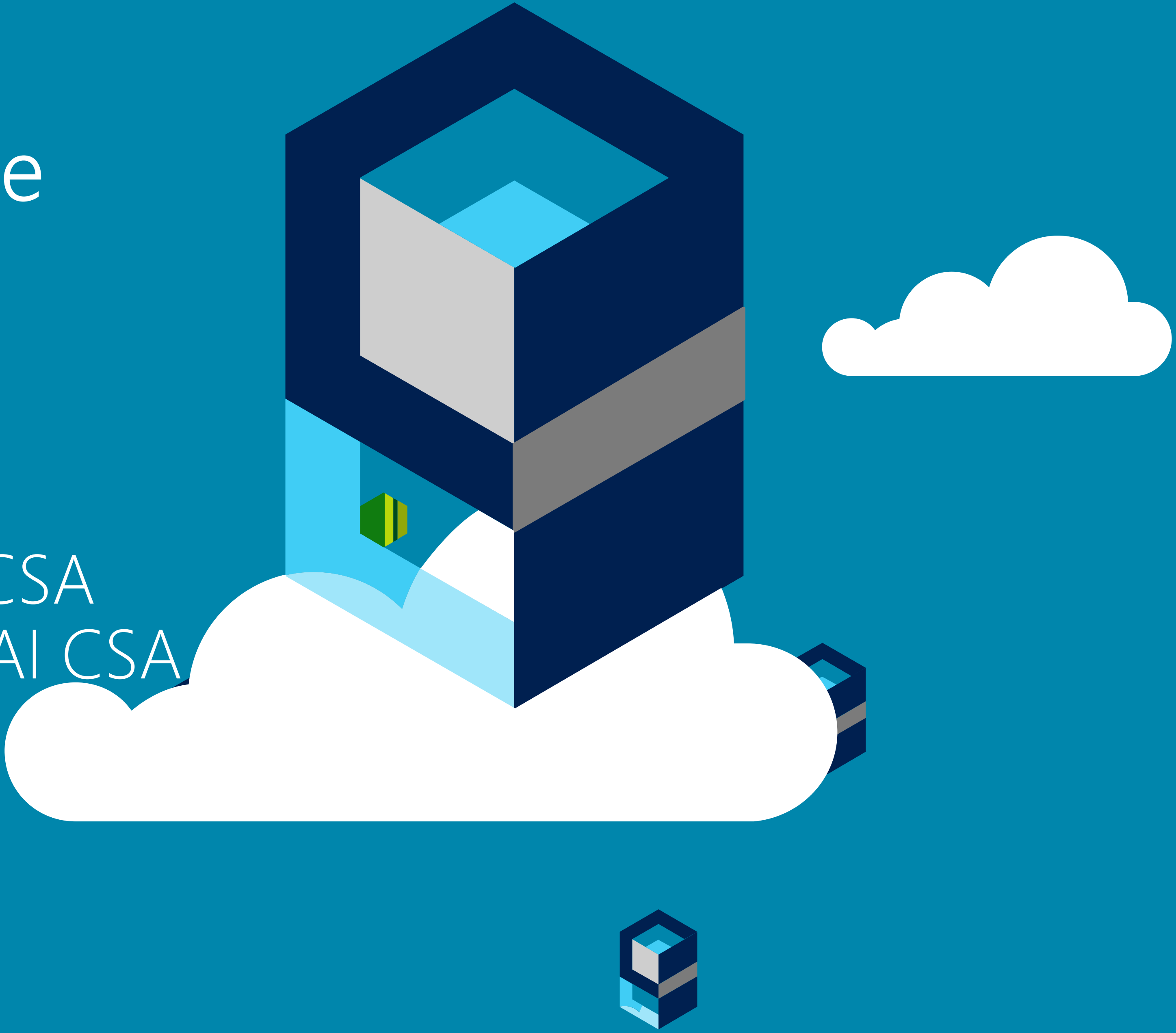
Microsoft

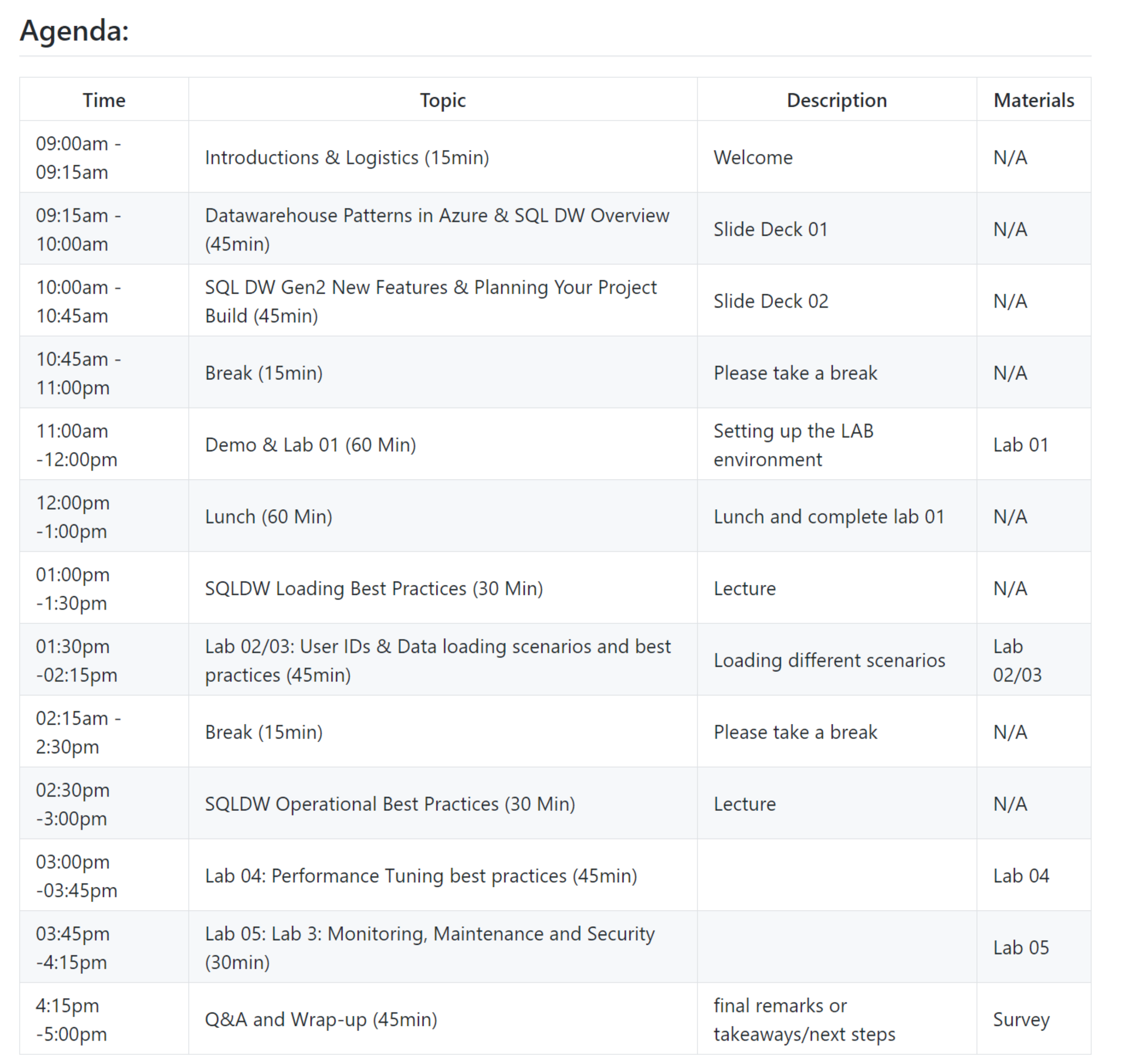
02. SQL DW Gen-2 – Planning your Build

Azure Data Warehouse
In-A-Day

Steve Young - Data & AI CSA
Rebecca Young - Data & AI CSA



Agenda



Agenda:

Time	Topic	Description	Materials
09:00am - 09:15am	Introductions & Logistics (15min)	Welcome	N/A
09:15am - 10:00am	Datawarehouse Patterns in Azure & SQL DW Overview (45min)	Slide Deck 01	N/A
10:00am - 10:45am	SQL DW Gen2 New Features & Planning Your Project Build (45min)	Slide Deck 02	N/A
10:45am - 11:00pm	Break (15min)	Please take a break	N/A
11:00am -12:00pm	Demo & Lab 01 (60 Min)	Setting up the LAB environment	Lab 01
12:00pm -1:00pm	Lunch (60 Min)	Lunch and complete lab 01	N/A
01:00pm -1:30pm	SQLDW Loading Best Practices (30 Min)	Lecture	N/A
01:30pm -02:15pm	Lab 02/03: User IDs & Data loading scenarios and best practices (45min)	Loading different scenarios	Lab 02/03
02:15am - 2:30pm	Break (15min)	Please take a break	N/A
02:30pm -3:00pm	SQLDW Operational Best Practices (30 Min)	Lecture	N/A
03:00pm -03:45pm	Lab 04: Performance Tuning best practices (45min)		Lab 04
03:45pm -4:15pm	Lab 05: Lab 3: Monitoring, Maintenance and Security (30min)		Lab 05
4:15pm -5:00pm	Q&A and Wrap-up (45min)	final remarks or takeaways/next steps	Survey

Azure SQL Data Warehouse Gen 2 - Performance Features

Data Movement

https://azure.microsoft.com/en-us/blog/smarter-faster-safer-azure-sql-data-warehouse-is-simply-unmatched/

Data Movement - Instant Data Movement (IDM)

Instant Data Movement (IDM) - extremely efficient movement between data warehouse compute nodes.

At the heart of every distributed database system is the **need to align two or more tables**

That are partitioned on a different key to produce a final or intermediate result set.

Distributed Data Movement

ProductSales

SalesAccountTerritory

	AccountID	SalesAmt	…	SATerritoryID	AccountID	…
Node 1:	47	$1,234.36	…	444	37	…
Node 2:	36	$2,345.47	…	333	25	
Node 3:	14	$3,456.58	…	111	36	…
Node 4:	25	$4,567.69	…	222	47	…
Node 5:	48	$5,678.70	…	445	14	
Node 6:	37	$6,789.81	…	334	48	…
	…	…	…	…	…	…

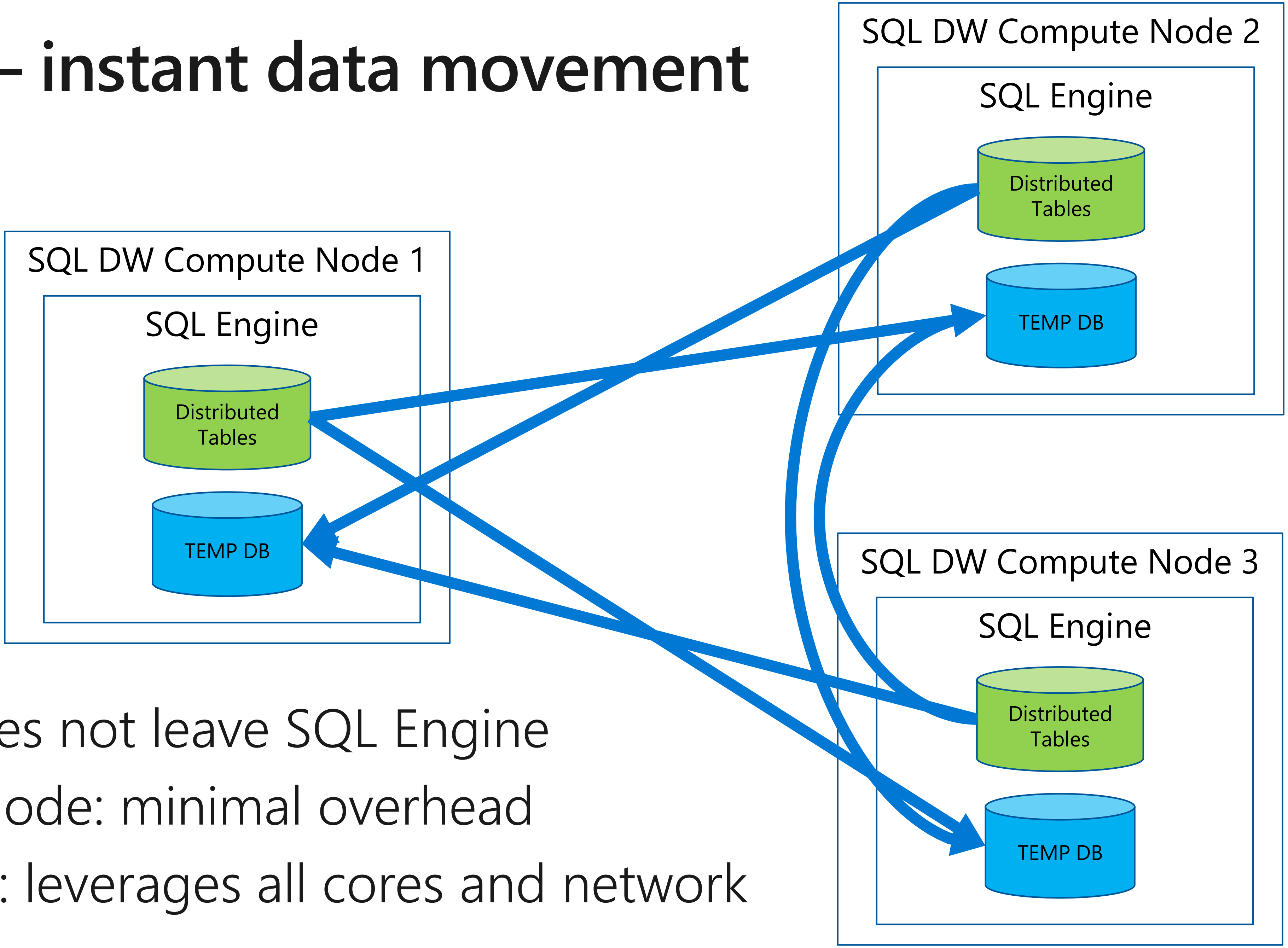
```
CREATE TABLE ProductSales
WITH (DISTRIBUTION=HASH(AccountID))
AS...

CREATE TABLE SalesAccountTerritory
WITH (DISTRIBUTION=HASH(SATerritoryID))
AS...

SELECT TOP 25 a.SalesAccountTerritoryName
      ,TotalSales = SUM(p.SalesAmt)
FROM ProductSales p
JOIN SalesAccountTerritory a
ON   a.AccountID = p.AccountID
GROUP BY a.SalesAccountTerritoryName
ORDER BY 2 DESC
```

Shuffle

D	SATName	…
	West	…
	East	
	SouthWest	…
	NorthEast	…
	South	
37	North	…
…	…	…

SATName	TotalSales
North	$6,789.81
South	$5,678.70
NorthEast	$4,567.69
SouthWest	$3,456.58
East	$2,345.47
West	$1,234.36

Gen2 – instant data movement

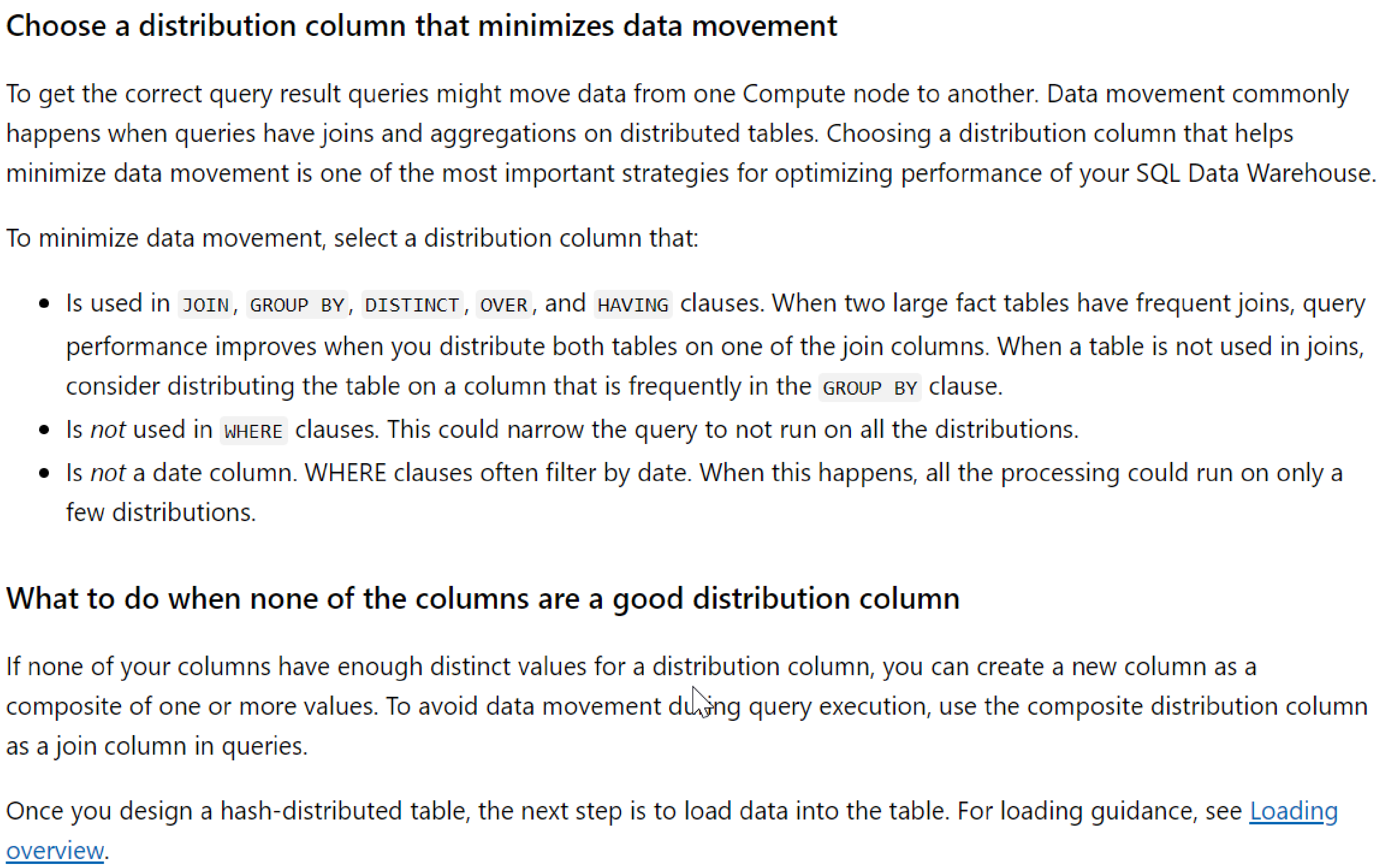


Data does not leave SQL Engine

Batch-mode: minimal overhead

Scalable: leverages all cores and network

Distribution Column



Choose a distribution column that minimizes data movement

To get the correct query result queries might move data from one Compute node to another. Data movement commonly happens when queries have joins and aggregations on distributed tables. Choosing a distribution column that helps minimize data movement is one of the most important strategies for optimizing performance of your SQL Data Warehouse.

To minimize data movement, select a distribution column that:

- Is used in `JOIN`, `GROUP BY`, `DISTINCT`, `OVER`, and `HAVING` clauses. When two large fact tables have frequent joins, query performance improves when you distribute both tables on one of the join columns. When a table is not used in joins, consider distributing the table on a column that is frequently in the `GROUP BY` clause.
- Is *not* used in `WHERE` clauses. This could narrow the query to not run on all the distributions.
- Is *not* a date column. WHERE clauses often filter by date. When this happens, all the processing could run on only a few distributions.

What to do when none of the columns are a good distribution column

If none of your columns have enough distinct values for a distribution column, you can create a new column as a composite of one or more values. To avoid data movement during query execution, use the composite distribution column as a join column in queries.

Once you design a hash-distributed table, the next step is to load data into the table. For loading guidance, see Loading overview.

Resource Classes

Gen2 – Simpler resource model

With Gen2, dynamic resource pools were introduced with a 3-10-22-70 model for resource allocations.

Resource Class	Percent Resources	Concurrency
SmallRc	3%	32
MediumRc	10%	10
LargeRc	22%	4
XLargeRc	70%	1

```
EXEC sp_addrolemember 'largerc', 'loaduser';
```

Gen2 – Simpler model

At MediumRc, an example…

MediumRC		Gen1 Model		Gen2 Model	
Service Level	Total Slots	Slots	Concurrency	Slots	Concurrency
DW1000c	40	8	5	4	10
DW1500c	60	8	7	6	10
DW2000c	80	16	5	8	10
DW2500c	100	16	6	10	10
DW3000c	120	16	7	12	10
DW5000c	200	32	6	20	10
DW6000c	240	32	7	24	10
DW7500c	300	64	4	30	10
DW10000c	400	64	6	40	10
DW15000c	600	64	9	60	10
DW30000c	1200	64	18	120	10

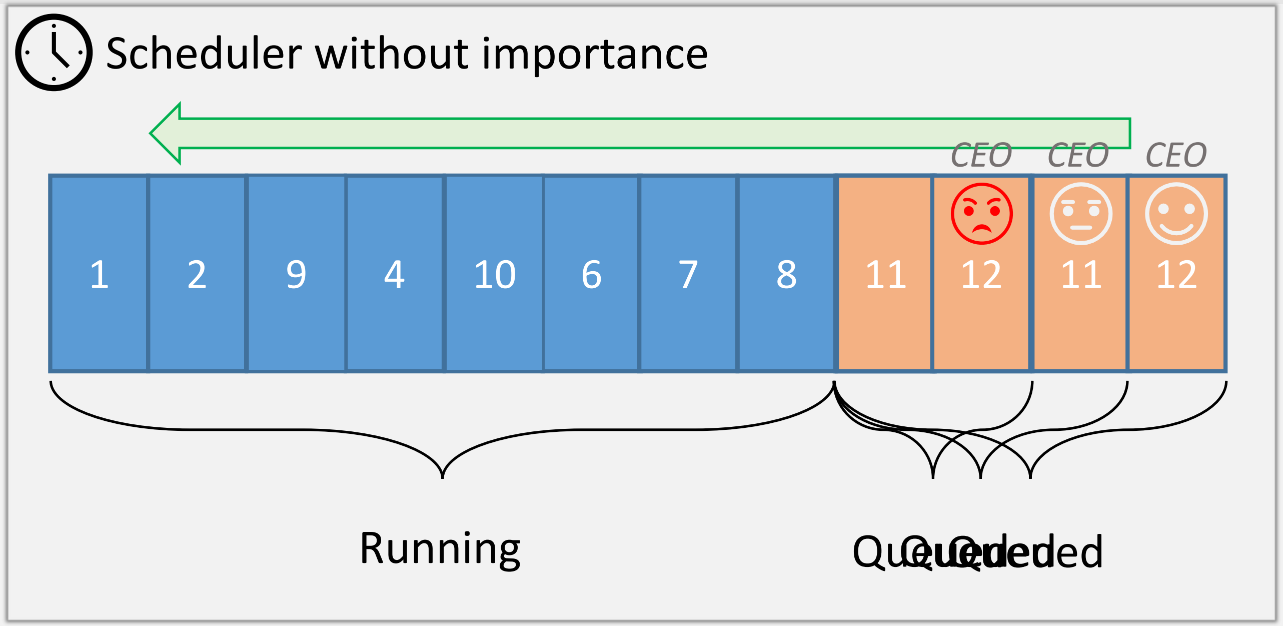
Workload Management

WORKLOAD IMPORTANCE – NO IMPORTANCE

Overview

Workload importance allows you to prioritize the queries that get access to resources.

It helps ensure that high-business value work is executed first on a busy data warehouse.


Scheduler without importance

WORKLOAD CLASSIFICATION

Overview

Allows you to map a query to an allocation of resources via pre-determined rules

Use this in combination with workload importance to effectively share resources across different workload types

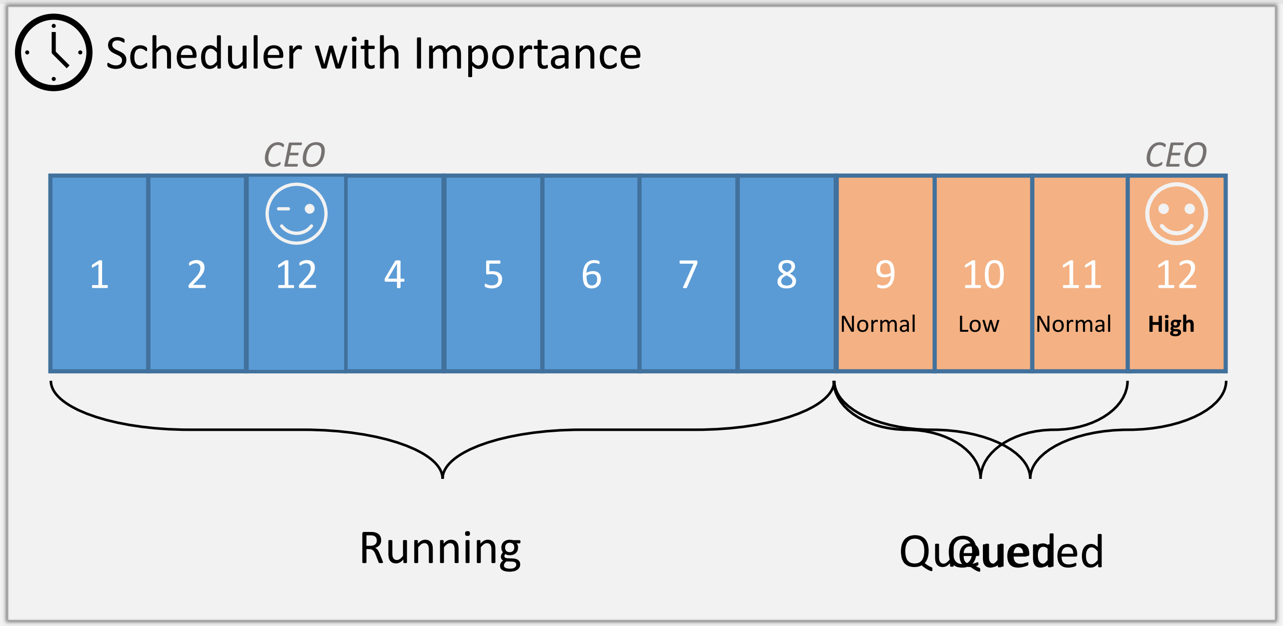
```
CREATE WORKLOAD CLASSIFIER classifier_name
WITH
(
   [WORKLOAD_GROUP = '<Resource Class>' ]
   [IMPORTANCE = {    LOW            |
                      BELOW_NORMAL   |
                      NORMAL         |
                      ABOVE_NORMAL   |
                      HIGH
                 }
   ]
   [MEMBERNAME = ‘security_account’]
)
```

WORKLOAD IMPORTANCE –IMPORTANCE

Overview

Workload importance allows you to prioritize the queries that get access to resources.

It helps ensure that high-business value work is executed first on a busy data warehouse.



Scheduler with Importance

WORKLOAD ISOLATION (PREVIEW)

Overview

Isolation allocates fixed resources to workloads within a data warehouse. These limits are strictly enforced for memory, and only enforced under load for CPU and IO.

Resource classes are implemented by assigning users to database roles. When a user runs a query, the query runs with the user's resource class

EXEC sp_addrolemember 'largerc', 'loaduser’;

EXEC sp_droprolemember 'largerc', 'loaduser';

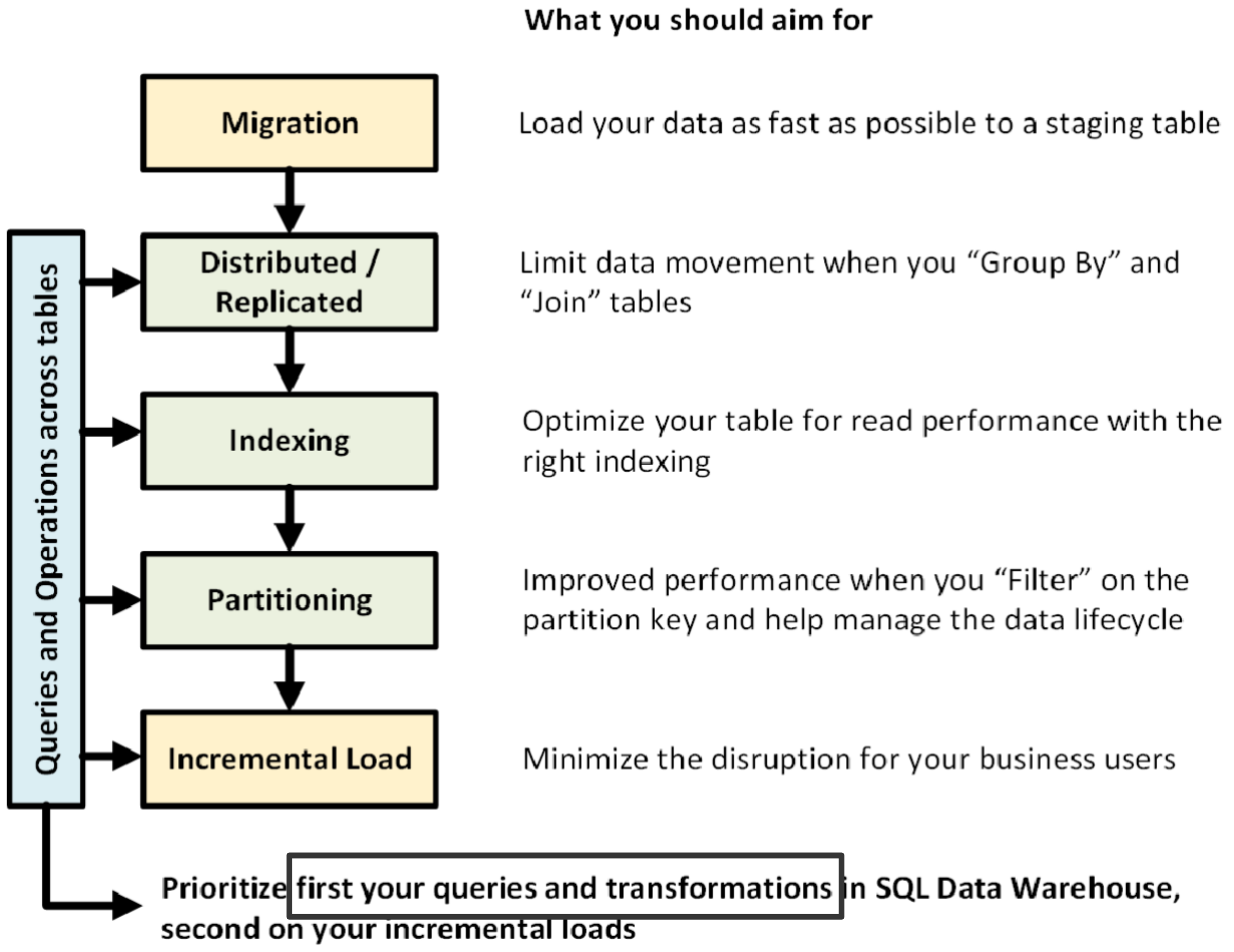
```
CREATE WORKLOAD GROUP group_name
WITH
(
    [ MIN_PERCENTAGE_RESOURCE = value ]
    [ CAP_PERCENTAGE_RESOURCE = value ]
    [ MAX_CONCURRENCY = value ]
)
```

https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-workload-classification

DW Creation Framework Architecture Decisions

DW Creation Framework Architecture Decisions

“Improve your probability of Success!!!!!”



What you should aim for

Migration – Load your data as fast as possible to a staging table

Distributed / Replicated – Limit data movement when you “Group By” and “Join” tables

Indexing – Optimize your table for read performance with the right indexing

Partitioning – Improved performance when you “Filter” on the partition key and help manage the data lifecycle

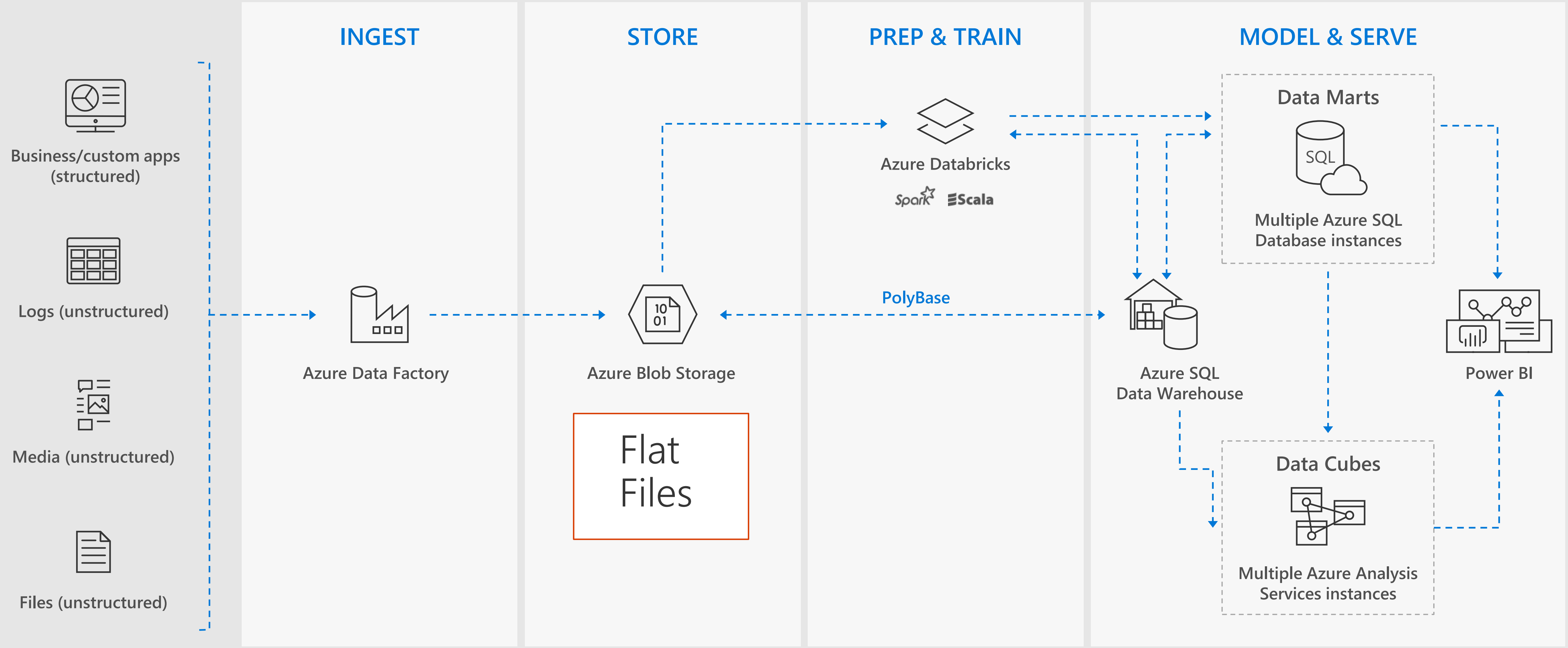
Incremental Load – Minimize the disruption for your business users

Queries and Operations across tables

Prioritize first your queries and transformations in SQL Data Warehouse, second on your incremental loads

https://docs.microsoft.com/en-us/azure/sql-data-warehouse/cheat-sheet

Hub & Spoke Architecture for Analytics (BI)



Microsoft Azure supports other services like Azure HDInsight and Azure Data Lake in various layers to allow customers a truly customized solution.

Choose a Light House Project

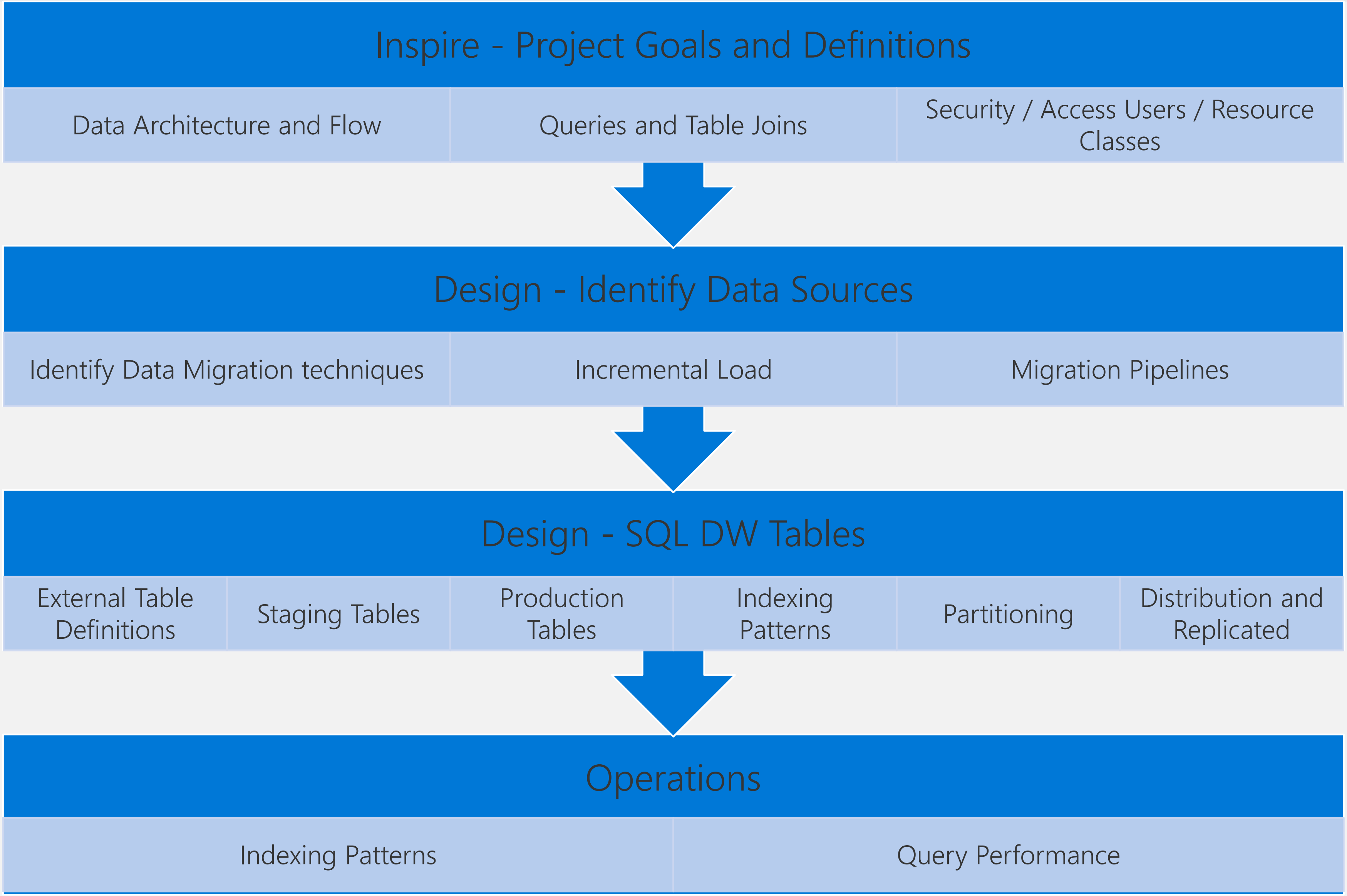
Don’t let your first project be your last !!!

- Get Comfortable with the technology – Kick the tires
- Take a smaller project, treat like a production roll out.
- Provides insights not only about the technology, but the organization’s maturity in being able to do a data project
  - Can the Input Files be generated, Tested, Errors traced
  - Handle change management
  - If you don’t measure, how do you know if you succeeded? Failed?

This Photo by Unknown Author is licensed under CC BY-SA

DW Creation Framework

DATA WAREHOUSE DESIGN WORKFLOW



Inspire - Project Goals and Definitions

- Data Architecture and Flow
- Queries and Table Joins
- Security / Access Users / Resource Classes

Design - Identify Data Sources

- Identify Data Migration techniques
- Incremental Load
- Migration Pipelines

Design - SQL DW Tables

- External Table Definitions
- Staging Tables
- Production Tables
- Indexing Patterns
- Partitioning
- Distribution and Replicated

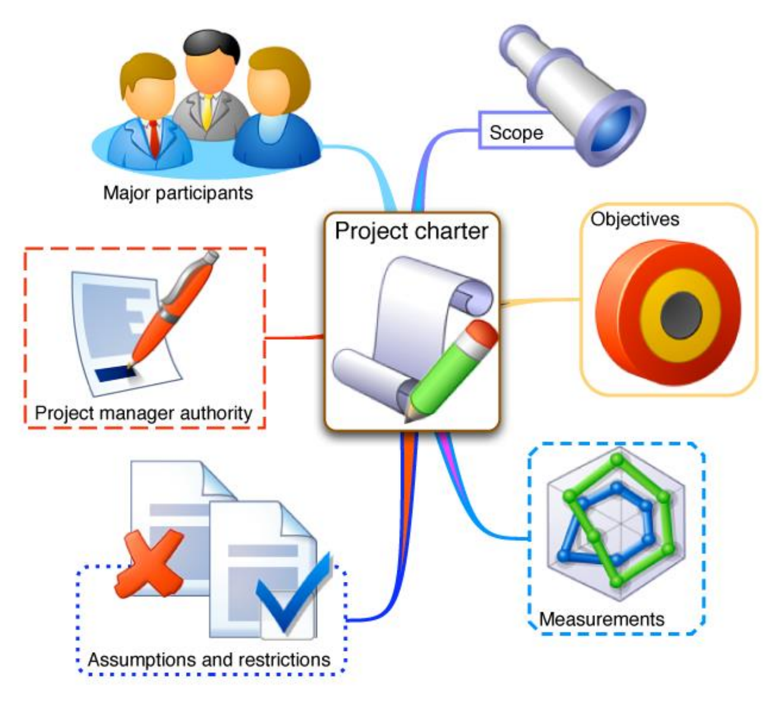
Operations

- Indexing Patterns
- Query Performance

WORKFLOW – PROJECT GOALS AND DEFINITIONS

Imagine a future state solution and develop use cases and query patterns

- Data Architecture and Flow - Identification
  - What sources are to be used
  - What format should they take
  - What is the update pattern (Incremental, One Time, Reload)
  - Does “history” change?
- Queries and Table Joins
  - Capture existing queries, reports and ad-hoc analysis
  - What queries are important?
  - Priorities (Resource Classes)
- Security / Access Users

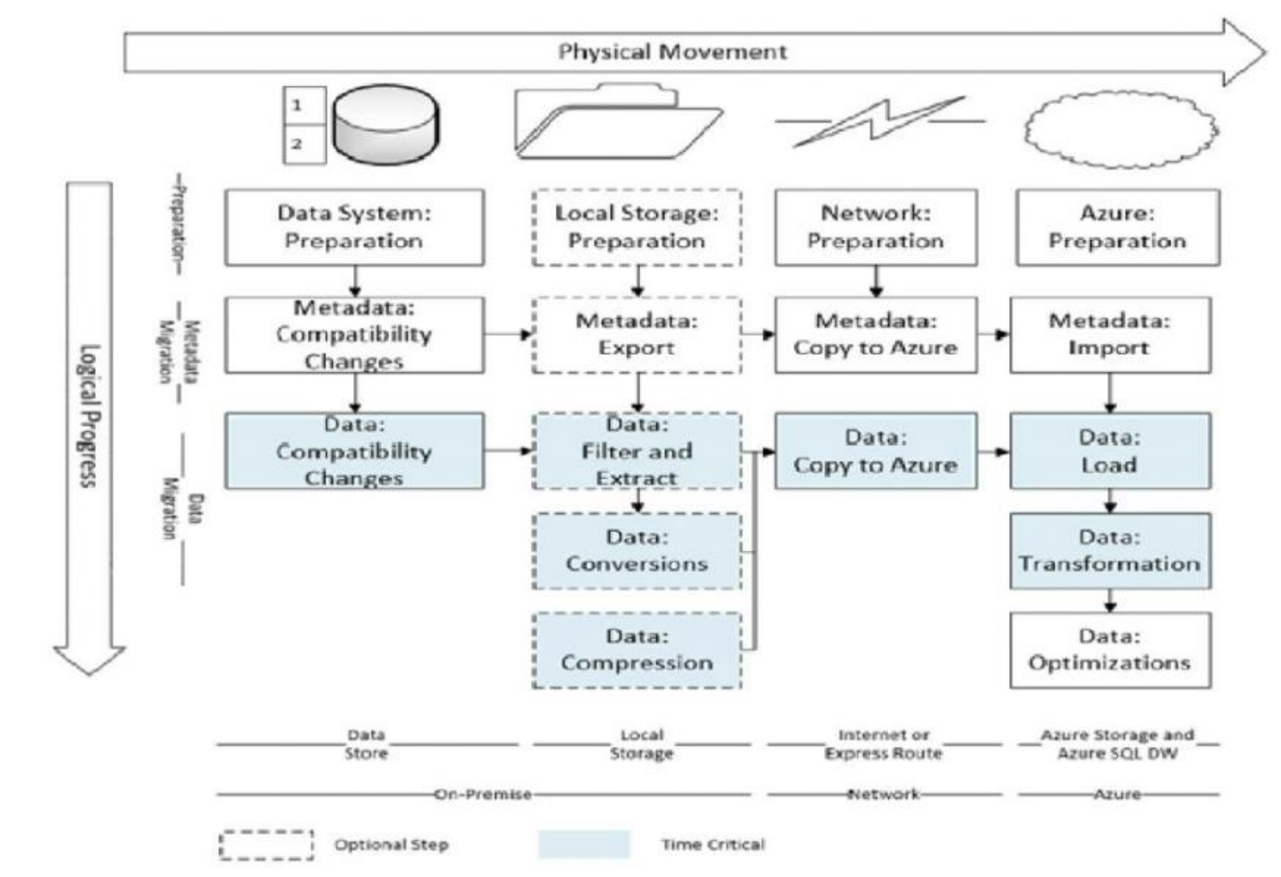


This Photo by Unknown Author is licensed under CC BY-SA-NC

WORKFLOW – IDENTIFY DATA SOURCES

Imagine a future state solution and develop use cases and query patterns

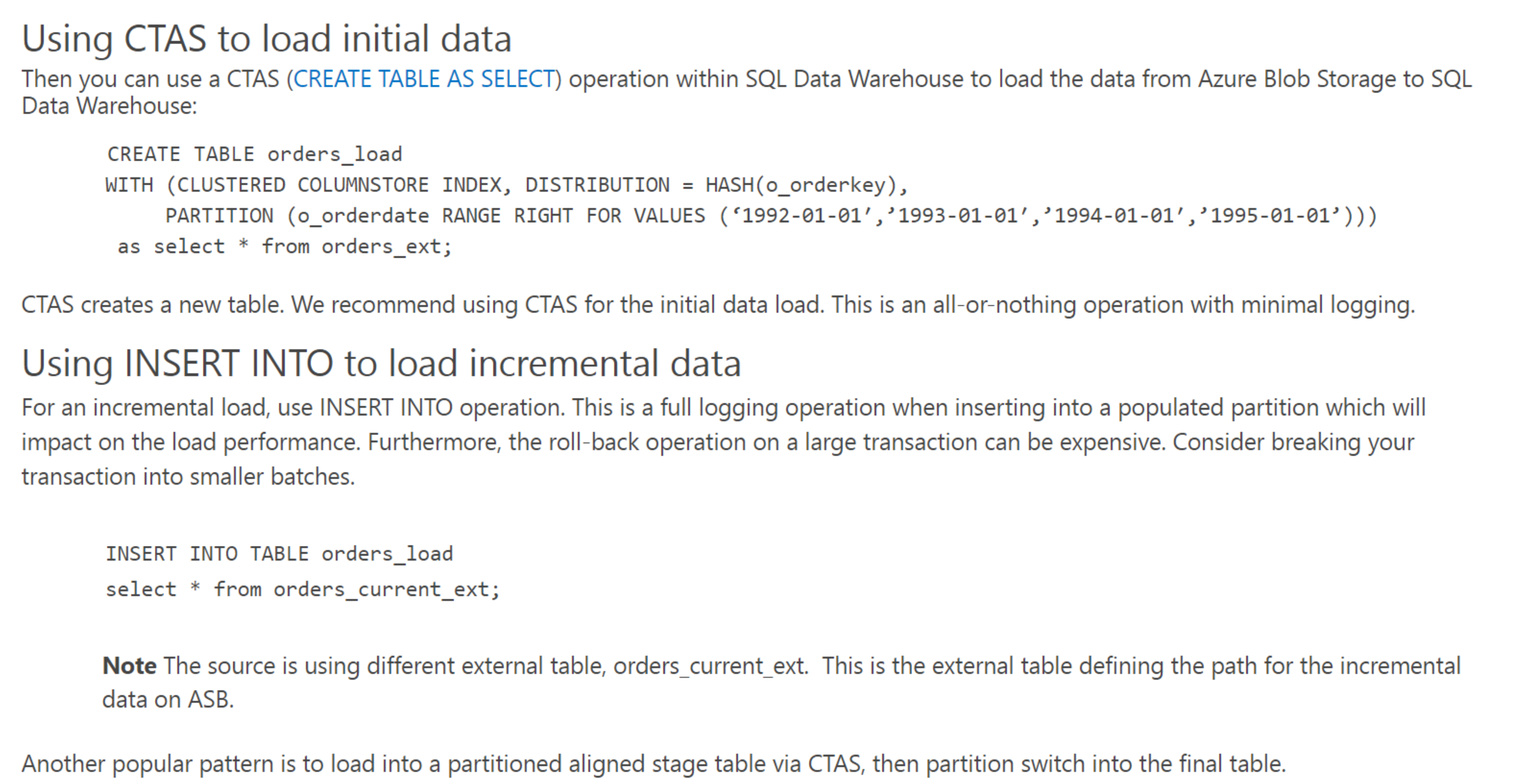
- Identify Data Migration tasks
  - File formats
  - Update schedule
- Incremental Load
  - Are these required?
- Migration Pipelines
  - Azure Data Factory
  - Azure Data Lake / Blob storage



https://blogs.msdn.microsoft.com/sqlcat/2016/08/18/migrating-data-to-azure-sql-data-warehouse-in-practice/
→ https://blogs.msdn.microsoft.com/sqlcat/2017/05/17/azure-sql-data-warehouse-loading-patterns-and-strategies/
https://techcommunity.microsoft.com/t5/DataCAT/Azure-SQL-Data-Warehouse-loading-patterns-and-strategies/ba-p/305456

WORKFLOW – IDENTIFY DATA SOURCES

Imagine a future state solution and develop use cases and query patterns



Using CTAS to load initial data

Then you can use a CTAS (CREATE TABLE AS SELECT) operation within SQL Data Warehouse to load the data from Azure Blob Storage to SQL Data Warehouse:

```
CREATE TABLE orders_load
WITH (CLUSTERED COLUMNSTORE INDEX, DISTRIBUTION = HASH(o_orderkey),
     PARTITION (o_orderdate RANGE RIGHT FOR VALUES (‘1992-01-01’,’1993-01-01’,’1994-01-01’,’1995-01-01’)))
 as select * from orders_ext;
```

CTAS creates a new table. We recommend using CTAS for the initial data load. This is an all-or-nothing operation with minimal logging.

Using INSERT INTO to load incremental data

For an incremental load, use INSERT INTO operation. This is a full logging operation when inserting into a populated partition which will impact on the load performance. Furthermore, the roll-back operation on a large transaction can be expensive. Consider breaking your transaction into smaller batches.

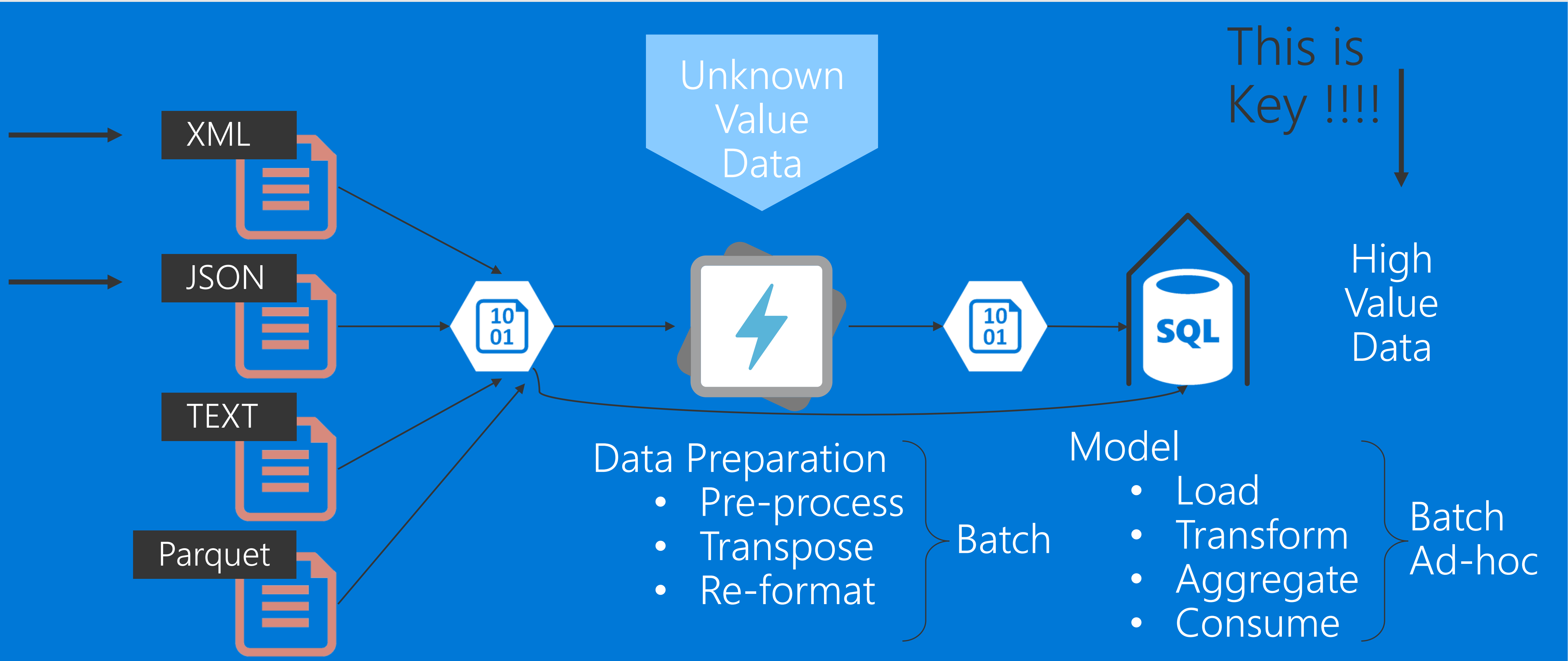
```
INSERT INTO TABLE orders_load
select * from orders_current_ext;
```

Note The source is using different external table, orders_current_ext. This is the external table defining the path for the incremental data on ASB.

Another popular pattern is to load into a partitioned aligned stage table via CTAS, then partition switch into the final table.

https://techcommunity.microsoft.com/t5/DataCAT/Azure-SQL-Data-Warehouse-loading-patterns-and-strategies/ba-p/305456

WORKFLOW – IDENTIFY DATA SOURCES



Unknown Value Data

This is Key !!!!

High Value Data

Data Preparation
- Pre-process
- Transpose
- Re-format

Batch

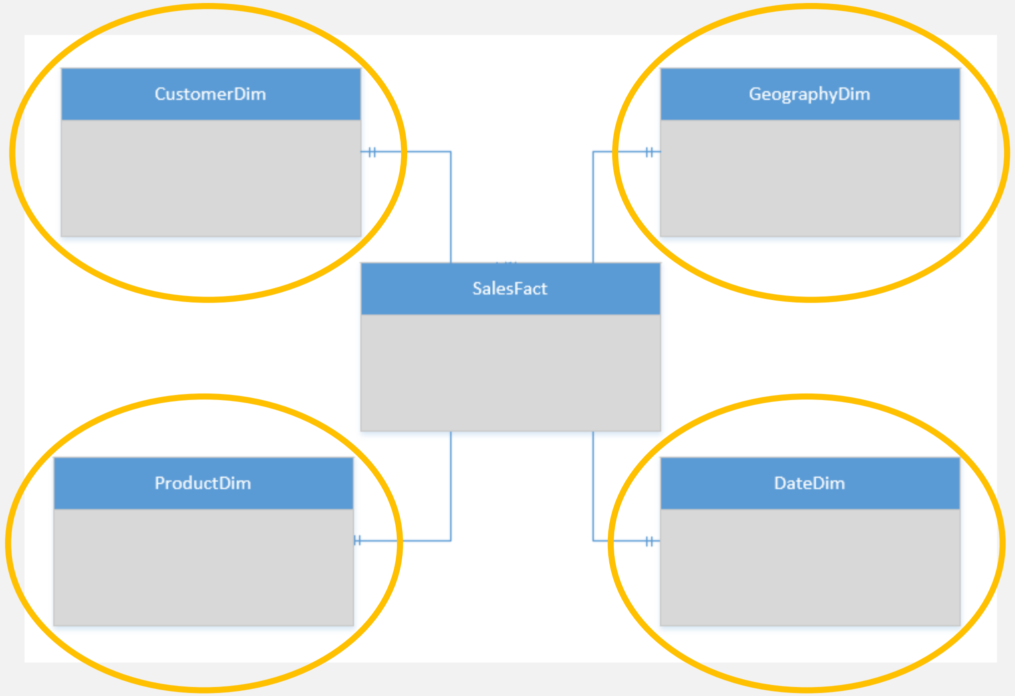
Model
- Load
- Transform
- Aggregate
- Consume

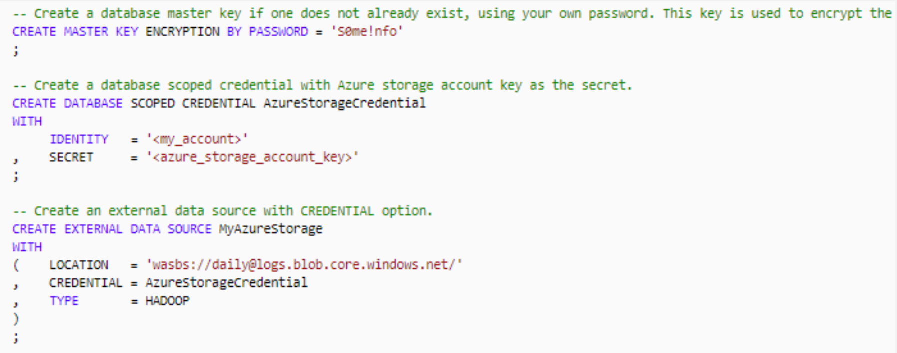
Batch
Ad-hoc

WORKFLOW – DESIGN SQL DW TABLES

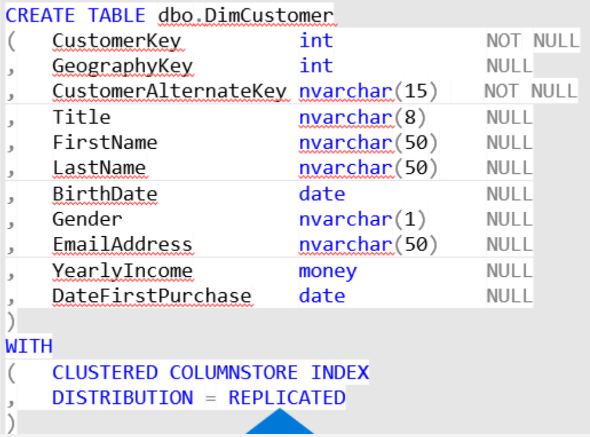
Imagine a future state solution and develop use cases and query patterns

- External Table Definitions
- Staging Tables
- Production Tables
- Indexing Patterns
- Partitioning
- Distribution and Replicated





```
-- Create a database master key if one does not already exist, using your own password. This key is used to encrypt the
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'S0me!nfo'
;

-- Create a database scoped credential with Azure storage account key as the secret.
CREATE DATABASE SCOPED CREDENTIAL AzureStorageCredential
WITH
     IDENTITY   = '<my_account>'
,    SECRET     = '<azure_storage_account_key>'
;

-- Create an external data source with CREDENTIAL option.
CREATE EXTERNAL DATA SOURCE MyAzureStorage
WITH
(    LOCATION   = 'wasbs://daily@logs.blob.core.windows.net/'
,    CREDENTIAL = AzureStorageCredential
,    TYPE       = HADOOP
)
;
```



```
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
)
```

WORKFLOW – OPERATIONS

Imagine a future state solution and develop use cases and query patterns

- Indexing Patterns
- Query Performance
- Data Management Views

Microsoft

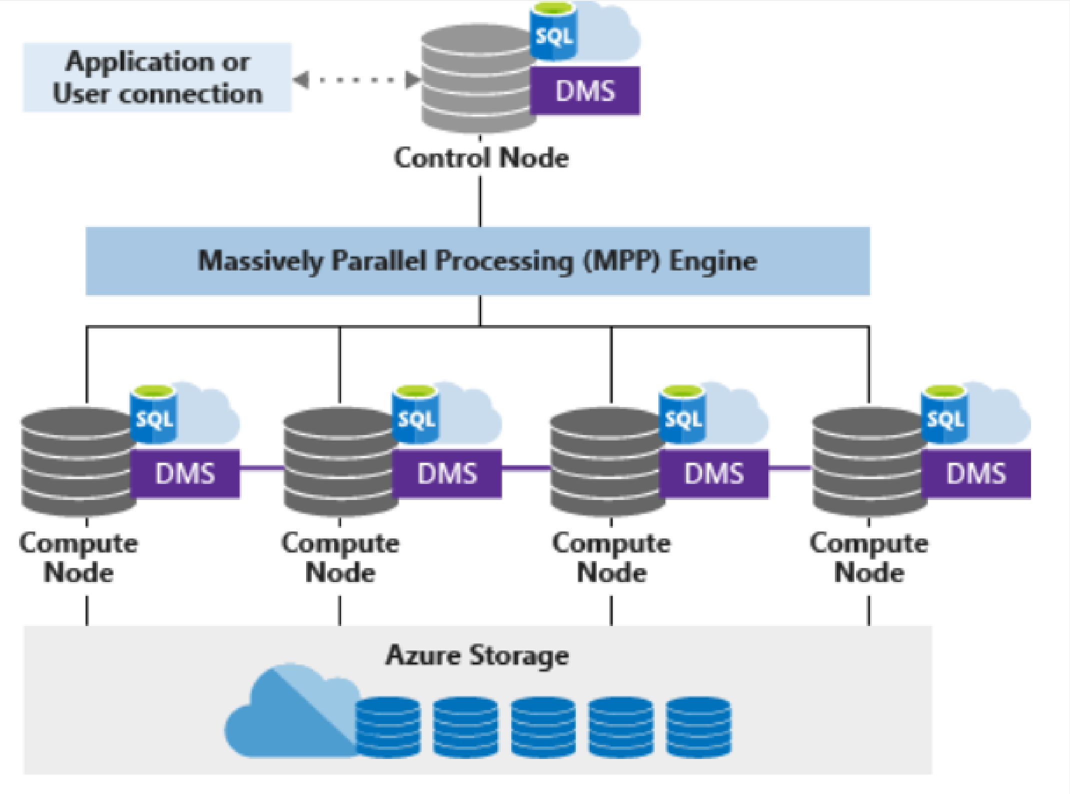
Q&A

SQL Datawarehouse Architecture

ARCHITECTURE - OVERVIEW

Decoupled Storage allows:

- Independently size compute power irrespective of your storage needs.
- Grow or shrink compute power without moving data.
- Pause compute capacity while leaving data intact, so you only pay for storage.
- Resume compute capacity during operational hours.

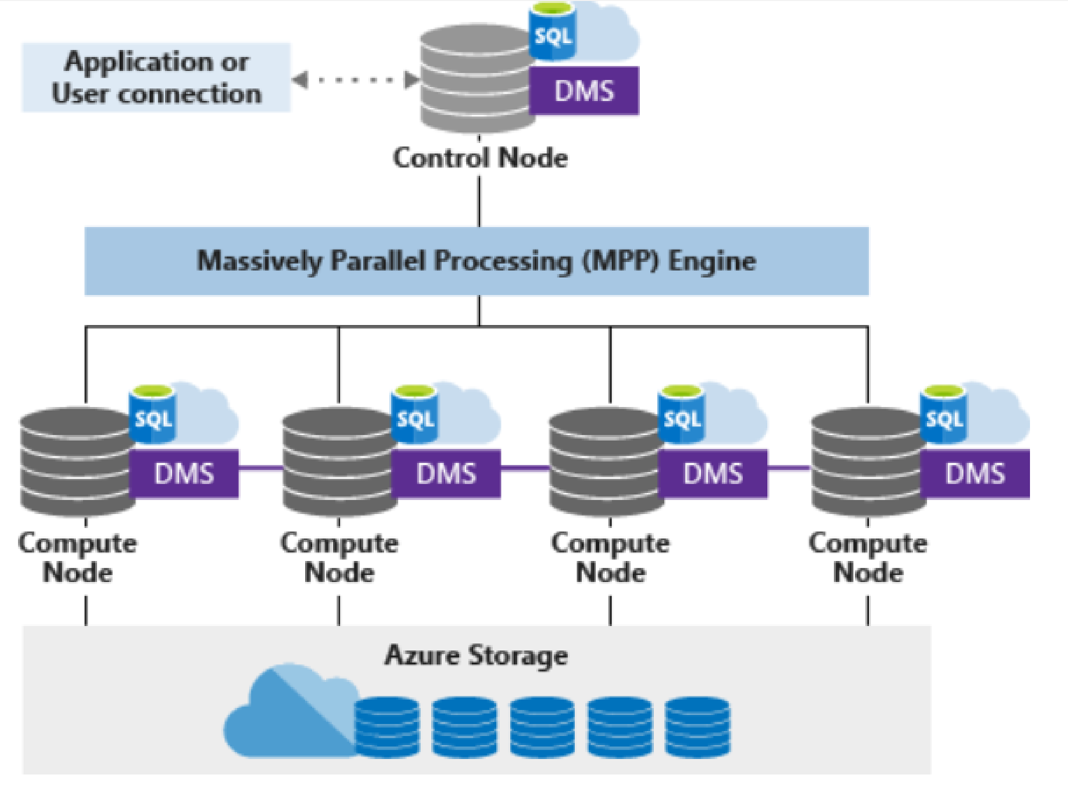


Reference - https://docs.microsoft.com/en-us/azure/sql-data-warehouse/massively-parallel-processing-mpp-architecture

ARCHITECTURE - OVERVIEW

Overview

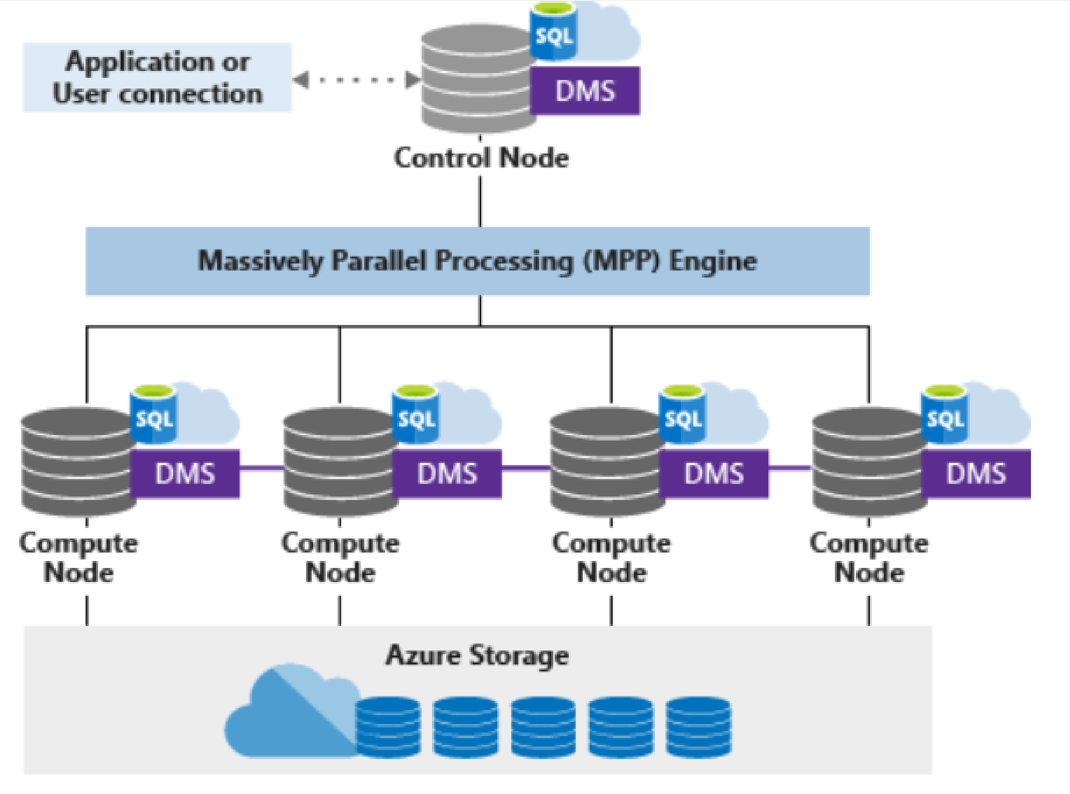
- SQL Data Warehouse uses a node-based architecture.
- Applications connect and issue T-SQL commands to a Control node, single point of entry
- The Control node runs the MPP engine which optimizes queries for parallel processing, and then passes operations to Compute nodes to do their work in parallel.
- The Compute nodes store all user data in Azure Storage and run the parallel queries.
- The Data Movement Service (DMS) moves data across the nodes as necessary to run queries in parallel and return accurate results.



ARCHITECTURE – AZURE STORAGE

Overview

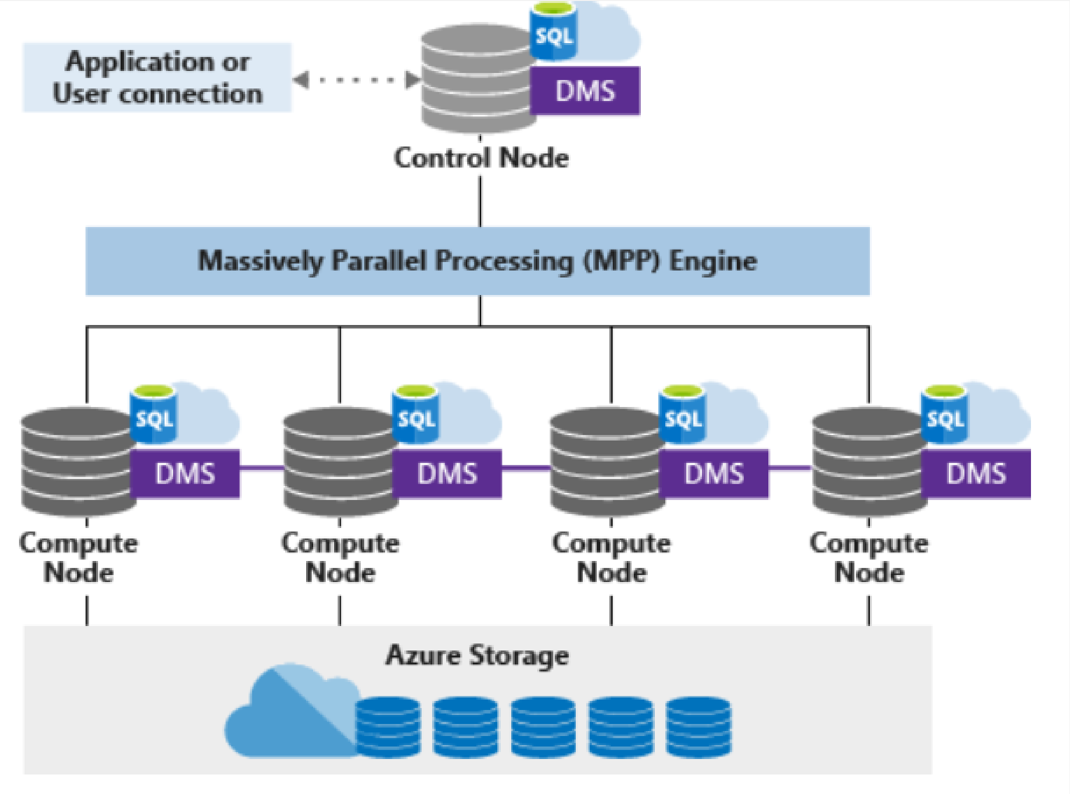
- Data is stored and managed by Azure storage
- The data itself is sharded into distributions to optimize the performance of the system. You can choose which sharding pattern to use to distribute the data when you define the table. SQL Data Warehouse supports these sharding patterns:
    - Hash
    - Round Robin
    - Replicate



ARCHITECTURE – CONTROL NODE

Overview

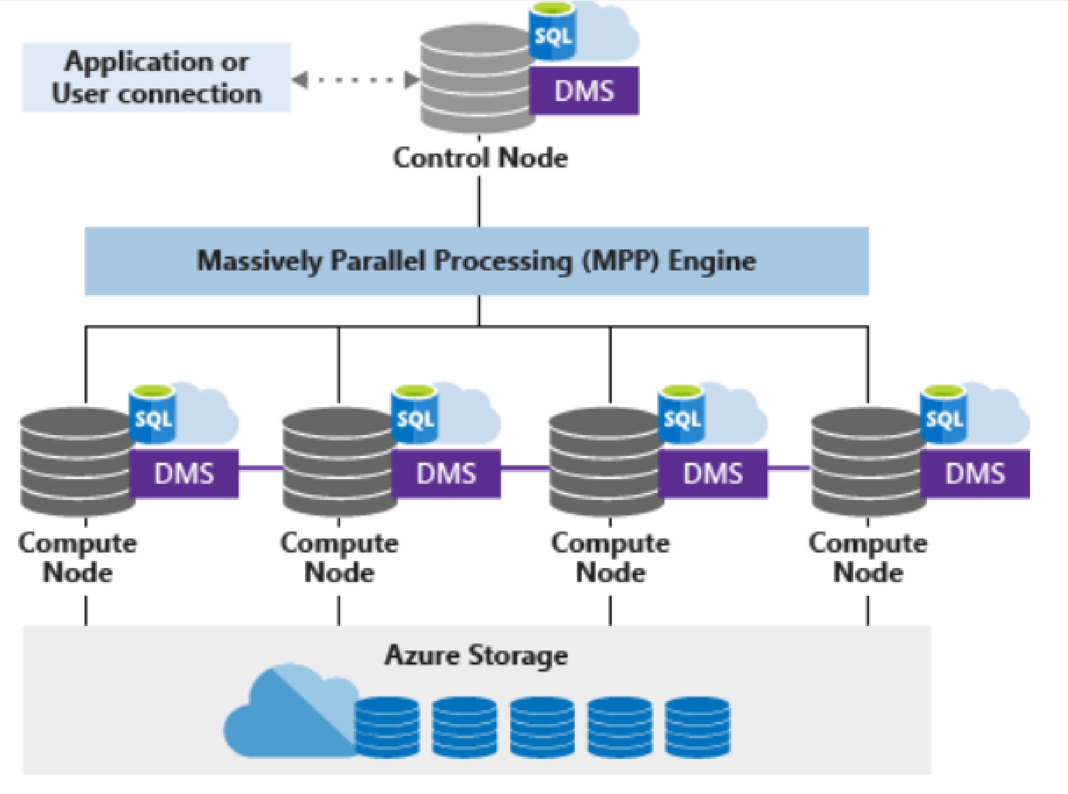
- The brain of the data warehouse
- The front end that interacts with all applications and connections.
- MPP engine runs on the Control node to optimize and coordinate parallel queries.
- The Control node transforms a submit a T-SQL query into queries that run against each distribution in parallel.



ARCHITECTURE – COMPUTE NODES

Overview

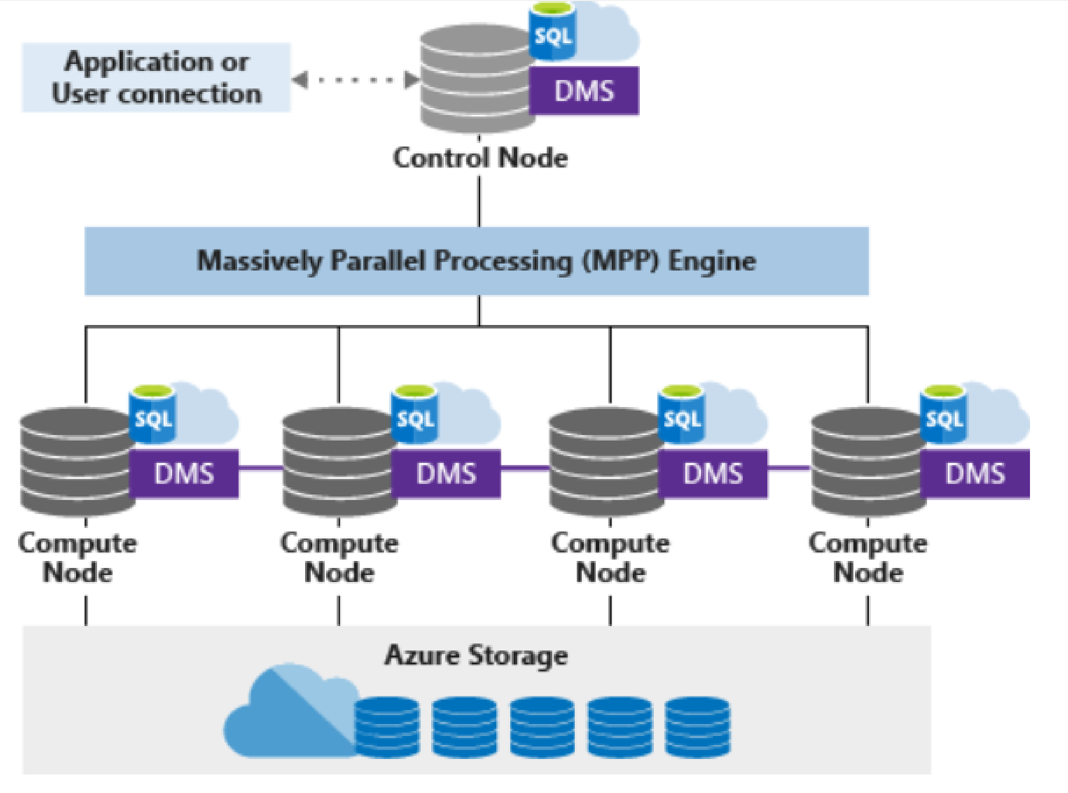
- Provide the computational power.
- Distributions map to Compute nodes for processing.
- Scale Up compute resources, SQL Data Warehouse re-maps the distributions to the available Compute nodes which ranges from 1 to 60
- Each Compute node has a node ID that is visible in system views. https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sql-data-warehouse-and-parallel-data-warehouse-catalog-views?view=aps-pdw-2016-au7



ARCHITECTURE – DATA MOVEMENT SERVICE

Overview

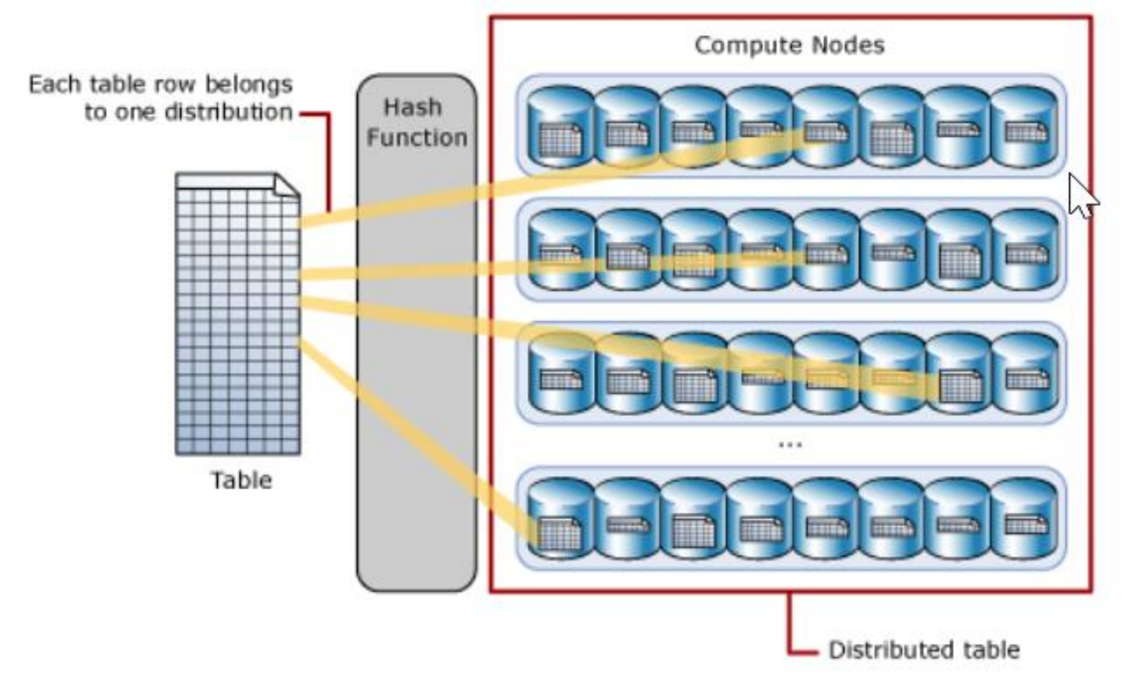
- The data transport technology that coordinates data movement between the Compute nodes.
- Some queries require data movement to ensure the parallel queries return accurate results.



ARCHITECTURE – TABLE DISTRIBUTIONS

Hash Table Distributions

- A hash distributed table can deliver the highest query performance for joins and aggregations on large tables.
- Uses a hash function to deterministically assign each row to one distribution.
- In the table definition, one of the columns is designated as the distribution column.
- The hash function uses the values in the distribution column to assign each row to a distribution.



ARCHITECTURE – TABLE DISTRIBUTIONS

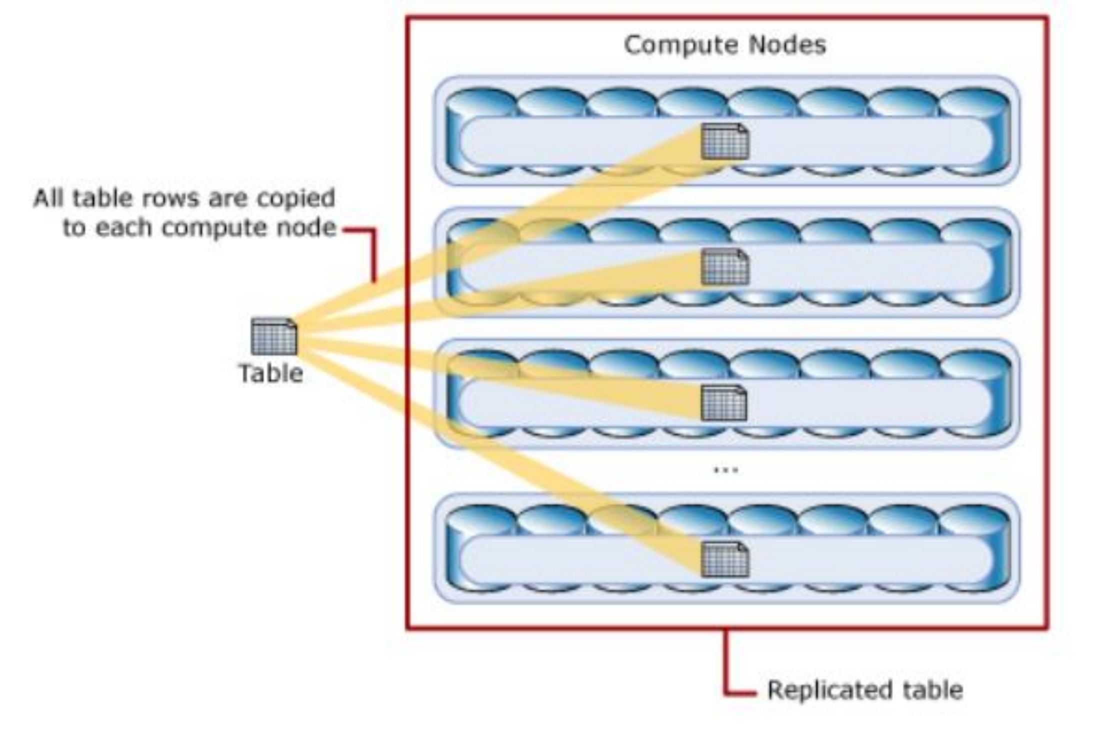
Round-robin distributed Table Distributions

- Round-robin table is the simplest table to create and delivers fast performance when used as a staging table for loads.
- A round-robin distributed table distributes data evenly across the table but without any further optimization.
- Query performance can often be better with hash distributed tables.
- Joins on round-robin tables require reshuffling data and this takes additional time.

ARCHITECTURE – TABLE DISTRIBUTIONS

Round-robin distributed Table Distributions

- A replicated table provides the fastest query performance for small tables.
- A table that is replicated caches a full copy of the table on each compute node. Consequently, replicating a table removes the need to transfer data among compute nodes before a join or aggregation. Replicated tables are best utilized with small tables. Extra storage is required and there is additional overhead that is incurred when writing data which make large tables impractical.



DATABRICKS – STRUCTURED STREAMING

Overview

The Databricks SQL DW connector supports batch and structured streaming support for writing real-time data into Azure SQL Data Warehouse.

It uses Polybase and the Databricks structured streaming API to stream data from Kafka, Kinesis sources directly into SQL DW at a user-configurable rate.

Source: https://docs.azuredatabricks.net/spark/latest/data-sources/azure/sql-data-warehouse.html#streaming-support

```
# Prepare streaming source; this could be Kafka,
Kinesis, or a simple rate stream.
df = spark.readStream \
  .format("rate") \
  .option("rowsPerSecond", "100000") \
  .option("numPartitions", "16") \
  .load()

# Apply some transformations to the data then use
# Structured Streaming API to continuously write the
data to a table in SQL DW.
df.writeStream \
  .format("com.databricks.spark.sqldw") \
  .option("url", <azure-sqldw-jdbc-url>) \
  .option("tempDir",
"wasbs://<containername>@<storageaccount>.blob.core.
windows.net/<directory>") \
  .option("forwardSparkAzureStorageCredentials",
"true") \
  .option("dbTable", <table-name>) \
  .option("checkpointLocation", "/tmp_location") \
  .start()
```