Implement Relational Data Stores

Implement relational data stores

Learning Objectives

- Azure SQL Database offerings
  - PaaS vs IaaS
  - PaaS deployment options
  - Purchasing models and Service Tier
  - Elastic Database
  - Managed Database
  - Security
- Azure SQL Datawarehouse
  - Traditional vs Modern Architecture
  - Azure Synapse Analytics
  - MPP Architecture
  - Storage and Sharding Pattern
  - Data Distribution and Table types
  - Partitioning
- Loading methods
  - PolyBase vs SSIS
  - PolyBase steps and demos
- Azure SQL Server vs Datawarehouse

Introduction

DBaaS

Azure SQL Database Deployment Types

RELATIONAL - DATABASE - AS - A - SERVICE

SQL SERVER INSTANCE

Single Database
Own set of resources

Elastic Pool
Collection of databases sharing resources

Managed Instance
Dedicated engine instance running collection of databases

Why SQL Server in Azure?

Fully Managed

Predictable performance and pricing

DB COLLECTION

Elastic pool for unpredictable workloads

DTU COMPUTE
SHARED RESOURCES

99.99% availability built-in

52m DOWN PER YEAR

DISASTER RECOVERY

Geo-replication and restore services

Supports existing SQL Server tools, libraries, and APIs

Scalability with no downtime

Secure and compliant for your sensitive data

Azure IaaS vs PaaS Database offerings?

SQL Server on Azure VMs

SQL Server inside a
fully-managed VM in Azure

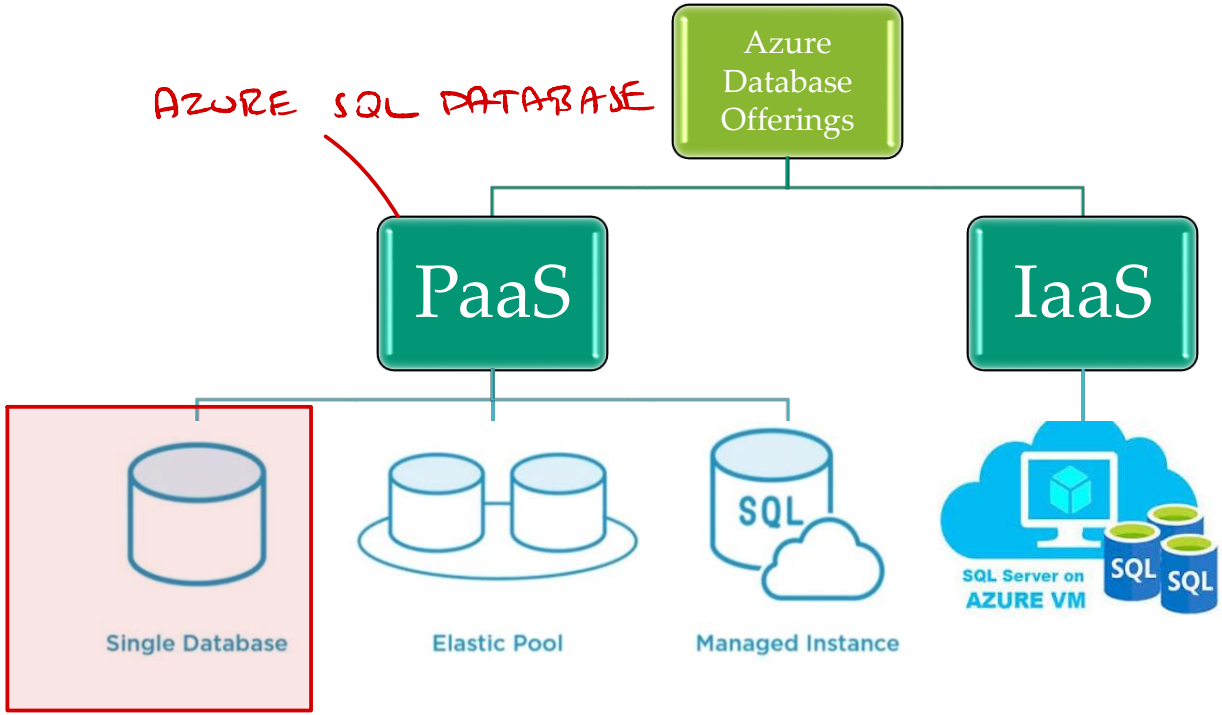
Azure SQL Database

Database-as-a-service (DBaaS)
hosted in Azure

SQL Server and Azure SQL Database Comparison

SQL Server	Azure SQL Database
Responsibility Comparison	
Maintain operating system	Choosing the service tier
Maintain SQL Server	Testing HA & DR
Maintain HA & DR	Maintain performance tuning
Maintain security	Maintain change control
Maintain performance tuning	Maintain security
Maintain change control	
Benefit Comparison	
99.95 % availability	99.99% availability
Full control over SQL Server engine	Change resources with no downtime
Full parity with matching version of on-premises SQL server	Most commonly used SQL Server features available
Easy migration from SQL Server on-premises	Built-in backups, patching, recovery
Private IP address within Azure VNet	Ability to assign necessary resources to individual databases (CPU, Storage)
	Built-in advanced intelligence and security

Azure Database Deployment options



SQL SERVER IS A LOGICAL SERVER (MANAGED)

SQL Server(PaaS) Deployment Options

Single Database

Single database

Each DB with its own guaranteed compute, memory, and storage

ISOLATED DB
DTU OR VCORES

Elastic pool

Fixed resources will be shared by all databases in the pool

FIXED RESOURCES ARE DYNAMICALLY SHARED

Managed Instance

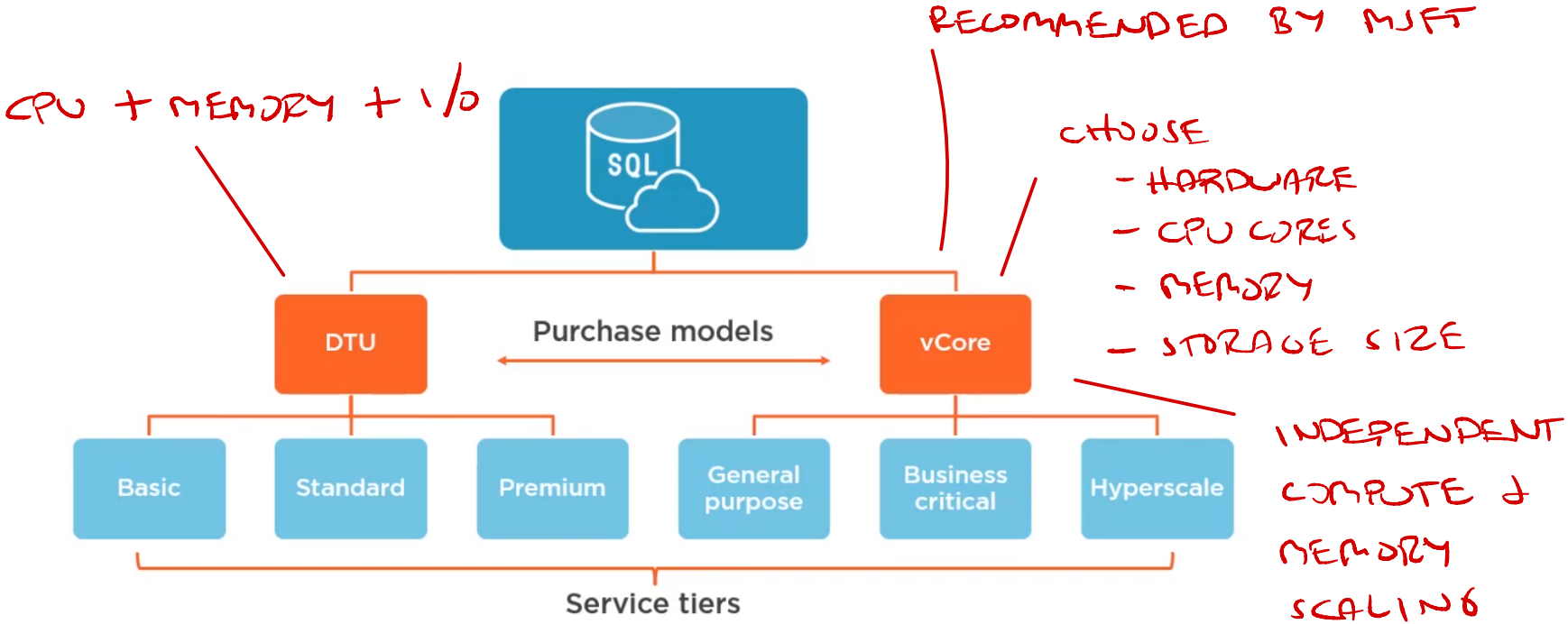
Managed instance

Each managed instance has its guaranteed resources

DEDICATED SQL SERVE INSTANCED MANAGED ON VM

- SET OF DBS THAT CAN BE USED TOGETHER
- EASY MIGRATION

Azure SQL Database Service Tiers



vCore-based vs DTU-based Model

vCore-based

ALL DEPLOYMENT OPTIONS

- For Single database, elastic pool and managed instance
- Best for customers who need flexibility, control, and transparency
- Straightforward way to translate on-premises workload to the cloud
- Microsoft recommends vCore-based model

DTU-based

- Only for single database and elastic pool
- Best for customers who want single, preconfigured resource options
- Might need to calculate the needed DTUs before migration
- If the DTU-based purchasing model needs your performance and business requirements, you should continue using it.

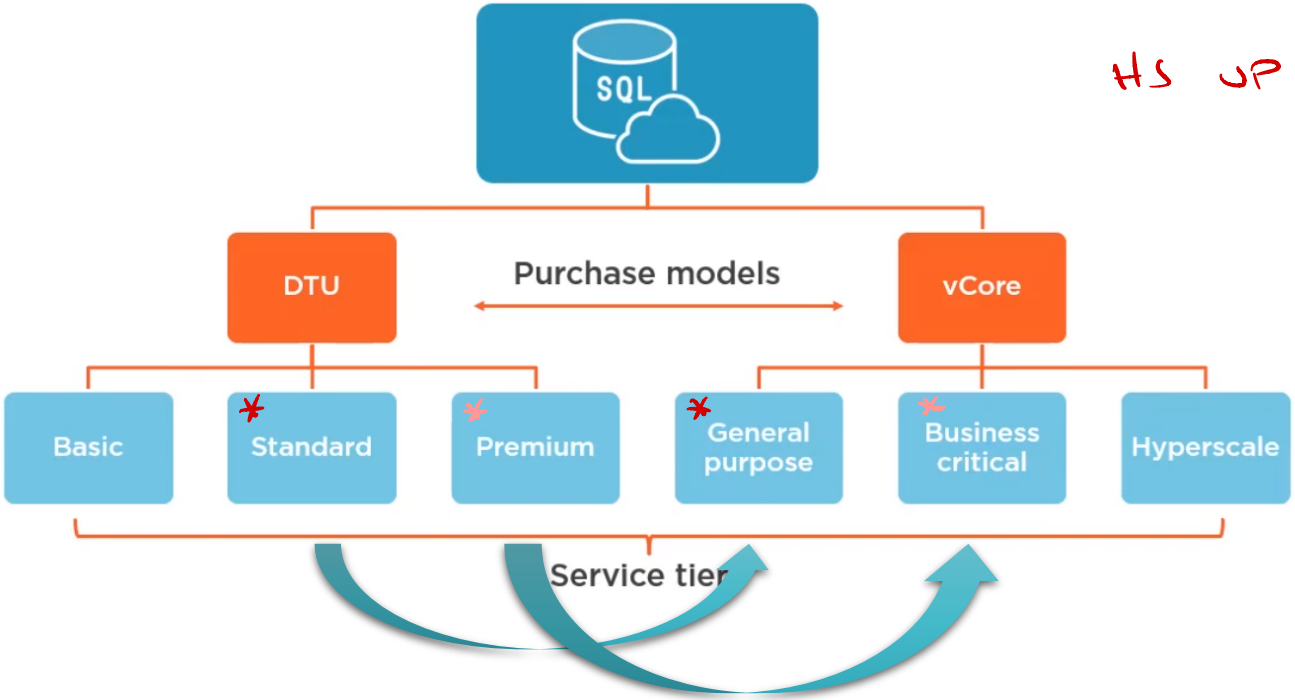
Converting DTU-based Model to vCore-based

- If you single database or elastic pool consumes more than 300 DTUs, converting to the vCore-based model might reduce your costs
- You can use API of your choice or Azure Portal to convert to vCore based model with 'no downtime'.
- Azure SQL Database managed instance only supports vCore-based purchasing model.

Azure SQL Database Service Tiers

GP & BC UP TO 4TB

HS UP TO 100TB



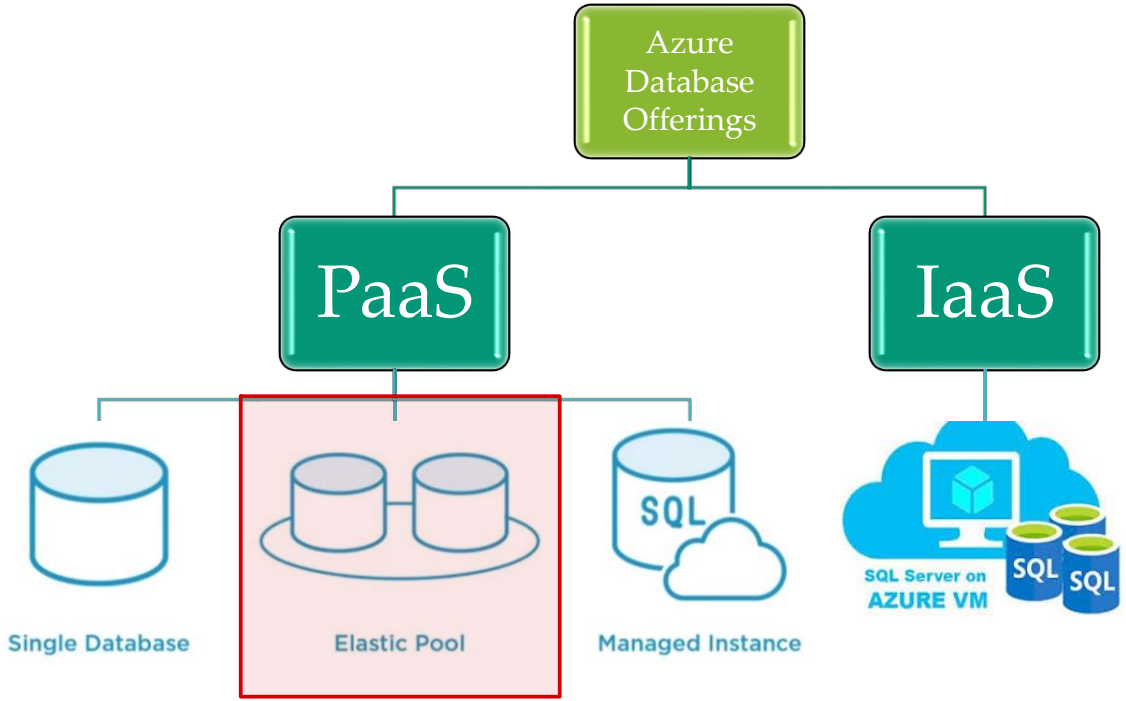
Azure SQL Database Options

SINGLE DB
EASTIC POOL
MANAGED INST

DTU
VCORE

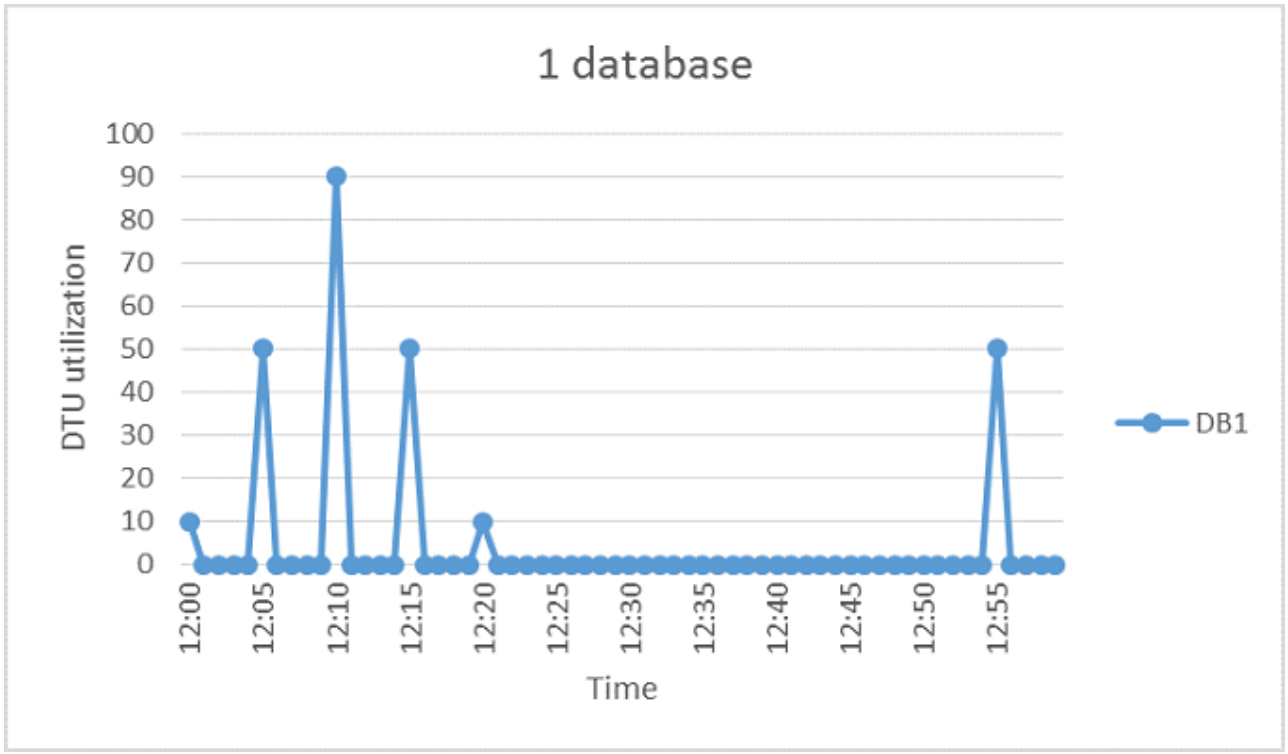
DTU B / S / P
VCORE GP / BC / HS

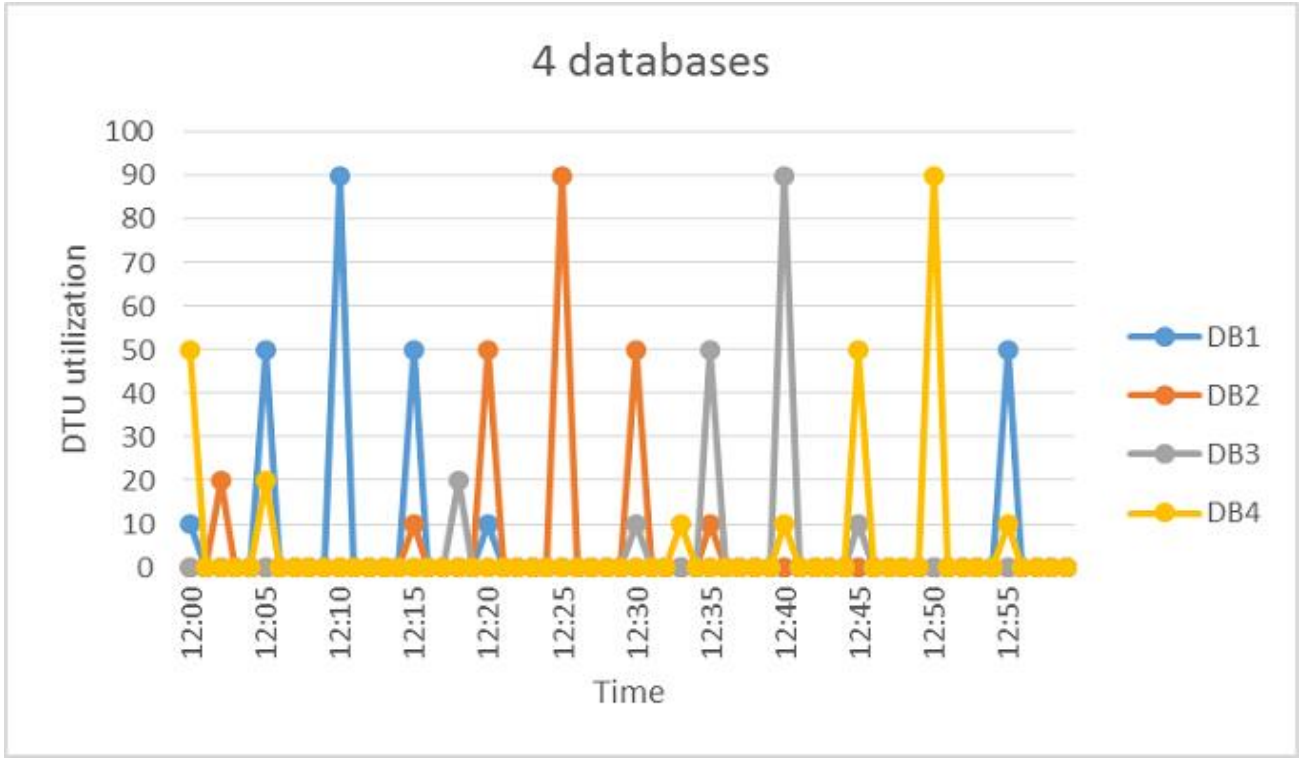
Azure Database Elastic Pool

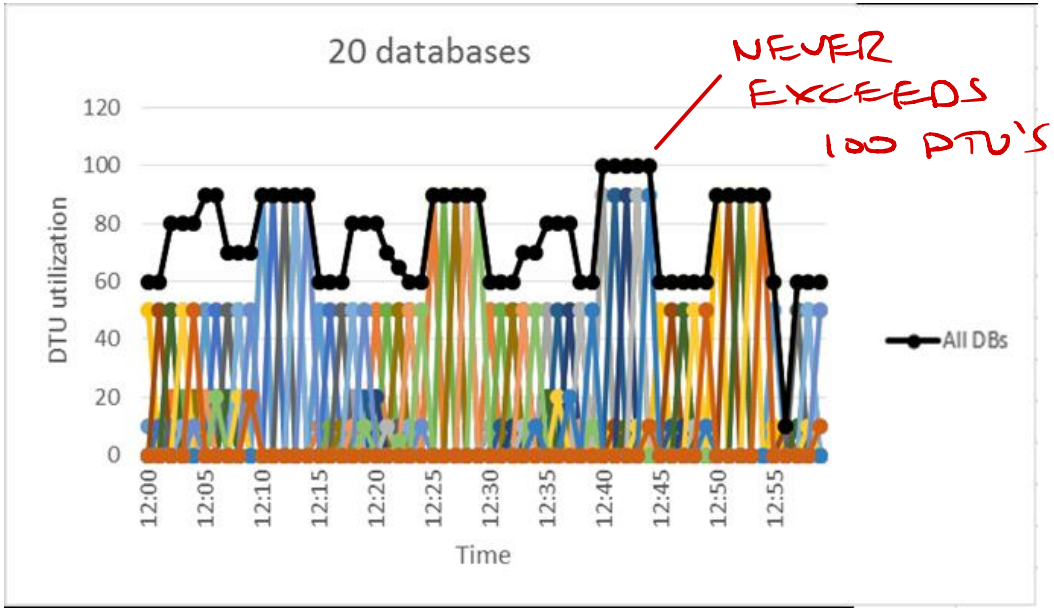


SQL Database Elastic Pools

An Azure SQL Elastic Pool allows you to allocate a shared set of compute resources to a collection of Azure SQL databases,


1 database
DTU utilization
100
90
80
70
60
50
40
30
20
10
0
12:00
12:05
12:10
12:15
12:20
12:25
12:30
12:35
12:40
12:45
12:50
12:55
Time
DB1


4 databases
DTU utilization
100
90
80
70
60
50
40
30
20
10
0
12:00
12:05
12:10
12:15
12:20
12:25
12:30
12:35
12:40
12:45
12:50
12:55
Time
DB1
DB2
DB3
DB4



ALL 20 DB'S CAN SHARE THE 100 DTU'S INSTEAD OF CONSUMING OWN RESOURCE

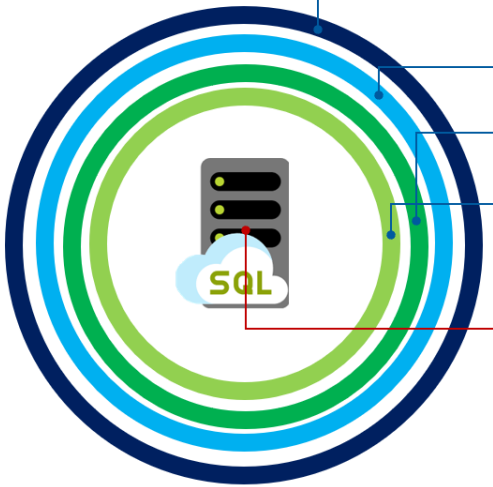
SHARED DTU'S FOR ALL DATABASES

Azure SQL Database Elastic Pool

- So Azure SQL elastic pool is a cost- effective solution for managing and scaling multiple databases that have varying and unpredictable usage demands.
- The databases in an elastic pool are on a single Azure SQL Database server and share a set number of resources at a set price.
- Elastic pool enables developers to optimize the price performance for a group of databases within a prescribed budget.
- Elastic pools prevent over-provisioning or under-provisioning of resources.

Security

4 LAYERS OF DEFENSE



NETWORK SECURITY — VALID IP ACCESS

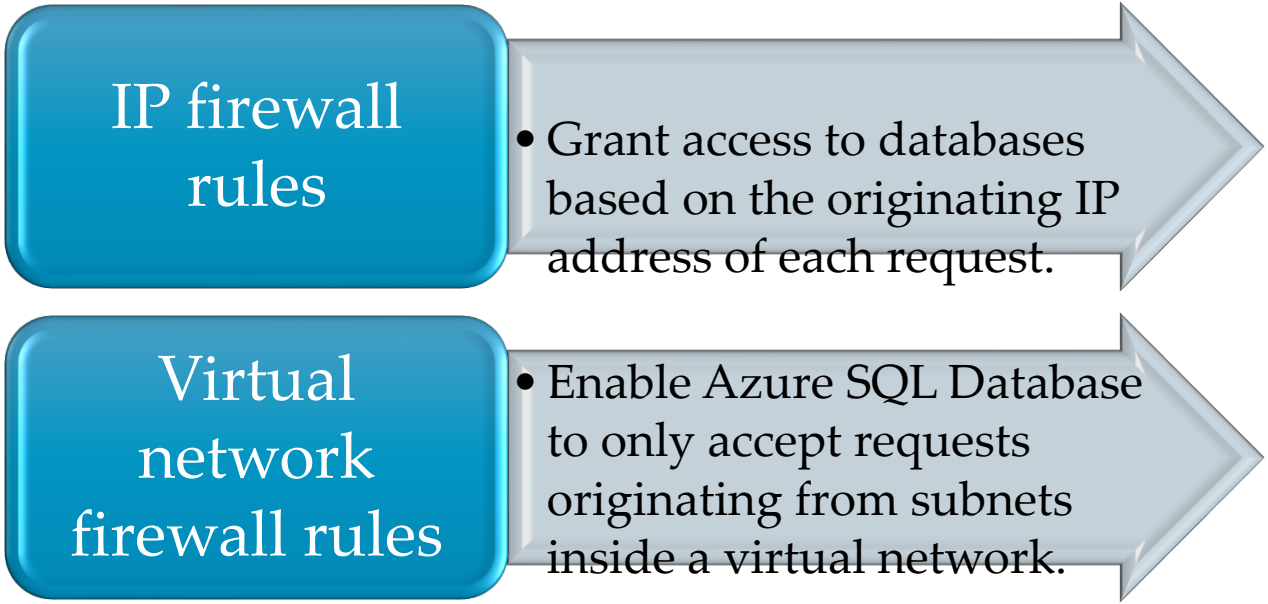
ACCESS MANAGEMENT — AUTH

THREAT PROTECTION — MONITOR FOR SUS PATS

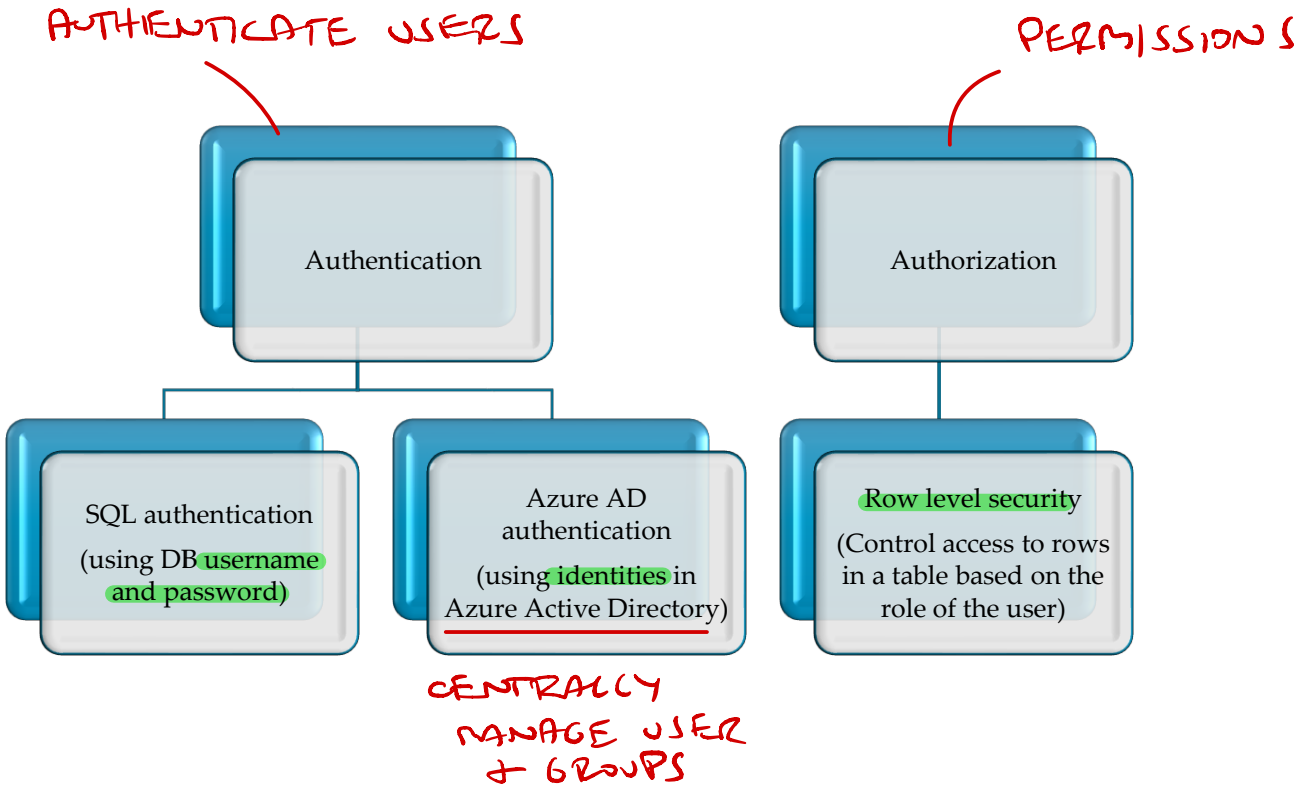
INFORMATION PROTECTION — ENCRYPTION

CUSTOMER DATA

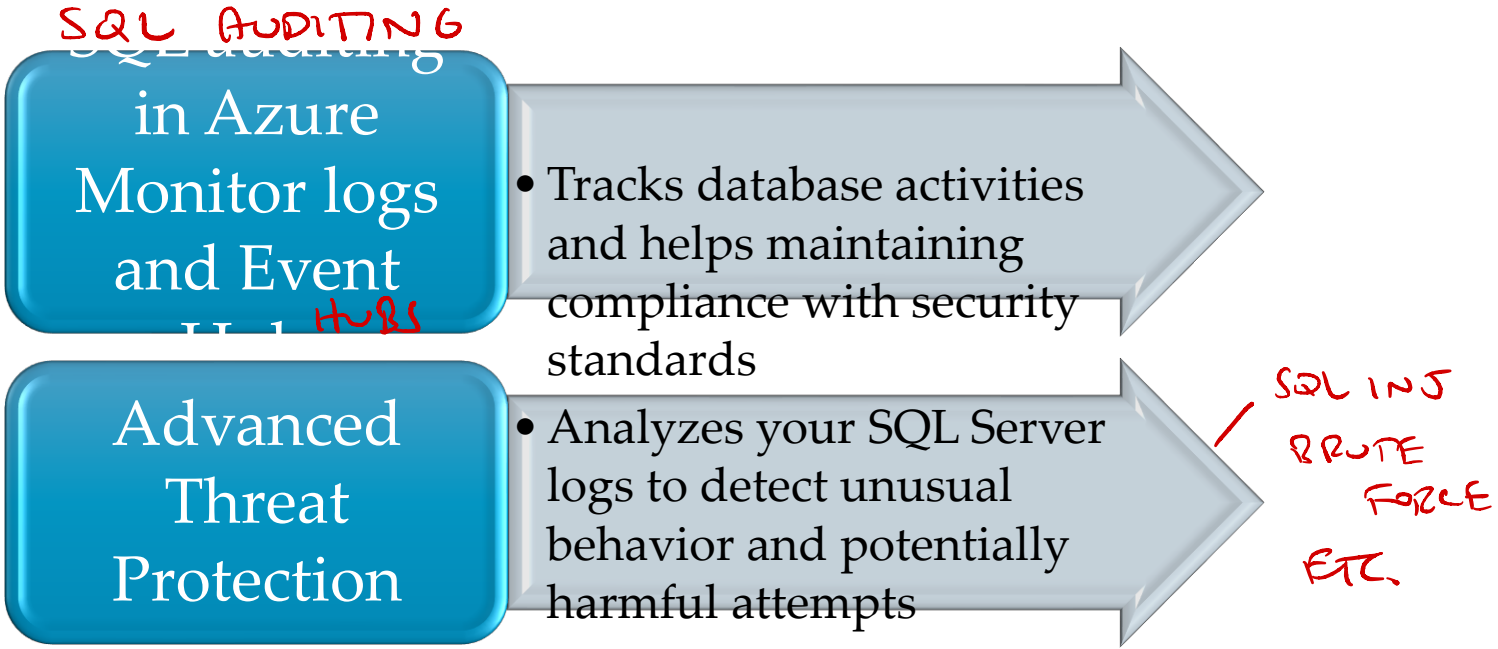
Network security



Access Management

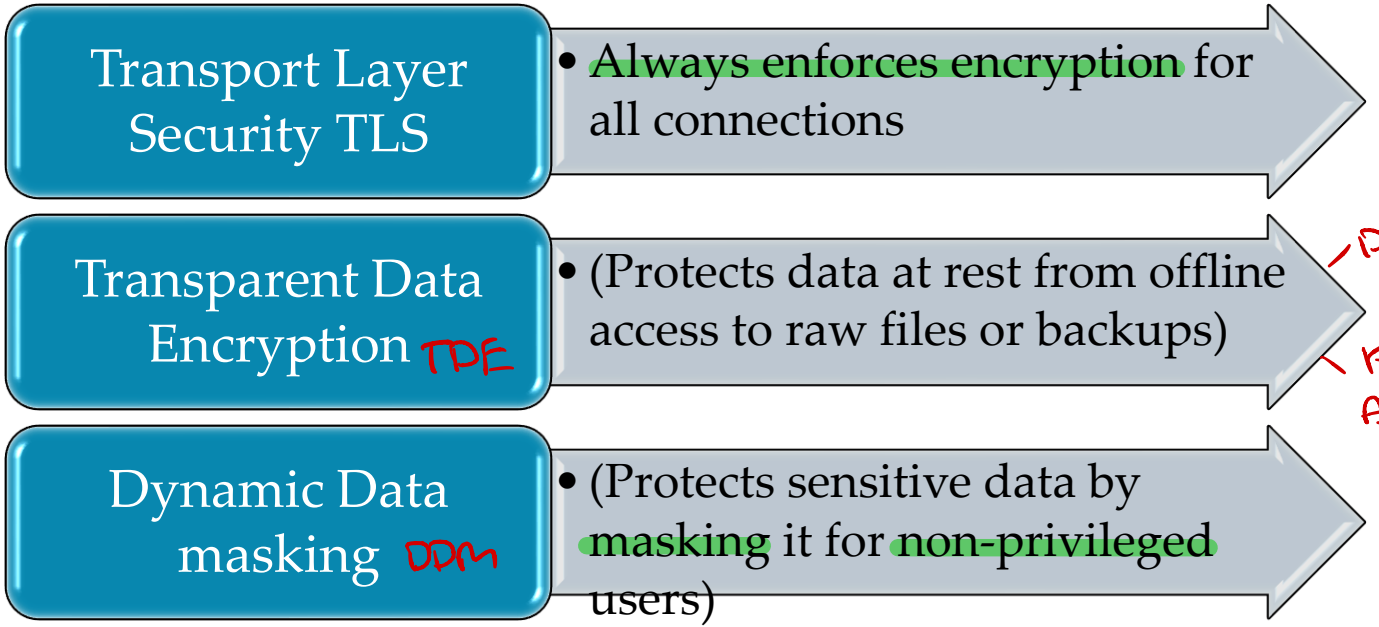


Threat Protection

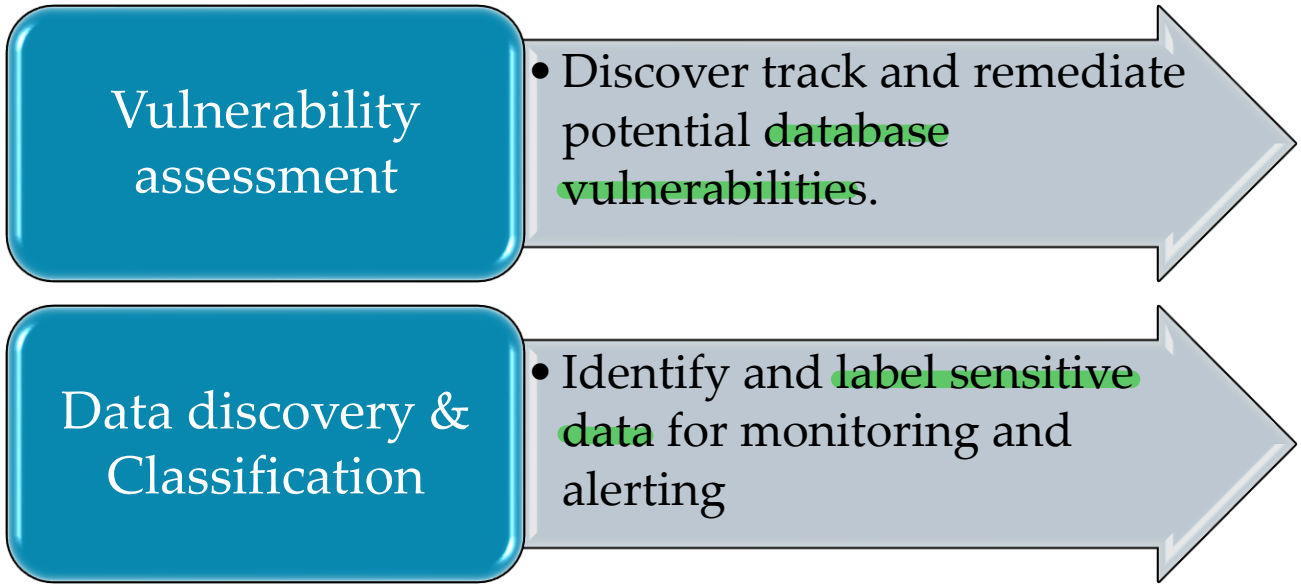


Information Protection

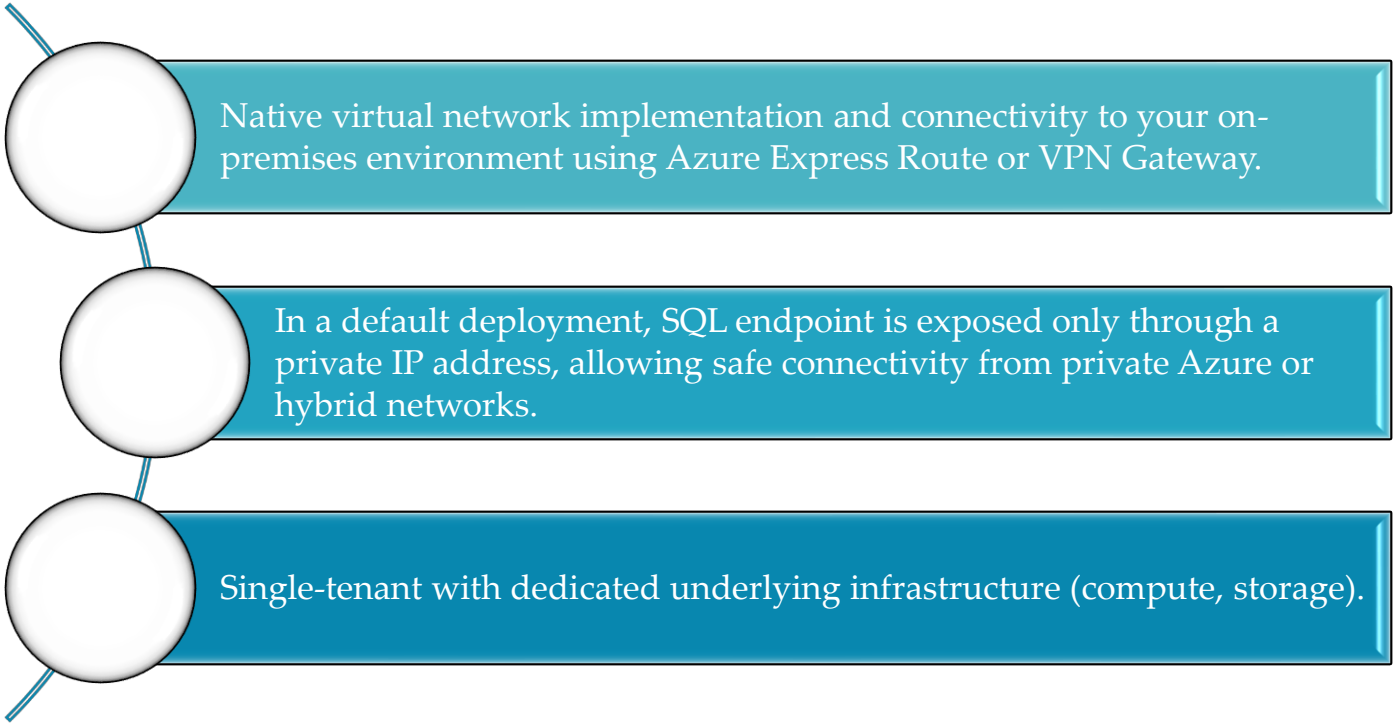
ALL DATA IN TRANSIT IS ENCRYPED (TLS)



Security Management



Managed Instance Advance Security

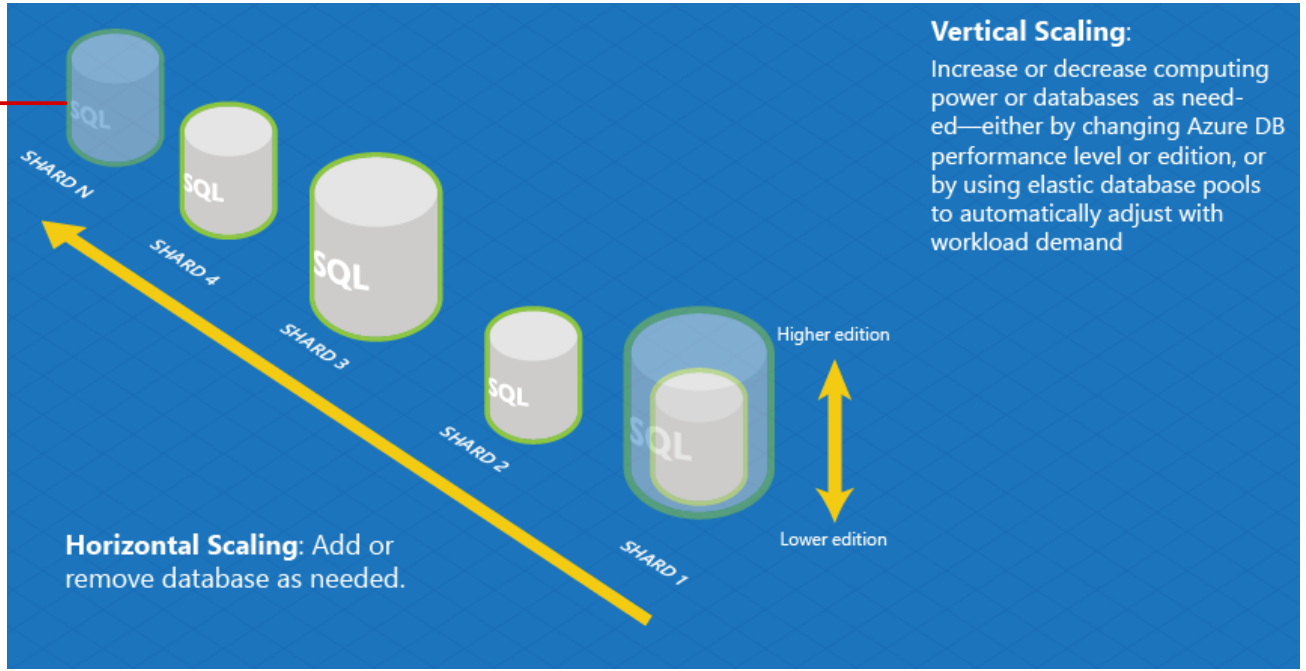


Azure SQL Database Scaling

Azure SQL Database supports two types of scaling:

- **Vertical scaling:** Scale up or down the database by adding more compute power.
- **Horizontal scaling:** Scale out or Add more databases and to shard your data into multiple database nodes.

EACH SHARD DB CAN BE SCALED INDEPENDENTLY

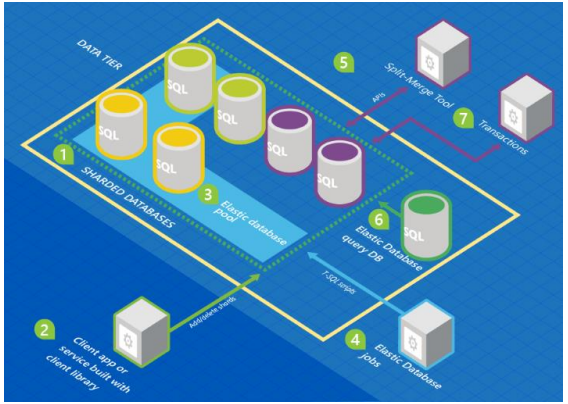


Azure SQL Database Scaling

HORIZONTAL SCALING OPTIONS

Read Scale Out

- Allows you to use the capacity of the read-only replicas for read-only queries.
- Feature is intended for the applications that include logically separated read-only workloads, such as analytics
- Benefit: When Secondary nodes handles heavy reports and analytical queries, primary writable node saved resources that might be used to improve the performance
- Secondary nodes is asynchronous POSSIBLE SMALL DELAY
- Premium (DTU-based model) or in the Business Critical (vCore-based model)
- Also available in Hyperscale with secondary replica creation



Global Scale-out/Sharding

- Split your data into multiple database nodes. / NOT COPIES
- Every database shard is an independent database where you can add or remove resources as needed.
- Application may access only the shard associated to that region without affecting other shards.
- Why?
  - Data or transaction throughput exceed the capabilities of individual database
  - Tenants may require physical isolation
  - Different sections of a database may need to reside in different geographies for compliance, performance, or geopolitical reasons.

EXAMPLE: US SHARD, EU SHARD (ORG DOMAIN SHARDS)

Vertical vs Horizontal Scaling

Azure SQL Database supports two types of scaling:

- **Vertical scaling:**
  - Scale up or down the database by adding more compute power. OR REDUCING
  - CPU Power, Memory, IO throughput, and storage
  - DTU and vCore models to scale
  - Dynamic Scalability (Note: this is not auto-scale) MANUALLY ↑ ↓ RESOURCES
  - Any change that you made will be almost instant .

- **Horizontal scaling:** Scale out or Add more databases and to shard your data into multiple database nodes.

EACH SHARD CAN BE VERTICALLY SCALED

Azure SQL Database Scaling

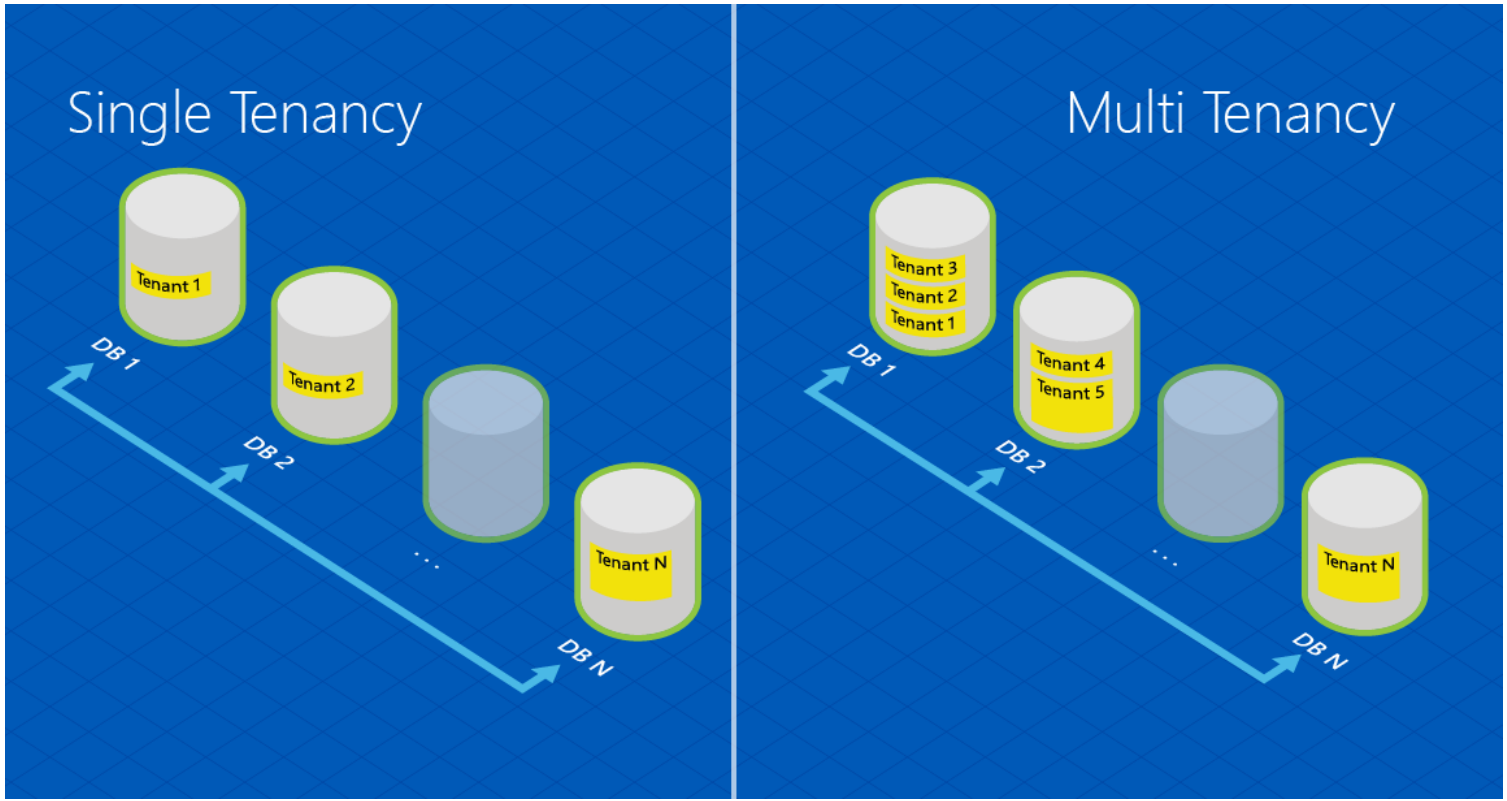
VERTICAL SCALING OPTION

Change Service Tier

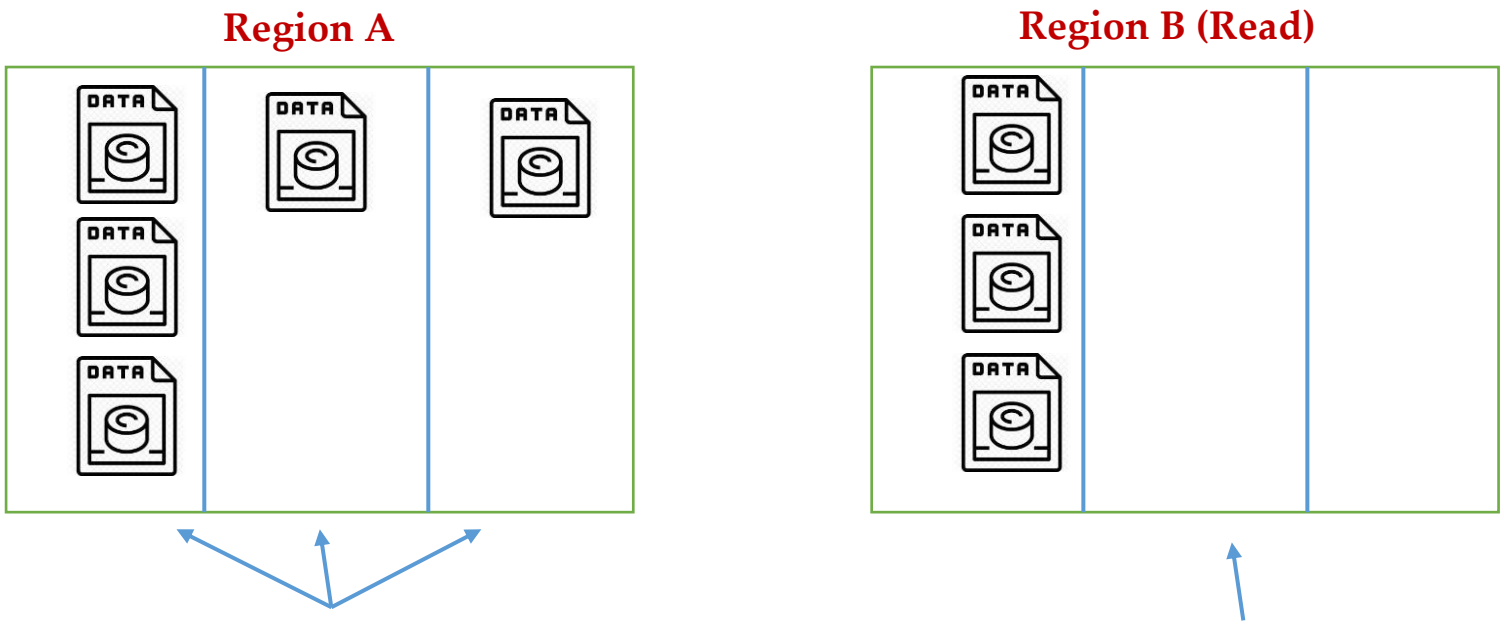
- From Standard/General Purpose to Premium/Business Critical.
- In Standard/General Purpose - Data stored on Azure premium disks
- In Premium/Business Critical - Data stored on local SSD

Single Tenancy vs Multi Tenancy

TENANT = CUSTOMER



Azure SQL DB – HA and DR Options



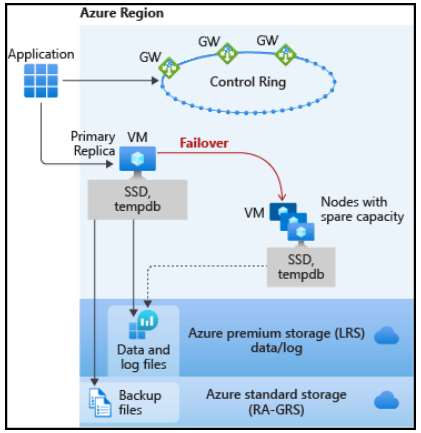
Storage Clusters

Each cluster is physically separate in what's called an availability zone, with its own separate utilities and networking.

Region B

Hundreds of miles away from the primary region to prevent data loss in the event of a natural disaster.

Azure SQL DB – High Availability Architecture

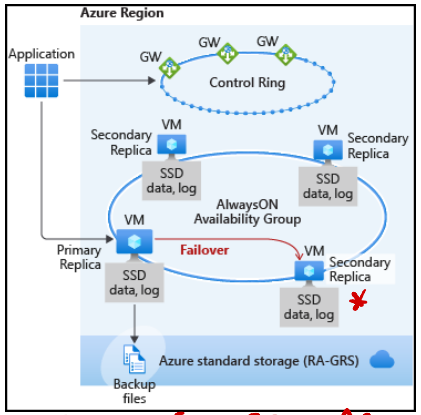


FAILOVERS CAN BE PLANNED OR UNPLANNED

Standard availability model

- Separation of compute and storage
- Performance degradation during maintenance activities
- Budget-oriented business applications
- Basic, Standard, and General Purpose service tier availability

SPARE TAKE OVER FROM PRIMARY NODE DURING FAILOVER.



Premium availability model

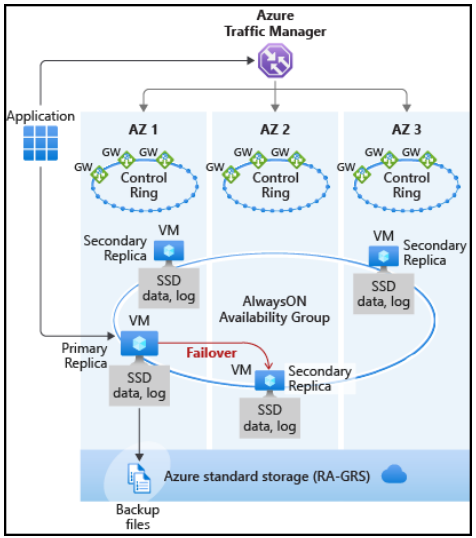
- Integrates compute resources and storage (locally attached SSD) on a single node
- Replicating both compute and storage to additional nodes creating a three to four-node cluster.
- Data is synchronized to at least one secondary replica before committing each transaction.
- Targets mission critical applications with high IO performance and high transaction rate
- Guarantees minimal performance impact during maintenance activities.
- Premium and Business Critical service tier availability
- Read Scale-Out feature * IN CONFIGURE SCREEN

* CAN BE CONFIGURED AS A READ-ONLY NODE FOR FAILURES

Azure SQL DB –Zone Redundancy

AZ = AVAILABILITY ZONES

SETTING IN CONFIGURE SCREEN

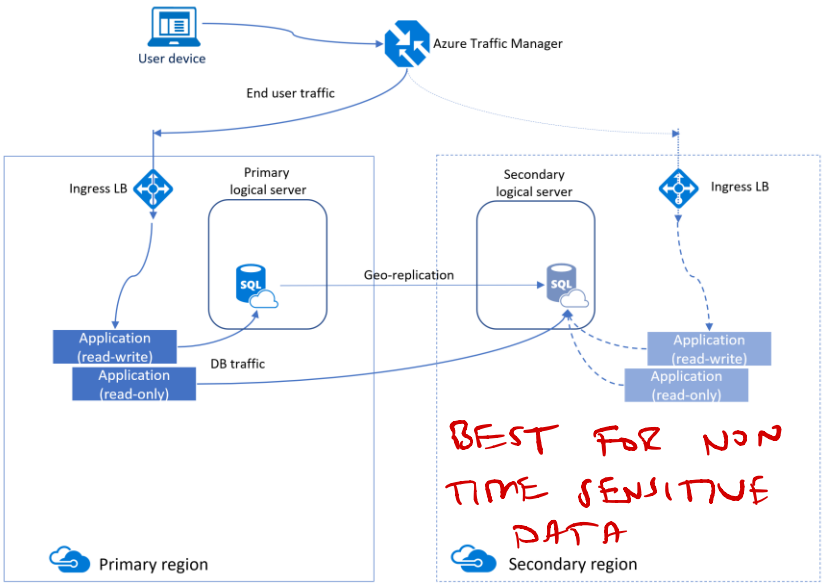


Zone redundant configuration (SIMILAR TO PREMIUM AVA MODEL)

- Place different replicas of the Business Critical database to different availability zones in the same region
- Does not create additional database redundancy
- Enable it at no extra cost
- Supported in the Premium and Business Critical service tiers
- Not available in SQL Managed Instance.
- Increased network latency may increase the commit time and thus impact the performance of some OLTP workloads.

B/C DATA IS SYCHRONIZED TO SECONDARY REPLICA BEFORE TRANSACTION IS COMMITTED

Azure SQL DB – Geo Replication



BEST FOR NON TIME SENSITIVE DATA

THERE WILL BE A LAG IN REPLICATION

- **Create a readable secondary database in the same region or cross-region**
- **Use cases:**
  - Can failover to the secondary database in case of an outage
  - Migrate a database from one server to another server in the same or cross region with minimal downtime.
- We can create up to four secondaries for each primary database.
- **Data Loss:**
  - Uses the Always-on feature to replicate committed transactions to the secondary database asynchronously.
  - May lag the primary database at any point in time.
- **Manual - Forced Failover**
  - This will make your secondary database immediately online and start accepting connections. Forced failover may result in data loss.

- **Issues**:
  - Supports only manual failover
  - End-point connection must be changed in the application after the failover
  - Must have the same firewall rules and the logins to run applications successfully without any discrepancies

CAN BE MITIGATED W/ FAILOVER GROUPS

GROUPS OF DATABASES CAN BE ADDED TO FAILOVER GRPS.

Compare geo-replication with failover groups

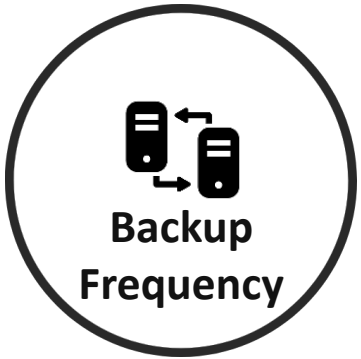
Auto-failover groups simplify the deployment and usage of geo-replication and add the additional capabilities as described in the following table:

	Geo-replication	Failover groups
Automatic failover	No	Yes
Fail over multiple databases simultaneously	No	Yes
User must update connection string after failover	Yes	No
SQL Managed Instance support	No	Yes
Can be in same region as primary	Yes	No
Multiple replicas	Yes	No
Supports read-scale	Yes	Yes

LISTENER ENDPOINTS IN FAILOVER GROUPS DO NOT CHANGE

AZURE SQL DB BACKUP & RESTORE

TYPES OF BACKUPS



1

Full Backup

- Backs up the whole database
- Taken every week

2

Differential backup

- Captures only the data that has changed since the last full backup
- Taken every 12 hours

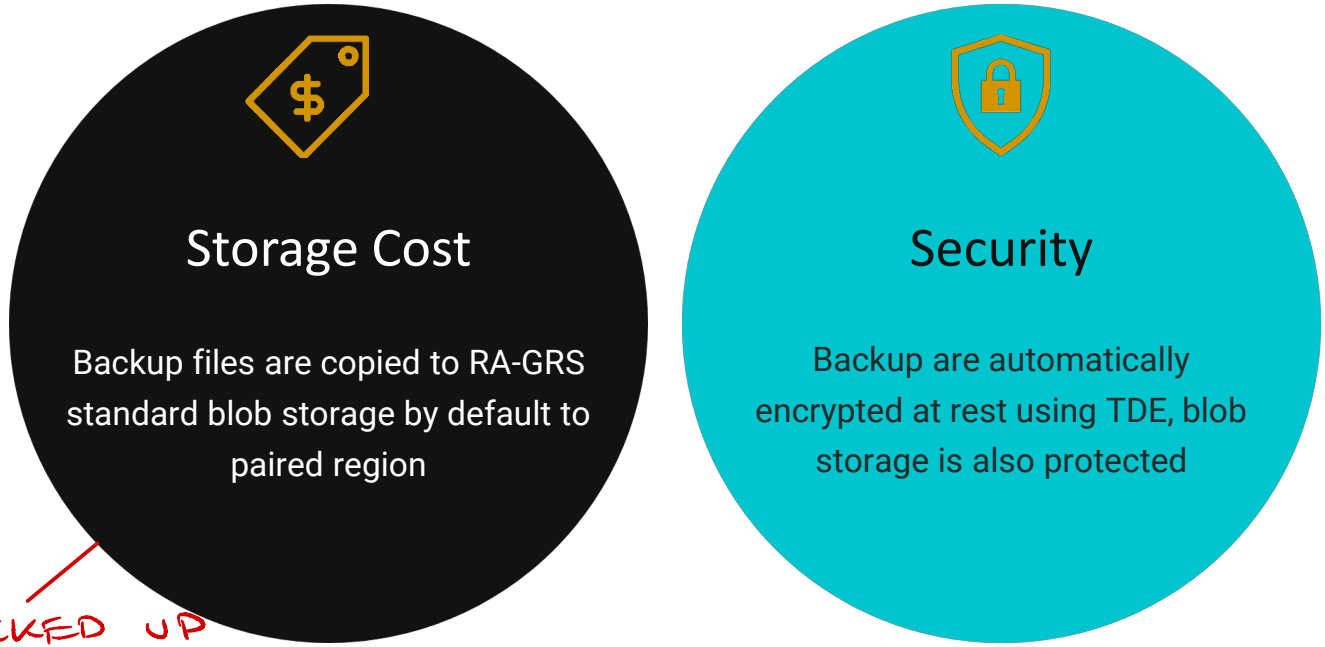
3

Transaction log backup

- Records of all the committed and uncommitted transactions
- Taken every 5-10 mins

CONFIGURE BACKUPS IN SERVER SETTINGS IN AZURE

Storage cost and Security of Backup files



Backup Retention Period

Backup storage redundancy

- Point in time restore (PITR) - 7-35 days
- Long-term retention - Up to 10 years

LTR

Long term retention (LTR)

- One or more long term retention periods to your database to meet regulatory, compliance or other business purposes
- Full backups can be taken up to 10 years
- Stored in RA-GRS blob storage
- Any change of the LTR policy applies to the future backups

LTR and Managed Instance

- LTR is not yet available for databases in Managed Instances
- You can use SQL Agent jobs to schedule copy only database backups as an alternative to LTR beyond 35 days
- These backups can be kept in the Azure blob storage

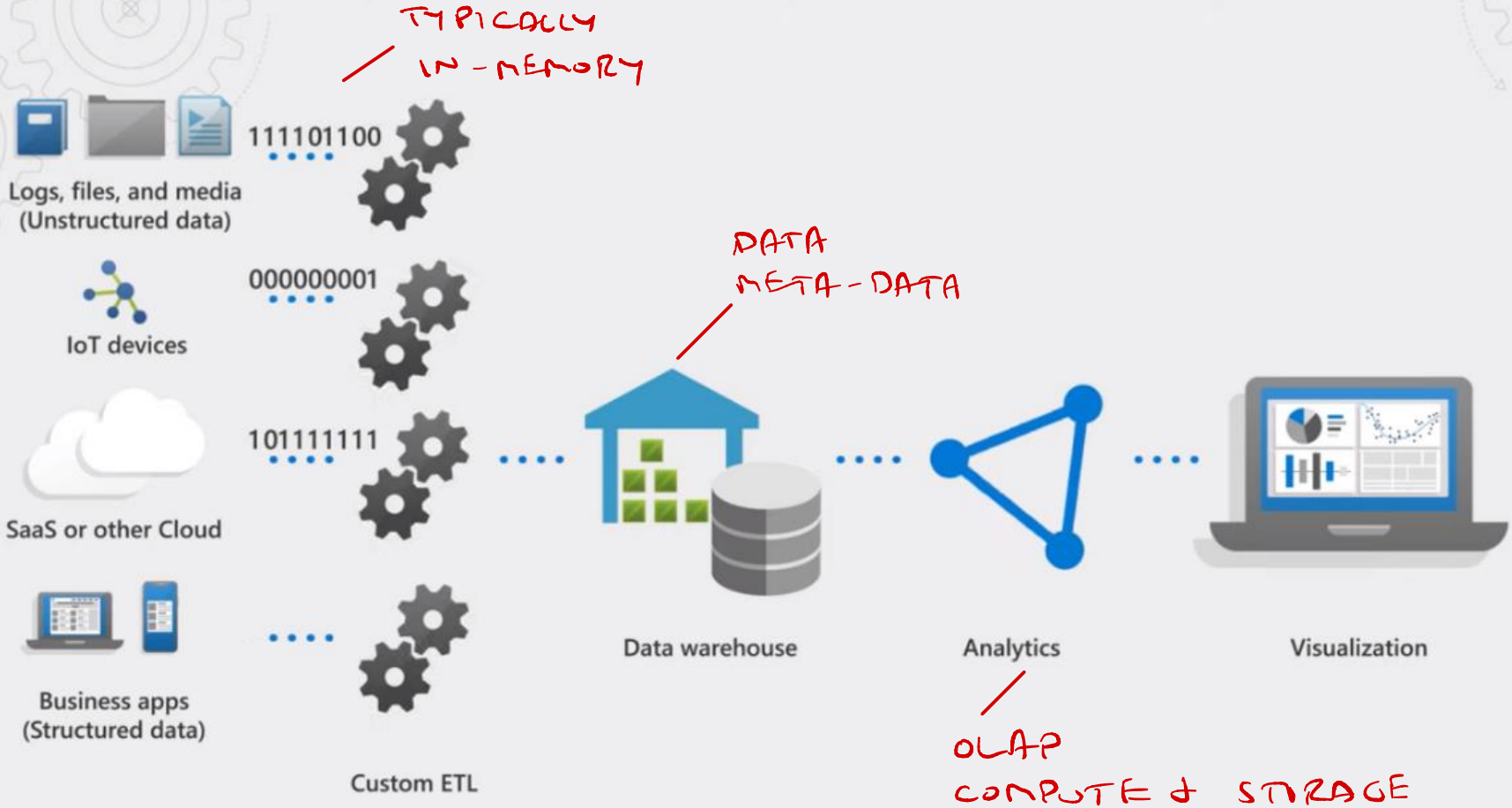
Backup Restore

Backup usage

- Point-in-time restore of existing database
- Point-in-time restore of deleted database
- Geo-restore
- Restore from long-term backup
- *If you delete an Azure Logical SQL Server, all elastic pools and databases that belong to that logical server are also deleted and cannot be restored*

Restore Time is impacted By

- Size of database & Compute size of the database
- Number of transaction logs and Amount of activity
- Network bandwidth if the restore is to a different region
- Concurrent restore requests being processed region



Source: Microsoft

COMPUTE DWU = CPU + MEMORY + IO

• DATA WAREHOUSING UNIT

Modern Data Warehousing

• COMPUTE CAN BE PAUSED

CLEAN / PREP DATA QUALITY

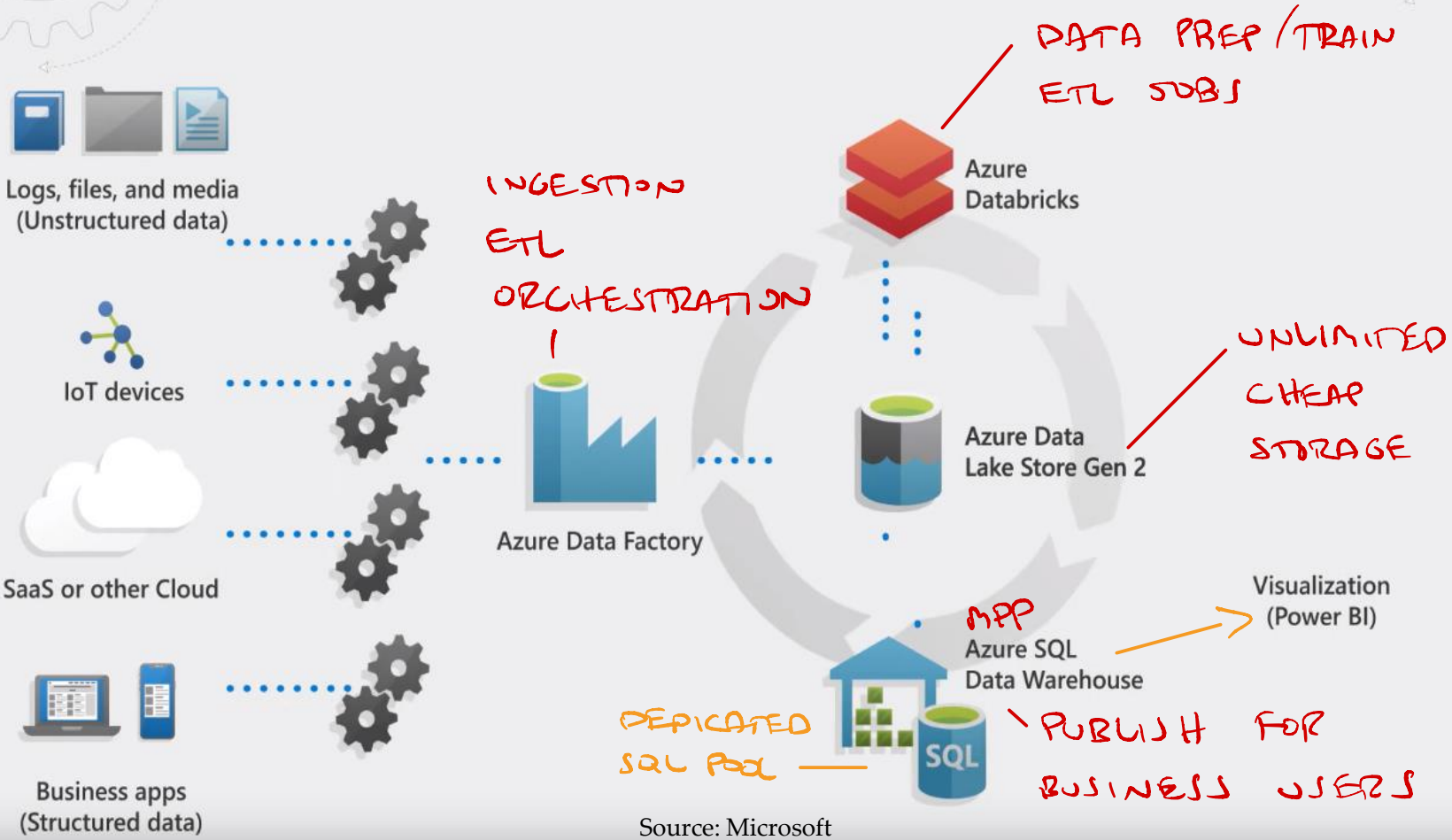
DIMENSIONAL MODELING

COMPUTE & STORAGE DECOUPLED

Source: Microsoft

Modern Data Warehouse

IN TERMS OF SERVICES

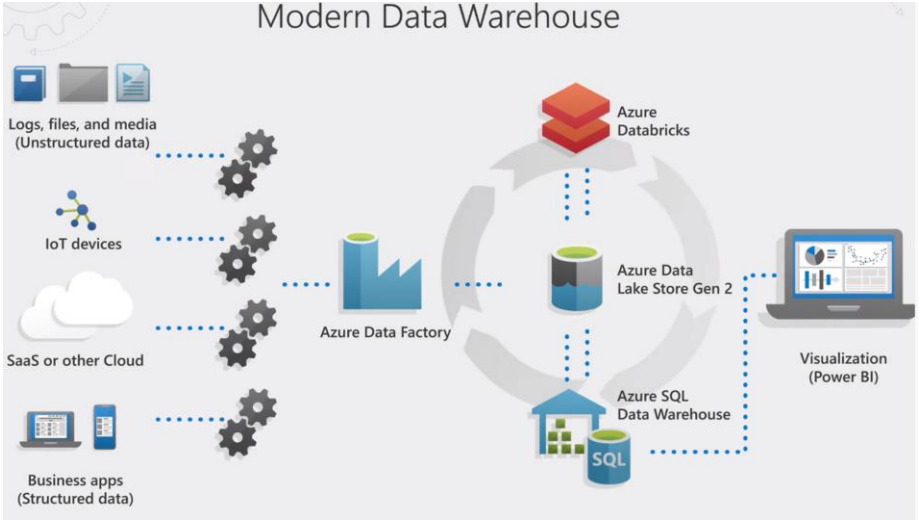


Source: Microsoft

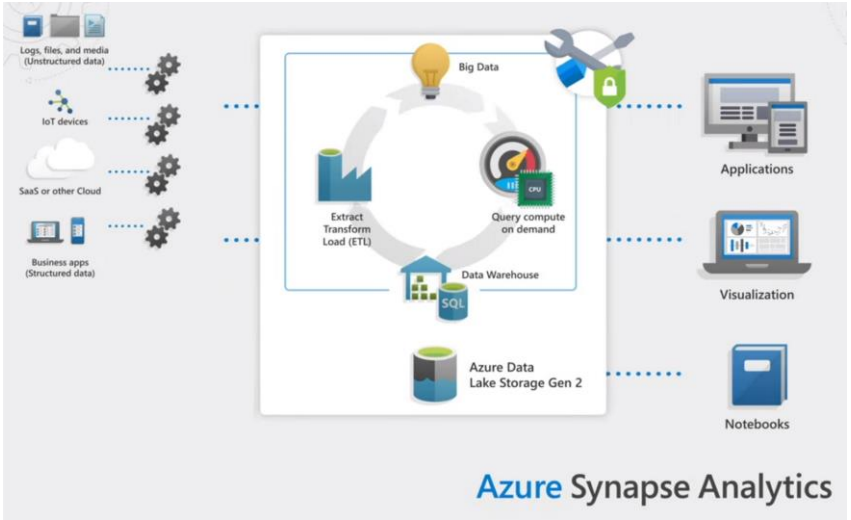
Azure Synapse Analytics

"Synapse is the next generation of Azure SQL Data Warehouse, blending big data analytics, data warehousing, and data integration into a single unified service that provides end-to-end analytics with limitless scale."

Modern vs Synapse Architecture

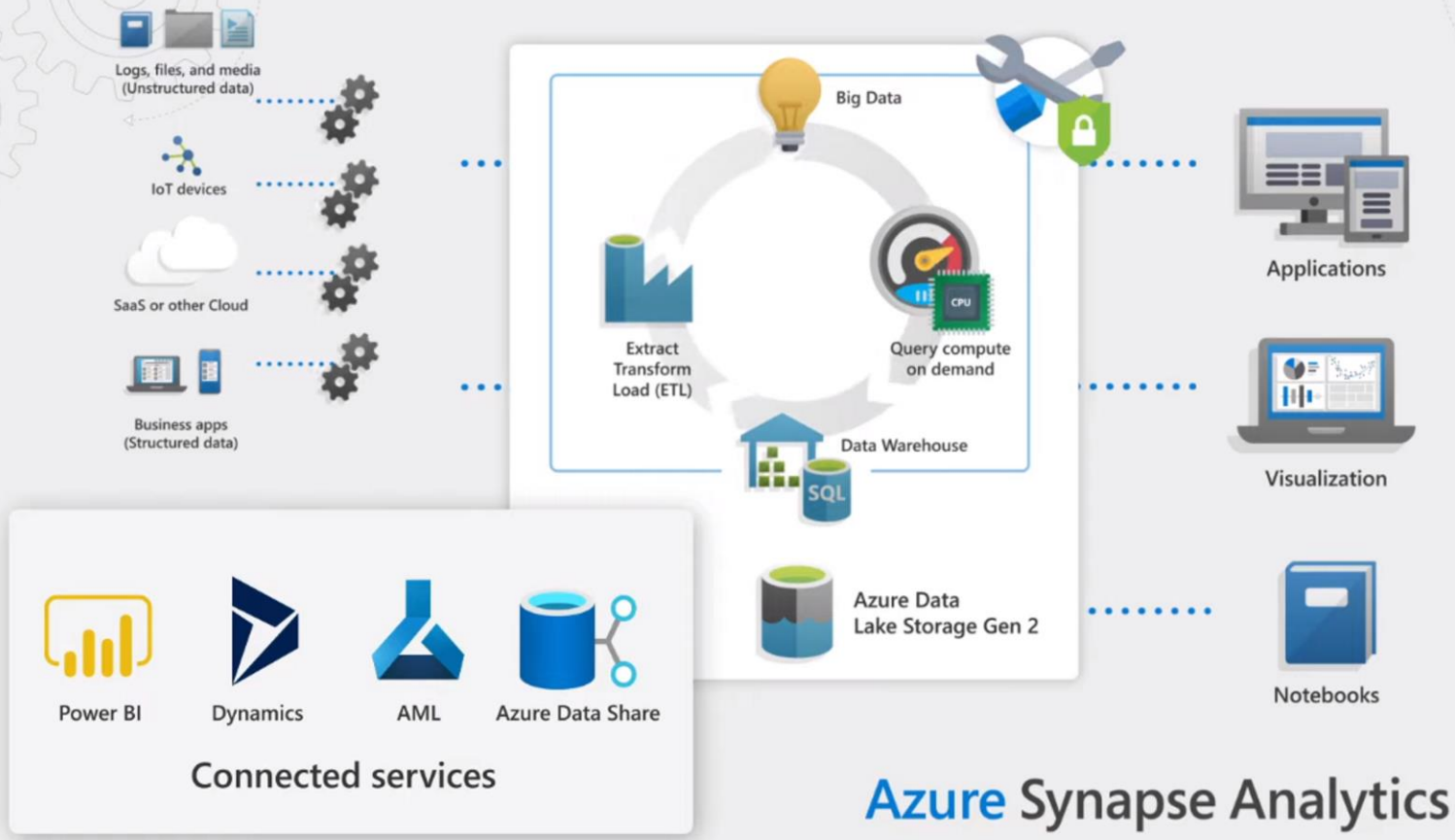


Modern Data Warehouse Architecture



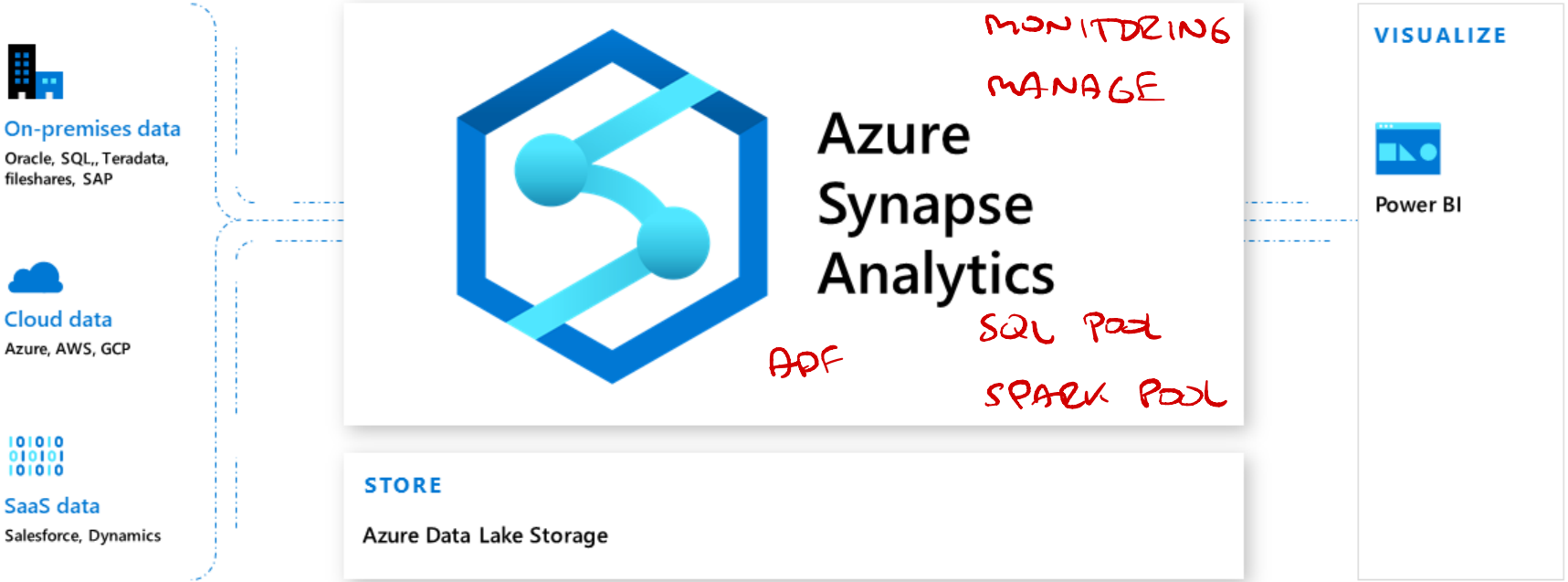
Synapse Analytics Architecture

Source: Microsoft



Source: Microsoft

Azure Synapse Analytics

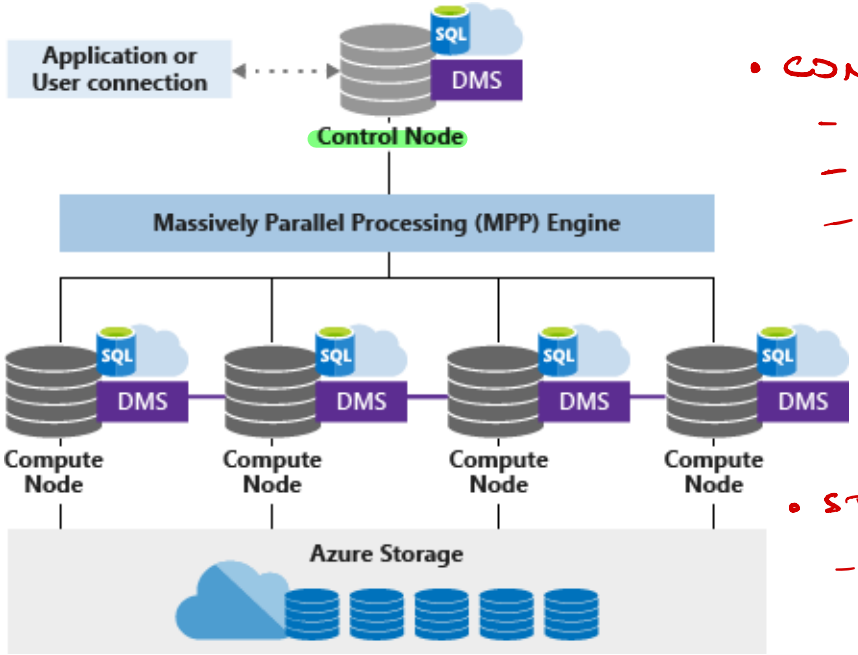


Source: Microsoft

Azure Synapse MPP Architecture

NODE BASED ARCHITECTURE

DWU	Loading 3 Tables	Ran Report
100	15	20
500	3	4



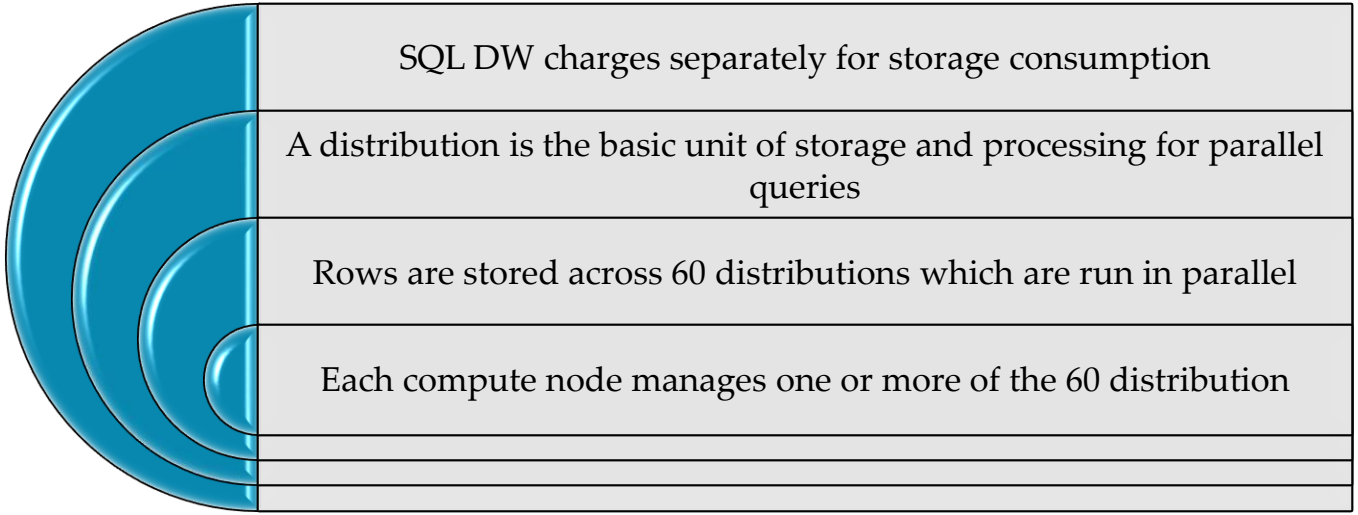
- CONTROL NODE
  - BRAIN
  - FRONT END
  - OPTIMIZE & COORDINATE PARALLEL QUERIES

- COMPUTE NODE
  - COMPUTATIONAL POWER
  - DISTRIBUTION MAP
  - DWU
  - 1-60 COMPUTE NODES

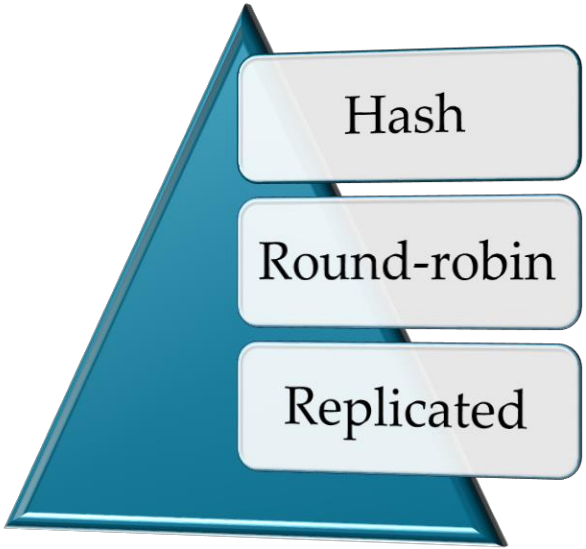
- STORAGE NODE
  - DECOUPLED FROM COMPUTE TO KEEP UNUSED DATA AT REST.

Source: Microsoft

Azure Storage and Distribution



Sharding Patterns

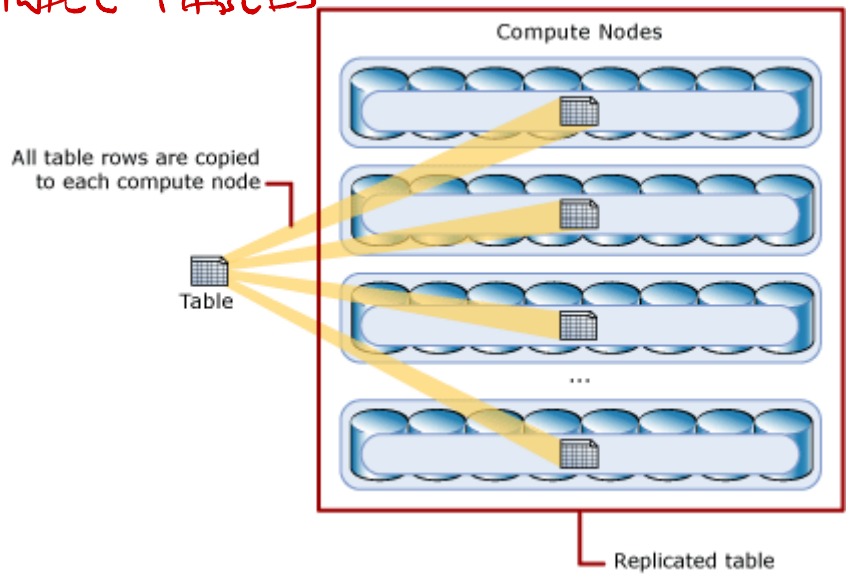


Replicated Tables

FAST QUERY PERFORMANCE FOR SMALL TABLES

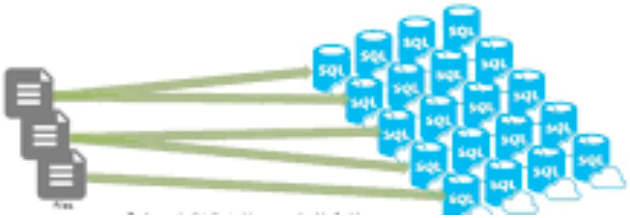
- Caches a full copy on each compute node.
- Used for small tables

```
CREATE TABLE [dbo].[BusinessHierarchies](
    [BookId] [nvarchar](250)  ,
    [Division] [nvarchar](100) ,
    [Cluster] [nvarchar](100) ,
    [Desk] [nvarchar](100) ,
    [Book] [nvarchar](100) ,
    [Volcker] [nvarchar](100) ,
    [Region] [nvarchar](100)
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX
    DISTRIBUTION = REPLICATE
,
)
;
```



Source: Microsoft

Round Robin tables



- Generally use to load staging tables
- Distribute data evenly across the table without additional optimization
- Joins are slow, because it requires to reshuffle data
- Default distribution type

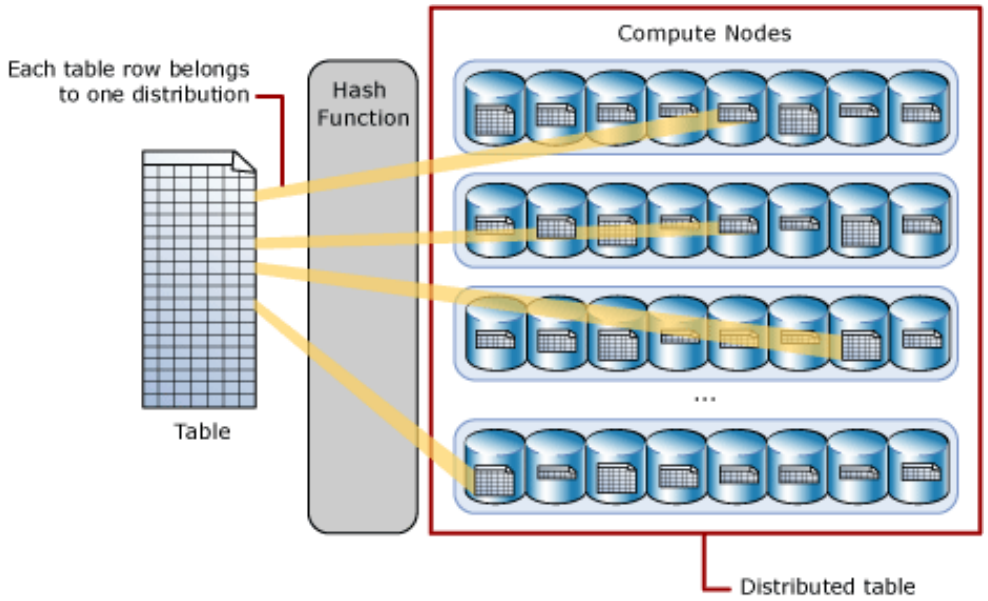
```
CREATE TABLE [dbo].[Dates](
    [Date] [datetime2](3) ,
    [DateKey] [decimal](38, 0) ,
    ..
    ..
    [WeekDay] [nvarchar](100) ,
    [Day Of Month] [decimal](38, 0)
)

WITH
(
    CLUSTERED COLUMNSTORE INDEX
     DISTRIBUTION = ROUND_ROBIN
,
)
;
```

Source: Microsoft

Hash Distribution Tables

- Highest performance for large tables
- Each row belong to one particular distribution BASED ON HASH-KEY
- It is used mostly for larger tables



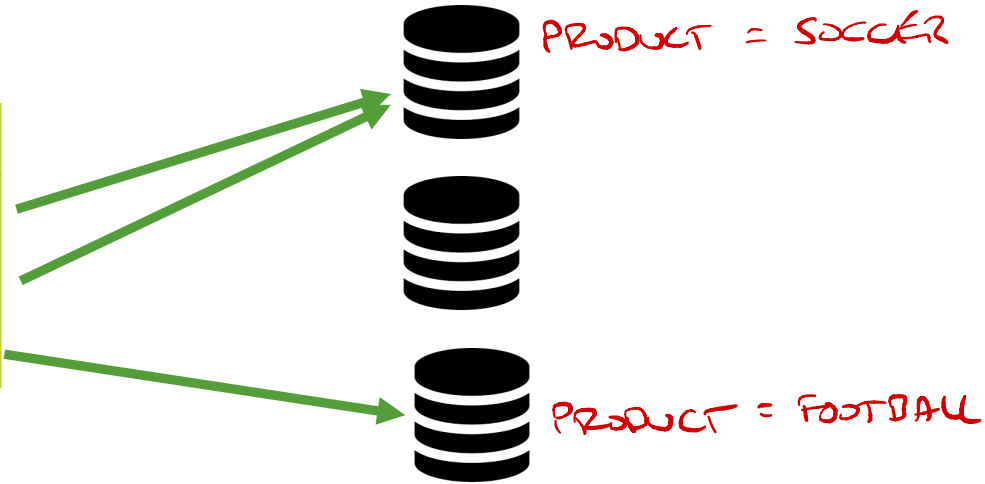

```
CREATE TABLE [dbo].[EquityTimeSeriesData](
   [Date] [varchar](30) ,
   [BookId] [decimal](38, 0) ,
   [P&L] [decimal](31, 7) ,
   [VaRLower] [decimal](31, 7)
)
WITH
(
   CLUSTERED COLUMNSTORE INDEX
,  DISTRIBUTION = HASH([P&L])
)
;
```

Source: Microsoft

Hash Distribution Tables

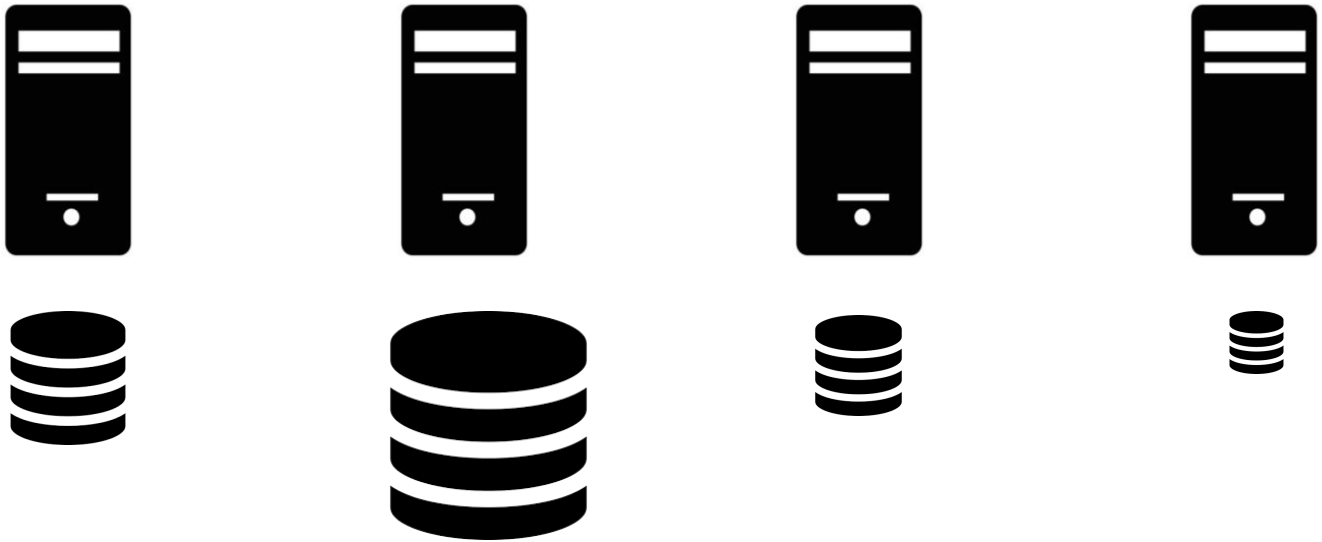
HASH-KEY

Record	Product	Store
1	Soccer	New York
2	Soccer	Los Angeles
3	Football	Phoenix

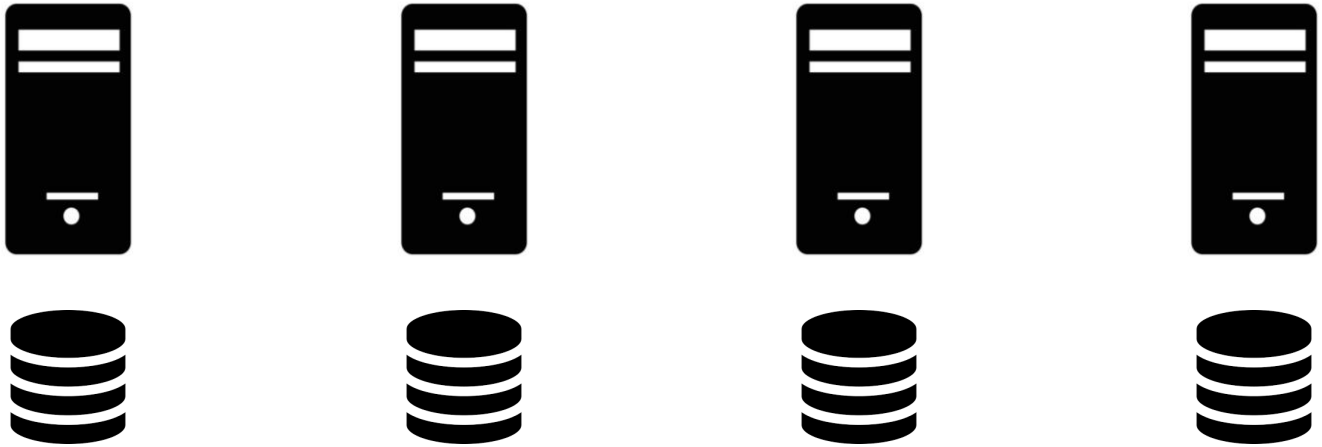


Avoid Data Skew

PICK EVENLY DISTRIBUTED HASH KEYS TO AVOID SKEW



Even Distribution



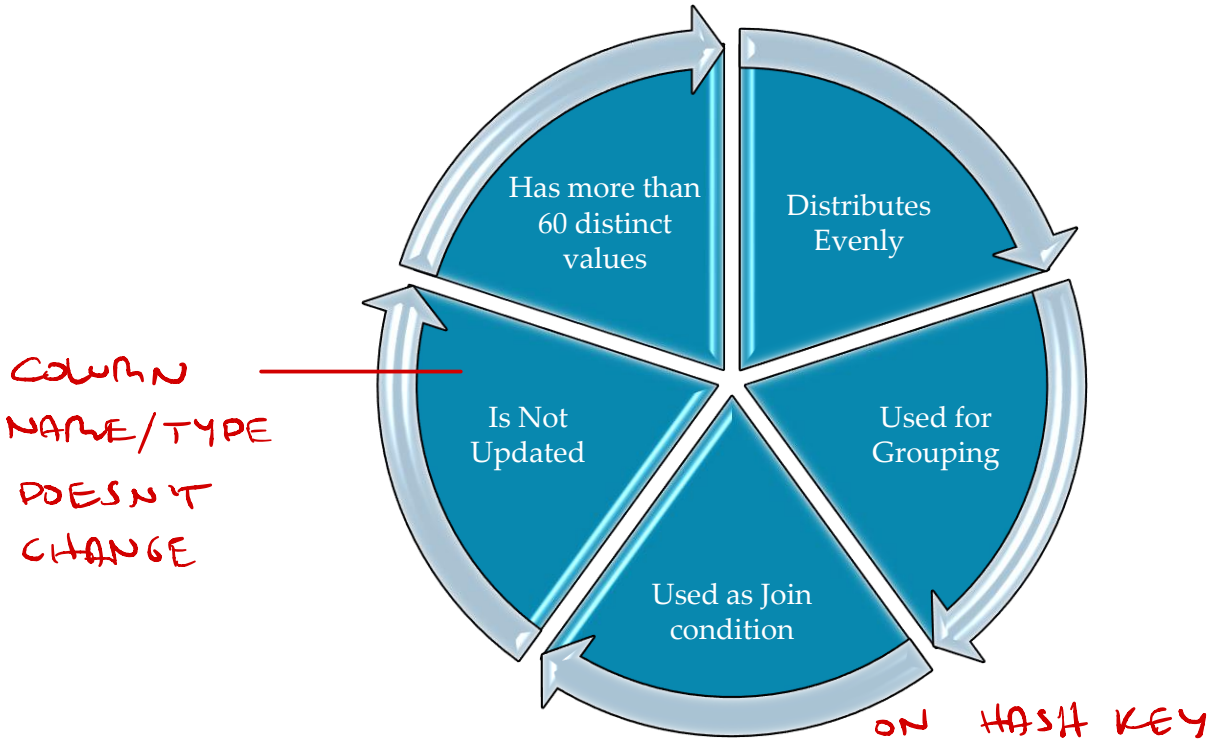
Distribution key

Determines the method in which Azure SQL Data Warehouse spreads the data across multiple nodes.

Azure SQL Data Warehouse uses up to 60 distributions when loading data into the system.

\ DISTRIBUTIONS ARE ASSIGNED TO DIFFERENT COMPUTE NODES

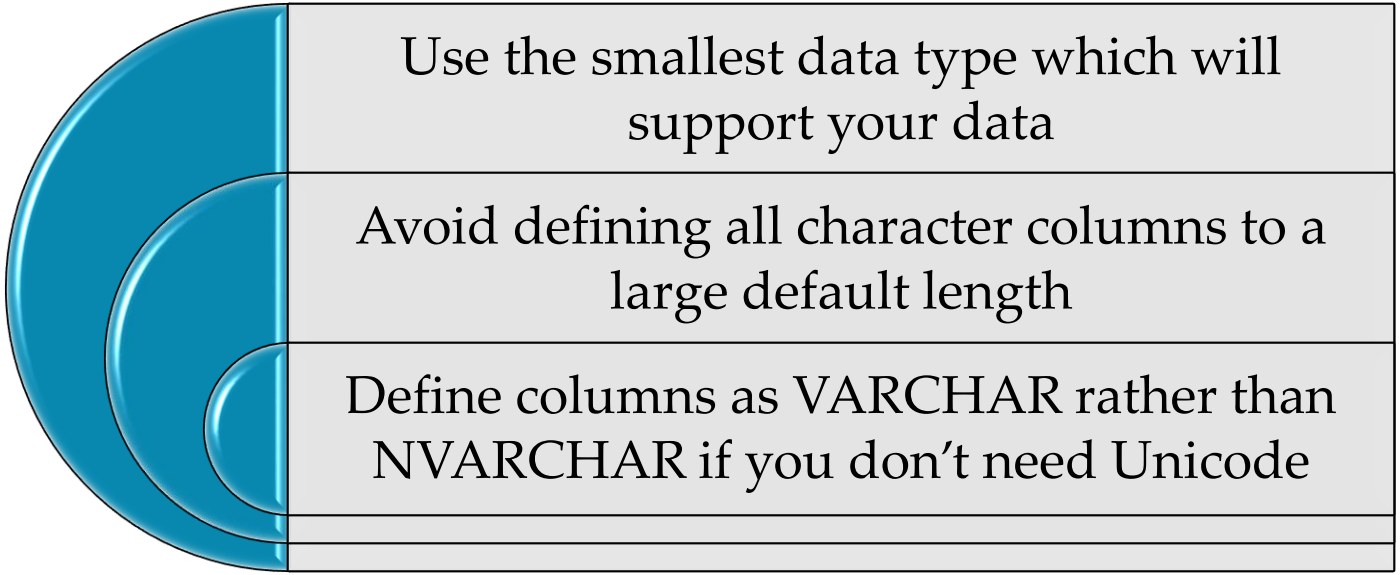
Good Hash Key



What Data Distribution to Use?

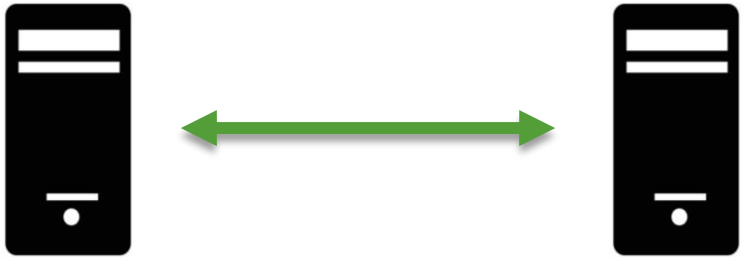
Type	Great fit for	Watch out if…
Replicated	Small-dimension tables in a star schema with less than 2GB of storage after compression	• Many write transaction are on the table (insert/update/delete) • You change DWU provisioning frequently • You use only 2-3 columns, but your table has many columns • You index a replicated table
Round-robin (default)	• Temporary/Staging table • No obvious joining key or good candidate column.	Performance is slow due to data movement
hash	• Fact tables • Large dimension tables	The distribution key can't be updated

Data types



NVARCHAR USES 2x SPACE (BYTES) AS VARCHAR

Data types

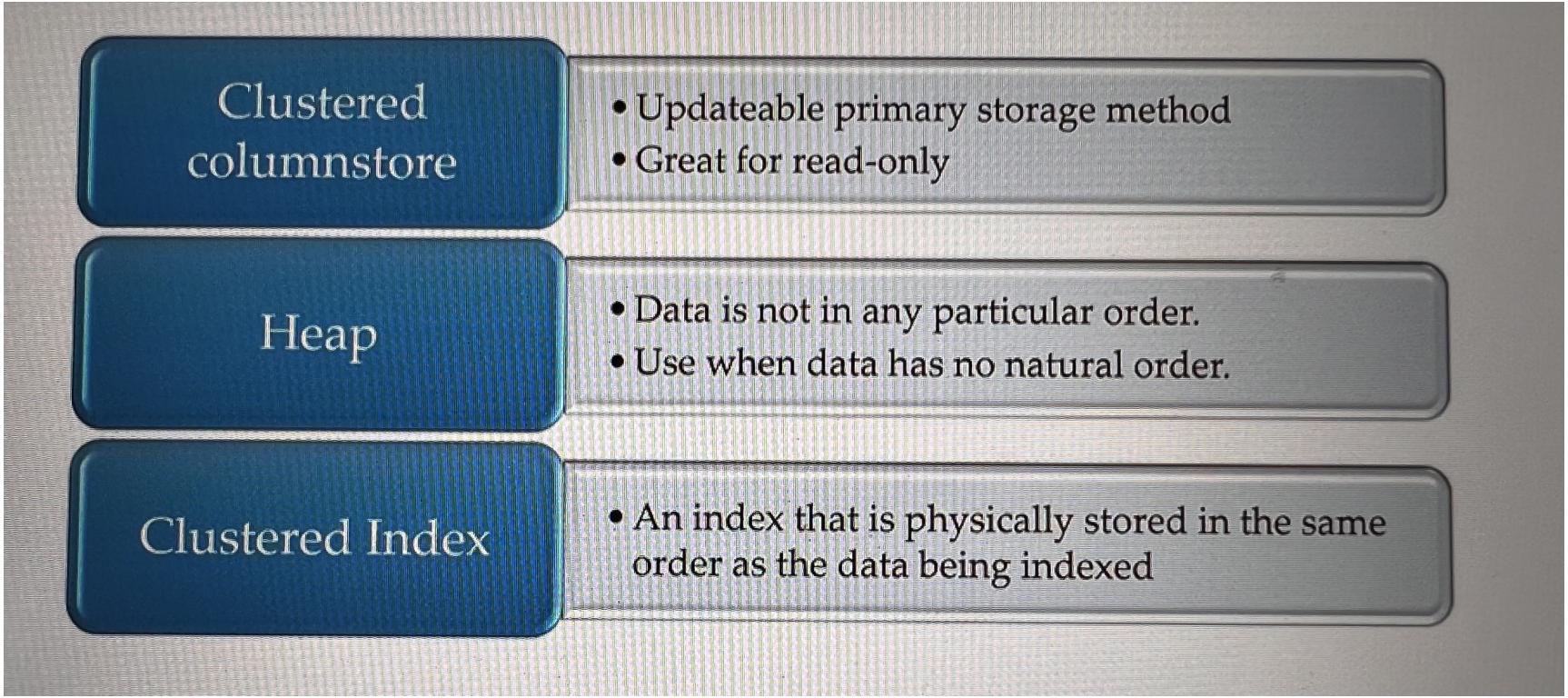


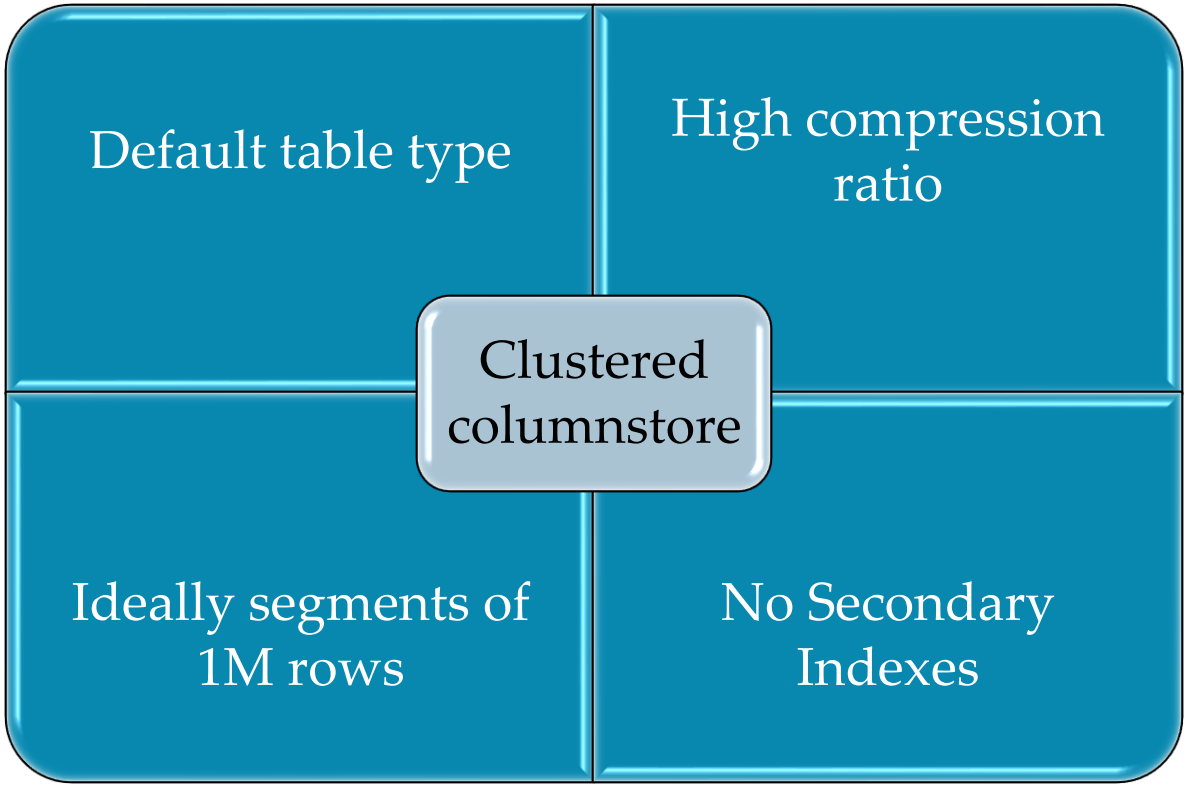
The goal is to not only save space but also move data as efficiently as possible.

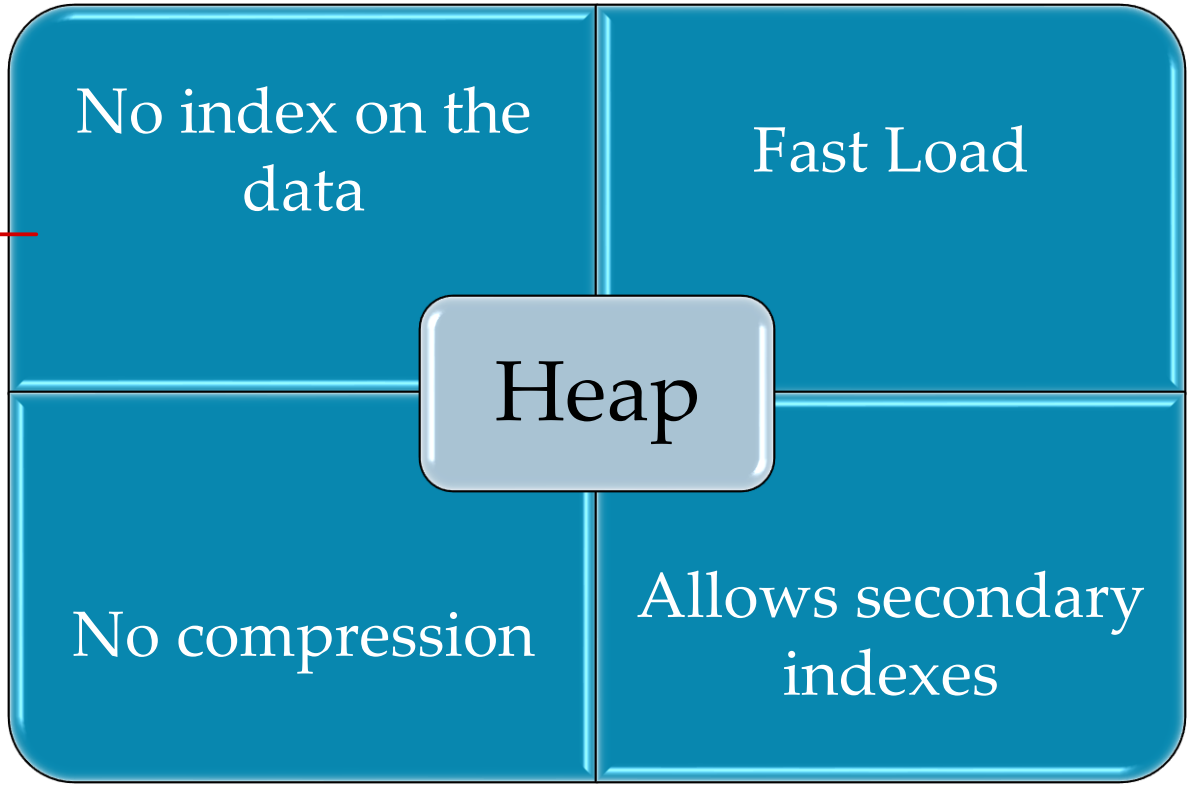
Data types

Some complex data types (XML, geography, etc) are not supported on Azure SQL Data Warehouse yet.

Table types


Clustered columnstore
• Updateable primary storage method
• Great for read-only
Heap
• Data is not in any particular order.
• Use when data has no natural order.
Clustered Index
• An index that is physically stored in the same order as the data being indexed


Default table type
High compression ratio
Clustered columnstore
Ideally segments of 1M rows
No Secondary Indexes


No index on the data
Fast Load
Heap
No compression
Allows secondary indexes

NO ORDERING

CLUSTERED - INDEX

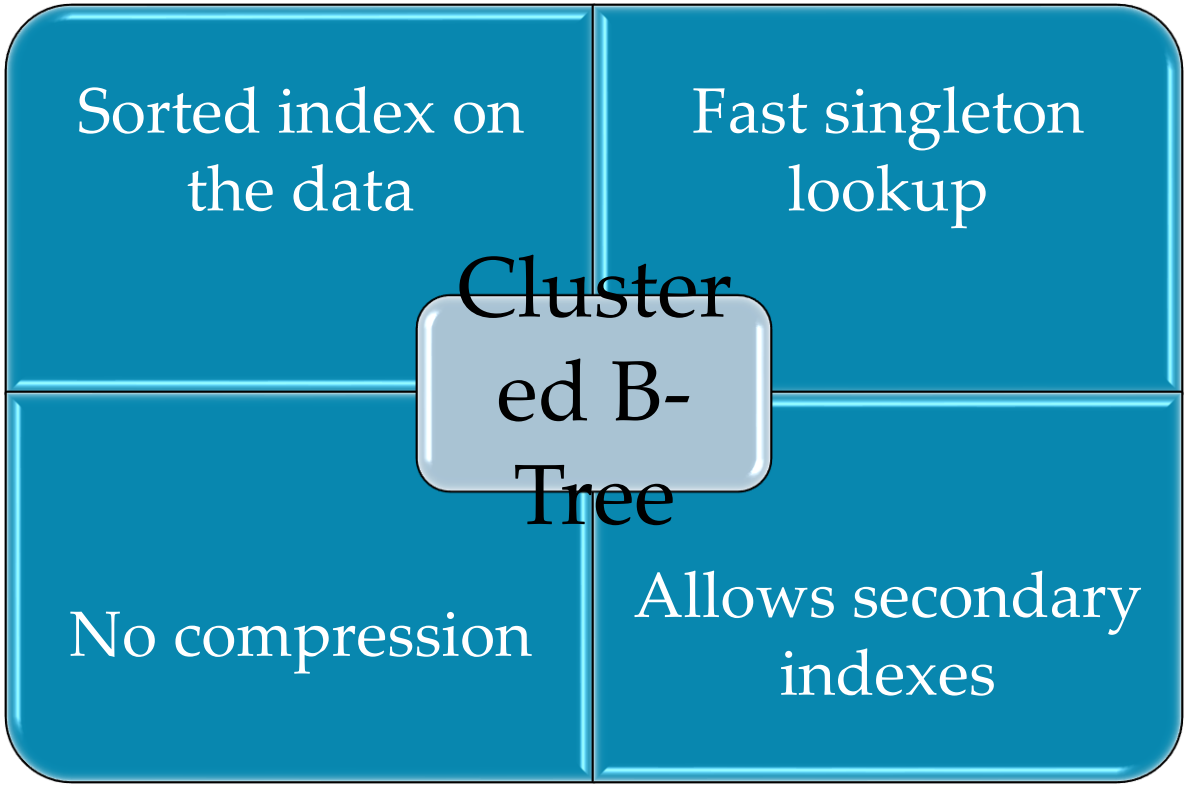
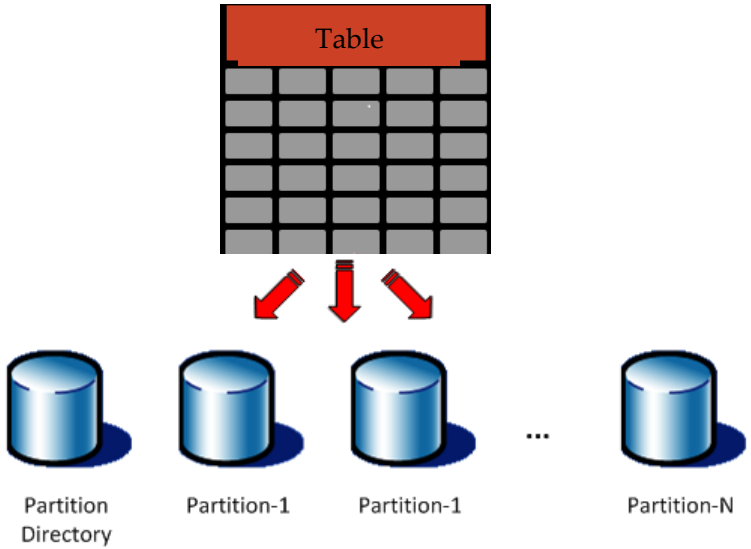
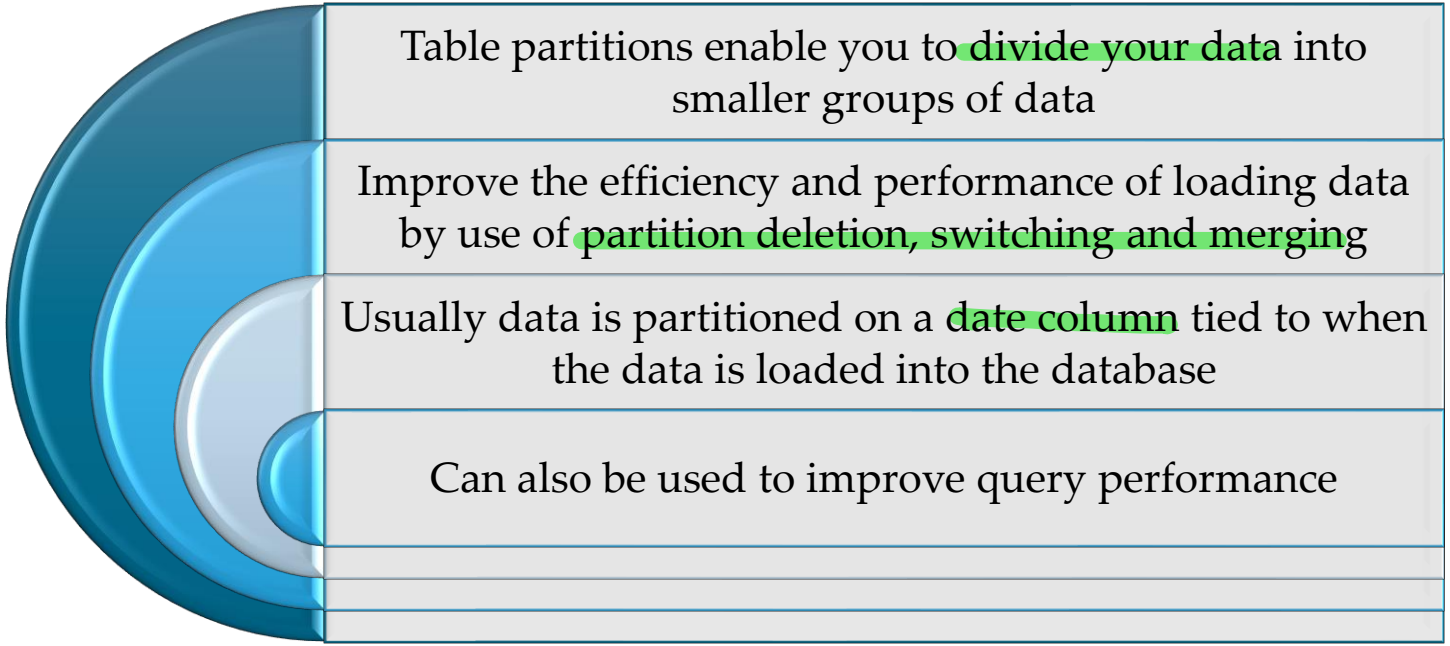


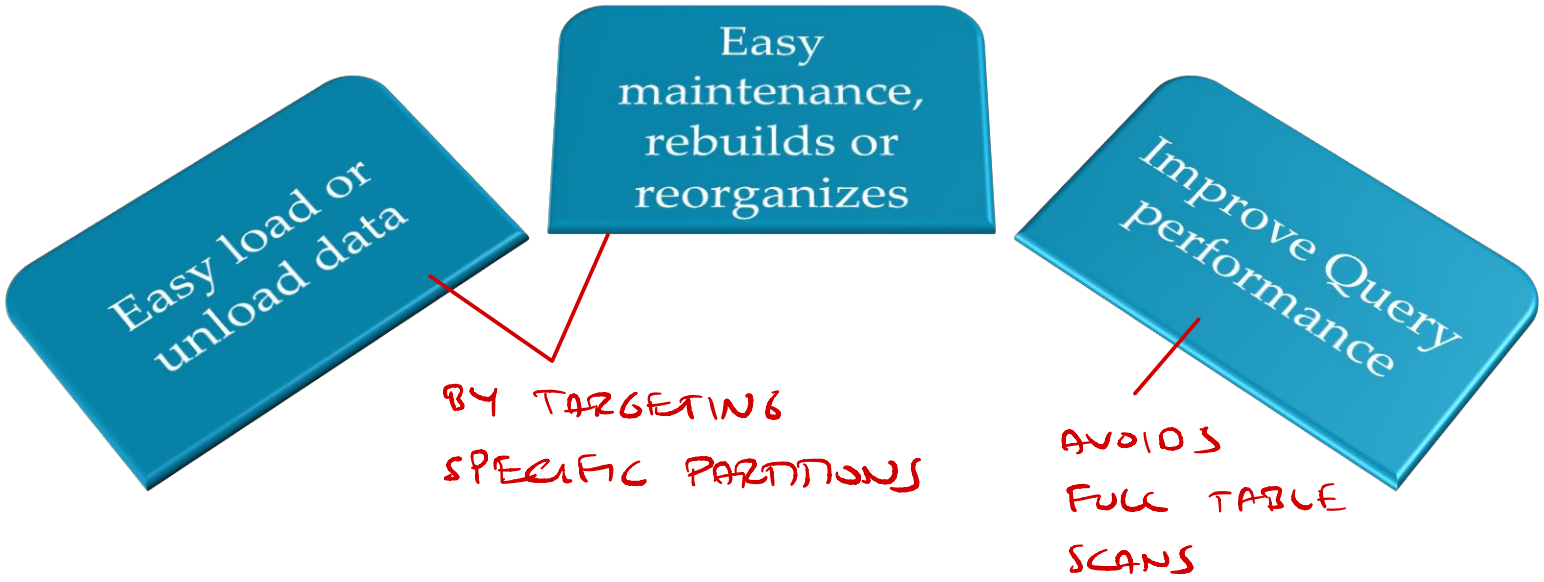
Table Partitioning



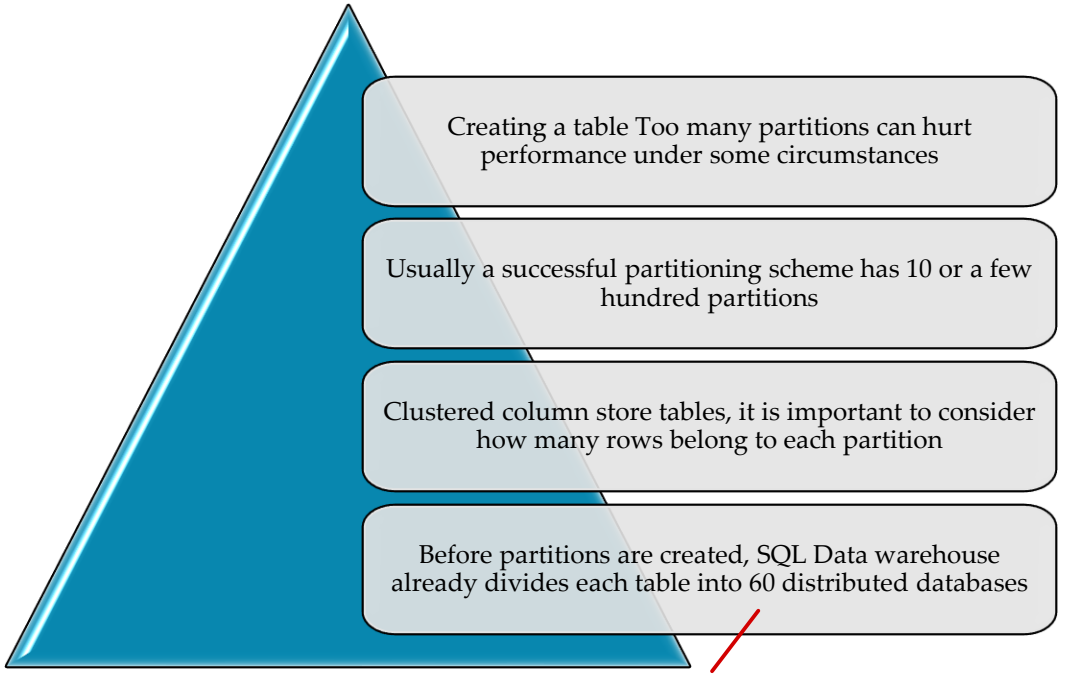
Partitioning



Why Partitioning?



Partitions best practices



A highly granular partitioning scheme can work in SQL Server but hurt performance in Azure SQL Data Warehouse.

Example

OVER-PARTIONING

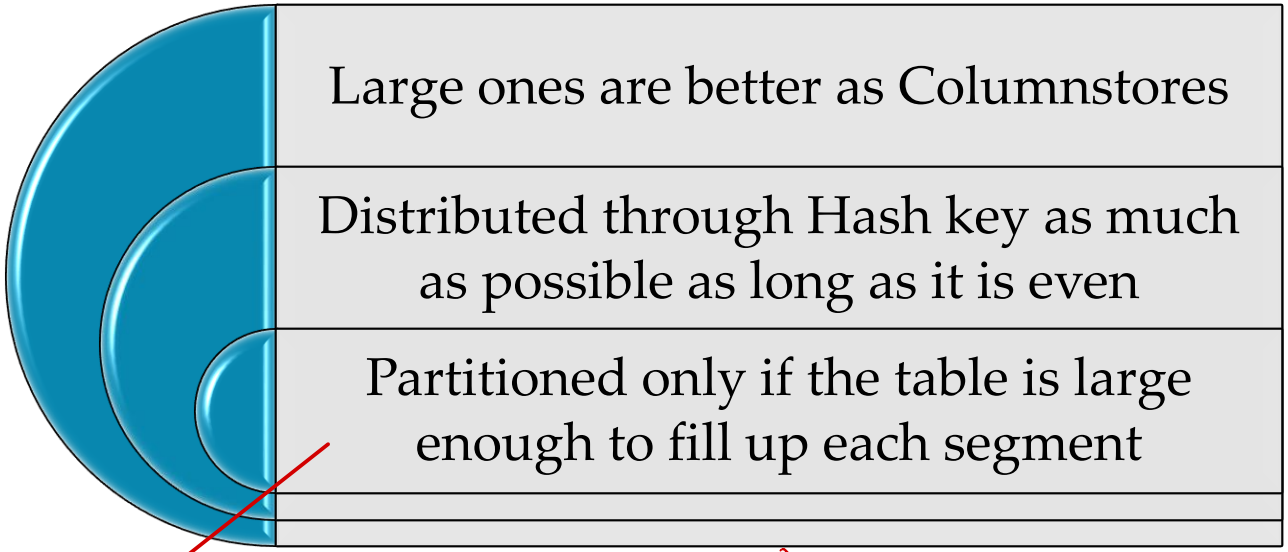
60 Distributions  DAILY 365 Partitions  21900 Data Buckets

21900 Data Buckets  Ideal Segment Size (1M Rows)  21 900 000 000 Rows

Lower Granularity (week, month) can perform better depending on how much data you have.

How do we apply these principles to a Dimensional model?

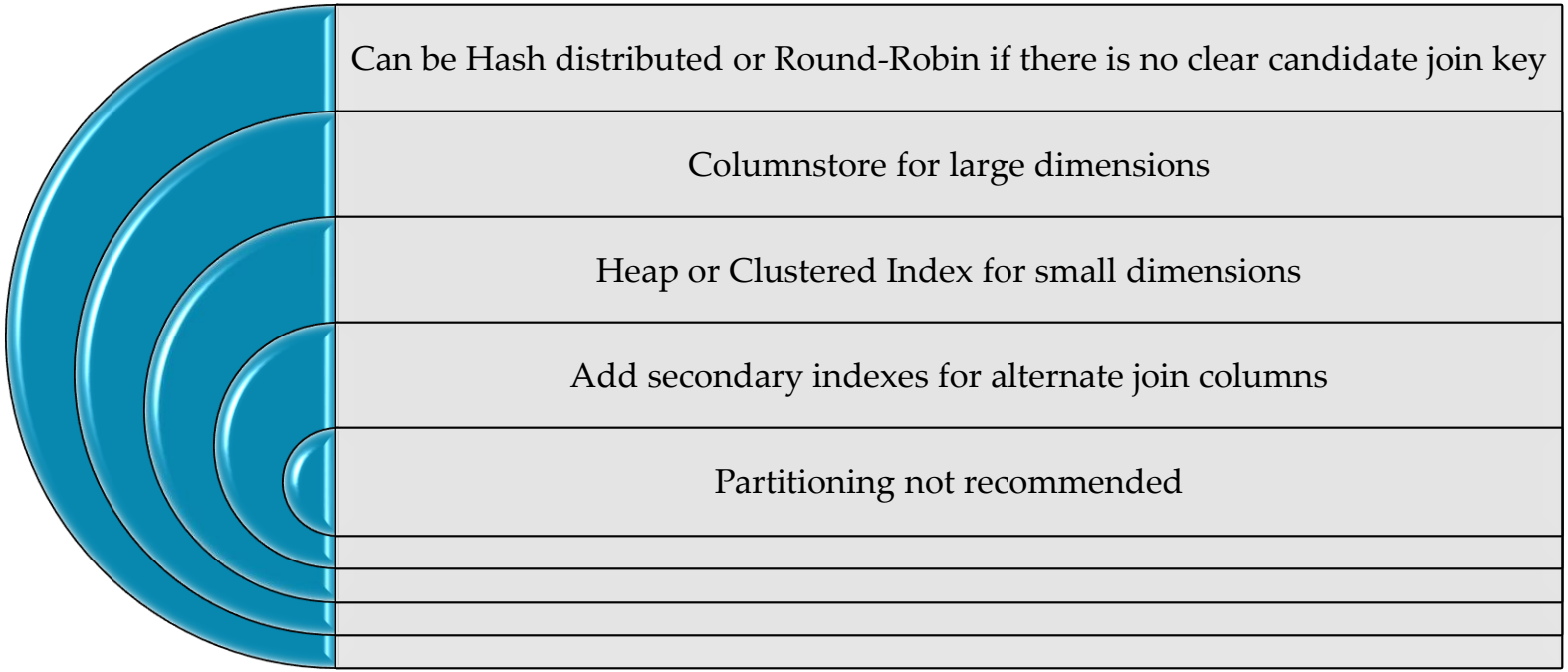
Fact Tables



IDEAL SEGMENT (PARTITION) = 1M ROWS

IDEAL TABLE SIZE = 60 PARTITION X 1M ROWS = 60M ROWS

Dimension Tables



Best Practices for Data Laod

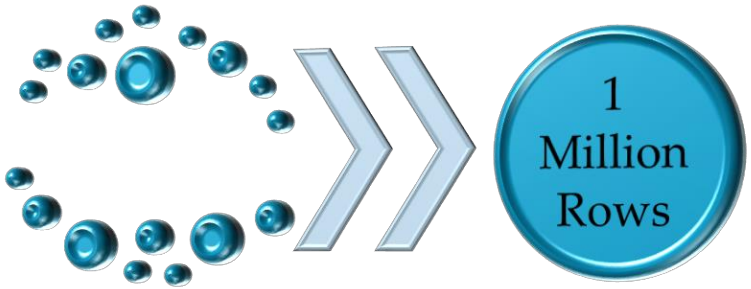
Data Warehouse Readers

Your DWUs have a direct impact on how fast you can load data in parallel

No of DWU	**100**	**200**	**300**	**400**	**500**	**600**	**1000**	**1500**	**2000**
Readers	8	16	24	32	40	48	60	60	60
Writers	60	60	60	60	60	60	60	60	60

Optimize Insert Batch Size

Avoid trickle insert pattern. Ideal batch size is 1 million or more direct or in a file

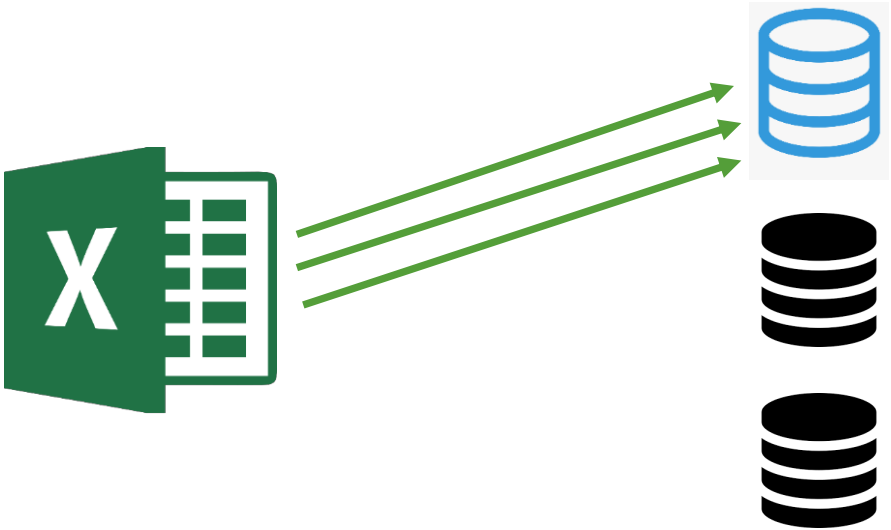


Combine rows to make it a batch of 1 million

Ideal batch size

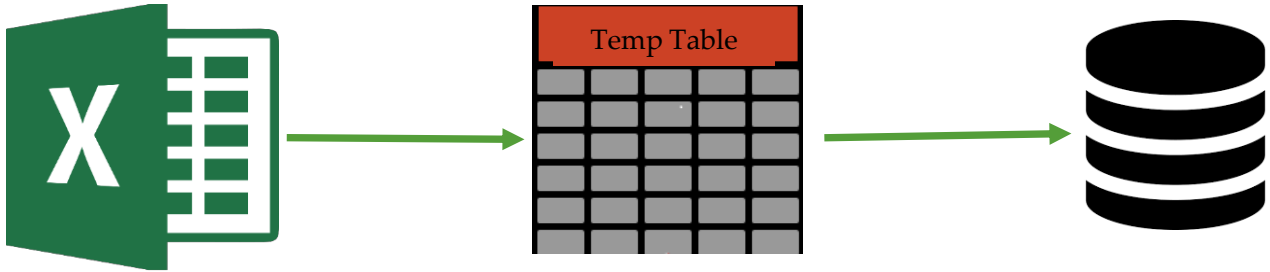
Avoid ordered data

Data ordered by distribution key can introduce hot spots that slow down the load operation



Using temporary tables

Stage and transform on a Temp Heap table before moving to permanent storage.



CREATE TABLE AS

```
CREATE TABLE #tmp_fct
WITH
(
DISTRIBUTION = ROUND_ROBIN
)
AS
SELECT *
FROM
[dbo].[FactInternetSales];
```

- Fully Parallel operation
- It is minimally logged
- It can change: distribution, table type, partitioning

Loading Methods

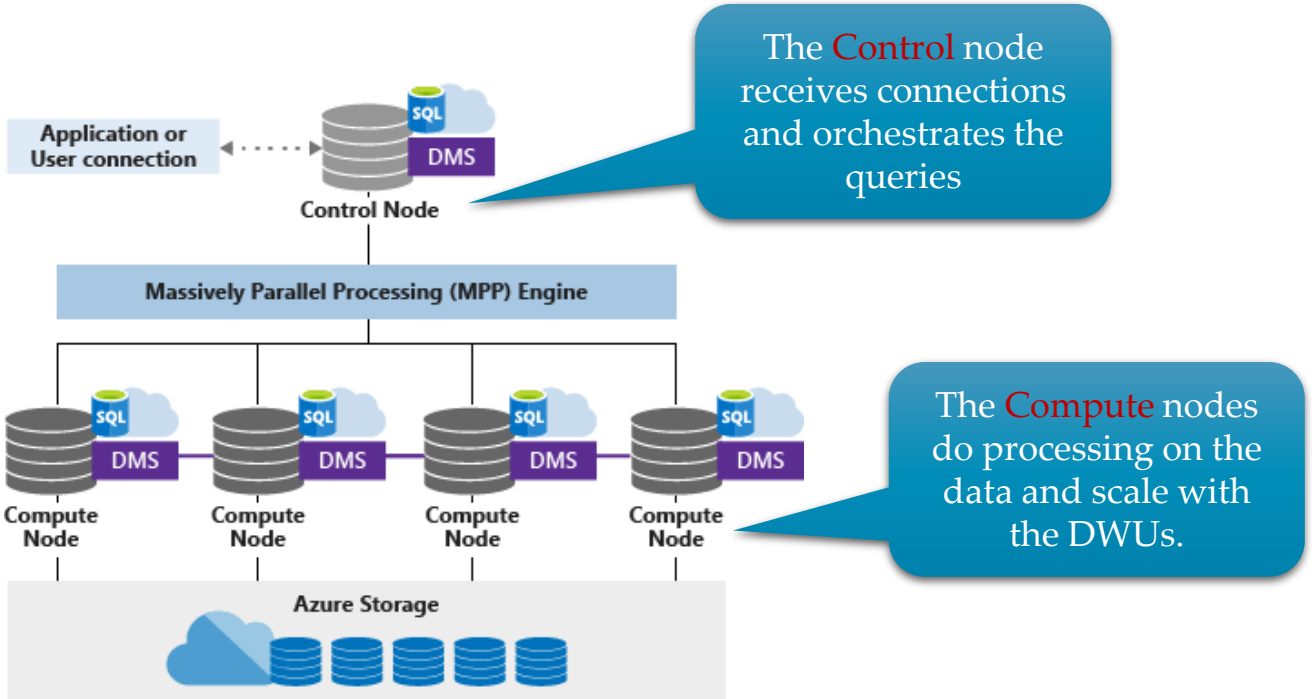
Single client loading methods

- SSIS
- Azure Data Factory
- BCP
- Can add some parallel capabilities but are bottlenecked at the control node

Parallel readers loading methods

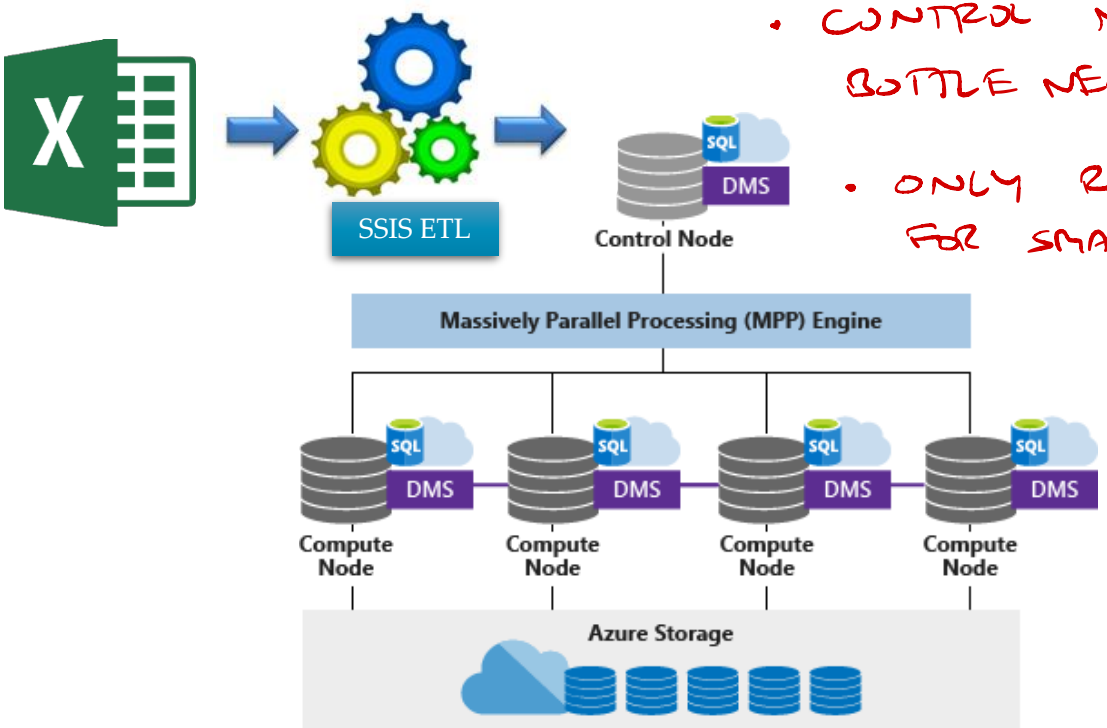
- PolyBase
- Reads from Azure blob Storage and loads the contents into Azure SQL DW
- Bypasses the Control Node and loads directly into the Compute Nodes

Control Node



Source: Microsoft

Loading with SSIS

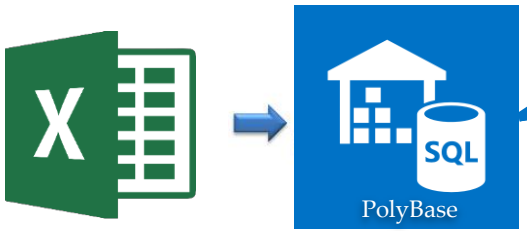


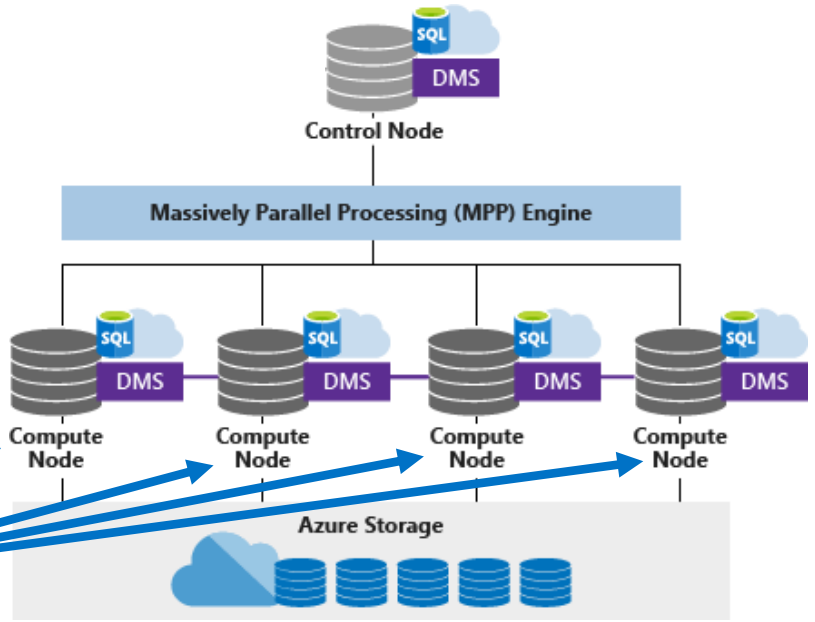
- CONTROL NODE CREATES BOTTLE NECK
- ONLY RECOMMENDED FOR SMALL LOADS

Source: Microsoft

Loading with PolyBase

- RECOMMENDED FOR LARGE LOADS
- BYPASSES CONTROL NODE FOR FULL MPP CAPABILITIES





Source: Microsoft

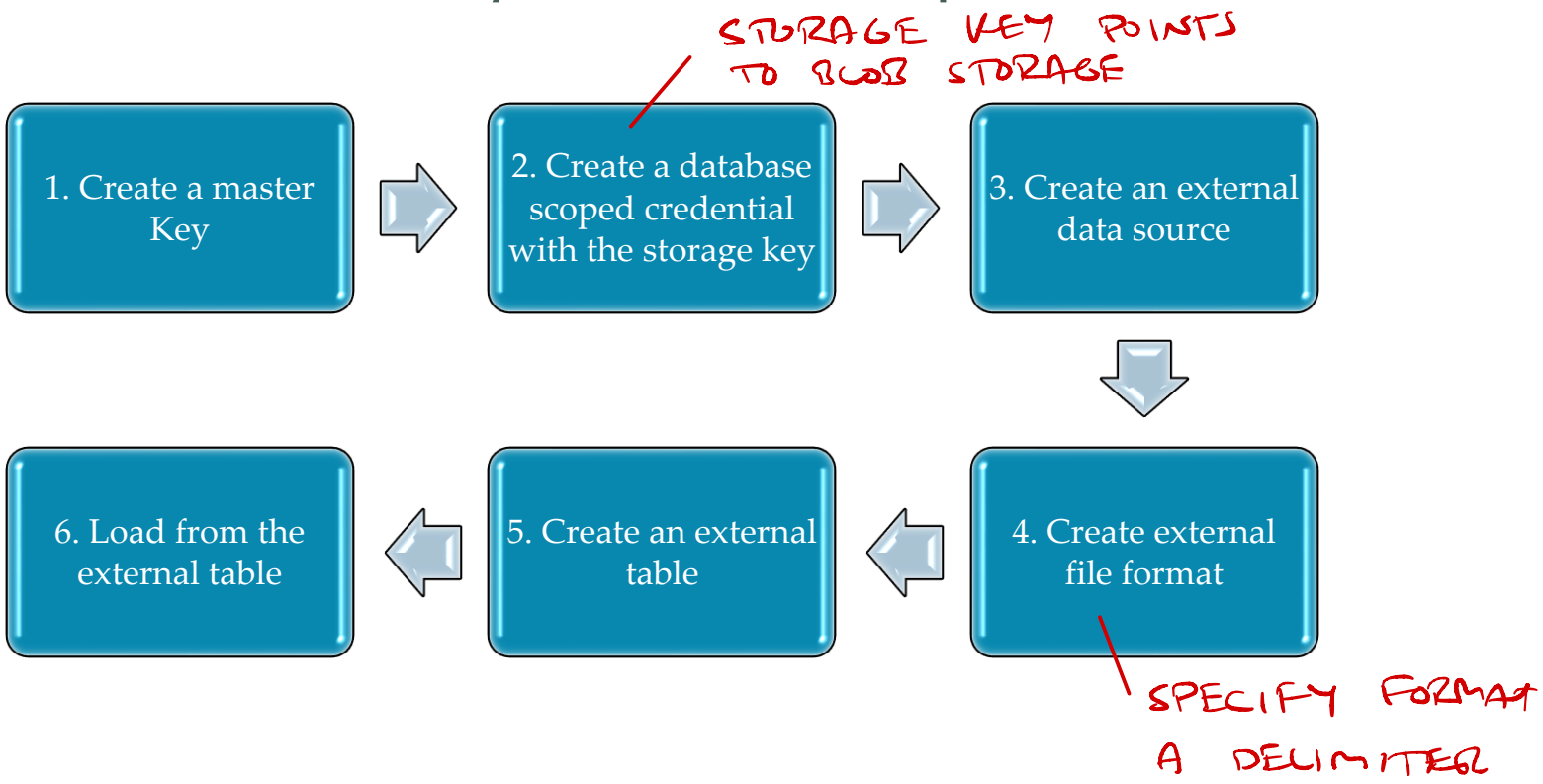
PIPE, COMMA, TAB DELIMITED

PolyBase can load data from UTF-8 delimited text files and popular Hadoop file formats like RC File, ORC, and Parquet

CAN COMPRESS FILES BEFORE LOADING

Multiple readers will not work against a compressed file

PolyBase Setup



Demo – PolyBase

1. Export table to flat file
2. Create blob storage account
3. Upload flat file to blob storage
4. Run PolyBase 6 steps process
5. Monitor and confirm successful migration
6. Confirm 60 distributions in destination table

Azure Synapse Analytics

Azure SQL Warehouse Scaling

Scaling in SQL Data Warehouse

- SQL DW allows us to scale or pause at will
- Scale up during heavy demand
- Pause to cut cost
- DWUs are CPU, memory, and I/O bundled into units of compute scale
- Increasing DWU's
  - increase query performance
  - Also increase maximum number of concurrent queries and concurrent sl
- Modify with GUI, PowerShell, or TSQL

Azure SQL DW Backup and Restore

DIFFERENT FROM SQL DATABASE

- Snapshots of your data warehouse are taken throughout the day creating restore points
- These restore points are available for 7 days
  - Retention period of 7 days cannot be changed
- SQL pool supports an 8 hour recovery point objective (RPO)
- Replicated to paired region once a day
- You can also take user-defined snapshots
  - Retention period 7 days cannot be changed
  - 42 max restore point possible
  - Can be created using PowerShell or portal — OVERVIEW
- When you drop a SQL pool, a final snapshot is created and saved for seven days
  - SQL Pool should not be in paused state *

CAN RESTORE FROM AUTOMATIC OR USER-DEFINED RESTORE POINTS

Azure SQL DB vs. Azure SQL DW

CRUD = CREATE, READ, UPDATE, DELETE

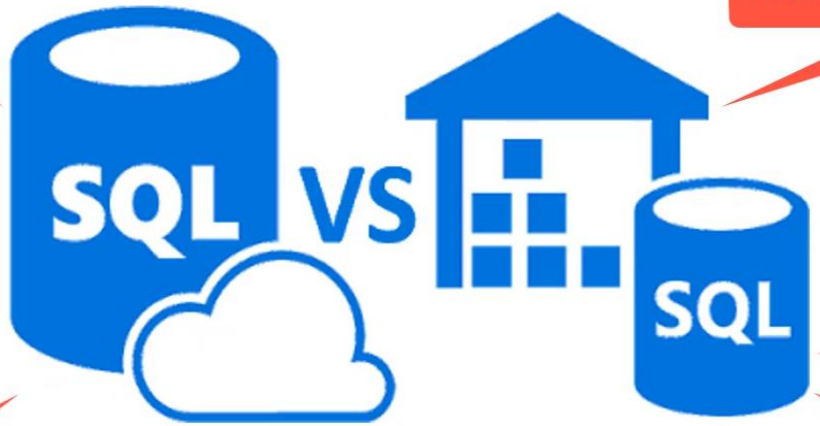
OLTP/CRUD

SYMMETRIC MULTI-PROCESSING

SMP

Vertical scale

No Polybase



OLAP/querying and reporting

60 COMPUTE NODES

MPP

Horizontal Scale

Polybase

Can pause the virtual server to save cost

Azure SQL Database

Azure SQL Data Warehouse

Dynamic Data Masking

CAN IMPLEMENT IN AZURE PORTAL : SQL DB > DYNAMIC DATA MASKING

OR WITH T-SQL SCRIPTS

ID	PersonName	EmailAddress	CreditCardNumber	SocialSecurityNumber
1	Anoop Kumar	abcdefgh@hotmail.com	1234-5678-4321-8765	123-45-6789
1	Rahul Gupta	amitguptaabcdefg@hotmail.com	8765-1234-5678-4321	231-45-6787
1	Amit Goel	amitgoelabcdefgh@hotmail.com	4321-1234-5678-4321	321-45-6700

ID	PersonName	EmailAddress	CreditCardNumber	SocialSecurityNumber
9590	AXXXr	aXXX@XXXX.com	xxxx-xxxx-xxxx-8765	xxxx
7604	RXXXa	aXXX@XXXX.com	xxxx-xxxx-xxxx-4321	xxxx
8453	AXXXl	aXXX@XXXX.com	xxxx-xxxx-xxxx-4321	xxxx

Random Number function – generates random number based on selected boundaries and actual data type. Can be applied only numbers, not string

Email function – Exposes the first letter and replace the domain with xxx.com

Default function – Full masking of data.
For numeric – 0
For String – XXXX characters

Credit Card function - Only the last four digits of Credit card are shown

Custom String function – You can define the exposed prefix, the padding string and exposed suffix

Dynamic Data Masking

- **Limit the exposure of sensitive data to non-privileged users**
  - You can decide the level of exposure of data
- **No change in physical layer**
  - Data in the database is not changed
  - Not the same as data encryption
- **No additional development effort needed at application level**
- **Security: Should not be used as a primary security layer**
  - Dynamic Data Masking should not be used as an isolated measure to fully secure sensitive data
  - ad-hoc query permissions can apply techniques to gain access to the actual data.
- **Other considerations**
  - Masked columns can be updated if user has permission
  - Export masked from source data results in masked data in target table

Dynamic Data Masking

- **How to Enable DDM?**
  - Portal OVERVIEW > DYNAMIC DATA MASKING
  - Powershell
    - Get-AzSqlDatabaseDataMaskingRule
    - New-AzSqlDatabaseDataMaskingRule
    - Remove-AzSqlDatabaseDataMaskingRule
    - Set-AzSqlDatabaseDataMaskingRule
  - Rest API

1

Data at Rest

Inactive data that is not currently being transmitted across a network

2

Data in motion

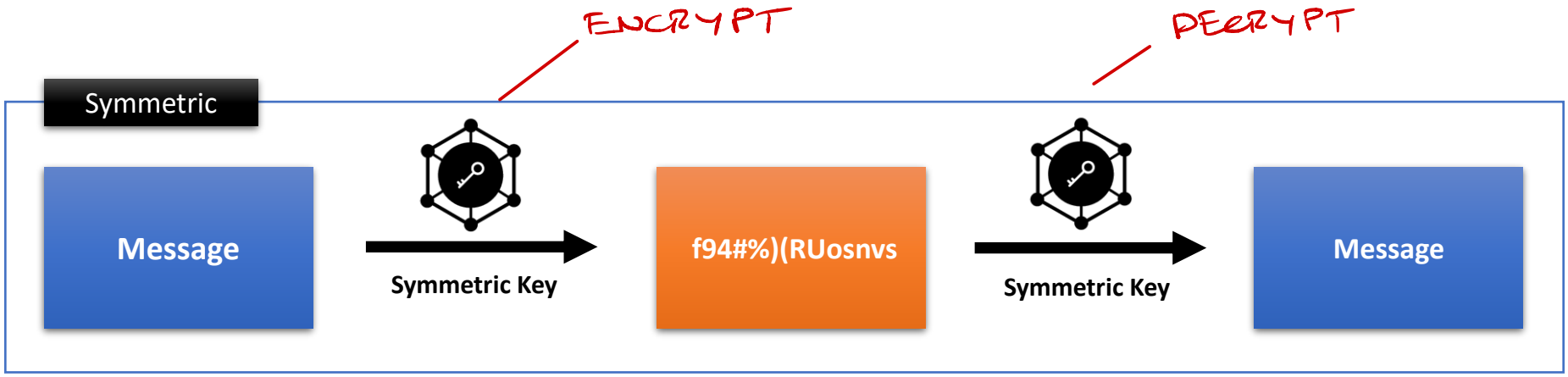
Data that is currently travelling across a network

3

Data in Use

Data currently in the process of being generated, updated, appended or erased

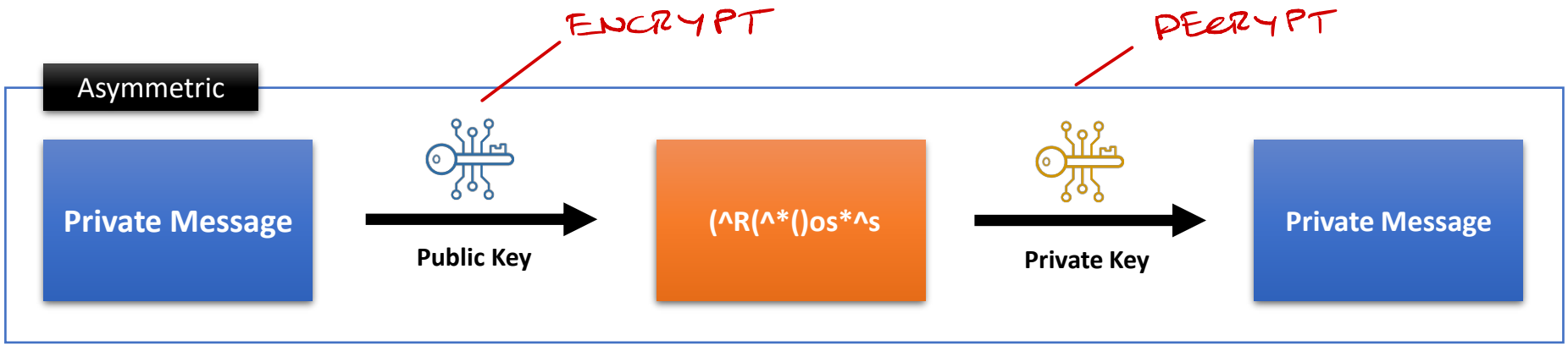
Types of Encryption



SAME KEY TO ENCRYPT AND DECRYPT

- Highly performant
- Secure key handling is difficult

Types of Encryption



DIFFERENT KEYS TO ENCRYPT & DECRYPT

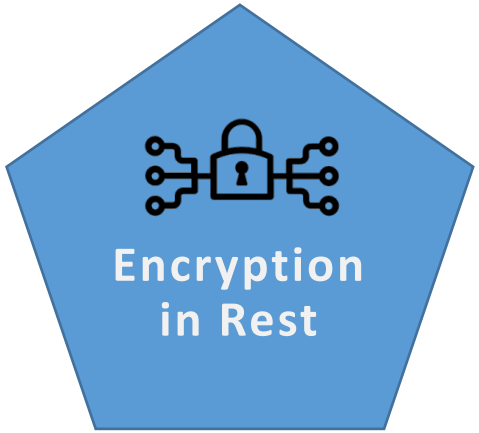
- Set of keys (Public and Private) where one is use to encrypt and other to decrypt
- Computationally more expensive, but solves the key handling problem

PUBLIC KEYS CAN BE STORED & SHARED IN AZURE KEY VAULT

Encryption in Rest

OPTION TO TURN OFF IN SQLDB & SYNAPSE

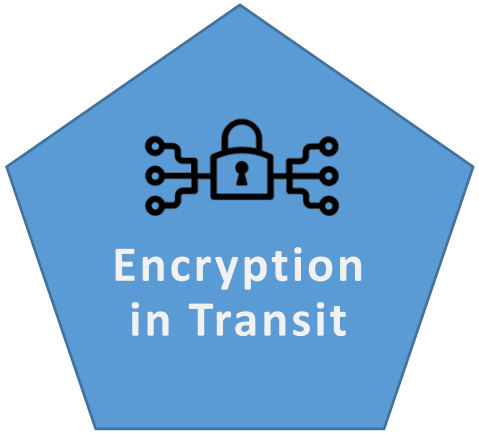
SQLDB SETTINGS CAN BE SET AT THE SERVER LVL



- By default all storage services encrypted data at rest
- 256-bit AES encryption
- Keys are stored in Azure Key Vault

OPTION TO USE MSFT MANAGED (DEFAULT) KEY OR CUSTOMER-MANAGED KEY

Encryption in transit



- Any communication over the internet to Azure services is typically encrypted via SSL/TLS protocols
- Utilize site-to-site VPN or point-to-site VPN connections
  WHEN USING ON-PREM TO VPN
- Or utilize ExpressRoute
- ExpressRoute communication is a private connection and not encrypted but workloads can be encrypted via SSL/TLS
- Storage services can be configured to require secure transfer
  L UNDER CONFIGURATION PAGE

Types of Encryption

Deterministic encryption:

- This will always generate the same encrypted value for any plain text value.
- You can perform point lookups, equality joins, grouping and indexing on encrypted columns.

Randomized encryption:

- This is more secure than deterministic encryption because the encrypted value is generated in a less predictable manner.
- But you can't perform searching, grouping, indexing or joins on encrypted columns.

Types of Encryption

Column Encryption Key (CEK)

- Used to encrypt values in specific columns
- Encrypted versions of each CEK is stored in the database.

Column Master Key (CMK)

- Used to encrypt all the CEKs
- Must be stored externally in a secure key store
- Key store providers: Azure key vault, certificate store, HSM