

# What's New in Database Engine

SQL Server 2016 [Other Versions](#) ▼

Published: May 16, 2016

Updated: September 14, 2016

Applies To: SQL Server 2016



Need help? [MSDN forum](#) | [stackoverflow](#) | Log an issue or suggestion at [Microsoft Connect](#)


This topic summarizes the enhancements introduced in the SQL Server 2016 release of the SQL Server Database Engine. The new features and enhancements increase the power and productivity of architects, developers, and administrators who design, develop, and maintain data storage systems.

To review what is new in the other SQL Server components, see [What's New in SQL Server 2016](#).

## Note

SQL Server 2016 is a 64-bit application. 32-bit installation is discontinued, though some elements run as 32-bit components.

## Try it out

- To download SQL Server 2016, go to [Evaluation Center](#) .
- Have an Azure account? Then go [Here](#) to spin up a Virtual Machine with SQL Server 2016 already installed.



For the current release notes, see [SQL Server 2016 Release Notes](#).

## Database Engine Feature Enhancements

This section contains the following subsections:

- [Columnstore Indexes](#)
- [Database Scoped Configurations](#)
- [In-Memory OLTP](#)
- [Query Optimizer](#)
- [Live Query Statistics](#)
- [Query Store](#)
- [Temporal Tables](#)
- [Striped Backups to Microsoft Azure Blob Storage](#)
- [File-Snapshot Backups to Microsoft Azure Blob Storage](#)
- [Managed Backup](#)

- [TempDB Database](#)
- [Built-in JSON Support](#)
- [PolyBase](#)
- [Stretch Database](#)
- [Support for UTF-8](#)
- [New Default Database Size and Autogrow Values](#)

## Columnstore Indexes

This release offers improvements for columnstore indexes including updateable nonclustered columnstore indexes, columnstore indexes on in-memory tables, and many more new features for operational analytics.

- A read-only nonclustered columnstore index is updateable after upgrade. A rebuild of the index is not required to make it updateable.
- There are performance improvements for analytics queries on columnstore indexes, especially for aggregates and string predicates.
- DMVs and XEvents have supportability improvements.

For more details, see these topics in the [Columnstore Indexes Guide](#) section of Books Online:

- [Columnstore Indexes Versioned Feature Summary](#) – includes what's new.
- [Columnstore Indexes Data Loading](#)
- [Columnstore Indexes Query Performance](#)
- [Get started with Columnstore for real time operational analytics](#)
- [Columnstore Indexes for Data Warehousing](#)
- [Columnstore Indexes Defragmentation](#)

## Database Scoped Configurations

The new [ALTER DATABASE SCOPED CONFIGURATION \(Transact-SQL\)](#) statement gives you control of certain configurations for your particular database. The configuration settings affect application behavior.

The new statement is available in both SQL Server 2016 and SQL Database V12.

## In-Memory OLTP

### Storage format change

The storage format for memory-optimized tables is changed between SQL Server 2014 and 2016. For upgrade and attach/restore from SQL Server 2014, the new storage format is serialized and the database is restarted once during database recovery.

- [Upgrade to SQL Server 2016](#)

### ALTER TABLE is log-optimized, and runs in parallel

Now when you execute an ALTER TABLE statement on a memory-optimized table, only the metadata changes are written to the log. This greatly reduces log IO. Also, most ALTER TABLE scenarios now run in parallel, which can greatly shorten the duration of the statement.

- For non-parallel exceptions, including LOBs, see [Altering Memory-Optimized Tables](#).

## Statistics

[Statistics for memory-optimized](#) tables are now updated automatically. In addition, sampling is now a supported method to collect statistics, allowing you to avoid the more expensive fullscan method.

## Parallel and heap scan for memory-optimized tables

Memory-optimized tables, and indexes on memory-optimized tables, now support parallel scan. This improves the performance of analytical queries.

In addition, heap scan is supported, and can be performed in parallel. In the case of a memory-optimized table, a heap scan refers to scanning all the rows in a table using the in-memory heap data structure used for storing the rows. For a full table scan, heap scan is more efficient than using an index.

## Transact-SQL Improvements for memory-optimized tables

There are several Transact-SQL elements that were not supported for memory-optimized tables in SQL Server 2014, which are now supported in SQL Server 2016:

- UNIQUE constraints and indexes are supported.
- FOREIGN KEY references between memory-optimized tables are supported.
  - These foreign keys can reference only a primary key, and cannot reference a unique key.
- CHECK constraints are supported.
- A non-unique index can allow NULL values in its key.
- TRIGGERS are supported on memory-optimized tables.
  - Only AFTER triggers are supported. INSTEADOF triggers are not supported.
  - Any trigger on a memory-optimized table must use WITH NATIVE\_COMPILATION.
- Full support for all SQL Server code pages and collations with indexes and other artifacts in memory-optimized tables and natively compiled T-SQL modules.
- Support for [Altering Memory-Optimized Tables](#):
  - ADD and DROP indexes. Change bucket\_count of hash indexes.
  - Make schema changes: add/drop/alter columns; add/drop constraint.
- A memory-optimized table can now have several columns whose combined lengths are longer than the length of the 8060 byte page. An example is a table that has three columns of type `nvarchar(4000)`. In such examples, some columns are now stored off-row. Your queries are blissfully unaware of whether a column is on-row or off-row.
- [LOB \(large object\) types](#) `varbinary(max)`, `nvarchar(max)`, and `varchar(max)` are now supported in memory-optimized tables.

For overall information, see:

- [Transact-SQL Constructs Not Supported by In-Memory OLTP](#)
- [\[Unsupported SQL Server Features for In-Memory OLTP\]\(Unsupported%20SQL%20Server%20Features%20for%20In-Memory OLTP.md\)](#)

## Transact-SQL Improvements for natively compiled modules

There are some Transact-SQL elements that were not supported for natively compiled modules in SQL Server 2014, which are now supported in SQL Server 2016:

- Query constructs:
  - UNION and UNION ALL
  - SELECT DISTINCT
  - OUTER JOIN
  - Subqueries in SELECT
- INSERT, UPDATE and DELETE statements can now include the [OUTPUT clause](#).
- LOBs can now be used in the following ways in a native proc:
  - Declaration of variables.
  - Input parameters received.
  - Parameters passed into string functions, such as into LTrim or Substring, in a native proc.
- Inline (meaning single statement) table-valued functions (TVFs) can now be natively compiled.
- Scalar user-defined functions (UDFs) can now be natively compiled.
- Increased support for a native proc to call:
  - Built-in [security functions](#).
  - Built-in [math functions](#).
  - Built-in function @@SPID.
- EXECUTE AS CALLER is now support, which means the EXECUTE AS clause is no longer required when creating a natively compiled T-SQL module.

For overall information, see:

- [Supported Features for Natively Compiled T-SQL Modules](#)
- [Altering Natively Compiled T-SQL Modules](#)

## Performance and scaling improvements

- There is no longer any limitation on data size. See [Estimate Memory Requirements for Memory-Optimized Tables] (Estimate%20Memory%20Requirements%20for%20Memory-Optimized Tables.md).
- There are now multiple concurrent threads responsible to [persist to disk the changes to memory-optimized tables](#).
- Parallel plan support for [Accessing Memory-Optimized Tables Using Interpreted Transact-SQL](#).

## Enhancements in SQL Server Management Studio

- The [Determining if a Table or Stored Procedure Should Be Ported to In-Memory OLTP](#) no longer requires the configuration of data collectors or management data warehouse. The report can now run directly on a production database.
- [PowerShell Cmdlet for Migration Evaluation](#) for evaluating the migration fitness of multiple objects in a SQL Server database.
- Generate migration checklists by right-clicking on a database, and selecting Tasks > Generate In-Memory OLTP migration checklists.

## Cross-feature support

- Support for using temporal system-versioning with In-Memory OLTP. For more information, see [System-Versioned Temporal Tables with Memory-Optimized Tables](#)
- Query store support for natively compiled code from In-Memory OLTP workloads. For more information, see [Using the Query Store with In-Memory OLTP](#).
- [Row-Level Security in Memory-Optimized Tables](#)
- [Using Multiple Active Result Sets \(MARS\)](#) connections can now access memory-optimized tables and natively compiled stored procedures.
- [Transparent Data Encryption \(TDE\)](#) support. If a database is configured for ENCRYPTION, files in the [The Memory Optimized Filegroup](#) are now also encrypted.

For more information, see [In-Memory OLTP \(In-Memory Optimization\)](#).

## Query Optimizer

### Compatibility Level Guarantees

When you upgrade your database to SQL Server 2016, there will be no plan changes seen if you remain at the older compatibility levels that you were using (for example, 120 or 110). New features and improvements related to query optimizer, will be available only under latest compatibility level.

### Trace Flag 4199

In general, you do not need to use trace flag 4199 in SQL Server 2016 since most of the query optimizer behaviors controlled by this trace flag are enabled unconditionally under the latest compatibility level (130) in SQL Server 2016.

### New Referential Integrity Operator

A table can reference a maximum of 253 other tables and columns as foreign keys (outgoing references). SQL Server 2016 increases the limit for the number of other table and columns that can reference columns in a single table (incoming references), from 253 to 10,000. For restrictions, see [Create Foreign Key Relationships](#). A new referential integrity operator is introduced (under compatibility level 130), which performs the referential integrity checks in place. This improves overall performance for UPDATE and DELETE operations, on tables that have a large number of incoming references, thereby making it feasible to have large number of incoming references. For more information, see [Query Optimizer Additions in SQL Server 2016](#)

### Parallel update of sampled statistics

Data sampling to build statistics is now done in parallel (under compatibility level 130), to improve the performance of statistics collection. For more information, see [Update Statistics](#).

### Sublinear threshold for update of statistics

Automatic update of statistics is now more aggressive on large tables (under compatibility level 130). The threshold to trigger auto-update of statistics is 20%, starting SQL Server 2016, for larger tables, this threshold will start decreasing (still a percentage) as the number of rows increase in the table. You will no longer need to set trace flag 2371 to reduce the threshold.

### Other enhancements

The Insert in an Insert-select statement is multi-threaded or can have a parallel plan (under compatibility level 130). To get a parallel plan, INSERT ... SELECT statement must use the TABLOCK hint. For more information, see [Parallel Insert Select](#)

### Live Query Statistics

Management Studio provides the ability to view the live execution plan of an active query. This live query plan provides real-time insights into the query execution process as the controls flow from one query plan operator to another. For more information, see [Live Query Statistics](#).

## Query Store

Query store is a new feature that provides DBAs with insight on query plan choice and performance. It simplifies performance troubleshooting by enabling you to quickly find performance differences caused by changes in query plans. The feature automatically captures a history of queries, plans, and runtime statistics, and retains these for your review. It separates data by time windows, allowing you to see database usage patterns and understand when query plan changes happened on the server. The query store presents information by using a Management Studio dialog box, and lets you force the query to one of the selected query plans. For more information, see [Monitoring Performance By Using the Query Store](#).

## Temporal Tables

SQL Server 2016 now supports system-versioned temporal tables. A temporal table is a new type of table that provides correct information about stored facts at any point in time. Each temporal table consists of two tables actually, one for the current data and one for the historical data. The system ensures that when the data changes in the table with the current data the previous values are stored in the historical table. Querying constructs are provided to hide this complexity from users. For more information, see [Temporal Tables](#).

## Striped Backups to Microsoft Azure Blob Storage

In SQL Server 2016, SQL Server backup to URL using the Microsoft Azure Blob storage service now supports striped backups sets using block blobs to support a maximum backup size of 12.8 TB. For examples, see [Code Examples](#).

## File-Snapshot Backups to Microsoft Azure Blob Storage

In SQL Server 2016, SQL Server backup to URL now supports using Azure snapshots to backup databases in which all database files are stored using the Microsoft Azure Blob storage service. For more information, see [File-Snapshot Backups for Database Files in Azure](#).

## Managed Backup

In SQL Server 2016 SQL Server Managed Backup to Microsoft Azure uses the new block blob storage for backup files. There are also several changes and enhancements to Managed Backup.

- Support for both automated and custom scheduling of backups.
- Support backups for system databases.
- Support for databases that are using the Simple recovery model.

For more information, see [SQL Server Managed Backup to Microsoft Azure](#)

### Note

For SQL Server 2016, these new managed backup features do not yet have corresponding UI support in SQL Server Management Studio.

## TempDB Database

There are several enhancements to TempDB:

- Trace Flags 1117 and 1118 are not required for tempdb anymore. If there are multiple tempdb database files all files will grow at the same time depending on growth settings. In addition, all allocations in tempdb will use uniform

extents.

- By default, setup adds as many tempdb files as the CPU count or 8, whichever is lower.
- During setup, you can configure the number of tempdb database files, initial size, autogrowth and directory placement using the new UI input control on the Database Engine Configuration - TempDB section of SQL Server Installation Wizard.
- The default initial size is 8MB and the default autogrowth is 64MB.
- You can specify multiple volumes for tempdb database files. If multiple directories are specified tempdb data files will be spread across the directories in a round-robin fashion.

## Built-in JSON Support

SQL Server 2016 adds built-in support for importing and exporting JSON and working with JSON strings. This built-in support includes the following statements and functions.

- Format query results as JSON, or export JSON, by adding the **FOR JSON** clause to a **SELECT** statement. Use the **FOR JSON** clause, for example, to delegate the formatting of JSON output from your client applications to SQL Server. For more info, see [Format Query Results as JSON with FOR JSON \(SQL Server\)](#).
- Convert JSON data to rows and columns, or import JSON, by calling the **OPENJSON** rowset provider function. Use **OPENJSON** to import JSON data into SQL Server, or convert JSON data to rows and columns for an app or service that can't currently consume JSON directly. For more info, see [Convert JSON Data to Rows and Columns with OPENJSON \(SQL Server\)](#).
- The **ISJSON** function tests whether a string contains valid JSON. For more info, see [ISJSON \(Transact-SQL\)](#)
- The **JSON\_VALUE** function extracts a scalar value from a JSON string. For more info, see [JSON\\_VALUE \(Transact-SQL\)](#).
- The **JSON\_QUERY** function extracts an object or an array from a JSON string. For more info, see [JSON\\_QUERY \(Transact-SQL\)](#).
- The **JSON\_MODIFY** function updates the value of a property in a JSON string and return the updated JSON string. For more info, see [JSON\\_MODIFY \(Transact-SQL\)](#).

## PolyBase

PolyBase allows you to use T-SQL statements to access data stored in Hadoop or Azure Blob Storage and query it in an adhoc fashion. It also lets you query semi-structured data and join the results with relational data sets stored in SQL Server. PolyBase is optimized for data warehousing workloads and intended for analytical query scenarios.

For more information, see [PolyBase Guide](#).

## Stretch Database

Stretch Database is a new feature in SQL Server 2016 that migrates your historical data transparently and securely to the Microsoft Azure cloud. You can access your SQL Server data seamlessly regardless of whether it's on-premises or stretched to the cloud. You set the policy that determines where data is stored, and SQL Server handles the data movement in the background. The entire table is always online and queryable. And, Stretch Database doesn't require any changes to existing queries or applications – the location of the data is completely transparent to the application. For more info, see [Stretch Database](#).

## Support for UTF-8

[bcp Utility](#), [BULK INSERT](#), and [OPENROWSET](#) now support the UTF-8 code page. For more information, see those topics and [Create a Format File \(SQL Server\)](#).

## New Default Database Size and Autogrow Values

New values for the model database and default values for new databases (which are based on model). The initial size of the data and log files is now 8 MB. The default auto-growth of data and log files is now 64MB.

## Transact-SQL Enhancements

Numerous enhancements support the features described in the other sections of this topic. The following additional enhancements are available.

- The TRUNCATE TABLE statement now permits the truncation of specified partitions. For more information, see [TRUNCATE TABLE \(Transact-SQL\)](#).
- [ALTER TABLE \(Transact-SQL\)](#) now allows many alter column actions to be performed while the table remains available.
- The full-text index DMV [sys.dm\\_fts\\_index\\_keywords\\_position\\_by\\_document \(Transact-SQL\)](#) returns the location of keywords in documents. This DMV has also been added in SQL Server 2012 SP2 and SQL Server 2014 SP1.
- A new query hint **NO\_PERFORMANCE\_SPOOL** can prevent a spool operator from being added to query plans. This can improve performance when many concurrent queries are running with spool operations. For more information, see [Query Hints \(Transact-SQL\)](#).
- The [FORMATMESSAGE \(Transact-SQL\)](#) statement is enhanced to accept a msg\_string argument.- The maximum index key size for NONCLUSTERED indexes has been increased to 1700 bytes.
- New DROP IF syntax is added for drop statements related to AGGREGATE, ASSEMBLY, COLUMN, CONSTRAINT, DATABASE, DEFAULT, FUNCTION, INDEX, PROCEDURE, ROLE, RULE, SCHEMA, SECURITY POLICY, SEQUENCE, SYNONYM, TABLE, TRIGGER, TYPE, USER, and VIEW. See individual syntax topics for syntax.
- A MAXDOP option has been added to [DBCC CHECKTABLE \(Transact-SQL\)](#), [DBCC CHECKDB \(Transact-SQL\)](#), and [DBCC CHECKFILEGROUP \(Transact-SQL\)](#) to specify the degree of parallelism.
- SESSION\_CONTEXT can now be set. Includes the [SESSION\\_CONTEXT \(Transact-SQL\)](#) function, [CURRENT\\_TRANSACTION\\_ID \(Transact-SQL\)](#) function, and the [sp\\_set\\_session\\_context \(Transact-SQL\)](#) procedure.
- Advanced Analytics Extensions allow users to execute scripts written in a supported language such as R. Transact-SQL supports R by introducing the [sp\\_execute\\_external\\_script \(Transact-SQL\)](#) stored procedure, and the [external scripts enabled Server Configuration Option](#). For more information, see [SQL Server R Services](#).
- Also to support R, the ability to create an external resource pool. For more information, see [CREATE EXTERNAL RESOURCE POOL \(Transact-SQL\)](#). New catalog views and DMVs ([sys.resource\\_governor\\_external\\_resource\\_pools \(Transact-SQL\)](#) and [sys.dm\\_resource\\_governor\\_external\\_resource\\_pool\\_affinity \(Transact-SQL\)](#)). Additional arguments are available for [sp\\_execute\\_external\\_script \(Transact-SQL\)](#) and [CREATE WORKLOAD GROUP \(Transact-SQL\)](#). Additional columns are added to some of the existing resource governor catalog views and DMVs.
- The [CREATE USER](#) syntax is enhanced with the ALLOW\_ENCRYPTED\_VALUE\_MODIFICATIONS option to support the Always Encrypted feature. For more information see [Migrate Sensitive Data Protected by Always Encrypted](#).
- The [COMPRESS \(Transact-SQL\)](#) and [DECOMPRESS \(Transact-SQL\)](#) functions convert values into and out of the GZIP algorithm.
- The [DATEDIFF\\_BIG \(Transact-SQL\)](#) and [AT TIME ZONE \(Transact-SQL\)](#) functions and the [sys.time\\_zone\\_info \(Transact-SQL\)](#) view are added to support date and time interactions.
- A credential can now be created at the database level (in addition to the server level credential that was previously available). For more information, see [CREATE DATABASE SCOPED CREDENTIAL \(Transact-SQL\)](#).
- Eight new properties are added to [SERVERPROPERTY \(Transact-SQL\)](#): InstanceDefaultDataPath, InstanceDefaultLogPath, ProductBuild, ProductBuildType, ProductMajorVersion, ProductMinorVersion, ProductUpdateLevel, and ProductUpdateReference.
- The input length limit of 8,000 bytes for the [HASHBYTES \(Transact-SQL\)](#) function is removed.
- New string functions [STRING\\_SPLIT \(Transact-SQL\)](#) and [STRING\\_ESCAPE \(Transact-SQL\)](#) are added.
- Autogrow options: Trace flag 1117 is replaced by the AUTOGROW\_SINGLE\_FILE and AUTOGROW\_ALL\_FILES option of ALTER DATABASE, and trace flag 1117 has no affect. For more information, see [ALTER DATABASE File and Filegroup](#)



[Options \(Transact-SQL\)](#) and the new `is_autogrow_all_files` column of [sys.filegroups \(Transact-SQL\)](#).

- Allocation of mixed extents: For user databases, default allocation for the first 8 pages of an object will change from using mixed page extents to using uniform extents. Trace flag 1118 is replaced with the SET `MIXED_PAGE_ALLOCATION` option of ALTER DATABASE, and trace flag 1118 has no affect. For more information, see [ALTER DATABASE SET Options \(Transact-SQL\)](#), and the new `is_mixed_page_allocation_on` column of [sys.databases \(Transact-SQL\)](#).

## System View Enhancements

- Two new views support row level security. For more information, see [sys.security\\_predicates \(Transact-SQL\)](#) and [sys.security\\_policies \(Transact-SQL\)](#).
- Seven new views support the Query Store feature. For more information, see [Query Store Catalog Views \(Transact-SQL\)](#).
- 24 new columns are added to [sys.dm\\_exec\\_query\\_stats \(Transact-SQL\)](#) provide information about memory grants.
- Two new query hints (`MIN_GRANT_PERCENT` and `MAX_GRANT_PERCENT`) are added to specify memory grants. See [Query Hints \(Transact-SQL\)](#).
- [sys.dm\\_exec\\_session\\_wait\\_stats \(Transact-SQL\)](#) provides a per session report similar to the server wide [sys.dm\\_os\\_wait\\_stats \(Transact-SQL\)](#).
- [sys.dm\\_exec\\_function\\_stats \(Transact-SQL\)](#) provides execution statistics regarding scalar valued functions.
- Beginning with SQL Server 2016, entries in [sys.dm\\_db\\_index\\_usage\\_stats \(Transact-SQL\)](#) are retained as they were prior to SQL Server 2008 R2.
- Information about statements submitted to an instance of SQL Server can be returned by the new dynamic management function [sys.dm\\_exec\\_input\\_buffer \(Transact-SQL\)](#).
- Two new views support [SQL Server R Services](#): [sys.dm\\_external\\_script\\_requests](#) and [sys.dm\\_external\\_script\\_execution\\_stats](#).

## Security Enhancements

### Row-Level Security

Row-level security introduces predicate based access control. It features a flexible, centralized, predicate-based evaluation that can take into consideration metadata (such as labels) or any other criteria the administrator determines as appropriate. The predicate is used as a criterion to determine whether or not the user has the appropriate access to the data based on user attributes. Label based access control can be implemented by using predicate based access control. For more information, see [Row-Level Security](#).

### Always Encrypted

With Always Encrypted, SQL Server can perform operations on encrypted data, and best of all the encryption key resides with the application inside the customer's trusted environment and not on the server. Always Encrypted secures customer data so DBAs do not have access to plain text data. Encryption and decryption of data happens transparently at the driver level minimizing changes that have to be made to existing applications. For more information, see [Always Encrypted \(Database Engine\)](#).

### Dynamic Data Masking

Dynamic data masking limits sensitive data exposure by masking it to non-privileged users. Dynamic data masking helps prevent unauthorized access to sensitive data by enabling customers to designate how much of the sensitive data to reveal with minimal impact on the application layer. It's a policy-based security feature that hides the sensitive data in the result set of a query over designated database fields, while the data in the database is not changed. For more information, see [Dynamic Data Masking](#).

### New Permissions

- The **ALTER ANY SECURITY POLICY** permission is available as part of the implementation of row level security.
- The **ALTER ANY MASK** and **UNMASK** permissions are available as part of the implementation of dynamic data masking.
- The **ALTER ANY COLUMN ENCRYPTION KEY**, **VIEW ANY COLUMN ENCRYPTION KEY**, **ALTER ANY COLUMN MASTER KEY DEFINITION**, and **VIEW ANY COLUMN MASTER KEY DEFINITION** permissions are available as part of the implementation of the Always Encrypted feature.
- The **ALTER ANY EXTERNAL DATA SOURCE** and **ALTER ANY EXTERNAL FILE FORMAT** permissions are visible in SQL Server 2016 but only apply to the Analytics Platform System (SQL Data Warehouse).
- The **EXECUTE ANY EXTERNAL SCRIPT** permissions are available as part of the support for R scripts.
- The **ALTER ANY DATABASE SCOPED CONFIGURATION** permissions is available to authorize the use of the [ALTER DATABASE SCOPED CONFIGURATION \(Transact-SQL\)](#) statement.

## Transparent Data Encryption

- Transparent Data Encryption has been enhanced with support for Intel AES-NI hardware acceleration of encryption. This will reduce the CPU overhead of turning on Transparent Data Encryption.

## AES Encryption for Endpoints

- The default encryption for endpoints is changed from RC4 to AES.

## New Credential Type

- A credential can now be created at the database level (in addition to the server level credential that was previously available). For more information, see [CREATE DATABASE SCOPED CREDENTIAL \(Transact-SQL\)](#).

# High Availability Enhancements

SQL Server 2016 Standard Edition now supports Always On Basic Availability Groups. Basic availability groups provide support for a primary and secondary replica. This capability replaces the obsolete Database Mirroring technology for high availability. For more information about the differences between basic and advanced availability groups, see [Basic Availability Groups \(Always On Availability Groups\)](#).

Load-balancing of read-intent connection requests is now supported across a set of read-only replicas. The previous behavior always directed connections to the first available read-only replica in the routing list. For more information, see [Configure load-balancing across read-only replicas](#).

The number of replicas that support automatic failover has been increased from two to three.

Group Managed Service Accounts are now supported for Always On Failover Clusters. For more information, see [Group Managed Service Accounts](#). For Windows Server 2012 R2, an update is required to avoid temporary downtime after a password change. To obtain the update, see [gMSA-based services can't log on after a password change in a Windows Server 2012 R2 domain](#).

Always On Availability Groups supports distributed transactions and the DTC on Windows Server 2016. For more information, see [Support for distributed transactions](#).

You can now configure Always On Availability Groups to failover when a database goes offline. This change requires the setting the **DB\_FAILOVER** option to **ON** in the [CREATE AVAILABILITY GROUP \(Transact-SQL\)](#) or [ALTER AVAILABILITY GROUP \(Transact-SQL\)](#) statements.

Always On now supports encrypted databases. The Availability Group wizards now prompt you for a password for any databases that contain a database master key when you create a new Availability Group or when you add databases or add replicas to an existing Availability Group.

Two availability groups in two separate Windows Server Failover Clusters (WSFC) can now be combined into a Distributed Availability Group. For more information, see [Distributed Availability Groups \(Always On Availability Groups\)](#).

Direct seeding allows a secondary replica to be automatically seeded over the network (rather than manual seeding that requires a physical backup of the target database to be restored on the secondary). Direct seeding is specified by setting **SEEDING\_MODE=AUTOMATIC** in the [CREATE AVAILABILITY GROUP \(Transact-SQL\)](#) or [ALTER AVAILABILITY GROUP \(Transact-SQL\)](#) statements. You must also specify **GRANT CREATE ANY DATABASE** with [ALTER AVAILABILITY GROUP \(Transact-SQL\)](#) on each secondary replica that is used with direct seeding.

**Performance improvements** – The synchronization throughput of availability groups has been increased ~10x through parallel and faster compression of log blocks on the primary replica, an optimized synchronization protocol, and parallel decompression and redo of log records on the secondary replica. This increases the freshness of readable secondaries and reduces database recovery time in case of failover. Note that redo for memory-optimized tables is not yet parallel in SQL Server 2016.

## Replication Enhancements

- Replication of memory-optimized tables are now supported. For more information, see [Replication to Memory-Optimized Table Subscribers](#).
- Replication is now supported to Azure SQL Database. For more information, see [Replication to SQL Database](#).

## Tools Enhancements

### Management Studio

Download the latest [SQL Server Management Studio \(SSMS\)](#)

- SQL Server Management Studio supports the Active Directory Authentication Library (ADAL) which is under development for connecting to Microsoft Azure. This replaces the certificate-based authentication used in SQL Server 2014 Management Studio.
- SQL Server Management Studio installation requires installing .NET 4.6 as a pre-requisite. .NET 4.6 will be automatically installed by setup when SQL Server Management Studio is installed.
- A new query result grid option supports keeping Carriage Return/Line Feed (newline characters) when copying or saving text from the results grid. Set this from the Tools/Options menu.
- SQL Server Management Tools is no longer installed from the main feature tree; for details see [Install SQL Server Management Tools with SSMS](#).
- SQL Server Management Studio installation requires installing .NET 4.6.1 as a pre-requisite. .NET 4.6.1 will be automatically installed by setup when SQL Server Management Studio is installed.

### Upgrade Advisor

SQL Server 2016 Upgrade Advisor Preview is a standalone tool that enables users of prior versions to run a set of upgrade rules against their SQL Server database to pinpoint breaking and behavior changes and deprecated features as well as providing help with the adoption of new features such as Stretch Database.

You can download Upgrade Advisor Preview [here](#) or you can install it by using the Web Platform Installer.