Analiza Big Data z Azure Data Lake

Tomasz Krawczyk

Plan

- Big Data
- Azure Data Lake
  - Azure Data Lake Store and Analytics
  - U-SQL
    - Deep Dive (ADLA Catalog, Extentions, Partitioning)
  - Azure Data Lake Analytics Runtime
  - Costs and Optimization
  - Azure Data Lake and Azure Data Factory
- Demo and Q&A

Why Big Data ?

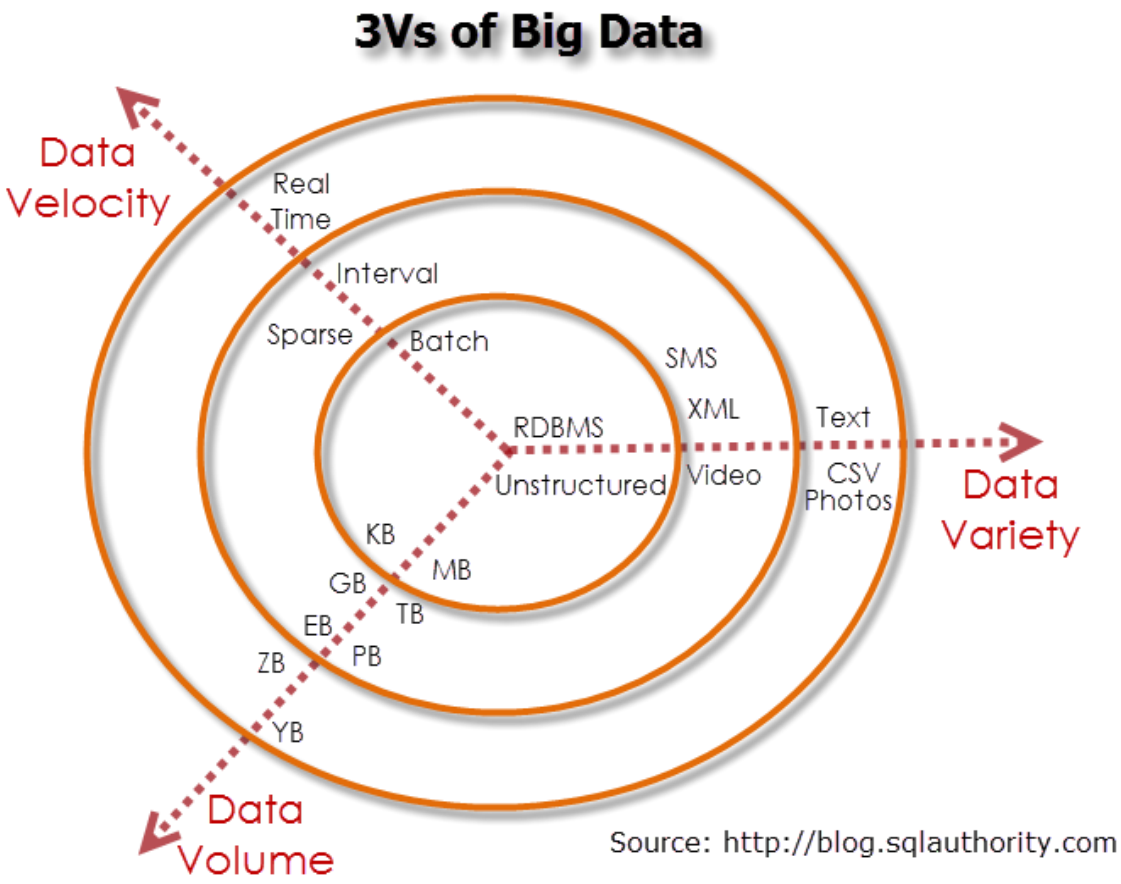
- Up to 80% of business processes in companies will be based on Big Data by 2020
- By 2020, the number of devices will grow to 50 billion or more
  - Currently 6+ billion devices connected to the Internet
- 40 Zetta bytes by 2020
- 163 Zetta bytes by 2025

40 Zetta bytes by 2020

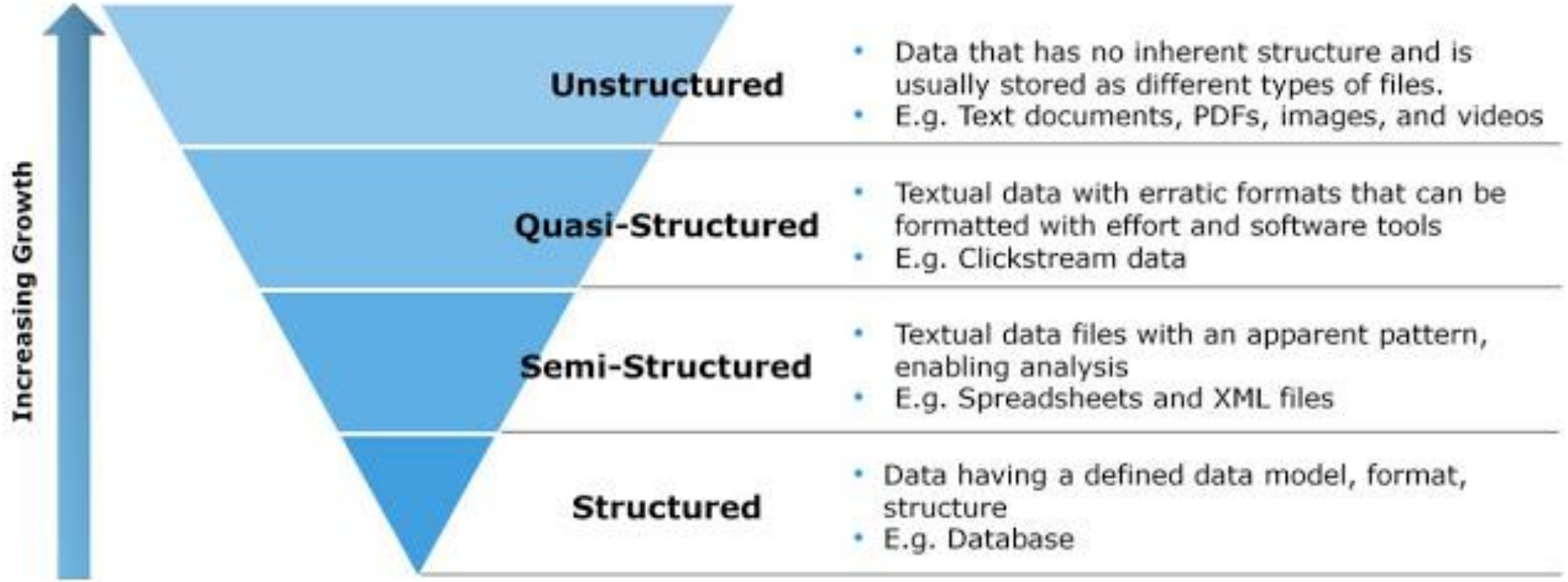
- The „rice comparison”
  - **Byte** One grain of rice
  - **Kilobyte** Cup of rice
  - **Megabyte** 8 bags of rice
  - **Gigabyte** 3 semi trucks
  - **Terabyte** 2 container ships
  - **Petabyte** Blankets Manhattan
  - **Exabyte** Blankets west coast states
  - **Zettabyte** Fills the Pacific Ocean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

Source: Microsoft //connec

3Vs of Big Data



The Structure of Big Data



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

Schema-on-Read vs Schema-on-Write

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.

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.

How do you process all the data ?

Data Lake Approach

› What is a Data Lake ?

„A data mart (a subset of a data warehouse) as akin to a bottle of water…”cleansed, packaged and structured for easy consumption” while a data lake is more like a body of water in its natural state. „

Pentaho CTO James Dixon

Data Lake Approach

What is a Data Lake ?

I Ingest

S Store

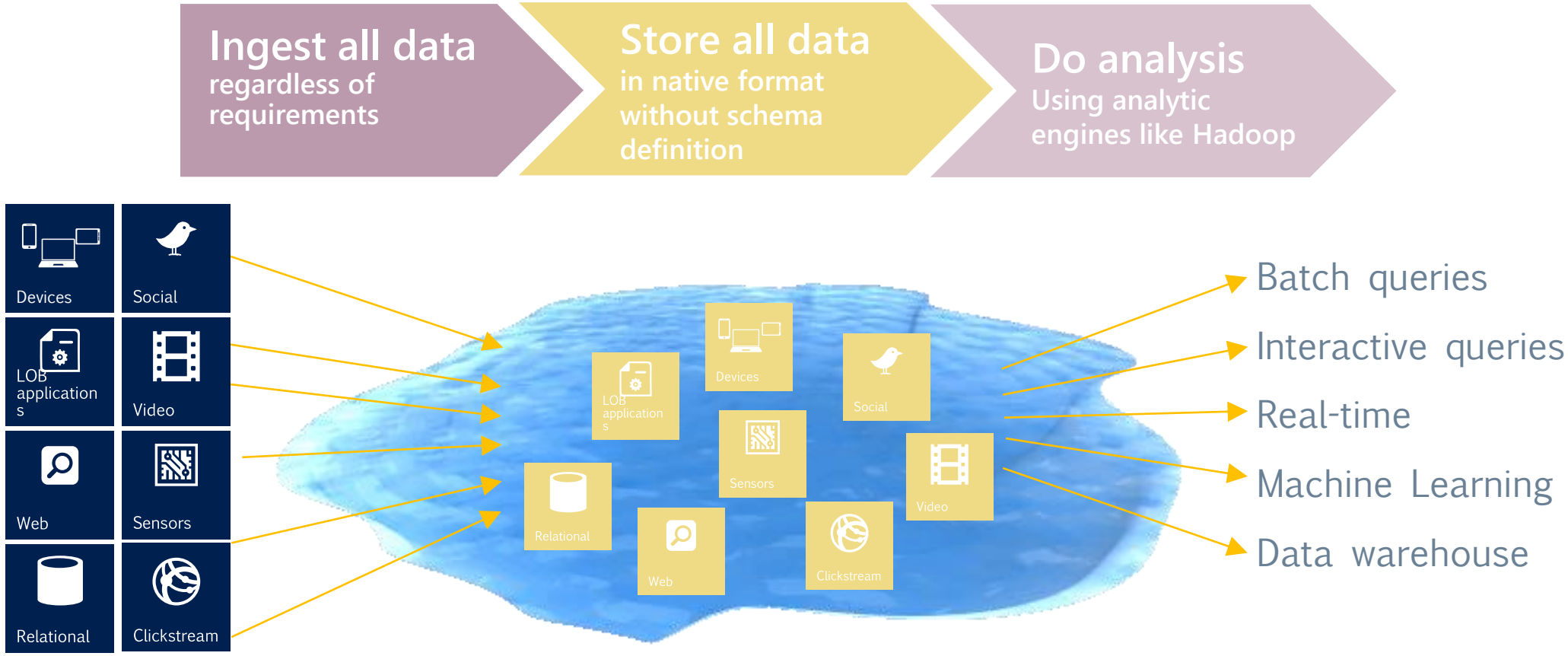
A Analyse

S Surface

A Act

Make me more money

Data Lake Approach



Cloud Service Models and Azure

IaaS

Virtual Machines

SaaS

Office 365

WorkPress

PaaS

App Services

Cloud Services

Cluster as a Service

Query/Job as a Service

HDInsight

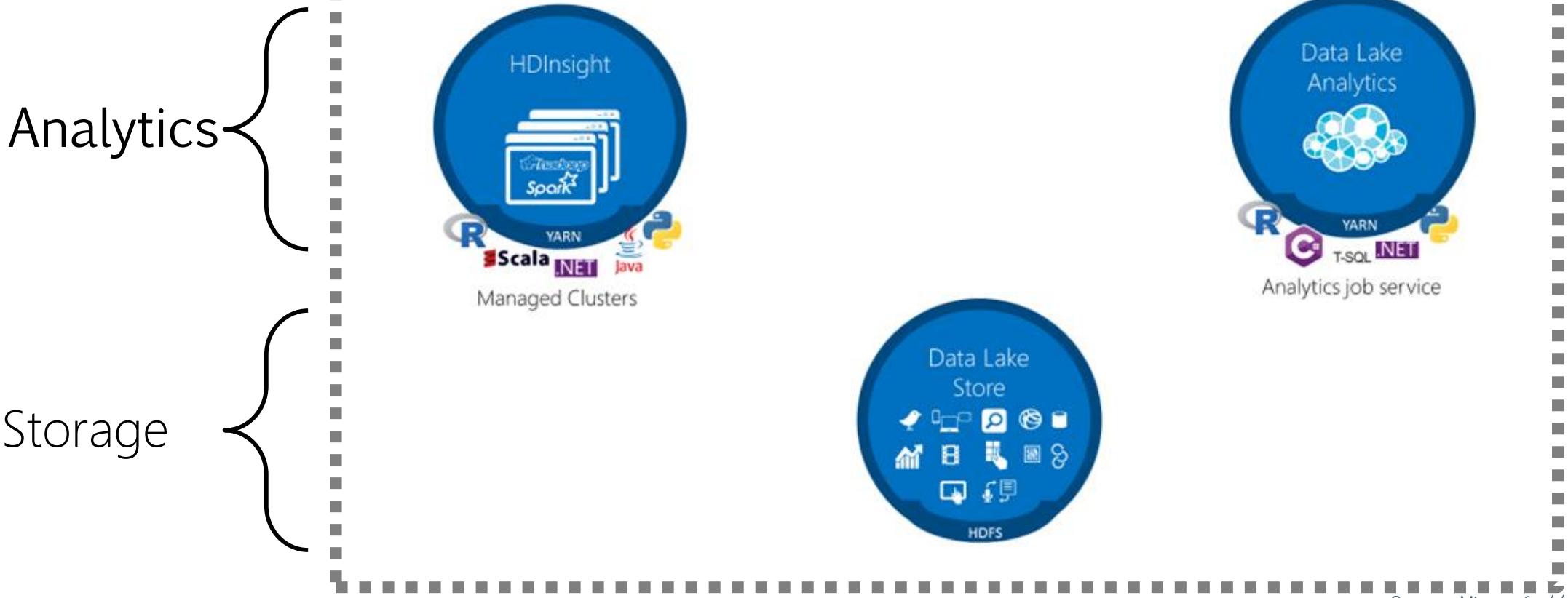
Data Lake Analytics

Azure Data Lake

Mike Rys

Saveen Reddy

Microsoft Azure Data Lake



Source: Microsoft //connec

Azure Data Lake Store

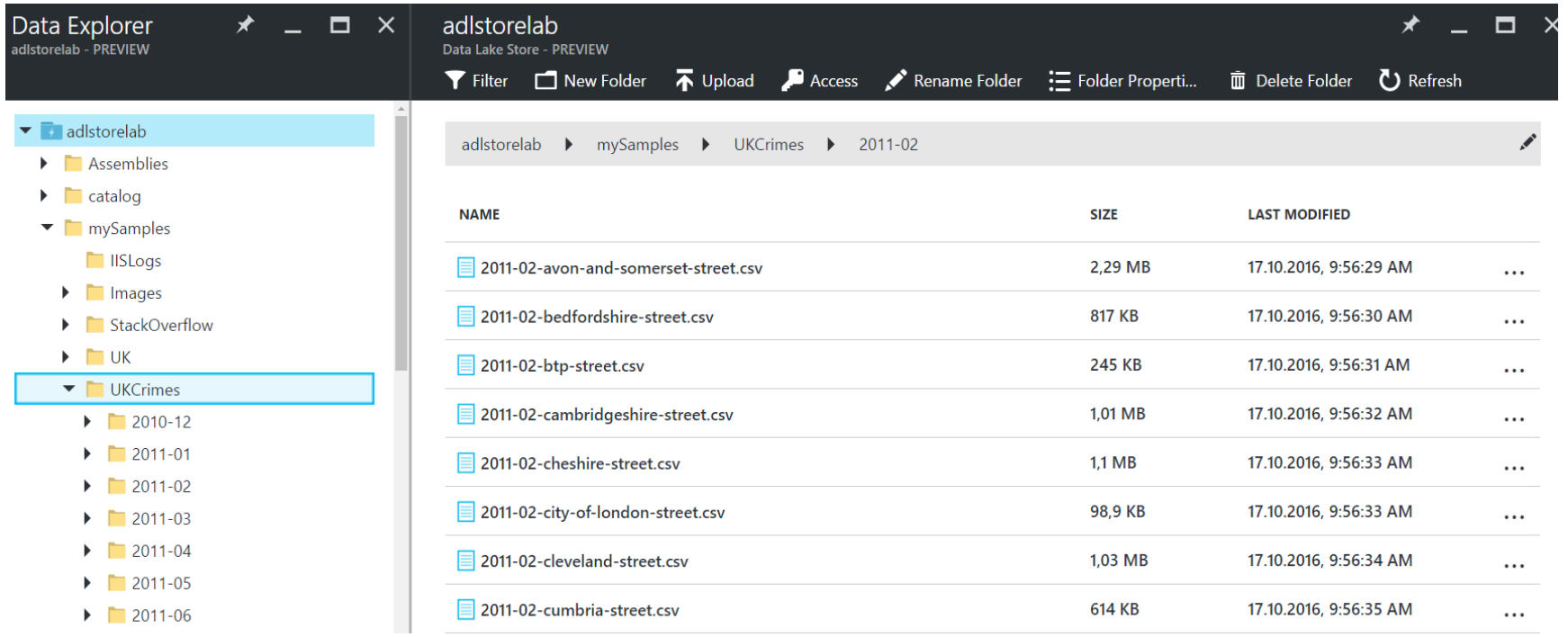
- Built for Hadoop
- WebHDFS-compatible REST interface
- Unlimited storage, petabyte files
- Performance-tuned for big data analytics
- Highly-available and secure
- Integrates with HDInsight, Cloudera, Hortonworks
- Supports files and folders objects
- Files are split apart into Extents (250 MB)
- For availability and reliability, extents are replicated (3 copies).
- Enables: Parallel read and Parallel write

Azure Data Lake Store vs Blob

	Azure Data Lake Store	Azure Blob Storage
Purpose	Optimized storage for big data analytics workloads	General purpose object store for a wide variety of storage scenarios
Use Cases	Batch, interactive, streaming analytics and machine learning data such as log files, IoT data, click streams, large datasets	Any type of text or binary data, such as application back end, backup data, media storage for streaming and general purpose data
Key Concepts	Data Lake Store account contains folders, which in turn contains data stored as files	Storage account has containers, which in turn has data in the form of blobs
Structure	Hierarchical file system	Object store with flat namespace
API	REST API over HTTPS	REST API over HTTP/HTTPS
Hadoop File System Client	Yes	Yes
Data Operations - Authentication	Based on Azure Active Directory Identities	Based on shared secrets - Account Access Keys and Shared Access Signature Keys.
Data Operations - Authorization	POSIX Access Control Lists (ACLs). ACLs based on Azure Active Directory Identities can be set file and folder level.	For account-level authorization – Use Account Access Keys For account, container, or blob authorization - Use Shared Access Signature Keys

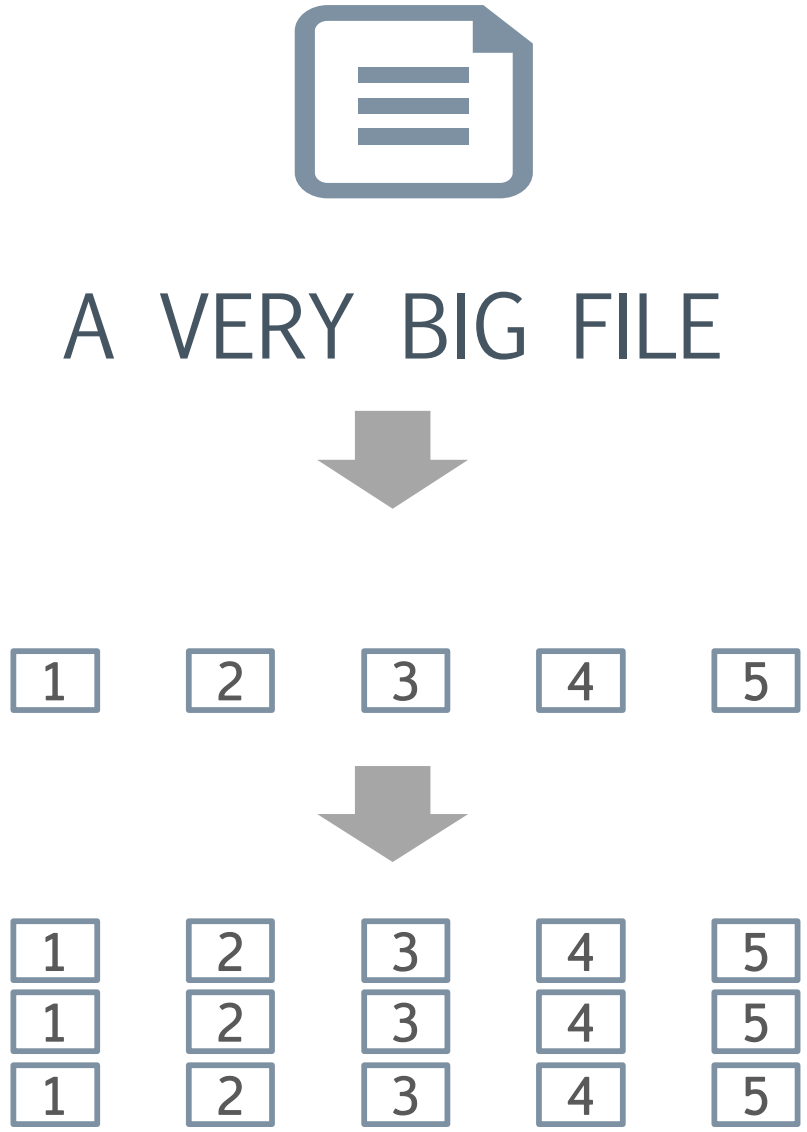
Azure Data Store

- Hierarchical structre
  - Folders
  - Files
- Authorization : POSIX



Azure Data Lake Store

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



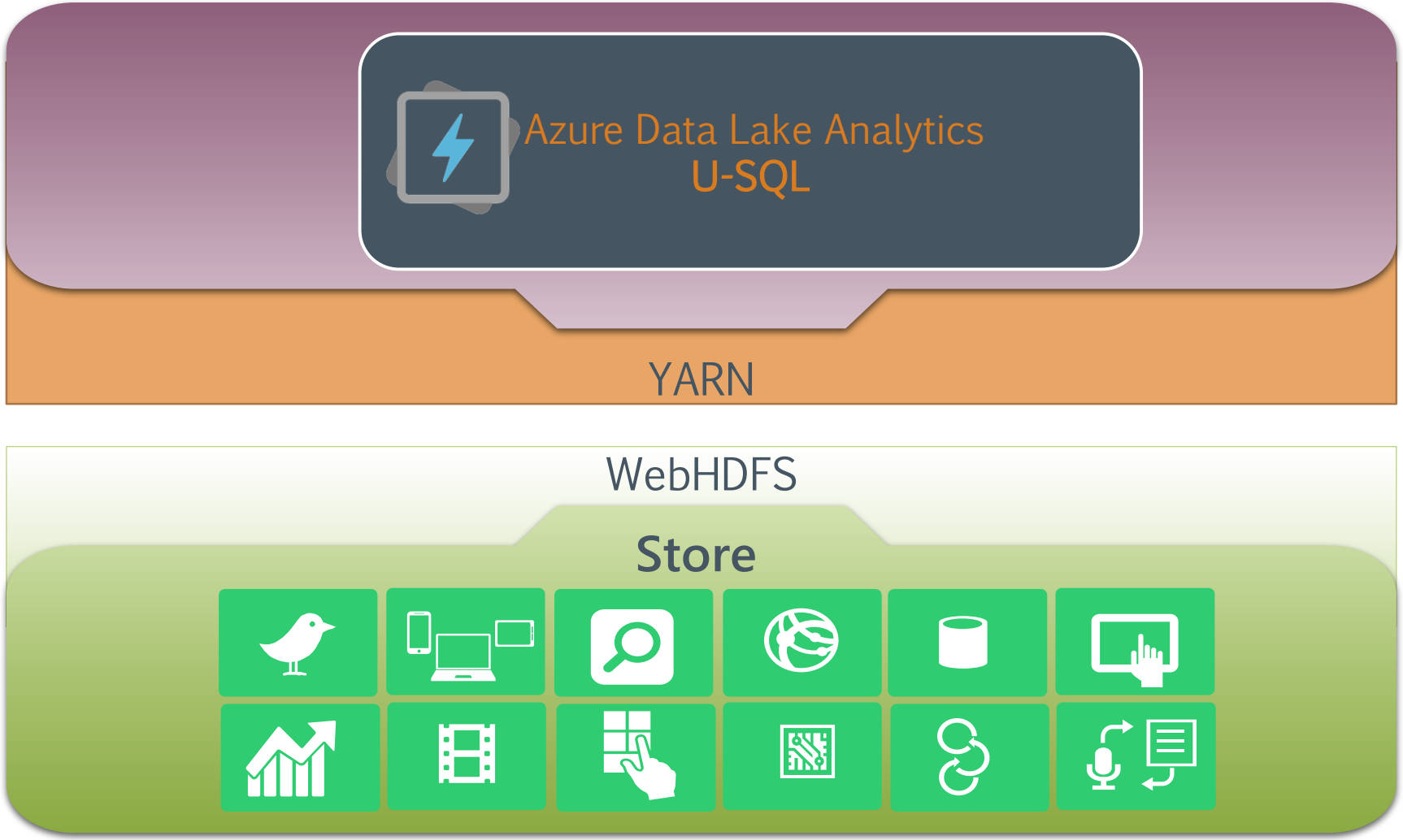
Working with Azure Data Lake Store

- Local computer
  - Azure Data Factory
  - 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
- HDInsight Cluster (+Spark)
- **WebHDFS**-compatible REST interface

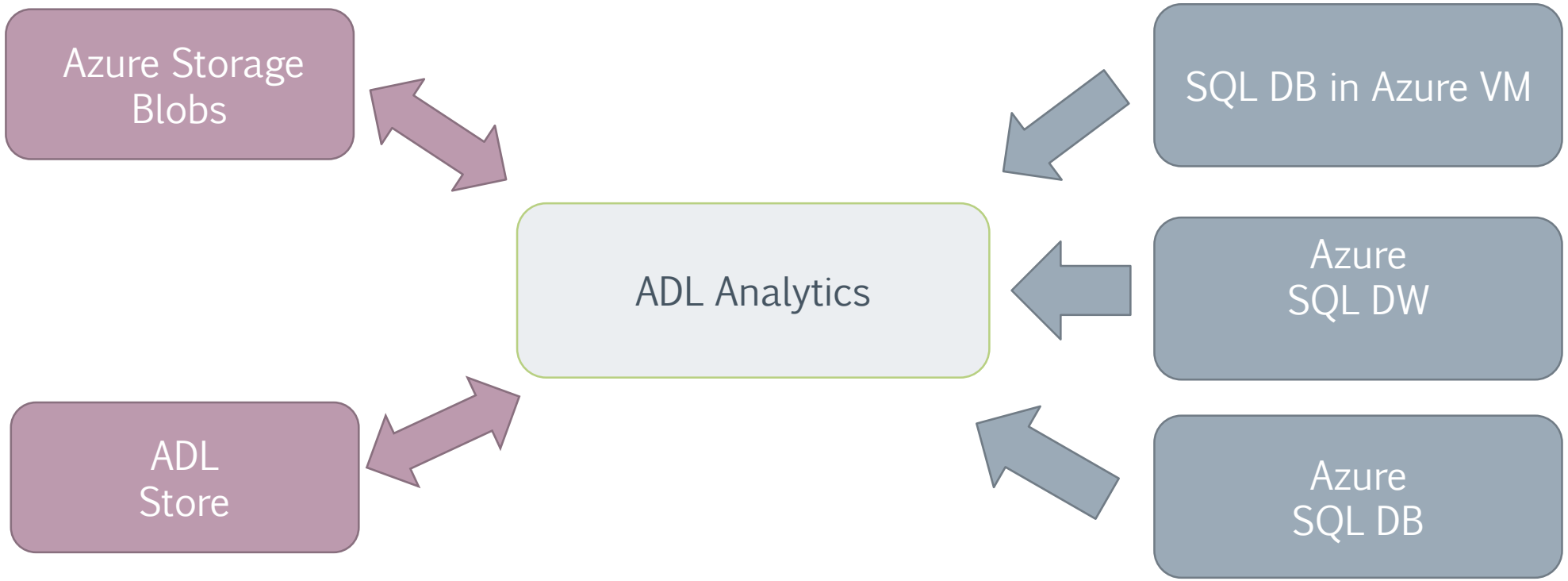
Azure Data Lake Analytics

- **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

Azure Data Lake Analytics



Work across all your cloud Data

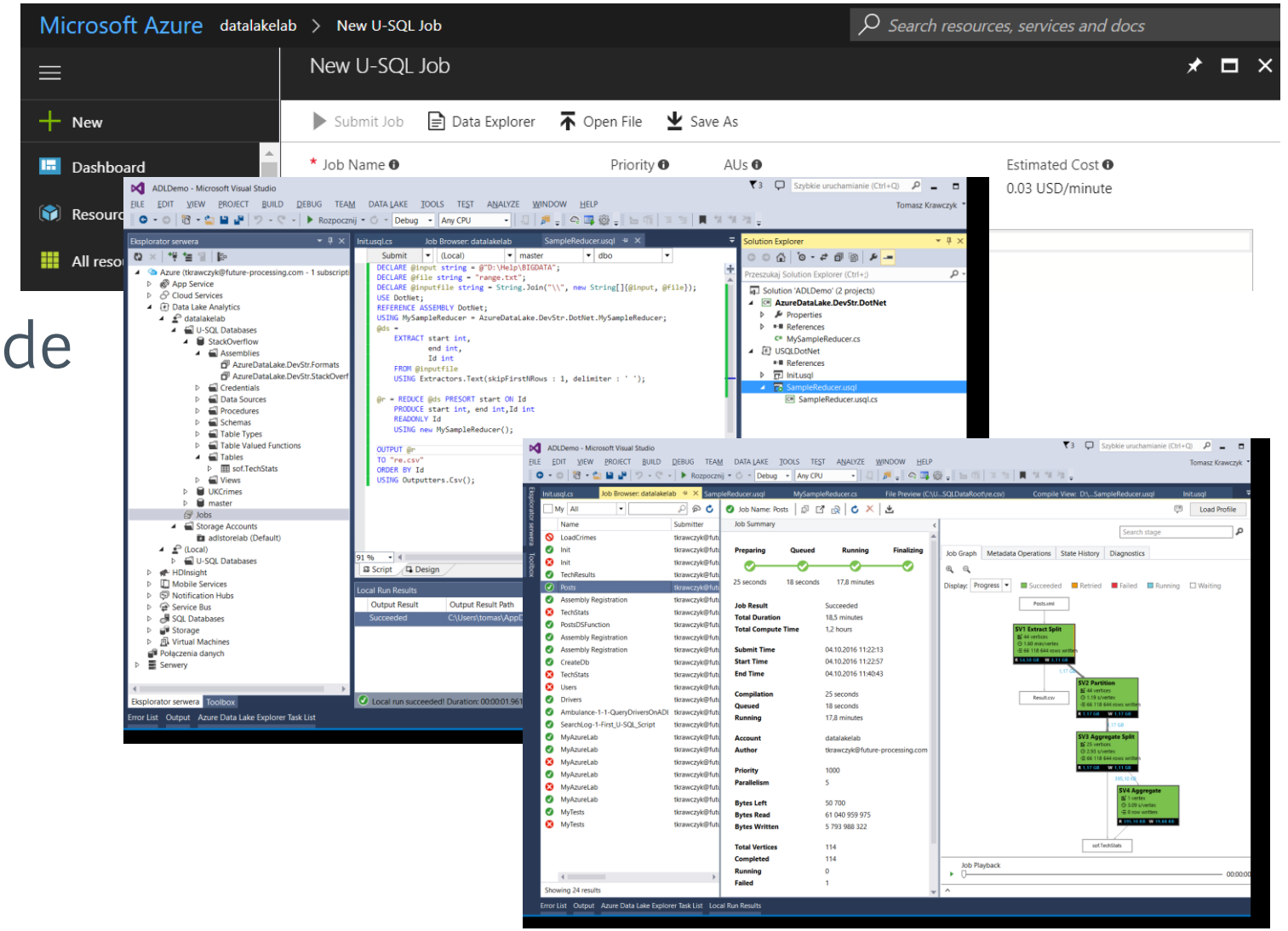


Azure Data Lake - Where we are?

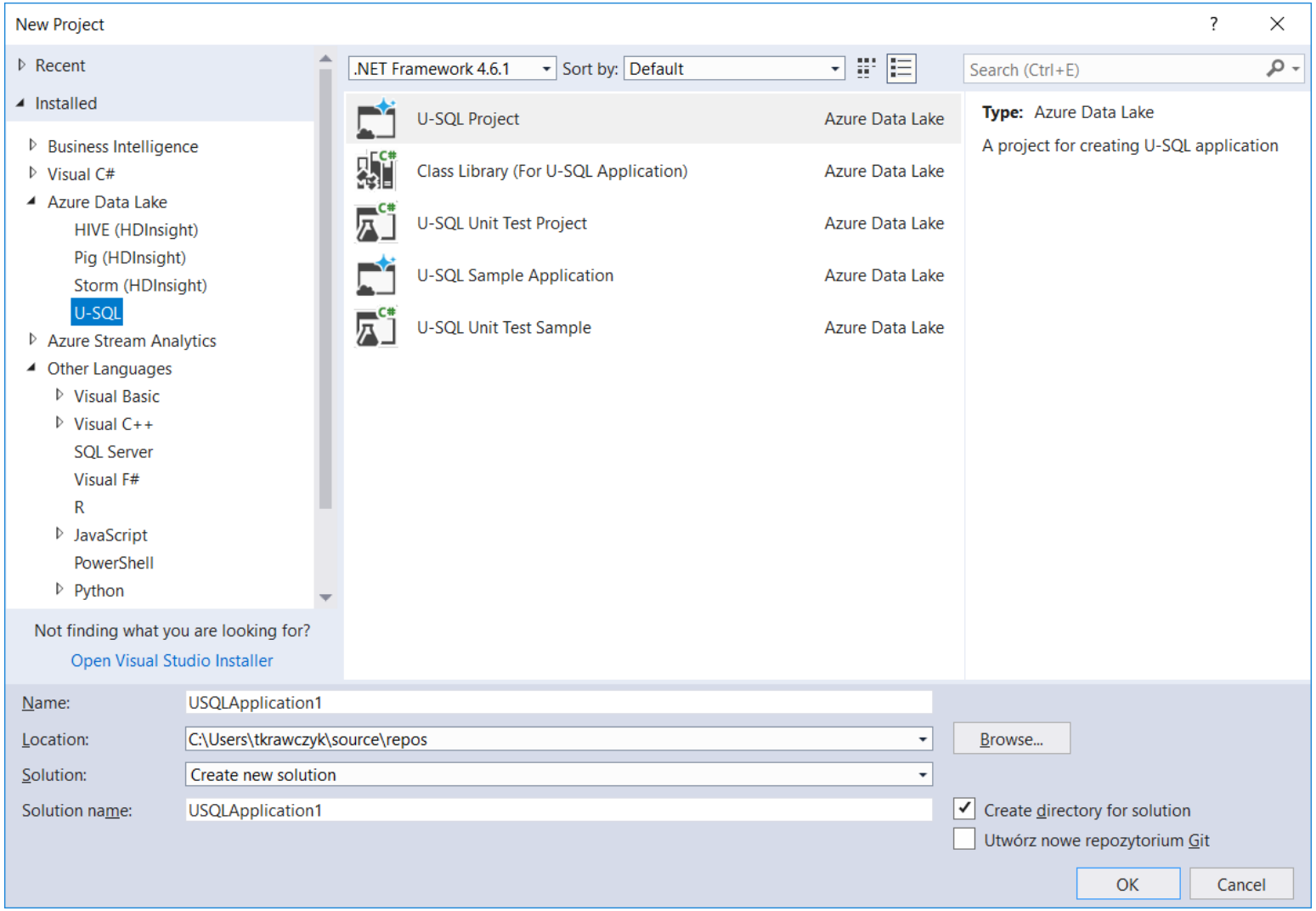
- BATCH
  - AVAILABLE NOW
- INTERACTIVE
  - PUBLIC PREVIEW IN 2018
- MACHINE LEARNING
  - PUBLIC PREVIEW IN 2017
- STREAMING ANALYTICS
  - ON ROADMAP

Developer Tools

- Azure Portal
- Visual Studio
  - ADLA Tools
- Visual Studio Code
- Power Shell



Developers Tools – Visual Studio

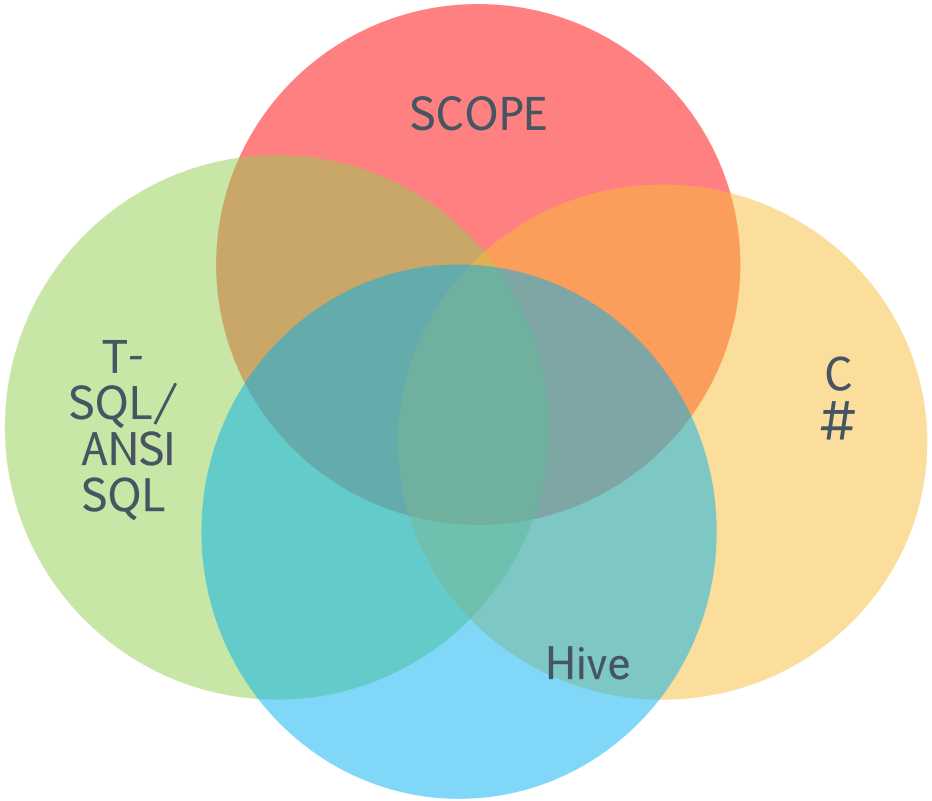


DEMO(s)

- Azure Portal
  - Create Azure Data Lake
  - Visual Studio
  - Data Explorer

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



SQL DECLARATIVITY + C# EXTENSIBILITY = U-SQL

Source M. RYs

U–SQL Script

```
DECLARE @projectsInput string = @"Projects\{file}.csv";
DECLARE @eventDate DateTime = System.DateTime.Parse("2017/04/01");
DECLARE @numbers int = 2;
REFERENCE ASSEMBLY USQLCSharpDemo;
USING ImageColorsProcessor = USQLCSharpDemo.ImageColorProducer;
@projects =
    EXTRACT project string,
            startDate DateTime,
            endDate DateTime,
            file string
    FROM @projectsInput
    USING Extractors.Csv(skipFirstNRows : 1, quoting : true);
@rs =
    EXTRACT content byte[],
            fileName string
    FROM @imgFiles
    USING new BinaryExtractor();
@assignments =
    SELECT user.ToUpper() AS user,
           new SqlArray<string>(projects.Split(new char[]{'|'},
StringSplitOptions.RemoveEmptyEntries)) AS projs
    FROM @usersprojects;
@agg =
    SELECT project,
           COUNT( * ) AS units
    FROM @details WHERE project.StartsWith("My")
    GROUP BY project;
@myprojects =
    SELECT us.project,
           p.endDate
    FROM @details AS us
         JOIN
             @projects AS p
         ON p.project == us.project
    WHERE user.StartsWith("Me")
ORDER BY p.endDate DESC
FETCH 10 ROWS;
OUTPUT @myprojects
TO "myprojects.csv"
USING Outputters.Csv();
```

C# Types

.NET Assemblies

Apply Schema on Read

Extractor

Rowset(s)

External Extractor

SQL Dialect (SELECT FROM)

SQL Aggregation(s)

.NET Methods

SQL Dialect (JOIN, WHERE, ORDER BY ...)

Output

U-SQL DECLARE VARIABLES

```
DECLARE @text1 string = “Big Data";
DECLARE @text2 string = @“Azure as a Big Data Platform";
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,
“2017"});

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 Data Types

- **Numeric**
  - sbyte
  - int
  - long
  - float
  - double
  - decimal
  - short
  - byte
  - uint
  - ulong
  - ushort
- **Text**
  - char
  - String (128 kB)
- **Complex**
  - MAP<k,v>
  - ARRAY<v>
- **Miscellaneous**
  - bool
  - Guid
  - DateTime
  - byte[]
- **RowSets**

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 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 1 (@postCodes)

Rowset 2 (@topCities)

Rowset 3 (@topCitiesWithGPS)

U-SQL EXTRACT

```
Project,StartDate,EndDate
"BigBang",2015-01,2016-12
"BigData",2016-08,2016-10
"Project1",2016-01,2016-09
"Project2",2016-11,2016-12
"Project3",2014-01,2018-12
"Project4",2016-01,2017-08
"Project5",2016-01,2017-12
"Project6",2015-01,2017-02
"Project7",2015-01,2016-12
"Project8",2015-08,2016-08
"Project9",2014-01,2016-12
"NewProject",2017-12,2018-12
```

```
DECLARE @projectsInput string =
@"Projects\projects.csv";
@projects =
    EXTRACT project string,
            startDate DateTime,
            endDate DateTime
    FROM @projectsInput
    USING Extractors.Csv(
        skipFirstNRows : 1,
        quoting : true);
```

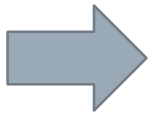
Apply Schema on Read

Extractor
with additional options

U-SQL EXTRACT

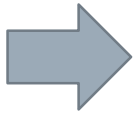
Projects.csv

```
Project,StartDate,EndDate
"BigBang",2015-01,2016-12
"BigData",2016-08,2016-10
"Project1",2016-01,2016-09
"Project2",2016-11,2016-12
"Project3",2014-01,2018-12
"Project4",2016-01,2017-08
"Project5",2016-01,2017-12
"Project6",2015-01,2017-02
"Project7",2015-01,2016-12
"Project8",2015-08,2016-08
"Project9",2014-01,2016-12
"NewProject",2017-12,2018-12
```



USQL Extract

```
DECLARE @projectsInput string =
@"Projects\projects.csv";
@projects =
    EXTRACT project string,
            startDate DateTime,
            endDate DateTime
    FROM @projectsInput
    USING Extractors.Csv(
           skipFirstNRows : 1,
           quoting : true);
```

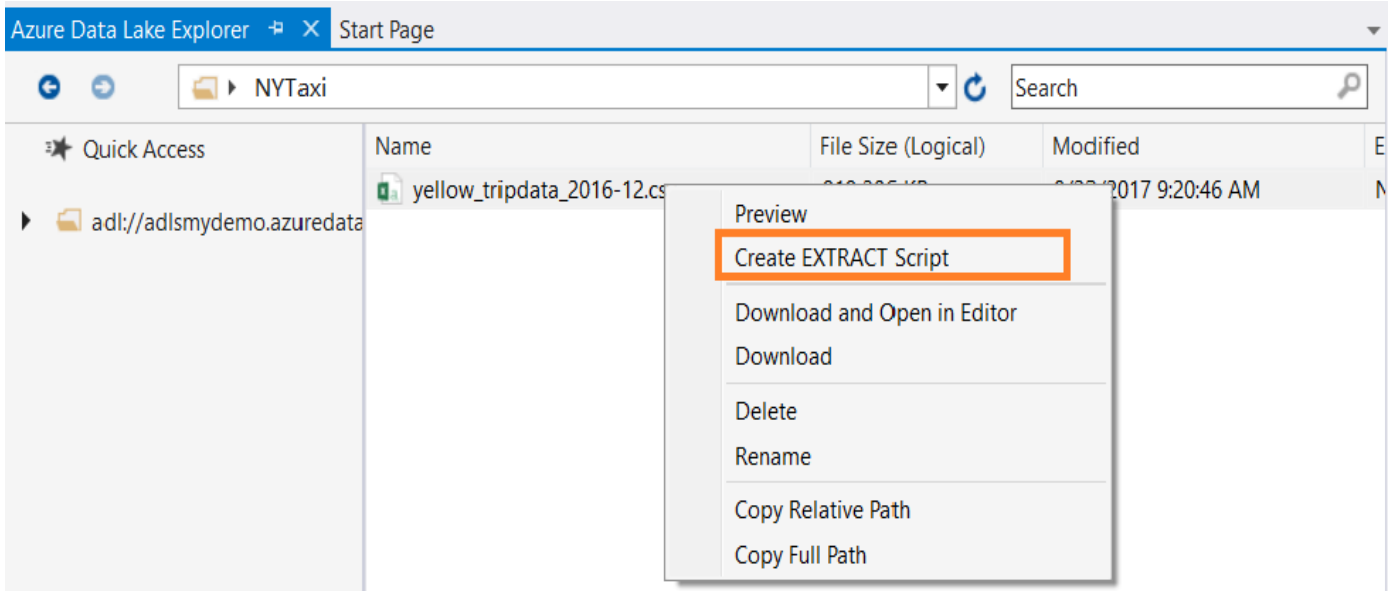


Rowset @projects

projects	startDate	endDate
BigBang	2015-01-01	2016-12-01
BigData	2016-01-01	2016-09-01
…	…	..

U-SQL EXTRACTOR

Create EXTRACT Script via Azure Data Lake Explorer



U-SQL EXTRACTORS and OUTPUTERS

- List of EXTRACTORS AND OUTPUTERS
  - CSV
  - TEXT
  - TSV
  - GZIP (*)
  - PARQUET (*)
- API
  - IExtractor
  - IOutputter

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);
```

FileSets ({Date},{Input})

Virtual Columns

DEMO(s)

- Sample\Rowsets
- Sample\Arrays
- Sample\Filesets
- Demos_001\Demo0IILogs

U-SQL FILTERING AND SORTING

- WHERE
  - AND & OR
  - ==, >=, != (C# OPERATOR(s))
  - CONTAINS (C# string)
- ORDER BY
  - ROWSETS
    - requires a FETCH
  - OUTPUTS

```
@distances =
    SELECT CrimeId,
           CityName,
           CrimeType,
           Year,
           Month
    FROM @merged
    WHERE
CrimeType.StartWith("Soc") AND
Year == 2016
ORDER BY Month DESC
FETCH FIRST 10 ROWS;
```

U-SQL -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 WINDOWING FUNCTIONS

- RANKING FUNCTIONS
  - RANK
  - DENSE_RANK
  - NTILE
  - ROW_NUMBER
- ANALIYTIC WINDOW FUNCTIONS
  - CUME_DIST
  - PERCENT_RANK
  - PERCENTILE_CONT
  - PERCENTILE_DISC
  - CUME_DIST

```
@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;
```

U–SQL

```
CREATE DATABASE IF NOT EXISTS AzureMeetup201704;
USE AzureMeetup201704;
CREATE SCHEMA IF NOT EXISTS sof;
DROP TABLE IF EXISTS sof.TechStats;
CREATE TABLE sof.TechStats
(
    Category string,
    Year int,
    Month int,
    ViewCount int,
    INDEX idx_TechStats
    CLUSTERED(Category)
    DISTRIBUTED BY HASH(Category) INTO 3
);
INSERT INTO sof.TechStats
(
    Category,
    Year,
    Month,
    ViewCount
)
SELECT Category,
       Year,
       Month,
       ViewCount
FROM @postsByCategory
     WHERE Category != "other";
@ds =
    SELECT Category,
           Year,
           Month,
           SUM(ViewCount) AS ViewCount
    FROM sof.TechStats
    GROUP BY Category,
             Year,
             Month;
```

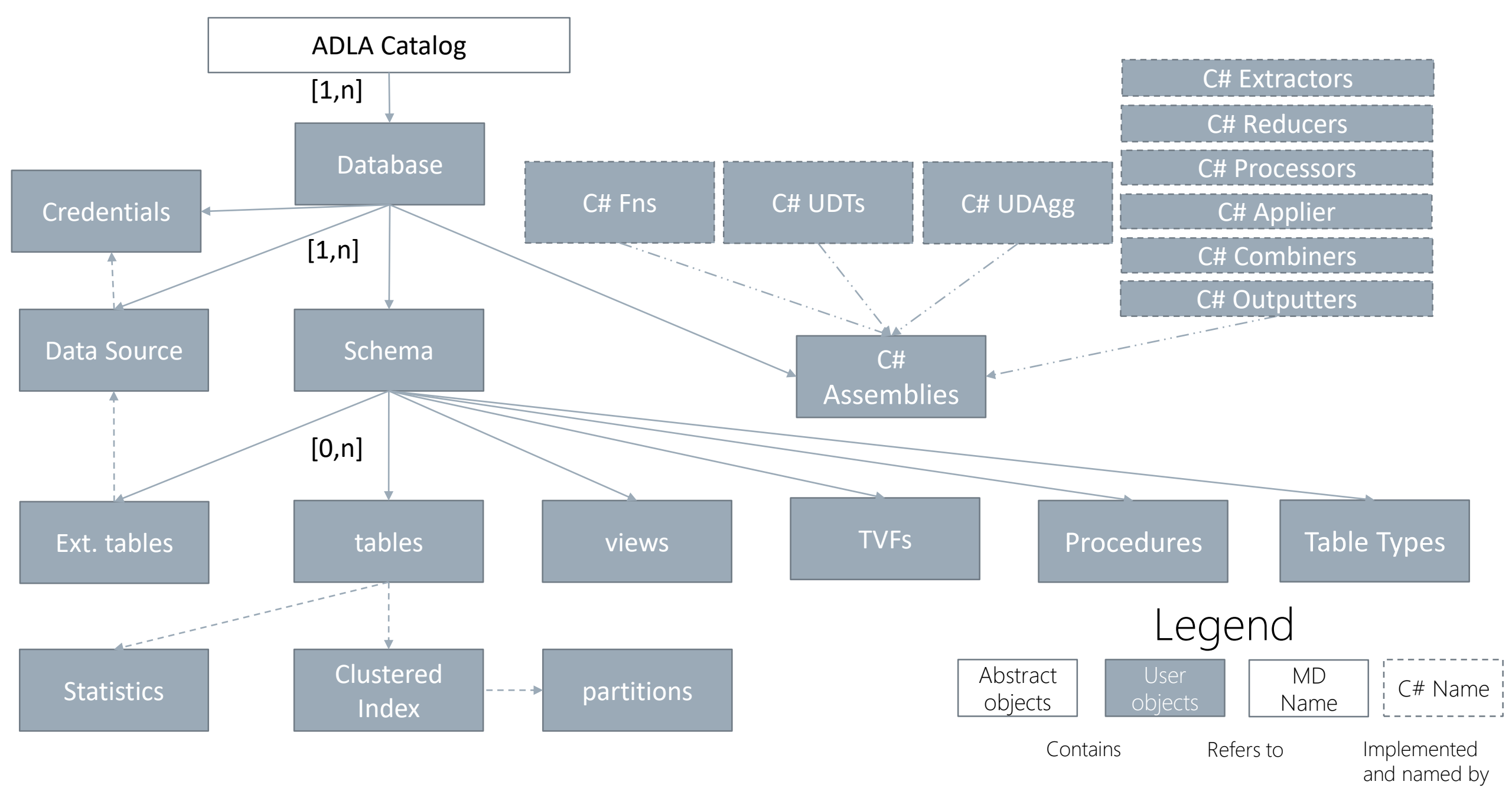
Database(s)

Schema(s)

Table(s)

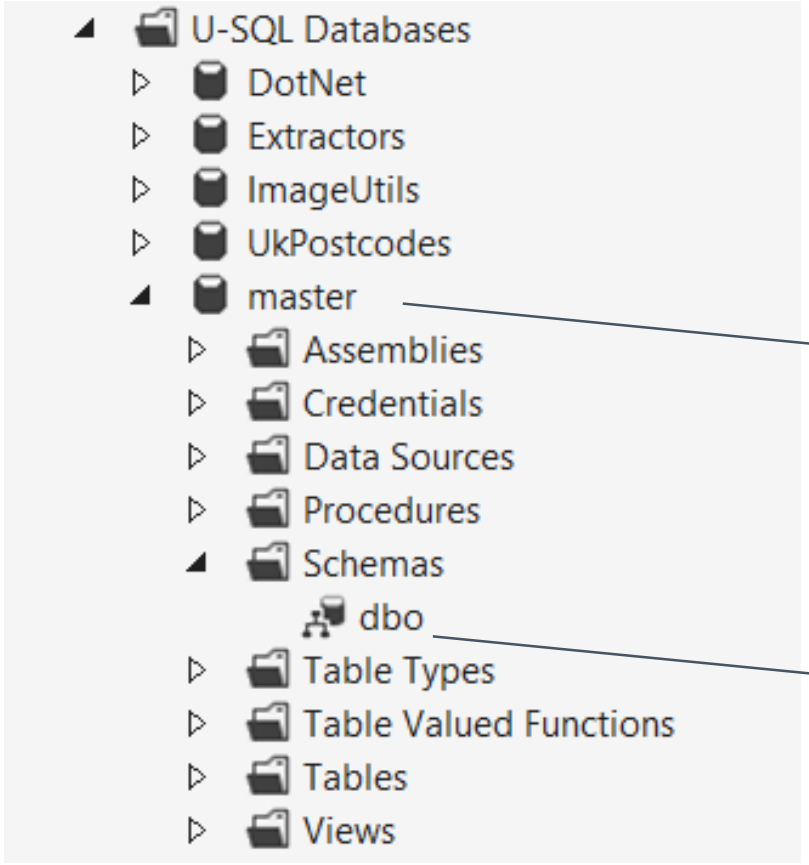
SQL INSERT

SQL SELECT (Aggregations,Windowing functions etc.)



Source: Microsoft\M Rys

U-SQL DATABASES AND SCHEMES




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

Default data base

Default schema

U-SQL TABLES

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

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

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 VIEWS and FUNCTIONS

VIEWS

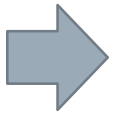
```
CREATE VIEW IF NOT EXISTS vCrimes
    AS
EXTRACT CrimeID string,
        Month string,
        Date DateTime,
        Input string
FROM @"\UKCrimesCities\{Date:yyyy}-
{Date:MM}\{Input}-street.csv"
USING Extractors.Csv(silent : false,
 skipFirstNRows : 1);
```

FUNCTIONS

```
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 Stored Procedures

```
DROP PROCEDURE IF EXISTS DemoDb.dm.SampleSP;
CREATE PROCEDURE DemoDb.dm.SampleSP(@startDate DateTime,
@endDate DateTime, @outputName string)
BEGIN

    @sample =
        SELECT *
        FROM(
            VALUES
            (
                1,
                new DateTime(2017, 01, 01, 05, 00, 00),
                new DateTime(2017, 01, 01, 06, 00, 00),
                100.00
            )
 )
AS T (id, begin, end, value);


    @rs =
        SELECT id,
               begin,
               end,
               value
        FROM @sample
        WHERE begin >= @startDate AND end <= @endDate;

    OUTPUT @rs
    TO @outputName
    USING Outputters.Csv();
   END;
```



```
DECLARE @startDate DateTime =
            new DateTime(2017, 01, 01, 05, 00, 00);
DECLARE @endDate DateTime =
            new DateTime(2017, 01, 01, 07, 00, 00);
DECLARE @outputName string = "sp_result.csv";
DemoDb.dm.SampleSP
(
    @startDate,
    @endDate,
    @outputName
);
```

U-SQL Extensions

- .NET Assemblies
  - API For Extractors, Reducers, Processors, Appliers, Combiners, Outputters
- Python
- R Language
- Cognitive API
  - Detecting Objects in Images (Tagging)
  - Detecting Emotion in Faces in Images
  - Detecting Text in Images (OCR)
  - Text Key Phrase Extraction
  - Text Sentiment Analysis

Source: Microsoft

U-SQL .NET Extentions

- .NET EXTENTIONS
  - C# Functions/Methods
  - C# UDTs
  - C# UDAggs
  - Extractors
  - Reducers
  - Processors
  - Appliers
  - Combiners
  - Outpputers

U-SQL .NET Extentions - METHODS

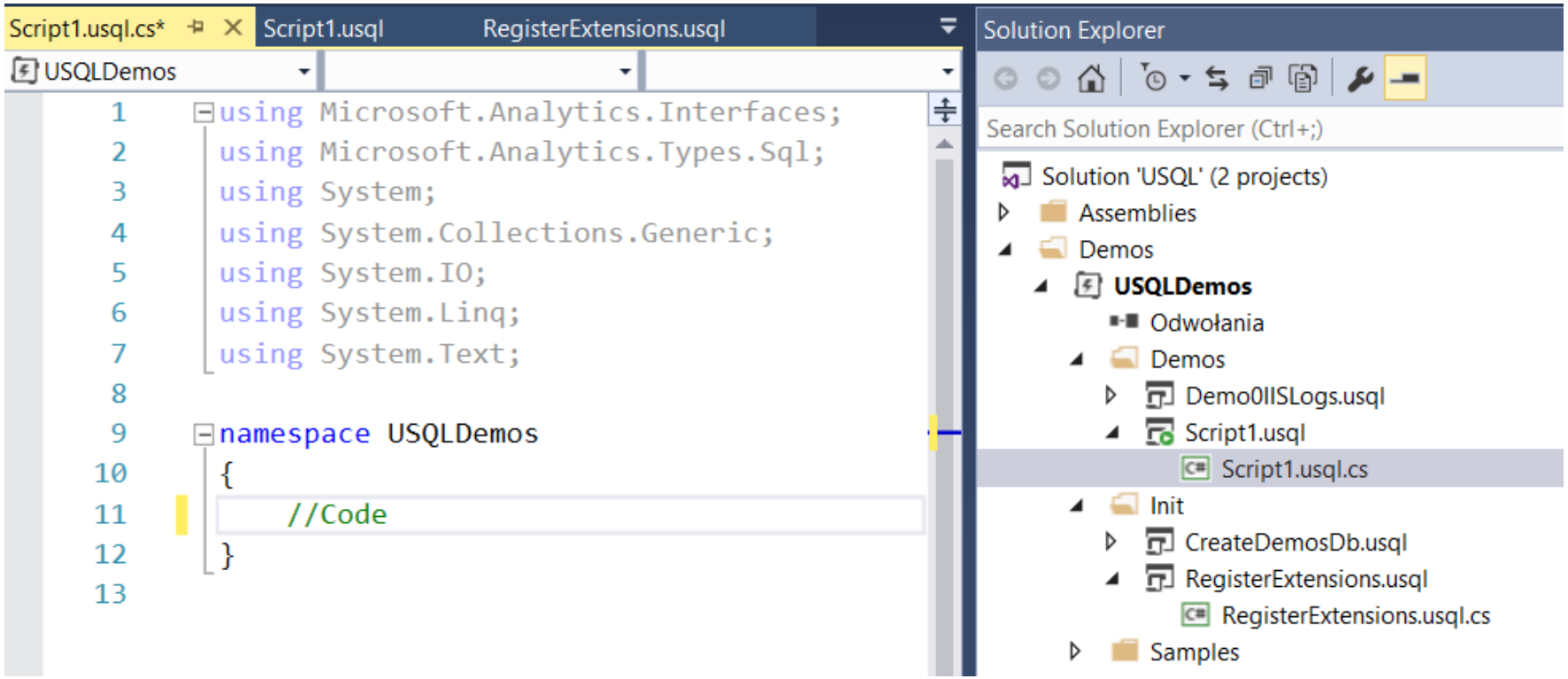
```
@distances =
    SELECT CrimeId,
           CityName,
           CrimeType,
           Year,
           Month,
           Gps.ComputeDistance
           (sLatitude,
            sLongitude,
            dLatitude,
            dLongitude) AS Distance
    FROM @merged;
```

```
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
        }
```

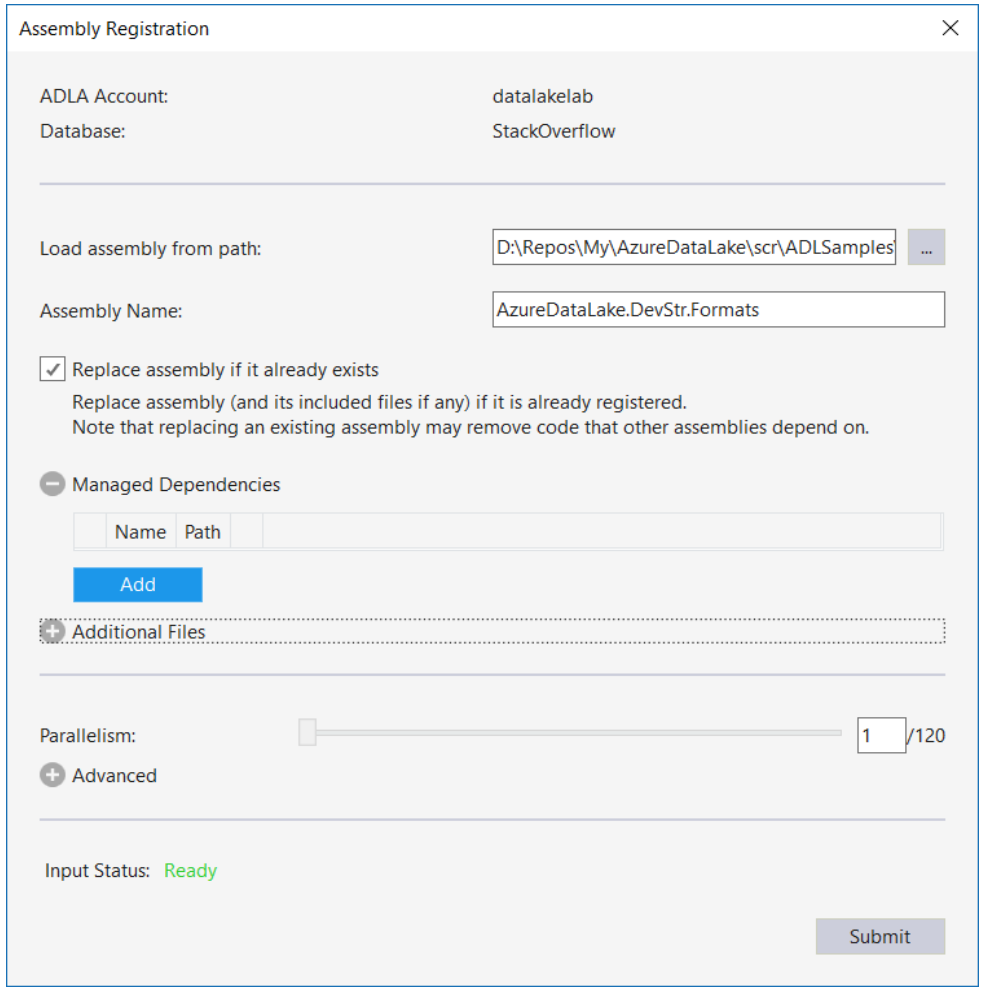
C# Method

U-SQL .NET Extentions

- C# Functions/Methods
  - Script .cs
  - .NET Assemblies



U-SQL REGISTERING ASSEMBLIES




```
DECLARE @AssemblyPath string =
@"Assemblies/";
DECLARE @AssemblyExt string =
@AssemblyPath+"ADLAExt.dll";
USE DemoDb;
DROP ASSEMBLY IF EXISTS ADLAExt;
CREATE ASSEMBLY ADLAExt FROM
@AssemblyExt;
```

U-SQL USING ASSEMBLIES

```
USE DATABASE [DemoDb];
REFERENCE ASSEMBLY [ADLAExt];

USING IpConverter = ADLAExt.Utils.IpConverter;

@ds =
    SELECT IpConverter.ToIp4Format(c_ip) AS Ip,
           date.Date AS Date
    FROM @iisLogs;
```

Use data base (optional)

Add reference to assembly

Create alias

Use method from assembly

U-SQL SYSTEM ASSEMBLIES

- Preloaded System Assemblies
  - mscorlib.dll
  - System.dll
  - System.Core.dll
  - System.Data.dll
  - Microsoft.Analytics.Interfaces.dll
  - Microsoft.Analytics.Types.dll

- Example:

```
REFERENCE SYSTEM ASSEMBLY [System.XML];
```

U-SQL Creating Extentions

- Methods
  - Static class + static method

```
public static class IpConverter
   {
       public static string ToIp4Format(string ip)
       {
           return IPAddress.Parse(ip).MapToIPv4().ToString();
       }
   }
```

- Extractors, Reducers, Processors …
  - New project type: Class Library
  - (For U-SQL Applications)
  - Interfaces
    - IExtractors, IReducers, IProcessors …

```
public class NameReverseProcessor : IProcessor
    {
        public override IRow Process(IRow input,
                                     IUpdatableRow output)
        {
            var s = input.Get<string>("name");
            output.Set<string>("reversed", Reverse(s));
            return output.AsReadOnly();
        }

        private static string Reverse(string s)
        {
            char[] charArray = s.ToCharArray();
            Array.Reverse(charArray);
            return new string(charArray);
        }
    }
```

U-SQL EXTRACTOR

```
[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();
           }
       }
   }
```

WHOLE FILE

RETURN CONTENT IN FIRST COLUMN

```
@rs =
    EXTRACT content byte[],
            fileName string
    FROM @imgFiles
    USING new BinaryExtractor();
```

DEMO(s)

- CreateDemosDb
- Create and Call Stored Procedure
- ADLAExt (Project)
- RegisterExtentions
- Demos_001\Demo0IILogs
- Demos_001\Demo1Log4Net

U-SQL PROCESSORS

```
USE DATABASE [DemoDb];
REFERENCE ASSEMBLY [ADLAExt];

USING NameReverseProcessor =
ADLAExt.Processors.NameReverseProcessor;

@sample =
    SELECT *
    FROM(
        VALUES
        (
            "ABC"
        ),
        (
            "DEF"
        ),
        (
            "GHI"
        )
                        ) AS T(name);

@reversed =
    PROCESS @sample
    PRODUCE name,
            reversed string
    READONLY name
    REQUIRED name
    USING new NameReverseProcessor();

OUTPUT @reversed
TO "reversed.csv"
USING Outputters.Csv();
```

```
public class NameReverseProcessor : IProcessor
   {
       public override IRow Process(IRow input, IUpdatableRow output)
       {
           var s = input.Get<string>("name");
           output.Set<string>("reversed", Reverse(s));
           return output.AsReadOnly();
       }

       private static string Reverse(string s)
       {
           char[] charArray = s.ToCharArray();
           Array.Reverse(charArray);
           return new string(charArray);
       }
   }
```

U-SQL REDUCERS

Id	begin	end	value
1	2017-01-01 05:00:00	2017-01-01 06:00:00	100
1	2017-01-01 06:01:00	2017-01-01 07:00:00	200
1	2017-01-01 08:00:00	2017-01-01 09:00:00	900
2	2017-01-01 06:01:00	2017-01-01 07:00:00	2
2	2017-01-01 07:01:00	2017-01-01 09:01:00	9

REDUCER

Id	begin	end	value
1	2017-01-01 05:00:00	2017-01-01 07:00:00	300
1	2017-01-01 08:00:00	2017-01-01 09:00:00	900
2	2017-01-01 05:00:00	2017-01-01 09:00:00	12

U-SQL REDUCERS

Id	begin	end	saidi
1	2017-01-01 05:00:00	2017-01-01 06:00:00	100
1	2017-01-01 06:01:00	2017-01-01 07:00:00	200
1	2017-01-01 08:00:00	2017-01-01 09:00:00	900
2	2017-01-01 06:01:00	2017-01-01 07:00:00	2
2	2017-01-01 07:01:00	2017-01-01 09:01:00	9

```
@results =
    REDUCE @sample
    PRESORT begin
    ON id
    PRODUCE id int,
            begin DateTime,
            end DateTime,
            saidi double
    READONLY id
    REQUIRED begin,
             end,
             saidi
    USING new SaidiRangeReducer(180);
```

PRESORT

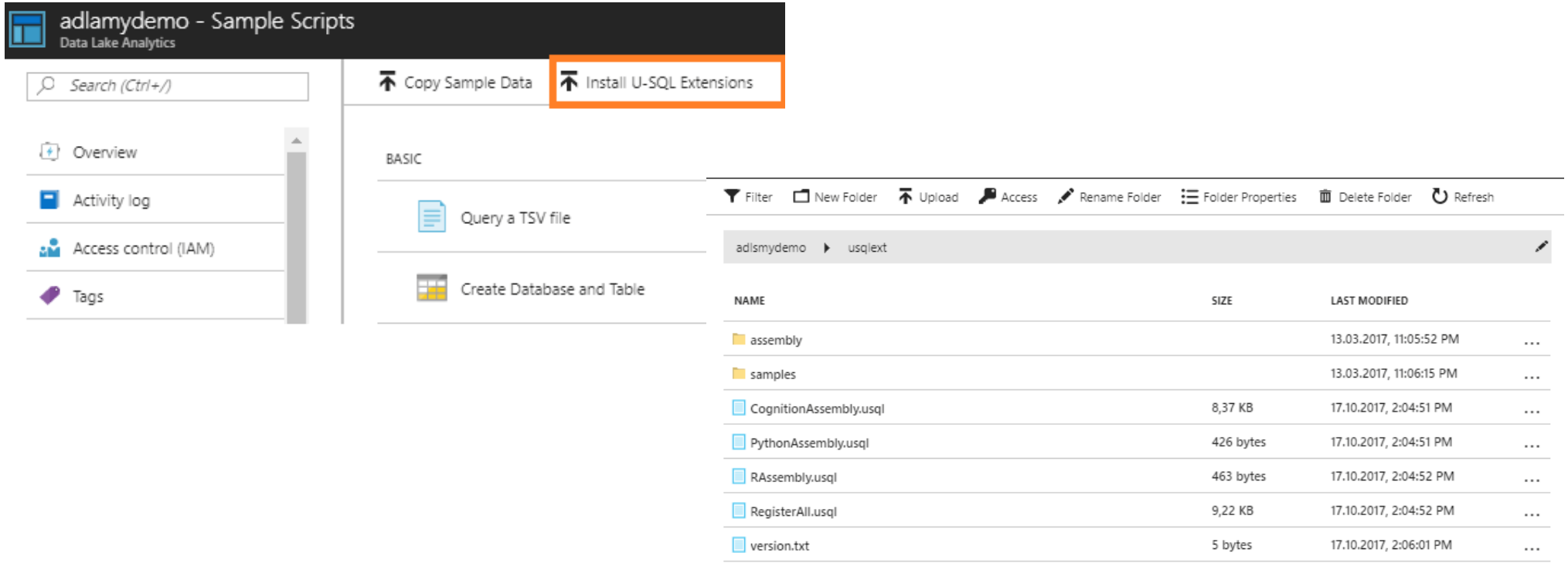
GROUP BY

```
public override IEnumerable<IRow> Reduce(IRowset input, IUpdatableRow output)
        {
            // Init aggregation values
            var firstRowProcessed = false;
            var begin = DateTime.MinValue;
            var end = DateTime.MinValue;
            var saidivalue = 0.0;
            // requires that the reducer is PRESORTED on begin and READONLY on the reduce key.
            foreach (var row in input.Rows)
            {
                if (!firstRowProcessed)
                {
                    firstRowProcessed = true;
                    begin = row.Get<DateTime>(BeginColName);
                    end = row.Get<DateTime>(EndColName);
                    saidivalue = row.Get<double>(SaidiColName);
                }
                else
                {
                    var b = row.Get<DateTime>("begin");
                    var e = row.Get<DateTime>("end");
                    var tmpsaidi = row.Get<double>("saidi");
                    if ((b - end).TotalSeconds <= _maxDuration)
                    {
                        saidivalue += tmpsaidi;
                    }
                    else
                    {
                        output.Set<double>("saidi", saidivalue);
                        output.Set<DateTime>("begin", begin);
                        output.Set<DateTime>("end", end);

                        yield return output.AsReadOnly();
                        saidivalue = tmpsaidi;
                        begin = b;
                    }
                    end = e;

                }
            }
            output.Set<DateTime>("begin", begin);
            output.Set<DateTime>("end", end);
            output.Set<double>("saidi", saidivalue);
            yield return output.AsReadOnly();
        }
    }
```

DEMO(s)

- Samples\Processor
- Demos_002\DemoColorHist
- Samples\Reducer

U-SQL Python, R Language, Cognitive



U-SQL Python, R Language, Cognitive

- Register Assemblies
  - CognitionAssembly.usql
  - PythonAssembly.usql
  - RAssembly.usql
  - **RegisterAll.usql**

```
CREATE DATABASE IF NOT EXISTS master;
USE DATABASE master;

DROP ASSEMBLY IF EXISTS [ExtPython];
CREATE ASSEMBLY IF NOT EXISTS [ExtPython]
FROM @"/usqlext/assembly/python/ExtPy.dll"
WITH ADDITIONAL_FILES =
(
    @"/usqlext/assembly/python/ExtPy.pdb",
    @"/usqlext/assembly/python/UsqlPythonInvokePackage.zip",
    @"/usqlext/assembly/python/UsqlPythonDeployPackage.zip",
    @"/usqlext/assembly/python/version.python"
);
```

```
CREATE DATABASE IF NOT EXISTS master;
USE DATABASE master;

DROP ASSEMBLY IF EXISTS ExtR;
CREATE ASSEMBLY IF NOT EXISTS ExtR
FROM @"/usqlext/assembly/R/ExtR.dll"
WITH ADDITIONAL_FILES = (
    @"/usqlext/assembly/R/ExtR.pdb",
    @"/usqlext/assembly/R/DynamicInterop.dll",
    @"/usqlext/assembly/R/RDotNet.dll",
```

U-SQL Python, R Language

- BASED ON REDUCER(s)

```
@PyOutput =
    REDUCE @Extended
    ON Par
    PRODUCE Par int,
            SepalLength double,
            SepalWidth double,
            PetalLength double,
            PetalWidth double,
            Species string,
            SepalRatio double,
            PetalRatio double
    USING new Extension.Python.Reducer(pyScript : @myPyScript);

@ROutput =
    REDUCE @PyOutput
    ON Par
    PRODUCE Par,
            RowId int,
            ROutput string
    READONLY Par
    USING new Extension.R.Reducer(command : @myRScript,
                                  rReturnType:"charactermatrix",
                                     stringsAsFactors:true);
```

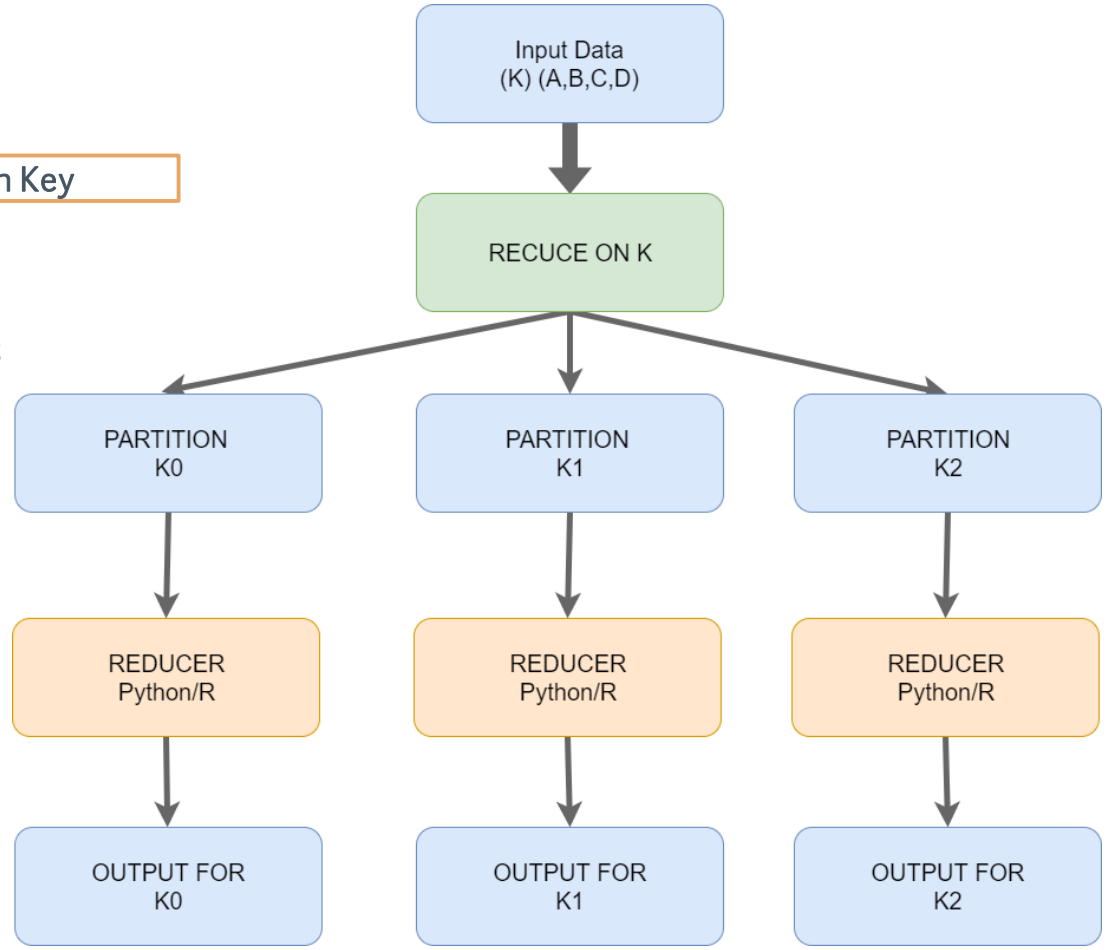
Python Reducer

Python Reducer

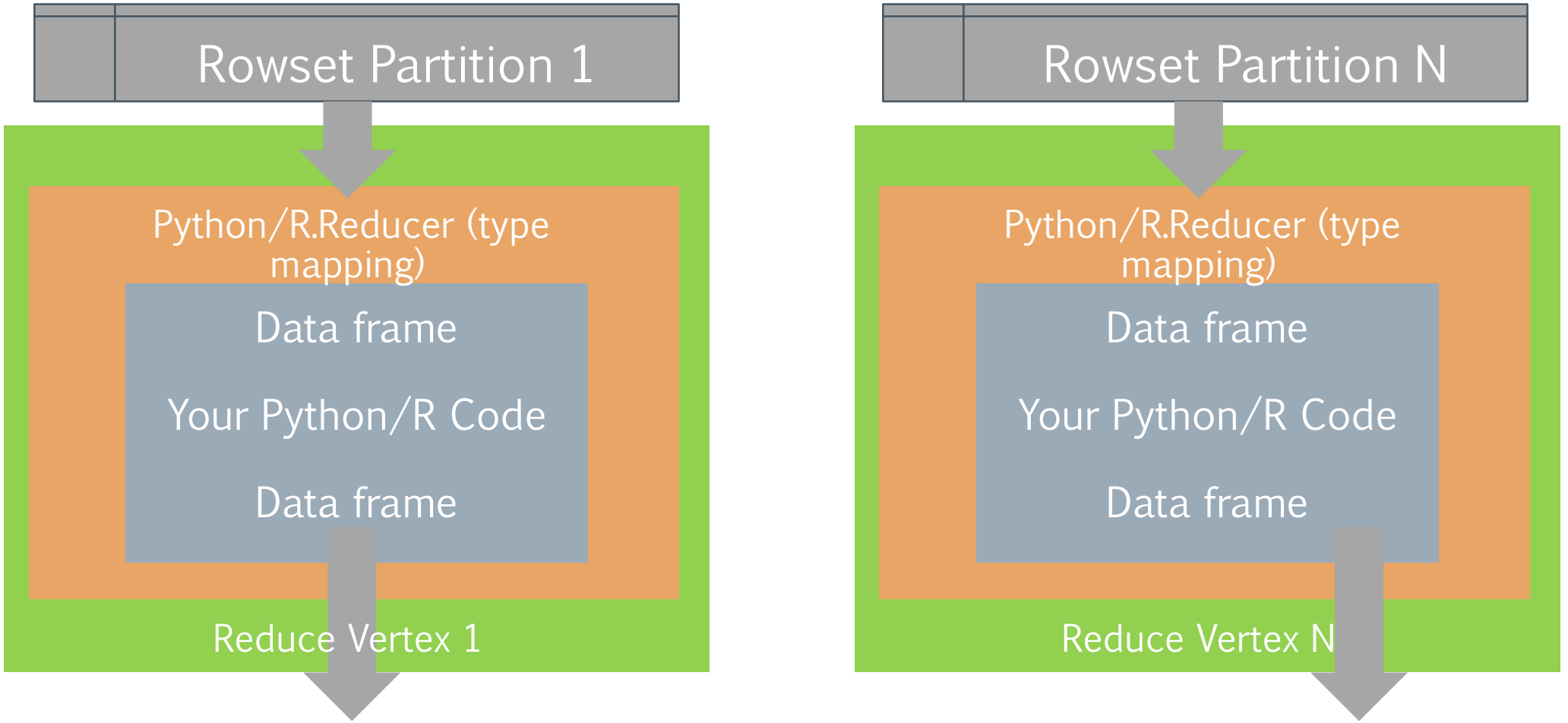
U-SQL Python, R Language

```
@ROutput =
    REDUCE @PyOutput
    ON Par
    PRODUCE Par,
            RowId int,
            ROutput string
    READONLY Par
    USING new Extension.R.Reducer(command : @myRScript);
```

Partition Key



U-SQL Python, R Language



U-SQL Python

- Datatypes
  - String and numeric columns from U-SQL are converted as-is between Pandas and U-SQL
  - U-SQL Nulls are converted to and from Pandas NA values
- Schemas
  - Index vectors in Pandas are not supported in U-SQL.
  - U-SQL datasets cannot have duplicate column names
  - U-SQL datasets column names that are not strings.
- Python Versions
  - Only Python 3.5.1 (compiled for Windows) is supported.

U-SQL Python

- Modules
  - All the standard Python modules are included.
- Additional Python modules
  - pandas
  - numpy
  - Numexpr
- **Python Custom Module**

U-SQL Custom Module

› U-SQL DEPLOY RESOURCE

```
DEPLOY RESOURCE
"/Samples/Data/Python/modules.zip";
DECLARE @myScript = @"
import sys
sys.path.insert(0, 'modules.zip')
import mymodule

def usqlml_main(df):
    del df['number']
    df['hello_world'] =
str(mymodule.hello_world)
    return df
";
# demo module
hello_world = "Hello World! This is code from a custom module"
```

U-SQL R

```
// R script to run
// Put commas between commands if rReturnType:"charactermatrix", no commas
otherwise
DECLARE @myRScript = @"
require(RevoScaleR),
rxDTree(Species ~ SepalLength, data=inputFromUSQL)
";

@InputData =
    EXTRACT SepalLength double,
            SepalWidth double,
            PetalLength double,
            PetalWidth double,
            Species  string
    FROM @IrisData
    USING Extractors.Csv();

@ExtendedData =
    SELECT 0 AS Par,
           *
    FROM @InputData;

@RScriptOutput = REDUCE @ExtendedData ON Par
PRODUCE Par, RowId int, ROutput string
READONLY Par
USING new Extension.R.Reducer(command:@myRScript,
rReturnType:"charactermatrix", stringsAsFactors:true);

OUTPUT @RScriptOutput TO @OutputFileModelSummary USING Outputters.Tsv();
```

R script with RevoScaleR

R Reducer

U-SQL Python/R Limitations

- Python Input and Output size limitations
  - the total size for the input and output cannot exceed 6 GB (max input dataframe size < 2 GB)
- R Functional limitations
  - The R Engine can't be instantiated twice in the same process.
- R Input and Output size limitations
  - Because the input and output DataFrames must exist in memory in the R code, the total size for the input and output cannot exceed 500 MB.
- Only R 3.2.2 is supported.

U-SQL Cognitive

- The following cognitive capabilities are available:
  - Imaging: Detect faces
  - Imaging: Detect emotion
  - Imaging: Detect objects (tagging)
  - Imaging: OCR (optical character recognition)
  - Text: Key Phrase Extraction
  - Text: Sentiment Analysis

U-SQL Cognitive

```
REFERENCE ASSEMBLY ImageCommon;
REFERENCE ASSEMBLY ImageTagging;

DECLARE @input string = @"D:\AppData\BIGDATA\Images\{FileName}";

///Extract images
@imgs =
    EXTRACT FileName string,
            ImgData byte[]
    FROM @input
    USING new Cognition.Vision.ImageExtractor();

//// Extract the number of objects on each image and tag them
@objects =
    PROCESS @imgs
    PRODUCE FileName,
            NumObjects int,
            Tags string
    READONLY FileName
    USING new Cognition.Vision.ImageTagger();

//// Split tags - convert tag1;tag2;tag3 to ARRAY[]
{"tag1",tag2",tag3"}
@objects =
    SELECT FileName,
           new SQL.ARRAY<string>(Tags.Split(new char[]{';'},
StringSplitOptions.RemoveEmptyEntries)) AS ObjTags
    FROM @objects;
```

```
//// Split tags - convert tag1;tag2;tag3 to ARRAY[] {"tag1",tag2",tag
@objects =
    SELECT FileName,
           new SQL.ARRAY<string>(Tags.Split(new char[]{';'},
StringSplitOptions.RemoveEmptyEntries)) AS ObjTags
    FROM @objects;

//// Transform to table:
///  FileName tag1
///  FileName tag2
///  FileName tag3

@objects =
    SELECT o.FileName,
           t.Tag
    FROM @objects AS o
         CROSS APPLY
            EXPLODE(o.ObjTags) AS t(Tag);
//// Find files with car
@carObjects =
    SELECT DISTINCT FileName
    FROM @objects AS o
    WHERE o.Tag.Contains("car") OR o.Tag.Contains("auto") OR
o.Tag.Contains("vehicle");

OUTPUT @carObjects
TO @"/my/cognition/output/cars.csv"
USING Outputters.Csv();

OUTPUT @objects
TO @"/my/cognition/output/objects.csv"
USING Outputters.Csv();
```

DEMO(s)

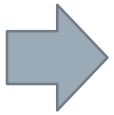
- RegisterCognitiveAssemblies
- Register Python and R Extentions
- Demos_003\PythonModules
- Demos_001\Demo2CognitiveTagger
- Demos_001\Demo3CognitiveFaces

U-SQL Packages

- PACKAGE(s)
  - Allows bundling of commonly used together U-SQL assamblies, variables and resources
  - A package declaration can consist of:
    - using statement
    - declare statements
    - import other package(s) statements
    - deploy statement
    - IF statement

U-SQL Packages

```
DROP PACKAGE IF EXISTS
DemoDb.dm.LoadMyAssemblies;
CREATE PACKAGE
DemoDb.dm.LoadMyAssemblies(@all string =
"yes")
BEGIN
 IF @all == "yes" THEN
    EXPORT SYSTEM ASSEMBLY [System.Xml];
    EXPORT ASSEMBLY [ADLAExt];
    EXPORT @listOfAssemblies
="System.Xml;ADLAExt";
 ELSE
    EXPORT ASSEMBLY [ADLAExt];
    EXPORT @listOfAssemblies ="ADLAExt";
 END;
END;
```



```
IMPORT PACKAGE DemoDb.dm.LoadMyAssemblies("no") AS mypackage;
USING SaidiRangeReducer = ADLAExt.Reducers.SaidiRangeReducer;

@results =
    SELECT *
    FROM(
        VALUES
        (
            mypackage.@listOfAssemblies
        )
        ) AS T(assemblies);

OUTPUT @results
TO "package.csv"
USING Outputters.Csv();
```

	assemblies
1	"ADLAExt"

DEMO(s)

- Samples\CreatePackage
- Samples\ImportPackage

usql.databases

U-SQL CATALOG VIEWS

- Currently **not available** in the local run environment
- Catalog Views
  - usql.databases
  - usql.schemas
  - usql.objects
  - usql.tables
  - usql.views
  - usql.functions
  - usql.types
  - usql.columns
  - usql.index_columns
  - usql.stats
  - usql.stats_columns
  - usql.distributions
  - usql.partitions
  - usql.partition_range_values
  - usql.partition_parameters

π

U-SQL CATALOG VIEWS

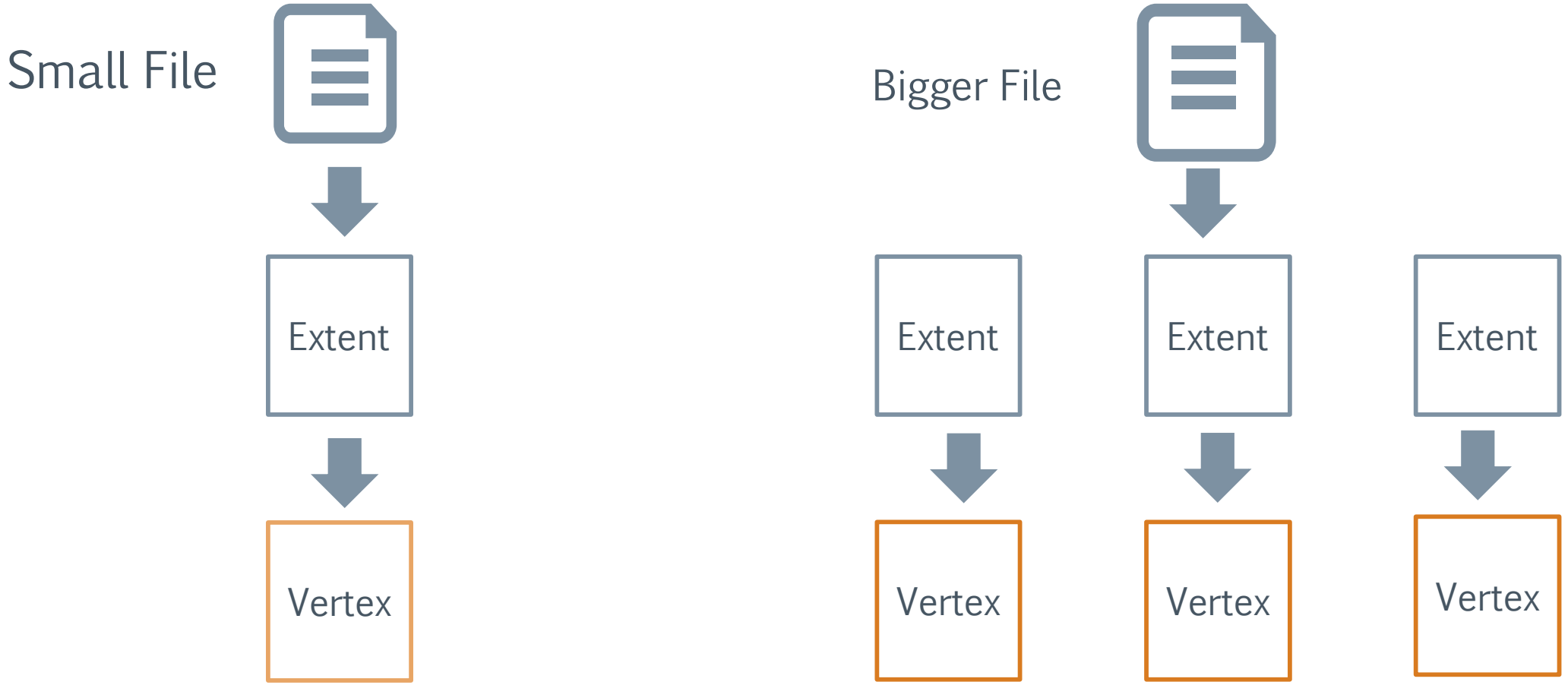
```
@res =
   SELECT "[" + db.name + "].[" + s.name + "].[" + t.name + "]"
AS table_name,
          c.name AS col_name,
          c.column_id AS col_pos,
          ct.qualified_name AS col_type,
          c.max_length == - 1 ?
            ct.qualified_name == "System.String" ?
              128 * 1024
            : ct.qualified_name == "System.Byte[]" ?
                4 * 1024 * 1024
              : - 1
          : c.max_length AS col_max_length
   FROM usql.databases AS db
   JOIN usql.schemas AS s ON db.database_id_guid ==
s.database_id_guid
   JOIN usql.tables AS t ON s.schema_id_guid == t.schema_id_guid
   JOIN usql.columns AS c ON c.object_id_guid == t.object_id_guid
   JOIN usql.types AS ct ON c.type_id_guid == ct.type_id_guid;

 OUTPUT @res
 TO "/output/tableinfo.csv"
 ORDER BY table_name, col_pos
 USING Outputters.Csv(outputHeader : true);
```

DEMO(s)

- Samples\CatalogView

Azure Data Lake Partitioning

Azure Data Lake Processing

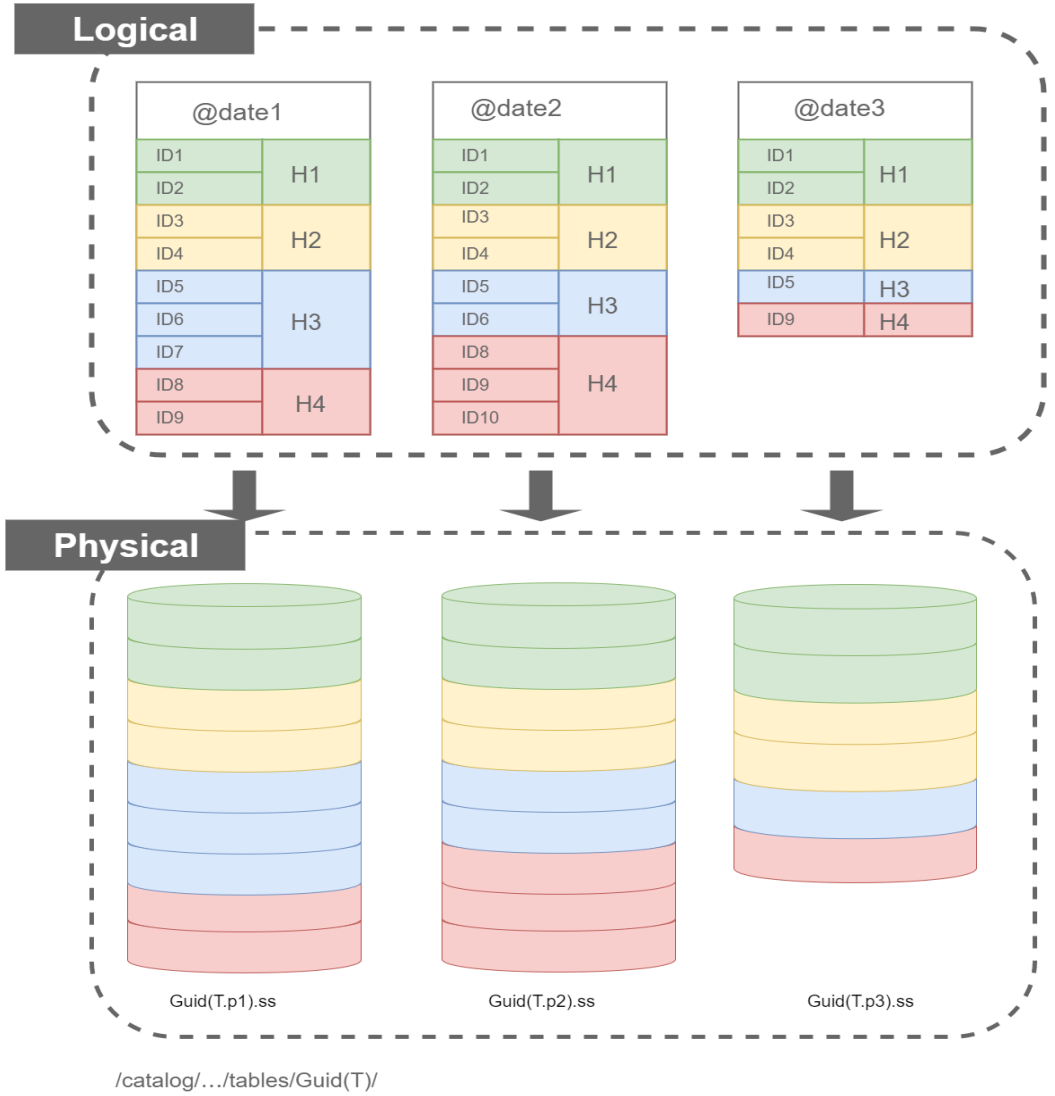


Azure Data Lake Partitioning

- There are two types of partitioning supported by the Azure Data Lake:
  - Horizontal partitioning – Clustering and Distribution Schema
  - Vertical partitioning - Partitioning

U-SQL TABLES

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



U-SQL PARITIONING - BENEFITS

- **Clustering**
  - Design for most frequent/costly queries
  - Manage data skew in distribution bucket
  - Provide locality of same data values
  - Provide seeks and range scans for query predicates (index lookup)

- **Distribution**
  - Design for most frequent/costly queries
  - Manage data skew in partition/table
  - Manage parallelism in querying (by number of distributions)
  - Manage minimizing data movement in joins
  - Provide distribution seeks and range scans for query predicates (distribution bucket elimination)

Source: Azure Data Lake Analytics U-SQL Performance –M. Rys

U-SQL PARITIONING - BENEFITS

- **Partitions**
  - Partitions are addressable
  - Enables finer-grained data lifecycle management at partition level
  - Manage parallelism in querying by number of partitions
  - Query predicates provide partition elimination
  - Predicate has to be constant-foldable

- Scenarios
- Managing large amounts of incrementally growing structured data
  - Example: Keep adding daily data for years.
- Queries with strong locality predicates
  - Point-in-time, for specific market etc
- Managing windows of data
  - provide data for last x months for processing

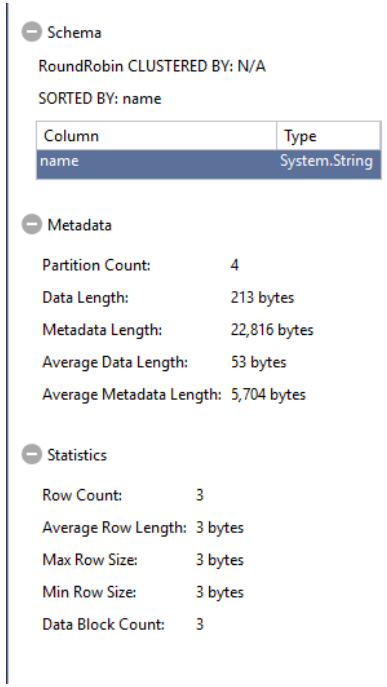
Source: Azure Data Lake Analytics U-SQL Performance –M. Rys

U-SQL TABLES DISTRIBUTION SCHEMA

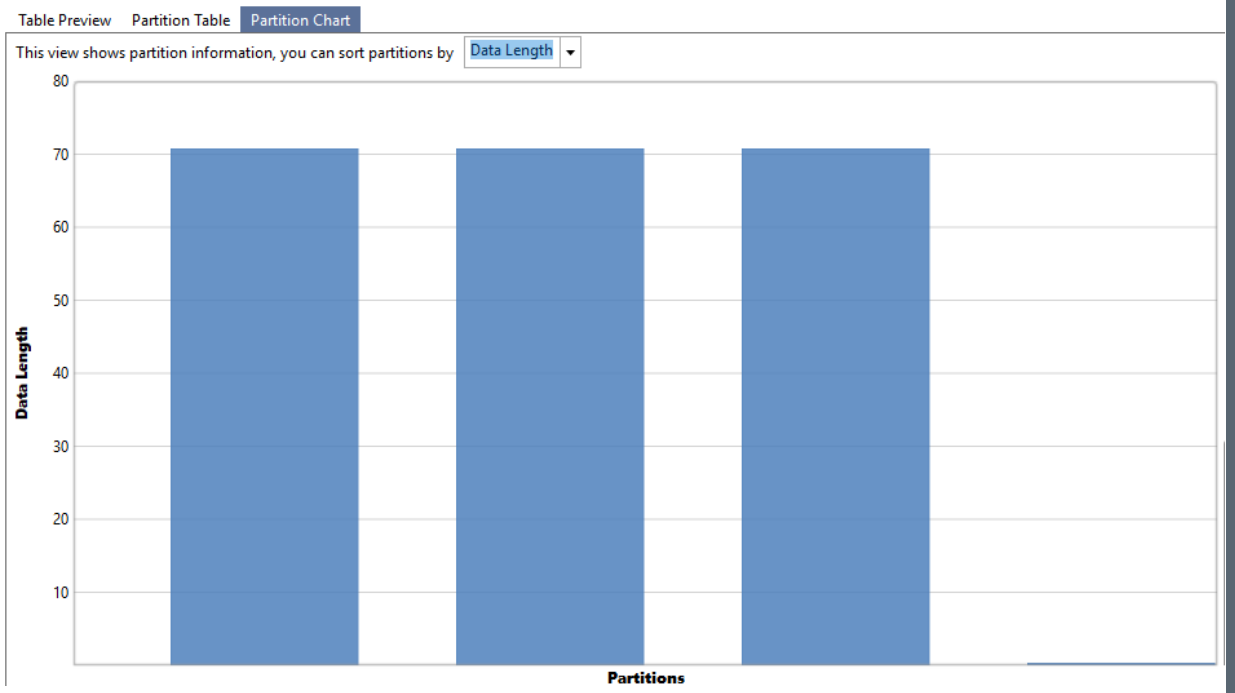
- Currently U-SQL supports four distribution schemes:
  - RANGE
    - Based on a set of ordered columns
  - HASH
    - Based on a set of columns
  - DIRECT HASH
    - Based on single column of an integral type
  - ROUND ROBIN
    - ROUND ROBIN assigns rows to distributions individually in round robin fashion without reference to the values they contain. Each distribution should have approximately the same number of rows.

U-SQL PARITIONING – DISTRIBUTION(s)

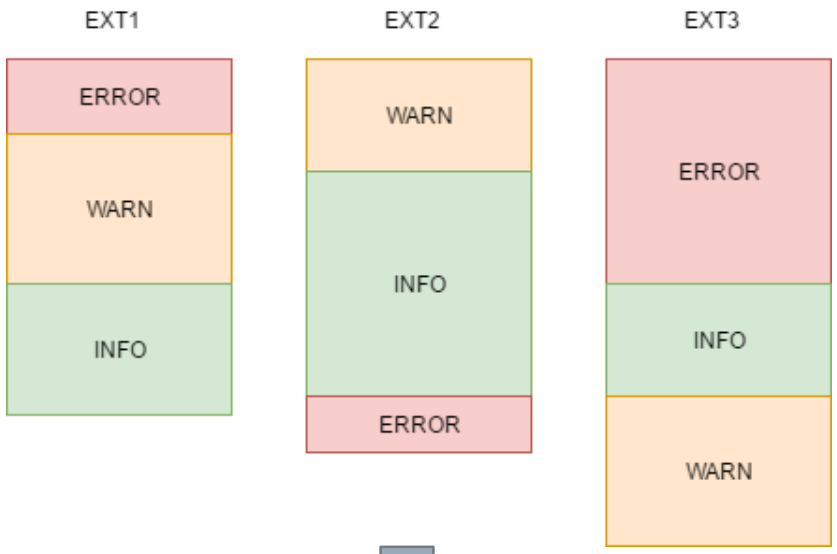
```
USE DemoDb;
@sample =
    SELECT *
    FROM(
        VALUES
        (
            "ABC"
        ),
        (
            "DEF"
        ),
        (
            "GHI"
        )
                        ) AS
T(name);

CREATE TABLE IF NOT EXISTS
dm.DistSchema
(
    INDEX clx_DistSchema
    CLUSTERED(name ASC)
    DISTRIBUTED BY
    ROUND ROBIN
    INTO 4
) AS
SELECT name
FROM @sample;
```

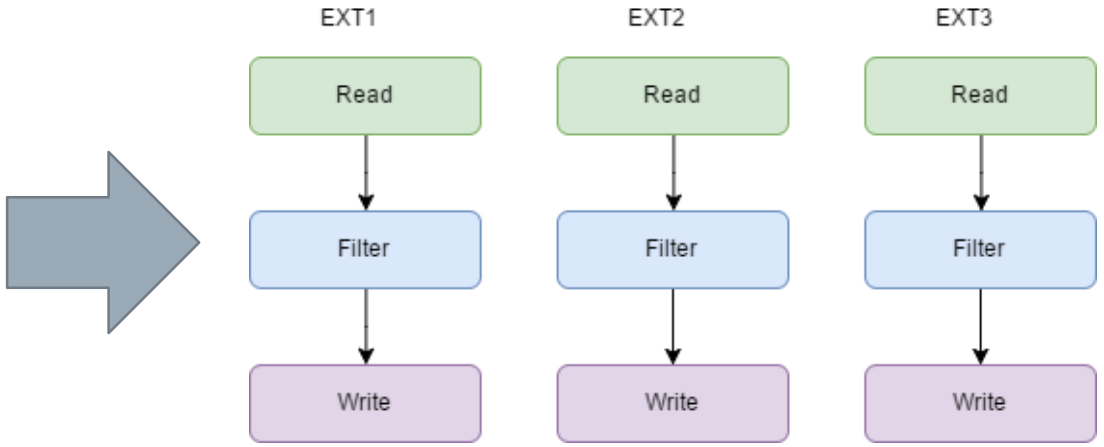


DISTRIBUTED SCHEMA



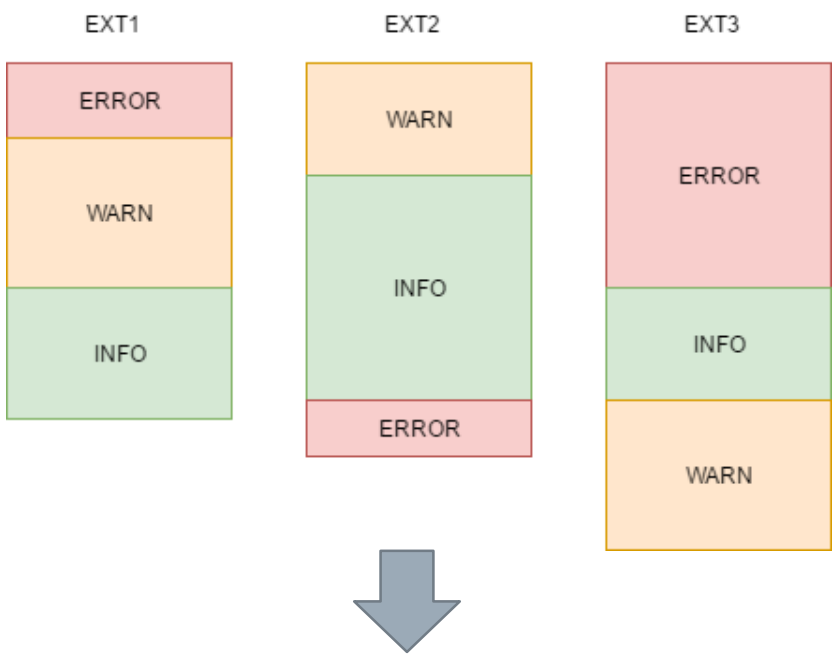
U-SQL PARITIONING

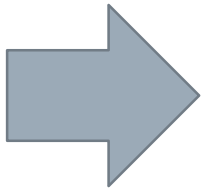


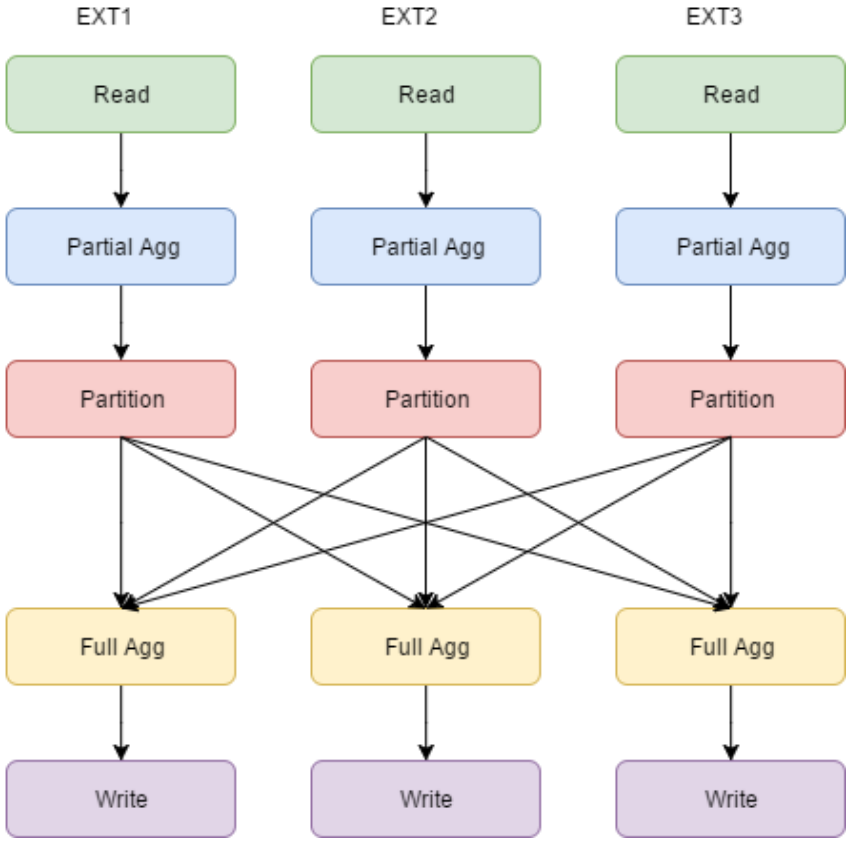



```
@ds =
    SELECT *
    FROM @logs
    WHERE Level == "ERROR";
```

U-SQL PARITIONING

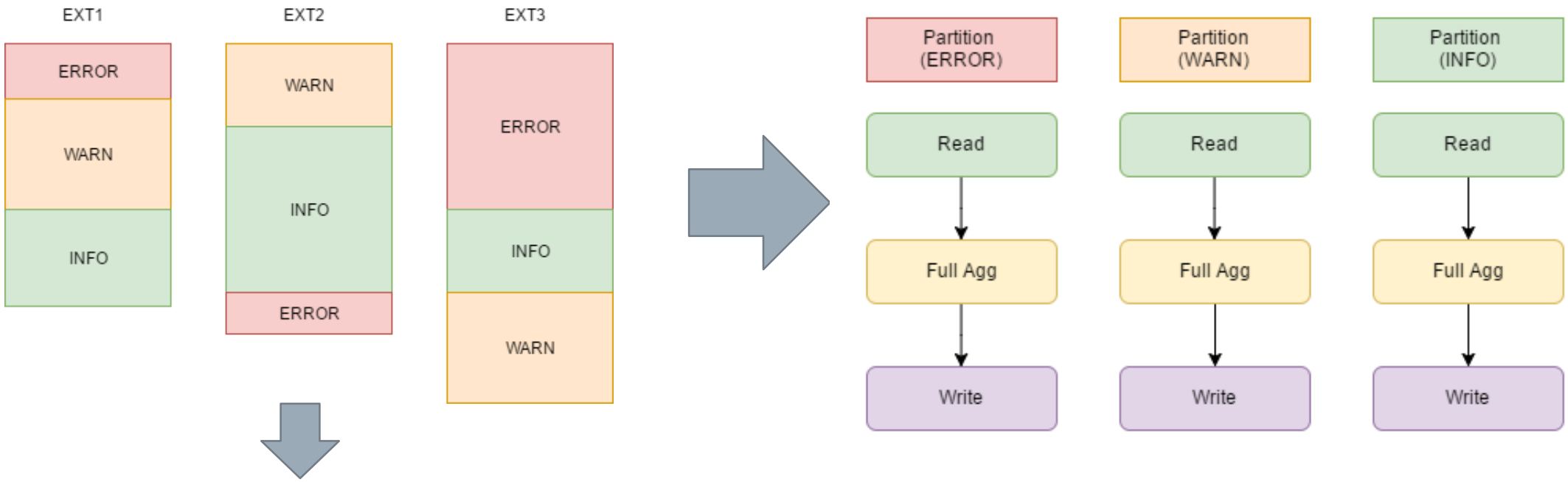







```
@ds =
    SELECT Level,
           COUNT( * ) AS Total
    FROM @logs
    GROUP BY Level;
```

U-SQL PARITIONING




```
@ds =
    SELECT Level,
           COUNT( * ) AS Total
    FROM @logs
    GROUP BY Level;
```

DEMO(s)

- Samples\TableDistributionScheme
- Samples\Partitions
  - CreateTable
  - TruncateTable
  - CreatePackagePartitions
  - StaticPartionLoad
  - GetData
  - TruncateTable
  - DynamicPartitionLoad
  - GetData
  - DropPartion
  - TruncateTable
  - DynamicPartitionLoad
  - GetBadData

Azure Data Lake Analytics Runtime

Data Lake Analytics Pricing

JobCost = (minutes x ADLUnits x ADLU Cost) /60

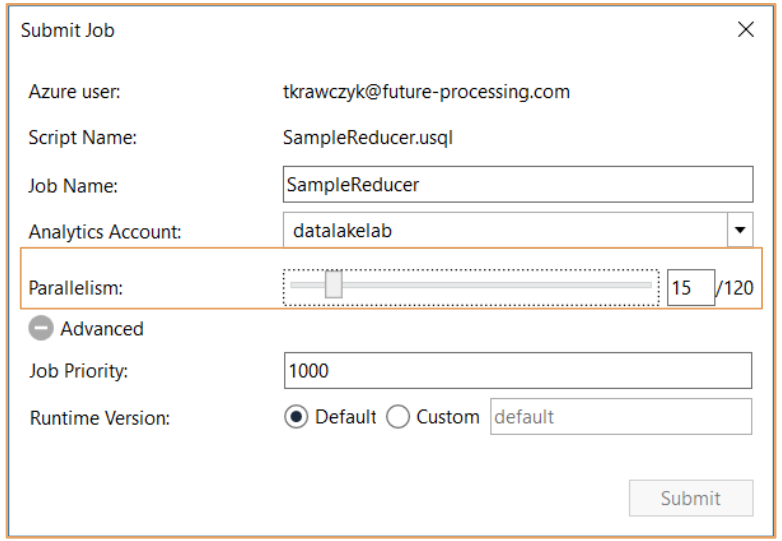
(ADLU Cost: 1h= 1,69 €)

1 ADLAU ~=

A VM with 2 cores

and 6 GB of memory

Parallelism N = N ADLAUs



Data Lake Analytics

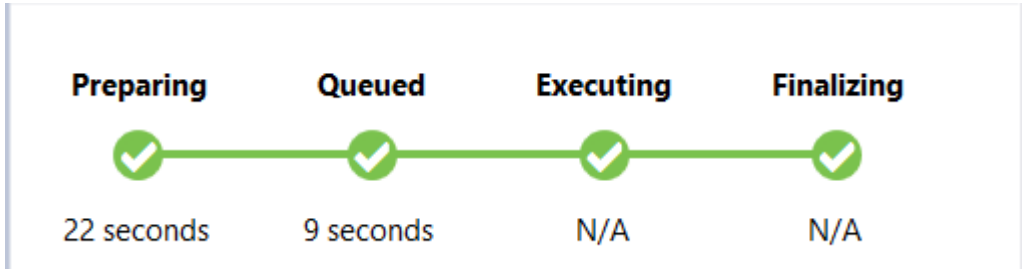
› ADLUA (VERTEX) = Virtual Machine

```
public static string GetVMInfo()
        {
            var sb = new StringBuilder();
            var myManagementClass = new
                ManagementClass("Win32_PerfRawData_Counters_HyperVDynamicMemoryIntegrationService");
            var myManagementCollection =
                myManagementClass.GetInstances();
            var myProperties =
                myManagementClass.Properties;
            var myPropertyResults =
                new Dictionary<string, object>();

            foreach (var obj in myManagementCollection)
            {
                foreach (var myProperty in myProperties)
                {
                    myPropertyResults.Add(myProperty.Name,
                        obj.Properties[myProperty.Name].Value);
                }
            }

            foreach (var myPropertyResult in myPropertyResults)
            {
                var item = $"{myPropertyResult.Key}:{myPropertyResult.Value}";
                sb.AppendLine(item);
            }
            return sb.ToString();
        }
```

```
Processor:Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz;
Microsoft Windows Server 2012 R2 Datacenter
OSArchitecture:64-bit
SystemDevice:\Device\HarddiskVolume1
SystemDirectory:C:\Windows\system32
SystemDrive:C:
TotalSwapSpaceSize:
TotalVirtualMemorySize:8314420
TotalVisibleMemorySize:2096692
```

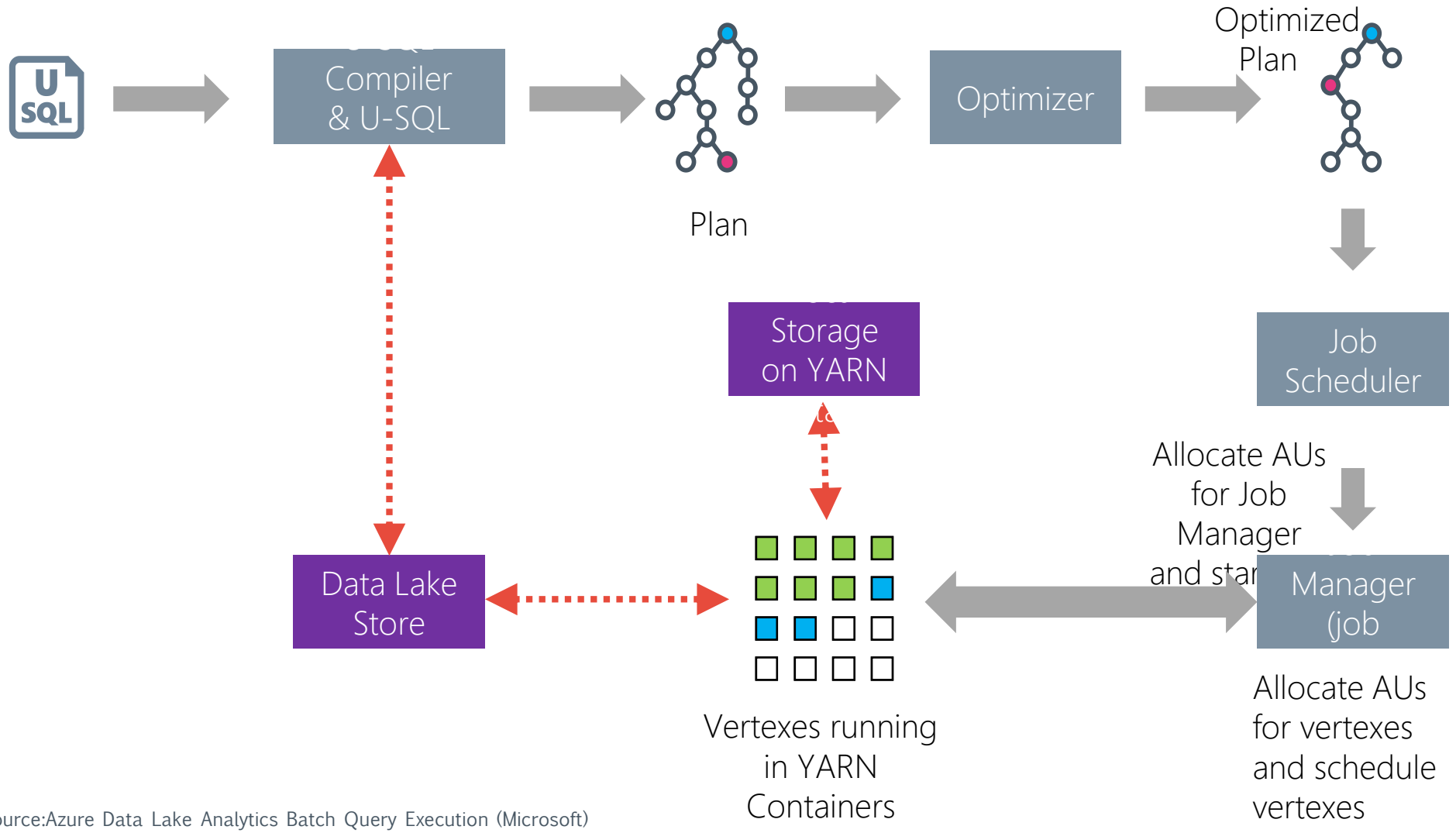
U-SQL Job Lifecycle States

```
@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;
```



Result = Succeeded | Failed | Cancelled

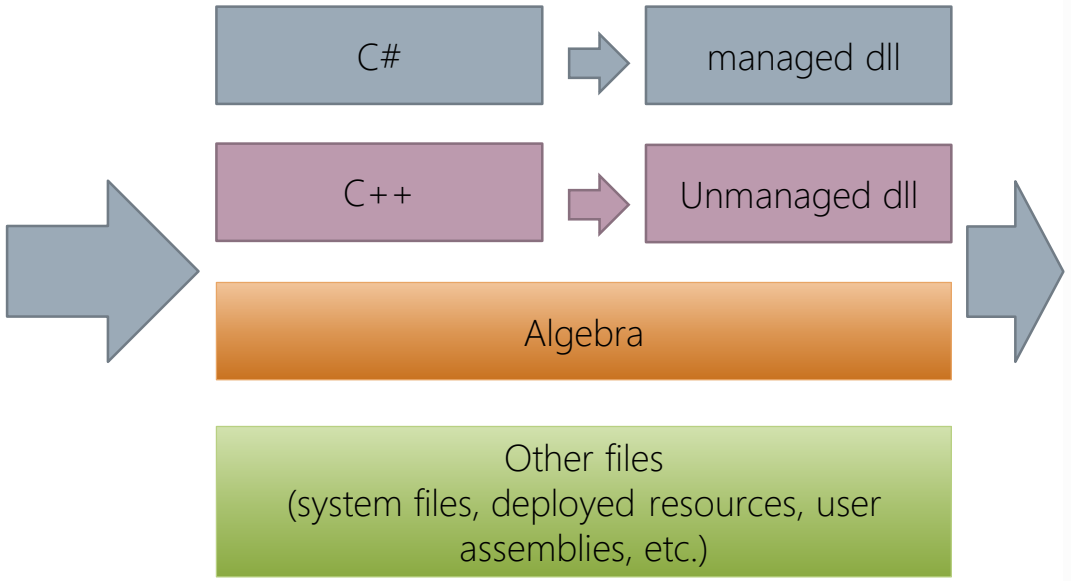
U-SQL Batch Job Execution Lifetime



Source:Azure Data Lake Analytics Batch Query Execution (Microsoft)

Data Lake Analytics Compilation & Deployment

```
USE DATABASE [DemoDb];
REFERENCE ASSEMBLY [ADLAExt];

USING IpConverter =
ADLAExt.Utils.IpConverter;

@ds =
    SELECT
IpConverter.ToIp4Format(c_ip)
AS Ip,
           date.Date AS Date
    FROM @iisLogs;
```



Algebra.xml
- _ScopeCodeGen_.dll
- _ScopeCodeGen_.pdb
- _ScopeCodeGenEngine_.dll
- _ScopeCodeGenEngine_.pdb

PartitionLastRows.xml

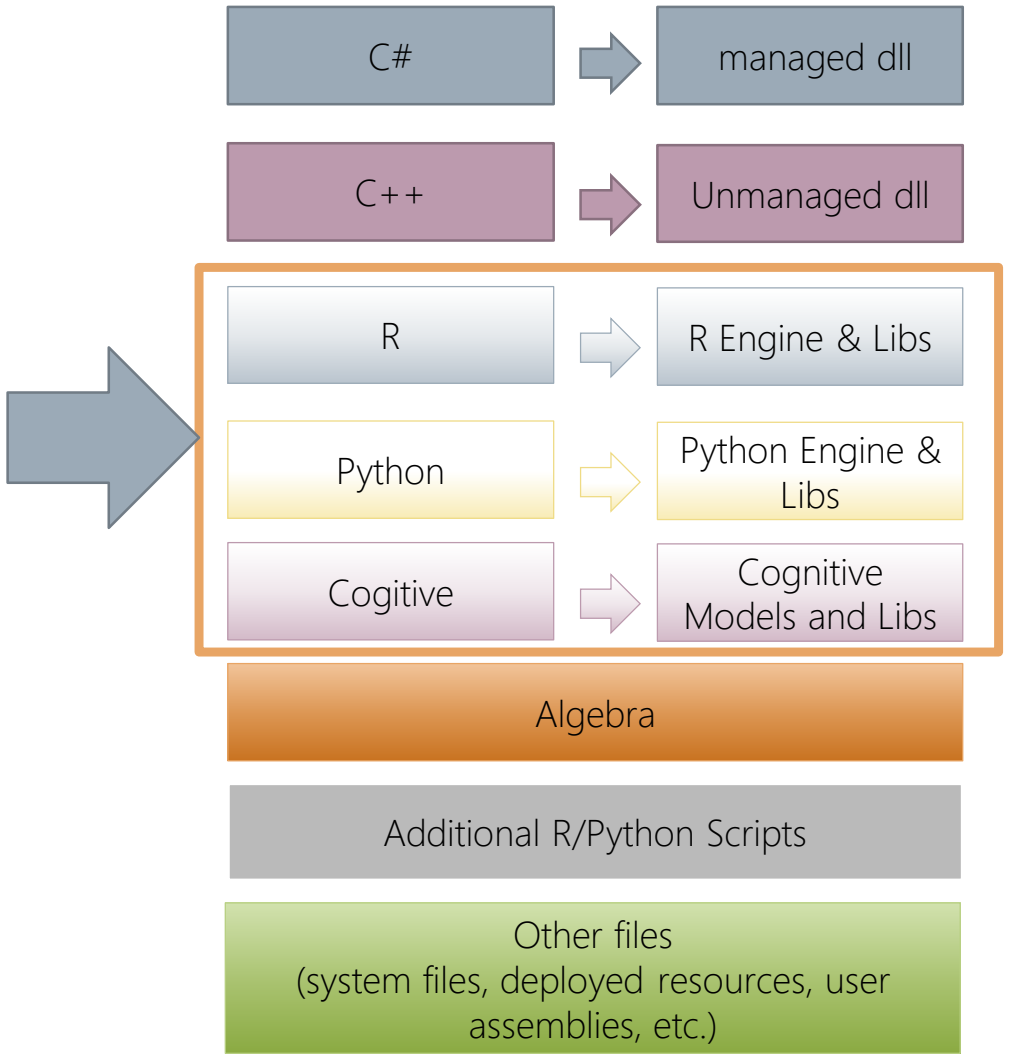
ScopeVertexDef.xml
- _ScopeCodeGen_.dll.cs
- _ScopeCodeGenEngine_.dll.cpp
- _ScopeCodeGenCompileOutput_.txt
- _ScopeCodeGenCompileOptions_.txt
- _ScopeCodeGenEngine_.cppresources

query.abr
- _ScopeDiagnosisInfo_.xml
- _SystemInternalInfo_.xml

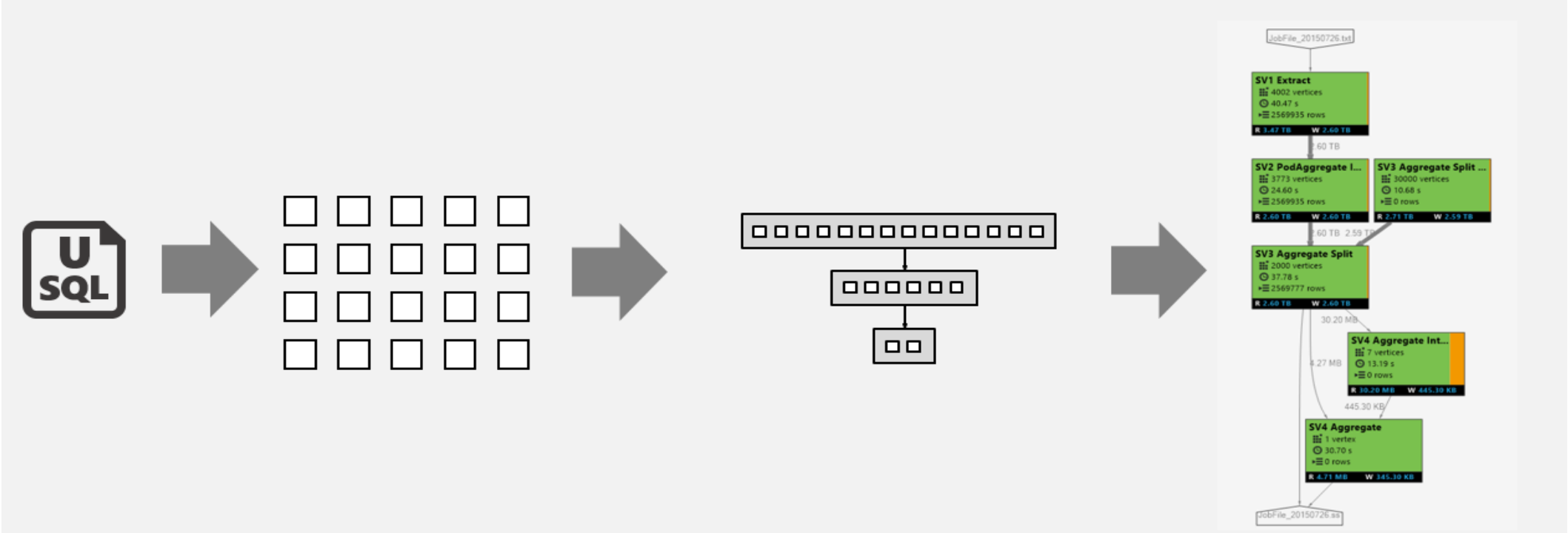
Profile
- _ScopeRuntimeStatistics_.xml

Data Lake Analytics Compilation & Deployment R/Python

```
@PyOutput =
    REDUCE @Extended
    ON Par
    PRODUCE Par int,
            SepalLength double,
            SepalWidth double,
            PetalLength double,
            PetalWidth double,
            Species string,
            SepalRatio double,
            PetalRatio double
    USING new Extension.Python.Reducer(pyScript : @myPyScript);

@ROutput =
    REDUCE @PyOutput
    ON Par
    PRODUCE Par,
            RowId int,
            ROutput string
    READONLY Par
    USING new Extension.R.Reducer(command : @myRScript,
                                  rReturnType:"charactermatrix",
                                     stringsAsFactors:true);
```



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"

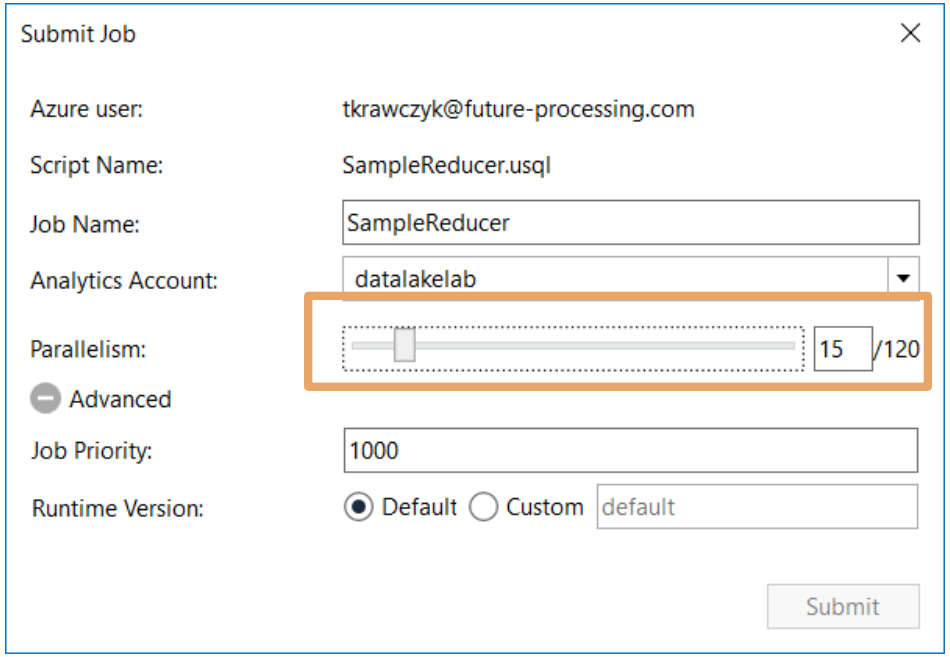
Source: S. Reddy Presentation

DEMO(s)

- Demo_003\Demo_GetADLUInfo

Azure Data Lake Analytics Optimization

› Allocation

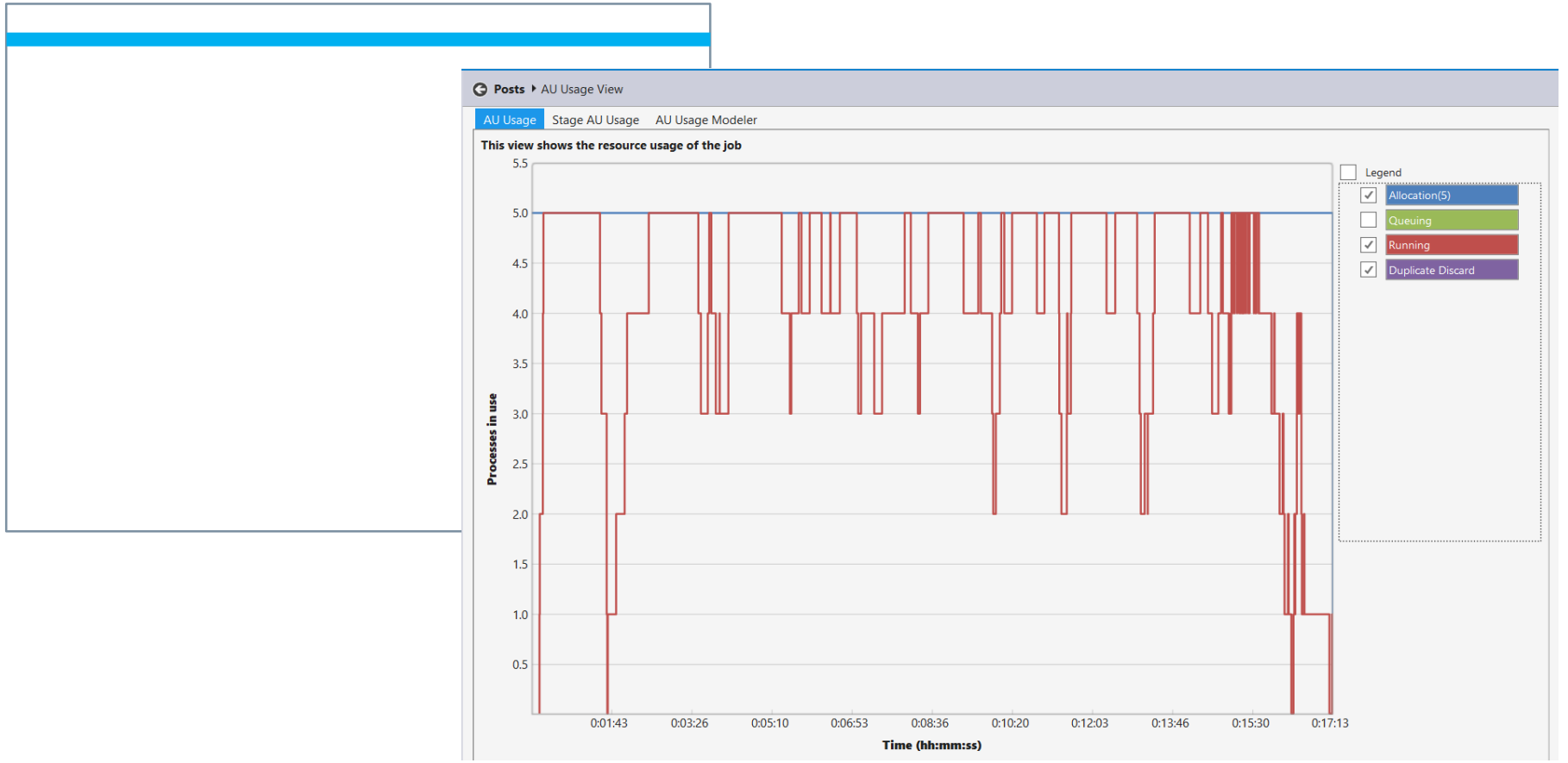


JobCost = (minutes x ADLUnits x ADLU Cost) /60
(ADLU Cost: 1h= 1,69 €)

Azure Data Lake Analytics Optimization

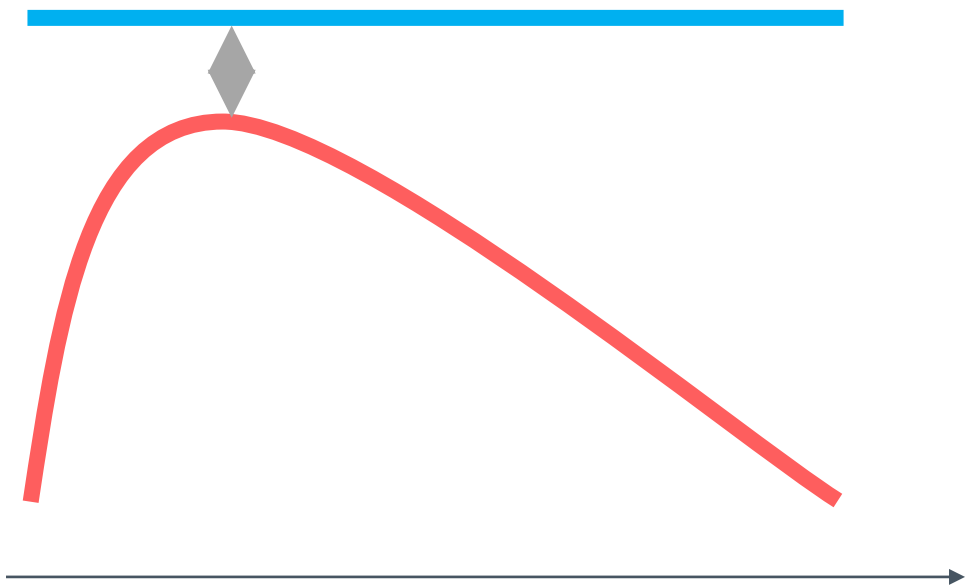
› Allocation

Blue line: Allocated



Azure Data Lake Analytics Optimization

- Allocation
  - Over Allocation

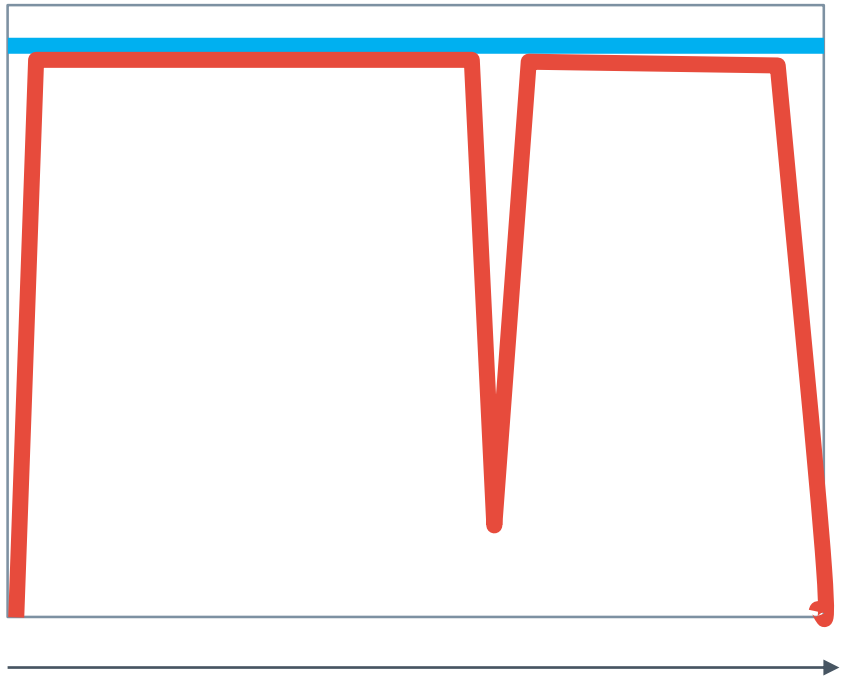


Blue line: Allocated

Red line: Running

Azure Data Lake Analytics Optimization

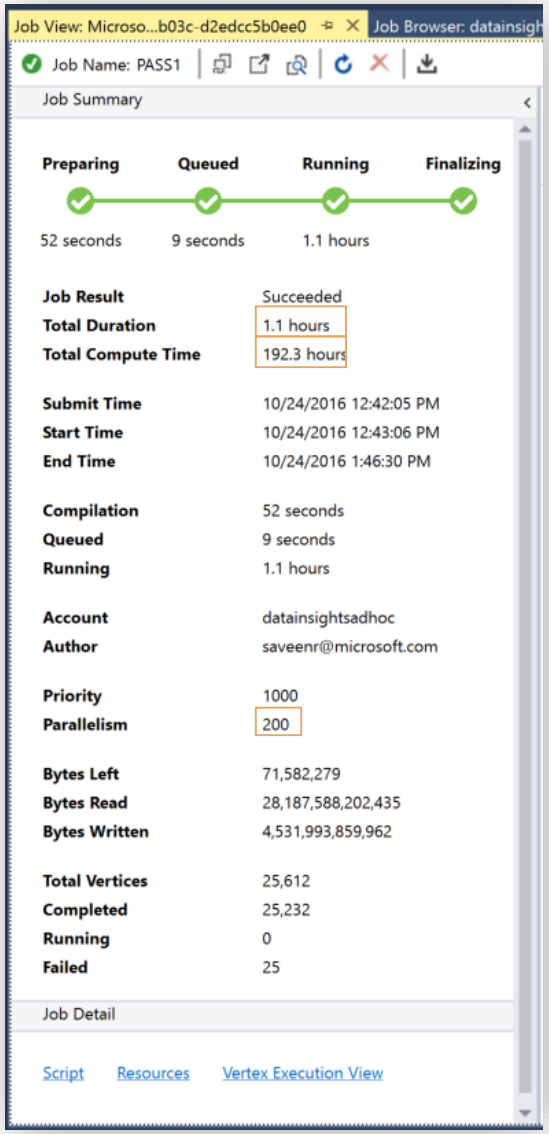
- Allocation
  - Under Allocation



Blue line: Allocated

Red line: Running

Azure Data Lake Analytic Efficiency

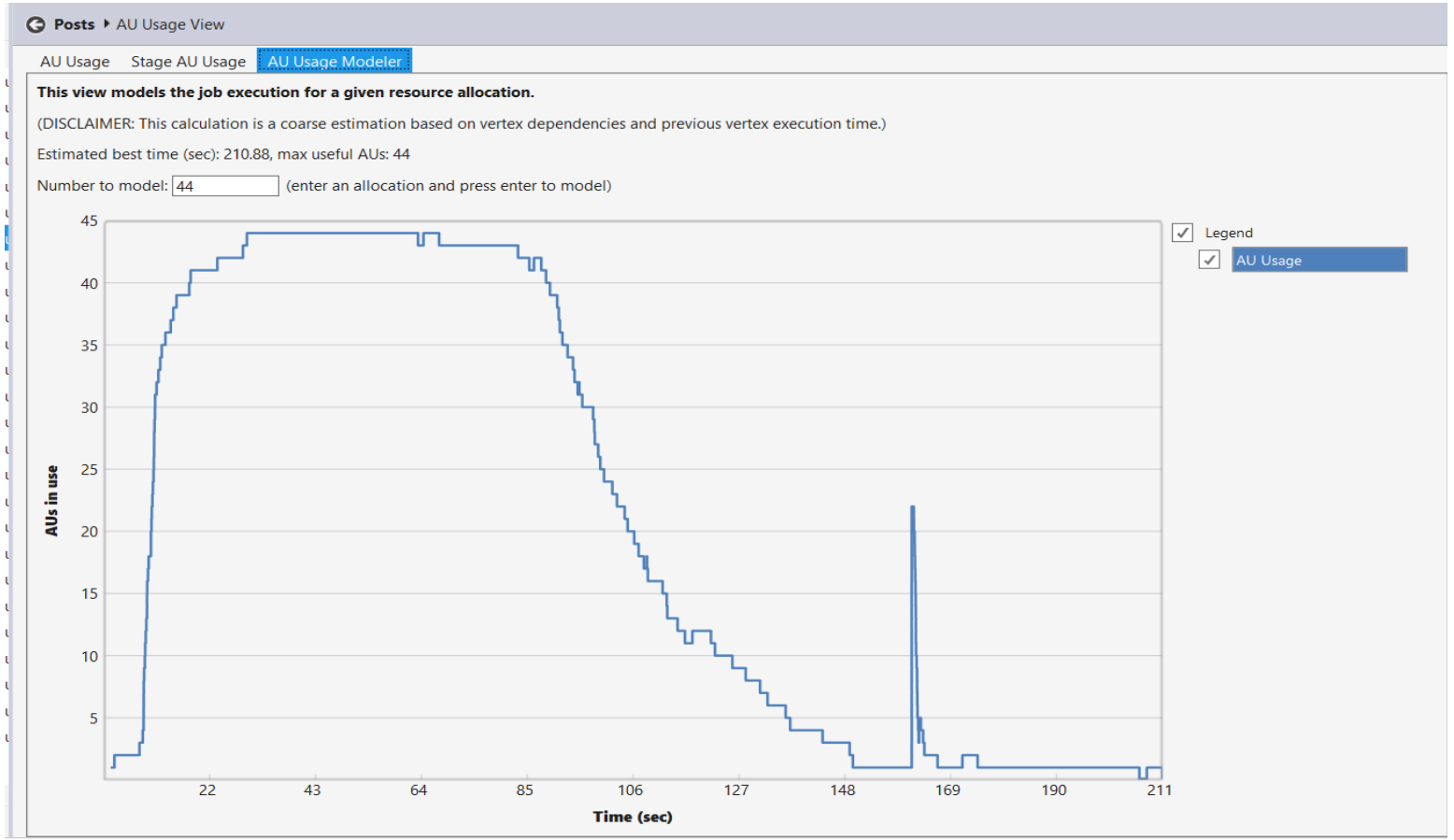


Allocation Efficiency =
(Compute hours) / (AUs * duration)

Example Job
= 192.3 hours / (200 AUs * 1.1 hours)
= 192.3 hours / 220 AUHours
= 0.874
= 87.4% Efficiency very very good!

Source: Saveen Reddy

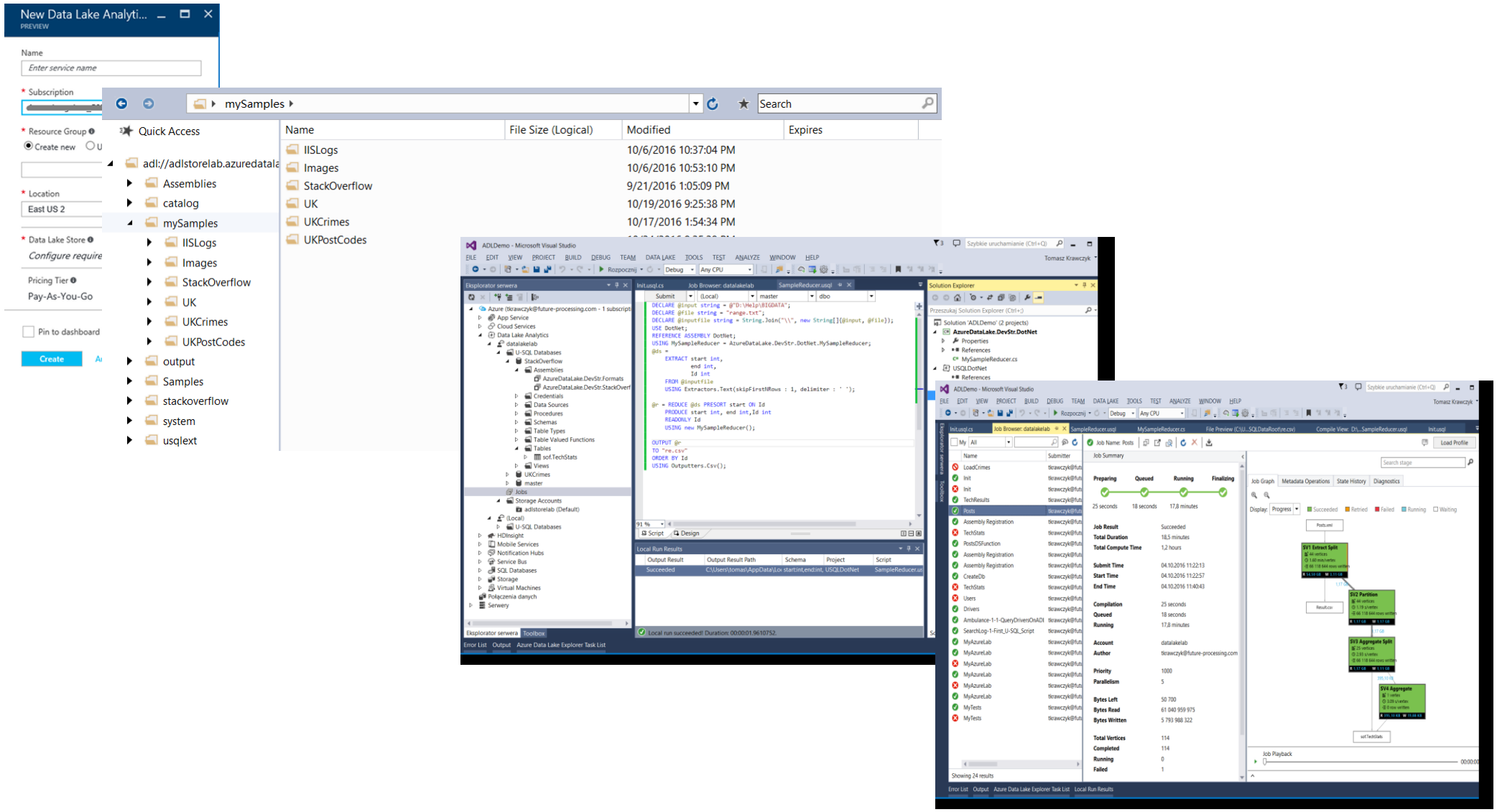
Data Lake Analytics - AU Usage Modeler



DEMO(s)

- Demos_001\Demo4StackoverflowTechRadar
- Demos_001\Demo5StackoverflowTechRadar

Building „End-to-End” Solution



Azure Data Integration
What we have?

- Azure Data Factory
  - Part of Azure (model PaaS)
  - Batch processing (based on time series orchestarction model)
  - Provides orchestraction ,movement services (on-premises and cloud) and monitoring service
- SSIS –SQL Server Integration Services
  - Part of SQL Server (Data Factory V2)
  - ETL to/from SQL Service
  - **Rich desinger** (and other tools e.g. BIML)

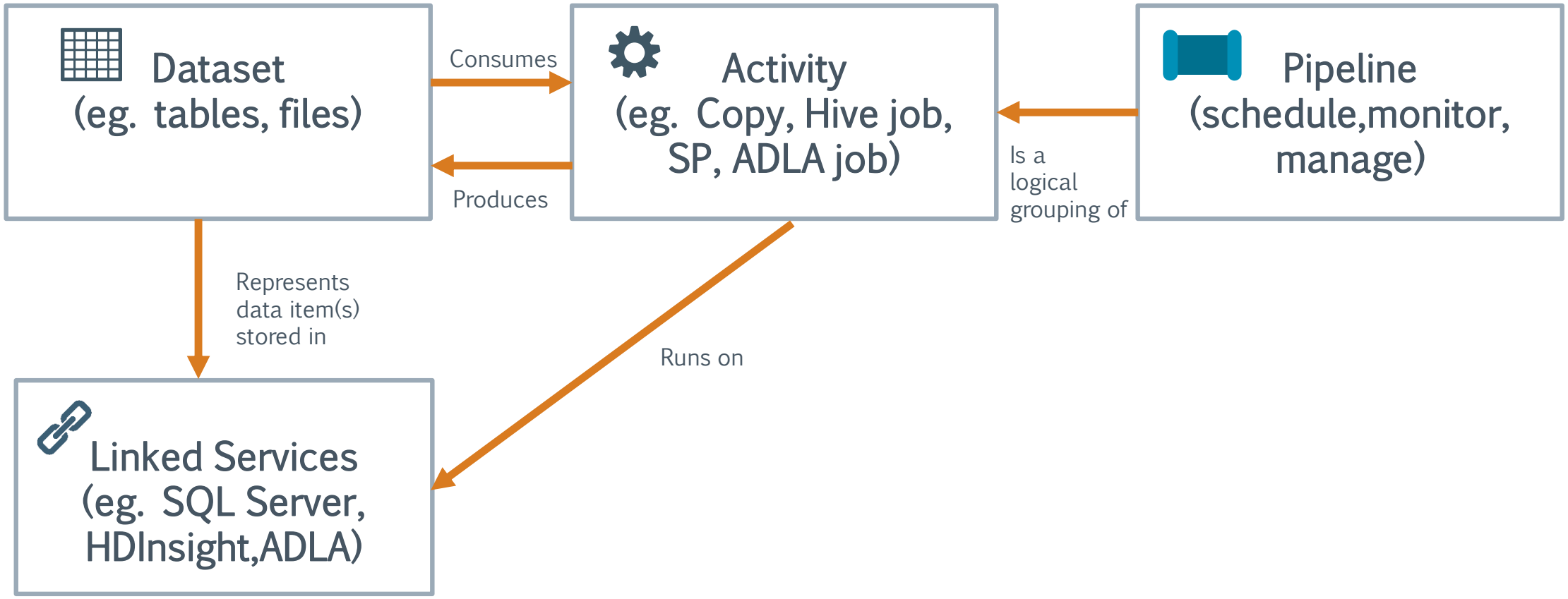
Azure Data Factory

- Fully managed service to support orchestration of data movement and transformation
- Connect to relational or non-relational data that is on-premises or in the cloud
- Allows monitor and manage data processing pipelines
- Globally deployed service infrastructure
- Cost Effective

Azure Data Factory Versions

- Azure Data Factory V1
  - Working version
- Azure Data Factory V2
  - Public Preview
  - Without designer

Azure Data Factory V1 Pipelines



Azure Data Lake & Azure Data Factory

- Data movement activities
  - Copy Activity
    - ADLS - Source and Destination
- Data transformation activities
  - Data Lake Analytics U-SQL
    - Scripts (Azure Blob Store)
    - Stored Procedures

Azure Data Lake & Azure Data Factory

ADF Pipeline

U-SQL Script

```
{
    "type": "DataLakeAnalyticsU-SQL",
    "typeProperties": {
        "scriptPath": "usql/MdMLoader.usql",
        "scriptLinkedService": "ABS_USQL",
        "degreeOfParallelism": 5,
      "parameters": {

      }
    },
    "inputs": [
        {
            "name": "MDM_WSKAZANIA_C2X-ADLS"
        },
        {
            "name": "MDM_WSKAZANIA_AMI-ADLS"
        }
    ],
    "outputs": [
        {
            "name": "U-SQLResult"
        }
    ],
    "scheduler": {
        "frequency": "Day",
        "interval": 1
    },
    "name": "Run U-SQL",
    "linkedServiceName": "ADLADemo"
}
```

U-SQL Script

```
DECLARE @currentDate DateTime = DateTime.UtcNow.AddDays(-1);
DECLARE @baseMdMFilesPath string = @"/mdm/";
DECLARE @MdmWskAMIPath string = @baseMdMFilesPath + "MDM_WSKAZANIA_AMI_" +
@currentDate.ToString("yyyyMMdd")+".csv";
DECLARE @currentPartition DateTime = @currentDate.ToUniversalTime();
// Add new partition if not extis
ALTER TABLE mdm.MDM_WSKAZANIA_AMI
ADD IF NOT EXISTS  PARTITION(@currentPartition);


@ami_ds =
    EXTRACT METER_CBP_ID string,
            READING_DATE DateTime,
            READING_TIME string,
            TIME_ZONE_ID string,
            MEASURMENT_ID string,
            READING_VALUE decimal
    FROM @MdmWskAMIPath
    USING Extractors.Csv(skipFirstNRows:1);

@ami_ds =
    SELECT METER_CBP_ID,
           READING_DATE,
           READING_DATE.ToUniversalTime().Hour AS  READING_HOUR,
           TIME_ZONE_ID,
           MEASURMENT_ID,
           READING_VALUE
    FROM @ami_ds;


INSERT INTO mdm.MDM_WSKAZANIA_AMI
(
    METER_CBP_ID,
    READING_HOUR,
    TIME_ZONE_ID,
    MEASURMENT_ID,
    READING_VALUE
)
PARTITION (@currentPartition)
SELECT METER_CBP_ID,
       READING_HOUR,
       TIME_ZONE_ID,
       MEASURMENT_ID,
       READING_VALUE
FROM @ami_ds;
```

Azure Data Lake & Azure Data Factory

```
{
"name": "plStandardizeBankingData",
"properties": {
"description": "Standardize JSON data into CSV, with friendly column names &
consistent output for all event types. Creates one output (standardized) file
per day.",
"activities": [
{
"type": "DataLakeAnalyticsU-SQL",
"typeProperties": {
"script":
"BankingADLDB.dbo.uspCreateStandardizedDataset(System.DateTime.Parse(@DateSli
ceStart), System.DateTime.Parse(@DateSliceEnd));",
"degreeOfParallelism": 30,
"priority": 100,
"parameters": {
"DateSliceStart": "$$Text.Format('{0:yyyy-MM-ddTHH:mm:ssZ}', SliceStart)",
"DateSliceEnd": "$$Text.Format('{0:yyyy-MM-ddTHH:mm:ssZ}', SliceEnd)"
}
},
"inputs": [
{
"name": "dsBankingADLSRawData"
}
],
"outputs": [
{
"name": "dsBankingADLSStandardizedData"
}
```

U-SQL Stored Procedure

Azure Data Lake & Azure Data Factory

π

DEMO(s)

- AzureDataFactory\Crimes

Azure Data Lake

Q&A

π

- Resources:
  https://msdn.microsoft.com/en-us/library/azure/mt591959.aspx
- Examples:
  https://github.com/devstr/usql
- Contact:
  tkrawczyk@future-processing.com