SQL Server 2016

Session 17

New Features of SQL Server 2016

Objectives

- List new features of SQL Server 2016
- Explain Real-time Operational Analytics
- Describe native JSON support
- Explain AlwaysOn feature
- Describe enhancements made to In-Memory OLTP (Hekaton)

New Features of SQL Server 2016

- Always Encrypted
- Stretch Database
- Real-time Operational Analytics
- PolyBase with SQL Server
- Native JSON Support
- Support for in-database analytics with R-integration
- Query Store
- Dynamic Data Masking

SQL Server 2016

Enhancements in SQL Server 2016

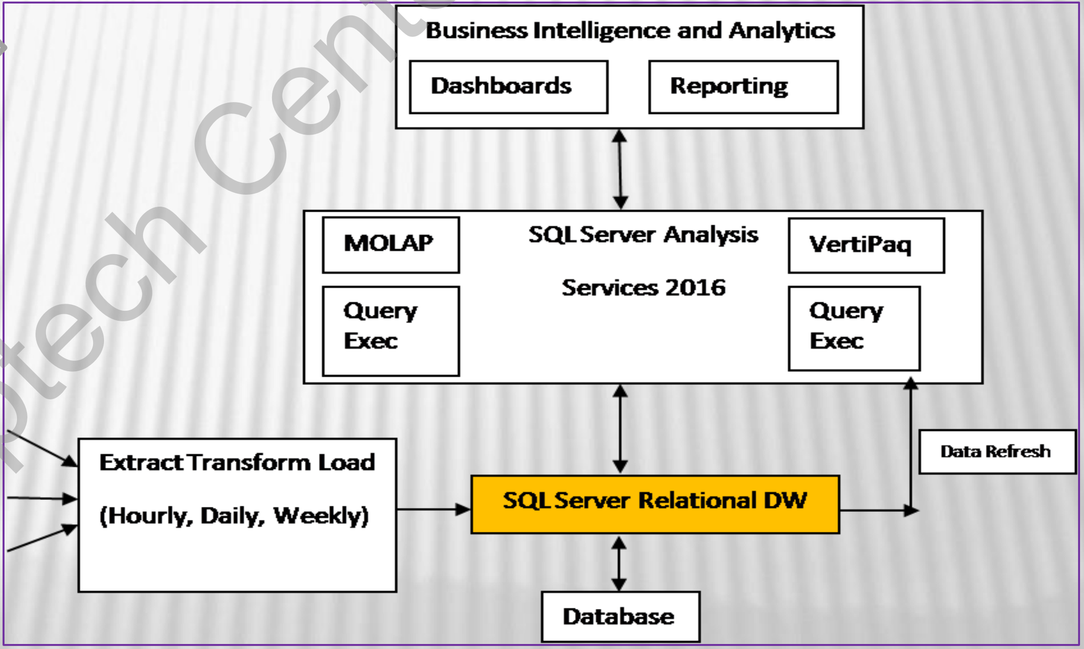
- Enhancements to AlwaysOn
- Enhancements to In-Memory OLTP
- Enhancements in T-SQL
- Row level Security Enhancements

Conventional Methods of Data Modeling 1-2

The conventional methods of data modeling are as follows:

- Multidimensional Online Analytical Processing (MOLAP) with pre-collected data
- Tabular model with data in the database with regular Extract, Transform, and Load (ETL)

A typical deployment wherein SQL Server is used for relational Data Warehouse and SQL Server Analysis Services (SSAS) for data modeling


Business Intelligence and Analytics
Dashboards
Reporting
MOLAP
SQL Server Analysis Services 2016
VertiPaq
Query Exec
Query Exec
Data Refresh
Extract Transform Load (Hourly, Daily, Weekly)
SQL Server Relational DW
Database

Conventional Methods of Data Modeling 2-2

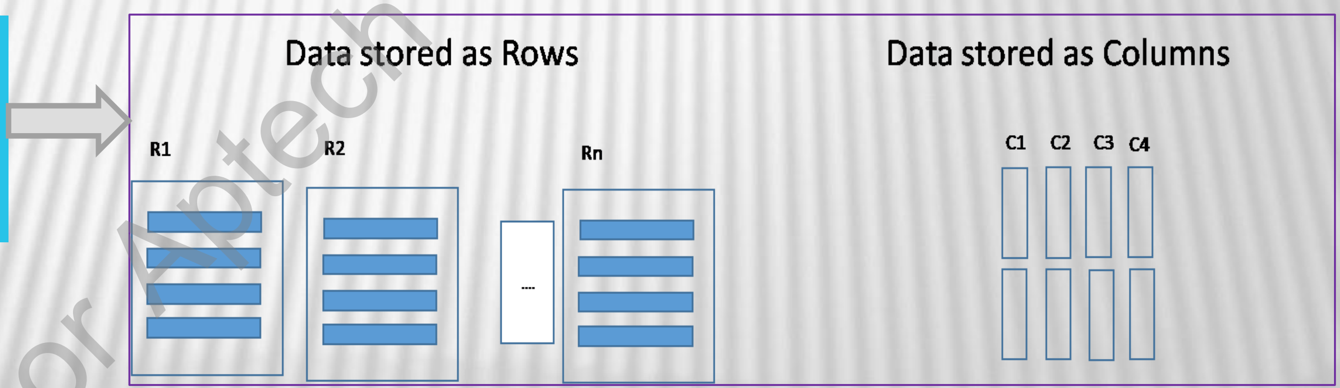
SQL Server 2016

Following are some challenges in this method:

- **Cube Processing**: Requires Periodic refreshing and aggregation of data
- **Data Latency**: Latest data fetched is from previous refresh cycle and is highly unacceptable
- **Very Large Database Support**: Loading data to separate Business Intelligence (BI) servers may not always be a feasible solution

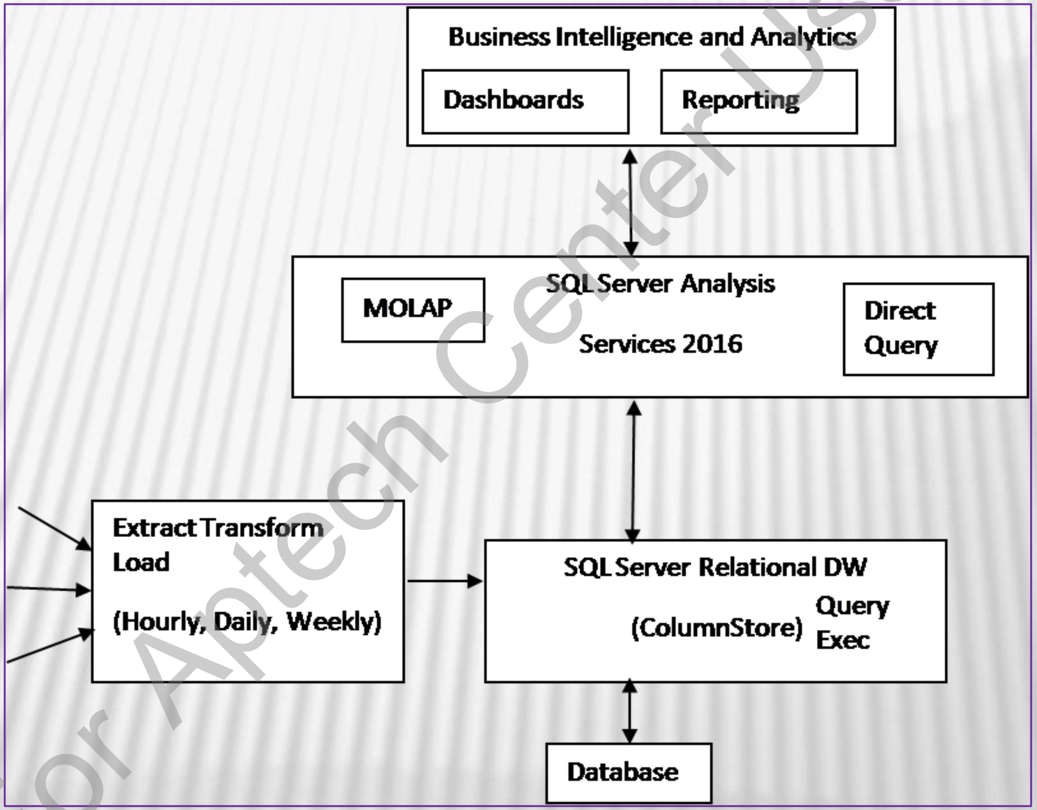
Clustered Columnstore Index (CCI) addresses some of these drawbacks.

Storage of Data as Rows and as Columns



Recommended Configuration

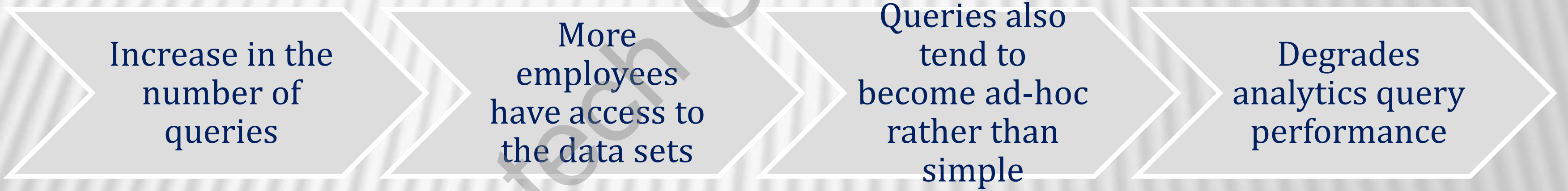
Columnstore indexes eliminate the need to pre-collect data for report queries. Recommended configuration is shown in the following figure:



SQL Server 2016

Real Time Operational Analytics 1-2

- Data and analytics help businesses to take decisions more efficiently.
- The evolution of social media and Internet of Things (IOT) has contributed to enormous growth of data.
- Power Pivot is one of the tools used to access such huge data sets.
- Power Pivot is not adequate for some of the needs.

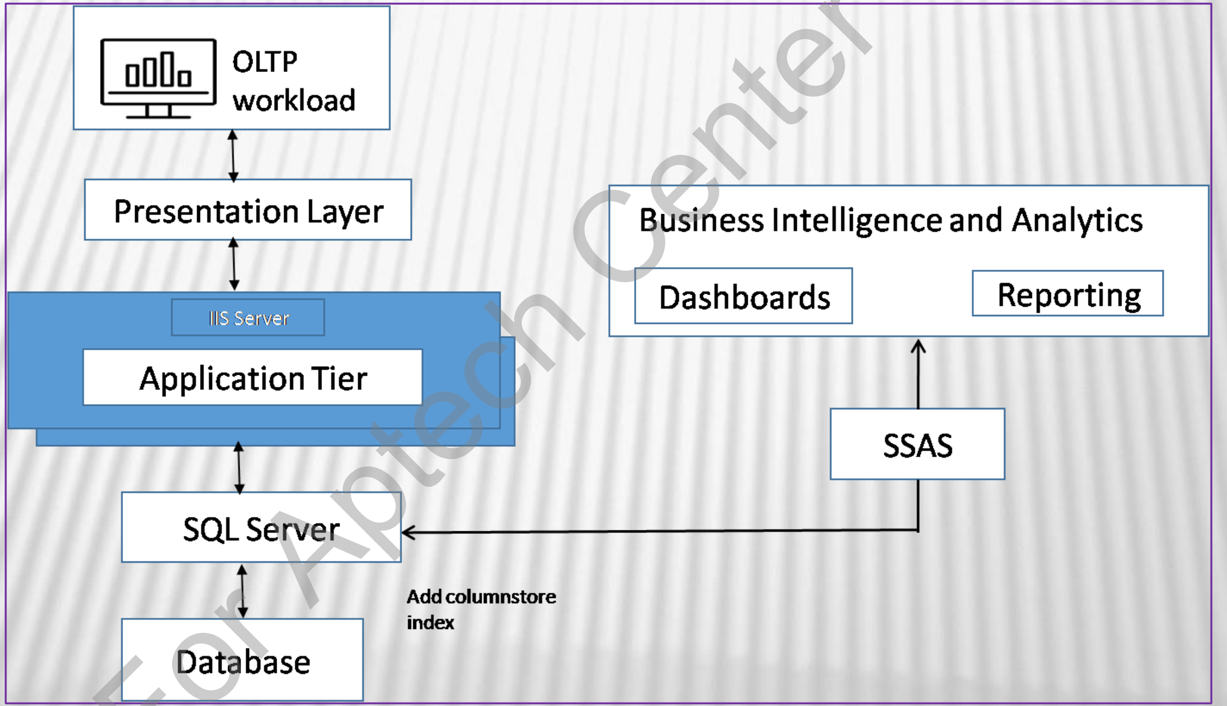

Increase in the number of queries
More employees have access to the data sets
Queries also tend to become ad-hoc rather than simple
Degrades analytics query performance

SQL Server 2016 takes care of these challenges with the improvements to In-Memory analytics

SQL Server 2016

Real Time Operational Analytics 2-2

Real Time Operational Analytics enables users to run analytics queries directly on operational workload using columnstore indexes. One possible configuration to run Analytics queries is:



Steps to Perform Real Time Operational Analytics

SQL Server 2016

Identify the table(s) and the columns required to run analytics

↓

Create columnstore indexes for those tables

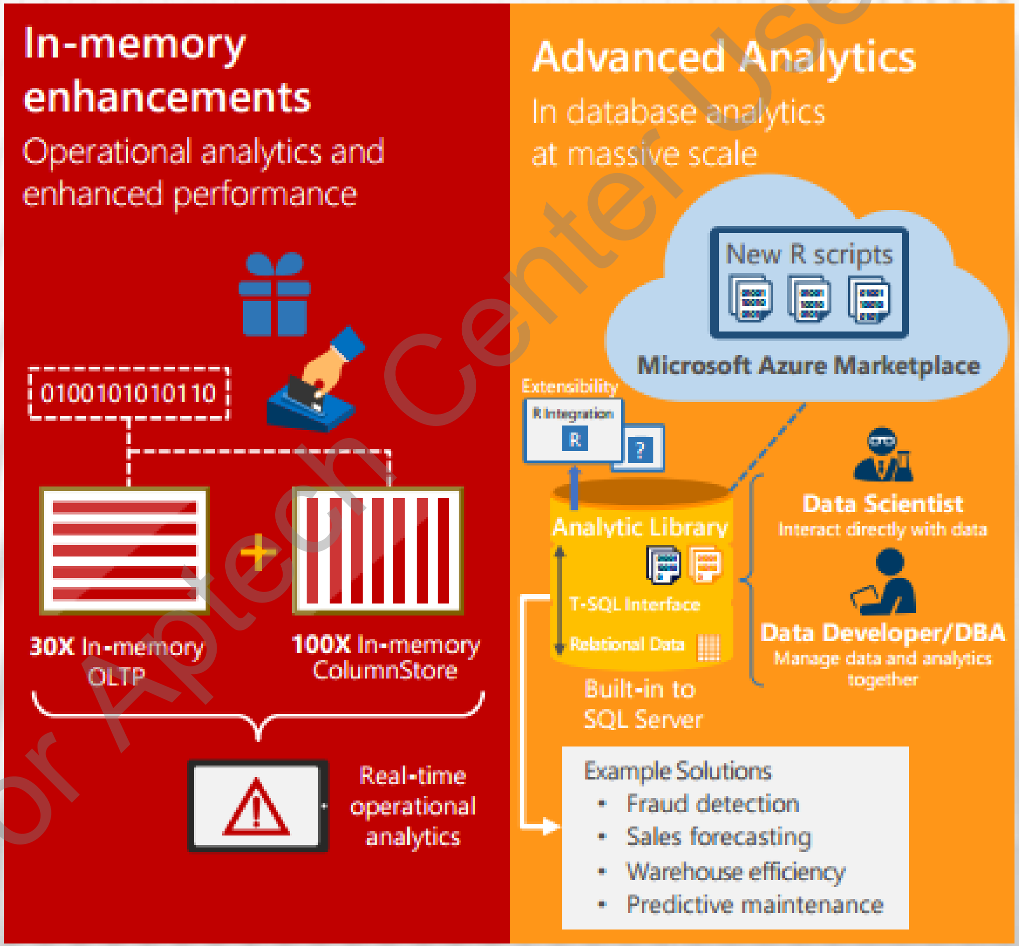
↓

Set up Analytics framework to fetch data directly from Operational Workload

In-memory Enhancements and Advanced Analytics

SQL Server 2016

Figure shows the representation of In-memory enhancements and Advanced Analytics.



NonClustered Columnstore Index

SQL Server 2016

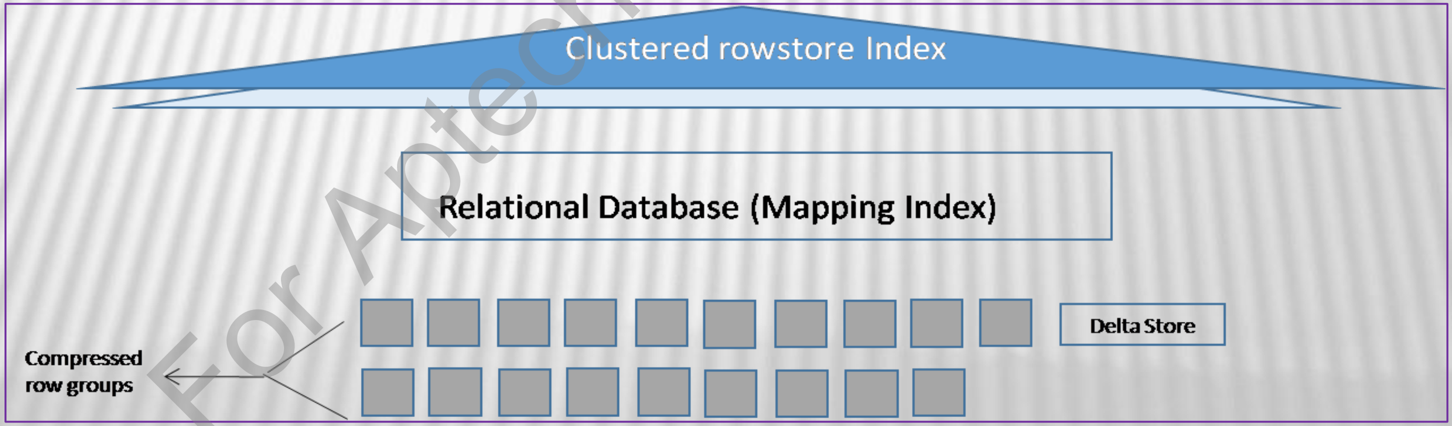
Standard configuration requires:

- Separate Data Warehouse for Data analytics
- Data collected from multiple sources

Hence, SQL Server 2016 provides:

- Updateable NonClustered Columnstore Index (NCCI)
  - Real time analytics on operational schema
  - Existing OLTP need not be changed
  - Single NCCI can replace other OLTP workload indexes

Following figure shows a table with NCCI:



Challenges with NCCI Configuration and Solution

Challenges	• Performance of analytics queries • Normalized schema for OLTP databases • Normalization might involve complex joins between many tables • Using the same OLTP database might lead to very poor performance
Solution	• Use of NCCI which can execute many analytics queries in few seconds • A dedicated Data Warehouse could improve the performance of analytics query
Reduce impact on operational workload	• Filtered Columnstore Index • Compression Delay • Offloading Analytics Queries to Readable Secondary

Overall, it is a trade-off between real time operational analytics and its impact on operational workload

SQL Server 2016

Native JSON Support

JavaScript Object Notation (JSON)

- A subset of JavaScript programming language
- Language-independent, lightweight
- Provides a human readable text data exchange format

SQL Server 2016

- Native support for JSON data
- Supports JSON data with its NVARCHAR data type
- The NVARCHAR data type provides more flexibility

Developers can concentrate on application logic than having to write functions to parse JSON data

Exporting Tabular Data as JSON Data 1-5

SQL Server 2016

Following are the clauses to export tabular data as JSON data:

- The FOR JSON [AUTO|PATH] clause

  Use this clause in SQL queries to fetch data in JSON format

- The FOR JSON AUTO clause

  SQL Server 2016 automatically formats the nested JSON sub arrays of query result

Exporting Tabular Data as JSON Data 2-5

SQL Server 2016

Example 1:

Using FOR JSON AUTO Clause:

- Create table Employees
- Insert some data

```
CREATE TABLE
Employees (
 ID INT IDENTITY (1,1)
NOT NULL,
FirstName VARCHAR
(255),
LastName VARCHAR
(255),
 Grade INT,
 Age DECIMAL (3,1)
 )
```

Table is created

```
INSERT INTO Employees
(FirstName, LastName, Grade, Age)
 SELECT 'Mark', 'Thomas', 2, 45
 UNION ALL
 SELECT 'Salmon', 'John', 1, 56
 UNION ALL
 SELECT 'Kirsten', 'Powell', 3, 28
```

Rows are inserted

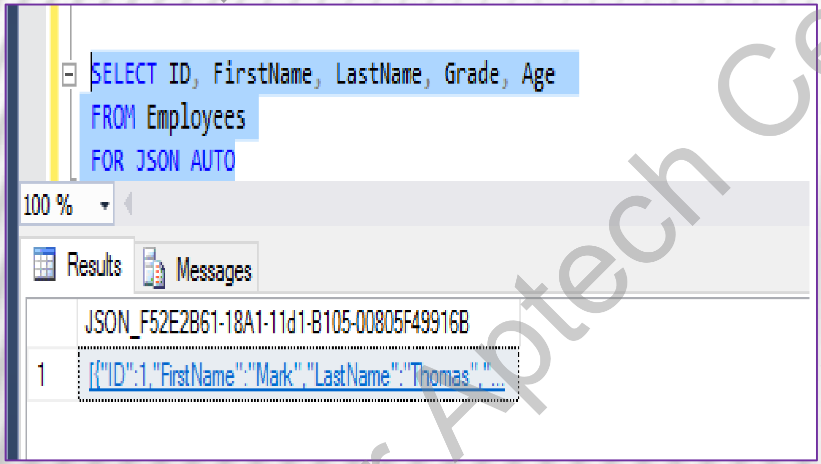
Exporting Tabular Data as JSON Data 3-5

SQL Server 2016

Example 1 (Continued):

Using FOR JSON AUTO Clause:

- Create SQL Query to retrieve table data using FOR JSON AUTO clause
- The Query and its outcome in the SSMS



```
SELECT ID, FirstName, LastName, Grade, Age
FROM Employees
FOR JSON AUTO
```

```
   JSON_F52E2B61-18A1-11d1-B105-00805F49916B
1  [{"ID":1,"FirstName":"Mark","LastName":"Thomas","...
```

With the FOR JSON AUTO clause, SQL Server automatically formats the JSON output based on the query structure

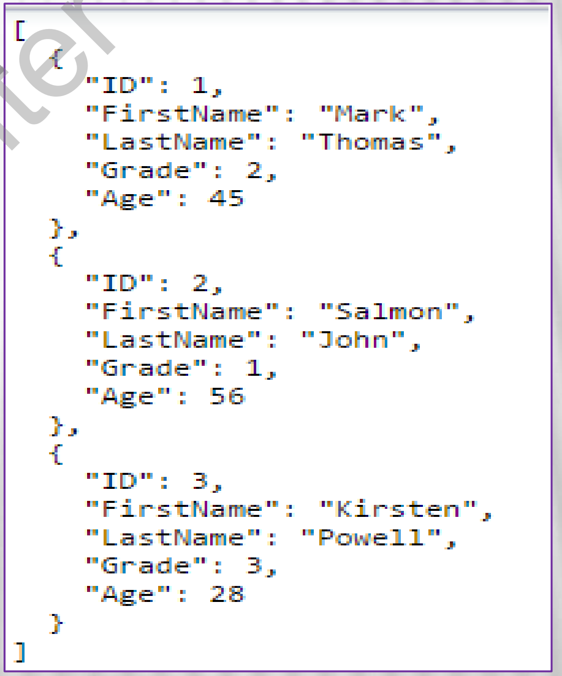
Exporting Tabular Data as JSON Data 4-5

SQL Server 2016

Example 1 (Continued):

Using FOR JSON AUTO Clause:

- Copy the JSON data to a text editor such as Notepad and format it
- The output is viewed as



```
[
  {
    "ID": 1,
    "FirstName": "Mark",
    "LastName": "Thomas",
    "Grade": 2,
    "Age": 45
  },
  {
    "ID": 2,
    "FirstName": "Salmon",
    "LastName": "John",
    "Grade": 1,
    "Age": 56
  },
  {
    "ID": 3,
    "FirstName": "Kirsten",
    "LastName": "Powell",
    "Grade": 3,
    "Age": 28
  }
]
```

Exporting Tabular Data as JSON Data 5-5

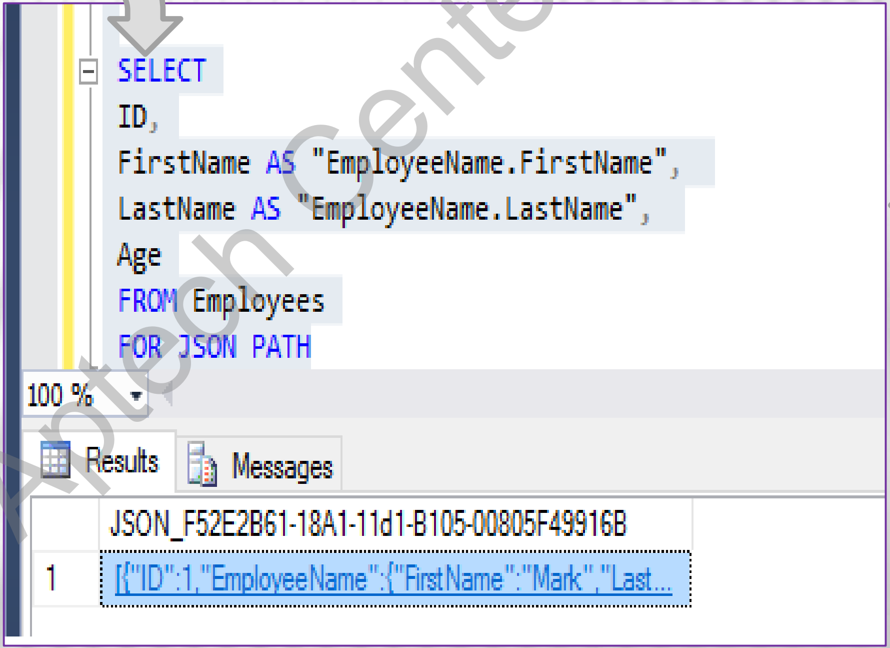
SQL Server 2016

Example 2:

Using FOR JSON PATH Clause:

- The SQL Query and its JSON output in the SSMS for the Employees table that is already created

In this SELECT query, SQL Server obtains the result of the query and formats it in the output as a JSON document.



```
SELECT
ID,
FirstName AS "EmployeeName.FirstName",
LastName AS "EmployeeName.LastName",
Age
FROM Employees
FOR JSON PATH
```

```
   JSON_F52E2B61-18A1-11d1-B105-00805F49916B
1  [{"ID":1,"EmployeeName":{"FirstName":"Mark","Last...
```

SQL Server 2016

AlwaysOn

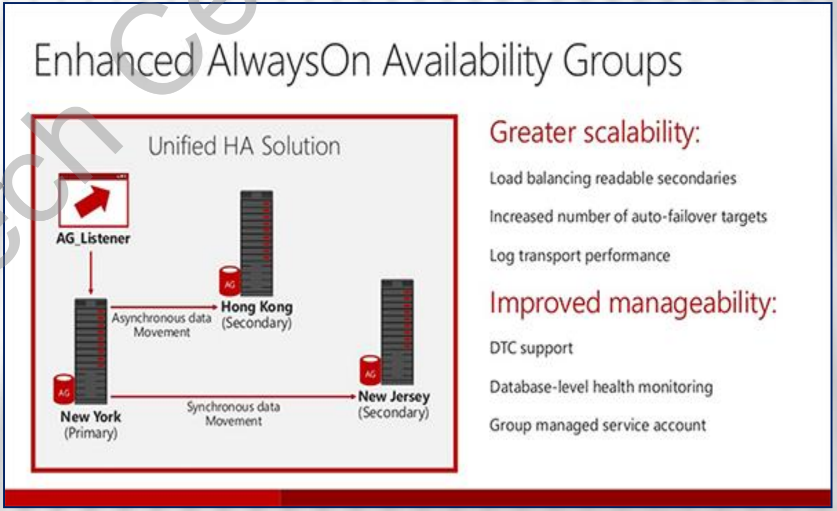
AlwaysOn is a feature that was first introduced in SQL Server 2012

- A substitute for data mirroring
- High-availability and disaster recovery solutions for mission-critical applications
- Requires a WSFC cluster for deployment

Major enhancements to AlwaysOn in SQL Server 2016 are:

- Scalability: Achieved through load balancing readable secondaries
- Manageability: Availability Group health monitors even the database health

Pictorial representation of Enhanced AlwaysOn Availability Groups


Enhanced AlwaysOn Availability Groups

Greater scalability:

Load balancing readable secondaries

Increased number of auto-failover targets

Log transport performance

Improved manageability:

DTC support

Database-level health monitoring

Group managed service account

Availability Group

An Availability group

Supports failover environment for primary and secondary databases

An availability replica

Hosts each set of Availability database.

Two types

Primary replica for the primary databases

One to eight secondary replicas for a set of secondary databases

Availability Databases

The database:

- Should be Online, read-write database
- Reside on the server instance
- Will be the primary database
- Can be accessed by clients

Secondary databases are created, after restoring the backups of the new primary database

Availability Replicas

Availability Replicas are:

- Set of two or more failover patterns
- Defined by each availability group
- Hosted for each availability group

A role is assigned for each Availability replica

Availability Modes

An availability replica has a property called availability mode. Following are the two availability modes:

Asynchronous-commit mode

- Reduced transaction latency on secondary databases
- Secondary databases are not in sync with primary database
- Chances of some data loss
- Asynchronous-commit replica uses this mode

Synchronous-commit mode

- Completely protected
- Secondary and primary databases are always in sync
- Increase in transaction latency
- Synchronous-commit replica uses this mode

Types of Failover

Three types: **Automatic**, **Manual**, and **Forced**

The availability mode of the secondary replica determines which failover type it supports from the following given options:

The synchronous-commit mode	**Planned manual failover (without data loss)**	**Automatic failover (without data loss)**	**The asynchronous-commit mode**
Supports two failover types: • Automatic failover • Planned manual failover	• Issued by the database administrator • Causes the secondary replica to change to primary replica and vice versa	• Any failure causing the secondary replica to change to primary replica • Changes to secondary replica once the original primary replica is available	• Supports only forced manual failover • There might be data loss

Active Secondary Replicas

Active secondary replicas are supported by AlwaysOn availability groups. The functionalities provided by Active secondary replicas are:

- Backup operations on secondary replicas
- Readable secondary replicas

Session-Timeout Period

The session-timeout property ensures that no two replicas wait for each other indefinitely

- Default value } 10 seconds
- Minimum value} 5 seconds
- Recommended value} 10 seconds or more

Automatic Page Repair

- The Availability replica takes care of automatic page repair of corrupted pages residing on a local database
- A fresh copy of the page by the primary replica is taken, if the secondary replica fails to read the corrupted page
- The first response by all secondary replicas is taken, if the primary replica fails to read the corrupted page
- The corrupt page is replaced with the fresh copy provided the request succeeds

Enhanced Features in SQL Server 2016

SQL Server 2016

- Enhanced AlwaysOn
- Enhanced Online Operations
- Scalability
- Hardware Acceleration for TDE Encryption/Decryption
- Buffer Pool Extension
- In-Memory OLTP (Hekaton) and Optimization Enhancements

  Key changes to In-Memory OLTP are as follows:
  - Enhanced In-Memory performance
  - Maximum memory for memory-optimized tables
  - Collation
  - Schema and data changes
  - More Transact-SQL features supported
  - Parallel Plans
  - Transparent data encryption
  - AlwaysOn
  - Number of sockets

Summary

SQL Server 2016

- New powerful features
  - Real-time operational analytics
  - Native JSON support
  - Stretch database
  - Always Encrypted
- Enhancements to existing features
  - AlwaysOn
  - In-Memory OLTP (Hekaton)
  - Optimization enhancements
- Real-time Operational Analytics: users can run analytics queries directly on operational workload
- Native support for JSON