

<https://github.com/joemk2/synapseanalytics>
실습 파일을 다운 받아주세요.

Synapse 소개 자료는 Azure Synapse/Documents/Azure Synapse Analytics.pdf 입니다.

Azure Synapse Analytics



Contents

- Azure Synapse Analytics
- Synapse SQL
- Dedicated SQL Pool
- Serverless SQL Pool
- Aparch Spark



Azure **Synapse**
Analytics

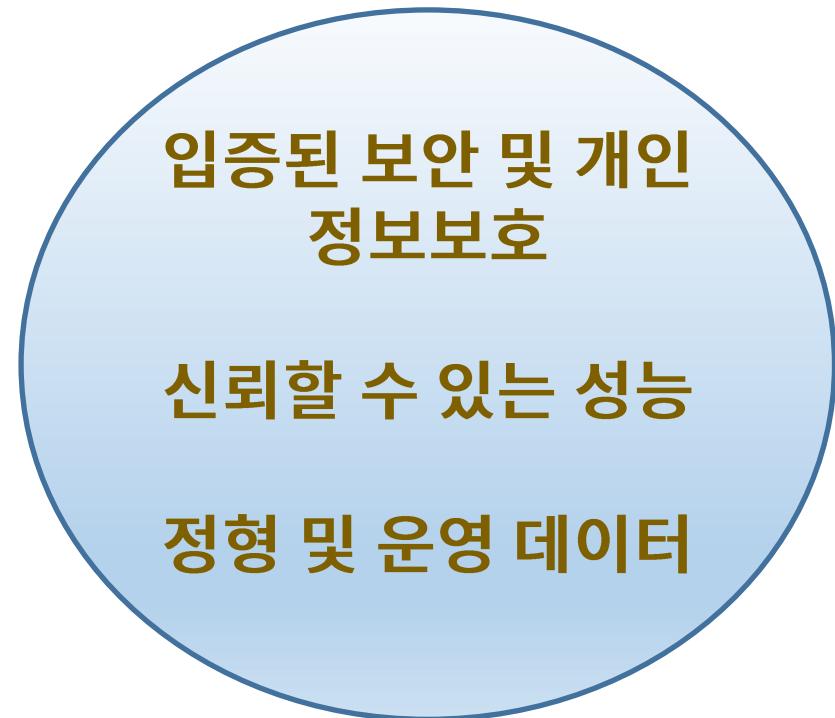
분석 플랫폼의 종류

Big Data



Data Lake

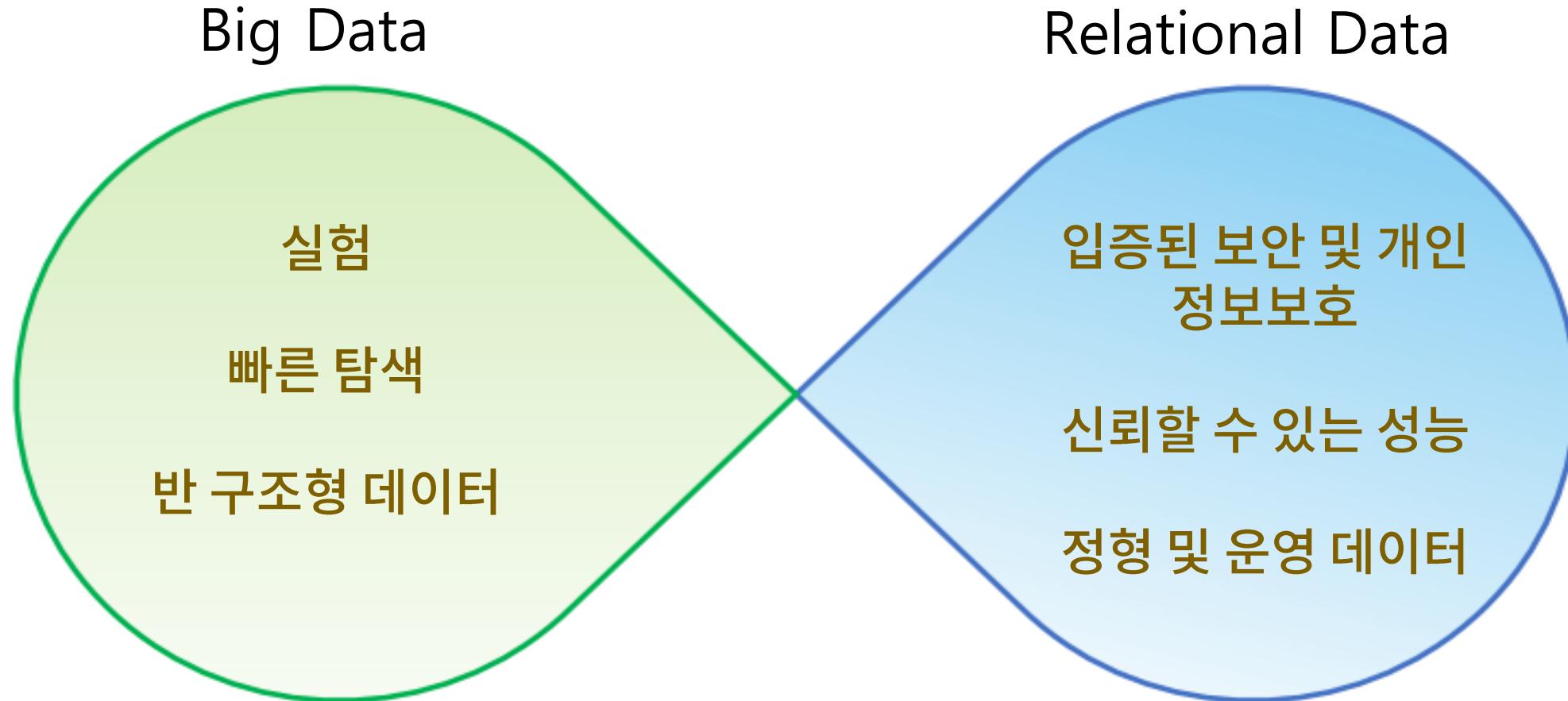
Relational Data



OR

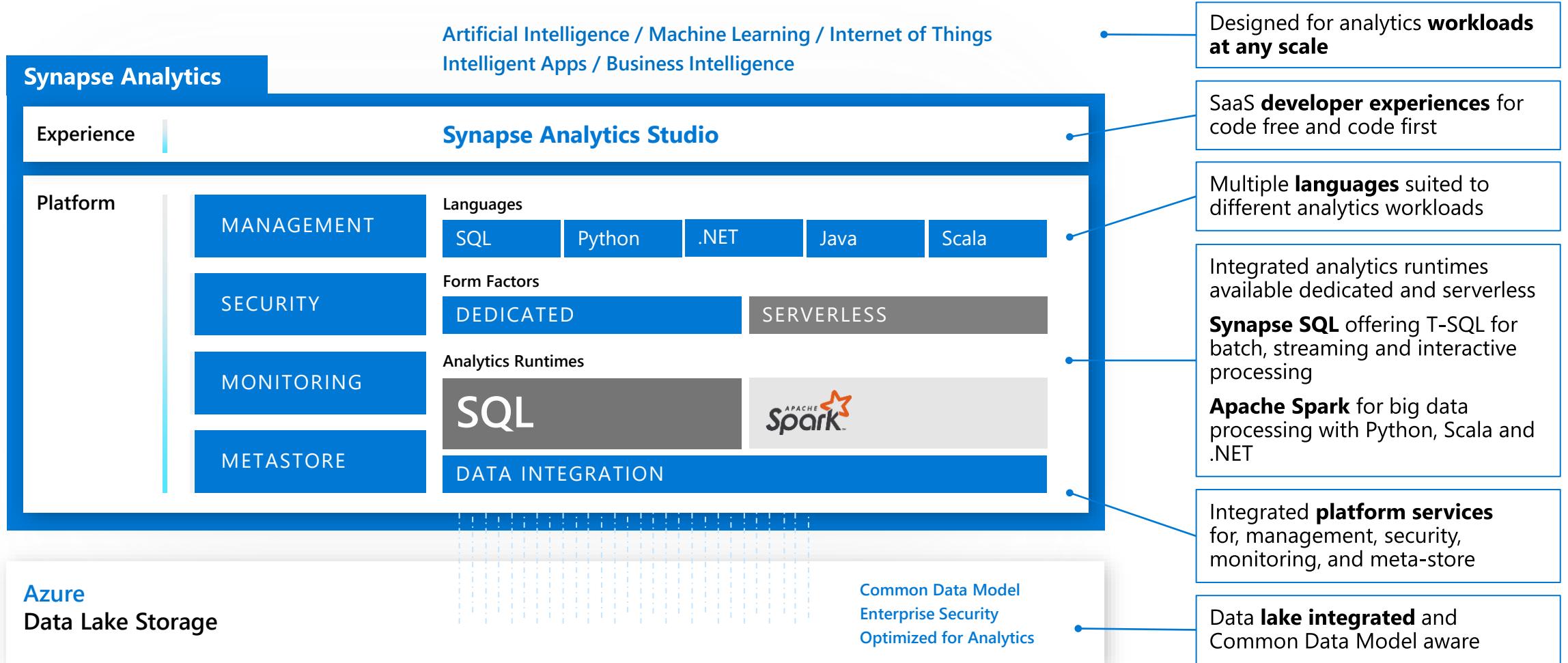
Data Warehouse

분석 플랫폼의 통합 – Azure Synapse Analytics



Big Data 분석 솔루션과 Data Warehouse 의 통합!

Synapse Analytics

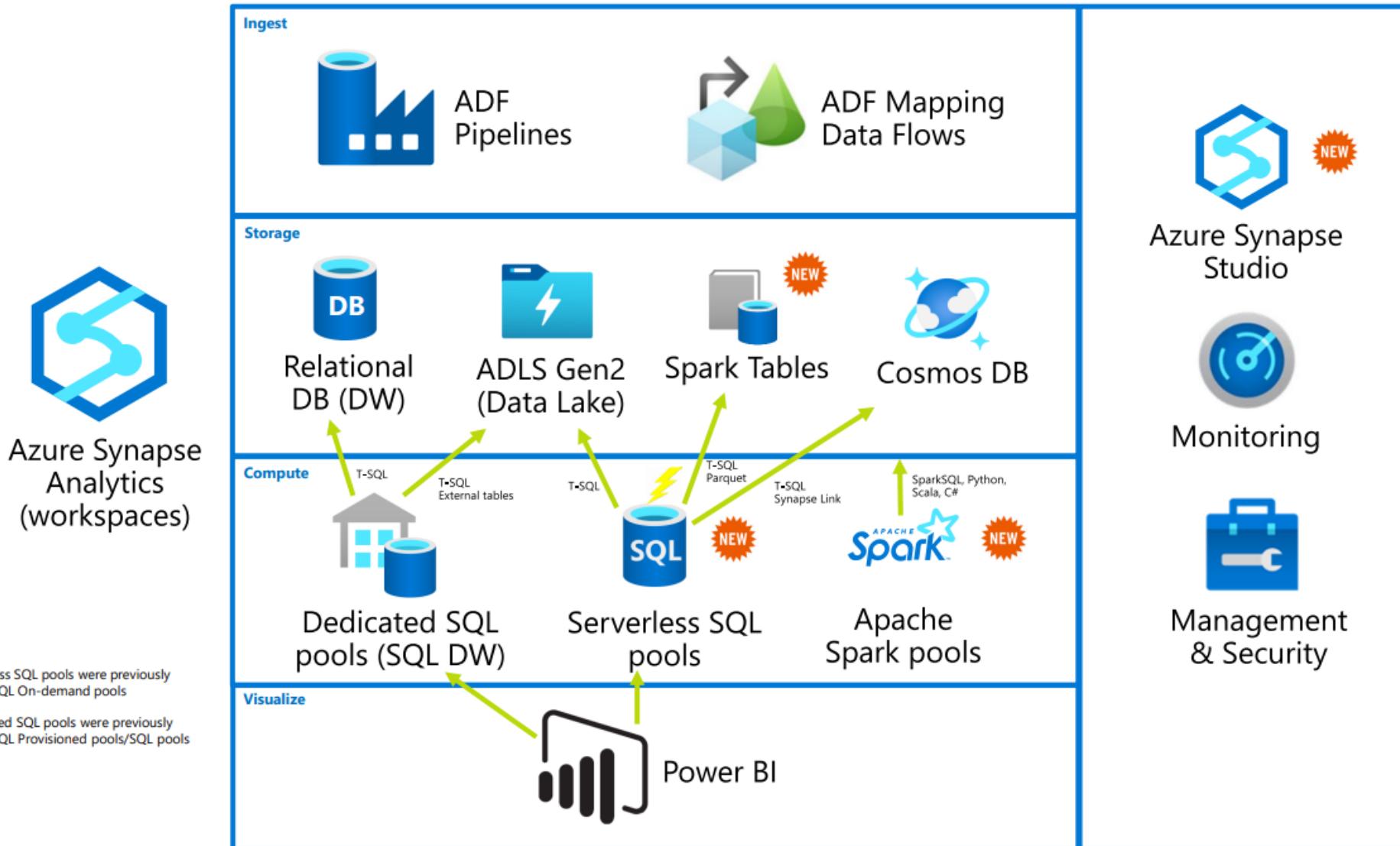


Synapse 를 이용한 데이터의 수집과 분석

Azure Synapse Analytics - *Data Lakehouse*



Synapse 를 이용한 데이터의 수집과 분석





Azure **Synapse**
Analytics
Synapse SQL

Synapse SQL 특징

Rich surface area

- T-SQL을 사용하여 데이터 분석
- 엔터프라이즈급 보안

Dedicated SQL Pool

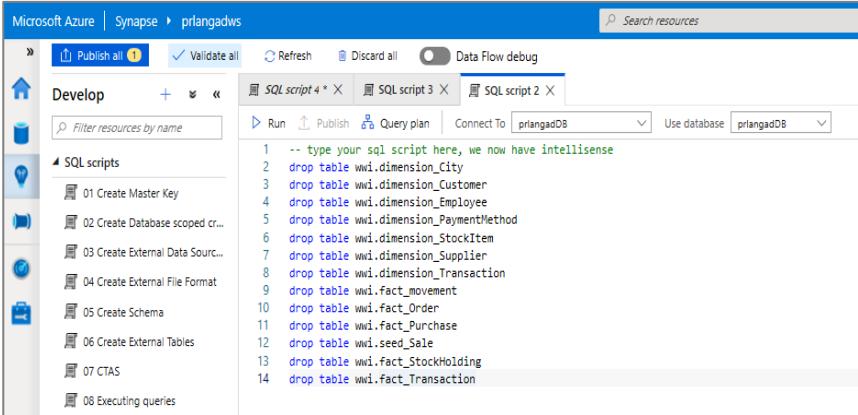
- 엔터프라이즈급 DW
- 인덱싱 & 캐싱 지원
- 외부 데이터 조회 및 적재
- 워크로드 관리

Serverless SQL Pool

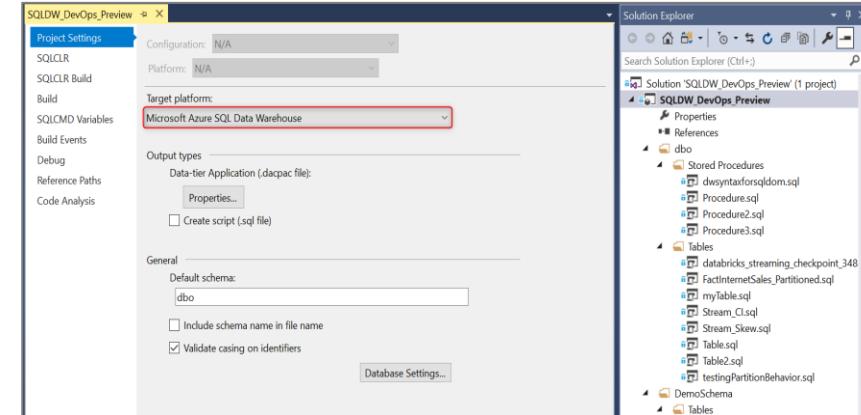
- 외부 데이터 조회
- 원시 데이터를 테이블 및 뷰로 모델화
- 쉬운 데이터 포맷 변환

Synapse SQL - Client and Tools

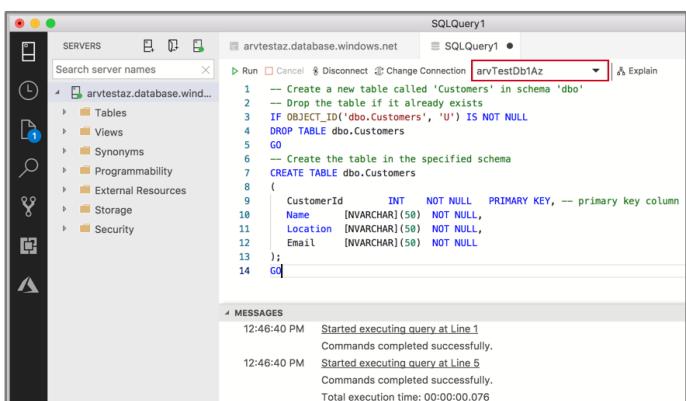
Azure Synapse Analytics



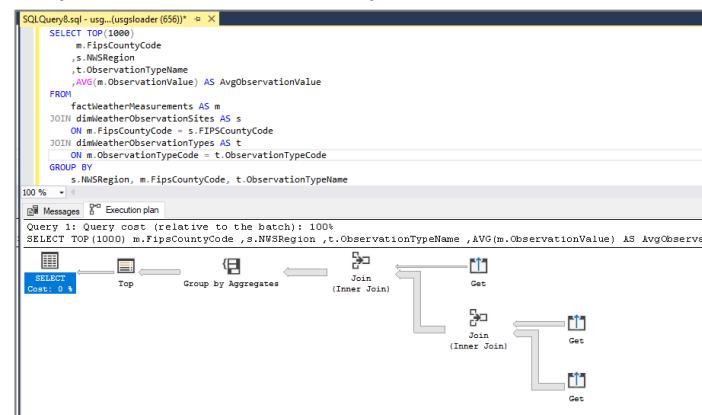
Visual Studio - SSDT database projects



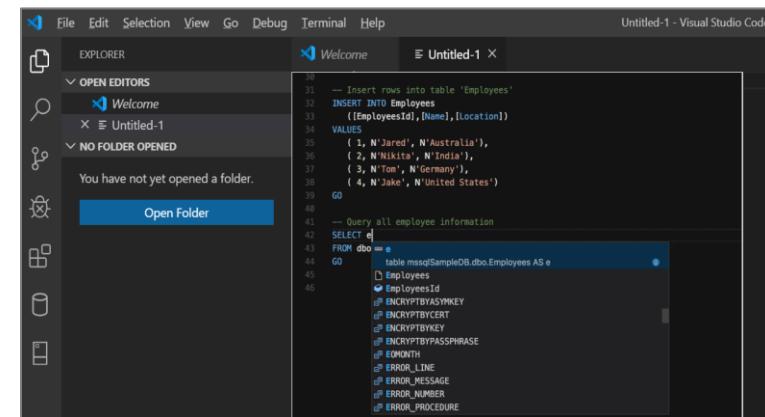
Azure Data Studio (queries, extensions etc.)



SQL Server Management Studio (queries, execution plans etc.)



Visual Studio Code



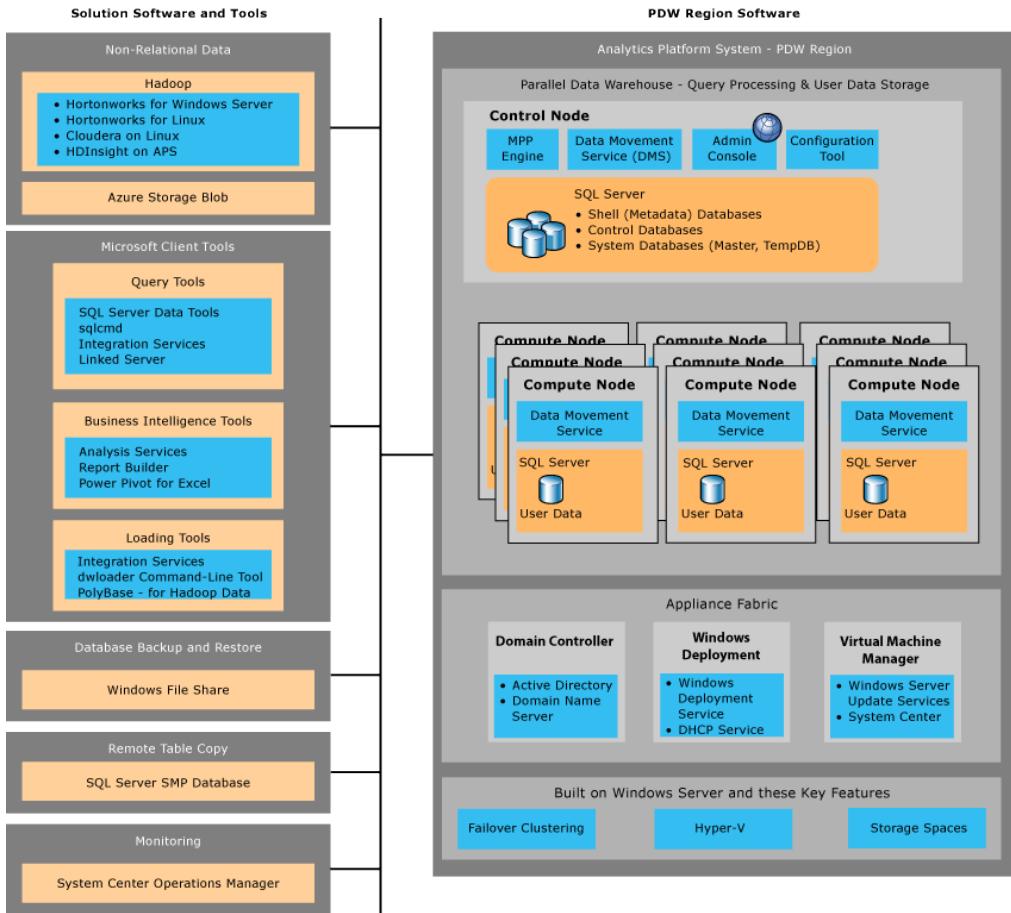


Azure Synapse Analytics

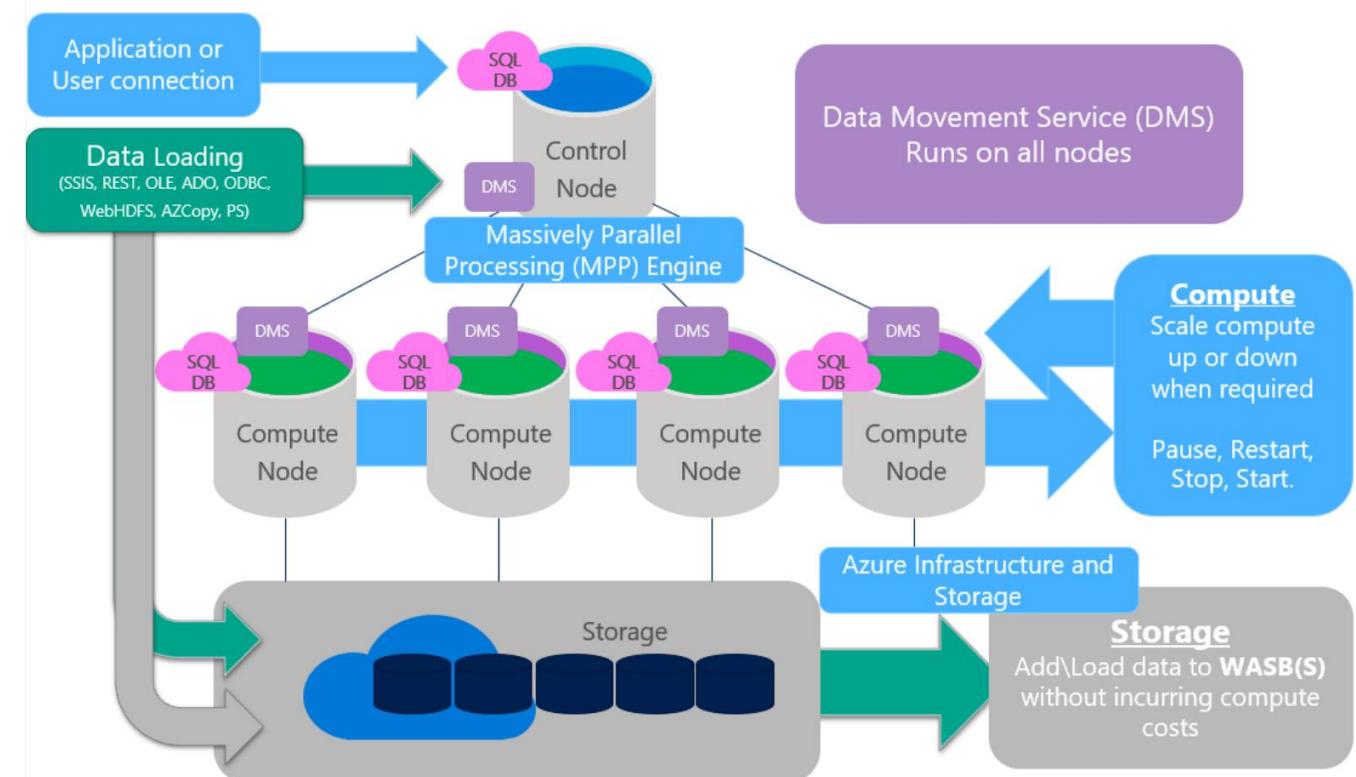
Dedicated SQL Pool

On-Premise PDW vs Synapse SQL Pool

PDW Architecture (Parallel Data Warehouse)

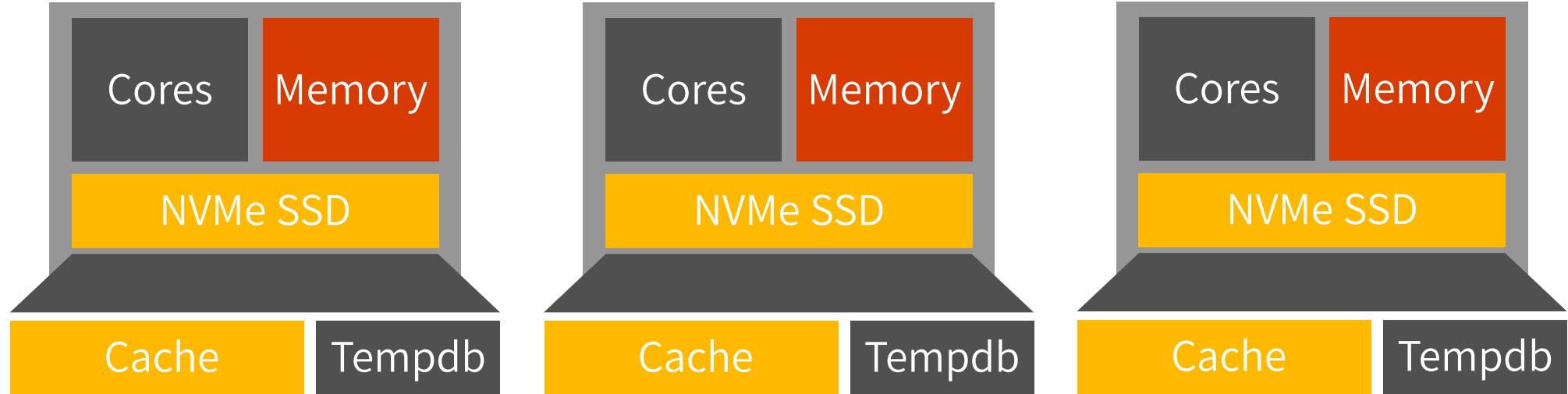


Synapse SQL Pool Architecture

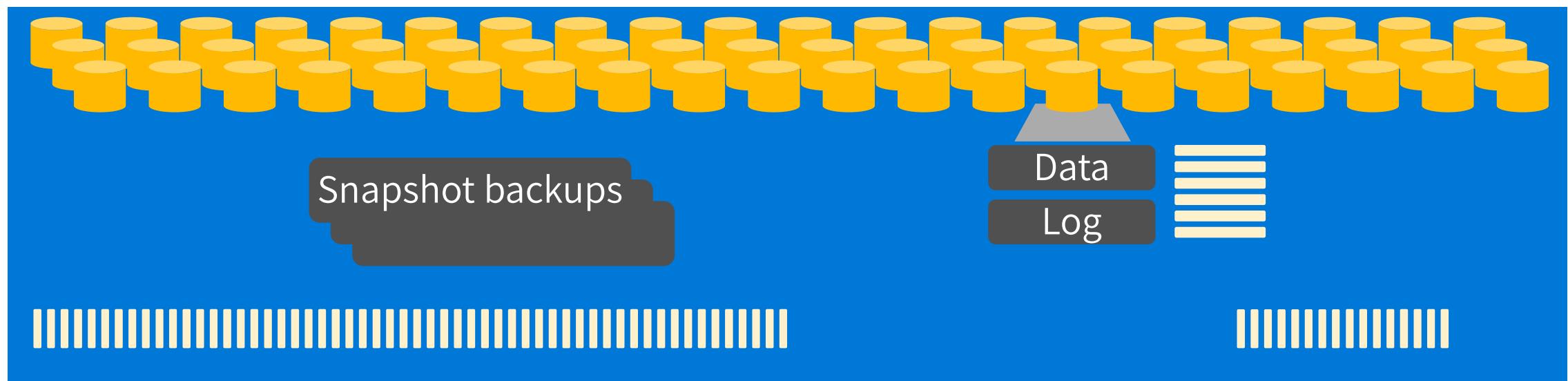


Compute, Storage 내부 구조

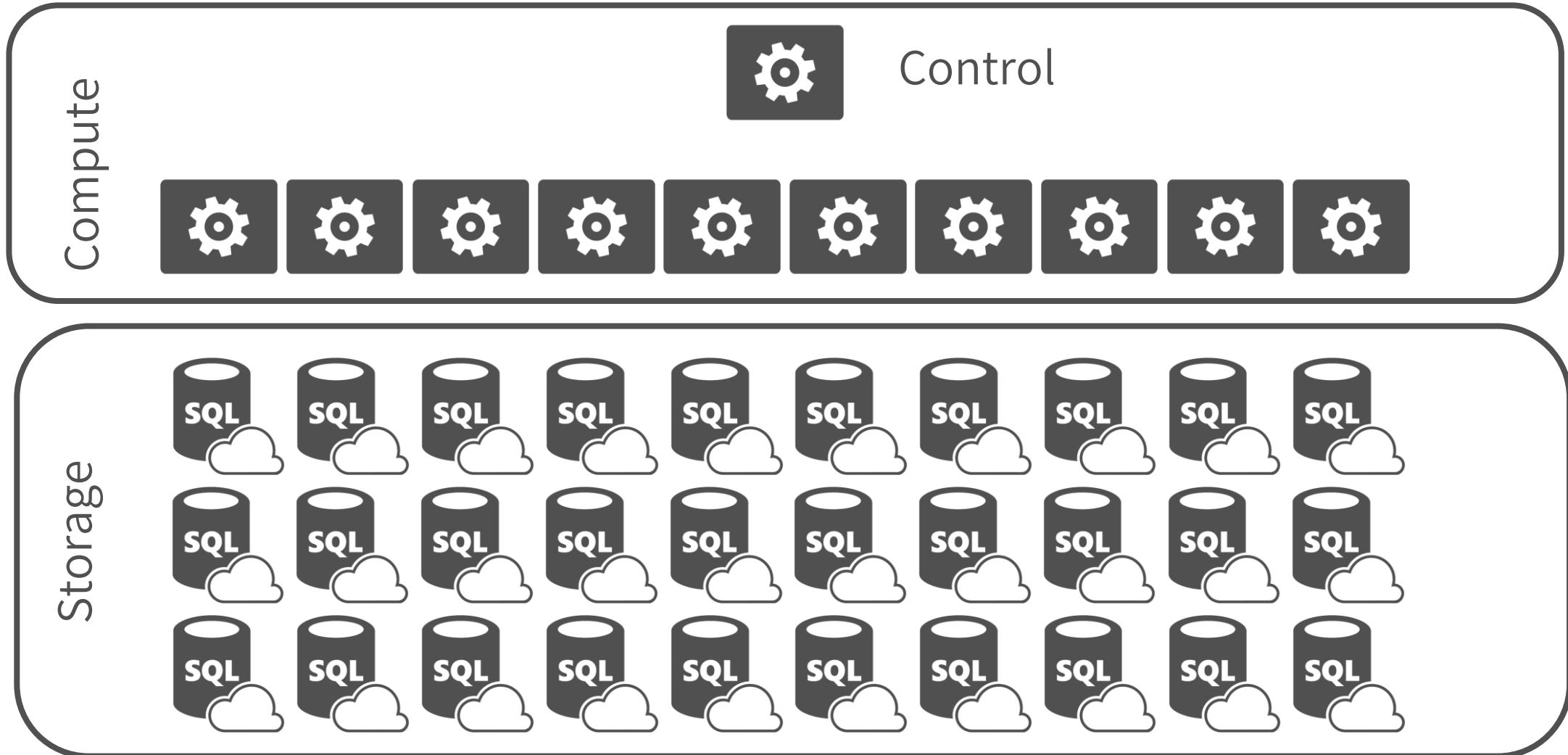
Compute



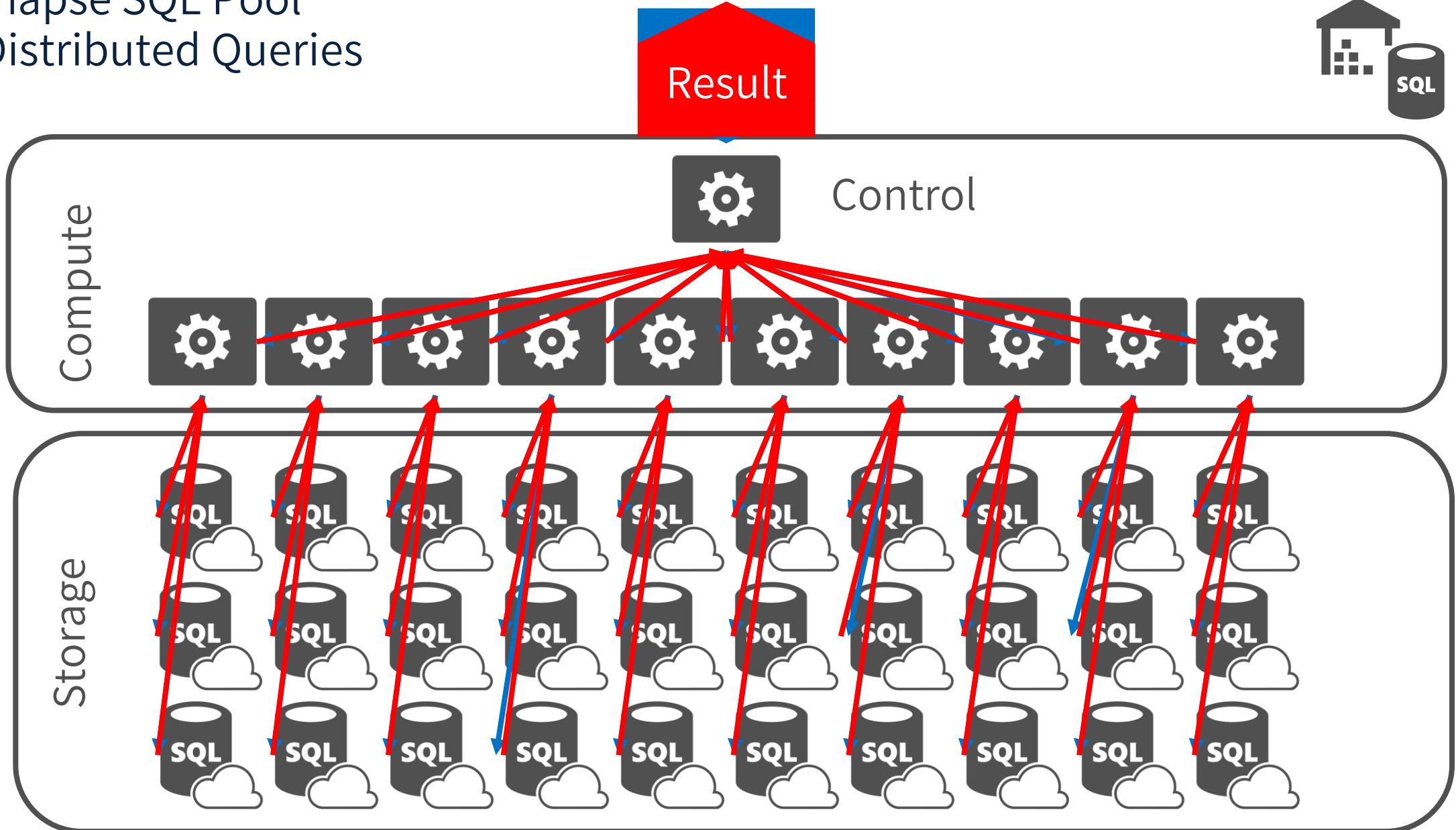
Remote Storage



Logical overview

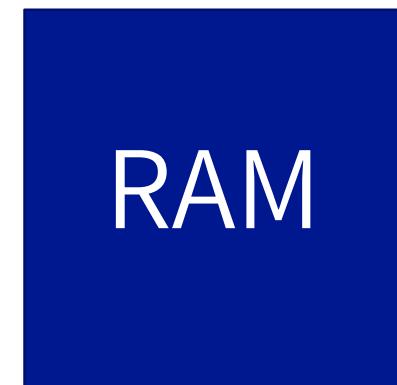
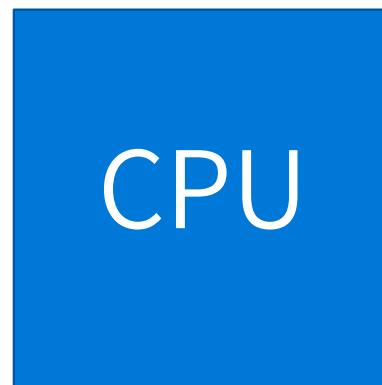


Synapse SQL Pool - Distributed Queries



Data Warehouse Units

Normalized amount of compute
Converts to billing units i.e. what you pay



DWUc
100
200
300
400
500
600
1000
1200
1500
2000
3000
6000
...
30000

Sizing Node Count / Cpu & Memory

Node Count

Performance level	Compute nodes
DW100c	1
DW200c	1
DW300c	1
DW400c	1
DW500c	1
DW1000c	2
DW1500c	3
DW2000c	4
DW2500c	5
DW3000c	6
DW5000c	10
DW6000c	12
DW7500c	15
DW10000c	20
DW15000c	30
DW30000c	60

CPU (Hyperthreaded vCore)

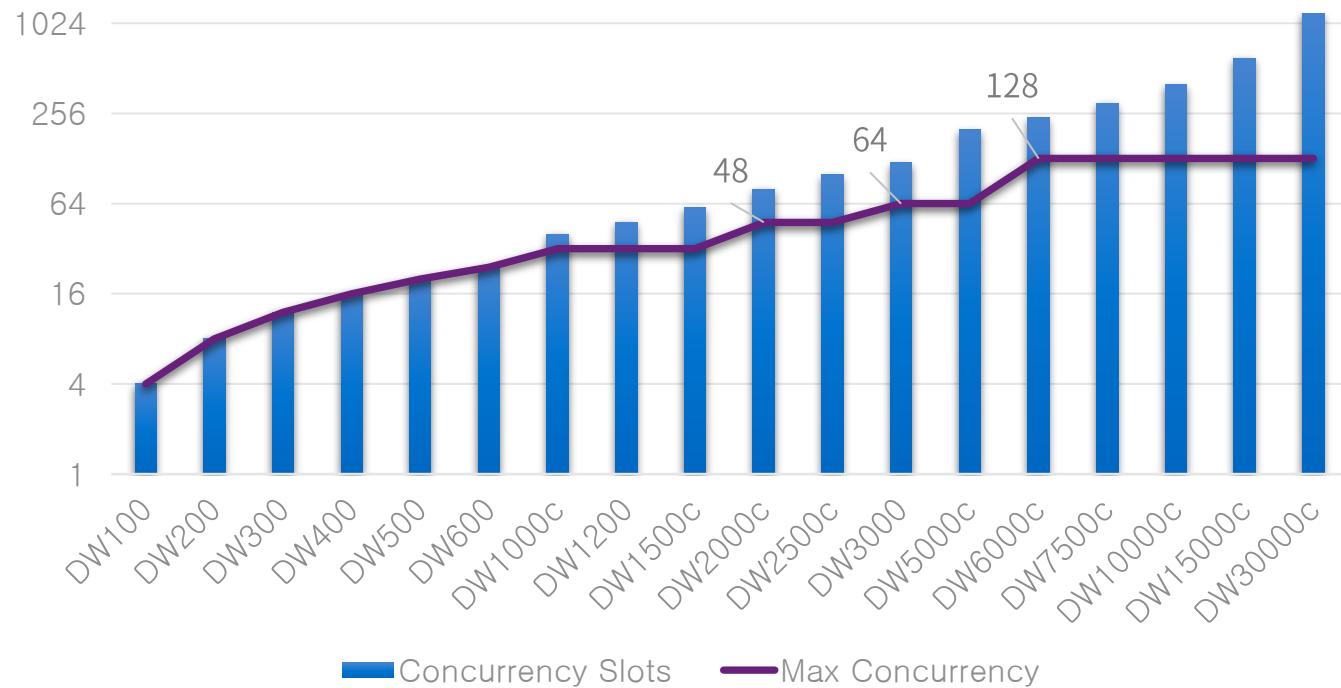
$$\begin{aligned} \text{Convert to DTU} &= \text{DWU} * 9 \\ \text{vCore} &= \text{DTU} / 100 \\ &= \text{DWU} * 9 / 100 \\ &= (6000 * 9) / 100 \\ &= 54000 / 100 \\ &= 540 \text{ vCore} \end{aligned}$$

Memory

$$\begin{aligned} \text{DWU} * 60 \\ (\text{DW6000c} * 60 = 3.6 \text{ Tb}) \end{aligned}$$

Performance level	Memory per data warehouse (GB)
DW100c	60
DW200c	120
DW300c	180
DW400c	240
DW500c	300
DW1000c	600
DW1500c	900
DW2000c	1,200
DW2500c	1,500
DW3000c	1,800
DW5000c	3,000
DW6000c	3,600
DW7500c	4,500
DW10000c	6,000
DW15000c	9,000
DW30000c	18,000

수행 가능한 동시 쿼리와 슬롯



서비스 수준	최대 동시 쿼리 수	사용 가능한 동시성 슬롯 수	쿼리당 필요한 최소의 리소스% (작업 그룹 사용 시)
DW100c	4	4	25%
DW200c	8	8	12.5%
DW300c	12	12	8%
DW400c	16	16	6.25%
DW500c	20	20	5%
DW1000c	32	40	3%
DW1500c	32	60	3%
DW2000c	48	80	2%
DW2500c	48	100	2%
DW3000c	64	120	1.5%
DW5000c	64	200	1.5%
DW6000c	128	240	0.75%
DW7500c	128	300	0.75%
DW10000c	128	400	0.75%
DW15000c	128	600	0.75%
DW30000c	128	1200	0.75%

리소스 클래스

정적 리소스 클래스

- 데이터의 집합 또는 크기가 고정적일 때 사용
- 현재 성능 수준에 관계없이 동일한 양의 메모리를 할당
- 서비스 성능 수준을 높이면 더 많은 쿼리를 수행 가능
- 정적 리소스 클래스 종류
 - Staticrc10, Staticrc20, Staticrc30, Staticrc40, Staticrc50, Staticrc60, Staticrc70, Staticrc80

동적 리소스 클래스

- 데이터 양이 증가하거나 변화하는 경우에 적합
- 서비스 수준을 높이면 쿼리에 더 많은 메모리 할당
- 동적 리소스 클래스 종류
 - smallrc, mediumrc, largerc, xlargerc
- 리소스 클래스 메모리 할당

서비스 수준	smallrc	mediumrc	Largerc	xlargerc
DW100c	25%	25%	25%	70%
DW200c	12.5%	12.5%	22%	70%
DW300c	8%	10%	22%	70%
DW400c	6.25%	10%	22%	70%
DW500c	5%	10%	22%	70%
DW1000c – DW3000c	3%	10%	22%	70%

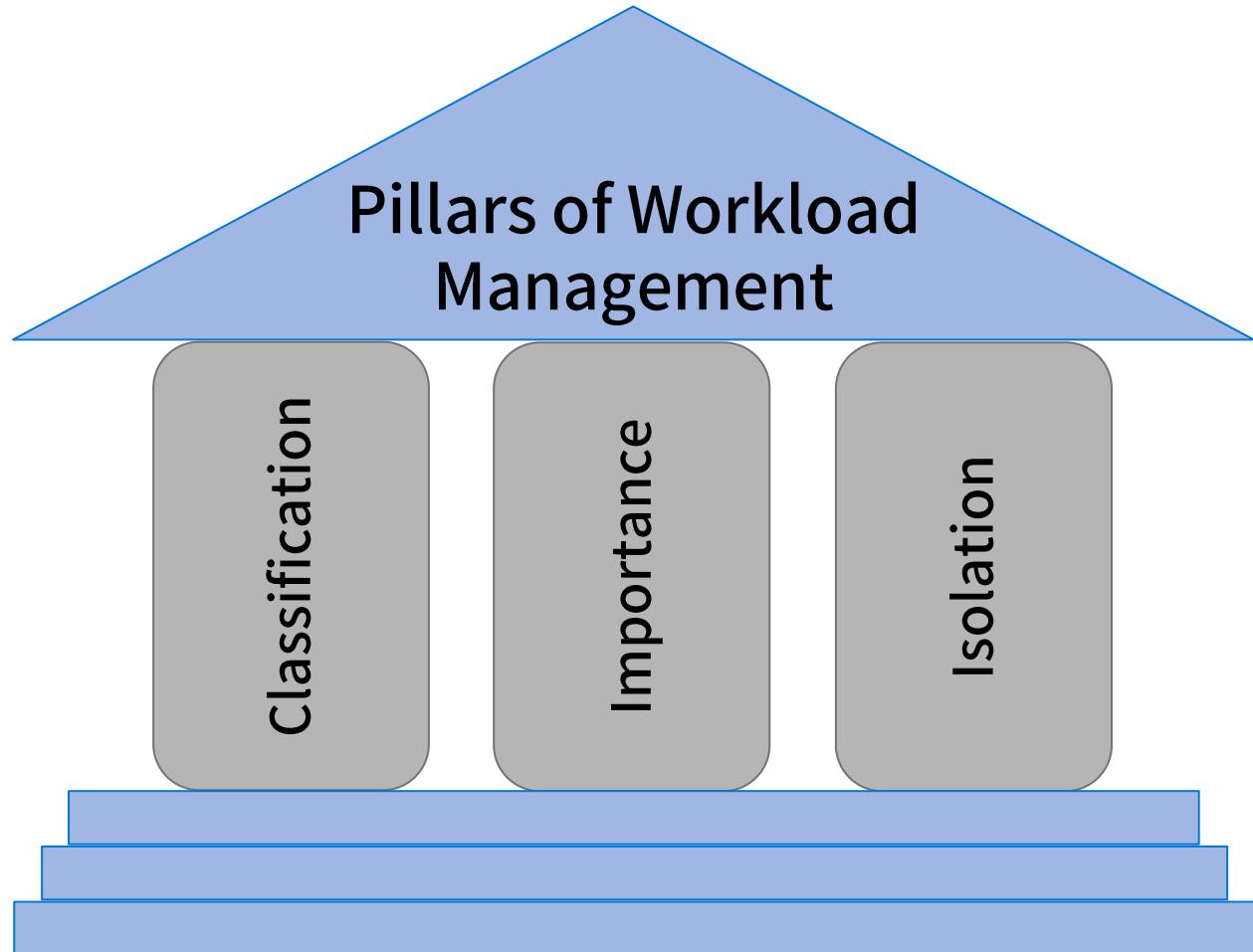
워크로드 관리

Overview

리소스의 효율적인 관리를 위해 사용

워크로드 관리 3대 요소

- 워크로드 분류 - 작업 요청을 그룹에 할당하고 중요도 설정
- 워크로드 중요도 - 그룹에 설정된 중요도에 따라 리소스의 액세스에 영향 받음
- 워크로드 독립성 - 워크로드 그룹에 리소스 예약



워크로드 관리

Overview

리소스의 효율적인 관리를 위해 사용

워크로드 관리 3대 요소

- 워크로드 분류 – 작업 요청을 그룹에 할당하고 중요도 설정
- 워크로드 중요도 – 그룹에 설정된 중요도에 따라 리소스의 액세스에 영향 받음
- 워크로드 독립성 – 워크로드 그룹에 리소스 예약

```
CREATE WORKLOAD CLASSIFIER classifier_name
WITH
(
    WORKLOAD_GROUP = 'name'
    , MEMBERNAME = 'security_account'
    [,] IMPORTANCE = {LOW| BELOW_NORMAL| NORMAL| ABOVE_NORMAL| HIGH} ]
    [,] WLM_LABEL = 'label' ]
    [,] WLM_CONTEXT = 'name' ]
    [,] START_TIME = 'start_time' ]
    [,] END_TIME = 'end_time' ]
)[;]
```

WORKLOAD_GROUP: maps to an existing resource class

IMPORTANCE: specifies relative importance of request

MEMBERNAME: database user, role, AAD login or AAD group

```
CREATE WORKLOAD GROUP group_name
```

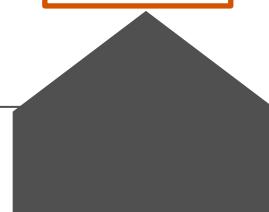
```
WITH
(
    MIN_PERCENTAGE_RESOURCE = value
    , CAP_PERCENTAGE_RESOURCE = value
    , REQUEST_MIN_RESOURCE_GRANT_PERCENT = value
    [,] REQUEST_MAX_RESOURCE_GRANT_PERCENT = value ]
    [,] IMPORTANCE = {LOW | BELOW_NORMAL | NORMAL | ABOVE_NORMAL | HIGH} ]
    [,] QUERY_EXECUTION_TIMEOUT_SEC = value ]
)[;]
```

테이블 분산 옵션

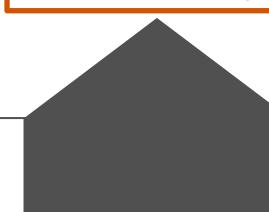
	Hash Distributed	Round Robin (Default)	Replicated
분산 방법	<ul style="list-style-type: none">해쉬 알고리즘 리턴 값에 의해 노드로 데이터 분산해쉬 알고리즘에 의한 결과값이 같으면 같은 노드에 데이터 분산해쉬 키 값은 Single Column Only	<ul style="list-style-type: none">모든 노드에 데이터가 분산비용 대비 단순한 저장 방식	<ul style="list-style-type: none">모든 노드에 데이터 복제
장점	<ul style="list-style-type: none">아래 작업 수행 시 속도가 빠름 <p>COUNT (DISTINCT <hashed_key>) OVER PARTITION BY <hashed_key> JOIN <table_name> ON <hashed_key> GROUP BY <hashed_key></p>	<ul style="list-style-type: none">초기 적재 속도가 빠름	<ul style="list-style-type: none">쿼리 계획을 단순화하고 데이터 이동 감소Hash 테이블과 조인 시 최적
사용 예	<ul style="list-style-type: none">Fact Table	<ul style="list-style-type: none">초기 데이터 이관시 Data Load스테이징 테이블에 사용	<ul style="list-style-type: none">Dimension Table (Fact Table 을 설명해주는 Table)Small Table
유의 사항	<ul style="list-style-type: none">특정 노드로 데이터 쓸림 발생NULLS, Default 값 사용 시 부적합Hash key 값은 update 되면 안됨	<ul style="list-style-type: none">쿼리 수행 시 노드간 데이터 이동 발생	<ul style="list-style-type: none">한 노드에서 두개의 복제 테이블이 조인 시 많은 공간이 소비됨

테이블 생성 옵션 : Round Robin, Hash, Replicated

```
CREATE TABLE [build].[FactOnlineSales]
(
    [OnlineSalesKey]      int          NOT NULL
    , [DateKey]            datetime     NOT NULL
    , [StoreKey]           int          NOT NULL
    , [ProductKey]         int          NOT NULL
    , [PromotionKey]       int          NOT NULL
    , [CurrencyKey]        int          NOT NULL
    , [CustomerKey]        int          NOT NULL
    , [SalesOrderNumber]   nvarchar(20) NOT NULL
    , [SalesOrderLineNumber] int          NULL
    , [SalesQuantity]      int          NOT NULL
    , [SalesAmount]         money        NOT NULL
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX
    DISTRIBUTION = ROUND_ROBIN
);
```



```
CREATE TABLE [build].[FactOnlineSales]
(
    [OnlineSalesKey]      int          NOT NULL
    , [DateKey]            datetime     NOT NULL
    , [StoreKey]           int          NOT NULL
    , [ProductKey]         int          NOT NULL
    , [PromotionKey]       int          NOT NULL
    , [CurrencyKey]        int          NOT NULL
    , [CustomerKey]        int          NOT NULL
    , [SalesOrderNumber]   nvarchar(20) NOT NULL
    , [SalesOrderLineNumber] int          NULL
    , [SalesQuantity]      int          NOT NULL
    , [SalesAmount]         money        NOT NULL
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX
    DISTRIBUTION = HASH([ProductKey])
);
```



```
CREATE TABLE dbo.DimCustomer
(
    CustomerKey           int          NOT NULL
    , GeographyKey         int          NULL
    , CustomerAlternateKey nvarchar(15) NOT NULL
    , Title                nvarchar(8)  NULL
    , FirstName             nvarchar(50) NULL
    , LastName              nvarchar(50) NULL
    , BirthDate             date        NULL
    , Gender                nvarchar(1) NULL
    , EmailAddress          nvarchar(50) NULL
    , YearlyIncome           money      NULL
    , DateFirstPurchase     date        NULL
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX
    DISTRIBUTION = REPLICATED
);
```



테이블 Partitions

Overview

- 테이블 데이터를 작은 그룹으로 분리하는 작업
- 대부분 날짜 컬럼을 파티션 키를 사용
- 모든 테이블에 파티션 기능 지원

RANGE RIGHT – Used for time partitions

RANGE LEFT – Used for number partitions

Benefits

- 데이터 로딩, 조회에 효율성 및 속도 증가
- 데이터 조회 시, 불필요한 스캔의 감소와 이로 인한 I/O 의 감소로 상당한 쿼리 속도 증가

```
CREATE TABLE partitionedOrderTable
(
    OrderId INT NOT NULL,
    Date DATE NOT NULL,
    Name VARCHAR(2),
    Country VARCHAR(2)
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = HASH([OrderId]),
    PARTITION (
        [Date] RANGE RIGHT FOR VALUES (
            '2000-01-01', '2001-01-01', '2002-01-01',
            '2003-01-01', '2004-01-01', '2005-01-01'
        )
    )
);
```

테이블 Partitions

논리적 Table 구조

OrderId	Date	Name	Country
85016	11-2-2018	V	UK
85018	11-2-2018	Q	SP
85216	11-2-2018	Q	DE
85395	11-2-2018	V	NL
82147	11-2-2018	Q	FR
86881	11-2-2018	D	UK
93080	11-3-2018	R	UK
94156	11-3-2018	S	FR
96250	11-3-2018	Q	NL
98799	11-3-2018	R	NL
98015	11-3-2018	T	UK
98310	11-3-2018	D	DE
98979	11-3-2018	Z	DE
98137	11-3-2018	T	FR
...

물리적 Data distribution (Hash distribution (OrderId), Date partitions)

Distribution1 (OrderId 80,000 – 100,000)

11-2-2018 partition

OrderId	Date	Name	Country
85016	11-2-2018	V	UK
85018	11-2-2018	Q	SP
85216	11-2-2018	Q	DE
85395	11-2-2018	V	NL
82147	11-2-2018	Q	FR
86881	11-2-2018	D	UK
...

11-3-2018 partition

OrderId	Date	Name	Country
93080	11-3-2018	R	UK
94156	11-3-2018	S	FR
96250	11-3-2018	Q	NL
98799	11-3-2018	R	NL
98015	11-3-2018	T	UK
98310	11-3-2018	D	DE
98979	11-3-2018	Z	DE
98137	11-3-2018	T	FR
...

...

x 60 distributions (shards)

- 각 Shard 는 같은 Date로 파티션으로 되어있음
- 최적의 압축과 Clustered Columnstore table의 퍼포먼스 향상을 위해서는 각 파티션에 최소 100만건의 row 가 필요

Index

Clustered Columnstore index (Default Primary)

- 최적의 데이터 압축 지원
- 최고의 쿼리 퍼포먼스 지원

Clustered index (Primary)

- 단일 행, 극소수 행을 조회할 때 최적
(Multi-Column Index 가능, But 1개 이상의 Index 생성 불가)

Heap (Primary)

- 임시 데이터를 Loading 시 최적
- 사이즈가 작은 Look up Table 조회 시 사용

Nonclustered indexes (Secondary)

- Table에 Multiple Columns Index 를 허용
- Single Table에 여러개의 Multiple Non-Clustered 허용
- Primary Indexes 대안으로 Non-Clustered Indexes 생성 가능
- Lookup 테이블 조회에 성능 기대

Clustered Columnstore index order

- 정렬된 Clustered Columnstore Index를 사용하면 테이블 조회 시
- 필요한 세그먼트만 읽음으로써 쿼리 시간을 크게 단축 가능

-- Create table with index

```
CREATE TABLE orderTable
(
    OrderId INT NOT NULL,
    Date DATE NOT NULL,
    Name VARCHAR(2),
    Country VARCHAR(2)
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX |
        HEAP | CLUSTERED INDEX (OrderId)
);
```

-- Add non-clustered index to table

```
CREATE INDEX NameIndex ON orderTable (Name);
```

-- When Create Clustered Columnstore index order

```
CREATE CLUSTERED COLUMNSTORE INDEX ORDER (OrderId);
```



Azure Synapse Analytics

Serverless SQL Pool

Serverless SQL Pool 특징

Quick data exploration

- Azure storage 내부 파일의 스키마와 데이터 조회가 가능
- Parquet, CSV, JSON 파일 포맷 지원
- BI Tool을 위해 Azure storage에 액세스 할 수 있는 Direct connector 지원

Logical Data Warehouse

- Azure storage 내부 raw files 을 virtual tables 와 views로 생성 가능
- SQL을 사용하는 Tool을 이용하여 file을 분석 가능
- enterprise-grade security model

Easy data transformation

- CSV을 parquet 포맷으로 변경 지원
- 컨테이너와 Storage 계정들을 오가며 데이터 이동 가능
- 외부 스토리지에 대한 쿼리 결과를 저장 가능

Storage file의 쉬운 탐색 기능

The screenshot displays two side-by-side views of the Microsoft Azure Synapse Analytics interface.

Left View: Shows the 'opendataset' view for the 'holidays' dataset. The dataset contains several partitions (part-00000 to part-00003) and a success file (_SUCCESS). A context menu is open over the 'New SQL script - Select TOP 100 rows' item, which is highlighted with a red box. The menu options include Publish all, Validate all, Refresh, Discard all, Storage accounts, Databases, and Datasets.

Right View: Shows a query editor window. The top navigation bar shows the same resource path: Microsoft Azure | Synapse Analytics > internalsandboxwe5. The query editor displays an SQL script named 'SQL script 1'. The 'Connect to' dropdown is set to 'SQL on-demand', also highlighted with a red box. The SQL code is as follows:

```
1 SELECT
2     TOP 100 *
3     FROM
4     OPENROWSET(
5         BULK 'https://internalsandboxwe.dfs.core.windows.net/opendataset/part-00001-bd1aba93-a85a-4909-8bf4-f79afb6c946f-c000.snappy.parquet'
6         FORMAT='PARQUET'
7     ) AS [r];
```

The results pane shows a table with four columns: VENDORID, TPEPICKUPDATETIME, TPEPDROPOFFDATETIME, and PASSENGERCOUNT. The data is as follows:

VENDORID	TPEPICKUPDATETIME	TPEPDROPOFFDATETIME	PASSENGERCOUNT
VTS	2009-05-07T23:1...	2009-05-07T23:2...	1
VTS	2009-05-07T16:3...	2009-05-07T16:3...	5
VTS	2009-05-08T14:5...	2009-05-08T15:0...	3
VTS	2009-05-07T15:5...	2009-05-07T16:1...	1

At the bottom of the results pane, a message indicates: '00:00:31 Query executed successfully.'

다양한 파일 포맷의 쿼리 가능

Overview

다양한 파일 포맷으로 저장된 데이터를 조회하기 위해 OPENROWSET 함수를 사용한다

Benefits

- CSV, parquet, and JSON 파일 조회 가능
- 모든 파일에 대해 통합된 T-SQL 사용 가능
- 데이터의 변환과 조회 결과 데이터 분석을 위해 standard SQL 사용
- Use JSON functions to get the data from underlying files.
- Use JSON functions to get data from PARQUET nested types

```
SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.csv',
    FORMAT = 'CSV')
WITH (
    country_code VARCHAR(4),
    country_name VARCHAR(50),
    year INT,
    population INT
) AS nyc
```

```
SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.parquet',
    FORMAT = 'PARQUET') AS nyc
```

```
SELECT TOP 10 *
    JSON_VALUE(jsonContent, '$.countryCode') AS country_code,
    JSON_VALUE(jsonContent, '$.countryName') AS country_name
    JSON_VALUE(jsonContent, '$.year') AS year
    JSON_VALUE(jsonContent, '$.population') AS population
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/json/taxi/*.json',
    FORMAT='CSV',
    FIELDTERMINATOR = '0x0b',
    FIELDQUOTE = '0x0b',
    ROWTERMINATOR = '0x0b'
)
WITH ( jsonContent varchar(MAX) ) AS json_line
```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

스키마 탑 자동 정의

Overview

OPENROWSET 은 외부 파일의 컬럼과 컬럼의 데이터 타입을 자동으로 정의한다.

Benefits

No need to up-front analyze file structure to query the file
OPENROWSET identifies columns and their types based on underlying file metadata.

Perfect solution for data exploration where schema is unknown.

The functionality is available for both parquet & CSV files.

```
SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.parquet',
    FORMAT = 'PARQUET') AS nyc
```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

```
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://azuresynapsesa.dfs.core.windows.net/default/RetailData/StoreDemoGraphics.csv',
        FORMAT = 'CSV',
        PARSER_VERSION='2.0',
        HEADER_ROW = TRUE) AS [result]
```

StoreId	RatioAge60	CollegeRatio	Income	HighIncome15...	LargeHH	MinoritiesRatio	More1FullTime...	DistanceNeare...	SalesN
2	0.232864734	0.248934934	10.55320518	0.463887065	0.103953406	0.114279949	0.303585347	2.110122129	1.1428
5	0.117368032	0.32122573	10.92237097	0.535883355	0.103091585	0.053875277	0.410568032	3.801997814	0.6818

스키마 탑입 수동 정의

Overview

사용자가 쿼리의 WITH 절에 컬럼명과 컬럼의 데이터 탑입을 명시할 수 있다.

Benefits

Define result schema at query time in WITH clause.

No need for external format files.

Explicitly define exact return types, their sizes, and collations.

Improve performance by column elimination in parquet files.

```
SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.csv',
    FORMAT = 'CSV')
WITH (
    country_code VARCHAR(4),
    country_name VARCHAR(50),
    year INT,
    population INT
) AS nyc
```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

사용자 의한 구문 분석 정의

Overview

OPENROWSET 함수를 이용하여 사용자가 파일의 구분자와 원하는 컬럼만 선택적으로 명시하여 결과 값을 불러올 수 있습니다.

Benefits

Ability to read CSV files with custom format

- With or without header row
- Handle any new-line terminator (Windows or Unix style)
- Use custom field terminator and quote character
- Read UTF-8 and UTF-18 encoded files
- Use only a subset of columns by specifying column position after column types

```
SELECT *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/population/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_code] VARCHAR (5) 2,
    [country_name] VARCHAR (100) 4,
    [year] smallint 7,
    [population] bigint 9
) AS [r]
WHERE
    country_name = 'Luxembourg'
    AND year = 2017
```

Second, fourth, seventh and ninth columns are returned

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

와일드 카드(*) 를 사용한 다수의 파일 조회

Overview

OPENROWSET 함수에서 WildCards 를 사용하여 여러 폴더안에 있는 다수의 파일을 동시에 질의하여 결과를 얻을 수 있습니다.

```
SELECT YEAR(pickup_datetime) AS [year],  
       SUM(passenger_count) AS passengers_total,  
       COUNT(*) AS [rides_total]  
  FROM OPENROWSET(  
    BULK 'https://XYZ.blob.core.windows.net/csv/tax/year=*/month=1/*.parquet',  
    FORMAT = 'PARQUET') AS nyc  
 GROUP BY YEAR(pickup_datetime)  
 ORDER BY YEAR(pickup_datetime)
```

Benefits

Offers reading multiple files/folders through usage of wildcards

Offers reading specific file/folder

Supports use of multiple wildcards

	year	passengers_total	rides_total
1	2001	14	10
2	2002	29	16
3	2003	22	16
4	2008	378	188
5	2009	594	353
6	2016	102093687	61758523
7	2017	184464988	113496932
8	2018	86272771	53925040
9	2019	37	29
...	2020	6	6

폴더 구조를 이용한 일부의 데이터 조회

Overview

OPENROWSET 함수를 사용하여 여러 개의 파일로부터 원하는 데이터만을 조회할 수 있습니다.

Benefits

Use filepath() function to access actual values from file paths.

Eliminate sub-folders/partitions before the query starts execution

Query Spark/Hive partitioned data sets

```
SELECT  
    r.filepath(1) AS [year]  
    ,r.filepath(2) AS [month]  
    ,COUNT_BIG(*) AS [rows]  
FROM OPENROWSET(  
    BULK 'https://XYZ.blob.core.windows.net/year=*/month=/*/*.parquet',  
    FORMAT = 'PARQUET') AS [r]  
WHERE r.filepath(1) IN ('2017')  
    AND r.filepath(2) IN ('10', '11', '12')  
GROUP BY r.filepath(),r.filepath(1),r.filepath(2)  
ORDER BY filepath
```

year	month	rows
2017	10	9768815
2017	11	9284803
2017	12	9508276

Logical Data Warehouse - Views

Overview

Serverless SQL pool 은 논리적 Data Warehouse 로서 Azure Storage 에 있는 파일들을 사용하여 View 를 생성할 수 있습니다.

Benefits

Create SQL views on externally stored data

Access files using the view from various tools and language

Leverage rich T-SQL language to process and analyze data in external files exposed via views

Create PowerBI reports on the views created on external data

```
USE [mydbname]
GO

DROP VIEW IF EXISTS populationView
GO

CREATE VIEW populationView AS
SELECT *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/population/\*.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_code] VARCHAR (5),
    [country_name] VARCHAR (100),
    [year] smallint,
    [population] bigint
) AS [r]
```

```
SELECT
    country_name, population
FROM populationView
WHERE
    [year] = 2019
ORDER BY
    [population] DESC
```

	country_name	population
1	China	1389618778
2	India	1311559204
3	United States	331883986
4	Indonesia	264935824
5	Pakistan	210797836
6	Brazil	210301591
7	Nigeria	208679114
8	Bangladesh	161062905
9	Russia	141944641
10	Mexico	127318112

Logical Data Warehouse - Tables

Overview

Serverless SQL Pool 은 논리적 Data Warehouse 로서 Azure Storage 에 있는 파일들을 사용하여 External Table 을 생성할 수 있습니다.

Benefits

Create external tables that reference set of files on Azure storage.

Join and transform multiple tables in the same query.

Enables you to analyze external files with the same experience that you have in classic databases.

Manage column statistics in external tables.

Manage access rights per table.

Create PowerBI reports on the views created on external data

```
USE [mydbname]
GO

DROP TABLE IF EXISTS dbo.Population
GO

CREATE EXTERNAL TABLE dbo.Population (
    country_code VARCHAR (5) COLLATE Latin1_General_BIN2,
    country_name VARCHAR (100) COLLATE Latin1_General_BIN2,
    year smallint,
    population bigint
)
WITH(
    LOCATION = '/csv/population/population-*/*.csv',
    DATA_SOURCE = MyAzureStorage,
    FILE_FORMAT = MyAzureCSVFormat
)
```

```
CREATE STATISTICS stat_country_name
ON dbo.Population(country_name);
```

```
SELECT
    country_name, population
FROM population
WHERE year = 2019
ORDER BY population DESC
```

	country_name	population
1	China	1389618778
2	India	1311559204
3	United States	331883986
4	Indonesia	264935824
5	Pakistan	210797836
6	Brazil	210301591
7	Nigeria	208679114
8	Bangladesh	161062905
9	Russia	141944641
10	Mexico	127318112

데이터 포맷 변환

Overview

SQL queries 을 사용하여 Azure Storage Files 의 데이터 포맷을 쉽게 변환할 수 있습니다.

(CSV to Parquet, Json to Parquet)

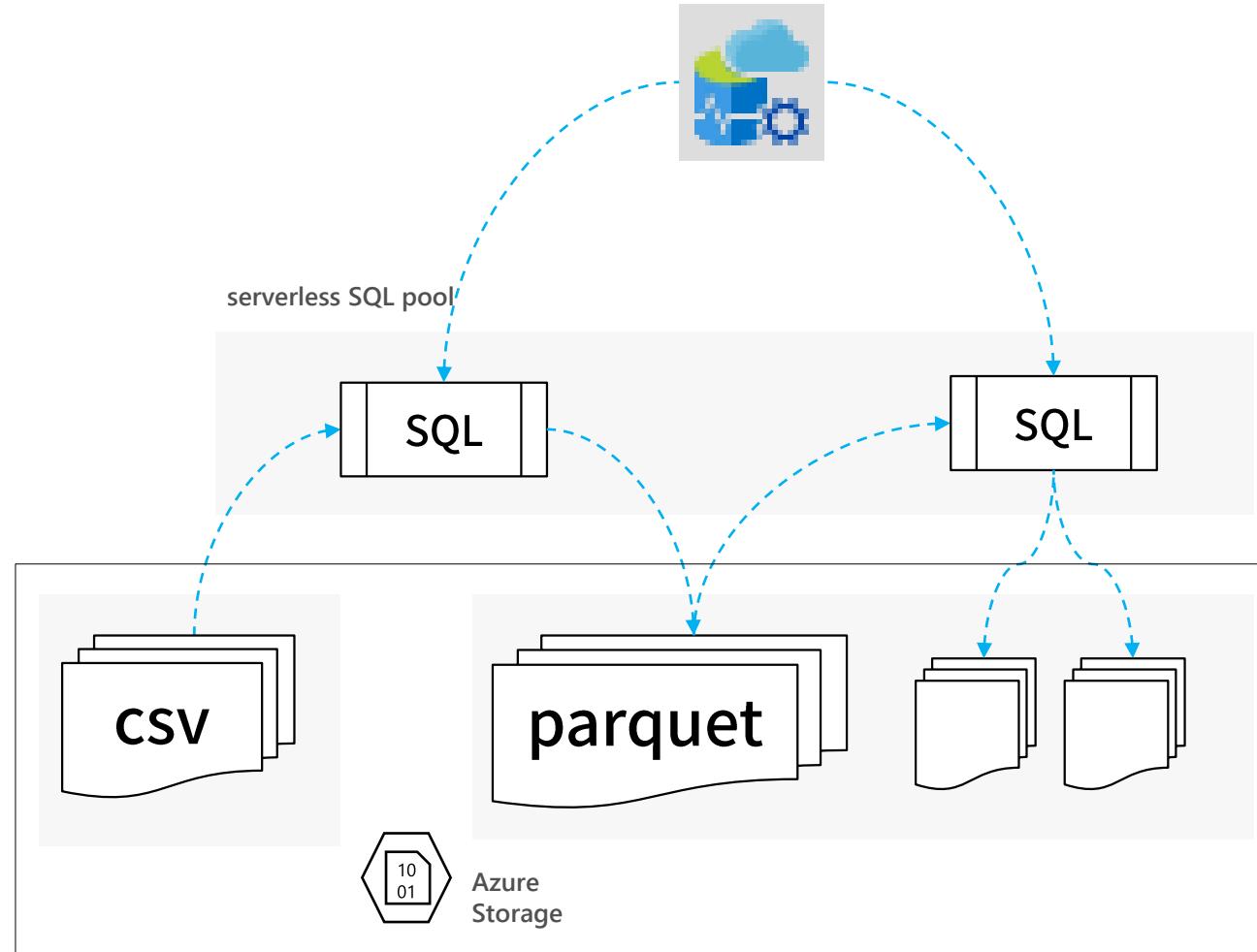
Benefits

Single statement transformations:

- convert CSV or JSON files to Parquet
- copy files from one storage account to another
- re-partition data to new location(s)
- store results of your query on Azure Storage

SQL ETL pipelines

- Use SQL commands to transform data
- Chain SQL statement for build ETL process
- Materialize reports created on the current snapshot of data



데이터 포맷 변환 with CETAS

Overview

Create External Tables As Select (CETAS) 를 이용하여 쉬운 데이터 포맷 변환이 가능하며 Azure storage 에 쿼리 결과를 저장할 수 있습니다.

Benefits

Select any data set and store it in parquet format.

Pre-calculate and store results of query and store them permanently on Azure storage.

Use saved data using external table.

Improve performance of your reports by permanently storing the result based on current snapshot of data as parquet files.

```
-- copy CSV dataset into parquet data set
CREATE EXTERNAL TABLE parquet.Population
WITH(
    LOCATION = '/parquet/population',
    DATA_SOURCE = MyAzureStorage,
    FILE_FORMAT = MyAzureParquetFormat )
AS
SELECT *
FROM csv.Population

-- pre-create report using new parquet data-set
CREATE EXTERNAL TABLE parquet.PopulationByMonth2017
WITH(
    LOCATION = '/parquet/population/bymonth/2017',
    DATA_SOURCE = MyAzureStorage,
    FILE_FORMAT = MyAzureParquetFormat )
AS
SELECT month = p.month, population = COUNT ( p.population )
FROM parquet.Population p
WHERE p.year = 2017
GROUP BY p.month

-- Reporting tools can now directly read data from pre-created report
SELECT *
FROM parquet.PopulationByMonth2017
```

Spark 테이블과의 자동 동기화

Overview

Spark Pool에서 테이블을 생성하면 Serverless SQL Pool에서 해당 테이블을 참조할 수 있는 Table이 자동 생성됩니다.

Benefits

Tables designed using Spark languages are immediately available in serverless SQL pool.

Schema definition matches original Spark table updates are applied in serverless SQL pool

No need to manually create SQL tables that match Spark tables

Spark and serverless SQL pool tables references the same external files.

The screenshot shows the Azure Data Studio interface. On the left is a sidebar with icons for Connections, Servers, Databases, Tables, Columns, Keys, Constraints, and External Tables. The 'Columns' icon is highlighted with a blue circle containing a white number '1'. The main area has two tabs: 'Cell 1' and 'Cell 2'. Cell 1 contains a code editor with the following SQL script:

```
1 %%sql
2 create table data1017 using parquet
3 location 'abfss://container@demostorage.dfs.core.windows.net/data/'
```

Cell 2 shows a query window titled 'SQLQuery_1 - sqlkon...oud!SA'. It contains the following SQL code:

```
1 SELECT TOP (10) [ExtractId]
2 , [DayOfWeekID]
3 , [DayOfWeekDescr]
4 , [DayOfWeekDescrShort]
5 , [ExtractDateTime]
6 , [LoadTS]
7 , [DeltaActionCode]
8 FROM [default]..[data1017]
```

The results pane shows the following data:

ExtractId	DayOfWeekID	DayOfWeekDescr	DayOfWeekDescrShort	ExtractDateTime
6b86b273ff34fce19d6b804eff5a...	1	Sunday	Sun	2020-01-22 00:00:00.000
d4735e3a265e16eee03f5a718b9b...	2	Monday	Mon	2020-01-22 00:00:00.000
4e07408562bedb8b60ce05c1aect...	3	Tuesday	Tue	2020-01-22 00:00:00.000
4b227777d4dd1fc61c6f884f4864...	4	Wednesday	Wed	2020-01-22 00:00:00.000
ef2d127de37b942baad06145e54b...	5	Thursday	Thu	2020-01-22 00:00:00.000
e7f6c011776e8db7cd330b54174f...	6	Friday	Fri	2020-01-22 00:00:00.000



Azure Synapse
Analytics
Apache Spark

Azure Synapse Apache Spark - 요약

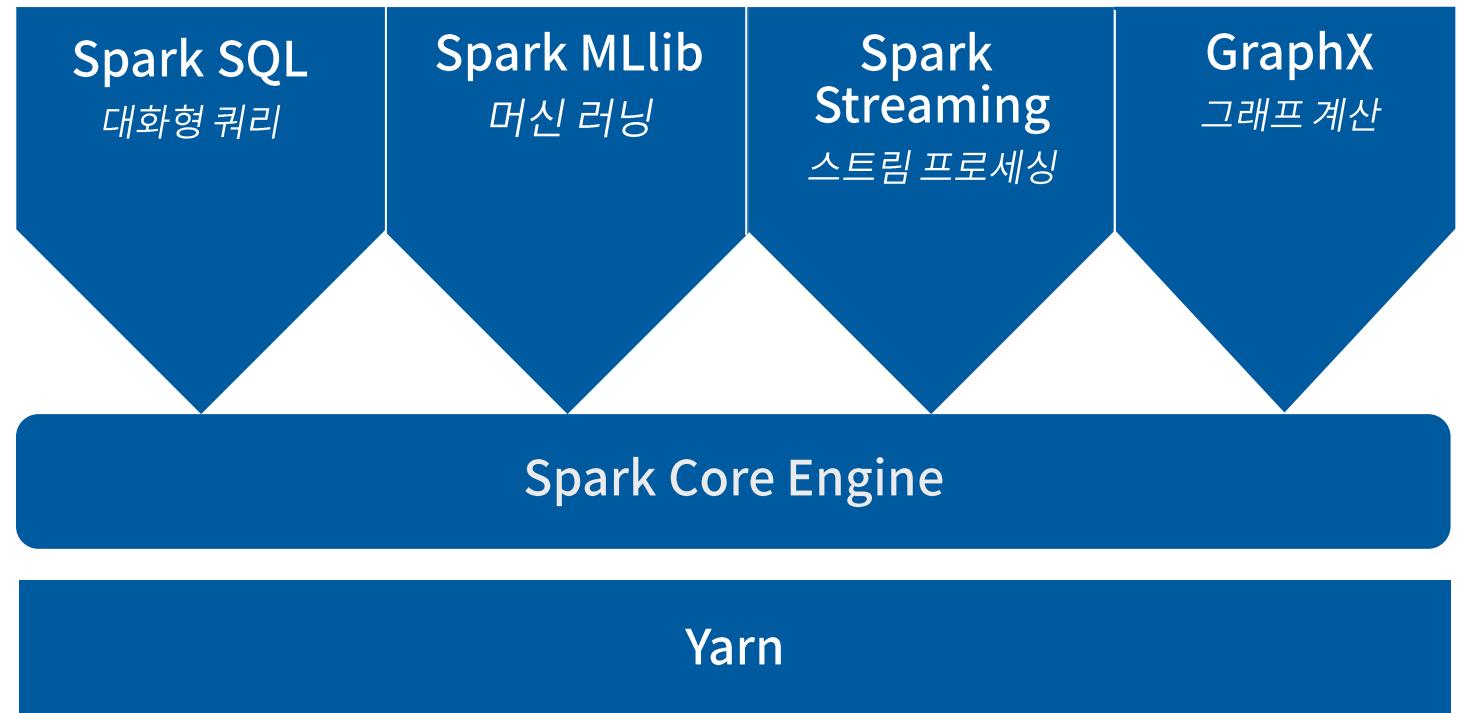
- **Apache Spark 3.2**
 - Delta Lake 1.2 지원
 - .Net Core 3.1 지원
 - Python 3.8 지원
- **Tightly coupled to other Azure Synapse services**
 - Integrated security and sign on
 - Integrated Metadata
 - Integrated and simplified provisioning
 - Integrated UX including interact based notebooks
 - Fast load of Synapse SQL (provisioned) pools
- **Core scenarios**
 - Data Prep/Data Engineering/ETL
 - Machine Learning via Spark ML and Azure ML integration
 - Extensible through library management
- **Efficient resource utilization**
 - Fast Start
 - Auto scale (up and down)
 - Auto pause
 - Min cluster size of 3 nodes
- **Multi Language Support**
 - .Net (C#), PySpark, Scala, Spark SQL, Java

Apache Spark

빅데이터 분석을 위한 통합 오픈소스 병렬 데이터 처리 프레임 워크

Spark Unifies:

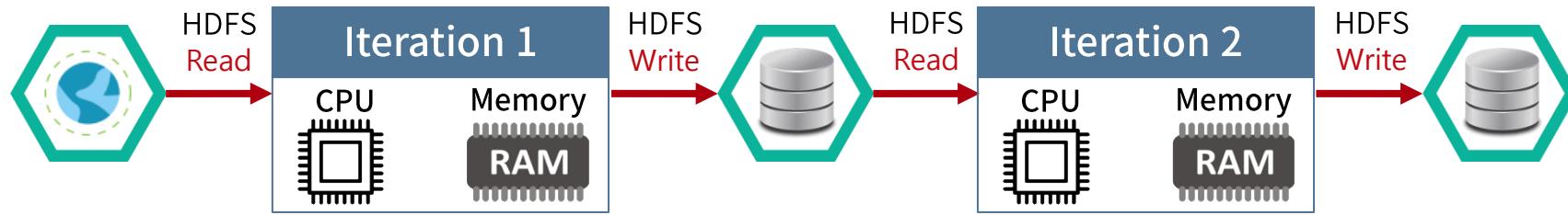
- 배치 처리
- Real-time 처리
- 대화형 쿼리
- 머신 러닝
- 딥 러닝
- 그래프 계산



<http://spark.apache.org>

Apache Spark

기존 접근 방식: 대화형 Query, Online event-hub 처리, 복잡한 작업을 위한 MapReduce jobs에
많은 디스크 I/O로 인한 속도가 저하됩니다.

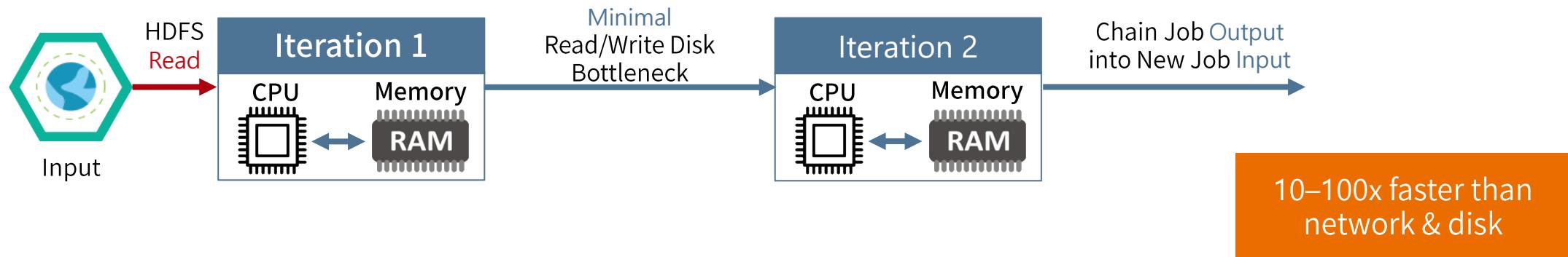


Apache Spark

기존 접근 방식: 대화형 Query, Online event-hub 처리, 복잡한 작업을 위한 MapReduce jobs에
많은 디스크 I/O로 인한 속도가 저하됩니다.

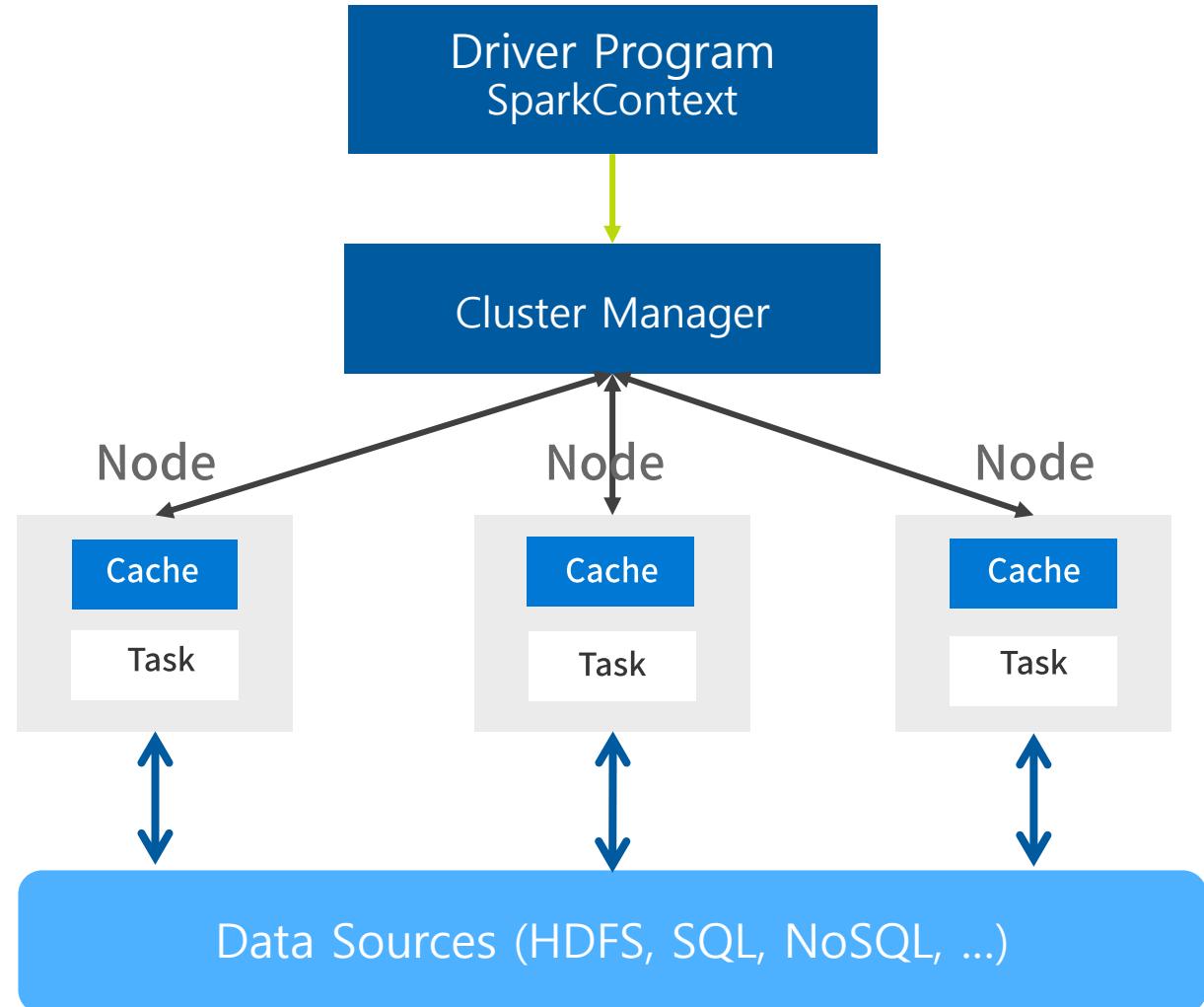


해결방안: 분산 처리 엔진에서 데이터를 메모리에 유지

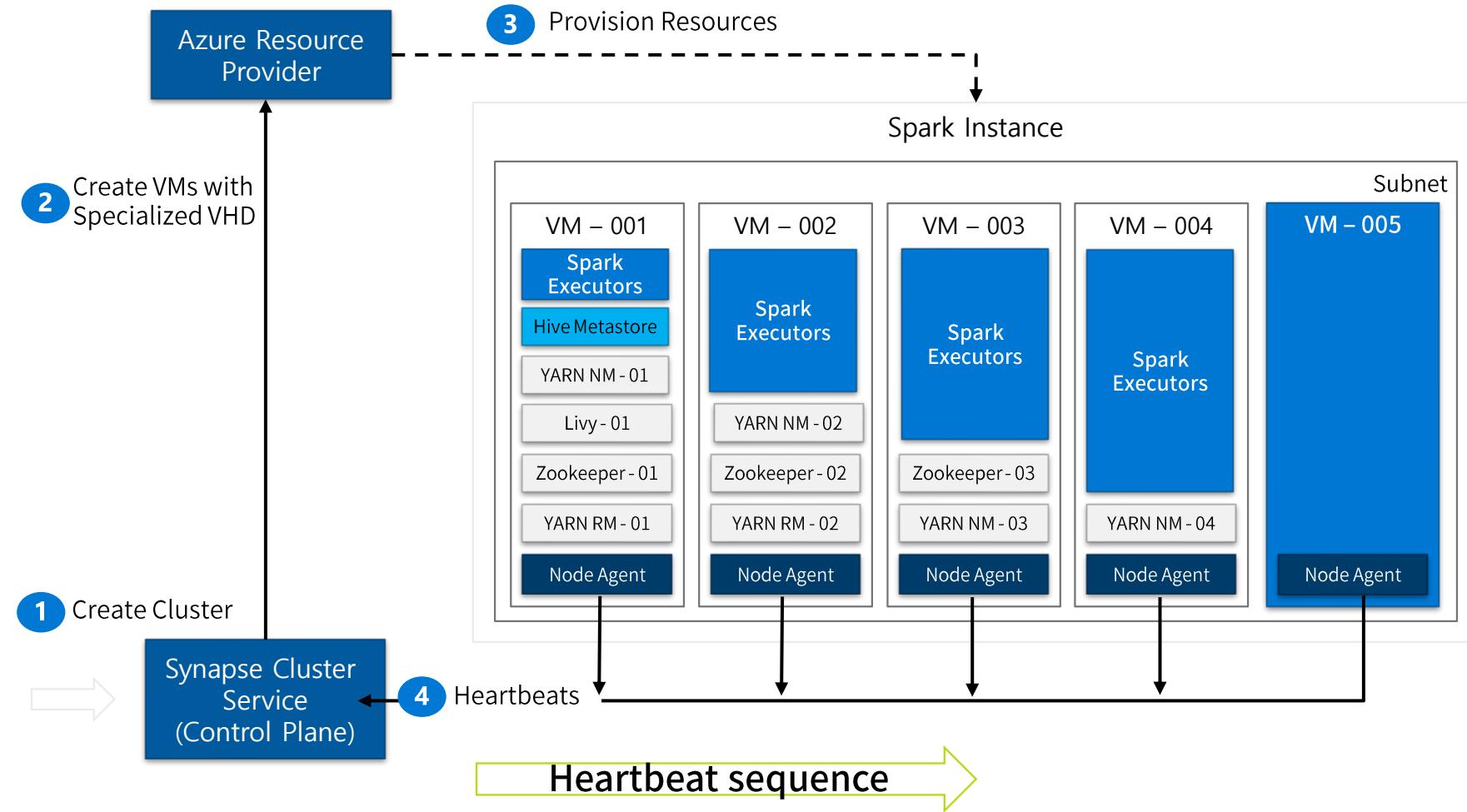


Apache Spark Architecture

- ‘Driver’는 사용자의 main() 함수를 구동하고, 다양한 병렬 작업들을 worker node에서 구동하게 합니다.
- 작업들의 결과는 ‘Driver’에 의해 취합되어 집니다.
- Worker node들은 HDFS, SQL과 같은 Data Source들로부터 데이터를 읽고 쓰게 됩니다.
- Worker node 변환된 데이터를 메모리에 RDD로 cache합니다.
- Worker node들과 Driver Node는 public cloud의 VM들과 같이 실행됩니다.



Synapse Spark Instance



1. Synapse Job Service는 Spark pool에 설명에 따라, Cluster Service에게 cluster 생성 요청을 보냅니다.
2. Cluster Service는 Azure SDK를 사용하여 specialized VHD를 이용하여 VM을 만들도록 Azure에 요청합니다.
3. Specialized VHD는 cluster 타입(e.g. Spark)에 의해 필요한 모든 서비스들이 포함되어져 있습니다.
4. VM이 부팅 되면, Node Agent는 heartbeat을 Cluster Service로 보내고 node configuration을 받습니다.
5. 첫번째 heartbeat을 기반으로 하여, node들은 초기화되고, role들이 assign 됩니다.
6. extra 노드들은 첫번째 heartbeat에서 삭제됩니다.
7. Cluster Service가 cluster가 준비되었다고 판단될 때, livy endpoint를 Job Service에게 반환합니다.

스토리지에 있는 파일을 이용한 Notebook 생성

The screenshot illustrates the Microsoft Azure Synapse Analytics workspace interface. On the left, the navigation pane includes Home, Data (selected), Develop, Orchestrate, Monitor, and Manage. The main area shows a dataset named 'nyctic' under the 'Data' section. A context menu is open over the dataset, with the 'New notebook' option highlighted by a red box and a red arrow pointing to the bottom right. The bottom half of the screen displays a notebook titled 'Notebook 4' with the following content:

```
%%pyspark
data_path = spark.read.load('abfss://nyctic@prlangaddemoa.dfs.core.windows.net/yellow/puYear=2015/puMonth=3/part-00133-tid-210938564719836543-aea5b543-5e83-')
data_path.show(10)
```

Below the code, it says "Command executed in 3mins 59s 249ms by prlangad on 11-14-2019 09:57:11.863 -08:00".

The notebook also shows a table of job execution status:

ID	DESCRIPTION	STATUS	STAGES	TASKS	SUBMISSION TIME	DURATION
Job 0	load at NativeMethodAccessorImpl.java:0	Succeeded	1/1	1/1	11/14/2019, 9:56:49 AM	7s
Job 1	showString at NativeMethodAccessorImpl.java:0	Succeeded	1/1	1/1	11/14/2019, 9:56:58 AM	1s
Job 2	showString at NativeMethodAccessorImpl.java:0	Succeeded	1/1	1/1	11/14/2019, 9:56:59 AM	11s

At the bottom, a preview of the taxi trip data is shown:

vendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passengerCount	tripDistance	puLocationId	doLocationId	startStation	startLat	endStation	endLat
2	2015-02-28 23:53:18	2015-03-01 00:00:29	6	1.63	null	null	-74.00084686279297	40.73069381713867	-73.9841537475586	40.74470520019531
1	N	1	7.5	0.5	0.5	0.3	1.76	0.0	10.56	
1	2015-03-28 19:21:05	2015-03-28 19:28:31	1	2.2	null	null	-73.9776530341797	40.763160705566406	-73.95502471923828	40.78600311279297
1	N	1	8.5	0.0	0.5	0.3	2.3	0.0	11.6	
1	2015-02-28 23:53:19	2015-03-01 00:12:08	5	3.23	null	null	-73.96012878417969	40.76215744018555	-73.9881591796875	40.728118896484375
1	N	1	14.5	0.5	0.5	0.3	4.74	0.0	20.54	
1	2015-03-28 19:21:05	2015-03-28 19:37:02	1	2.1	null	null	-73.98143005371094	40.7815055847168	-74.00091552734375	40.76177215576172

스토리지에 있는 파일을 이용한 Notebook 생성

The screenshot shows the Microsoft Azure Synapse Analytics Notebook interface. The left sidebar lists notebooks, with "SeattleSafetyDoc" selected. The main area displays a notebook with four cells:

- Cell 1:** Contains Python code for reading data from Azure Blob Storage using PySpark. It includes a command history entry and a table showing job execution details for Job 0.
- Cell 2:** Contains a single command: `seasafety_df.createOrReplaceTempView('seattlesafety')`.
- Cell 3:** Contains a command: `display(spark.sql('SELECT * FROM seattlesafety LIMIT 10'))`. Below it is a table view of the data, with the "Table" tab selected. The table has columns: dataType, dataSubtype, dateTime, category, address, latitude, and longitude. The data shows various emergency response events, primarily 911_Fire incidents.
- Cell 4:** Contains a command: `seasafety_df.coalesce(1).write.csv('abfss://default@euangsynapsenovstorage.dfs.core.windows.net/demodata/seattlesafety', mode='overwrite')`.

A red annotation on the left side points to the table view in Cell 3 with the text "View results in table format".

dataType	dataSubtype	dateTime	category	address	latitude	longitude
Safety	911_Fire	2011-03-04T10:00:26.000Z	Aid Response	517 3rd Av	47.602172	-122.330863
Safety	911_Fire	2015-06-08T02:59:35.000Z	Trans to AMR	10044 65th Av S	47.511314	-122.252346
Safety	911_Fire	2015-06-08T21:10:52.000Z	Aid Response	Aurora Av N / N 125th St	47.719572	-122.344937
Safety	911_Fire	2007-09-17T13:03:34.000Z	Medic Response	1st Av N / Republican St	47.623272	-122.355415
Safety	911_Fire	2007-11-19T17:46:57.000Z	Aid Response	7724 Ridge Dr Ne	47.684393	-122.275254
Safety	911_Fire	2008-06-15T14:32:33.000Z	Medic Response	6940 62nd Av Ne	47.678789	-122.262227
Safety	911_Fire	2007-06-18T23:05:58.000Z	Medic Response	5107 S Myrtle St	47.538902	-122.268825
Safety	911_Fire	2005-06-06T19:23:10.000Z	Aid Response	532 Belmont Av E	47.623505	-122.324033
Safety	911_Fire	2017-03-06T19:45:36.000Z	Trans to AMR	610 1st Av N	47.624659	-122.355403
Safety	911_Fire	2017-06-23T18:21:21.000Z	Automatic Fire Alarm Resd	7711 8th Av NW	47.685137	-122.366006

스토리지에 있는 파일을 이용한 Notebook 생성

The screenshot shows the Microsoft Azure Synapse Analytics Notebook interface. The top navigation bar includes 'Microsoft Azure', 'Synapse Analytics', 'euang-synapse-nov-ws', 'Search resources', and an email icon for 'euang@microsoft.com'. The left sidebar lists notebooks such as '00_DataPrep', '01_TrainingUseMllib_cleanup', 'automl_arclada_validate', 'Data Download_GreenCab', 'Data Download_HolidayData', 'Data Download_Weather', 'Data Download_YellowCab', 'Explore_Join_Aggregate', 'NYCTaxi_Docs_Final', 'NYCTaxi_Docs_Final_PySpark', 'Repro', 'SeattleSafetyDoc' (which is selected), and 'SparkPerf'. The main area has tabs for 'Data Download...', 'NYCTaxi_Docs...', 'SeattleSafetyD...', and 'Repro...'. A red box highlights the 'Language' dropdown set to 'PySpark (Python)'. Below it, 'Cell 1' contains Python code for reading data from Azure Blob Storage. A red arrow points from the 'SQL support' text to 'Cell 2', which shows a SQL command: `seasafety_df.createOrReplaceTempView('seattlesafety')`. Cell 3 displays the results of a query: `display(spark.sql('SELECT * FROM seattlesafety'))`, resulting in a pie chart titled 'Aid Response' with various categories like 'Automatic Fire Alarm False', 'Medic Response', etc. A red arrow points from the 'View results in chart format' text to this chart. Cell 4 contains the command: `seasafety_df.coalesce(1).write.csv('abfss://default@euangsynapsenovstorage.dfs.core.windows.net/demodata/seattlesafety', mode='overwrite')`. The bottom right of the interface shows chart configuration settings.

View results in chart format

SQL support

스토리지에 있는 파일을 이용한 Notebook 생성

The screenshot shows a Jupyter Notebook interface with the following details:

- Left Sidebar (Develop):** Shows a tree view of notebooks, with **NYTaxi_Docs_Final** selected.
- Top Bar:** Includes buttons for Publish all, Validate all, Refresh, Discard all, and a Data Download... button.
- Code Cell (Cell 9):** Contains Python code for data manipulation and visualization. The code includes:
 - Creating a temp table from a sampled DataFrame.
 - Plotting a histogram of tip amounts.
 - Creating a boxplot of tip amounts by passenger count.
 - Plotting fare amount vs tip amount.
- Exploratory Data Analysis Section:** A descriptive text block: "Look at the data and evaluate its suitability for use in a model, do this via some basic charts focussed on tip values and relationships."
- Plots:** Three plots are displayed:
 - Tip amount distribution:** A histogram showing the frequency of tip amounts. The x-axis ranges from 0 to 25, and the y-axis (Counts) ranges from 0 to 800. An arrow points from this plot to the text "Exploratory data analysis with graphs – histogram, boxplot etc".
 - Tip amount by Passenger count:** A boxplot showing the distribution of tip amounts for different numbers of passengers (0 to 6). The x-axis is "Passenger count" and the y-axis is "Tip Amount (\$)".
 - Tip amount by Fare amount:** A scatter plot showing the relationship between fare amount and tip amount. The x-axis is "Fare amount" and the y-axis is "Tip Amount (\$)".
- Bottom Bar:** Includes "Not Started" and "Configure session" buttons.

Exploratory data analysis with graphs – histogram, boxplot etc

Q & A

Thank you