SQL Server 2017

Kevin Howell

Data & AI Technical Solutions Professional

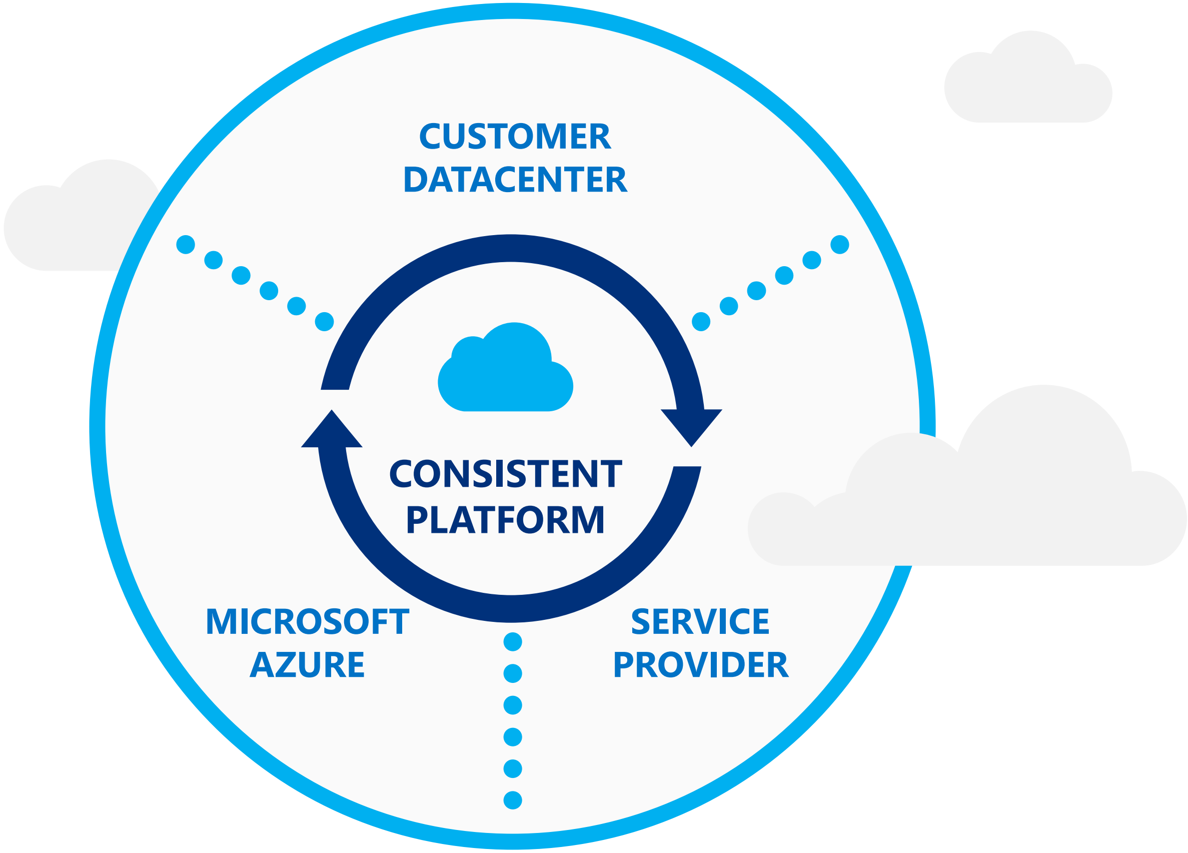
November 2017

Agenda – SQL Server 2017

- Microsoft Vision & Overview
- SQL 2017 New Features
- SSAS New Features
- SSIS New Features
- SQL 2017 Editions
- SQL on Linux
- Graph Data & Queries
- Spatial
- T-SQL
- JSON
- Adaptive Query Processing
- Automatic Tuning
- In-Memory OLTP & Columnstore
- High Availability
- Migration Tools
- Bots**

Microsoft vision for a new era

United platform for the modern service provider



Enterprise-grade

Global reach, scale, and security to meet business demands

Hybrid cloud

Consistent platform across multiple environments and clouds

People-focused

Expands technical skill sets to the cloud for new innovation

1/10th the cost of Oracle

SQL SERVER 2017

INDUSTRY-LEADING PERFORMANCE AND SECURITY NOW ON LINUX AND DOCKER

Choice of platform and language

- T-SQL
- Java
- C/C++
- C#/VB.NET
- PHP
- Node.js
- Python
- Ruby

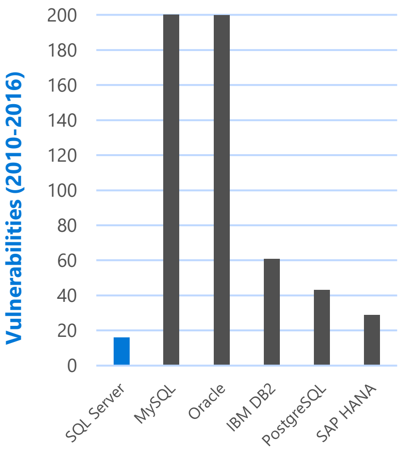
Industry-leading performance

#1 OLTP performance

#1 DW performance

#1 price/performance

Most secure over the last 7 years

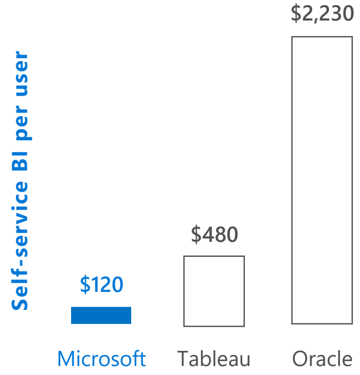


Only commercial DB with AI built-in

R and Python + in-memory at massive scale

Native T-SQL scoring

End-to-end mobile BI on any device



A fraction of the cost

In-memory across all workloads

Private cloud

Most consistent data platform

Public cloud

SQL Server 2017

Meeting you **where** you are

It's the same SQL Server Database Engine that has many features and services available for all your applications—regardless of your operational ecosystem.

{ }

1010
0101
0010

Any data

T-SQL
Java
C/C++
C#/VB.NET
PHP
Node.js
Python
Ruby

Any application

Anywhere

Windows Server
Linux
docker

Choice of platform

How we develop SQL

SQL Server and APS

Azure SQL Virtual Machines

Azure SQL Database

Azure SQL Data Warehouse

- Cloud-first but not cloud-only
- Use SQL Database to improve core SQL Server features and cadence
- Many interesting and compelling on-premises ←→ cloud scenarios

SQL Server 2017—new features

SQL Server 2016: Feature History

🟦 Enhanced Features

🟨 New Features

FEATURES		SQL Server 2017	SQL Server 2016	SQL Server 2014	SQL Server 2012	SQL Server 2008 R2	SQL Server 2008
Performance	In-Memory OLTP	✓	✓	✓	✗	✗	✗
	In-Memory ColumnStore	✓	✓	✓	✓	✗	✗
	Adaptive Query Processing & Automatic Plan Correction	✓	✗	✗	✗	✗	✗
	Native T-SQL Scoring	✓	✗	✗	✗	✗	✗
	Real-time operational analytics	✓	✓	✗	✗	✗	✗
Availability	Always On Availability Groups	✓	✓	✓	✓	✗	✗
	Cross Platform High Availability	✓	✗	✗	✗	✗	✗
	Windows Server Core Support	✓	✓	✓	✓	✗	✗
Security	Transparent Data Encryption	✓	✓	✓	✓	✓	✓
	Dynamic data masking	✓	✓	✗	✗	✗	✗
	Row-Level Security	✓	✓	✗	✗	✗	✗
	Always Encrypted	✓	✓	✗	✗	✗	✗
Cloud-Readiness	Backup to Microsoft Azure	✓	✓	✓	✗	✗	✗
	Gallery of VM images in Microsoft Azure	✓	✓	✓	✓	✓	✗
	Stretch Database into Azure	✓	✓	✗	✗	✗	✗
Management & Programmability	Policy Based Management	✓	✓	✓	✓	✓	✓
	Distributed Replay	✓	✓	✓	✓	✗	✗
	Graph Data and Queries	✓	✗	✗	✗	✗	✗
	Python	✓	✗	✗	✗	✗	✗
	In-database Advanced Analytics “R”	✓	✓	✗	✗	✗	✗

NEW FEATURES IN SQL SERVER 2017

Support for graph data and queries

Advanced Machine Learning with R & Python

Native T-SQL scoring

Adaptive Query Processing and Automatic Plan Correction

Cross Platform HA (OS level redundancy)

Database Engine new features

Linux/Docker support

- RHEL, Ubuntu, SLES, and Docker

Adaptive query processing

- Faster queries just by upgrading
- Interleaved execution
- Batch-mode memory grant feedback
- Batch-mode adaptive joins

Database Engine new features

Automatic tuning

- Automatic plan correction—identify, and optionally fix, problematic query execution plans causing query performance problems
- Automatic index management—make index recommendations (Azure SQL Database only)

Graph

- Store relationships using nodes/edges
- Analyze interconnected data using node/edge query syntax

```sql
SELECT r.name
FROM Person AS p, likes AS l1, Person AS p2, likes AS l2,
Restaurant AS r
WHERE MATCH(p-(l1)->p2-(l2)->r)
AND p.name = 'Chris'
```

Database Engine new features

Enhanced performance for natively compiled T-SQL modules

- OPENJSON, FOR JSON, JSON
- CROSS APPLY operations
- Computed columns

New string functions

- TRIM, CONCAT_WS, TRANSLATE, and STRING_AGG with support for WITHIN GROUP (ORDER BY)

Bulk import now supports CSV format and Azure Blob storage as file source

Database Engine new features

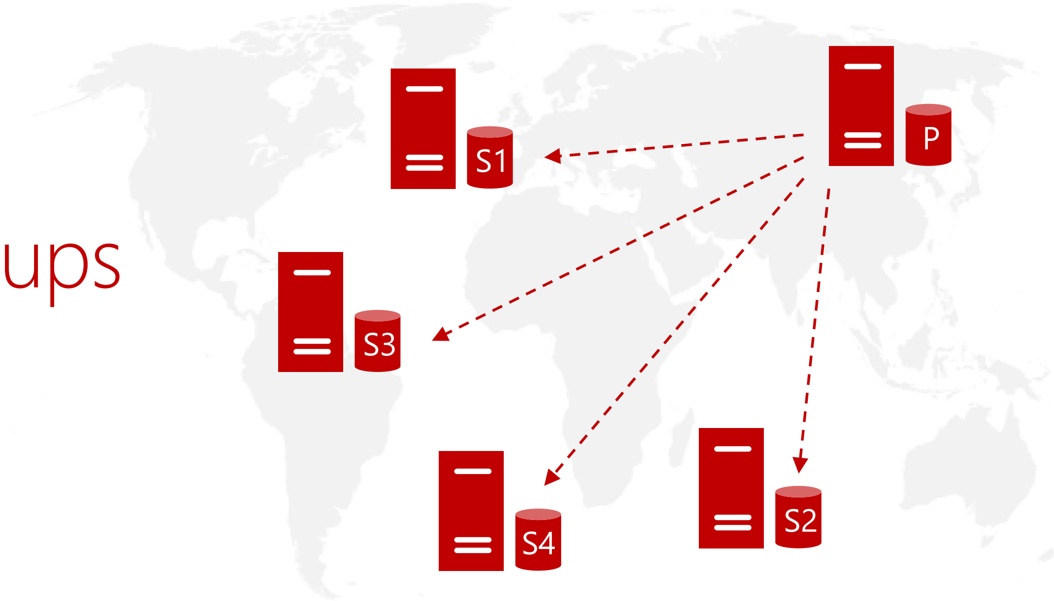
Native scoring with T-SQL PREDICT

Resumable online index rebuild

- Pause/resume online index rebuilds

Clusterless read-scale availability groups

- Unlimited, geo-distributed, linear read scaling



Integration Services new features

Integration Services scale out

- Distribute SSIS package execution more easily across multiple workers, and manage executions and workers from a single master computer

Integration Services on Linux

- Run SSIS packages on Linux computers
- Currently some limitations

Connectivity improvements

- Connect to the OData feeds of Microsoft Dynamics AX Online and Microsoft Dynamics CRM Online with the updated OData components

Analysis Services new features

1400 Compatibility level for tabular models

Object level security for tabular models

Get data enhancements

- New data sources, parity with Power BI Desktop and Excel 2016
- Modern experience for tabular models

Enhanced ragged hierarchy support

- New Hide Members property to hide blank members in ragged hierarchies

Detail Rows

- Custom row set contributing to a measure value
- Drillthrough action in more detail than the aggregated level in tabular models

Reporting Services new features

Comments

- Comments are now available for reports, to add perspective and collaborate with others—you can also include attachments with comments

Broader DAX support

- With Report Builder and SQL Server Data Tools, you create native DAX queries against supported tabular data models by dragging desired fields to the query designers

Standalone installer

- SSRS is no longer distributed through SQL Server setup
- Power BI Report Server

Machine Learning Services new features

Python support

- Python and R scripts are now supported
- Revoscalepy—Pythonic equivalent of RevoScaleR—parallel algorithms for data processing with a rich API

MicrosoftML

- Package of machine learning algorithms and transforms (with Python bindings), as well as pretrained models for image extraction or sentiment analysis

Editions, features, and capacity

SQL Server Editions

SQL Server Edition	Definition
Enterprise	The premium offering, SQL Server Enterprise Edition delivers comprehensive high-end datacenter capabilities with extremely fast performance, unlimited virtualization, and end-to-end business intelligence—enabling high service levels for mission critical workloads and end user access to data insights.
Standard	SQL Server Standard Edition delivers basic data management and a business intelligence database for departments and small organizations to run their applications. It supports common development tools for on-premises and the cloud—enabling effective database management with minimal IT resources.
Web	SQL Server Web Edition is a low total-cost-of-ownership option for web hosters and web VAPs to provide scalability, affordability, and manageability capabilities for small to large scale web properties.
Developer	SQL Server Developer Edition lets developers build any kind of application on top of SQL Server. It includes all the functionality of Enterprise Edition but is licensed for use as a development and test system, not as a production server. SQL Server Developer is an ideal choice for people who build SQL Server and test applications.
Express	Express Edition is the entry-level, free database and is ideal for learning and building desktop and small server data-driven applications. It's the best choice for independent software vendors, developers, and hobbyists who build client applications. If you need more advanced database features, SQL Server Express can be seamlessly upgraded to other higher-end versions of SQL Server. SQL Server Express LocalDB is a lightweight version of Express that has all of its programmability features, yet runs in user mode and has a fast, zero-configuration installation and a short list of prerequisites.

Capacity limits by edition

Feature	Enterprise/Developer	Standard	Web	Express
Maximum compute capacity used by a single instance—SQL Server Database Engine	Operating system maximum	Limited to lesser of four sockets or 24 cores	Limited to lesser of four sockets or 16 cores	Limited to lesser of one socket or four cores
Maximum compute capacity used by a single instance—Analysis Services or Reporting Services	Operating system maximum	Limited to lesser of four sockets or 24 cores	Limited to lesser of four sockets or 16 cores	Limited to lesser of one socket or four cores
Maximum memory for buffer pool per instance of SQL Server Database Engine	Operating system maximum	128 GB	64 GB	1410 MB
Maximum memory for columnstore segment cache per instance of SQL Server Database Engine	Unlimited memory	32 GB	16 GB	352 MB
Maximum memory-optimized data size per database in SQL Server Database Engine	Unlimited memory	32 GB	16 GB	352 MB
Maximum relational database size	524 PB	524 PB	524 PB	10 GB

(Developer Edition has the same limitations and features of Enterprise Edition, but is licensed only for nonproduction workloads.)

SQL Server on Linux

Evolution of SQL Server

Businesses are embracing choice

Heterogeneous environments

Multiple data types

1010
0101
0010

{ }

Different development languages

T-SQL	PHP
Java	Node.js
C/C++	Python
C#/VB.NET	Ruby

On-premises, cloud, and hybrid environments

Microsoft is delivering on choice

HDInsight on Linux

R Server on Linux

Linux in Azure

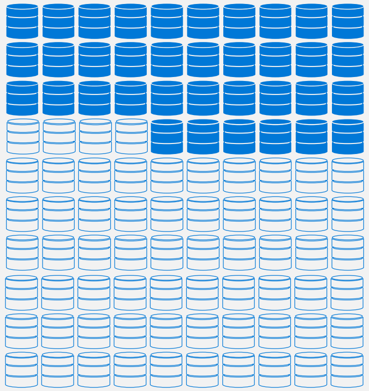
SQL Server drivers and connectivity

Visual Studio Code extension for SQL Server

The world is demanding SQL Server on Linux

20K+

applications for private preview

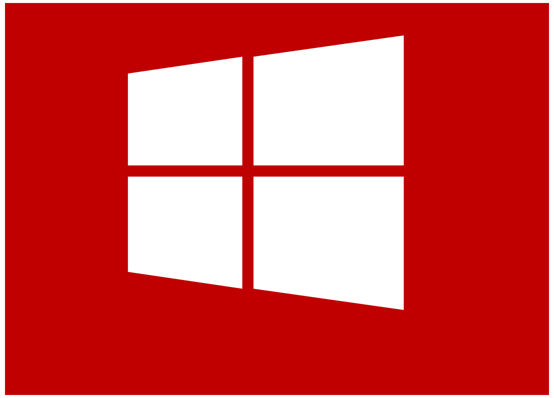


36%

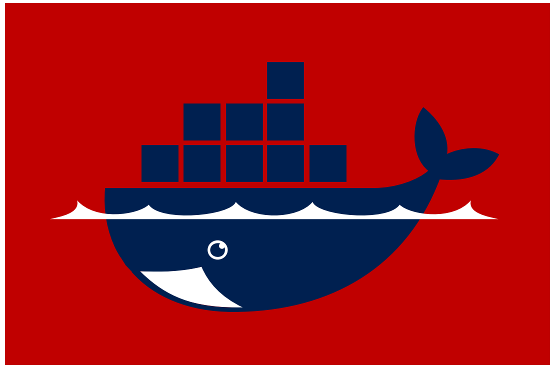
enterprise DB market runs on Linux

Power of the SQL Server Database Engine on the platform of your choice

Windows



Linux/Windows container



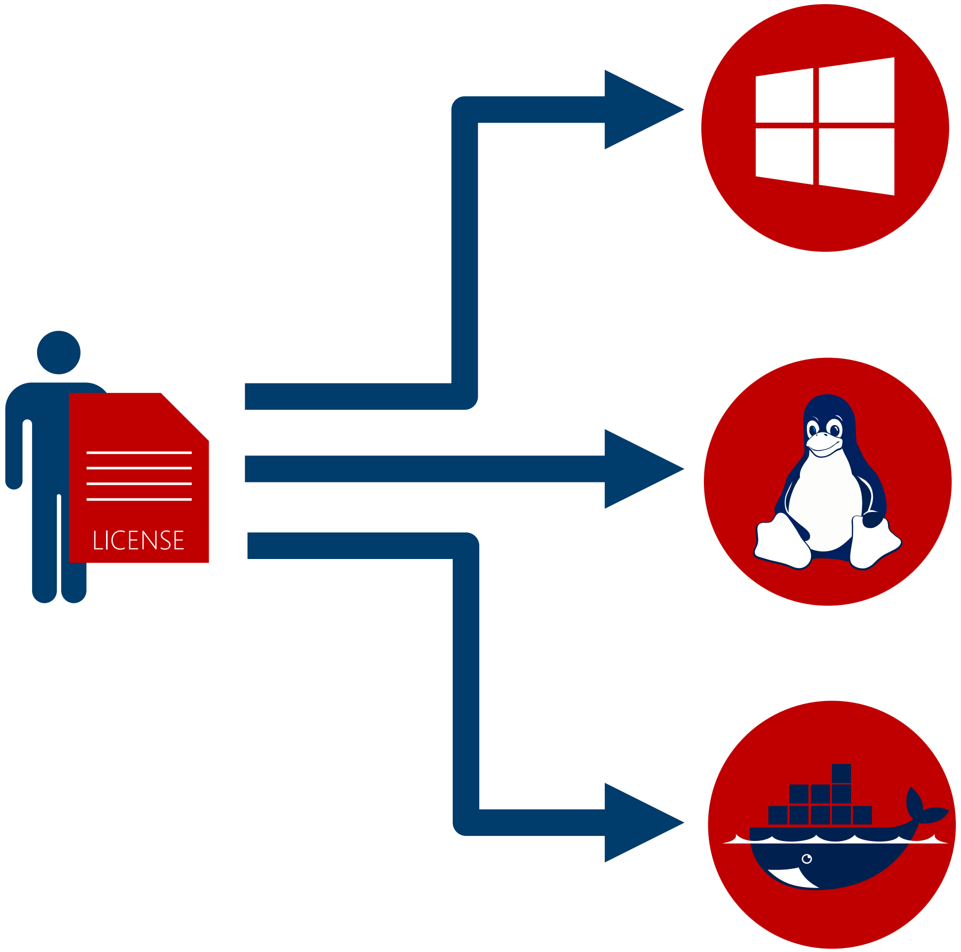
Linux

Linux distributions: RedHat Enterprise Linux (RHEL), Ubuntu, and SUSE Linux Enterprise Server (SLES)

Docker: Windows and Linux containers

Windows Server/Windows 10

Same license, new choice



Buying a SQL Server license gives you the option to use it on Windows Server, Linux, or Docker.

Regardless of where you run it—VM, Docker, physical, cloud, on-premises—the licensing model is the same; available features depend on which edition of SQL Server you use.

Linux-native user experience

Standard installation process

- Package-based installation using yum for Fedora-based distributions, apt-get for Debian-based distributions, and zypper for SLES
- Existing package update/upgrade processes for SQL upgrade

Familiar experience

- SQL Server service runs natively using systemd
- Linux file paths are supported in T-SQL statements and scripts (defining/changing the path, database backup files)
- Popular Linux high-availability solutions like Pacemaker and Corosync

Cross-platform tools

- SQL Server command-line tools (sqlcmd, bcp) available for Linux (and soon on macOS)
- Existing Windows tools such as SQL Server Management Studio (SSMS), SQL Server Data Tools (SSDT), and PowerShell module (sqlps) to manage SQL Server on Linux from Windows
- Visual Studio Code extension for SQL Server on macOS, Linux, or Windows

Supported platforms

Platform	Supported version(s)	Supported file system(s)
Red Hat Enterprise Linux	7.3	XFS or EXT4
SUSE Linux Enterprise Server	v12 SP2	EXT4
Ubuntu	16.04	EXT4
Docker Engine (on Windows, Mac, or Linux)	1.8+	N/A

For full system requirements, see System requirements for SQL Server on Linux

Installing SQL Server on Linux

Add the SQL Server repository to your package manager

```
sudo curl -o /etc/yum.repos.d/mssql-server.repo https://packages.microsoft.com/config/rhel/7/mssql-server-2017.repo
sudo yum update
```

Install the **mssql-server** package

```
sudo yum install -y mssql-server
```

Run **mssql-conf setup** to configure SA password and edition

```
sudo /opt/mssql/bin/mssql-conf setup
```

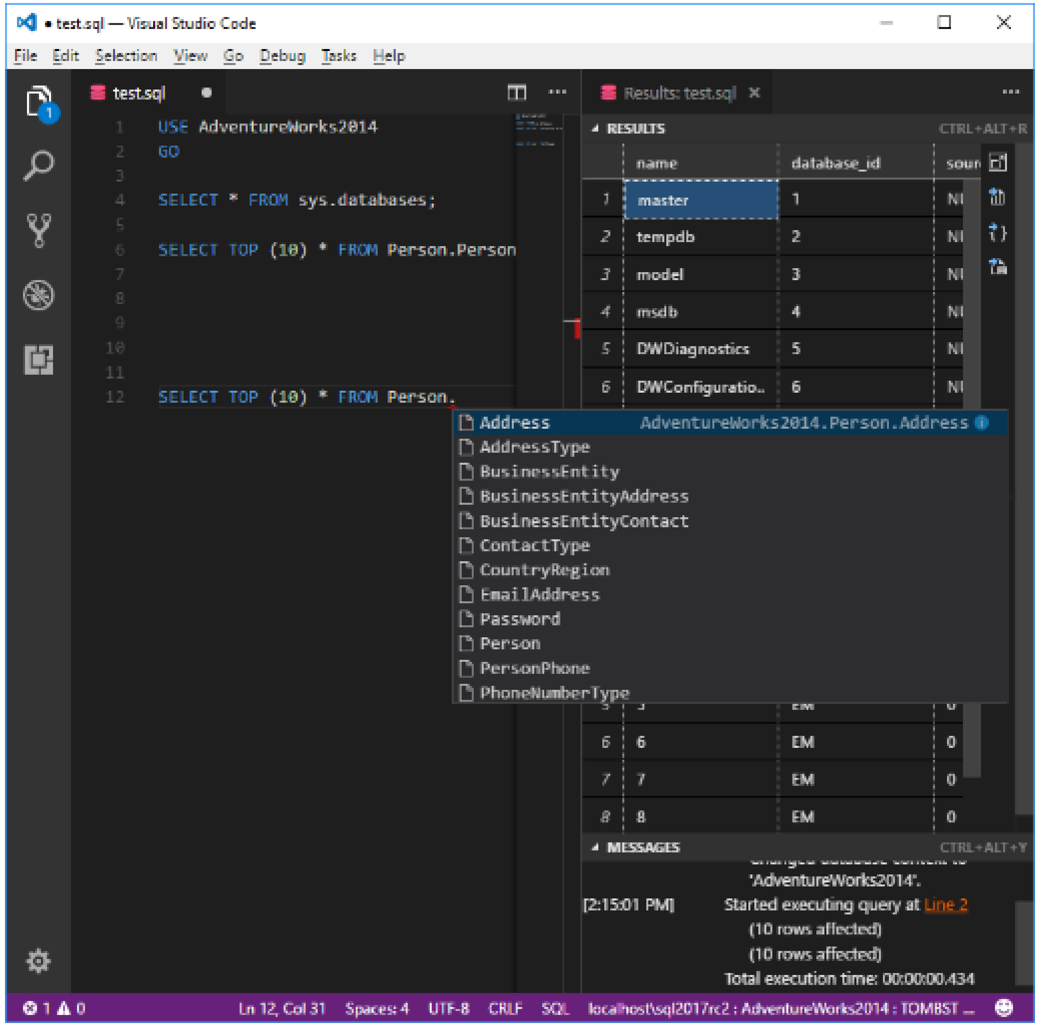
Configure the firewall to allow remote connections (optional)

```
sudo firewall-cmd --zone=public --add-port=1433/tcp --permanent
sudo firewall-cmd --reload
```

Follow the links from the SQL Server on Linux overview page to find detailed installation instructions for your platform.

Tools and programmability

- Windows-based SQL Server tools—like SSMS, SSDT, and Profiler—work when connected to SQL Server on Linux
- All existing drivers and frameworks supported
- Third-party tools continue to work
- Native command-line tools—sqlcmd, bcp
- Visual Studio Code mssql extension


● test.sql — Visual Studio Code
File Edit Selection View Go Debug Tasks Help
test.sql
Results: test.sql
RESULTS CTRL+ALT+R
Address AdventureWorks2014.Person.Address
AddressType
BusinessEntity
BusinessEntityAddress
BusinessEntityContact
ContactType
CountryRegion
EmailAddress
Password
Person
PersonPhone
PhoneNumberType
MESSAGES CTRL+ALT+Y
⊗ 1 ⚠ 0 Ln 12, Col 31 Spaces: 4 UTF-8 CRLF SQL localhost\sql2017rc2 : AdventureWorks2014 : TOMBST ...

```sql
USE AdventureWorks2014
GO

SELECT * FROM sys.databases;

SELECT TOP (10) * FROM Person.Person




SELECT TOP (10) * FROM Person.
```

	name	database_id	soun
1	master	1	NI
2	tempdb	2	NI
3	model	3	NI
4	msdb	4	NI
5	DWDiagnostics	5	NI
6	DWConfiguratio..	6	NI

6	6	EM	0
7	7	EM	0
8	8	EM	0

```
                 'AdventureWorks2014'.
[2:15:01 PM]    Started executing query at Line 2
                 (10 rows affected)
                 (10 rows affected)
                Total execution time: 00:00:00.434
```

What's available on Linux?

Operations features

- Support for RHEL, Ubuntu, SLES, Docker
- Package-based installs
- Support for Open Shift, Docker Swarm
- Failover clustering via Pacemaker
- Backup/Restore
- SSMS on Windows connected to Linux
- Command-line tools: sqlcmd, bcp
- Transparent Data Encryption
- Backup Encryption
- SCOM management pack
- DMVs
- Table partitioning
- SQL Server Agent
- Full-Text Search
- Integration Services
- Active Directory (integrated) authentication
- TLS for encrypted connections

What’s available on Linux?

Programming features

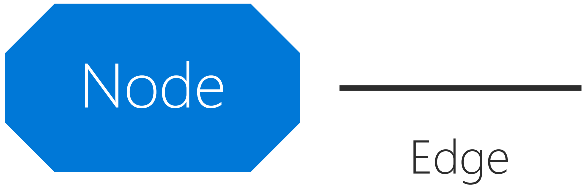
- All major language driver compatibility
- In-Memory OLTP
- Columnstore indexes
- Query Store
- Compression
- Always Encrypted
- Row-Level Security, Data Masking
- Auditing
- Service Broker
- CLR
- JSON, XML
- Third-party tools

Features not currently supported on Linux

Database Engine	• Transactional replication • Merge replication • Stretch DB • PolyBase • Distributed query with third-party connections • System extended stored procedures (XP_CMDSHELL, etc.) • Filetable • CLR assemblies with the EXTERNAL_ACCESS or UNSAFE permission set • Buffer Pool Extension
High Availability	• Database Mirroring
Security	• Extensible Key Management
SQL Server Agent	• Subsystems: CmdExec, PowerShell, Queue Reader, SSIS, SSAS, SSRS • Alerts • Log Reader Agent • Change Data Capture • Managed Backup
Services	• SQL Server Browser • SQL Server R Services • StreamInsight • Analysis Services • Reporting Services • Data Quality Services • Master Data Services

Graph processing

What is a graph?

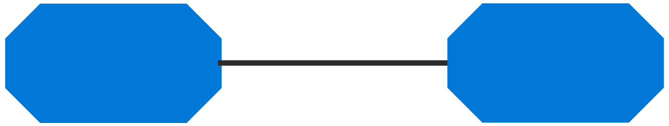


Node

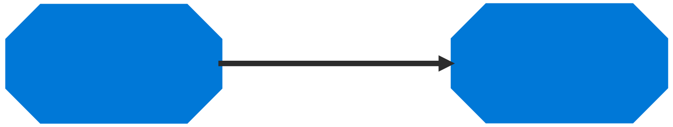
Edge

A graph is a collection of nodes and edges

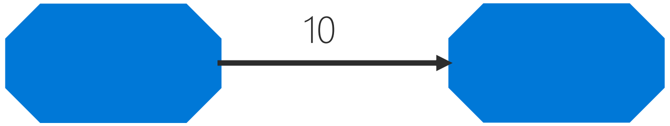
Undirected graph



Directed graph

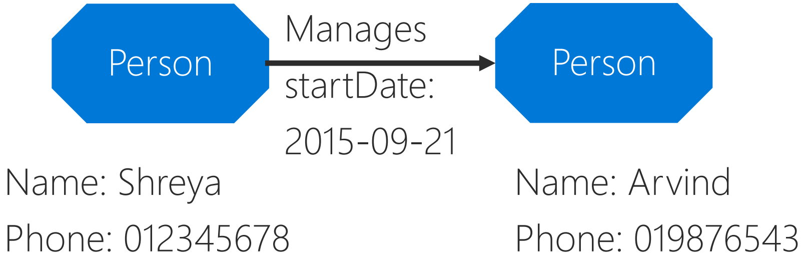


Weighted graph



10

Property graph



Person

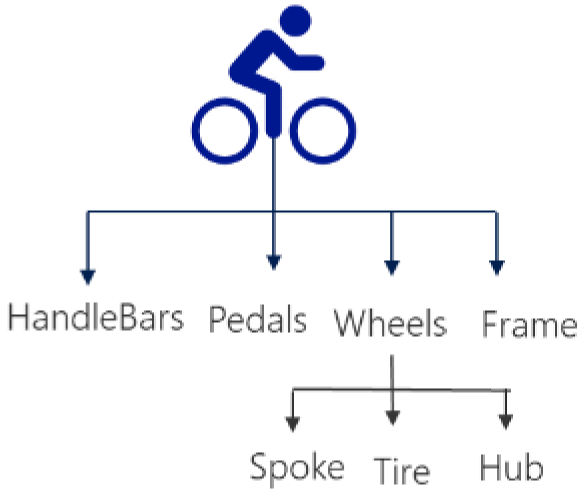
Manages
startDate:
2015-09-21

Person

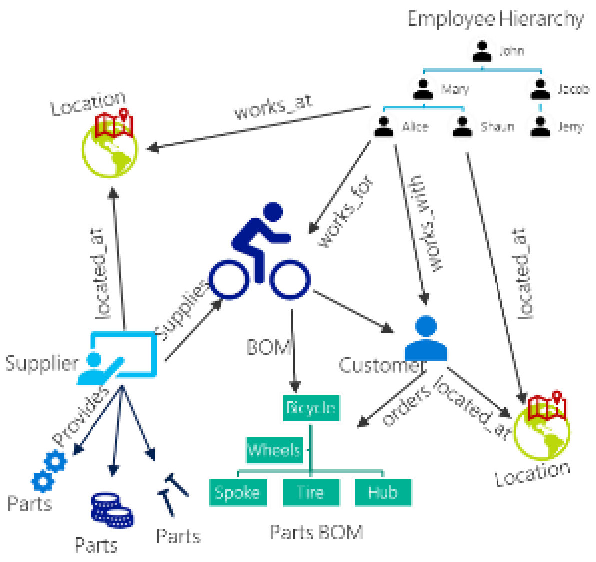
Name: Shreya
Phone: 012345678

Name: Arvind
Phone: 019876543

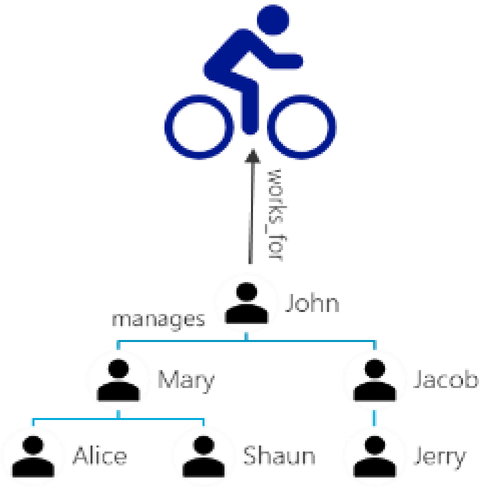
Typical scenarios for graph databases



Hierarchical or interconnected data, entities with multiple parents.



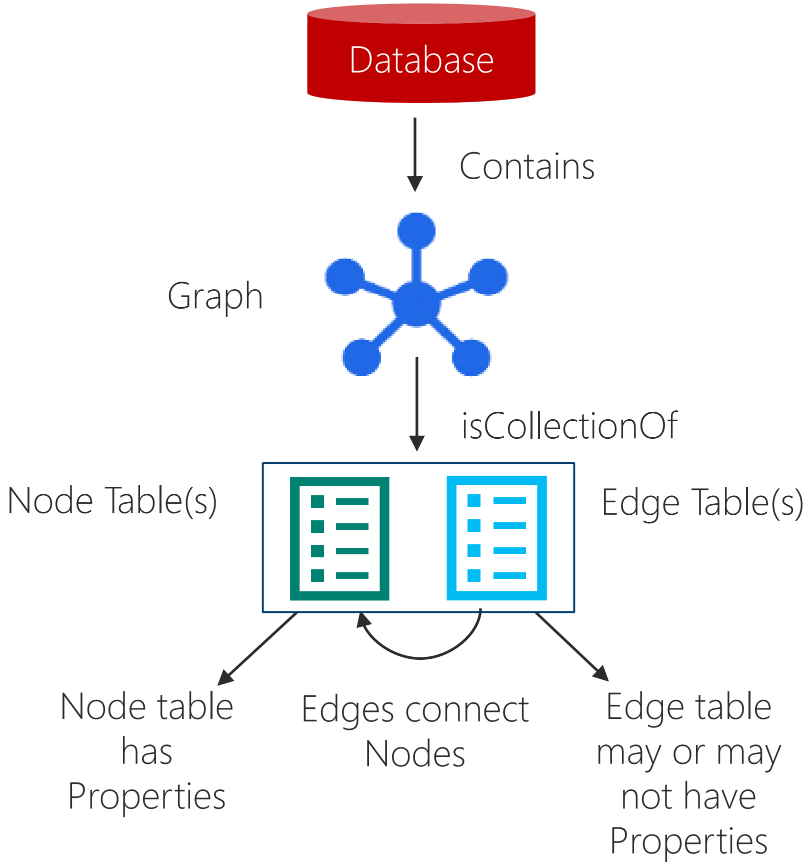
Complex many-to-many relationships. Organically grow connections as the business evolves.



Analyze interconnected data, materialize new information from existing facts. Identify connections that are not obvious.

Introducing SQL Server Graph

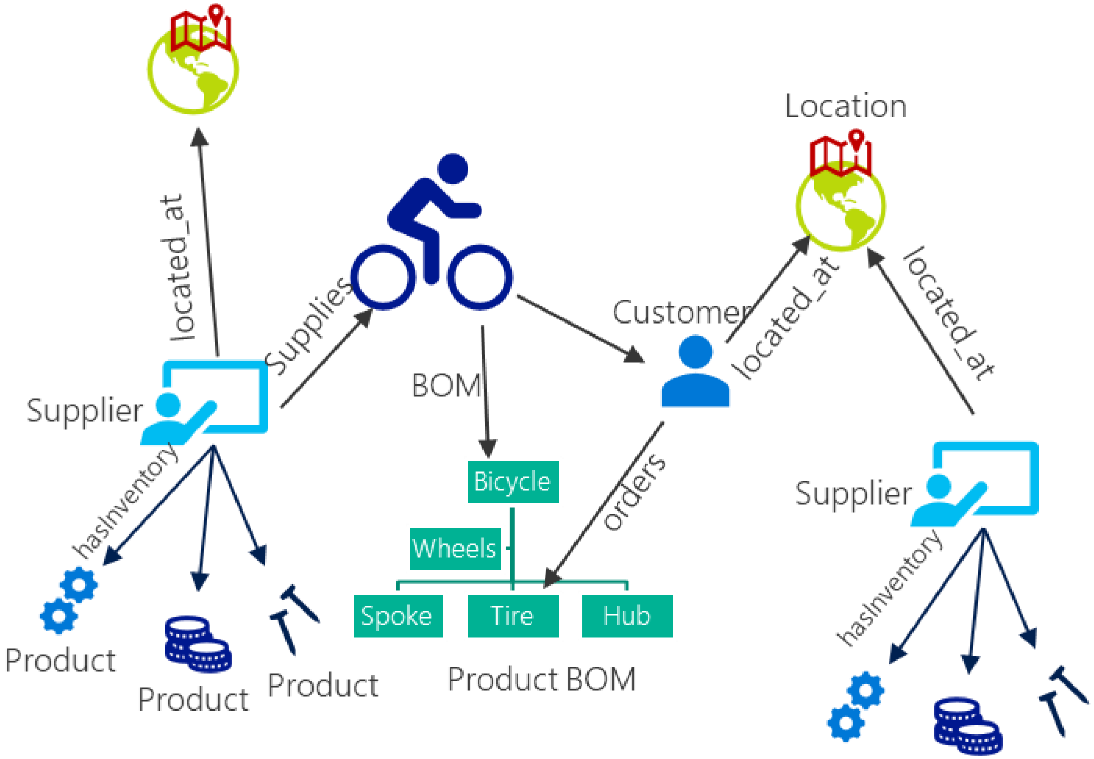
- A collection of node and edge tables in the database
- Language Extensions
  - **DDL Extensions**—create node and edge tables
  - **DML Extensions**—SELECT - T-SQL MATCH clause to support pattern matching and traversals; DELETE, UPDATE, and INSERT support graph tables
- Graph support is integrated into the SQL Server ecosystem



DDL Extensions

- Create node and edge tables
- Properties associated with nodes and edges

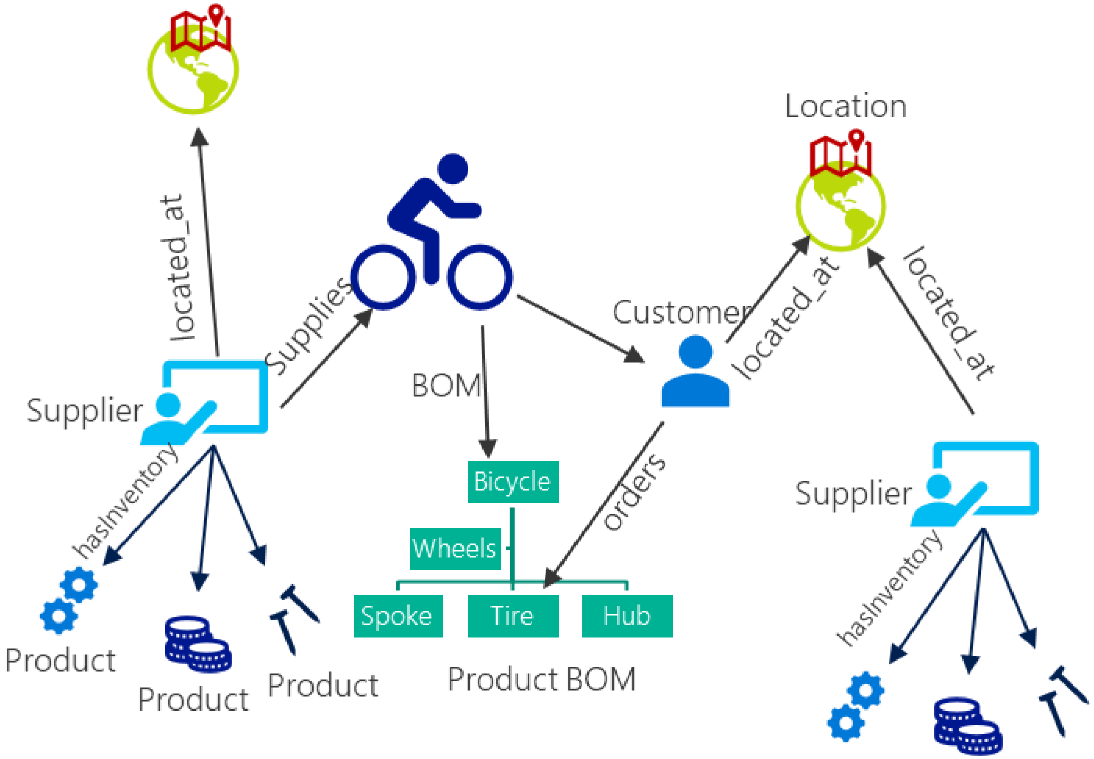
```
CREATE TABLE Product (ID INTEGER PRIMARY KEY,
name VARCHAR(100)) AS NODE;

CREATE TABLE Supplier (ID INTEGER PRIMARY KEY,
name VARCHAR(100)) AS NODE;

CREATE TABLE hasInventory AS EDGE;

CREATE  TABLE  located_at(address  varchar(100))
AS EDGE;
```



DML Extensions

Multihop navigation and join-free pattern matching using the **MATCH** predicate:

```
SELECT Prod.name as ProductName,
   Sup.name as SupplierName
FROM Product Prod, Supplier Sup,
  hasInventory hasIn,
  located_at supp_loc,
  Customer Cus,
  located_at cust_loc,
  orders, location loc
WHERE
MATCH(
  cus-(orders)->Prod<-(hasIn)-Sup
  AND
  cus-(cust_loc)->location<-(supp_loc)-Sup
) ;
```



Spatial

Spatial

Spatial data represents information about the physical location and shape of geometric objects. These objects can be point locations, or lines, or more complex objects such as countries, roads, or lakes.

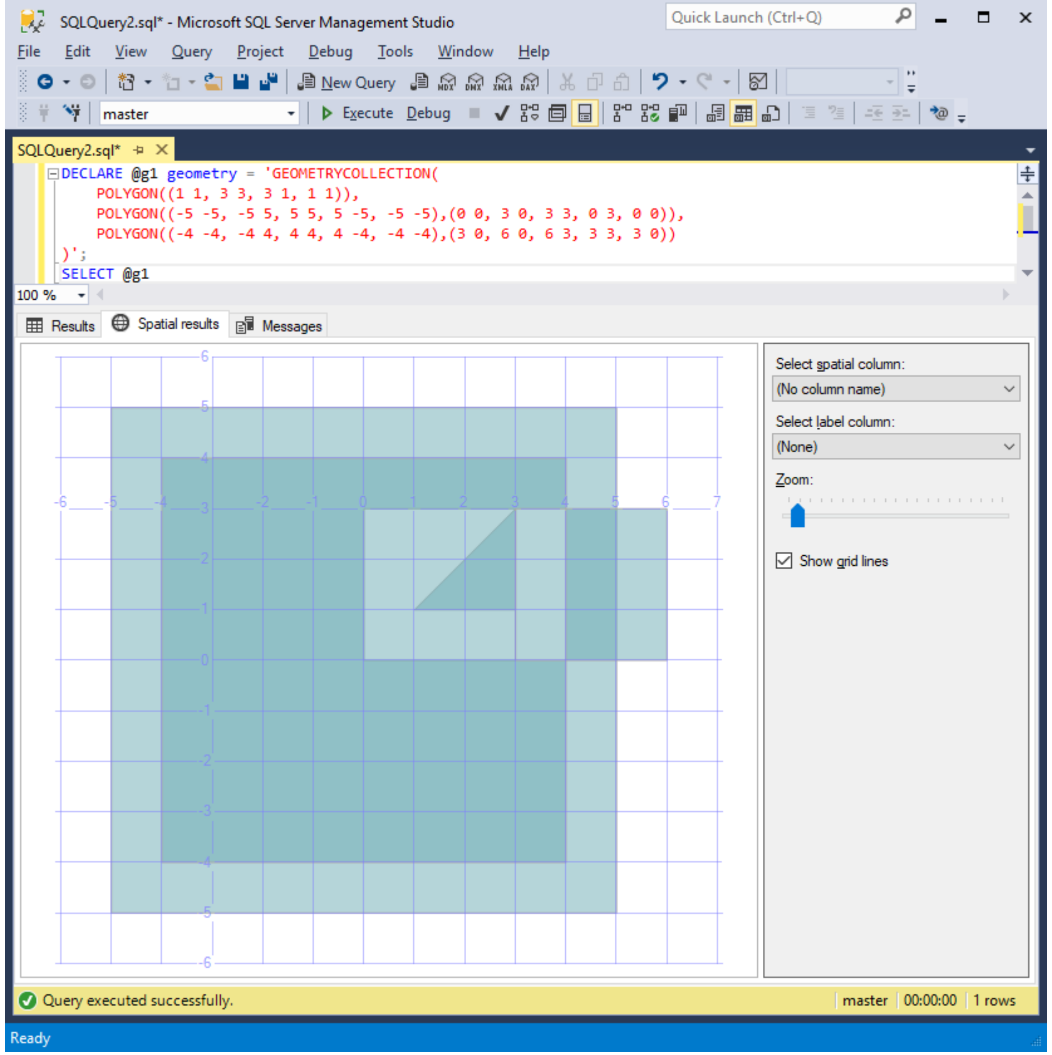
SQL Server supports two spatial data types: the **geometry** data type and the **geography** data type.

- The **geometry** type represents data in a Euclidean (flat) coordinate system.
- The **geography** type represents data in a round-earth coordinate system.

Spatial functionality

- Simple and compound spatial data types supported
- Import and export spatial data to industry-standard formats (Open Geospatial Consortium WKT and WKB)
- Functions to query the properties of, the behaviours of, and the relationships between, spatial data instances
- Spatial columns can be indexed to improve query performance

Spatial tooling



```
DECLARE @g1 geometry = 'GEOMETRYCOLLECTION(
    POLYGON((1 1, 3 3, 3 1, 1 1)),
    POLYGON((-5 -5, -5 5, 5 5, 5 -5, -5 -5),(0 0, 3 0, 3 3, 0 3, 0 0)),
    POLYGON((-4 -4, -4 4, 4 4, 4 -4, -4 -4),(3 0, 6 0, 6 3, 3 3, 3 0))
)';
SELECT @g1
```

Select spatial column: (No column name)

Select label column: (None)

Zoom:

☑ Show grid lines

Query executed successfully. | master | 00:00:00 | 1 rows

Ready

- SSMS includes the ability to display a visual representation of spatial results

Spatial enhancements (SQL Server 2017)

- The **FullGlobe** geometry data type—FullGlobe is a special type of polygon that covers the entire globe. FullGlobe has an area, but no borders or vertices.

Transact-SQL extensions

T-SQL **TRIM**

Removes the space character (char(32)) or other specified characters from the start or end of a string.

```
TRIM ( [ characters FROM ] string )
```

Removing the space character from both sides of a string

(equivalent to LTRIM(RTRIM(string)))

```
SELECT TRIM ('   test    ') AS Result ;

Result
-----------
test
```

Removes specified characters from both sides of a string

(Trimming multiple characters)

```
SELECT TRIM( '.,! ' FROM  '#     test    .') AS Result;

Result
---------------
#     test
```

T-SQL CONCAT_WS

Concatenates a variable number of arguments with a delimiter specified in the first argument.

```
CONCAT_WS ( separator, argument1, argument2 [, argumentN]…)
```

Concatenating with a delimiter

```
SELECT CONCAT_WS( ' - ','one','two','three','four') AS Result ;

Result
------------------------
one - two - three - four
```

Concatenation ignores NULL

```
SELECT CONCAT_WS( ' - ','one',NULL,'two',NULL,'three',NULL,'four') AS Result ;

Result
---------------------------------
one - two - three - four
```

T-SQL TRANSLATE

Returns the string provided as a first argument after some characters specified in the second argument are translated into a destination set of characters.

```
TRANSLATE ( inputString, characters, translations)
```

Replace square and curly braces with regular braces

```
SELECT TRANSLATE('2*[3+4]/{7-2}', '[]{}', '()()') AS Result ;

Result
-------------
2*(3+4)/(7-2)
```

Convert GeoJSON points into WKT

```
SELECT TRANSLATE('[137.4, 72.3]' , '[,]', '( )') AS Point, TRANSLATE('(137.4 72.3)' , '( )', '[,]') AS Coordinates ;

Point         Coordinates
------------- -------------
(137.4  72.3) [137.4,72.3]
```

T-SQL STRING_AGG

Concatenates the values of string expressions and places separator values between them. The separator is not added at the end of string.

```
STRING_AGG ( expression, separator ) [ <order_clause> ]

<order_clause> ::=
    WITHIN GROUP ( ORDER BY <order_by_expression_list> [ ASC | DESC ] )
```

Generate a list of names separated with a comma (without NULL values)

```
SELECT STRING_AGG ( ISNULL(FirstName,'N/A'), ',') AS csv FROM Person.Person ;

csv
-----------------------------------------------------------------------------------------------------------------------------
Syed,Catherine,Kim,Kim,Kim,Hazem,Sam,Humberto,Gustavo,Pilar,Pilar,Aaron,Adam,Alex,Alexandra,Allison,Amanda,Amber,Andrea,Angel
```

Generate a sorted list of emails per town

```
SELECT town, STRING_AGG (email, ';') WITHIN GROUP (ORDER BY email ASC) AS emails FROM dbo.Employee GROUP BY town ;

town    emails
------- ---------------------------------------------------------------------------------
Seattle catherine0@adventure-works.com;kim2@adventure-works.com;syed0@adventure-works.com
LA      hazem0@adventure-works.com;sam1@adventure-works.com
```

T-SQL BULK INSERT / OPENROWSET(BULK…)

Additional options added that provide support for CSV format data files.

```
[ [ , ] FORMAT = 'CSV' ]
[ [ , ] FIELDQUOTE = 'quote_characters']
```

`FORMAT = 'CSV'`

Specifies a comma separated value file compliant to the RFC 4180 standard.

`FIELDQUOTE = 'field_quote'`

Specifies a character that will be used as the quote character in the CSV file. If not specified, the quote character (") will be used as the quote character as defined in the RFC 4180 standard.

Data files and format files can now be loaded from Azure Blob storage.

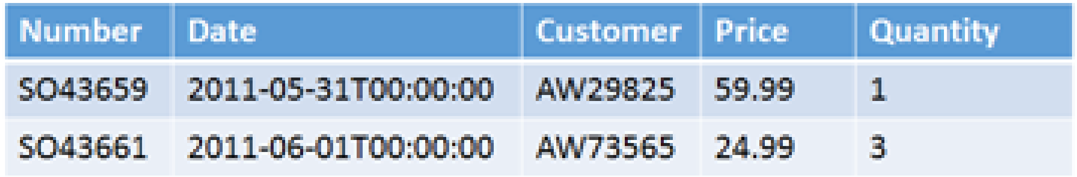
JSON

JSON support

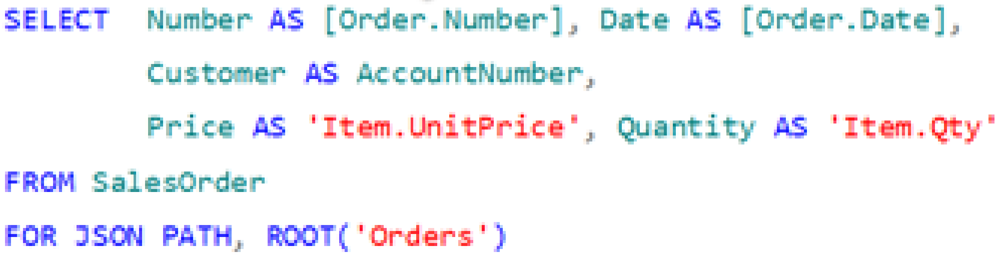
- Not a built-in data type—JSON is stored as varchar or nvarchar
- Format SQL data or query results as JSON
- Convert JSON to SQL data
- Query JSON data
- Index JSON data

FOR JSON

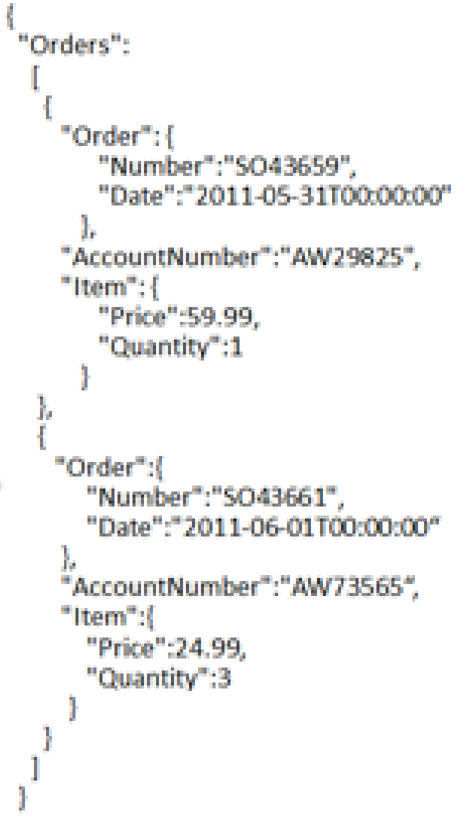
In **PATH** mode, you use the dot syntax—for example, 'Item.Price'—to format nested output. This example also uses the **ROOT** option to specify a named root element.



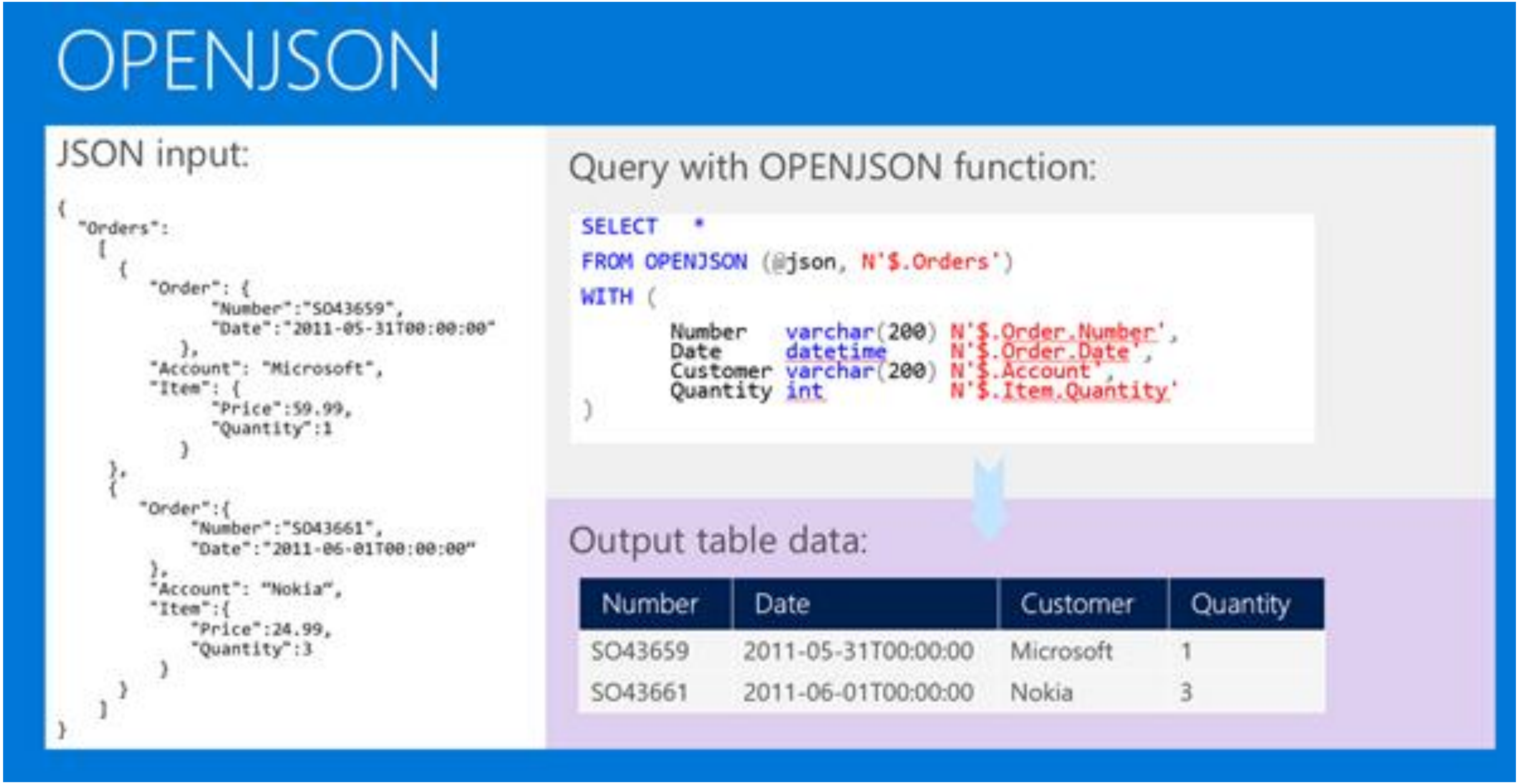
Number	Date	Customer	Price	Quantity
SO43659	2011-05-31T00:00:00	AW29825	59.99	1
SO43661	2011-06-01T00:00:00	AW73565	24.99	3



```
SELECT  Number AS [Order.Number], Date AS [Order.Date],
        Customer AS AccountNumber,
        Price AS 'Item.UnitPrice', Quantity AS 'Item.Qty'
FROM SalesOrder
FOR JSON PATH, ROOT('Orders')
```



```
{
 "Orders":
  [
   {
    "Order":{
       "Number":"SO43659",
       "Date":"2011-05-31T00:00:00"
     },
    "AccountNumber":"AW29825",
    "Item":{
       "Price":59.99,
       "Quantity":1
     }
   },
   {
    "Order":{
      "Number":"SO43661",
      "Date":"2011-06-01T00:00:00"
    },
    "AccountNumber":"AW73565",
    "Item":{
      "Price":24.99,
      "Quantity":3
    }
   }
  ]
}
```



OPENJSON

JSON input:

```
{
  "Orders":
    [
      {
          "Order": {
                "Number":"SO43659",
                "Date":"2011-05-31T00:00:00"
            },
          "Account": "Microsoft",
          "Item": {
                "Price":59.99,
                "Quantity":1
            }
      },
      {
         "Order":{
              "Number":"SO43661",
              "Date":"2011-06-01T00:00:00"
          },
          "Account": "Nokia",
          "Item":{
              "Price":24.99,
              "Quantity":3
           }
       }
    ]
}
```

Query with OPENJSON function:

```
SELECT   *
FROM OPENJSON (@json, N'$.Orders')
WITH (
        Number   varchar(200) N'$.Order.Number',
        Date     datetime     N'$.Order.Date',
        Customer varchar(200) N'$.Account',
        Quantity int          N'$.Item.Quantity'
)
```

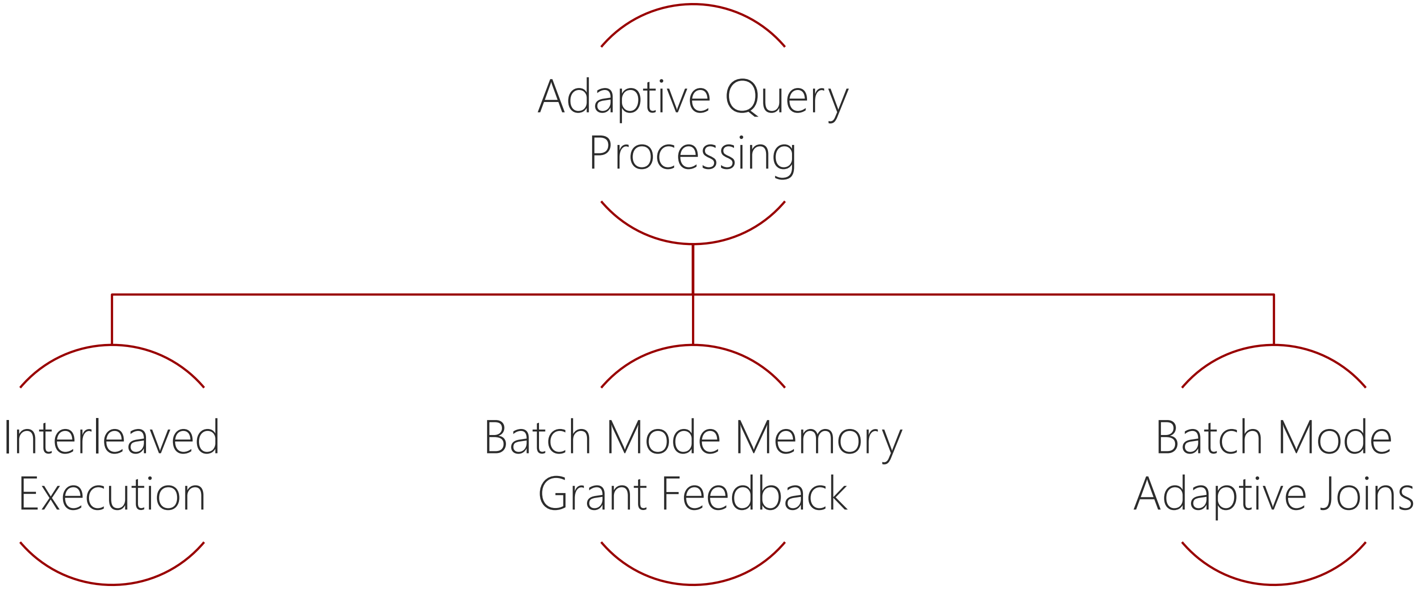
Output table data:

Number	Date	Customer	Quantity
SO43659	2011-05-31T00:00:00	Microsoft	1
SO43661	2011-06-01T00:00:00	Nokia	3

Adaptive query processing

Adaptive query processing

Three features to improve query performance



Adaptive Query Processing

Interleaved Execution

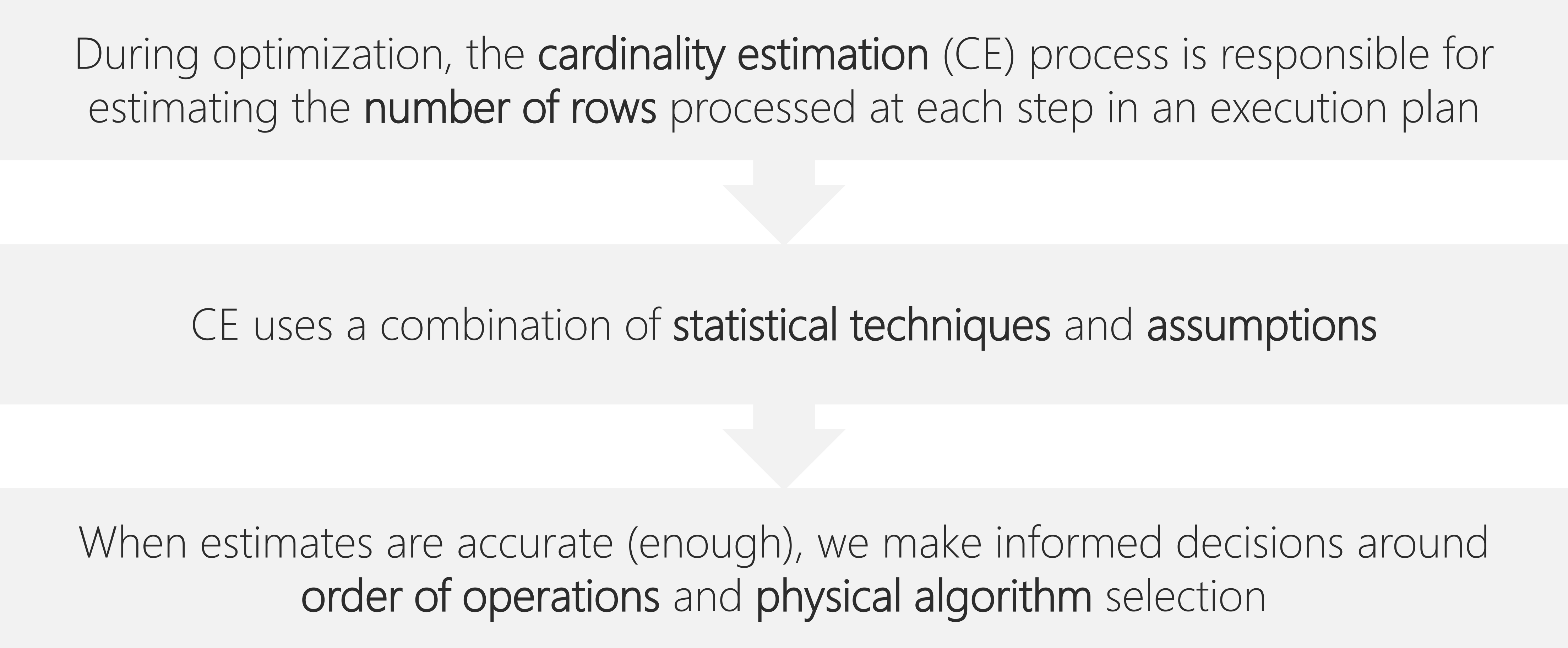
Batch Mode Memory Grant Feedback

Batch Mode Adaptive Joins

Enabled when the database is in SQL Server 2017 compatibility mode (140)

```
ALTER DATABASE current SET COMPATIBILITY_LEVEL = 140;
```

Query processing and cardinality estimation



During optimization, the **cardinality estimation** (CE) process is responsible for estimating the **number of rows** processed at each step in an execution plan

CE uses a combination of **statistical techniques** and **assumptions**

When estimates are accurate (enough), we make informed decisions around **order of operations** and **physical algorithm** selection

Common reasons for incorrect cardinality estimates

Missing statistics

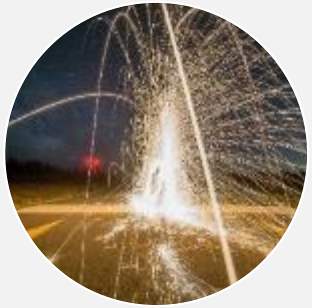
Stale statistics

Inadequate statistics sample rate

Bad parameter sniffing scenarios

Out-of-model query constructs

- For example, MSTVFs, table variables, XQuery



Assumptions not aligned with data being queried

- For example, independence versus correlation

Cost of incorrect estimates

Slow query response time due to inefficient plans

Excessive resource utilization (CPU, Memory, IO)

Spills to disk

Reduced throughput and concurrency

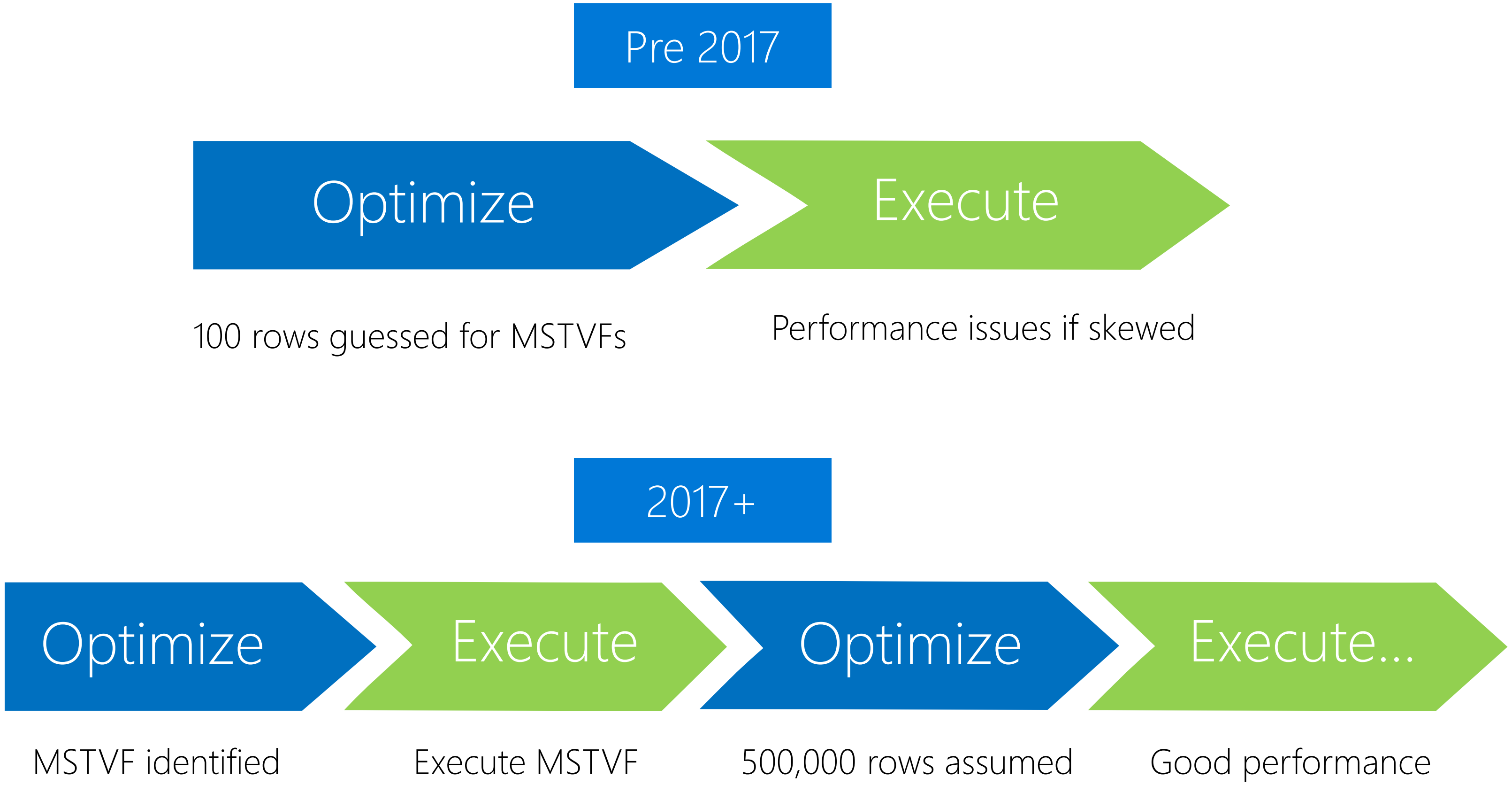
T-SQL refactoring to work around off-model statements

Interleaved execution

Problem: Multi-statement table valued functions (MSTVFs) are treated as a black box by QP and we use a fixed optimization guess.

Interleaved execution will materialize row counts for MSTVFs.

Downstream operations will benefit from the corrected MSTVF cardinality estimate.



Pre 2017

Optimize

Execute

100 rows guessed for MSTVFs

Performance issues if skewed

2017+

Optimize

Execute

Optimize

Execute…

MSTVF identified

Execute MSTVF

500,000 rows assumed

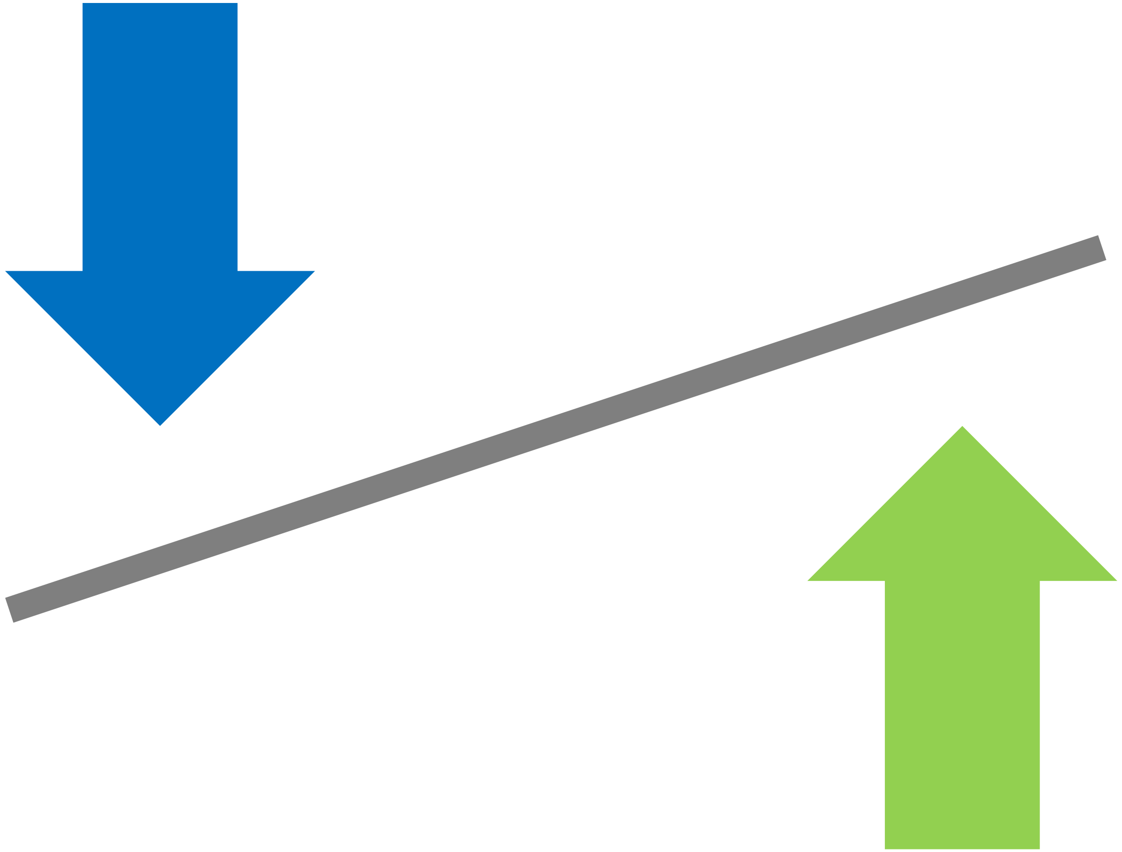
Good performance

Batch mode memory grant feedback

Problem: Queries can spill to disk or take too much memory, based on poor cardinality estimates.

Memory grant feedback (MGF) will adjust memory grants based on execution feedback.

MGF will remove spills and improve concurrency for repeating queries.

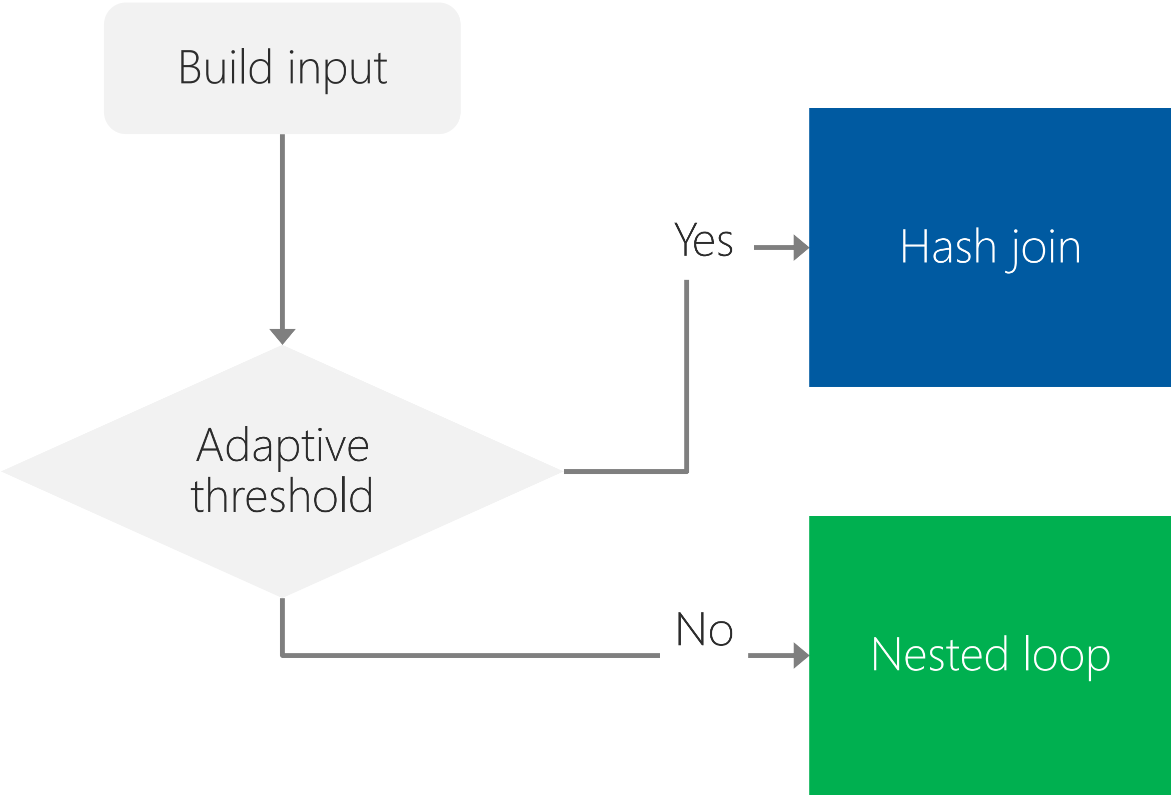


Batch mode adaptive joins

Problem: If cardinality estimates are skewed, we might choose an inappropriate join algorithm.

Batch mode adaptive joins (AJ) will defer the choice of hash join or nested loop until after the first join input has been scanned.

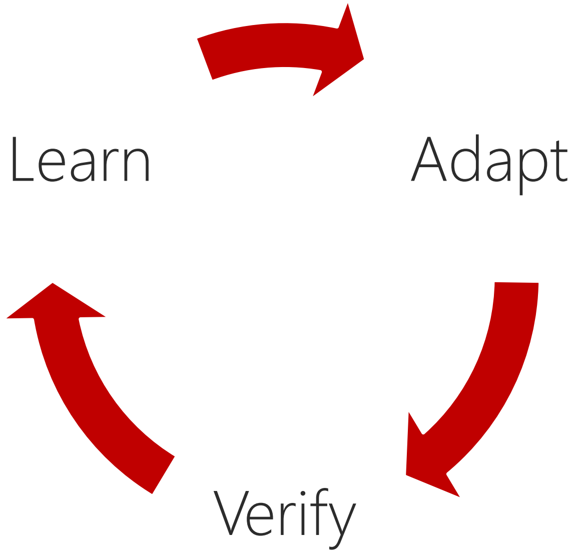
AJ uses nested loop for small inputs, and hash joins for large inputs.



Automatic tuning

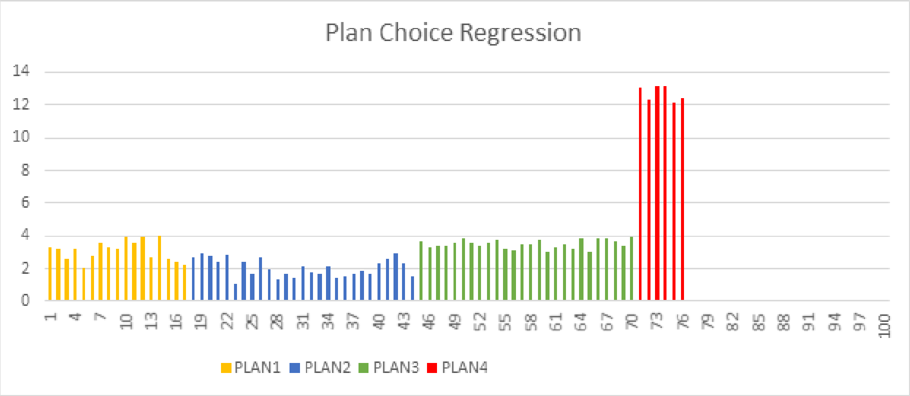
Automatic tuning

Automatic plan correction identifies problematic plans and fixes SQL plan performance problems:



Automatic plan choice detection

- The Database Engine continuously collects query plan performance information
- Potential plan choice regression is identified when a change of query plan for a query corresponds to a drop in query performance

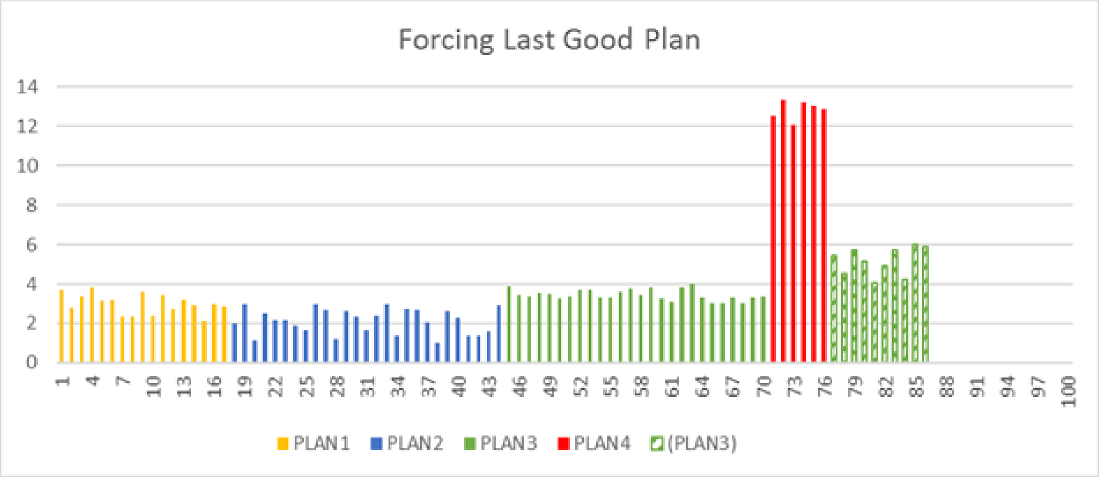


- Regressions (together with suggested mitigations) are reported in the DMV sys.dm_db_tuning_recommendations

Automatic plan correction

Automatically apply the mitigation identified in sys.dm_db_tuning_recommendations by enabling the AUTOMATIC_TUNING database property:

```
ALTER DATABASE current
SET AUTOMATIC_TUNING ( FORCE_LAST_GOOD_PLAN = ON );
```



In-Memory OLTP & ColumnStore

In-Memory Online Transaction Processing (OLTP)

In-Memory OLTP is the premier technology available in SQL Server and Azure SQL Database for optimizing performance of transaction processing, data ingestion, data load, and transient data scenarios.

Memory-optimized tables outperform traditional disk-based tables, leading to more responsive transactional applications.

Memory-optimized tables also improve throughput and reduce latency for transaction processing, and can help improve performance of transient data scenarios such as temp tables and ETL.

Steps for In-Memory OLTP

SQL Server provides In-Memory OLTP features that can greatly improve the performance of application systems.

Recommended to set the database to the latest compatibility level, particularly for In-Memory OLTP:

```
ALTER DATABASE CURRENT
SET COMPATIBILITY_LEVEL = 140;
GO
```

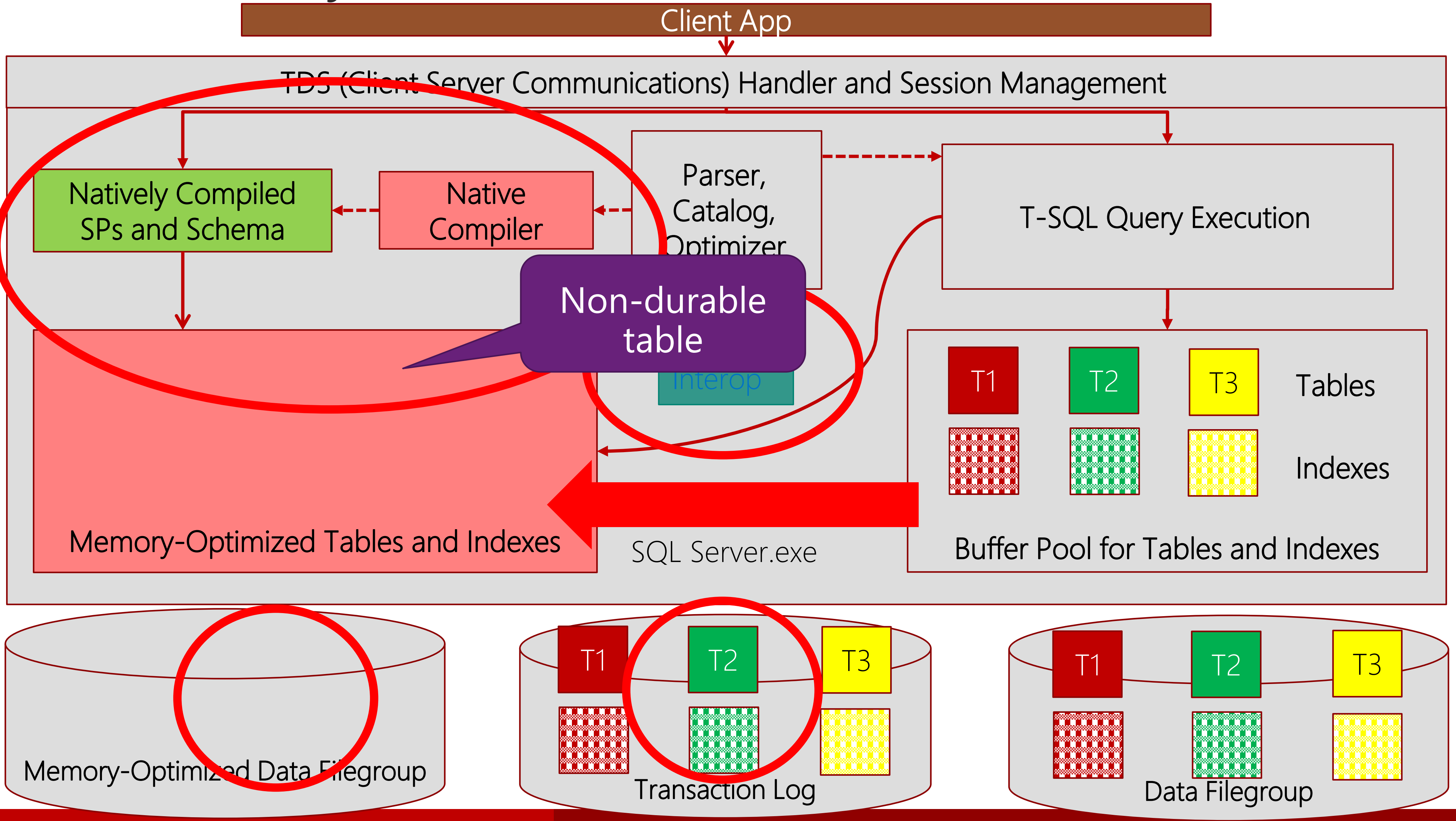
When a transaction involves both a disk-based table and a memory-optimized table, it's essential that the memory-optimized portion of the transaction operates at the transaction isolation level named SNAPSHOT.

```
ALTER DATABASE CURRENT SET MEMORY_OPTIMIZED_ELEVATE_TO_SNAPSHOT=ON
GO
```

Before you can create a memory-optimized table, you must first create a memory-optimized FILEGROUP and a container for data files:

```
ALTER DATABASE AdventureWorks ADD FILEGROUP AdventureWorks_mod CONTAINS memory_optimized_data
GO
ALTER DATABASE AdventureWorks ADD FILE (NAME='AdventureWorks_mod', FILENAME='c:\var\opt\mssql\data\AdventureWorks_mod') TO
FILEGROUP AdventureWorks_mod
GO
```

In-Memory OLTP architecture


Client App
TDS (Client Server Communications) Handler and Session Management
Natively Compiled SPs and Schema
Native Compiler
Parser, Catalog, Optimizer
T-SQL Query Execution
Non-durable table
Interop
T1
T2
T3
Tables
Indexes
Memory-Optimized Tables and Indexes
SQL Server.exe
Buffer Pool for Tables and Indexes
Memory-Optimized Data Filegroup
T1
T2
T3
Transaction Log
T1
T2
T3
Data Filegroup

Key

Existing SQL Component

In-Memory OLTP Component

Generated .dll

Memory-optimized tables

In short, memory-optimized tables are stored in main memory as opposed to on disk.

Memory-optimized tables are fully durable by default; data is persisted to disk in the background.

Memory-optimized tables can be accessed with T-SQL, but are accessed more efficiently with natively compiled stored procedures.

Memory-optimized tables

The primary store for memory-optimized tables is main memory; unlike disk-based tables, data does not need to be read in to memory buffers from disk.

To create a memory-optimized table, use the MEMORY_OPTIMIZED = ON clause

```
CREATE TABLE dbo.ShoppingCart (
ShoppingCartId INT IDENTITY(1,1) PRIMARY KEY NONCLUSTERED,
UserId INT NOT NULL INDEX ix_UserId NONCLUSTERED HASH WITH (BUCKET_COUNT=1000000),
CreatedDate DATETIME2 NOT NULL,
TotalPrice MONEY
) WITH (MEMORY_OPTIMIZED=ON)
GO
```

Insert records into the table

```
INSERT dbo.ShoppingCart VALUES (8798, SYSDATETIME(), NULL)
INSERT dbo.ShoppingCart VALUES (23, SYSDATETIME(), 45.4)
INSERT dbo.ShoppingCart VALUES (80, SYSDATETIME(), NULL)
INSERT dbo.ShoppingCart VALUES (342, SYSDATETIME(), 65.4)
```

Natively compiled stored procedures

Natively compiled stored procedures are Transact-SQL stored procedures that are compiled to native code and can access memory-optimized tables.

This allows for efficient execution of the queries and business logic in the stored procedure.

Native compilation enables faster data access and more efficient query execution than interpreted (traditional) Transact-SQL.

For information on creating natively complied stored procedures, see:

https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/creating-natively-compiled-stored-procedures

Natively compiled stored procedures implement a subset of T-SQL. For more information, see:

https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/supported-features-for-natively-compiled-t-sql-modules

In-Memory OLTP enhancements (SQL Server 2017)

- sp_spaceused is now supported for memory-optimized tables.
- sp_rename is now supported for memory-optimized tables and natively compiled T-SQL modules.
- CASE statements are now supported for natively compiled T-SQL modules.
- The limitation of eight indexes on memory-optimized tables has been eliminated.
- TOP (N) WITH TIES is now supported in natively compiled T-SQL modules.
- ALTER TABLE against memory-optimized tables is now substantially faster in most cases.
- Transaction log redo of memory-optimized tables is now done in parallel. This bolsters faster recovery times and significantly increases the sustained throughput of AlwaysOn Availability Group configuration.
- Memory-optimized filegroup files can now be stored on Azure Storage. Backup/Restore of memory-optimized files on Azure Storage is supported.
- Support for computed columns in memory-optimized tables, including indexes on computed columns.
- Full support for JSON functions in natively compiled modules, and in check constraints.
- CROSS APPLY operator in natively compiled modules.
- Performance of B-tree (nonclustered) index rebuild for MEMORY_OPTIMIZED tables during database recovery has been significantly optimized. This improvement substantially reduces the database recovery time when nonclustered indexes are used.

Columnstore

SQL Server performance features: Columnstore

Columnstore

A technology for storing, retrieving, and managing data by using a columnar data format called a columnstore. You can use columnstore indexes for real-time analytics on your operational workload.

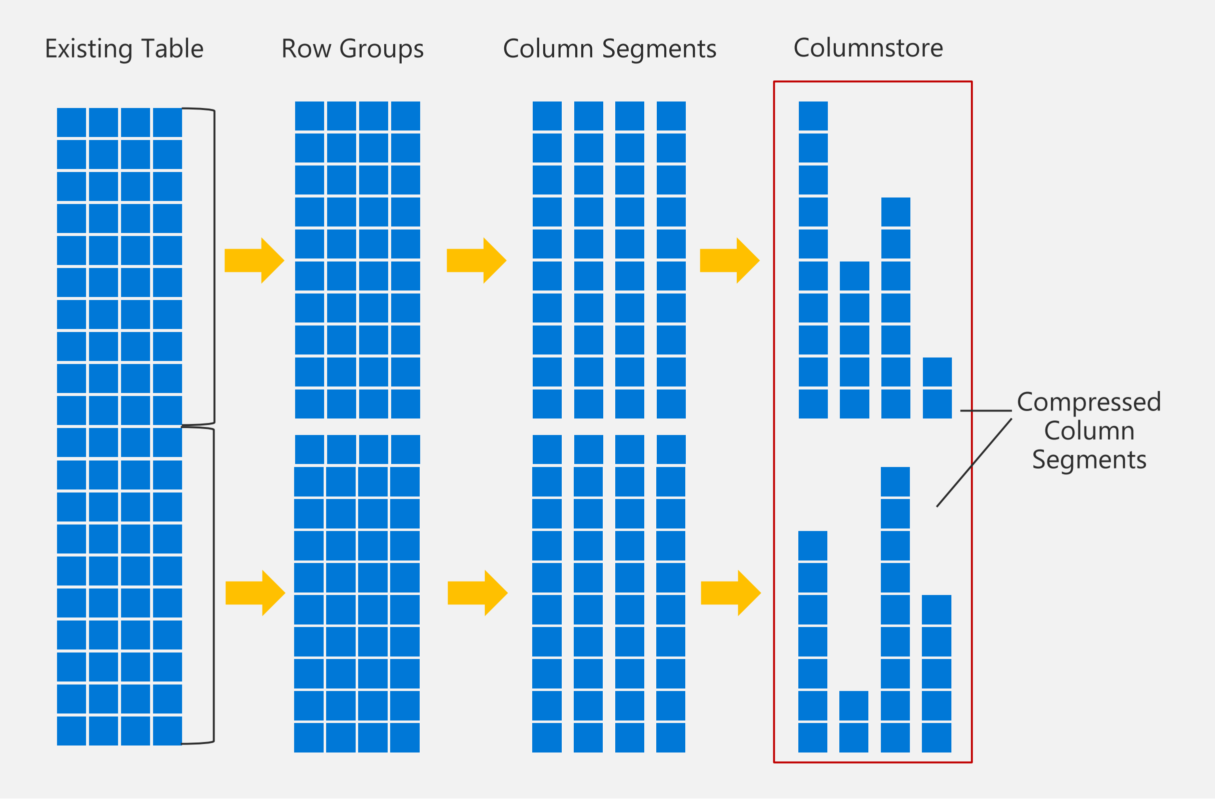
Key benefits

Provides a very high level of data compression, typically 10x, to reduce your data warehouse storage cost significantly. Indexing on a column with repeated values vastly improves performance for analytics.

Improved performance:

- More data fits in memory
- Batch-mode execution

Data stored as columns



Columnstore: Clustered vs. nonclustered indexes

Columnstore

Data that is logically organized as a table with rows and columns, and physically stored in a column-wise data format.

Rowstore

Data that is logically organized as a table with rows and columns, and then physically stored in a row-wise data format.

In SQL Server, rowstore refers to a table where the underlying data storage format is a heap, clustered index, or memory-optimized table.

Columnstore: Clustered vs. nonclustered indexes

Clustered index

The primary storage for the entire table.

Nonclustered index

A secondary index on the standard table (rowstore).

Both columnstore indexes offer high compression (10x) and improved query performance.

Nonclustered indexes enable a standard OLTP workload on the underlying rowstore, and a separate simultaneous analytical workload on the columnstore—with negligible impact to performance (Real-Time Operational Analytics).

Steps to creating a columnstore (NCCI)

Add a columnstore index to the table by executing the T-SQL

```
CREATE NONCLUSTERED COLUMNSTORE INDEX [IX_SalesOrderDetail_ColumnStore]
   ON Sales.SalesOrderDetail
   (UnitPrice, OrderQty, ProductID)
GO
```

Execute the query that should use the columnstore index to scan the table

```
SELECT ProductID, SUM(UnitPrice) SumUnitPrice, AVG(UnitPrice) AvgUnitPrice,
   SUM(OrderQty) SumOrderQty, AVG(OrderQty) AvgOrderQty
FROM Sales.SalesOrderDetail
   GROUP BY ProductID
   ORDER BY ProductID
```

Verify that the columnstore index was used by looking up its object_id and confirming that it appears in the usage stats for the table

```
SELECT * FROM sys.indexes WHERE name = 'IX_SalesOrderDetail_ColumnStore'
GO

SELECT *
FROM sys.dm_db_index_usage_stats
   WHERE database_id = DB_ID('AdventureWorks')
   AND object_id = OBJECT_ID('AdventureWorks.Sales.SalesOrderDetail');
```

Columnstore index enhancements (SQL Server 2017)

- Clustered columnstore indexes now **support LOB columns** (nvarchar(max), varchar(max), varbinary(max))
- **Online** nonclustered columnstore **index build** and **rebuild** support added

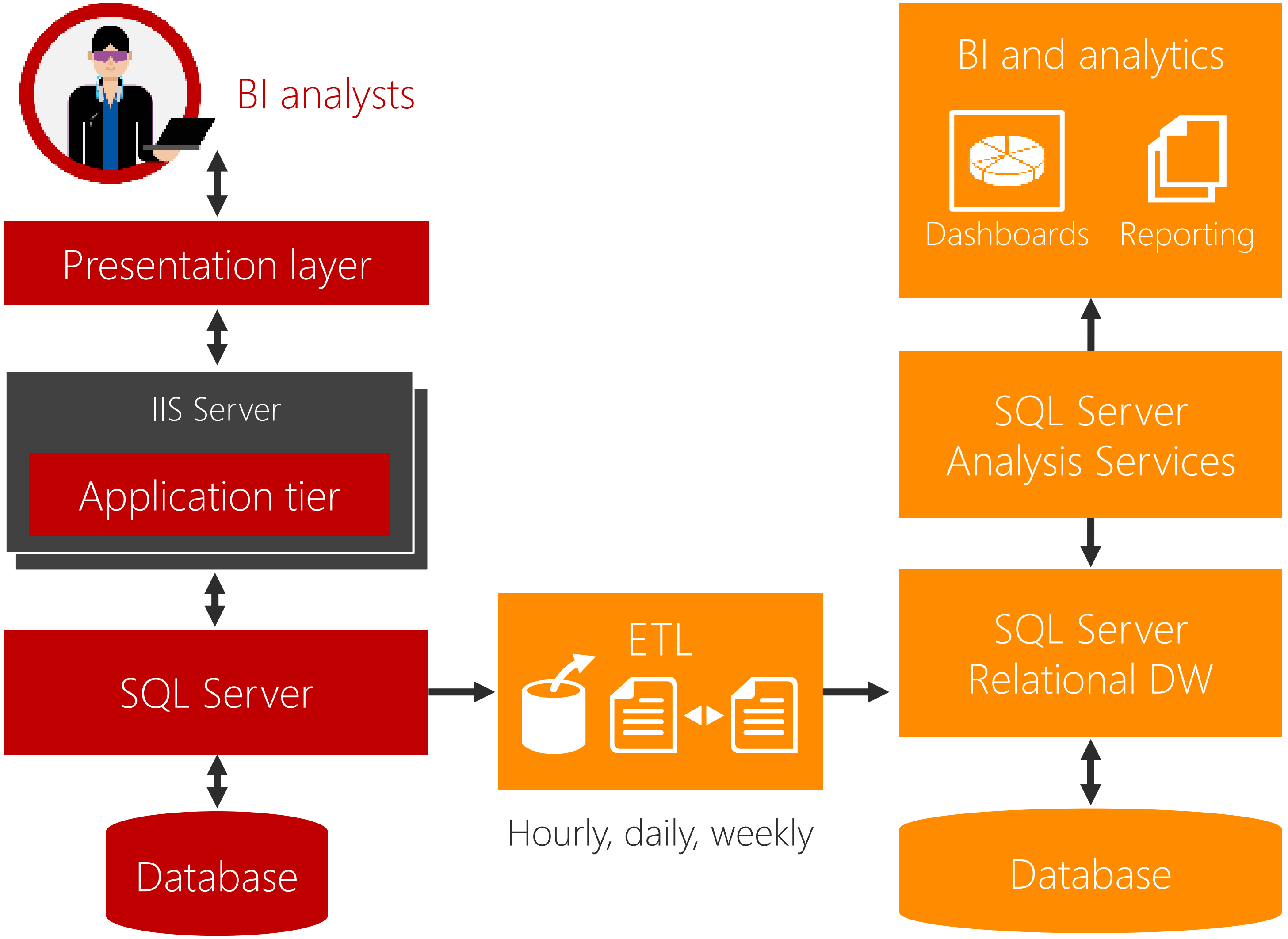
Real-time analytics/ HTAP

Real-time analytics/HTAP

SQL Server’s support for columnstore and In-Memory allows you to generate analytics in real time, direct from your transactional databases. This pattern is called Hybrid Transactional and Analytical Processing (HTAP), because it combines OLTP and OLAP in one database.

- Analytics can be performed on operational data with minimal overhead
- Improving the timeliness of analytics adds significant business value

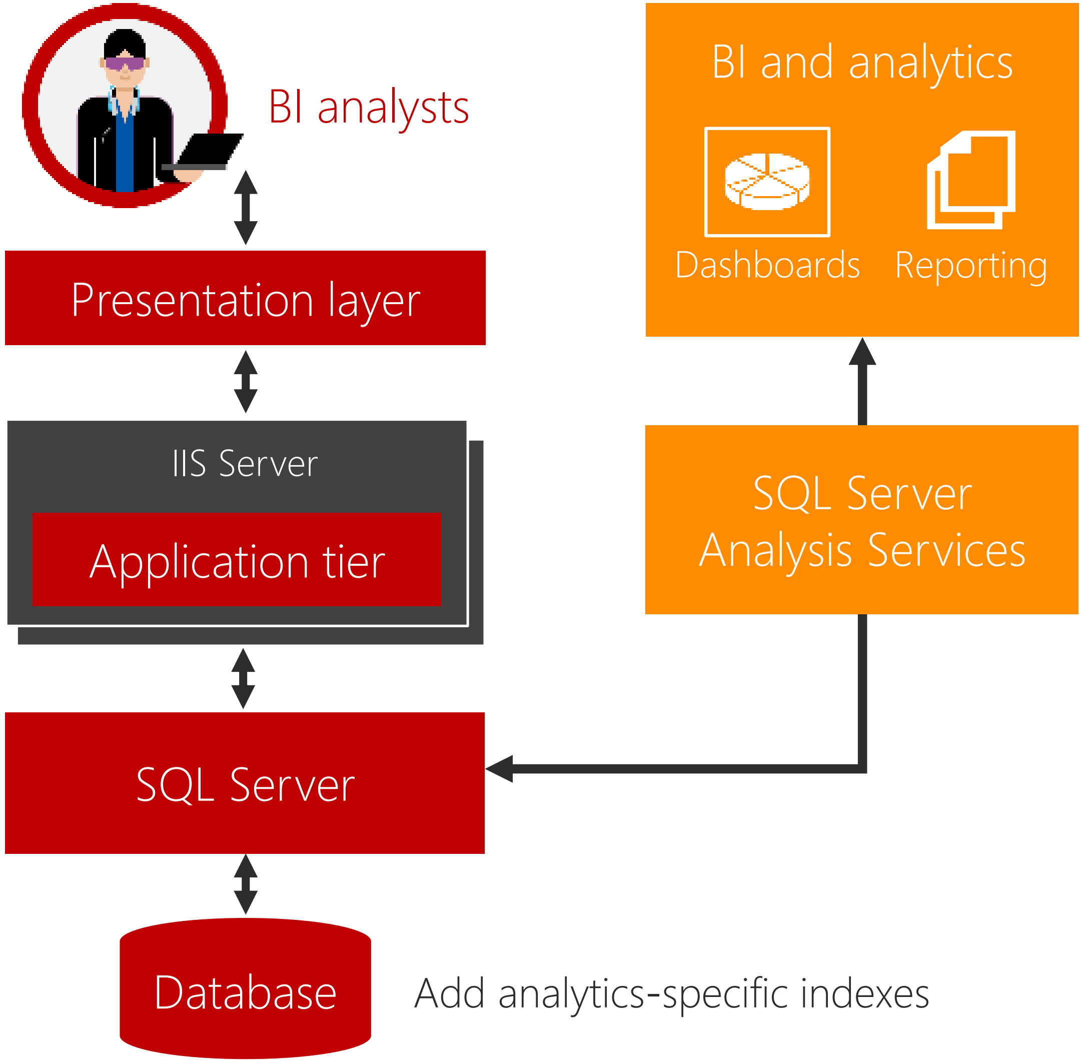
Traditional operational/analytics architecture



Key issues

- Complex implementation
- Requires two servers (capital expenditures and operational expenditures)
- Data latency in analytics
- High demand—requires real-time analytics

Minimizing data latency for analytics



Add analytics-specific indexes

Challenges

- Analytics queries are resource intensive and can cause blocking
- Minimizing impact on operational workloads
- Sub-optimal execution of analytics on relational schema

Benefits

- No data latency
- No ETL
- No separate data warehouse

Real-time analytics/HTAP

The ability to run analytics queries concurrently with operational workloads using the same schema.

Goals:

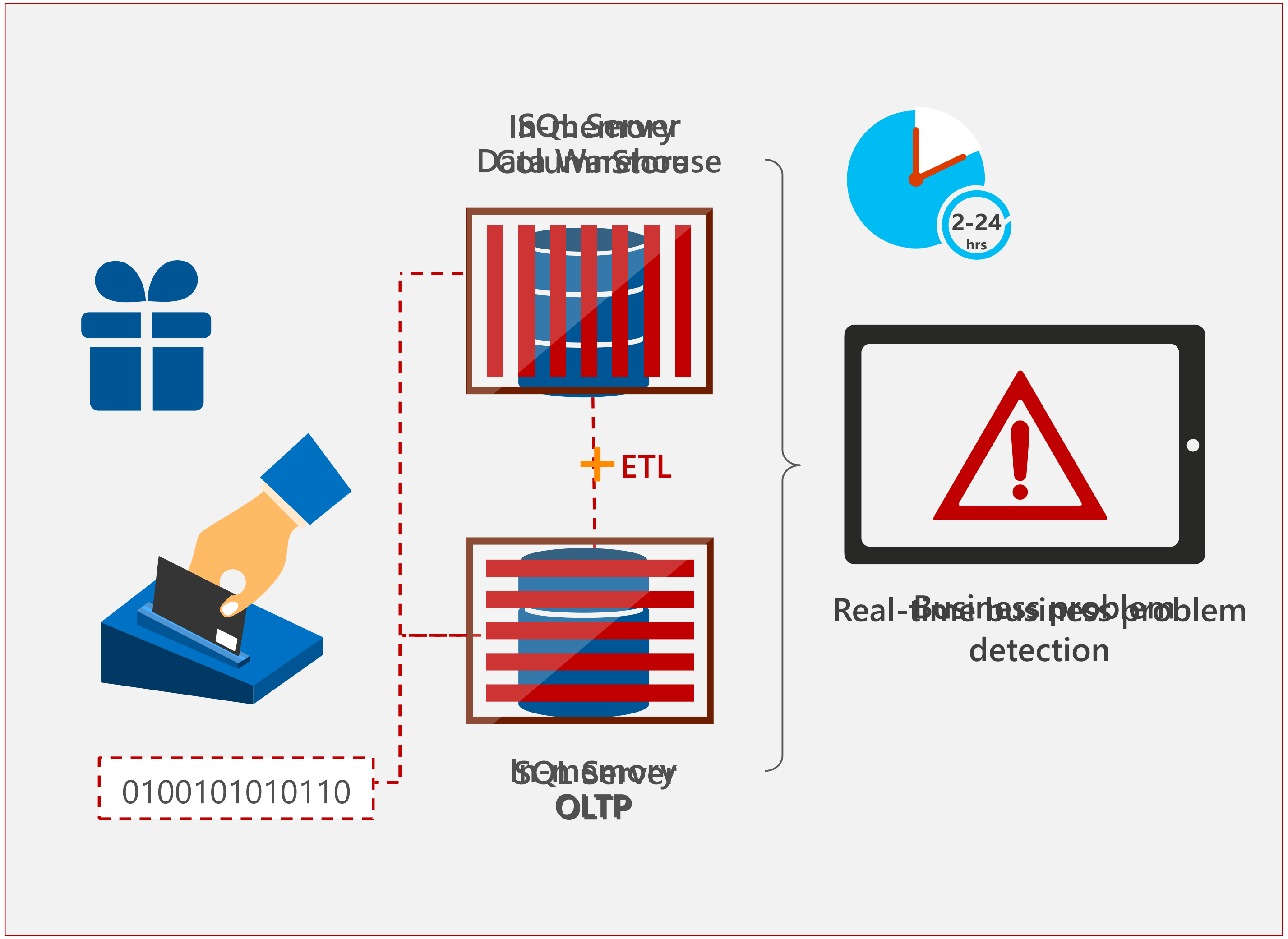
- Minimal impact on operational workloads with concurrent analytics
- Performance analytics for operational schema

Not a replacement for:

- Extreme analytics performance queries that are possible only using customized schemas (for example, Star/Snowflake) and preaggregated cubes
- Data coming from nonrelational sources
- Data coming from multiple relational sources requiring integrated analytics

Real-time operational analytics

In-memory **built-in**



- Improve transactional performance with row-based **in-memory OLTP**
- Speed analytics and reduce storage needs with **ColumnStore** compression
- Combine for **real-time operational analytics (HTAP)**
- Speed query performance without tuning using new **Adaptive Query Processing**
-
- Maintain performance when making app changes with **Automatic Plan Correction**

High availability

High availability and disaster recovery

Simple HADR

VM failure

- Resilience against guest and OS level failures
- Planned and unplanned events
- Minimum downtime for patching and upgrades
- Minutes RTO

Backup/restore

- Protection against accidental or malicious data corruption
- DR protection
- Minutes to hours RTO

Standard HADR

Failover cluster

- Instance level protection
- Automatic failure detection and failover
- Seconds to minutes RTO
- Resilience against OS and SQL Server failures

Basic Availability Groups

- AG with two replicas
- Replaces Database Mirroring

Log shipping

- Warm standbys for DR

Mission critical HADR

Availability Groups

- Database level protection
- Seconds RTO
- No data loss
- Recover from unplanned outage
- No downtime for planned maintenance
- Offload read/backup workload to active secondaries
- Failover to geographically distributed secondary site

Always On

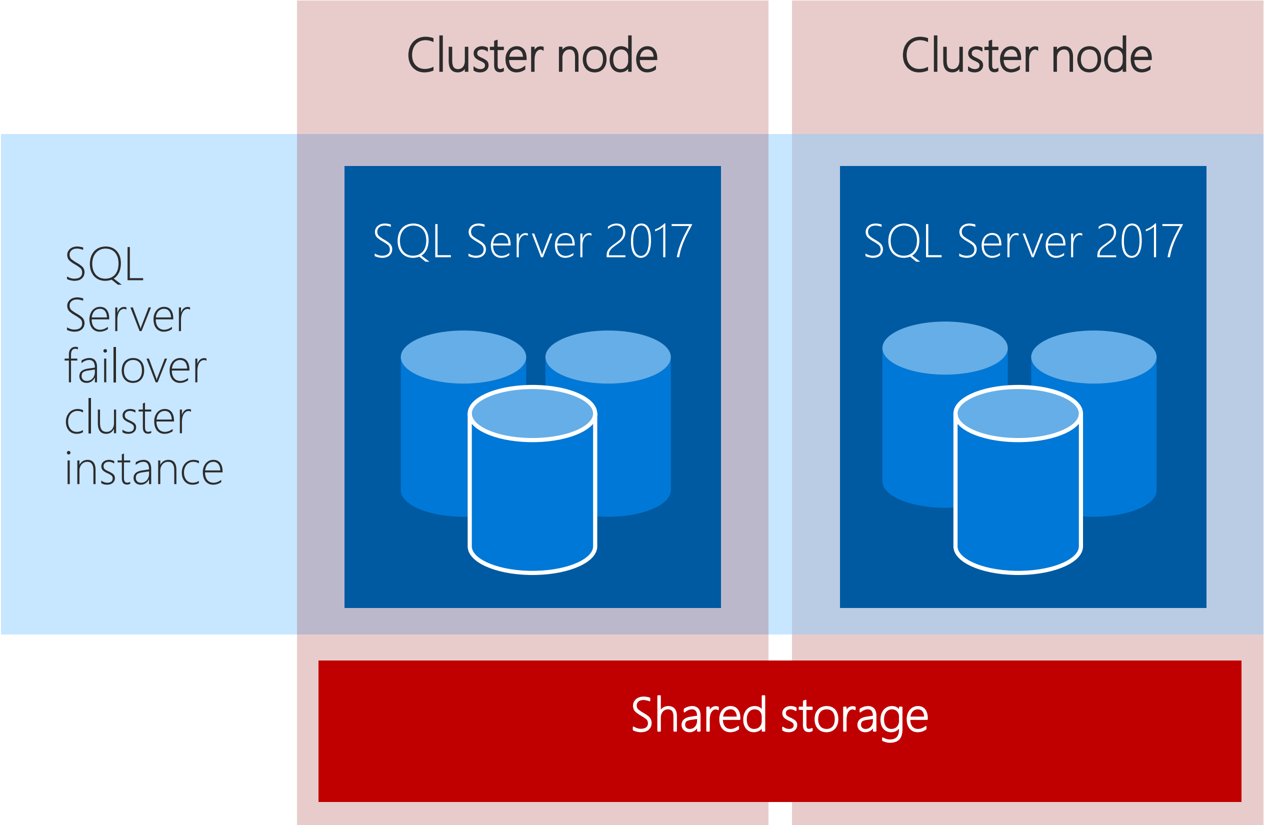
Failover cluster instances
for servers

- Failover on SQL Server instance level
- Shared storage (SAN/SMB)
- Failover can take minutes based on load
- Multi-node clustering
- Passive secondary nodes

Availability Groups
for groups of databases

- Failover on database level
- Direct attached storage
- Failover takes seconds
- Multiple secondaries
- Active secondaries

Failover cluster instances



Server failover

Shared storage

Multi-node clustering

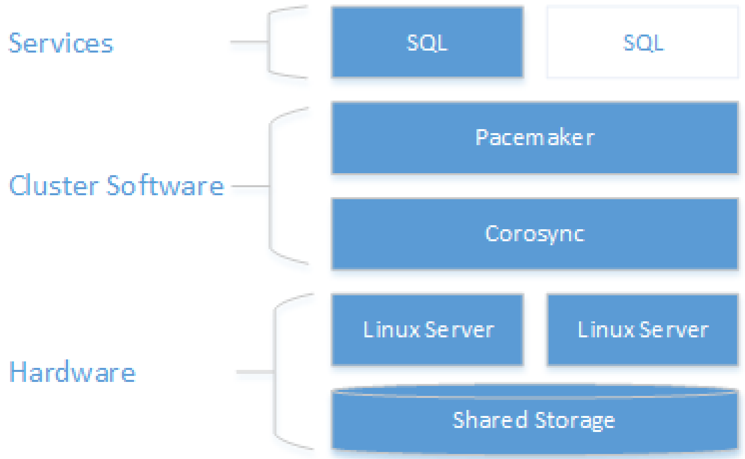
Passive secondary nodes

Failover in minutes

Windows and Linux failover clusters are supported

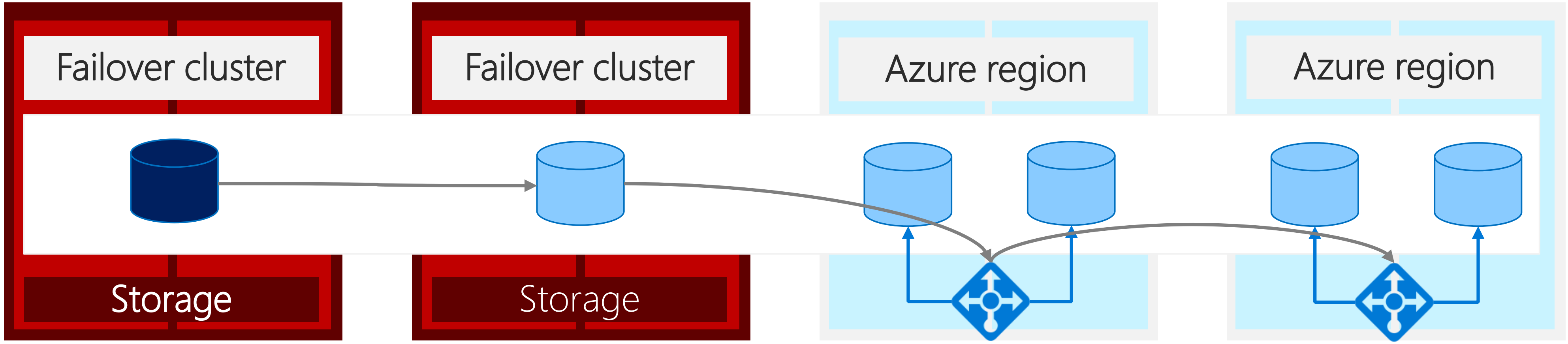
Configuring failover clusters on Linux

1. Set up and configure the operating system on each cluster node.
2. Install and configure SQL Server on each cluster node.
3. Configure shared storage and move database files.
4. Install and configure Pacemaker on each cluster node.
5. Create the cluster.



For full instructions, see: https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-shared-disk-cluster-configure

Always On Availability Groups



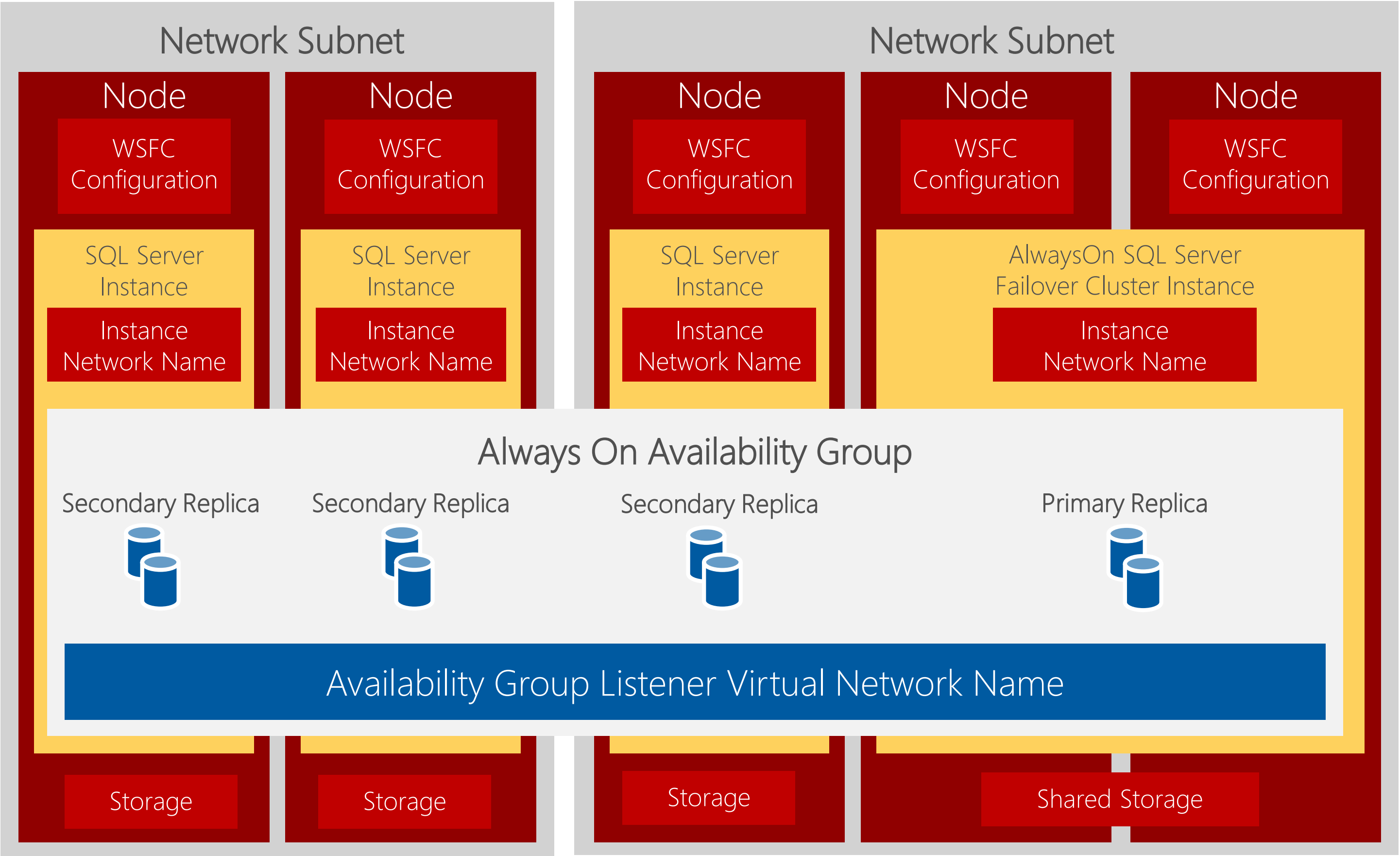
Availability Groups: High availability and disaster recovery solution where one or several databases failover together.

SQL Server 2017 supports one primary, and up to eight secondaries, for a total of nine replicas.

Secondaries can be enabled as read-only replicas, which can be load balanced.

Availability Groups and failover clustering (Windows)

Windows Server Failover Clustering (WSFC) cluster



Network Subnet

Node

WSFC Configuration

SQL Server Instance

Instance Network Name

Storage

Node

WSFC Configuration

SQL Server Instance

Instance Network Name

Storage

Network Subnet

Node

WSFC Configuration

SQL Server Instance

Instance Network Name

Storage

Node

WSFC Configuration

Node

WSFC Configuration

AlwaysOn SQL Server Failover Cluster Instance

Instance Network Name

Shared Storage

Always On Availability Group

Secondary Replica

Secondary Replica

Secondary Replica

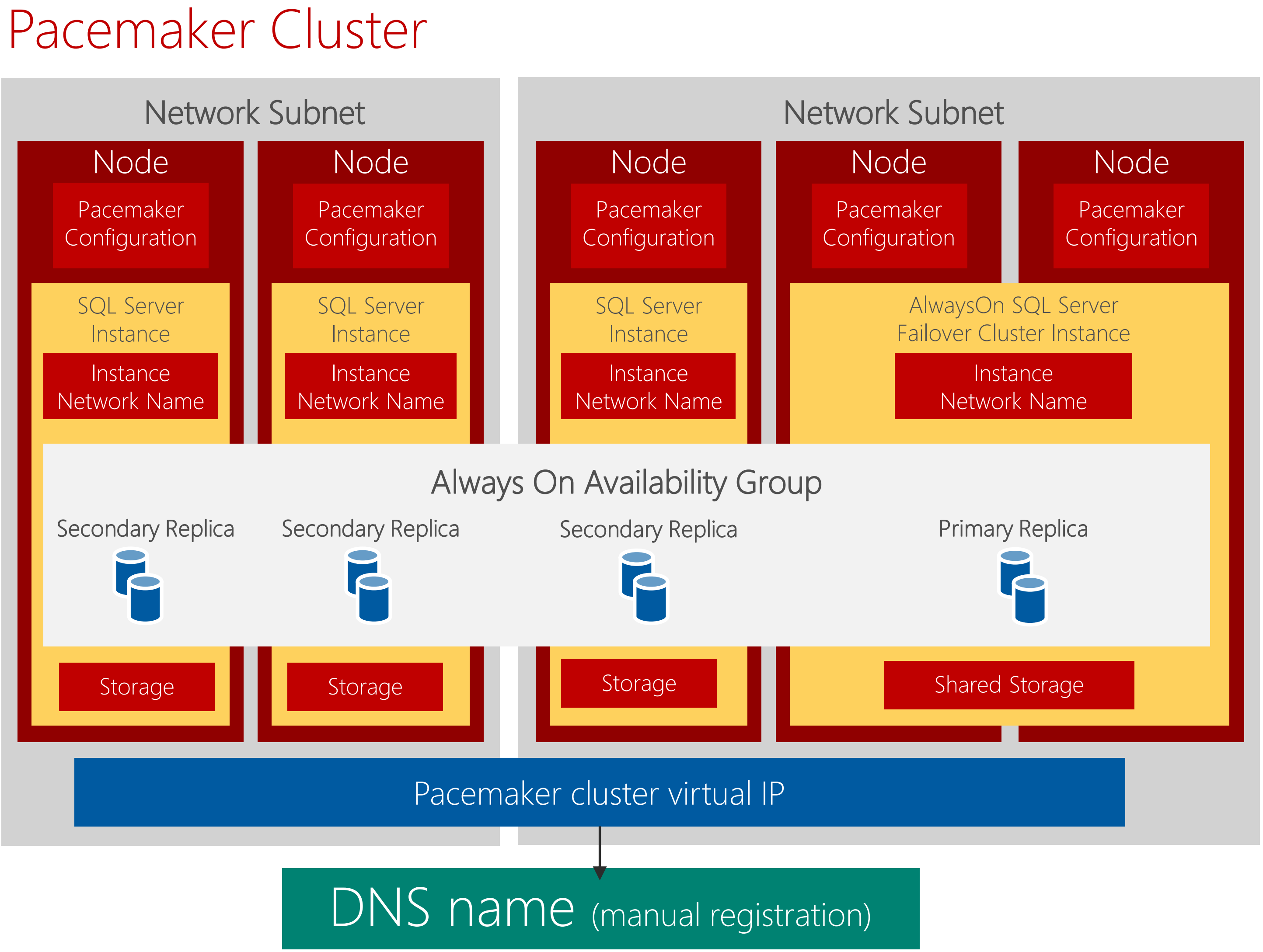
Primary Replica

Availability Group Listener Virtual Network Name

Always On:

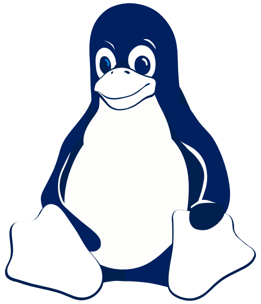
Failover Cluster Instances and Availability Groups work together to ensure data is accessible despite failures

Availability Groups and failover clustering (Linux)



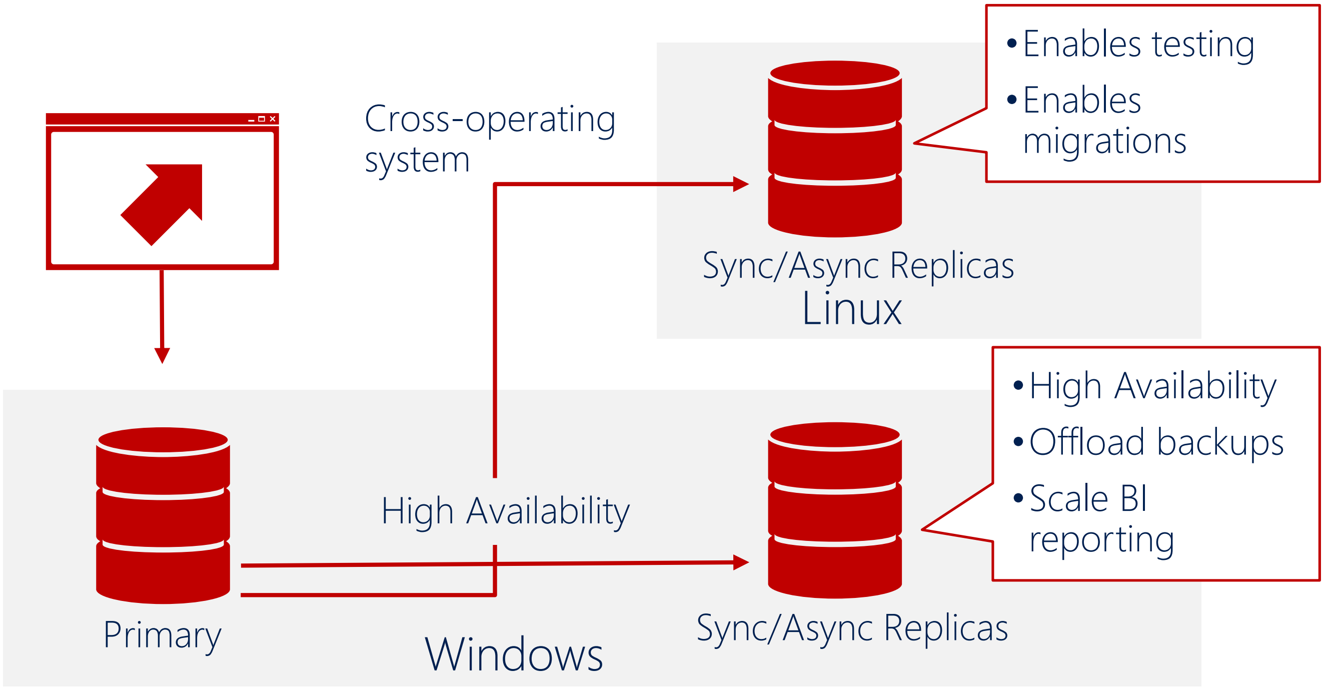
Always On:

Failover Cluster Instances and Availability Groups work together to ensure data is accessible despite failures



Mission critical availability on any platform

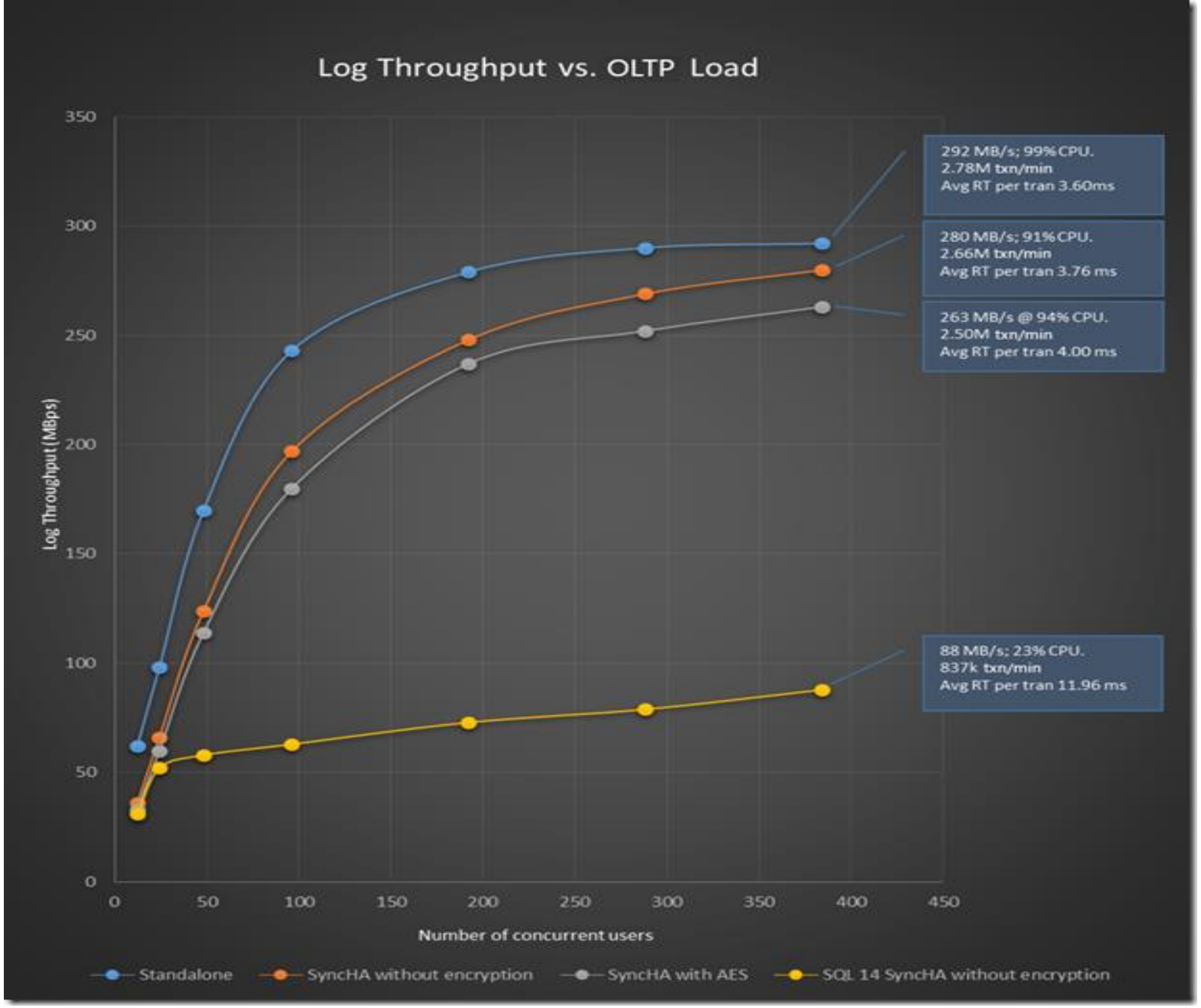
Always On cross-platform capabilities



- Enables testing
- Enables migrations

- High Availability
- Offload backups
- Scale BI reporting

- **Always On Availability Groups for Linux** NEW* **and Windows** for HA and DR
- Flexibility for **HA architectures** NEW*
- Ultimate HA with **OS-level redundancy and failover**
- **Load balancing** of readable secondaries



292 MB/s; 99% CPU.
2.78M txn/min
Avg RT per tran 3.60ms

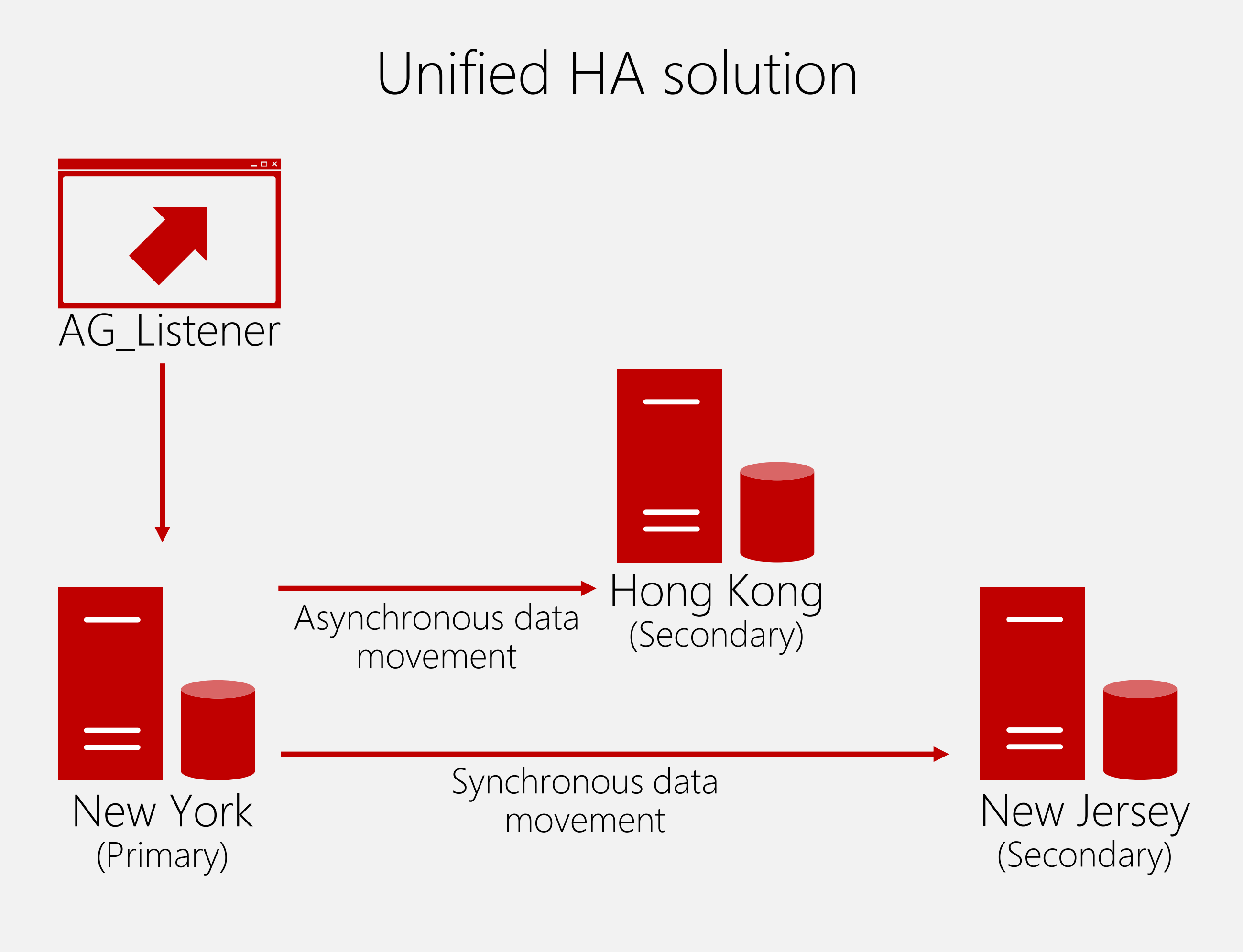
280 MB/s; 91% CPU.
2.66M txn/min
Avg RT per tran 3.76 ms

263 MB/s @ 94% CPU.
2.50M txn/min
Avg RT per tran 4.00 ms

88 MB/s; 23% CPU.
837k txn/min
Avg RT per tran 11.96 ms

https://blogs.msdn.microsoft.com/bobsql/2016/09/26/sql-server-2016-it-just-runs-faster-always-on-availability-groups-turbocharged/

Enhanced Always On Availability Groups (SQL Server 2017)



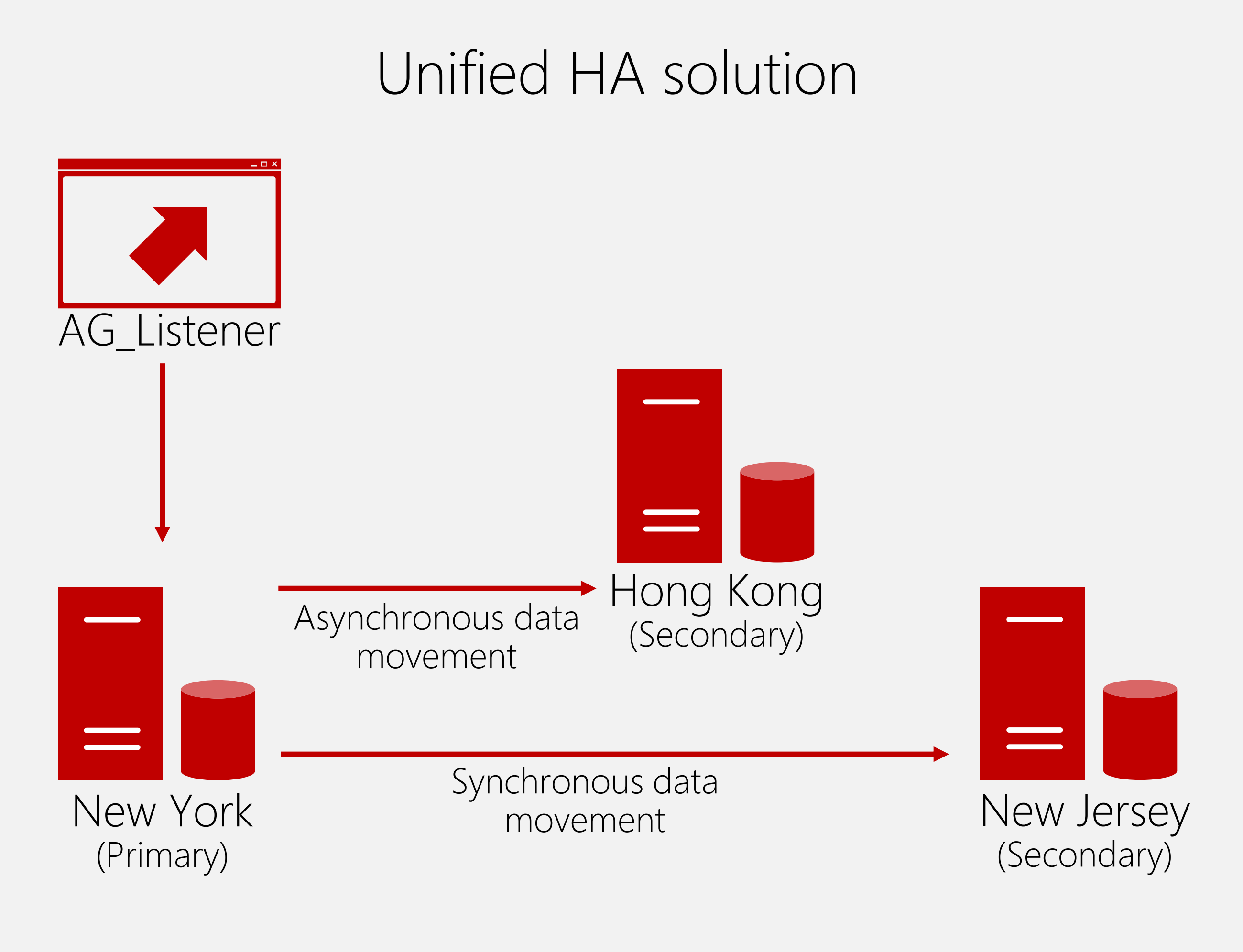
Guarantee commits on synchronous secondary replicas

Use **REQUIRED_COPIES_TO_COMMIT** with CREATE AVAILABILITY GROUP or ALTER AVAILABILITY GROUP.

When REQUIRED_COPIES_TO_COMMIT is set to a value higher than 0, transactions at the primary replica databases will wait until the transaction is committed on the specified number of synchronous secondary replica database transaction logs.

If enough synchronous secondary replicas are not online, write transactions to primary replicas will stop until communication with sufficient secondary replicas resumes.

Enhanced Always On Availability Groups (SQL Server 2017)



CLUSTER_TYPE

CLUSTER_TYPE Use with CREATE AVAILABILITY GROUP. Identifies the type of server cluster manager that manages an availability group. Can be one of the following types:

WSFC: Windows Server failover cluster. On Windows, it is the default value for CLUSTER_TYPE.

EXTERNAL: A cluster manager that is not a Windows Server failover cluster—for example, on Linux with Pacemaker.

NONE: No cluster manager. Used for a read-scale availability group.

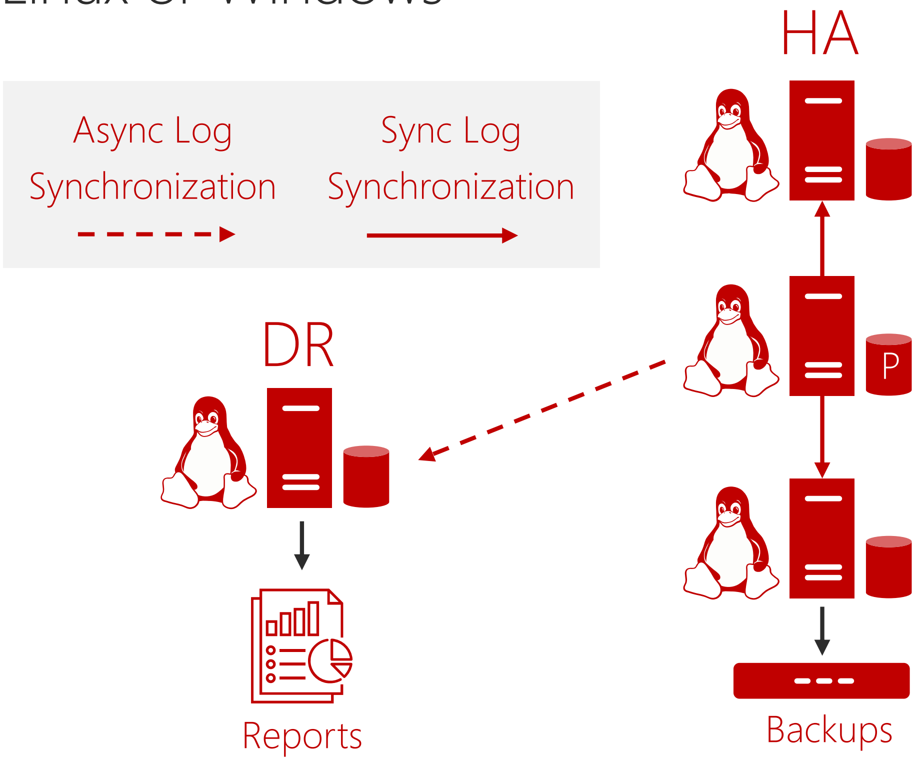
Build a mission critical enterprise application

Scenario

- All-Linux infrastructure
- Application-level protection
- Automatic and “within seconds” failover during unplanned outages
- No downtime during planned maintenance
- Performance-sensitive application
- DR required for regulatory compliance

Solution

HADR with Always On Availability Groups on Linux or Windows



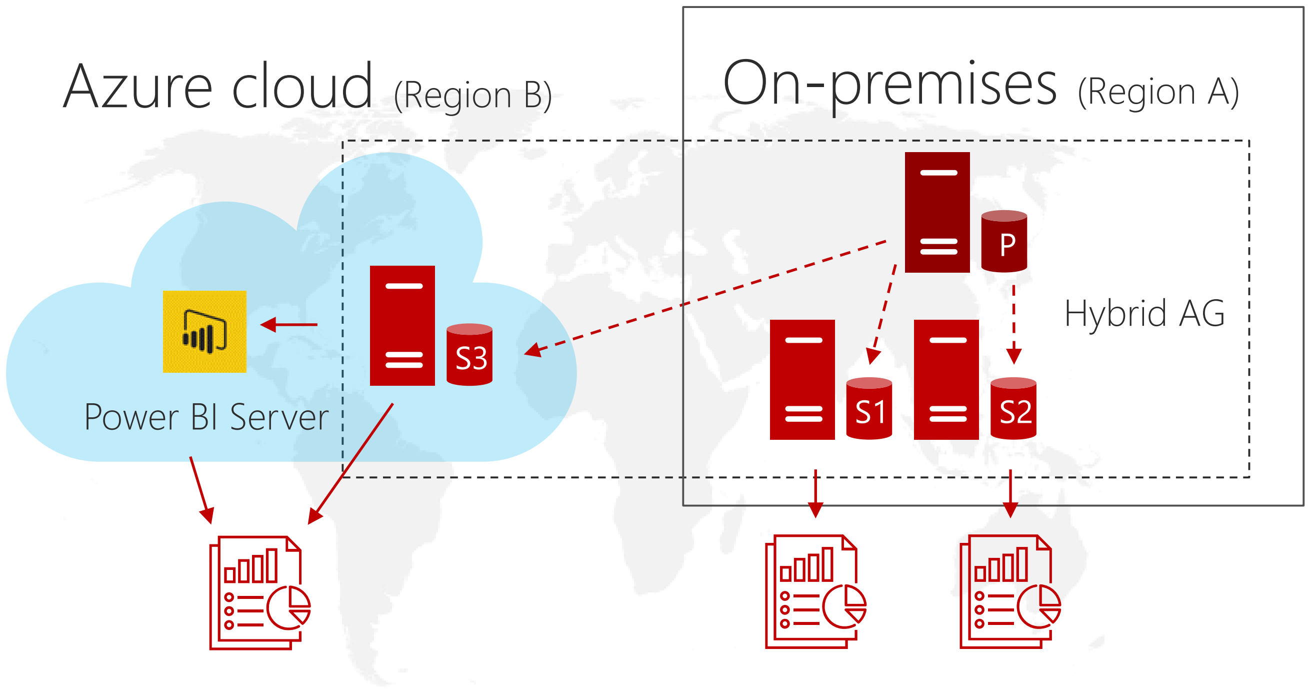
Provide responsive regional BI with Azure and AG

Scenario

- Primary replica in on-premises datacenter
- Secondary read-only replicas in on-premises datacenter used for reporting/BI
- BI generated in other geographical regions performs poorly because of network bandwidth limitations
- No on-premises datacenters in other geographical regions

Solution

Hybrid Availability Group with read-only secondary in Azure (other region)



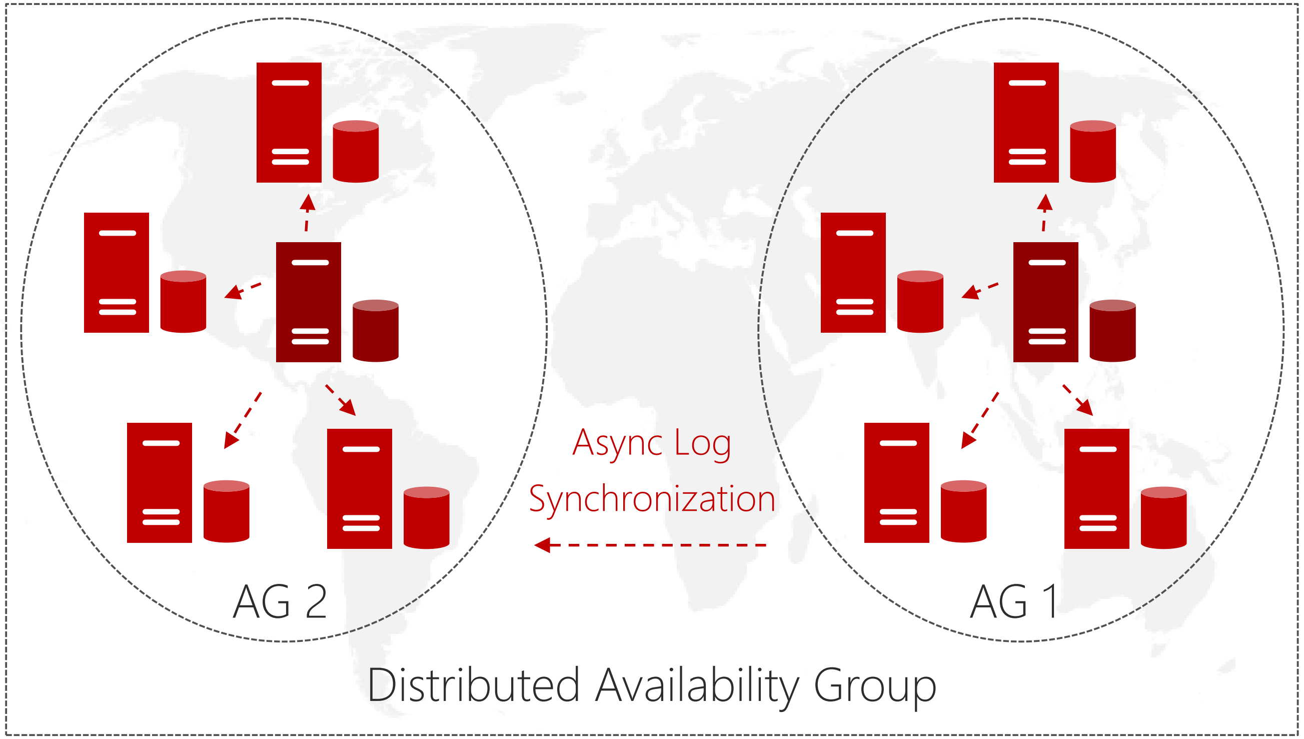
Scale/DR with Distributed Availability Groups

Scenario

- Availability Group must span multiple datacenters
- Not possible to add all servers to a single WSFC (datacenter networks/inter-domain trust)
- Secondary datacenter provides DR
- Geographically distributed read-only replicas required

Solution

Distributed Always On Availability Groups on Linux or Windows



Migration/testing

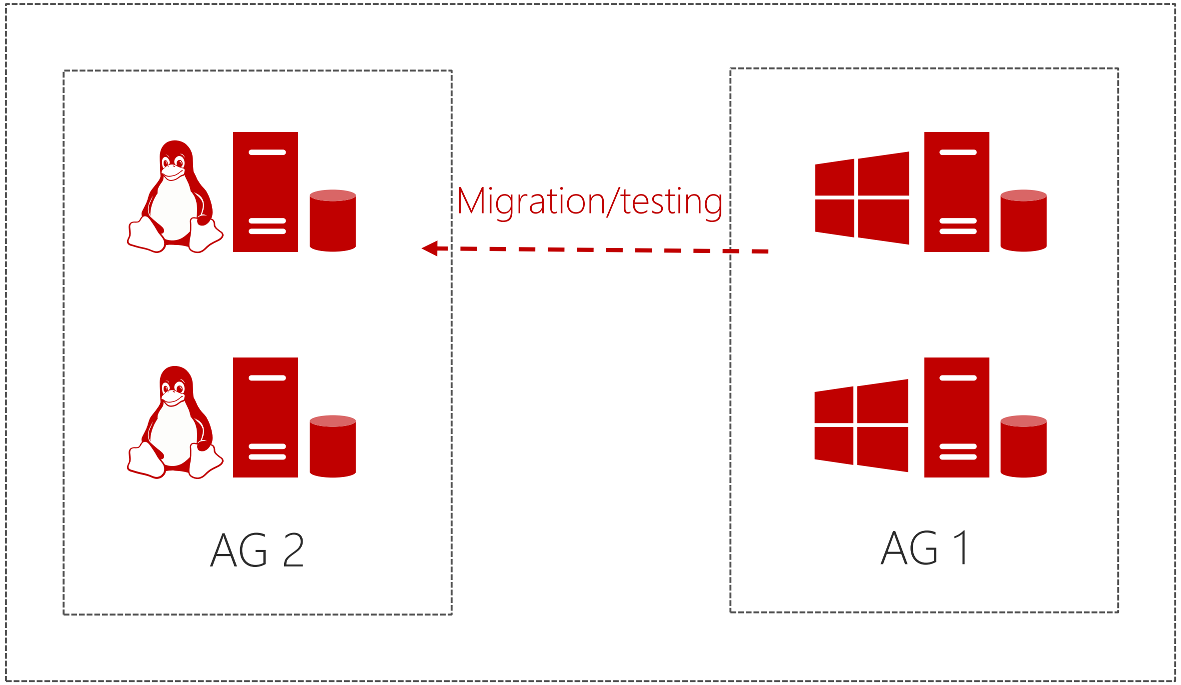
Scenarios

- ISV solution built on SQL Server on Windows
  - Linux Certification

- Enterprise moving to an all-Linux infrastructure
  - Rigorous business requirements
  - Seamless migration

Solution

Minimum downtime *and* HA for cross-platform migrations with Distributed Availability Groups



Improve read concurrency with read-scale Availability Groups

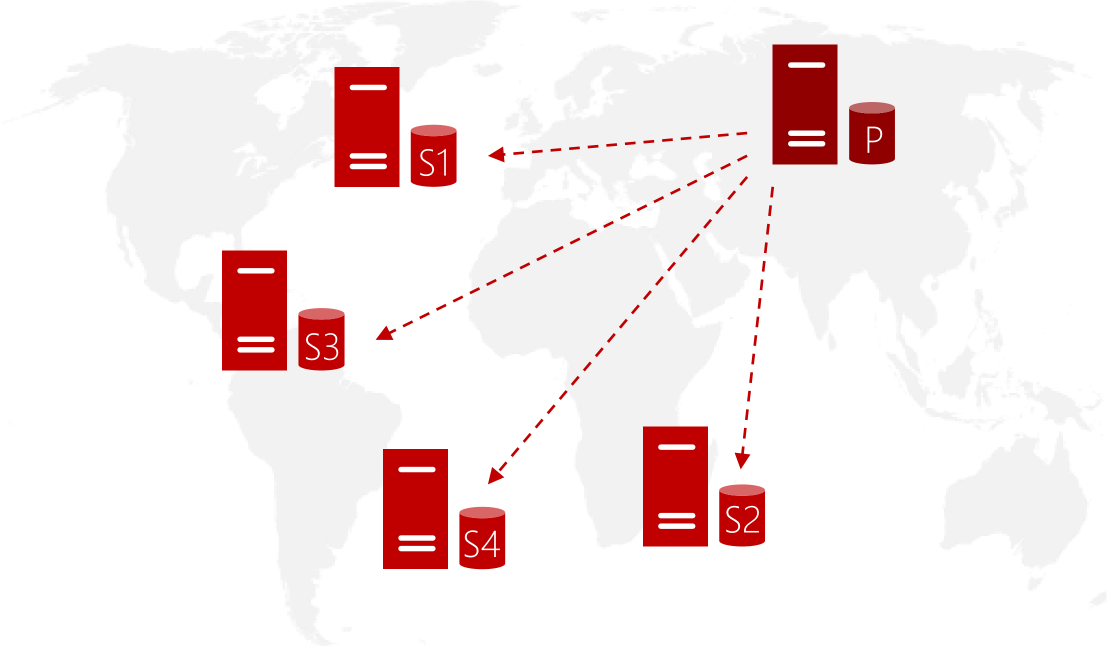
Scenario

- SaaS app (website)
- Catalog database with high volume of concurrent read-only transactions
- Bottlenecks on Availability Groups primary due to read workloads
- Increased response time
- HA/DR elements of Availability Groups not required

Solution

Read-scale Availability Groups

- **No cluster required**
- Both Linux and Windows



Temporal tables

Why temporal?

Time travel

Data audit

Slowly changing dimensions

Repair record-level corruptions

Data changes over time

- Tracking and analyzing changes is often important

Temporal in DB

- Automatically tracks history of data changes
- Enables easy querying of historical data states

Advantages over workarounds

- Simplifies app development and maintenance
- Efficiently handles complex logic in DB engine

Temporal enhancements (SQL Server 2017)

- System-versioned temporal tables now support CASCADE DELETE and CASCADE UPDATE
- Temporal tables retention policy support added

Upgrading and migrating to SQL Server 2017

Upgrade and migration tools

Data Migration Assistant (DMA)

- Upgrade from previous version of SQL Server (on-premises or SQL Server 2017 in Azure VM)

SQL Server Migration Assistant

- Migrate from Oracle, MySQL, SAP ASE, DB2, or Access to SQL Server 2017 (on-premises or SQL Server 2017 in Azure VM)

Azure Database Migration Service

- Migrate from SQL Server, Oracle, or MySQL to Azure SQL Database or SQL Server 2017 in Azure VM

Upgrading to SQL Server 2017

In-place or side-by-side upgrade path from:

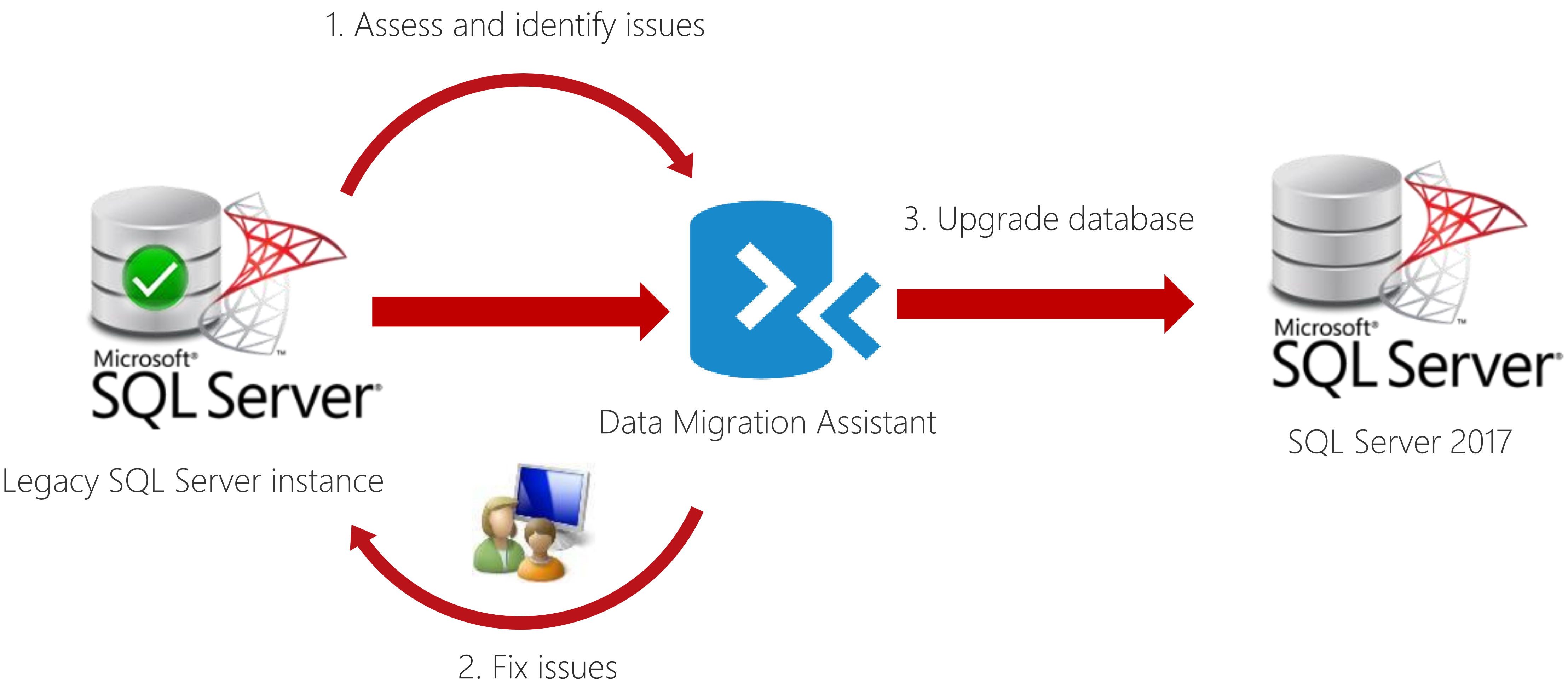
- SQL Server 2008
- SQL Server 2008 R2
- SQL Server 2012
- SQL Server 2014
- SQL Server 2016

Side-by-side upgrade path from:

- SQL Server 2005

Use Data Migration Assistant to prepare for migration

DMA: Assess and upgrade schema



1. Assess and identify issues

3. Upgrade database

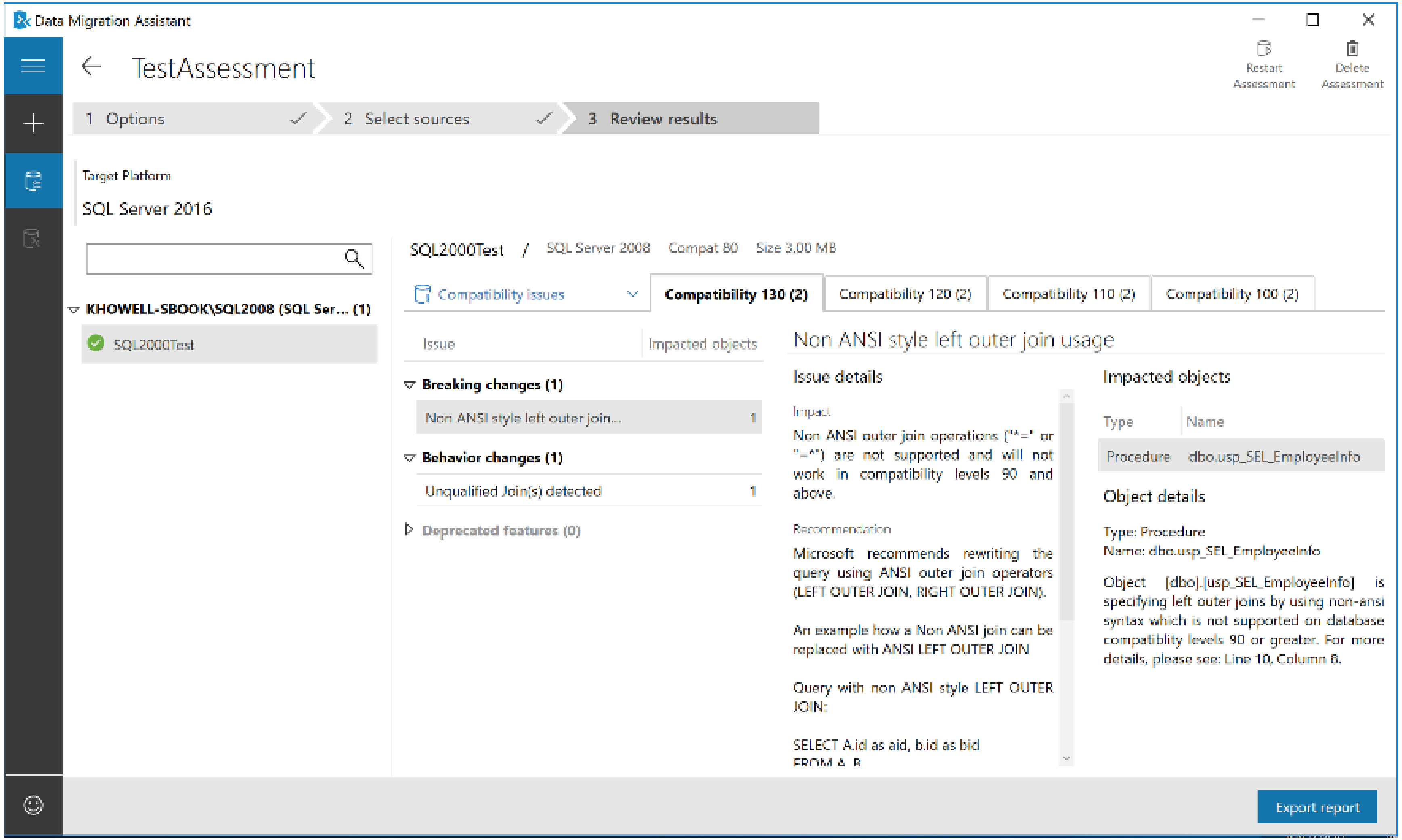
Legacy SQL Server instance

Data Migration Assistant

SQL Server 2017

2. Fix issues

Data Migration Assistant



TestAssessment

Target Platform

SQL Server 2016

KHOWELL-SBOOK\SQL2008 (SQL Ser... (1)

✓ SQL2000Test

SQL2000Test / SQL Server 2008 Compat 80 Size 3.00 MB

Compatibility issues | **Compatibility 130 (2)** | Compatibility 120 (2) | Compatibility 110 (2) | Compatibility 100 (2)

Issue	Impacted objects
Breaking changes (1)	
Non ANSI style left outer join...	1
Behavior changes (1)	
Unqualified Join(s) detected	1
Deprecated features (0)	

Non ANSI style left outer join usage

Issue details

Impact

Non ANSI outer join operations ("*=" or "=*") are not supported and will not work in compatibility levels 90 and above.

Recommendation

Microsoft recommends rewriting the query using ANSI outer join operators (LEFT OUTER JOIN, RIGHT OUTER JOIN).

An example how a Non ANSI join can be replaced with ANSI LEFT OUTER JOIN

Query with non ANSI style LEFT OUTER JOIN:

```
SELECT A.id as aid, b.id as bid
FROM A, B
```

Impacted objects

Type	Name
Procedure	dbo.usp_SEL_EmployeeInfo

Object details

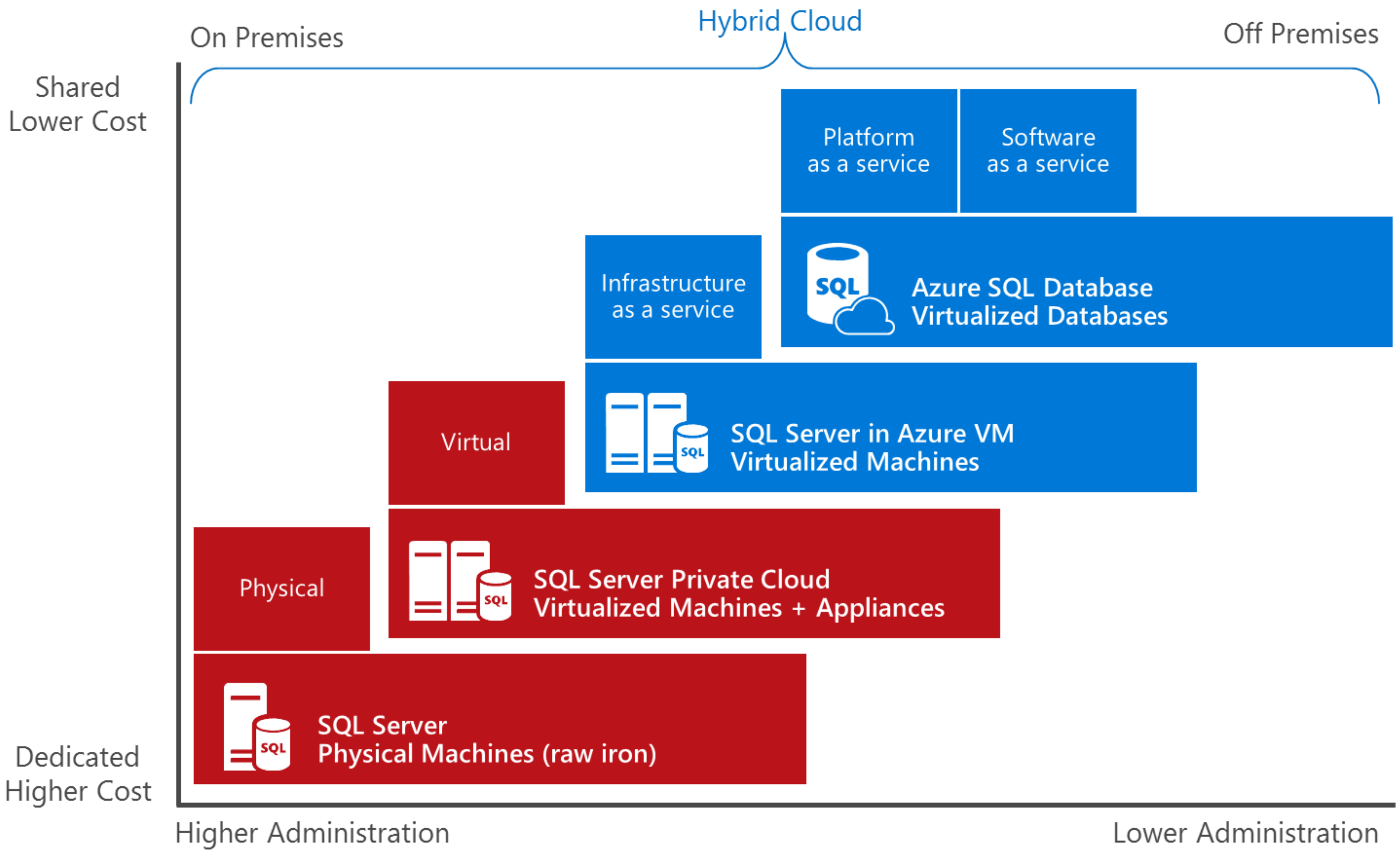
Type: Procedure
Name: dbo.usp_SEL_EmployeeInfo

Object [dbo].[usp_SEL_EmployeeInfo] is specifying left outer joins by using non-ansi syntax which is not supported on database compatiblity levels 90 or greater. For more details, please see: Line 10, Column 8.

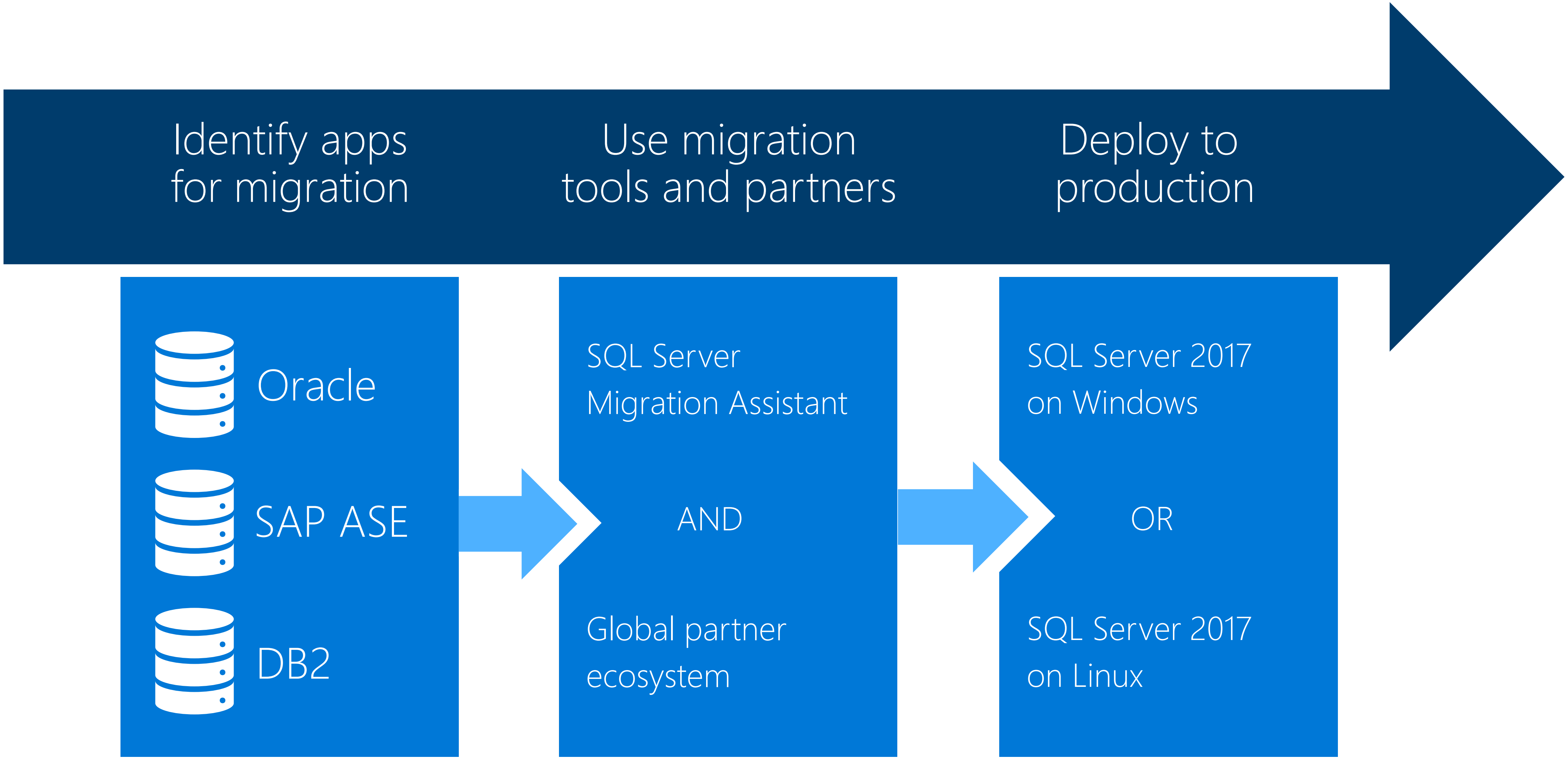
https://www.microsoft.com/en-us/download/details.aspx?id=53595

Choosing a migration target

"What's the best path for me?"



Migrating to SQL Server 2017 from other platforms



Identify apps for migration

Use migration tools and partners

Deploy to production

Oracle

SAP ASE

DB2

SQL Server Migration Assistant

AND

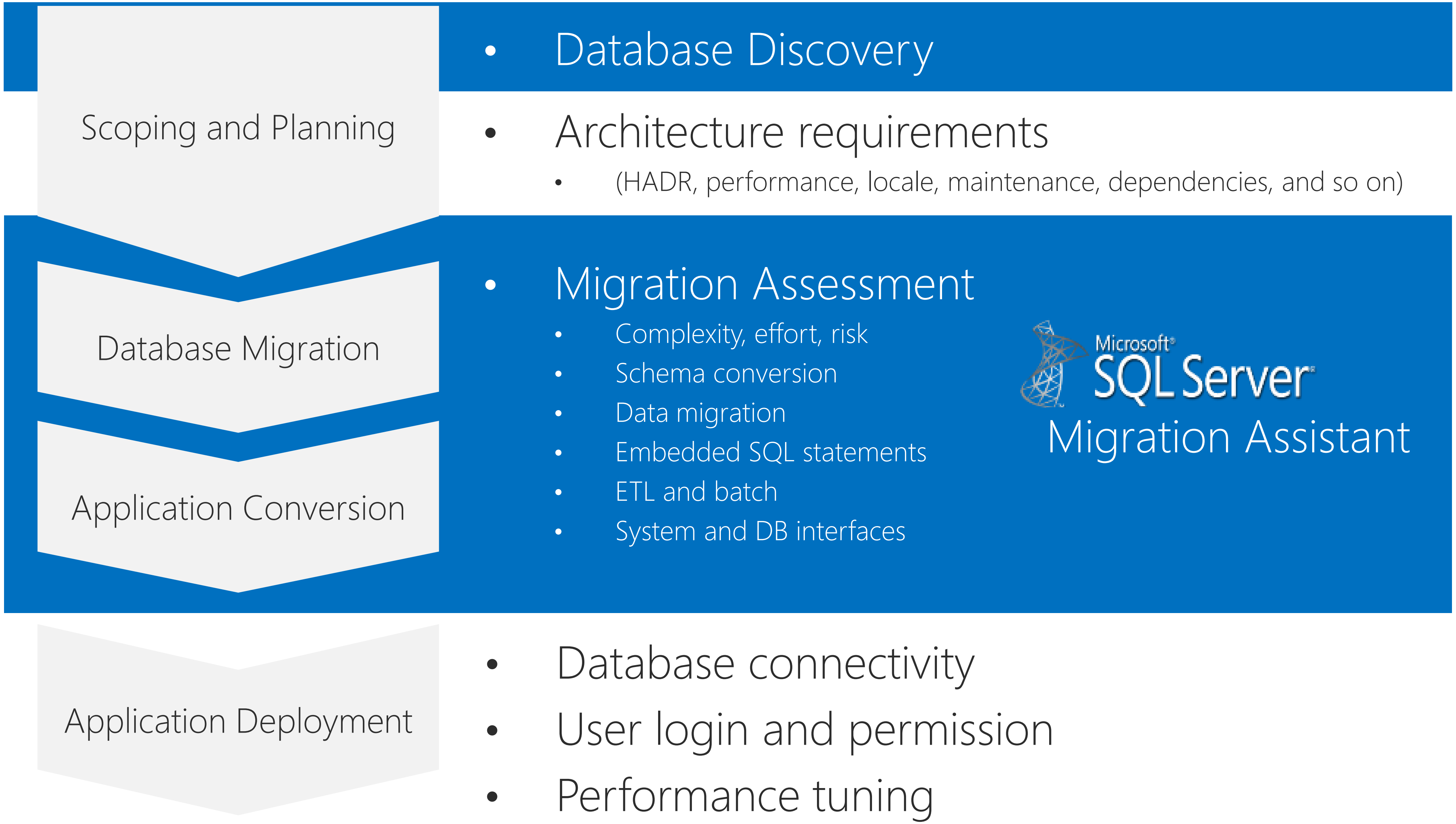
Global partner ecosystem

SQL Server 2017 on Windows

OR

SQL Server 2017 on Linux

Database and application migration process



Scoping and Planning

- Database Discovery
- Architecture requirements
  - (HADR, performance, locale, maintenance, dependencies, and so on)

Database Migration

Application Conversion

- Migration Assessment
  - Complexity, effort, risk
  - Schema conversion
  - Data migration
  - Embedded SQL statements
  - ETL and batch
  - System and DB interfaces

Microsoft SQL Server Migration Assistant

Application Deployment

- Database connectivity
- User login and permission
- Performance tuning

SQL Server Migration Assistant (SSMA)

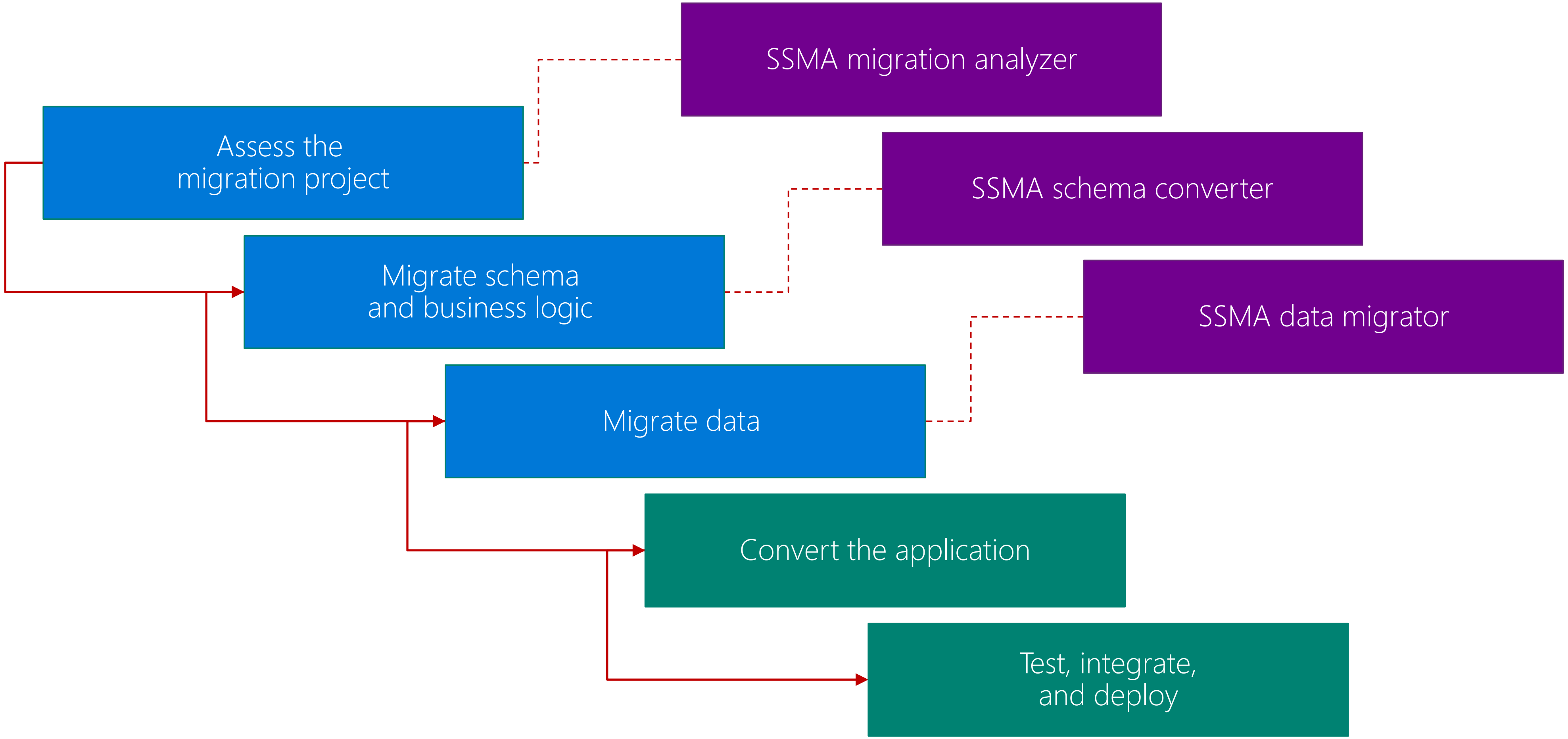
Automates and simplifies all phases of database migration

Migration Analyzer	Assess migration complexity
Schema Converter	Convert schema and business logic
Data Migrator	Migrate data
Migration Tester	Validate converted database code

Supports migration from DB2, Oracle, SAP ASE, MySQL, or Access to SQL Server

Using SQL Server Migration Assistant (SSMA)

SSMA: Automates components of database migrations to SQL Server;
DB2, Oracle, Sybase, Access, and MySQL analyzers are available



Azure solution paths

INFRASTRUCTURE-AS-A-SERVICE **(IaaS)**

SQL Server in an
Azure Virtual Machine

Full control and flexibility

Highly customized system to address the application's specific performance and availability requirements.

Move
existing apps

Development
and test
environment

Hybrid HA and
disaster recovery

PLATFORM-AS-A-SERVICE (PaaS)

Azure SQL Database

Simplified administration

Do not have to manage any VMs, OS or database software, including upgrades, high availability, and backups.

Cloud-designed
business apps

Websites and
mobile apps

Extend on-premises
apps

Azure migration tools and services

Assess → Migrate

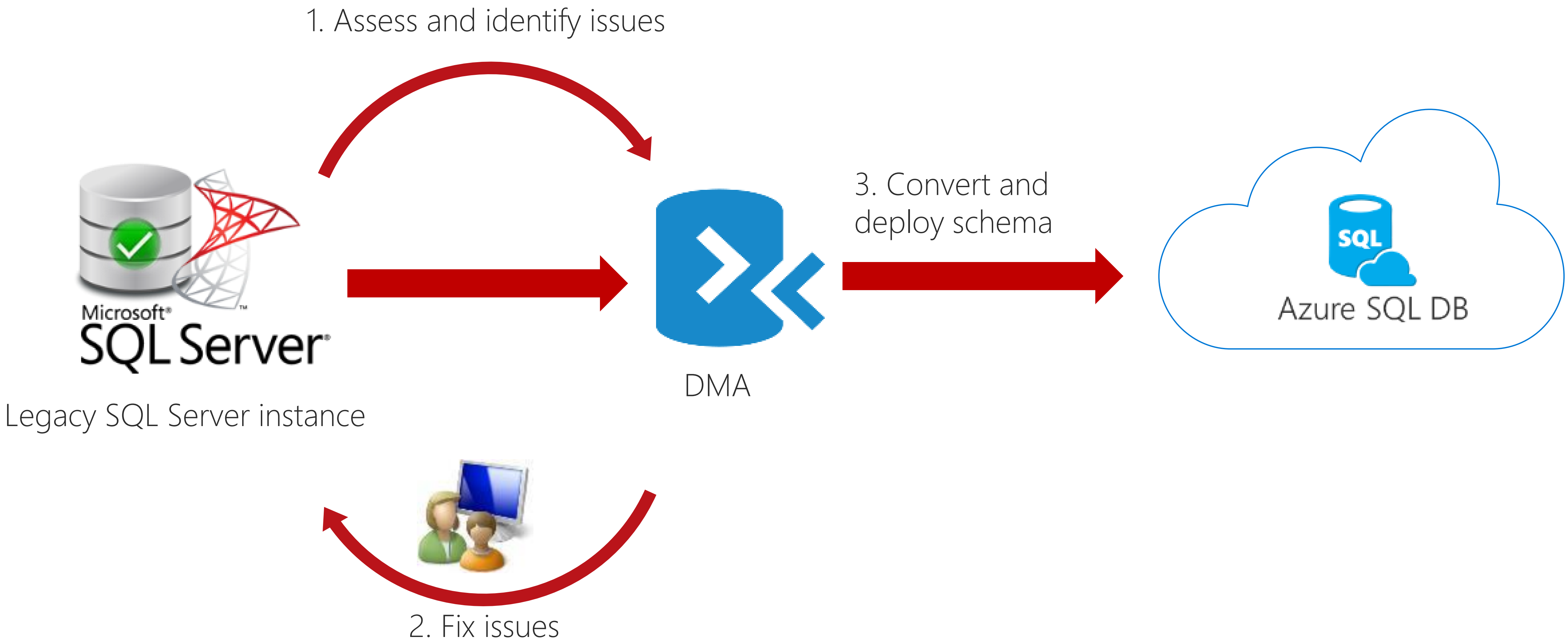
Data Migration Assistant

- Rich assessments at scale
- Feature recommendations
- Schema conversions

Azure Database Migration Service

- MS and non-MS source support
- Built for scale and reliability
- Built with enterprise security and privacy

DMA: Assess and migrate schema



1. Assess and identify issues

Legacy SQL Server instance

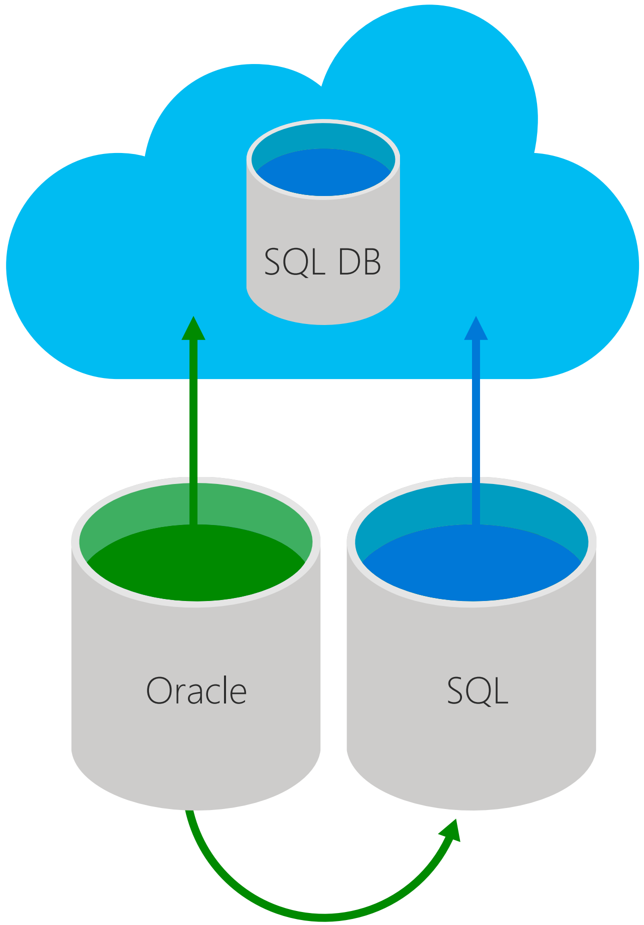
DMA

3. Convert and deploy schema

Azure SQL DB

2. Fix issues

Azure Database Migration Service



Accelerating your journey to the cloud

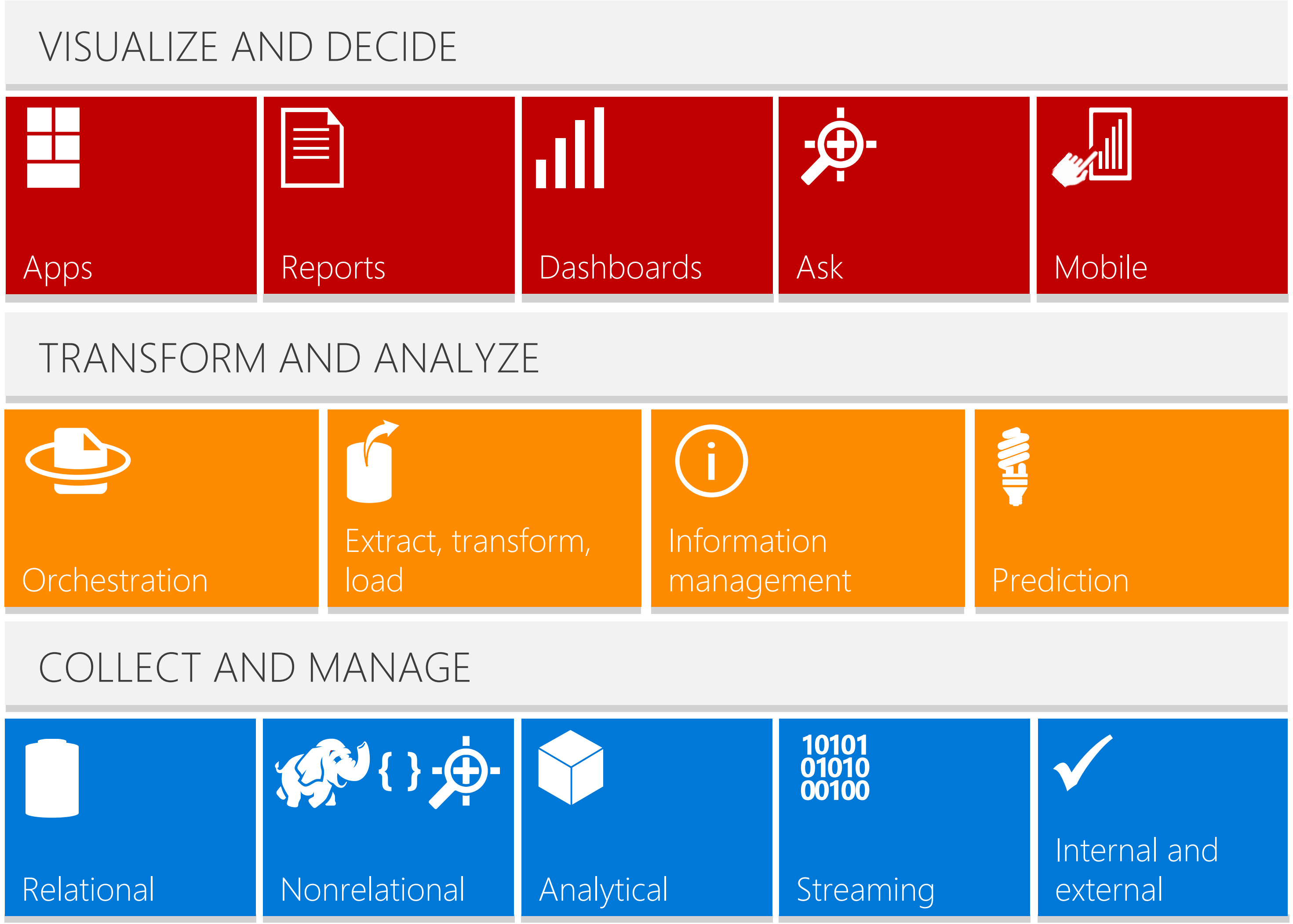
- Streamline database migration to Azure SQL Database (PaaS)
- Managed service platform for migrating databases
- Migrate SQL Server and third-party databases to Azure SQL Database

The Microsoft data platform

Microsoft Azure

Office

Microsoft® SQL Server®



VISUALIZE AND DECIDE

- Apps
- Reports
- Dashboards
- Ask
- Mobile

TRANSFORM AND ANALYZE

- Orchestration
- Extract, transform, load
- Information management
- Prediction

COLLECT AND MANAGE

- Relational
- Nonrelational
- Analytical
- Streaming
- Internal and external

Microsoft

The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.