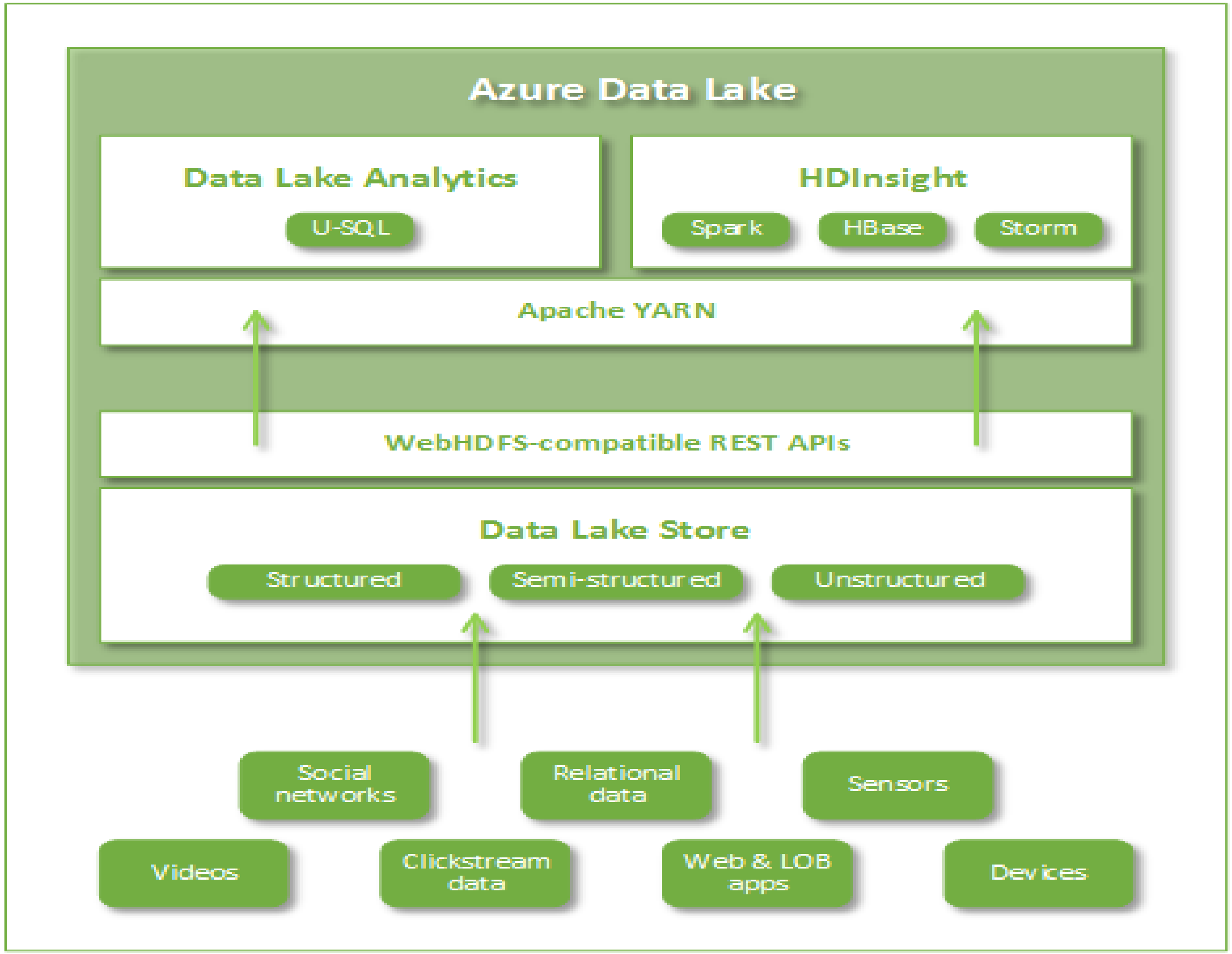


From: M. Rys Presentation

Azure Data Lake



Azure Data Lake



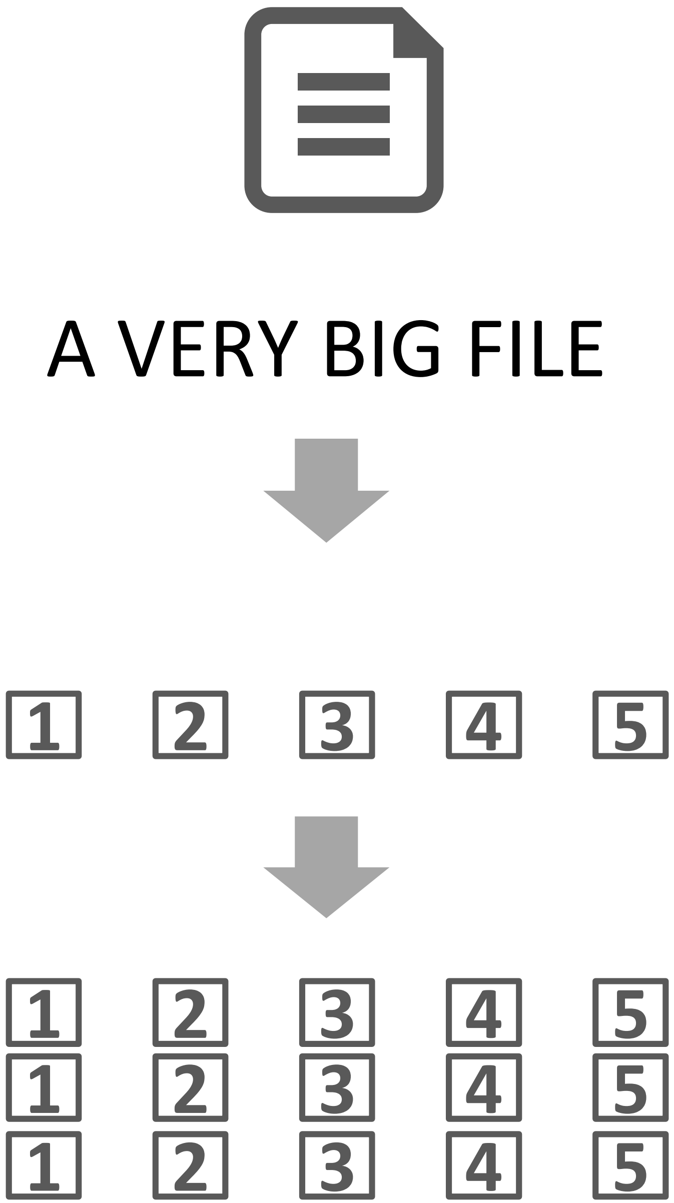
Microsoft

Overview of Azure Data Lake Store

- Built for Hadoop
  - WebHDFS-compatible REST interface
- Unlimited storage, petabyte files
- Performance-tuned for big data analytics
- Highly-available and secure
- Integrates with HDInsight, Cloudera, Hortonworks
- Supports files and folders objects

ADL Store Basics

- Files are split apart into Extents (250 MB)
- For availability and reliability, extents are replicated (3 copies).
- Enables:
  - Parallel read
  - Parallel write

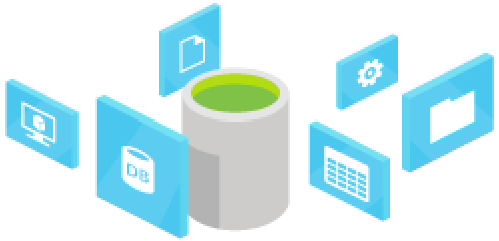


A VERY BIG FILE

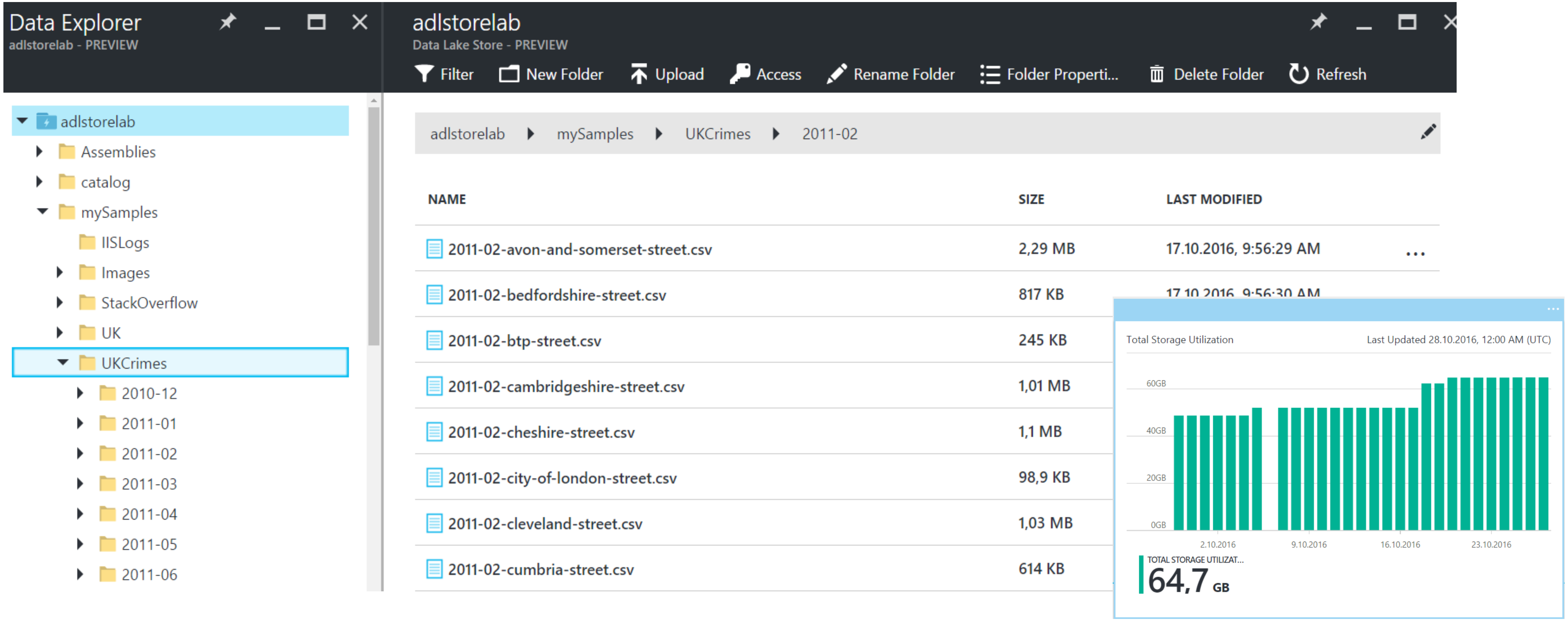
From: S. Reddy Presentation

Working with Azure Data Lake Store

- Local computer
  - Azure Portal
  - Azure PowerShell
  - Azure Cross-platform CLI
  - Using Data Lake Tools for Visual Studio
- Azure Storage Blob
  - Azure Data Factory
  - AdlCopy tool
  - DistCp running on HDInsight cluster
- Streamed data
  - Azure Stream Analytics
  - Azure HDInsight Storm
- Relational data
  - Apache Sqoop
  - Azure Data Factory



Azure Data Explorer



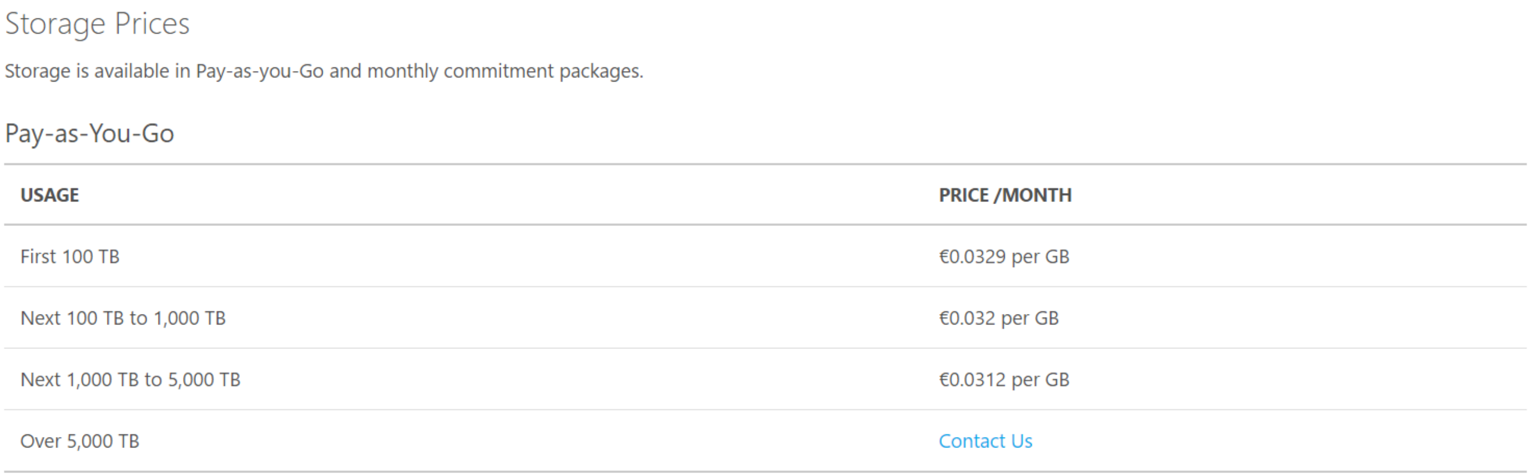
- adlstorelab
  - Assemblies
  - catalog
  - mySamples
    - IISLogs
    - Images
    - StackOverflow
    - UK
    - UKCrimes
      - 2010-12
      - 2011-01
      - 2011-02
      - 2011-03
      - 2011-04
      - 2011-05
      - 2011-06

adlstorelab ▸ mySamples ▸ UKCrimes ▸ 2011-02

NAME	SIZE	LAST MODIFIED
2011-02-avon-and-somerset-street.csv	2,29 MB	17.10.2016, 9:56:29 AM
2011-02-bedfordshire-street.csv	817 KB	17.10.2016, 9:56:30 AM
2011-02-btp-street.csv	245 KB	
2011-02-cambridgeshire-street.csv	1,01 MB	
2011-02-cheshire-street.csv	1,1 MB	
2011-02-city-of-london-street.csv	98,9 KB	
2011-02-cleveland-street.csv	1,03 MB	
2011-02-cumbria-street.csv	614 KB	

Total Storage Utilization
Last Updated 28.10.2016, 12:00 AM (UTC)
60GB
40GB
20GB
0GB
2.10.2016
9.10.2016
16.10.2016
23.10.2016
TOTAL STORAGE UTILIZAT...
64,7 GB

Azure Data Lake Store Pricing

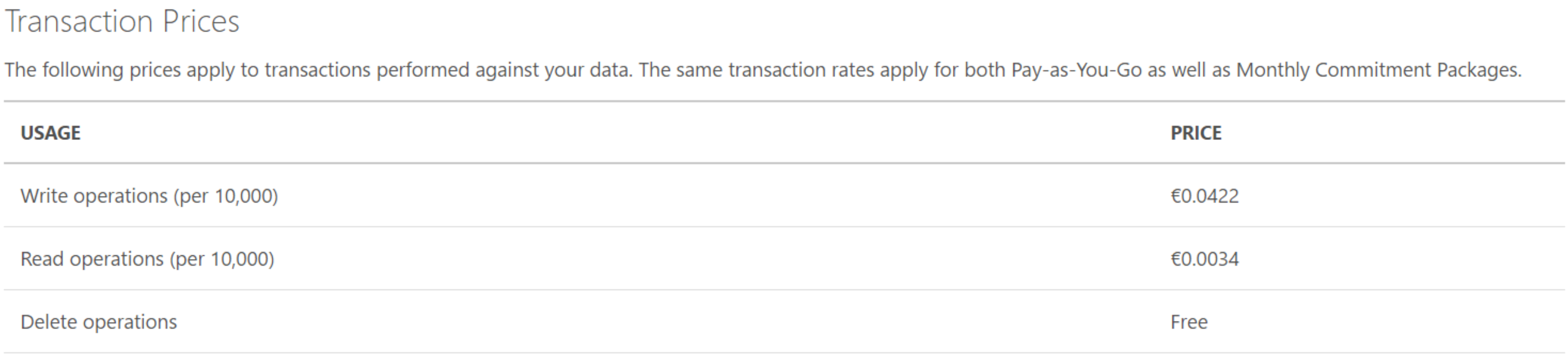


Storage Prices

Storage is available in Pay-as-you-Go and monthly commitment packages.

Pay-as-You-Go

USAGE	PRICE /MONTH
First 100 TB	€0.0329 per GB
Next 100 TB to 1,000 TB	€0.032 per GB
Next 1,000 TB to 5,000 TB	€0.0312 per GB
Over 5,000 TB	Contact Us



Transaction Prices

The following prices apply to transactions performed against your data. The same transaction rates apply for both Pay-as-You-Go as well as Monthly Commitment Packages.

USAGE	PRICE
Write operations (per 10,000)	€0.0422
Read operations (per 10,000)	€0.0034
Delete operations	Free

Azure Data Lake Analytics

A distributed analytics service built on Apache YARN that dynamically scales to your needs

Pay **PER QUERY** & Scale **PER QUERY**

FEDERATED QUERY across Azure data sources

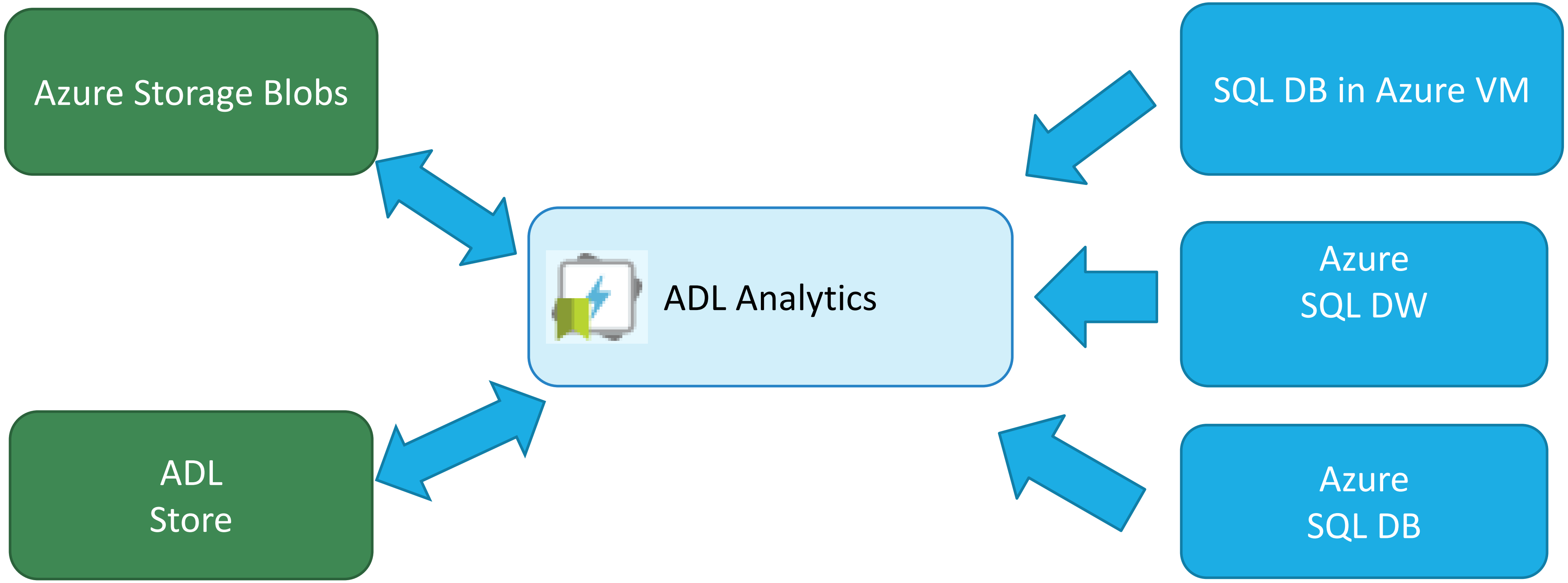
Includes **U-SQL**, a language that unifies the benefits of SQL with the expressive power of C#

No limits to **SCALE**

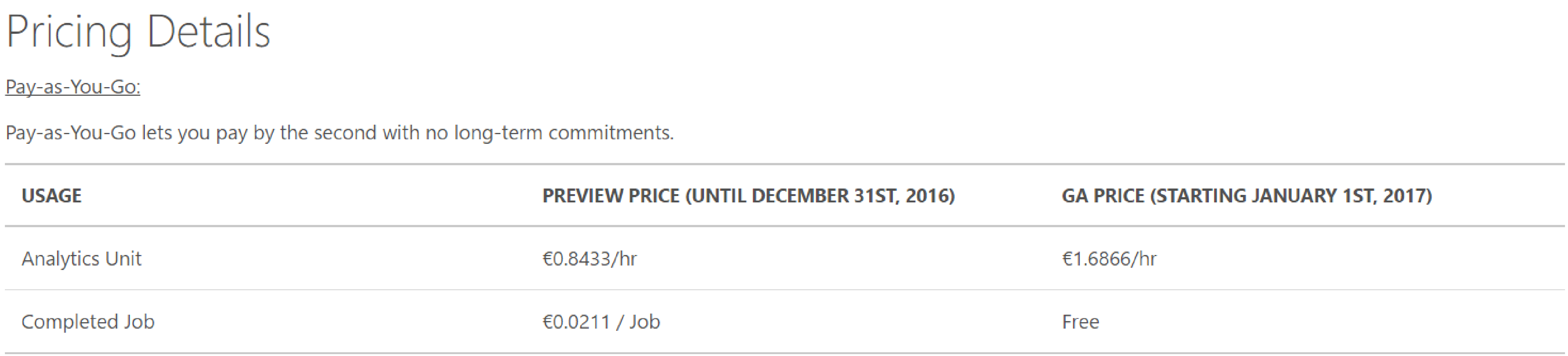
Optimized to work with **ADL STORE**

From: S. Reddy Presentation

Work across all your cloud Data



Data Lake Analytics Pricing



Pricing Details

Pay-as-You-Go:

Pay-as-You-Go lets you pay by the second with no long-term commitments.

USAGE	PREVIEW PRICE (UNTIL DECEMBER 31ST, 2016)	GA PRICE (STARTING JANUARY 1ST, 2017)
Analytics Unit	€0.8433/hr	€1.6866/hr
Completed Job	€0.0211 / Job	Free

$$JobCost = (seconds \times ADLU)/3600 + \textit{Completed Job Cost} + \textit{Data Lake Transactions Cost}$$

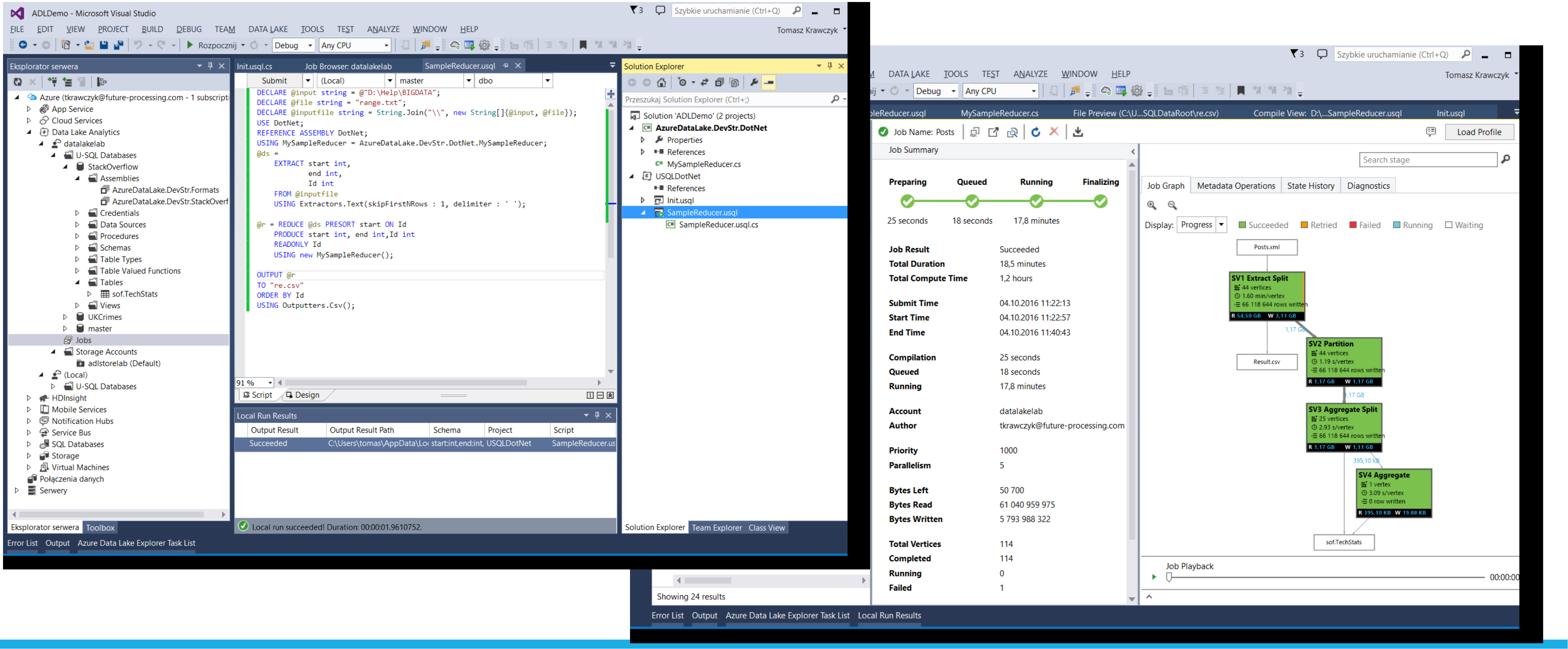
Azure Data Lake Analytics Unit

Parallelism N = N ADLAUs

1 ADLAU ~=

- A VM with 2 cores and 6 GB of memory

Data Lake Tools for Visual Studio



```
DECLARE @input string = @"D:\Help\BIGDATA";
DECLARE @file string = "range.txt";
DECLARE @inputfile string = String.Join("\\", new String[]{@input, @file});
USE DotNet;
REFERENCE ASSEMBLY DotNet;
USING MySampleReducer = AzureDataLake.DevStr.DotNet.MySampleReducer;
@ds =
    EXTRACT start int,
            end int,
            Id int
    FROM @inputfile
    USING Extractors.Text(skipFirstNRows : 1, delimiter : ' ');

@r = REDUCE @ds PRESORT start ON Id
    PRODUCE start int, end int,Id int
    READONLY Id
    USING new MySampleReducer();

OUTPUT @r
TO "re.csv"
ORDER BY Id
USING Outputters.Csv();
```

Local Run Results

Output Result	Output Result Path	Schema	Project	Script
Succeeded	C:\Users\tomas\AppData\Lo	start:int,end:int,	USQLDotNet	SampleReducer.us

Local run succeeded! Duration: 00:00:01.9610752.

Job Name: Posts

Job Summary

Preparing	Queued	Running	Finalizing
25 seconds	18 seconds	17,8 minutes	

Job Result	Succeeded
Total Duration	18,5 minutes
Total Compute Time	1,2 hours
Submit Time	04.10.2016 11:22:13
Start Time	04.10.2016 11:22:57
End Time	04.10.2016 11:40:43
Compilation	25 seconds
Queued	18 seconds
Running	17,8 minutes
Account	datalakelab
Author	tkrawczyk@future-processing.com
Priority	1000
Parallelism	5
Bytes Left	50 700
Bytes Read	61 040 959 975
Bytes Written	5 793 988 322
Total Vertices	114
Completed	114
Running	0
Failed	1

DINNER

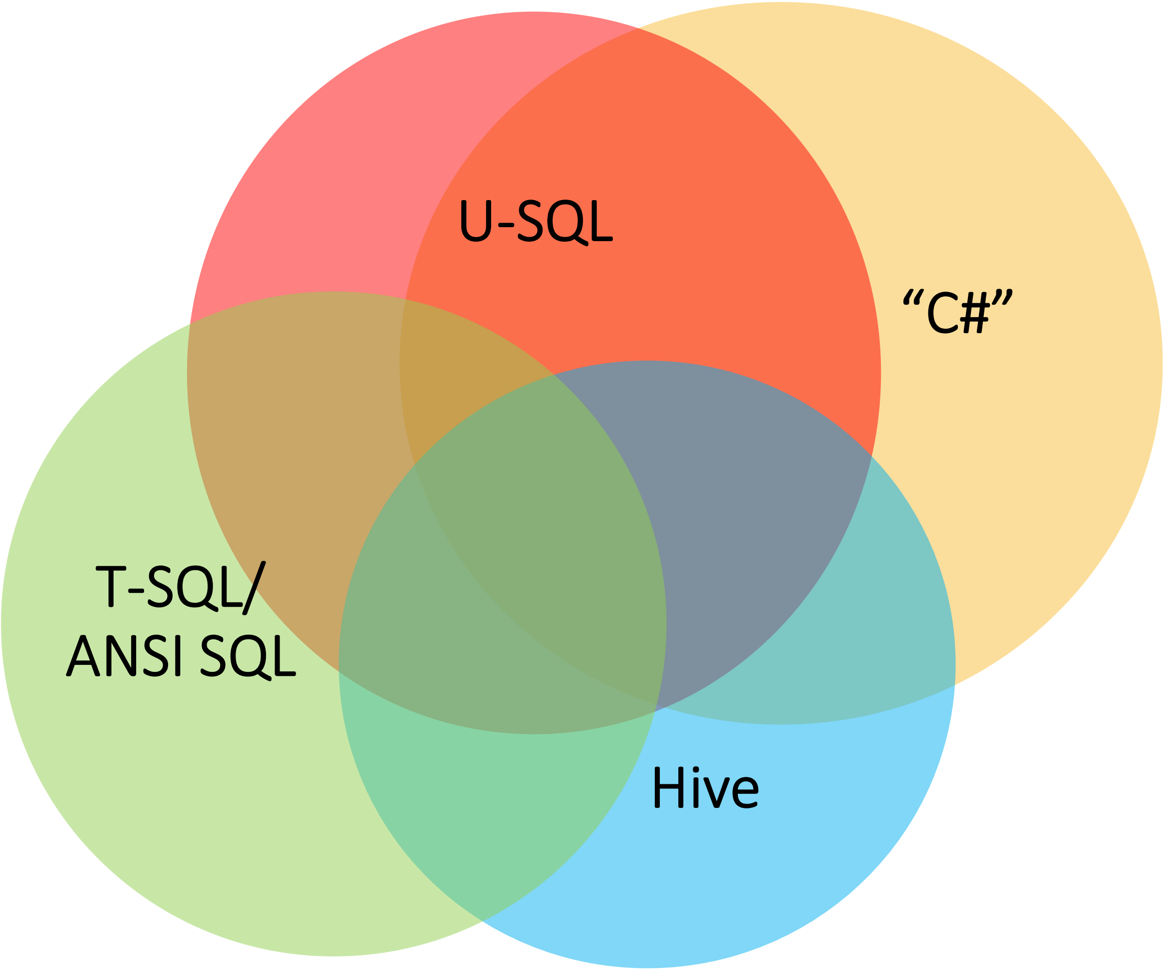
U –SQL
A new language for Big Data

Familiar syntax to millions of SQL & .NET developers

Unifies declarative nature of SQL with the imperative power of C#

Unifies structured, semi-structured and unstructured data

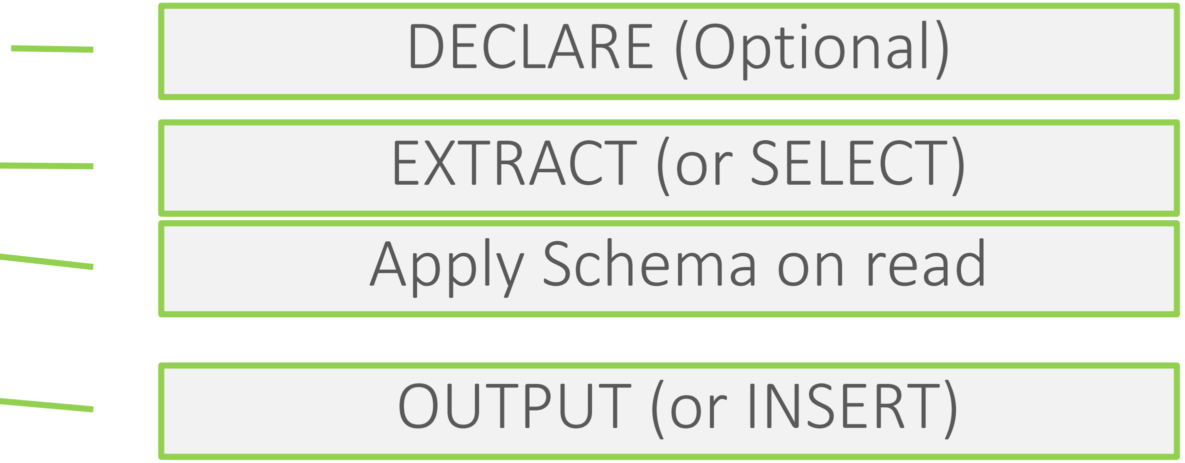
Distributed query support over all data



From: M. Rys Presentation

U-SQL SCRIPT

```
DECLARE @inputPostCodes string =
@"mySamples/UK/ukpostcodes.csv";

@postCodes =
    EXTRACT id string,
            postcode string,
            latitude string,
            longitude string
    FROM @inputPostCodes
    USING Extractors.Csv(skipFirstNRows : 1);

OUTPUT @postCodes
TO @"ukpostcodes.txt"
USING Outputters.Text();
```



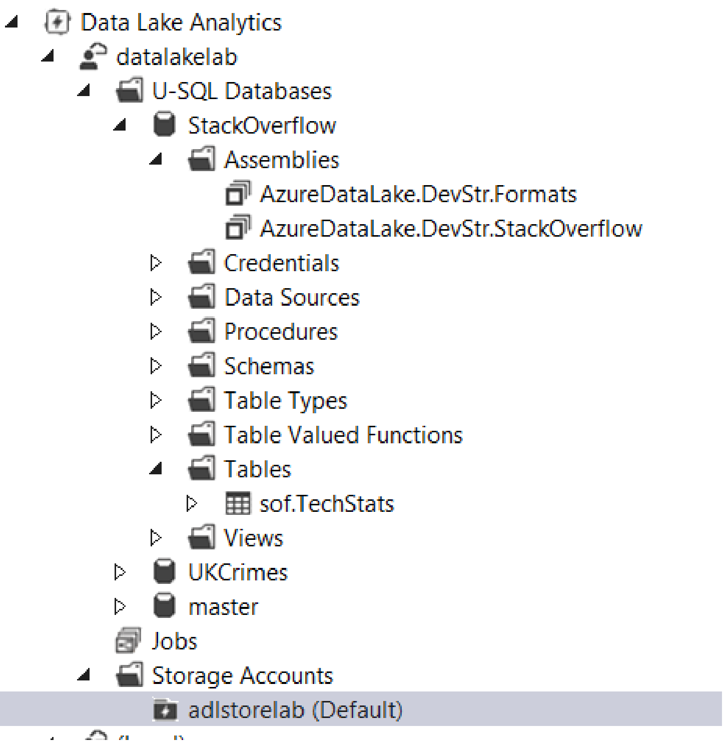
DECLARE (Optional)

EXTRACT (or SELECT)

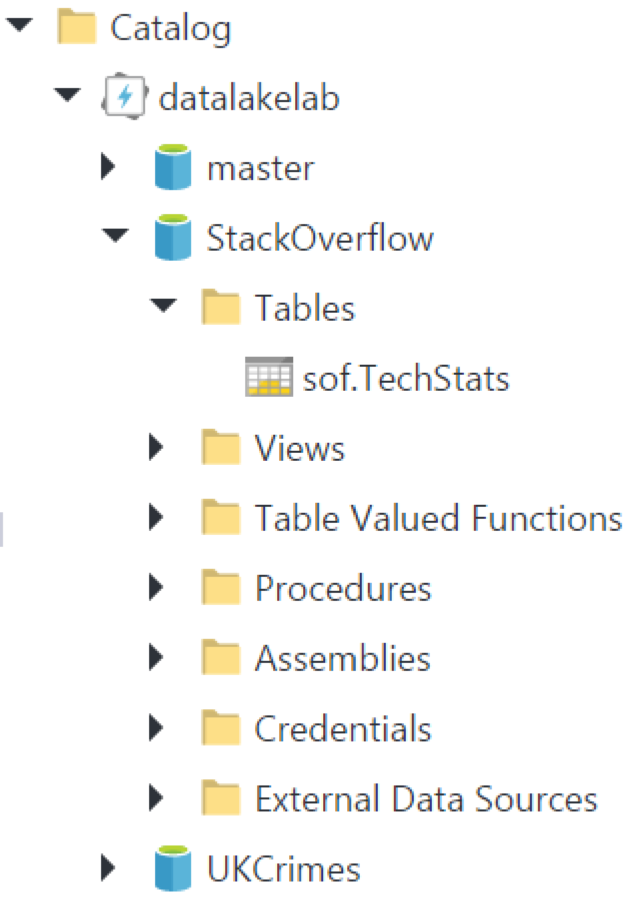
Apply Schema on read

OUTPUT (or INSERT)

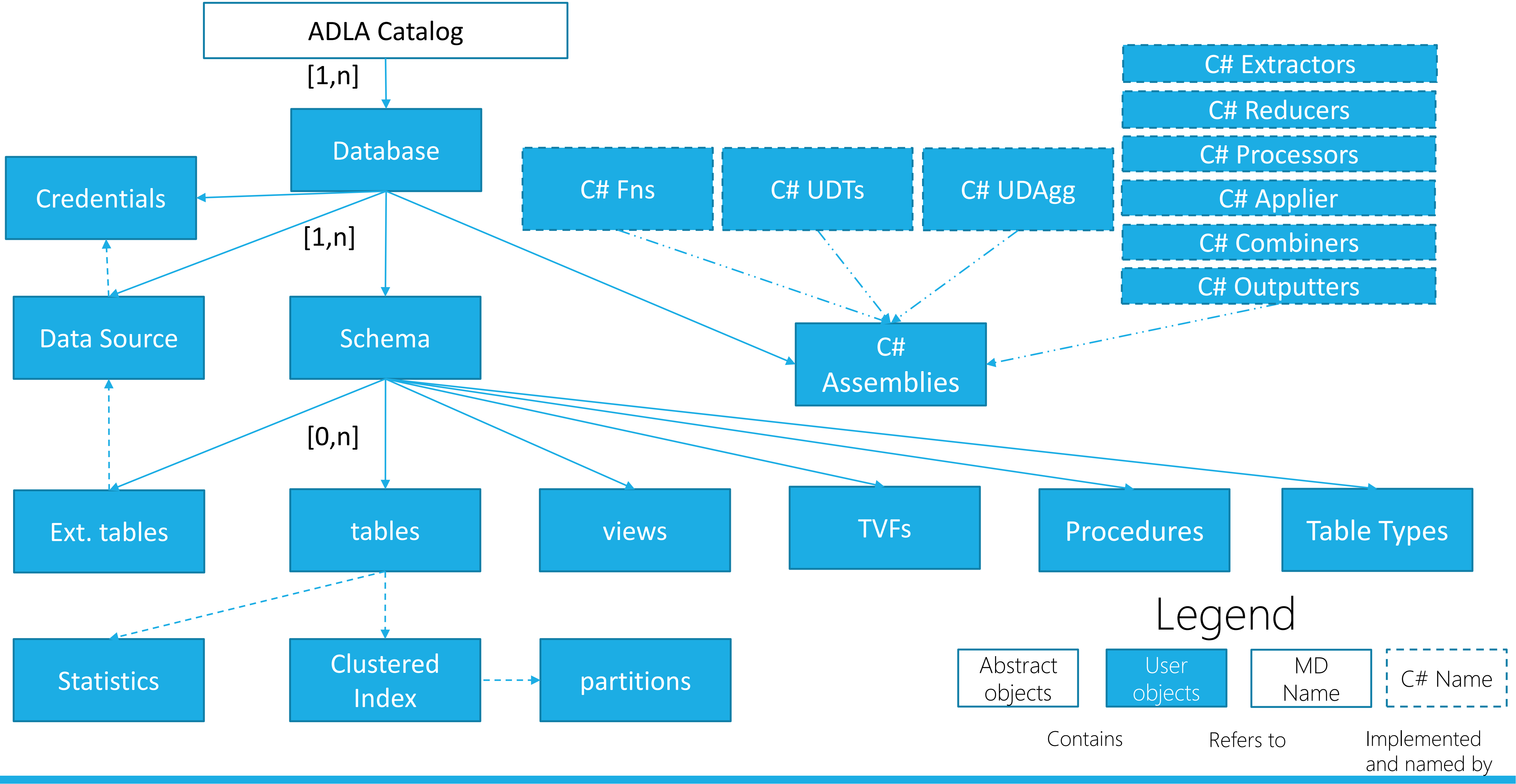
U-SQL Meta Data Catalog



- Data Lake Analytics
  - datalakelab
    - U-SQL Databases
      - StackOverflow
        - Assemblies
          - AzureDataLake.DevStr.Formats
          - AzureDataLake.DevStr.StackOverflow
        - Credentials
        - Data Sources
        - Procedures
        - Schemas
        - Table Types
        - Table Valued Functions
        - Tables
          - sof.TechStats
        - Views
      - UKCrimes
      - master
    - Jobs
    - Storage Accounts
      - adlstorelab (Default)



- Catalog
  - datalakelab
    - master
    - StackOverflow
      - Tables
        - sof.TechStats
      - Views
      - Table Valued Functions
      - Procedures
      - Assemblies
      - Credentials
      - External Data Sources
    - UKCrimes



ADLA Catalog

[1,n]

Database

Credentials

[1,n]

Data Source

Schema

C# Fns

C# UDTs

C# UDAgg

C# Extractors

C# Reducers

C# Processors

C# Applier

C# Combiners

C# Outputters

C# Assemblies

[0,n]

Ext. tables

tables

views

TVFs

Procedures

Table Types

Statistics

Clustered Index

partitions

Legend

Abstract objects

User objects

MD Name

C# Name

Contains

Refers to

Implemented and named by

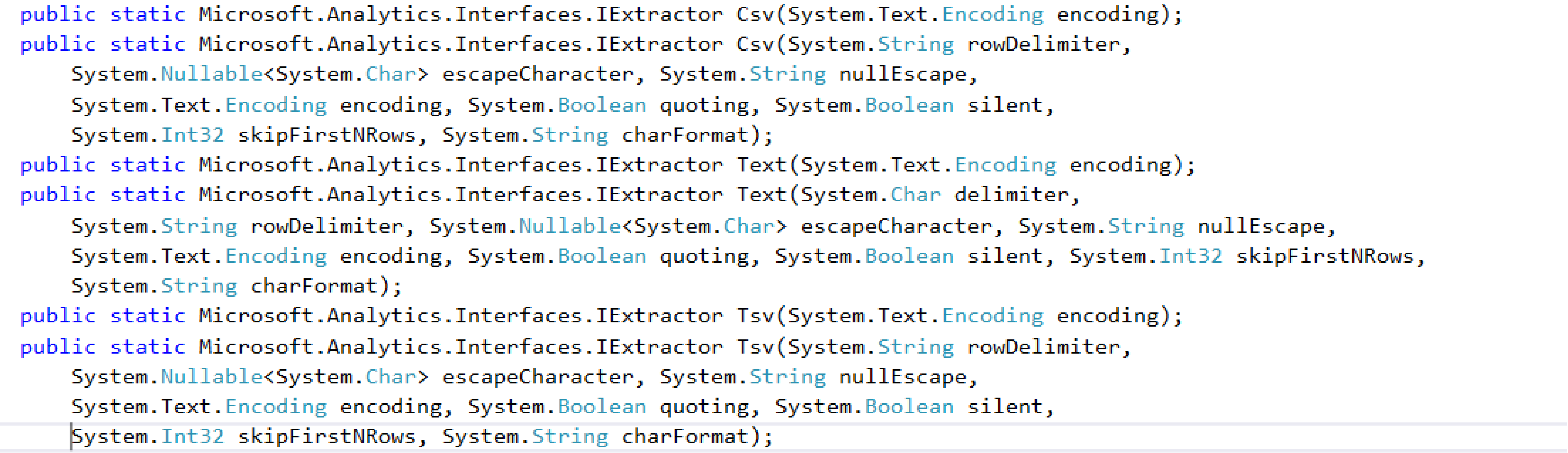
From: M. Rys Presentation

U-SQL DECLARE VARIABLES

```
DECLARE @text1 string = "Columbus Day";
DECLARE @text2 string = @"Columbus Day";
DECLARE @text3 char = 'a';
DECLARE @text4 string = "BEGIN" + @text1 + "END";
DECLARE @text5 string = string.Format("BEGIN{0}END", @text1);
DECLARE @text6 string = string.Join(" ", new String[]{@text1, "4.0"});

DECLARE @numeric1 sbyte = 0;
DECLARE @numeric2 short = 1;
DECLARE @numeric3 int = 2;
DECLARE @numeric4 long = 3L;
DECLARE @numeric5 float = 4.0f;
DECLARE @numeric6 double = 5.0;

DECLARE @d1 DateTime = System.DateTime.Parse("1979/03/31");
DECLARE @d2 DateTime = DateTime.Now;

DECLARE @misc1 bool = true;
DECLARE @misc2 Guid = System.Guid.Parse("BEF7A4E8-F583-4804-9711-7E608215EBA6");
DECLARE @misc4 byte [] = new byte[] { 0, 1, 2, 3, 4};
```

U-SQL EXTRAXTORS and OUTPUTTERS

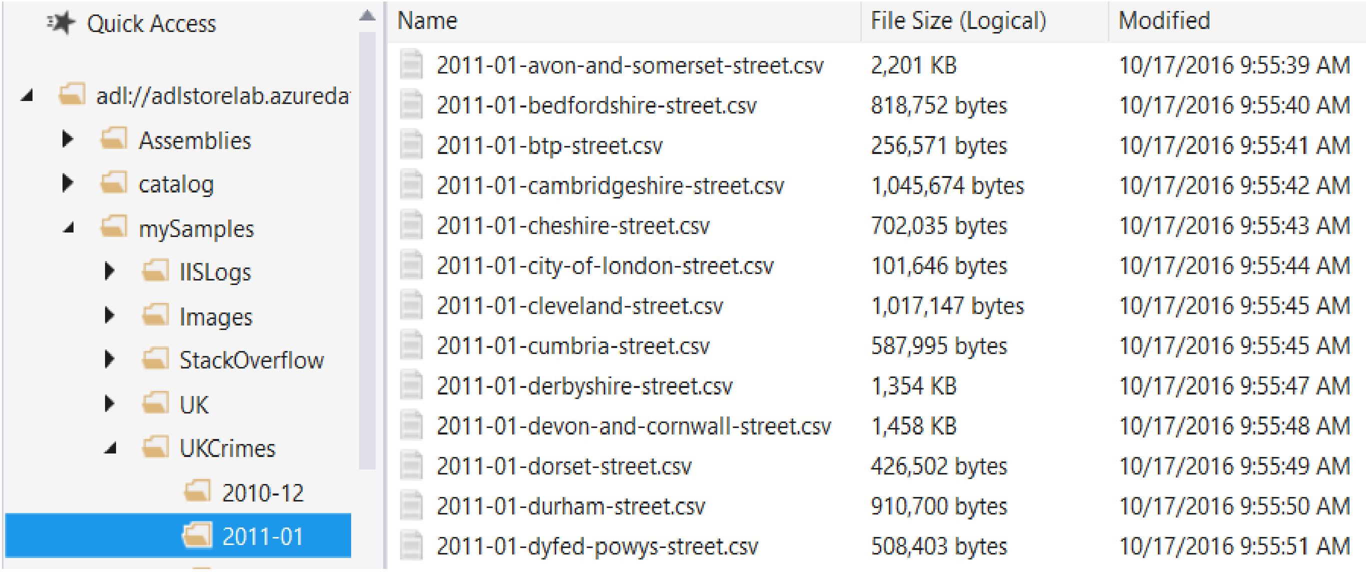
- Csv
- Text
- Tsv



```
public static Microsoft.Analytics.Interfaces.IExtractor Csv(System.Text.Encoding encoding);
public static Microsoft.Analytics.Interfaces.IExtractor Csv(System.String rowDelimiter,
    System.Nullable<System.Char> escapeCharacter, System.String nullEscape,
    System.Text.Encoding encoding, System.Boolean quoting, System.Boolean silent,
    System.Int32 skipFirstNRows, System.String charFormat);
public static Microsoft.Analytics.Interfaces.IExtractor Text(System.Text.Encoding encoding);
public static Microsoft.Analytics.Interfaces.IExtractor Text(System.Char delimiter,
    System.String rowDelimiter, System.Nullable<System.Char> escapeCharacter, System.String nullEscape,
    System.Text.Encoding encoding, System.Boolean quoting, System.Boolean silent, System.Int32 skipFirstNRows,
    System.String charFormat);
public static Microsoft.Analytics.Interfaces.IExtractor Tsv(System.Text.Encoding encoding);
public static Microsoft.Analytics.Interfaces.IExtractor Tsv(System.String rowDelimiter,
    System.Nullable<System.Char> escapeCharacter, System.String nullEscape,
    System.Text.Encoding encoding, System.Boolean quoting, System.Boolean silent,
    System.Int32 skipFirstNRows, System.String charFormat);
```

U-SQL FILESETS

```
DECLARE @inputCrimes = @"mySamples/UKCrimes/{Date:yyyy}-
{Date:MM}/{Input}-street.csv";
@crimes =
    EXTRACT CrimeID string,
            Month string,
            ReportedBy string,
            FallsWithin string,
            Longitude string,
            Latitude string,
            Location string,
            LSOACode string,
            LSOAName string,
            CrimeType string,
            LastOutcomeCategory string,
            Context string,
            Date DateTime,
            Input string
    FROM @inputCrimes
    USING Extractors.Csv(silent :
false,skipFirstNRows:1);
```



Name	File Size (Logical)	Modified
2011-01-avon-and-somerset-street.csv	2,201 KB	10/17/2016 9:55:39 AM
2011-01-bedfordshire-street.csv	818,752 bytes	10/17/2016 9:55:40 AM
2011-01-btp-street.csv	256,571 bytes	10/17/2016 9:55:41 AM
2011-01-cambridgeshire-street.csv	1,045,674 bytes	10/17/2016 9:55:42 AM
2011-01-cheshire-street.csv	702,035 bytes	10/17/2016 9:55:43 AM
2011-01-city-of-london-street.csv	101,646 bytes	10/17/2016 9:55:44 AM
2011-01-cleveland-street.csv	1,017,147 bytes	10/17/2016 9:55:45 AM
2011-01-cumbria-street.csv	587,995 bytes	10/17/2016 9:55:45 AM
2011-01-derbyshire-street.csv	1,354 KB	10/17/2016 9:55:47 AM
2011-01-devon-and-cornwall-street.csv	1,458 KB	10/17/2016 9:55:48 AM
2011-01-dorset-street.csv	426,502 bytes	10/17/2016 9:55:49 AM
2011-01-durham-street.csv	910,700 bytes	10/17/2016 9:55:50 AM
2011-01-dyfed-powys-street.csv	508,403 bytes	10/17/2016 9:55:51 AM

Virtual columns

WORKSHOP 1

Azure Data Lake

https://github.com/rkostrzewski/usql-workshop

U-SQL FILTERING

- ROWSET(s)
- TABLE(s)
- WHERE
- AND & OR
- ==,>=,!= (C# OPERATOR(s))
- CONTAINS (C# string)

```
@crimeInDay =
    SELECT CrimeType,
           Date,
           COUNT( * ) AS Count
    FROM @crimes
    WHERE Date >= @crimeDate AND (CrimeType.Contains("Other")
    OR CrimeType.StartsWith("Robb"))
```

U-SQL ROWSETS

```
@postCodes =
    EXTRACT id string,
            postcode string,
            latitude string,
            longitude string
    FROM @inputPostCodes
    USING Extractors.Csv(skipFirstNRows:1);

@topCities =
    EXTRACT id int,
            name string,
            population string,
            postcode string
    FROM @input10topCities
    USING Extractors.Text(delimiter : ';');

@topCitiesWithGPS =
    SELECT tc.name,tc.population, pc.latitude,pc.longitude
    FROM @topCities AS tc
         JOIN
             @postCodes AS pc
         ON pc.postcode == tc.postcode;
```

Rowset

Rowset

Rowset

WORKSHOP 2

U-SQL ARRAY and MAP

SQL.ARRAY<T> == IList<T>

```
@m = SELECT new SqlArray<string>
(
        tweet.Split(
        new char[]{''}).Where(x => x.StartsWith("@")))
        AS mentions
     FROM @t;

@m = SELECT m.Substring(1) AS m
          , "mention" AS category
     FROM @m CROSS APPLY EXPLODE(mentions) AS t(m)
```

SQL.MAP<T,U> ==IDictionary<T,U>

```
@ds =
    SELECT content,fileName, new SQL.MAP<int,string>() AS
colors
    FROM @rs;

@ds =
    PROCESS @ds
    PRODUCE content,colors,fileName
            READONLY fileName
    USING new ImageColorsProcessor(4);

@ds =
    SELECT fileName,
           order,
           colorName
    FROM @ds
         CROSS APPLY
             EXPLODE(colors) AS colors(order, colorName);
```

U-SQL SORTING

ROWSET

```
@distances =
    SELECT CrimeId,
           CityName,
           CrimeType,
           Year,
           Month
    FROM @merged
ORDER BY CityName DESC
FETCH FIRST 10 ROWS;
```

OUPUT

```
OUTPUT @ds
TO "result.csv"
ORDER BY fileName,
         order
USING Outputters.Csv();
```

SELECT with ORDER BY requires a FETCH FIRST

WORKSHOP 3

USQL -AGGREGATIONS

GROUP BY

HAVING

AGGREGATIONS

- MAX
- MIN
- SUM
- MAX
- MIN
- SUM
- **ARRAY_AGG**

```
@output =
    SELECT
        MAX(Duration) AS DurationMax,
        MIN(Duration) AS DurationMin,
        AVG(Duration) AS DurationAvg,
        SUM(Duration) AS DurationSum,
        VAR(Duration) AS DurationVarianve,
        STDEV(Duration) AS DurationStDev,
    FROM @searchlog
    GROUP BY Region
    HAVING DurationMin > 1;
```

U-SQL WINDOW FUNCTIONS

RANKING FUNCTIONS

- RANK
- DENSE_RANK
- NTILE
- ROW_NUMBER

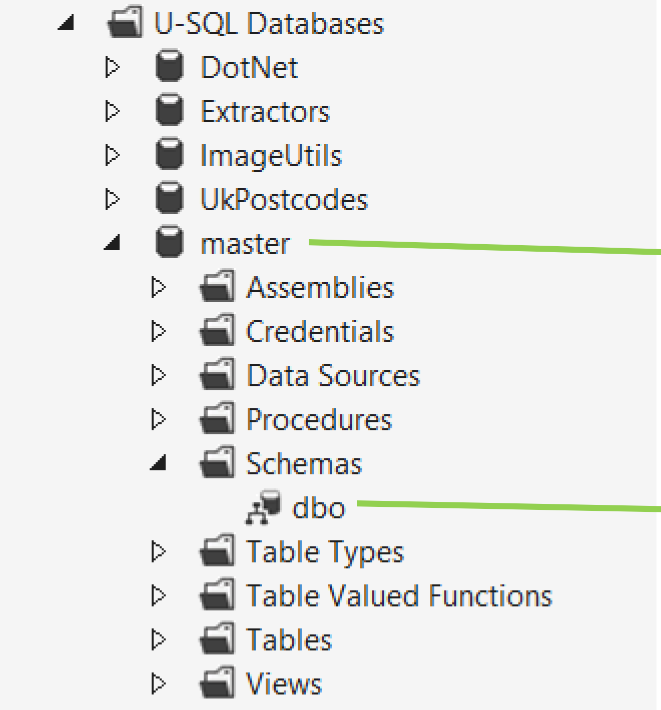
```
@result =
SELECT
    *,
    ROW_NUMBER() OVER (PARTITION BY Vertical ORDER BY
Latency) AS RowNumber,
    RANK() OVER (PARTITION BY Vertical ORDER BY Latency)
AS Rank,
    DENSE_RANK() OVER (PARTITION BY Vertical ORDER BY
Latency) AS DenseRank
FROM @querylog;
```

ANALIYTIC WINDOW FUNCTIONS

- CUME_DIST
- PERCENT_RANK
- PERCENTILE_CONT
- PERCENTILE_DISC
- CUME_DIST

WORKSHOP 4

U-SQL DATABASES AND SCHEMES



```
CREATE DATABASE IF NOT EXISTS UKCrimes;
USE DATABASE UKCrimes;
CREATE SCHEMA IF NOT EXISTS cr;
```

Default database

Default schema

U-SQL TABLES

- MANAGED TABLES and EXTERNAL TABLES
- ONLY INSERT
- CONSISTS OF FOUR THINGS:
  - A NAME
  - COLUMNS
  - A CLUSTERED INDEX
  - PARTITIONING SCHEME

```
DROP TABLE IF EXISTS vehiclesP;
CREATE TABLE vehiclesP(
      vehicle_id int
    , entry_id long
    , event_date DateTime
    , latitude float
    , longitude float
    , speed int
    , direction string
    , trip_id int?
    , INDEX idx CLUSTERED (vehicle_id ASC)
      PARTITIONED BY (event_date)
      DISTRIBUTED BY HASH (vehicle_id) INTO 4
);
```

U-SQL VIEWS and FUNCTIONS

VIEWS

```
CREATE VIEW IF NOT EXISTS vCrimes
    AS
EXTRACT CrimeID string,
        Month string,
        Date DateTime,
        Input string
FROM @"\UKCrimesCities\{Date:yyyy}-{Date:MM}\{Input}-
street.csv"
USING Extractors.Csv(silent : false,
 skipFirstNRows : 1);
```

FUNCTIONS (TVF)

```
CREATE FUNCTION tvf_Crimes(@input string)
RETURNS @result TABLE(CrimeID string,
 Month string)
AS
BEGIN
    @crimes =
    EXTRACT CrimeID string,
            Month string
    FROM @input
    USING Extractors.Csv(silent : false,
skipFirstNRows:1);

    @result = SELECT CrimeID,
            Month
            Input FROM @crimes;
    END;
```

U-SQL JOINS

- INNER JOIN
- FULL OUTER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- CROSS JOIN
- LEFT SEMIJOIN (IN)
- RIGHT SEMIJOIN (IN)
- LEFT ANTISEMIJOIN (NOT IN)
- RIGHT ANTISEMIJOIN (NOT IN)

```
@topCitiesWithGPS =
    SELECT tc.name,tc.population,
pc.latitude,pc.longitude
    FROM @topCities AS tc
         JOIN
             @postCodes AS pc
         ON pc.postcode ==
tc.postcode;
```

U-SQL C# METHODS

```
@distances =

    SELECT CrimeId,

           CityName,

           CrimeType,

           Year,

           Month,

           Gps.ComputeDistance

           (sLatitude, sLongitude, dLatitude, dLongitude)
AS Distance

    FROM @merged;
```

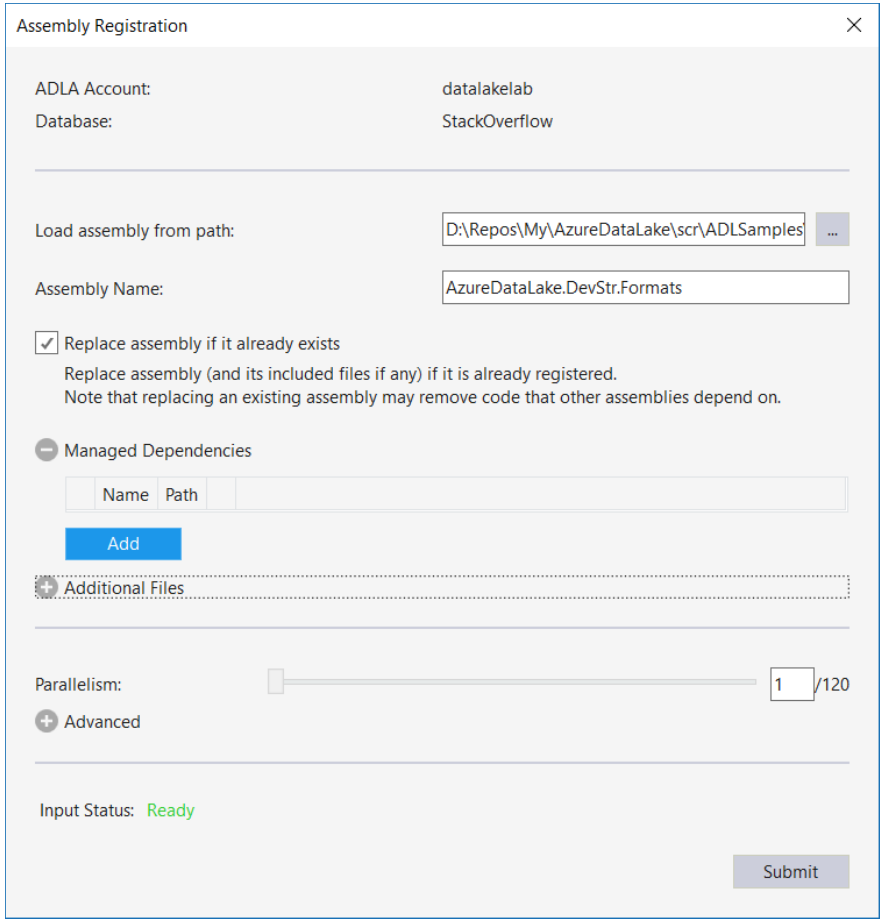
```
public static double ComputeDistance(double sLat, double
sLong, double dLat, double dLong)

        {

            var locA = new GeoCoordinate(sLat, sLong);

            var locB = new GeoCoordinate(dLat, dLong);

            return locA.GetDistanceTo(locB); // metres

        }
```

C# Method

IExtractor

```
[SqlUserDefinedExtractor(AtomicFileProcessing = true)]
    public class BinaryContentExtractor : IExtractor
    {
        public override IEnumerable<IRow> Extract(IUnstructuredReader input, IUpdatableRow output)
        {
            using (var ms = new MemoryStream())
            {
                input.BaseStream.CopyTo(ms);
                var content = ms.ToArray();
                output.Set(0, content);
                yield return output.AsReadOnly();
            }
        }
    }
```

U-SQL USING ASSEMBLIES



Assembly Registration

ADLA Account: datalakelab

Database: StackOverflow

Load assembly from path: D:\Repos\My\AzureDataLake\scr\ADLSamples

Assembly Name: AzureDataLake.DevStr.Formats

☑ Replace assembly if it already exists

Replace assembly (and its included files if any) if it is already registered.
Note that replacing an existing assembly may remove code that other assemblies depend on.

Managed Dependencies

Name	Path

Parallelism: 1 /120

Input Status: Ready

```
DECLARE @myAssemblyPath string =
@"D:\Repos\AzureDataLake.DevStr.Formats\bin\Debug\";

DECLARE @myAssemblyName string =
@myAssemblyPath+"AzureDataLake.DevStr.Formats.dll";

CREATE DATABASE IF NOT EXISTS Extractors;

USE DATABASE Extractors;

DROP ASSEMBLY IF EXISTS MyExtractors;

CREATE ASSEMBLY MyExtractors

FROM @myAssemblyName;
```

U-SQL USING ASSEMBLIES

```
DECLARE @imgFile string = @"D:\Help\BIGDATA\Images\{fileName}.jpg";
USE DATABASE Extractors;
REFERENCE ASSEMBLY MyExtractors;
USING BinaryExtractor = AzureDataLake.DevStr.Formats.BinaryContentExtractor;
REFERENCE ASSEMBLY ImageUtils.ImageUtils;
USING ImageColorsProcessor = AzureDataLake.DevStr.ImageUtils.ImageColorProducer;
@rs =
    EXTRACT content byte[],
            fileName string
    FROM @imgFile
    USING new BinaryExtractor();

@ds =
    SELECT content,fileName, new SQL.MAP<int,string>() AS colors
    FROM @rs;

@ds =
    PROCESS @ds
    PRODUCE content,colors,fileName
            READONLY fileName
    USING new ImageColorsProcessor(4);
```

Reference

Alias

External Extractor

External Processor

U-SQL

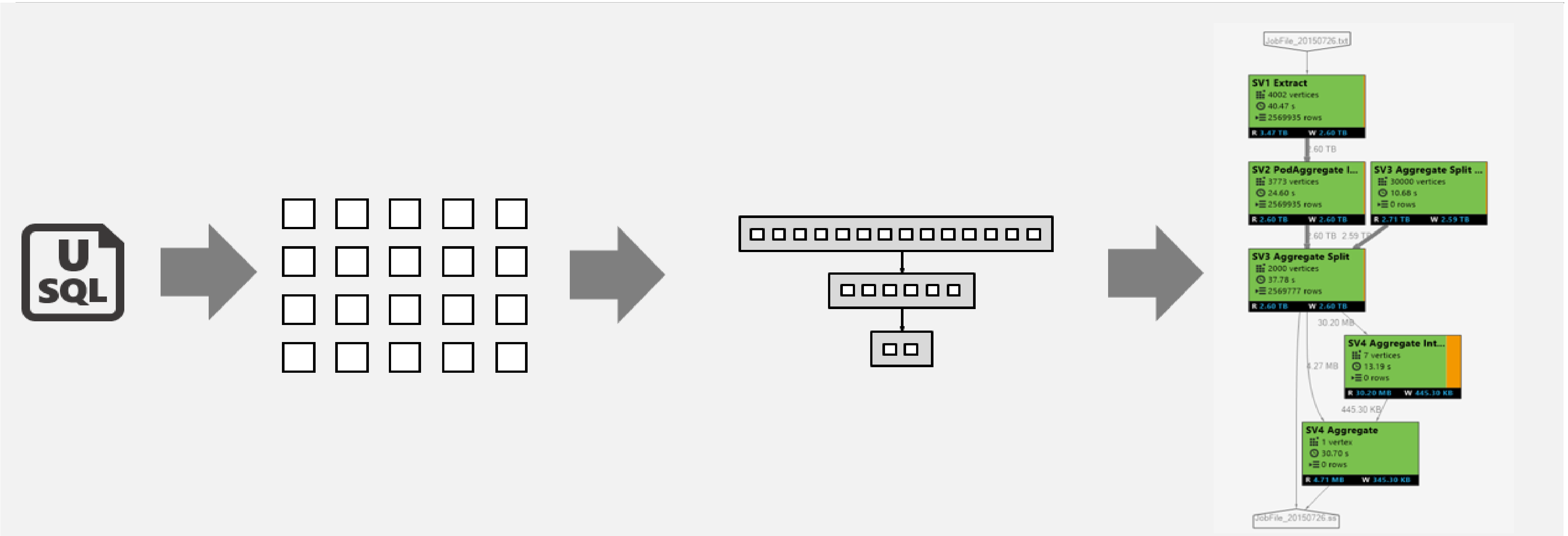
WORKSHOP 5

COFFEE BREAK

Azure Data Lake Analytics

Azure Data Lake Analytics Jobs Execution

U-SQL Script -> Job Graph Logical -> Physical Plan



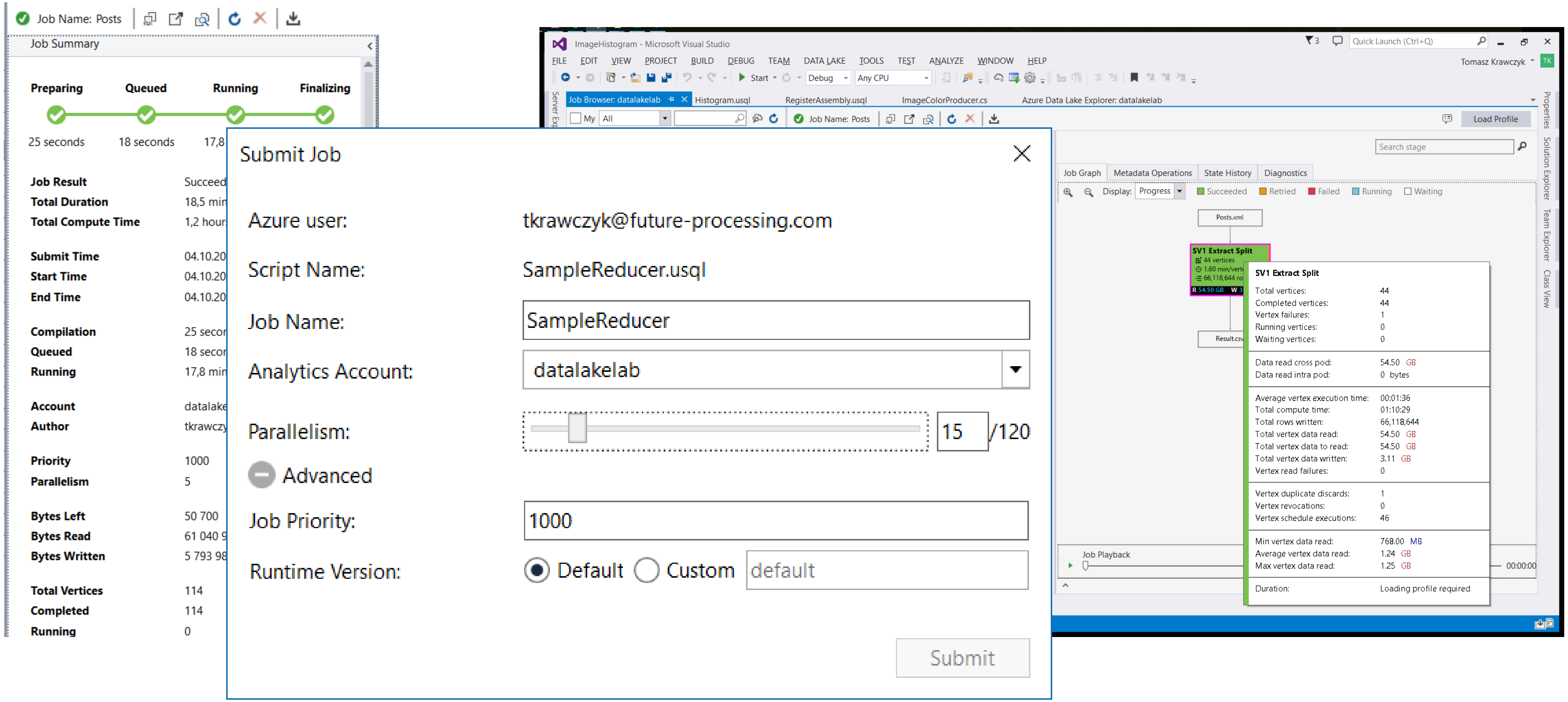
Each square = “a vertex” represents a fraction of the total

Vertexes in each SuperVertex (aka “Stage) are doing the same operation on a different part of the same data.

Visualized as a “Job Graph”

From: S. Reddy Presentation

Azure Data Lake Analytics Jobs



Job Name: Posts

Job Summary

Preparing	Queued	Running	Finalizing
25 seconds	18 seconds	17,8	

Job Result	Succeed
Total Duration	18,5 min
Total Compute Time	1,2 hour
Submit Time	04.10.20
Start Time	04.10.20
End Time	04.10.20
Compilation	25 secor
Queued	18 secor
Running	17,8 min
Account	datalake
Author	tkrawczy
Priority	1000
Parallelism	5
Bytes Left	50 700
Bytes Read	61 040 9
Bytes Written	5 793 98
Total Vertices	114
Completed	114
Running	0

Submit Job

Azure user: tkrawczyk@future-processing.com

Script Name: SampleReducer.usql

Job Name: SampleReducer

Analytics Account: datalakelab

Parallelism: 15 /120

Advanced

Job Priority: 1000

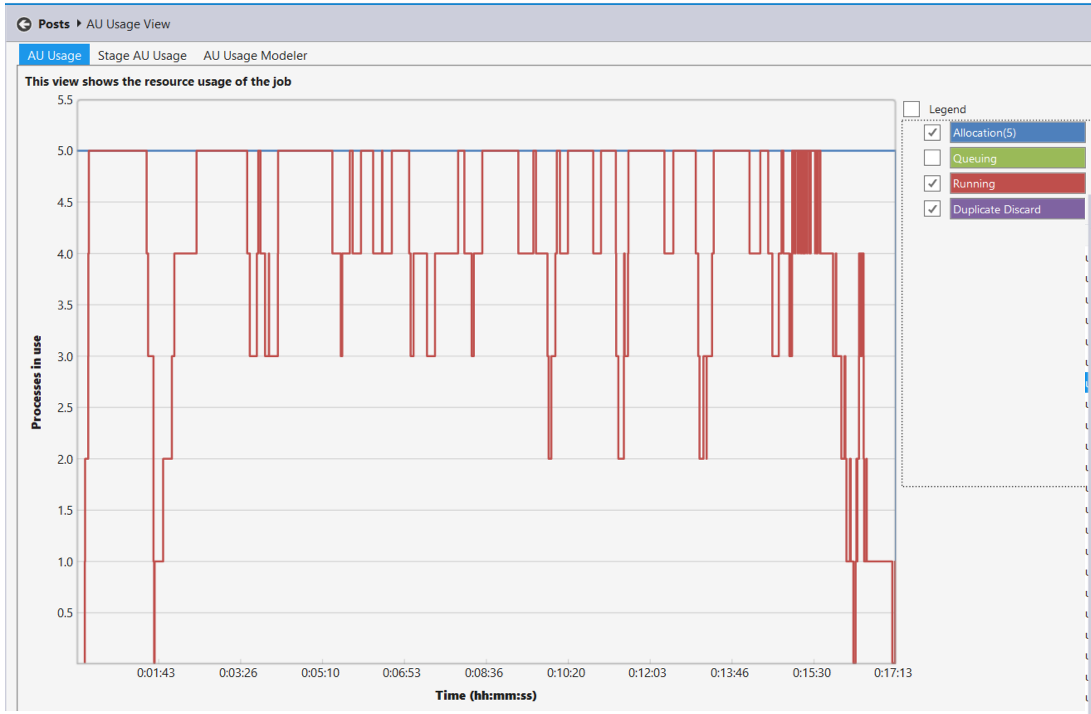
Runtime Version: ● Default ○ Custom default

Submit

SV1 Extract Split

Total vertices:	44
Completed vertices:	44
Vertex failures:	1
Running vertices:	0
Waiting vertices:	0
Data read cross pod:	54.50 GB
Data read intra pod:	0 bytes
Average vertex execution time:	00:01:36
Total compute time:	01:10:29
Total rows written:	66,118,644
Total vertex data read:	54.50 GB
Total vertex data to read:	54.50 GB
Total vertex data written:	3.11 GB
Vertex read failures:	0
Vertex duplicate discards:	1
Vertex revocations:	0
Vertex schedule executions:	46
Min vertex data read:	768.00 MB
Average vertex data read:	1.24 GB
Max vertex data read:	1.25 GB
Duration:	Loading profile required

Data Lake Analytics - AU Usage



Posts ▸ AU Usage View

AU Usage | Stage AU Usage | AU Usage Modeler

This view shows the resource usage of the job

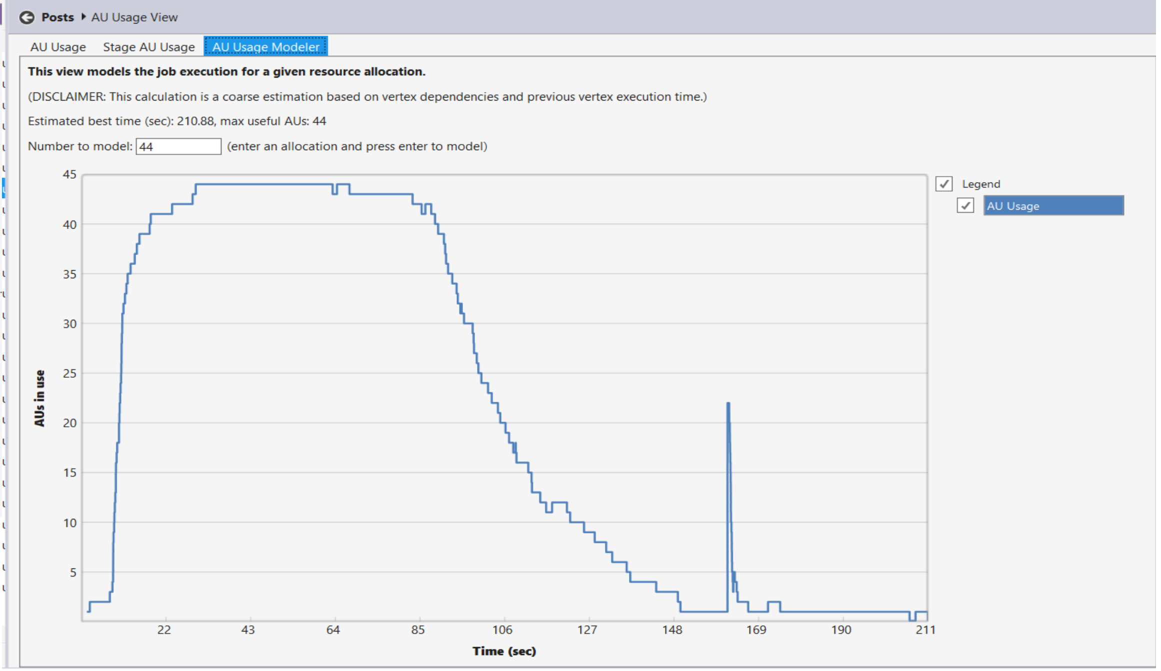
□ Legend

☑ Allocation(5)

□ Queuing

☑ Running

☑ Duplicate Discard



Posts ▸ AU Usage View

AU Usage | Stage AU Usage | AU Usage Modeler

This view models the job execution for a given resource allocation.

(DISCLAIMER: This calculation is a coarse estimation based on vertex dependencies and previous vertex execution time.)

Estimated best time (sec): 210.88, max useful AUs: 44

Number to model: 44 (enter an allocation and press enter to model)

☑ Legend

☑ AU Usage

Azure Data Lake Analytics

AZURE DEMO

Azure Data Lake

Q&A