# Cải tiến In-memory OLTP

Được giới thiệu từ phiên bản SQL Server 2014, In-Memory OLTP giúp tăng tốc khả năng xử lí công việc trong cùng một thời điểm và quá nhiều bước bằng cách di chuyển dữ liệu từ các bảng trên đĩa sang các memory-optimized tables[[1]](#footnote-1) dựa theo cách quản lí file[[2]](#footnote-2) và biên dịch các thủ tục giúp việc xử lí dữ liệu trong quá trình thực thi (runtime) nhanh hơn.

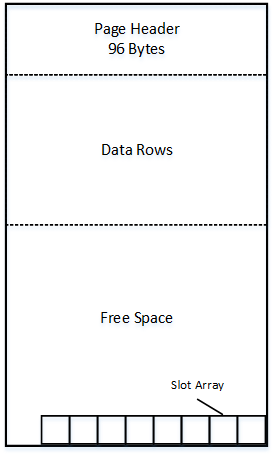
|  |  |  |
| --- | --- | --- |
| Tính nặng/Hạn chế | SQL Server 2014 | SQL Server 2016 |
| [Maximum size of durable table](#_Maximum_size_of) | 256 GB | 2 TB |
| [Large Object Type (LOB)](#_Large_Object_Type) | Không hỗ trợ | Hỗ trợ |
| Transparent Data Encryption (TDE) | Không hỗ trợ | Hỗ trợ |
| Offline Checkpoint Threads | 1 | 1 cho mỗi container |
| ALTER PROCEDURE | Không hỗ trợ | Hỗ trợ |
| Nested native procedure calls | Không hỗ trợ | Hỗ trợ |
| Natively-compiled scalar UDFs | Không hỗ trợ | Hỗ trợ |
| ALTER TABLE | Không hỗ trợ  DROP / re-CREATE | Hỗ trợ  Offline |
| DML triggers | Không hỗ trợ | Hỗ trợ 1 phần |
| Indexed on NULLable columns | Không hỗ trợ | Hỗ trợ |
| Non-BIN2 collation in index key columns | Không hỗ trợ | Hỗ trợ |
| Non-Latin codepages for [var]char columns | Không hỗ trợ | Hỗ trợ |
| Check/Unique Constraint | Không hỗ trợ | Hỗ trợ |
| Parallelism | Không hỗ trợ | Hỗ trợ |
| Multiple Active Result Set (MARS) | Không hỗ trợ | Hỗ trợ |
| SSMS Table Design | Không hỗ trợ | Hỗ trợ |

## Maximum size of durable table

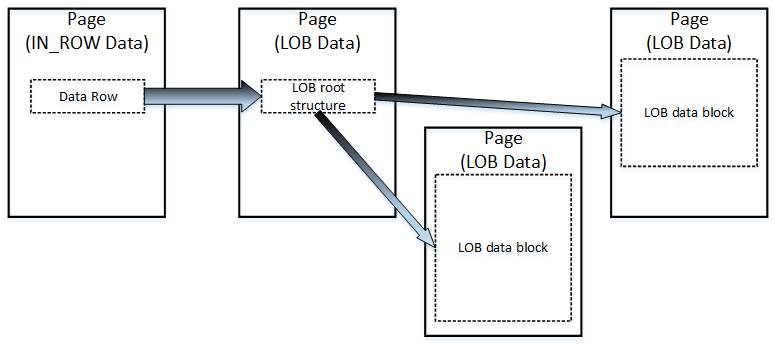
Memory-optimized tables trong SQL Server 2016 đem lại hiệu quả rất tốt cả về khả năng lưu trữ và tốc độ truy vấn. Trong bản phát hành SP1, SQL Server 2016 có thể lưu trữ tối đa 2 terabyte (TB) dữ liệu trong memory-optimized tables nhớ khi tạo bảng bằng cách sử dụng tùy chọn SCHEMA\_AND\_DATA trong câu lệnh CREATE TABLE. Hơn nữa, bạn có thể tạo một memory-optimized tables với kích thước hàng lớn hơn 8.060 byte[[3]](#footnote-3), ngay cả khi không có cột LOB khi tạo bảng.

## Large Object Type (LOB)

Như đã đề cập một phần bên trên, LOB cung cấp cho người sử dụng khả năng tạo bảng với kích thước hàng lớn hơn rất nhiều so với việc sử dụng Data pages thông qua việc sử dụng hai cơ chế lưu trữ mới được hỗ trợ Hash và nonclustered (Range)[[4]](#footnote-4).



Hình 1: Cấu trúc dữ liệu trong mỗi Data pages



Hình 2: Cấu trúc dữ liệu được sử dụng trong LOB

Giống như ROW\_OVERFLOW data, LOB sử dụng LOB root structure[[5]](#footnote-5), nơi sẽ chứa tập hợp các thông tin về trang, hàng dữ liệu khác và tất nhiên là con trỏ trỏ tới từng Data block đó. Điều này cải thiện sự hạn chế về mặt lưu trữ dữ liệu so với cách sử dụng Data pages truyền thống (8KB cho mỗi page với 8060 KB data). Cách lưu trữ này có điểm tương đồng so với B-Tree[[6]](#footnote-6).

## Transparent Data Encryption[[7]](#footnote-7)

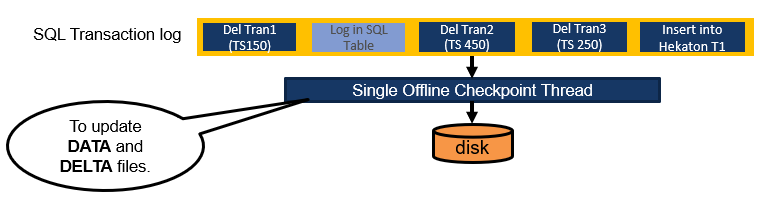
*TDE mã hóa các tệp cơ sở dữ liệu bên dưới. Không ai có quyền truy cập vật lý vào tệp có thể đọc dữ liệu mà không có khóa mã hóa.*

TDE mã hóa các tệp dữ liệu vật lý, logs và bản sao lưu mà không mã hóa trực tiếp bảng dữ liệu. Nghĩa là, nếu người dùng có quyền đọc đối với cơ sở dữ liệu có bật TDE, người dùng có thể truy vấn cơ sở dữ liệu và truy cập tất cả dữ liệu mà không cần có khóa mã hóa. Nếu bạn di chuyển các tệp được mã hóa sang một máy chủ khác, không ai có thể mở và xem chúng trên máy chủ đó.

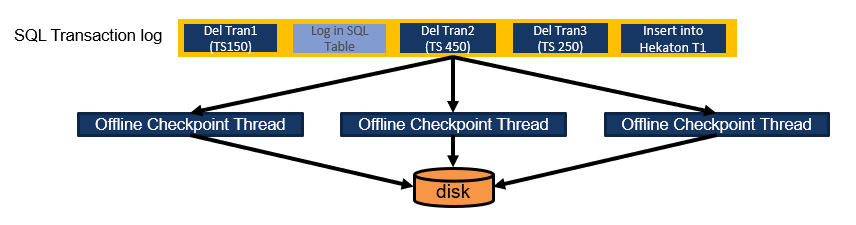
Trong phiên bản này, TDE được cải thiện để hỗ trợ tăng tốc mã hóa thông qua Intel AES-NI giúp giảm khối lượng công việ cho CPU/DTU.

## Offline Checkpoint Thread

* Dữ liệu được bảo vệ thông qua các checkpoint files[[8]](#footnote-8).
* Dữ liệu có thể được restore thông qua các checkpoint files này và transaction log.
* Số lượng Offline Checkpoint Thread được tăng lên từ duy nhất 1 trở thành 1 cho mỗi container, điều đó có nghĩa rằng SQL Server 2016 hỗ trợ Multi Offline Checkpoint Thread.



Hình 3: Offline Checkpoint Thread trong SQL Server 2014



Hình 4: Offline Checkpoint Thread trong SQL Server 2016

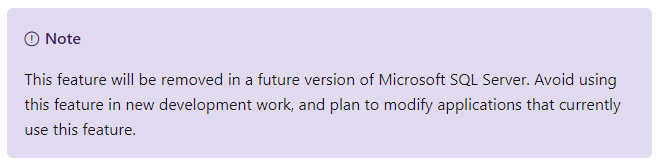
## ALTER PROCEDURE[[9]](#footnote-9)

Stored procedure[[10]](#footnote-10) là tập hợp một hoặc nhiều câu lệnh T-SQL thành một nhóm đơn vị xử lý logic và được lưu trữ trên Database Server. Khi một câu lệnh gọi chạy stored procedure lần đầu tiên thì SQL Server sẽ chạy nó và lưu trữ vào bộ nhớ đệm, gọi là plan cache, những lần tiếp theo SQL Server sẽ sử dụng lại plan cache nên sẽ cho tốc độ xử lý tối ưu.

Stored procedure rất tiện lợi cho người quản trị database (DBA), nó giúp DBA tạo ra những nhóm câu lệnh và gửi đến một bô phận khác mà họ sẽ không cần quan tâm đến nội dung bên trong stored procedure có gì, họ chỉ quan tâm đến tham số đầu vào và đầu ra.

Phiên bản SQL Server 2016 cho phép người sử dụng có khả năng chỉnh sửa các Procedure đã được biên dịch thông qua câu lệnh ALTER PROCEDURE.

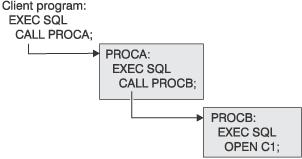
Tuy nhiên, có vẻ như Microsoft còn mặn mà với câu lệnh này:



Hình 5: ALTER PROCEDURE trong tương lai

## Nested native procedure calls

SQL Server 2016 cho phép người sử dụng có khả năng gọi những PROCEDURE lồng nhau, vốn là điều khá cơ bản trong các ngôn ngữ lập trình thông dụng khác.



Hình 6: Ví dụ về PROCEDURE lồng nhau

## Natively-compiled scalars UDFs[[11]](#footnote-11)

Trong SQL Server 2016, người dùng có thể tạo và hủy các natively compiled, scalar user-defined functions. Người dùng cũng có thể thay đổi các chức năng do người dùng xác định này. Native compilation cải thiện hiệu suất của các user-defined functions trong Transact-SQL. Khi bạn thay đổi một natively compiled, user-defined functions, ứng dụng vẫn luôn được hoạt động mặc dù quá trình compile đang diễn ra.

## ALTER TABLE[[12]](#footnote-12)

Sửa đổi bảng: thay đổi, thêm hoặc bớt các cột và ràng buộc. ALTER TABLE cũng chỉ định lại và xây dựng lại các phân vùng, hoặc tắt và bật các ràng buộc và trình kích hoạt.

## DML Trigger[[13]](#footnote-13)

DML Trigger là một loại thủ tục được lưu trữ đặc biệt tự động có hiệu lực khi sự kiện data manipulation language (DML) diễn ra ảnh hưởng đến bảng hoặc chế độ xem được xác định trong trigger. Các sự kiện DML bao gồm các câu lệnh INSERT, UPDATE hoặc DELETE. DML Trigger có thể được sử dụng để thực thi các quy tắc nghiệp vụ và tính toàn vẹn dữ liệu, truy vấn các bảng khác và bao gồm các câu lệnh T-SQL phức tạp. DML Trigger và câu lệnh kích hoạt nó được coi là một thao tác duy nhất, có thể được khôi phục lại từ bên trong trình kích hoạt. Nếu một lỗi nghiêm trọng được phát hiện (ví dụ: không đủ dung lượng đĩa), toàn bộ thao tác sẽ tự động quay trở lại.

## NULLable index key columns.

Các cột được gán PRIMARY\_KEY trong phiên bản SQL Server 2016 đã có thể nhận giá trị rỗng cho giá trị trong cột của mình. Trước đó, In-memory OLTP trong phiên bản 2014 hoàn toàn không hỗ trợ điều này[[14]](#footnote-14).

## Non-BIN2 collation in index key columns[[15]](#footnote-15)

Collation là một tập hợp các quy tắc để so sánh hai kí tự trong một tập hợp kí tự[[16]](#footnote-16). Và trong phiên bản này, SQL Server 2016 hỗ trợ Collation \_BIN2, giúp sắp xếp và so sánh dữ liệu cho bảng mã Unicode. Với những dữ liệu không thuộc bảng mã Unicode, việc sắp xếp và so sánh giống với các phép so sánh và sắp xếp nhị phân (binary).

## Non-Latin codepages for [var]char columns

Người dùng có thể sử dụng các bảng mã mở rộng khác cho các kí tự Non-Latin.

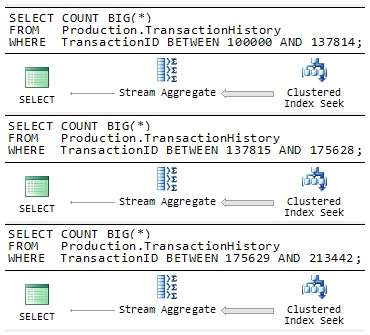
## Unique Constraints and Check Constraints[[17]](#footnote-17)

*UNIQUE và CHECK là 2 từ khóa được sử dụng để thực thi tính toàn vẹn dữ liệu trong bảng.*

* UNIQUE: Người dùng có thể sử dụng UNIQUE để đảm bảo không có giá trị trùng lặp nào được nhập vào các cột cụ thể nào đó cột đó không có ràng buộc khóa chính.
* CHECK: Giống IF

## Parallelism[[18]](#footnote-18)[[19]](#footnote-19)[[20]](#footnote-20)

Parallelism có thể được định nghĩa là cơ chế giúp đạt được thông lượng tốt hơn bằng cách thực hiện nhiều luồng cùng một lúc hoặc tự động hóa các tác vụ. Nó đạt được bằng cách giữ cho phần cứng luôn bận rộn khi sử dụng nhiều luồng, xử lý nhiều mục trên mỗi luồng và/hoặc thực hiện nhiều thao tác với cùng một dữ liệu đã có trong bộ nhớ. Bằng cách giữ cho phần cứng bận rộn, các luồng dành ít thời gian hơn để chờ đợi tài nguyên không khả dụng tại thời điểm đó và buộc phải chuyển đổi ngữ cảnh.



Hình 7: Manual Parallelism

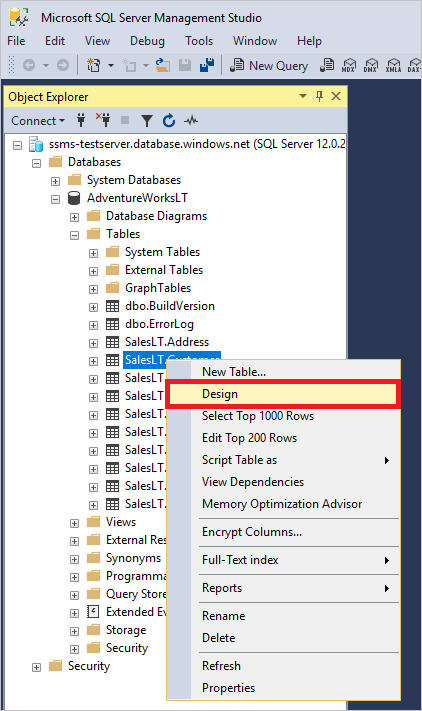
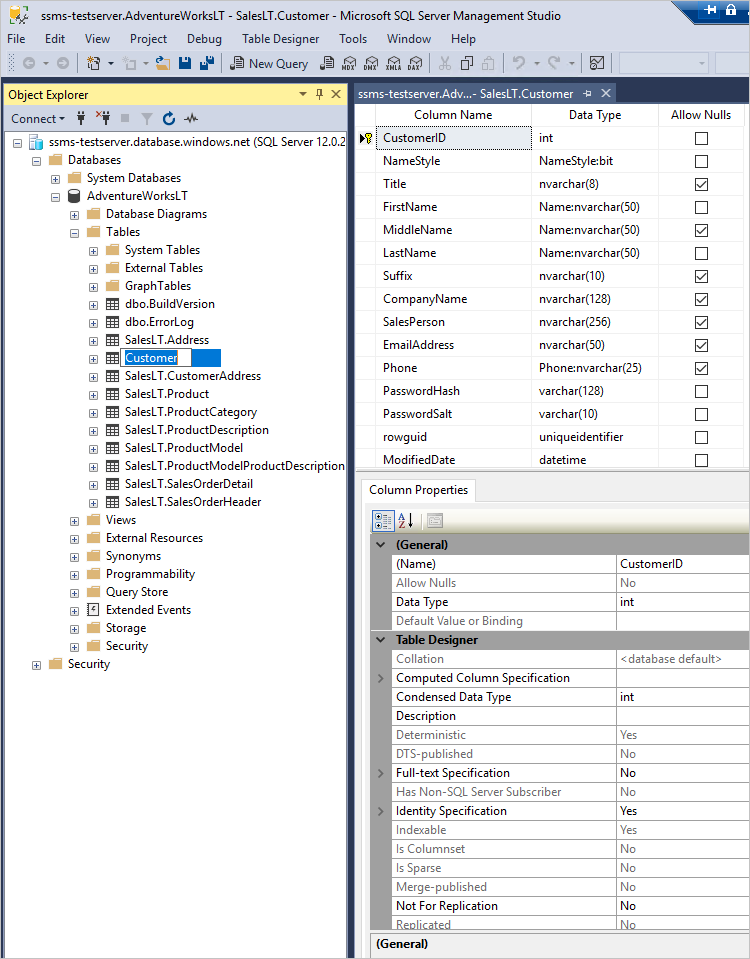
## Multiple Active Result Set (MARS)[[21]](#footnote-21)[[22]](#footnote-22)

Multiple Active Result Set (MARS) là một tính năng cho phép thực hiện nhiều batches trên một kết nối. Trong các phiên bản trước, mỗi lần chỉ có thể thực hiện một batches đối với một kết nối duy nhất. Thực thi nhiều batches với MARS không có nghĩa là thực hiện đồng thời các hoạt động.

Khi kết nối với một Memory-optimized tables hoặc chạy một stored procedure, công cụ cơ sở dữ liệu có thể bắt đầu tìm nạp các hàng từ tập kết quả mới trước khi truy xuất hoàn toàn các hàng từ một yêu cầu trước đó.

## SSMS Table Design[[23]](#footnote-23)

Table Designer là một công cụ trực quan nơi bạn thiết kế và trực quan hóa các bảng cơ sở dữ liệu. Sử dụng SQL Server Management Studio (SSMS) để tạo, chỉnh sửa hoặc xóa bảng, cột, khóa, chỉ mục, mối quan hệ và ràng buộc.

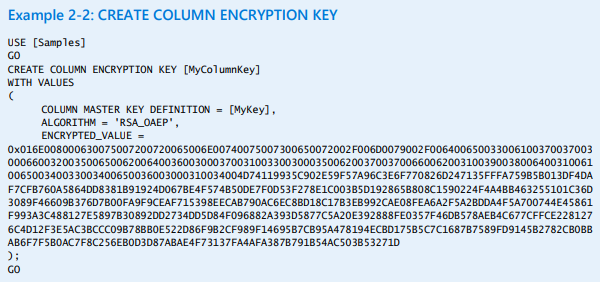
Hình 8: Sử dụng SSMS Table Design

# Always Encrypted

Always Encrypted là công nghệ mã hóa phía máy khách, trong đó dữ liệu được mã hóa tự động không chỉ khi được viết mà còn khi được đọc bởi một ứng dụng đã được phê duyệt. Không giống như Transparent Data Encryption (Mã hóa dữ liệu minh bạch) mã hóa dữ liệu trên đĩa nhưng cho phép bất kỳ ứng dụng nào truy vấn dữ liệu đọc dữ liệu, Always Encrypted yêu cầu ứng dụng khách của bạn sử dụng trình điều khiển (driver) được bật Always Encrypted để giao tiếp với cơ sở dữ liệu. Bằng cách sử dụng trình điều khiển này, ứng dụng chuyển dữ liệu được mã hóa đến cơ sở dữ liệu một cách an toàn, sau đó chỉ có thể được giải mã bởi ứng dụng có quyền truy cập vào khóa mã hóa. Bất kỳ ứng dụng nào khác truy vấn dữ liệu cũng có thể truy xuất các giá trị được mã hóa, nhưng ứng dụng đó không thể sử dụng dữ liệu mà không có khóa mã hóa, do đó khiến dữ liệu trở nên vô dụng. Do cấu trúc mã hóa này, người dùng sẽ không thể tìm thấy bản lưu của cơ sở dữ liệu dưới dạng raw data.

## Một số khái niệm cơ bản

* **Column master key**: Column master key là một certificate được lư trữ trong Windows certificate store, Hardware Security Module – HSM (Module bảo vệ phần cứng) của bên thứ ba , hoặc Azure Key Vault. Người dùng sẽ sử dụng Column master key để bảo vệ các khóa mã hóa cột khác nhau hoặc cả một bảng.
* **Column encryption keys**:



Hình 9: Column encryption key

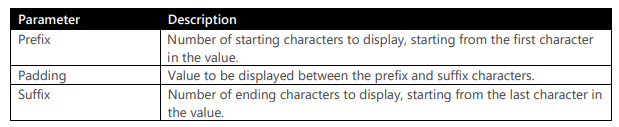
## Row-Level Security

Row-Level Security (RLS) cho phép bạn định cấu hình tables sao cho người dùng chỉ nhìn thấy các hàng trong bảng mà bạn cấp cho họ quyền truy cập. Tính năng này giới hạn hàng nào được trả lại cho người dùng, bất kể họ đang sử dụng ứng dụng nào. Người dùng có thể sử dụng một bộ lọc để lọc âm thầm các hàng mà người dùng có thể truy cập khi sử dụng câu lệnh INSERT, UPDATE hoặc DELETE. Ngoài ra, người quản trị có thể sử dụng các từ khóa sau để chặn người dùng ghi dữ liệu: AFTER INSERT, AFTER UPDATE, BEFORE UPDATE và BEFORE DELETE. Các truy vấn này trả về một lỗi cho ứng dụng cho biết rằng người dùng đang cố gắng sửa đổi các hàng mà người dùng không có quyền truy cập.

## Dynamic data masking[[24]](#footnote-24)

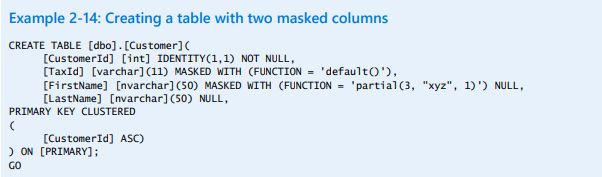
Khi người quản trị có nhu cầu che dấu một phần dữ liệu nhạy cảm trong một hàng khỏi nhóm người dùng nhất định, họ có thể sử dụng Dynamic data masking để obfucate dữ liệu trừ khi bạn phân quyền cụ thể cho người đó. Người quản trị có thể sử dụng một trong 4 chức năng tạo Dynamic data masking sau:

* **Default**: Sử dụng hàm này để che lấp đầy đủ các giá trị bằng cách trả về giá trị XXXX (hoặc ít hơn X nếu độ dài cột nhỏ hơn 4 ký tự) cho kiểu dữ liệu chuỗi, 0 cho kiểu dữ liệu số và nhị phân và 01.01.2000 00: 00: 00.0000000 cho kiểu dữ liệu ngày và giờ.
* **Email**: Sử dụng chức năng này để che dấu một phần các địa chỉ email như sau: aXXX@XXXX.com. Mô hình này không chỉ che giấu địa chỉ email mà còn che giấu độ dài của địa chỉ email.
* **Partial**: Sử dụng khi người quản trị muốn tự config cách mà dữ liệu được ẩn dấu, bao gồm các thuộc tính sau:

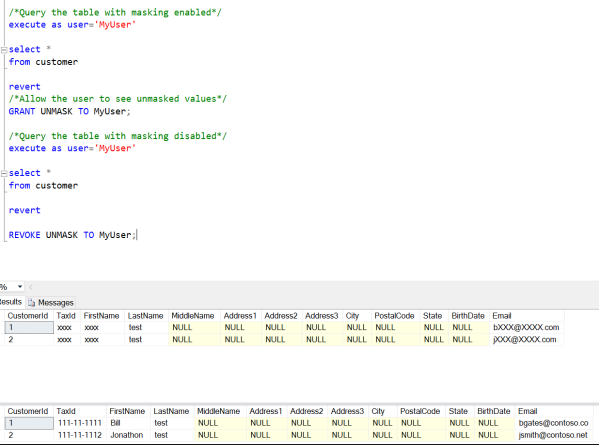


Hình 10: Partial Dynamic data masking

* **Random**: Sử dụng hàm này để che lấp đầy đủ các giá trị số bằng cách sử dụng giá trị ngẫu nhiên giữa ranh giới dưới và ranh giới trên mà người quản trị chỉ định.



Hình 11: Sử dụng Dynamic data masking



Hình 12: Kết quả truy vấn khi sử dụng Dynamic data masking

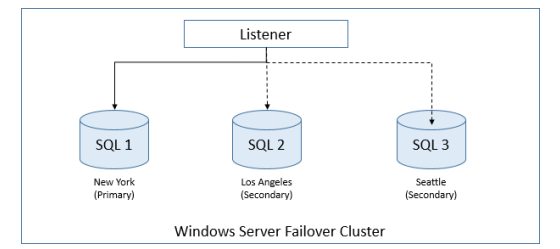
# Higher availability

## Database Mirroring (OLD)[[25]](#footnote-25)

## AlwaysOn Availability Groups[[26]](#footnote-26)

Được giới thiệu lần đầu tiên trong SQL Server 2012 Enterprise Edition, tính năng AlwaysOn Avavailable Groups cung cấp khả năng bảo vệ dữ liệu bằng cách gửi các transaction log trên bản sao chính đến một hoặc nhiều bản sao thứ cấp, một quá trình về mặt khái niệm tương tự như sao chép cơ sở dữ liệu. Trong SQL Server 2014, cải tiến đáng kể cho các AlwaysOn Avavailable Groups là sự gia tăng số lượng bản sao thứ cấp được hỗ trợ từ ba lên tám. SQL Server 2016 bao gồm một số cải tiến mới:

* AlwaysOn Basic Availability Groups
* Support for group Managed Service Accounts (gMSAs)
* Database-level failover
* Distributed Transaction Coordinator (DTC) support
* Load balancing for readable secondary replicas
* Up to three automatic failover targets
* Improved log transport performance

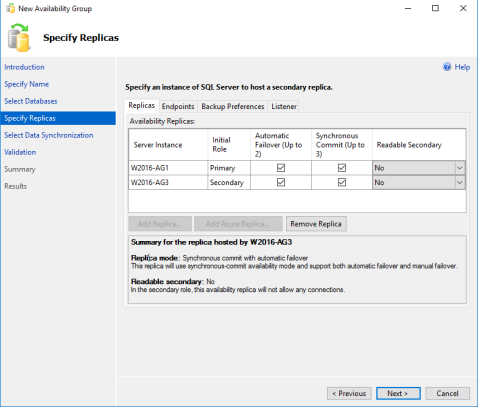


Hình 13: 3 Automatic failover target

## Supporting disaster recovery with basic availability groups

Bây giờ người quản trị có thể sử dụng basic availability groups[[27]](#footnote-27) trong Standard Edition để phòng ngừa khả năng mất mát trên cơ sở dữ liệu. Việc sử dụng các basic availability groups phải tuân theo các giới hạn sau:

* Two replicas (one primary and one secondary)
* One availability database
* No read access on secondary replica
* No backups on secondary replica
* No availability group listener
* No support in an existing availability group to add or remove a replica
* No support for upgrading a basic availability group to an advanced availability group



Hình 14: Cài đặt basic availability groups

## Using group Managed Service Accounts[[28]](#footnote-28)

Tài khoản dịch vụ được quản lý độc lập - standalone Managed Service Account (sMSA) là tài khoản miền được quản lý cung cấp tính năng quản lý mật khẩu tự động, quản lý tên dịch vụ chính (SPN) được đơn giản hóa và khả năng ủy quyền quản lý cho các quản trị viên khác. Loại managed service account (MSA) này đã được giới thiệu trong Windows Server 2008 R2 và Windows 7.

Tài khoản dịch vụ được quản lý theo nhóm (gMSA) cung cấp chức năng tương tự trong miền nhưng mở rộng chức năng đó trên nhiều máy chủ. Khi kết nối với dịch vụ được lưu trữ trên máy chủ, chẳng hạn như giải pháp Cân bằng tải mạng (Network Load Balanced solution), các giao thức xác thực hỗ trợ xác thực lẫn nhau yêu cầu tất cả các phiên bản của dịch vụ sử dụng cùng một nguyên tắc. Khi một gMSA được sử dụng làm nguyên tắc dịch vụ, hệ điều hành Windows quản lý mật khẩu cho tài khoản thay vì dựa vào quản trị viên để quản lý mật khẩu.

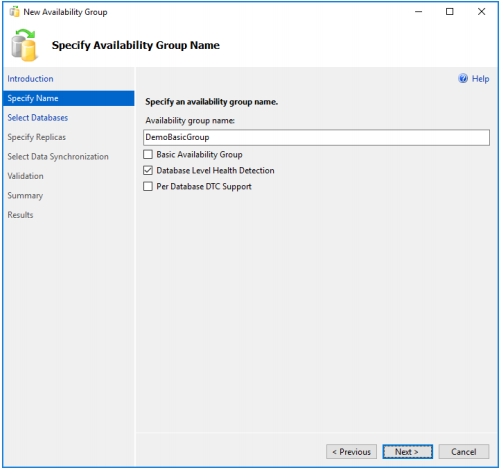
Để tuân thủ các yêu cầu về audit, DBA hoặc quản trị viên hệ thống trong một doanh nghiệp lớn phải thường xuyên đặt lại mật khẩu tài khoản dịch vụ trên các phiên bản SQL Server. Tuy nhiên, việc quản lý mật khẩu tài khoản dịch vụ cá nhân tiềm ẩn mức độ rủi ro cao vì sự cố có thể xảy ra bất kì lúc nào. Để giải quyết vấn đề này, Microsoft đã nâng cao Windows Server 2012 để bạn có thể dễ dàng quản lý mật khẩu cho tài khoản dịch vụ trong Active Directory bằng cách tạo một tài khoản dịch vụ duy nhất cho các phiên bản SQL Server của bạn và sau đó cấp quyền cho từng máy chủ đó. Theo mặc định, Active Directory thay đổi mật khẩu cho group Managed Service Account (gMSA) ba mươi ngày một lần, mặc dù quản trị viên có thể điều chỉnh khoảng thời gian thay đổi mật khẩu để đáp ứng các yêu cầu của doanh nghiệp. Trong SQL Server 2012 và SQL Server 2014, bạn chỉ có thể triển khai tính năng này trong các cấu hình độc lập. Trong SQL Server 2016, bây giờ bạn có thể sử dụng gMSA với cả nhóm khả dụng và cụm chuyển đổi dự phòng.

## Triggering failover at the database level

Kể từ SQL Server 2012, AlwaysOn Availability Groups và AlwaysOn Failover Cluster Instances[[29]](#footnote-29) (FCIs) sử dụng sp\_server\_diagnostics stored procedure [[30]](#footnote-30)to định kì monitor tình trạng của server. Các cảnh báo về FCI được đặt khi một trong bất kì điều kiện nào dưới đây xảy ra:

* The stored procedure returns an error condition.
* The SQL Service service is not running.
* The SQL Server instance is not responding.

Tuy nhiên, trong các phiên bản trước SQL Server 2016, việc kiểm tra này không tính đến lỗi cấp cơ sở dữ liệu. Bắt đầu từ SQL Server 2016, quản trị viên có thể bật Database Level Health Detection[[31]](#footnote-31) khi tạo một availability group, như thể hiện trong Hình 15. Bằng cách này, bất kỳ lỗi nào khiến cơ sở dữ liệu bị nghi ngờ hoặc chuyển sang trạng thái ngoại tuyến cũng sẽ kích hoạt chuyển đổi dự phòng của availability group.



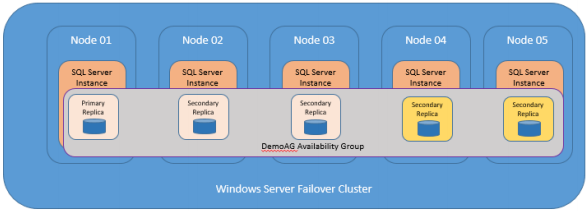
Hình 15: Tạo một availability group mới với Database Level Health Detection

## Supporting distributed transactions

Một trong những tính năng được hỗ trợ từ lâu trong FCI, nhưng không có trong các availability group, là sử dụng Điều phối viên giao dịch phân tán - Distributed Transaction Coordinator (DTC)[[32]](#footnote-32). Tính năng này là bắt buộc nếu ứng dụng của bạn thực hiện các truy vấn trong nhiều trường hợp. Khi chạy SQL Server 2016 trên Windows Server 2016, bây giờ bạn có thể triển khai hỗ trợ cho các distributed transactions khi bạn tạo một availability group mới.

## Defining automatic failover targets

Trong SQL Server 2012 và SQL Server 2014, bạn có thể xác định tối đa hai bản sao đang chạy trong một automatic failover set (tập hợp chuyển đổi dự phòng tự động), nhưng hiện tại SQL Server 2016 cho phép bản sao thứ ba hỗ trợ cấu trúc liên kết như được hiển thị trong Hình 16. Trong ví dụ này, các bản sao trên Node01, Node02 và Node03 được định cấu hình như một bộ chuyển đổi dự phòng tự động. Miễn là dữ liệu được đồng bộ hóa giữa bản sao chính và một trong các bản sao thứ cấp, chuyển đổi dự phòng có thể diễn ra theo cách tự động mà không mất dữ liệu.



Hình 16: Availability Group topology with three automatic failover targets.

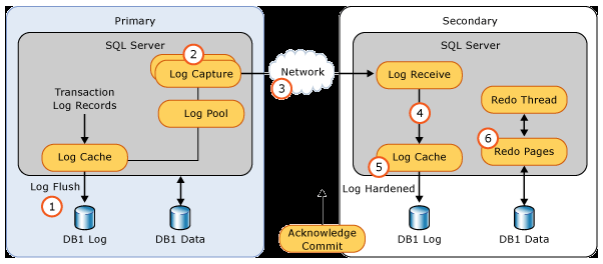
Khi định cấu hình chuyển đổi dự phòng Availability Group, bạn có thể chọn trong số các chế độ chuyển đổi dự phòng sau:

* **Automatic Failover:** Chuyển đổi dự phòng xảy ra tự động khi hỏng bản sao chính, chỉ được hỗ trợ khi cả bản sao chính và ít nhất một bản sao thứ cấp được định cấu hình với chế độ chuyển đổi dự phòng TỰ ĐỘNG và bản sao phụ hiện được đồng bộ hóa.
* **Planned Manual Failover** (without data loss): Một chuyển đổi dự phòng thường do quản trị viên khởi tạo cho các mục đích bảo trì. Điều này yêu cầu chế độ cam kết đồng bộ và các cơ sở dữ liệu hiện phải được đồng bộ hóa.
* **Forced Failover** (with possible data loss): Chuyển đổi dự phòng xảy ra khi nhóm khả dụng được định cấu hình với chế độ cam kết không đồng bộ hoặc cơ sở dữ liệu trong nhóm khả dụng hiện không được đồng bộ hóa.

## Improve log transport performance

Quá trình vận chuyển log:

1. **Log flush:** Log data is generated and flushed to disk on the primary replica in preparation for replication to the secondary replica. It then enters the send queue.
2. **Log capture:** Logs for each database are captured on the primary replica, compressed, and sent to the corresponding queue on the secondary replica. This process runs continuously as long as database replicas are connecting. If this process is not able to scan and enqueue the messages quickly enough, the log send queue continues to grow.
3. **Send:** The messages are removed from the queue and sent to each secondary replica across the network.
4. **Log receive/Log cache:** Each secondary replica gets messages from the primary replica and then caches the messages.
5. **Log hardened:** The log is flushed on the secondary replica, and then a notification is sent to the primary replica to acknowledge completion of the transaction.
6. **Redo pages:** The flushed pages are retrieved from the redo queue and applied to the secondary replica.



Hình 17: Log transport operations for AlwaysOn Availability Groups.

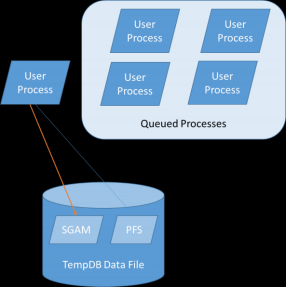
Sự tắc nghẽn có thể xảy ra trong quá trình này trong bước ghi nhật ký trên bản sao chính và bước làm lại trên bản sao thứ cấp. Trong các phiên bản trước của SQL Server, cả hai bước đều là đơn luồng. Do đó, tắc nghẽn có thể xảy ra trong quá trình xây dựng lại chỉ mục lớn trên các nhóm khả dụng có lưu trữ tốc độ cao và trên mạng cục bộ, vì các bước đơn luồng này gặp khó khăn khi theo kịp luồng bản ghi nhật ký. Tuy nhiên, trong SQL Server 2016 các bước này có thể sử dụng nhiều luồng chạy song song, dẫn đến cải thiện hiệu suất đáng kể. Hơn nữa, các chức năng nén trong bước ghi nhật ký đã được thay thế bằng một chức năng nén Windows mới hơn mang lại hiệu suất tốt hơn tới năm lần. Trong quá trình thử nghiệm với các thiết bị lưu trữ thông lượng cao, tốc độ lên đến 500 MB/s đã được ghi nhận. Xét rằng thông lượng này là một luồng được nén, bước làm lại sẽ nhận được 1 GB/s, giúp dành bộ nhớ và khả năng xử lí cho các tác vụ khác cần thiết.

# Improved database engine

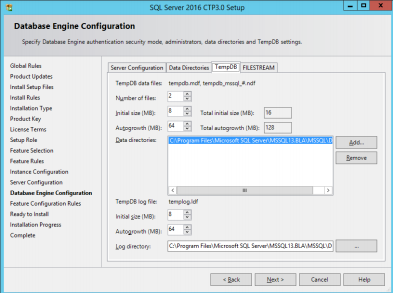
## TempDB[[33]](#footnote-33)[[34]](#footnote-34) enhancements

TempDB là một trong những thành phần mà hiệu suất rất quan trọng trong SQL Server vì công cụ cơ sở dữ liệu sử dụng nó cho các bảng tạm thời, tràn bộ nhớ truy vấn, xây dựng lại chỉ mục, Service Broker và vô số chức năng nội bộ khác. TempDB file behavior đã được nâng cao và tự động hóa trong SQL Server 2016 để loại bỏ nhiều vấn đề về hiệu suất liên quan đến cấu hình cơ bản của máy chủ. Những thay đổi này cho phép quản trị viên tập trung nỗ lực vào các vấn đề cấp bách hơn về hiệu suất và dữ liệu trong môi trường của họ.

Trong các phiên bản trước của SQL Server, cấu hình mặc định sử dụng một tệp dữ liệu cho TempDB. Hạn chế này đôi khi dẫn đến tranh chấp chốt trang, thường bị quản trị viên chẩn đoán nhầm là sự cố I/O lưu trữ cho SQL Server. Tuy nhiên, các trang cho TempDB thường nằm trong bộ nhớ và do đó không phải là vấn đề tranh chấp I/O. Thay vào đó, ba loại trang đặc biệt là nguyên nhân của vấn đề tranh chấp chốt trang: Global Allocation Map (GAM), Shared Global Allocation Map (SGAM) và Page Free Space (PFS). Mỗi tệp cơ sở dữ liệu có thể chứa nhiều loại trang này, chúng có nhiệm vụ xác định nơi ghi dữ liệu đến trong tệp dữ liệu vật lý. Bất cứ khi nào một quy trình trong SQL Server cần sử dụng bất kỳ tệp nào trong số này, một latch (chốt) sẽ được thực hiện. Latch tương tự như khóa nhưng nhẹ hơn. Latch được thiết kế để bật nhanh và tắt nhanh khi không cần thiết. Vấn đề với TempDB là mỗi tệp dữ liệu chỉ có một trang GAM, SGAM và PFS trên bốn gigabyte dung lượng và rất nhiều quy trình đang cố gắng truy cập các trang đó, như trong Hình 18. Các yêu cầu tiếp theo bắt đầu xếp hàng và thời gian chờ các quy trình ở cuối hàng đợi tăng từ mili giây lên giây.



Hình 18: Tranh chấp trong TempDB

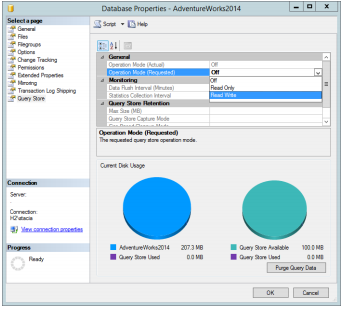


Hình 19: Cài đặt TempDB

## Query Store[[35]](#footnote-35)

Một trong những tình huống phổ biến nhất mà quản trị có thể gặp phải là người dùng báo cáo rằng một truy vấn đột nhiên chạy chậm hơn trước đây hoặc một công việc chiếm mất 10 phần thời gian thay vì 3 như thường lệ. Sự suy giảm hiệu suất này có thể là kết quả của các thay đổi trong dữ liệu gây ra thống kê lỗi hoặc thay đổi thông số thực thi, quá tải hiệu năng phần cứng. Trong các phiên bản trước của SQL Server, việc khắc phục các sự cố này yêu cầu quản trị thu thập dữ liệu từ bộ nhớ cache của kế hoạch và phân tích cú pháp nó bằng cách sử dụng Truy vấn XML (xQuery), việc này có thể tốn nhiều công sức. Ngay cả khi đó, quản trị có thể cũng không có được thông tin họ cần.

Tính năng Query Store mới trong SQL Server 2016 đơn giản hóa việc xác định các ngoại lệ về hiệu suất, quản lý hồi quy kế hoạch thực thi và cho phép nâng cấp dễ dàng hơn giữa các phiên bản của SQL Server. Nó có hai mục tiêu chính — để đơn giản hóa việc xác định các vấn đề về hiệu suất và đơn giản hóa việc khắc phục sự cố hiệu suất cho các truy vấn gây ra bởi những thay đổi trong kế hoạch thực thi. Query Store cũng hoạt động như một bộ ghi dữ liệu cho cơ sở dữ liệu, ghi lại số liệu thống kê về thời gian chạy truy vấn và cung cấp một bảng điều khiển để sắp xếp các truy vấn theo mức tiêu thụ tài nguyên. Bộ sưu tập dữ liệu khổng lồ này không chỉ đóng vai trò là tài nguyên cho các chức năng tự động của Query Store mà còn là tài nguyên khắc phục sự cố cho DBA.



Hình 20: Enabling Query Store

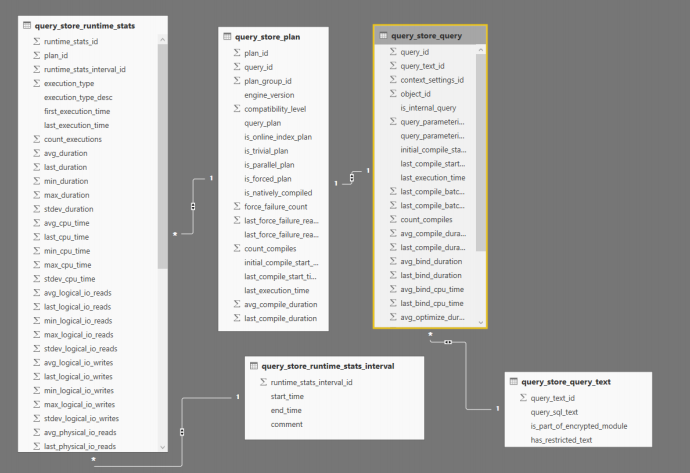
Query Store gồm 2 loại:

* **Plan store**: Duy trì kế hoạch thực thi
* **Run-time stats store**: Lưu trữ thống kê thời gian chạy duy trì các thống kê xung quanh việc thực thi truy vấn, chẳng hạn như CPU, I/O, bộ nhớ và các chỉ số khác.

SQL Server giữ lại dữ liệu này cho đến khi không gian được cấp cho Query Store đầy. Để giảm tác động đến hiệu suất, SQL Server ghi thông tin vào từng cửa hàng này một cách không đồng bộ. Giá trị kích thước mặc định cho Query Store là 100MB.

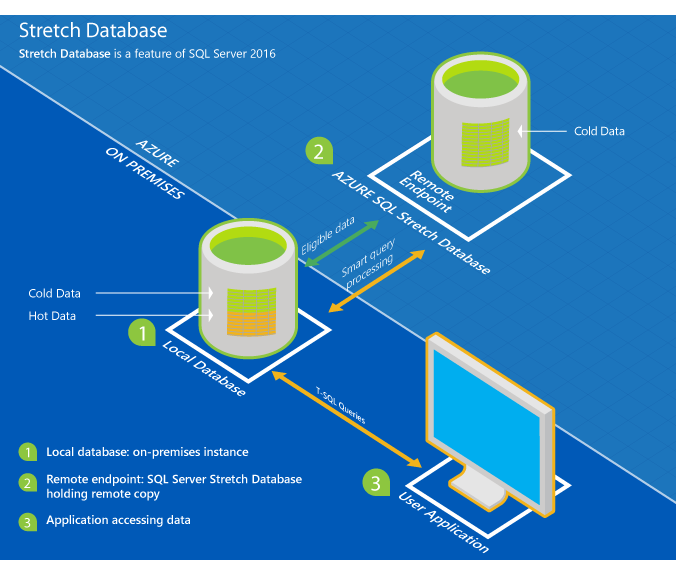
Quản trị viên có thể sử dụng 5 dạng xem Query Store sau:

* **query\_store\_runtime\_stats:** Run-time execution statistics for queries.
* **query\_store\_runtime\_stats\_interval:** Start and end times for the intervals over which run-time execution statistics are collected.
* **query\_store\_plan:** Execution plan information for queries.
* **query\_store\_query:** Query information and its overall aggregated run-time execution statistics.
* **query\_store\_query\_text:** Query text as entered by the user, including white space, hints, and comments.



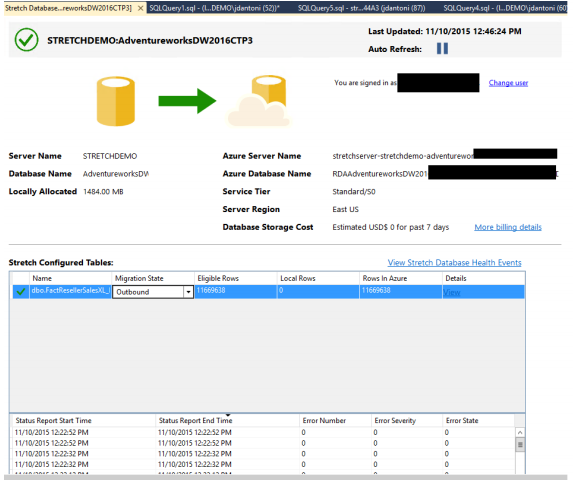
Hình 21: Query Store catalog views

## Stretch Database[[36]](#footnote-36)



Hình 22: Stretch Database

SQL Server 2016 giới thiệu một tính năng kết hợp mới được gọi là Stretch Database kết hợp sức mạnh của Cơ sở dữ liệu SQL Azure với phiên bản SQL Server tại chỗ để cung cấp bộ nhớ gần như vô tận với chi phí thấp hơn đáng kể, cộng với bảo mật cấp doanh nghiệp và chi phí quản lý gần như bằng không. Với Stretch Database, quản trị có thể lưu trữ cold data, được truy cập không thường xuyên trong Azure mà không thay đổi đối với mã ứng dụng. Tất cả các chính sách quản trị và bảo mật vẫn được quản lý từ cùng một cơ sở dữ liệu SQL Server cục bộ như trước đây.



Hình 23: Monitoring Stretch Database

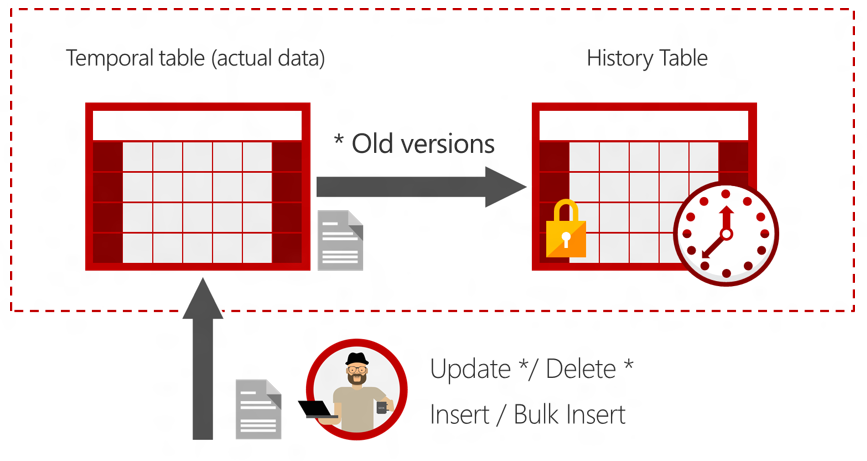
# Broader data access

## Temporal tables[[37]](#footnote-37)

Một thách thức chung với quản lý dữ liệu là quyết định cách xử lý các thay đổi đối với dữ liệu. Ít nhất, quản trị cần một cách dễ dàng để giải quyết một thay đổi ngẫu nhiên mà không cần đến khôi phục cơ sở dữ liệu. Đôi khi quản trị phải có khả năng cung cấp một bản ghi ghi lại cách một hàng đã thay đổi theo thời gian và ai đã thay đổi hàng đó. Hoặc quản trị có thể cần thực hiện phân tích xu hướng để so sánh các giá trị cho một danh mục tại các thời điểm khác nhau hoặc tìm giá trị của chỉ số kinh doanh tại một thời điểm cụ thể.

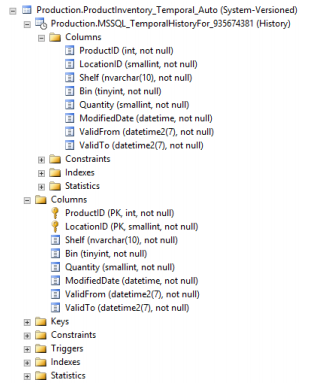
Để giải quyết các nhu cầu khác nhau này để xử lý các thay đổi đối với dữ liệu, SQL Server 2016 hiện hỗ trợ temporal tables, được giới thiệu như một tiêu chuẩn mới trong ANSI SQL 2011. Ngoài ra, Transact-SQL đã được mở rộng để hỗ trợ việc tạo các bảng tạm thời và truy vấn của các bảng này liên quan đến một thời điểm cụ thể.

Temporal tables cho phép bạn tìm trạng thái dữ liệu tại bất kỳ thời điểm nào. Khi bạn tạo một bảng thời gian, hệ thống thực sự tạo ra hai bảng. Một bảng là bảng hiện tại (còn được gọi là bảng thời gian), và bảng còn lại là bảng lịch sử. Theo mặc định, bảng lịch sử được tạo dưới dạng bảng nén trang để giảm việc sử dụng bộ nhớ. Khi dữ liệu thay đổi trong bảng hiện tại, công cụ cơ sở dữ liệu lưu trữ một bản sao của dữ liệu như trước khi thay đổi trong bảng lịch sử.



Hình 24: History Table

Việc sử dụng bảng thời gian có một số hạn chế. Đầu tiên, lập phiên bản hệ thống và các tính năng FileTable và FILESTREAM không tương thích. Thứ hai, bạn không thể sử dụng tùy chọn CASCADE khi bảng tạm thời là bảng tham chiếu trong mối quan hệ khóa ngoại. Cuối cùng, bạn không thể sử dụng kích hoạt INSTEAD OF trên bảng hiện tại hoặc lịch sử, mặc dù bạn có thể sử dụng kích hoạt AFTER trên bảng hiện tại.

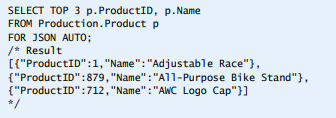


Hình 25: Một temporal table và bảng lịch sử của nó được hiển thị trong SSMS

## JSON

Với việc bổ sung hỗ trợ JSON cho SQL Server 2016, giờ đây bạn có thể hỗ trợ cả cấu trúc dữ liệu quan hệ và không quan hệ trong môi trường của mình và chọn mô hình thích hợp cho yêu cầu dữ liệu và khối lượng công việc của bạn. Sử dụng các hàm và toán tử T-SQL mới, bạn có thể dễ dàng xuất dữ liệu quan hệ sang JSON hoặc phân tích cú pháp và nhập JSON vào bộ lưu trữ quan hệ, ngay cả khi bạn đang làm việc với OLTP trong bộ nhớ, bảng quan hệ cột hoặc bảng quan hệ tạm thời. Bạn cũng có thể viết các truy vấn kết hợp kết quả của dữ liệu quan hệ và JSON. Làm việc với JSON trong SQL Server cũng giống như làm việc với XML, ngoại trừ JSON không phải là một kiểu dữ liệu.

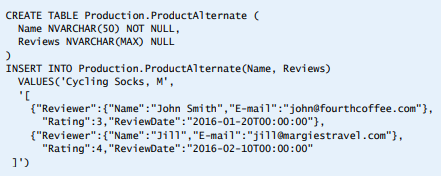
### Xuất dữ liệu dưới dạng JSON



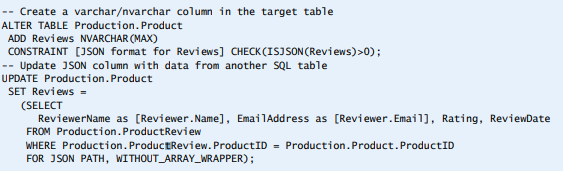
Hình 26: Exporting data to JSON

|  |  |  |
| --- | --- | --- |
| Category | SQL data type | JSON data type |
| Character | Char  Nchar  Varchar  Nvarchar | String |
| Numeric | Int  Bigint  Float  Decimal  Numeric | Number |
| Bit | Bit | boolean |
| Date and time | Date  Datetime  Datetime2  Time  Datetimeoffset | String |
| Binary | Binary | BASE64 encoded string |
| CLR | CLR geometry geography | Not supported (return error) |
| Other | Uniqueindentifier money | String |

### Nhập dữ liệu JSON



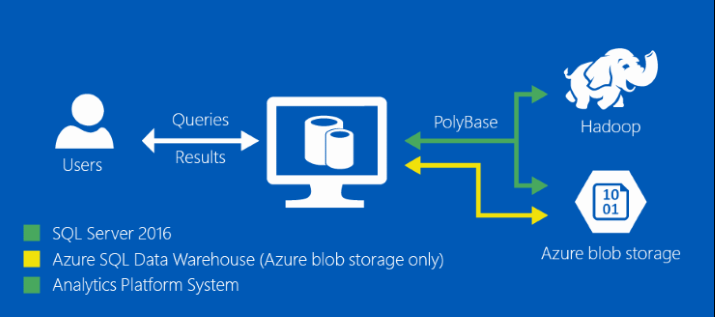
Hình 27: Nhập dữ liệu JSON dưới dạng từ khóa



Hình 28: Nhập dữ liệu JSON dưới dạng kí tự

## PolyBase[[38]](#footnote-38)

PolyBase đã được giới thiệu trong SQL Server 2014 dưới dạng giao diện dành riêng cho Microsoft Analytics Platform System[[39]](#footnote-39) (ABPS; trước đây gọi là Parallel Data Warehouse), nhờ đó bạn có thể truy cập dữ liệu được lưu trữ trong Hệ thống tệp phân tán Hadoop[[40]](#footnote-40) (HDFS) bằng cách sử dụng cú pháp SQL trong truy vấn. Trong SQL Server 2016, bây giờ bạn có thể sử dụng PolyBase để truy vấn dữ liệu trong Hadoop hoặc Azure Blob Storage và kết hợp kết quả với dữ liệu quan hệ được lưu trữ trong SQL Server. Để đạt được hiệu suất tối ưu, PolyBase có thể tự động tạo bảng cột lưu trữ, song song trích xuất dữ liệu từ các nguồn Hadoop và Azure hoặc đẩy các tính toán trên dữ liệu dựa trên Hadoop đến các cụm Hadoop khi cần thiết. Sau khi bạn cài đặt dịch vụ PolyBase và định cấu hình các đối tượng dữ liệu PolyBase, người dùng và ứng dụng của bạn có thể truy cập dữ liệu từ các nguồn không tương quan mà không cần bất kỳ kiến thức đặc biệt nào về lưu trữ Hadoop hoặc blob.



Hình 29: PolyBase

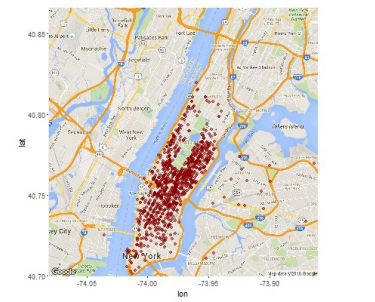
# More analytics

## Tabular enhancements

* Có thể truy cập nhiều nguồn dữ liệu hơn trong chế độ DirectQuery
* Lựa chọn sử dụng tất cả, một số hoặc không có dữ liệu trong khi lập mô hình ở chế độ DirectQuery
* Bảng tính toán
* Lọc chéo hai chiều
* Cải tiến thanh công thức
* Chức năng Biểu thức phân tích dữ liệu (DAX) mới
* Using DAX variables

## R integration

R là ngôn ngữ lập trình mã nguồn mở phổ biến được các nhà khoa học dữ liệu, nhà thống kê và nhà phân tích dữ liệu sử dụng để phân tích nâng cao, khám phá dữ liệu và học máy. Mặc dù sự phổ biến của nó, việc sử dụng R trong môi trường doanh nghiệp có thể là một thách thức. Nhiều công cụ dành cho R hoạt động trong môi trường máy tính đơn luồng, giới hạn bộ nhớ, điều này đặt ra những hạn chế về khối lượng dữ liệu mà bạn có thể phân tích. Ngoài ra, việc di chuyển dữ liệu nhạy cảm từ môi trường máy chủ sang màn hình sẽ xóa dữ liệu đó khỏi các biện pháp kiểm soát bảo mật được tích hợp trong cơ sở dữ liệu. SQL Server R Services, kết quả của việc Microsoft mua lại Revolution Analytics vào năm 2015, giải quyết những thách thức này bằng cách tích hợp bản phân phối R duy nhất vào nền tảng SQL Server. Bạn có thể thực thi mã R trực tiếp trong cơ sở dữ liệu SQL Server khi sử dụng SQL Server R Services (Trong cơ sở dữ liệu) và sử dụng lại mã trong một nền tảng khác, chẳng hạn như Hadoop. Ngoài ra, khối lượng công việc chuyển từ máy tính để bàn sang máy chủ và duy trì mức độ bảo mật cần thiết cho dữ liệu của bạn. Trong Enterprise Edition, R Services thực hiện các phép tính đa xử lý đa luồng, đa lõi và song song với tốc độ cao. Sử dụng SQL Server R Services, bạn có thể xây dựng các ứng dụng dự đoán, thông minh mà bạn có thể dễ dàng triển khai vào sản xuất.



Hình 30: Xem ô bản đồ được tạo bằng cách thực thi chức năng mapPlot tùy chỉnh.

# Better reporting

## Paginated report development enhancements

## Mobile report development

## KPI development

## Report access enhancements

# Improved Azure SQL Database

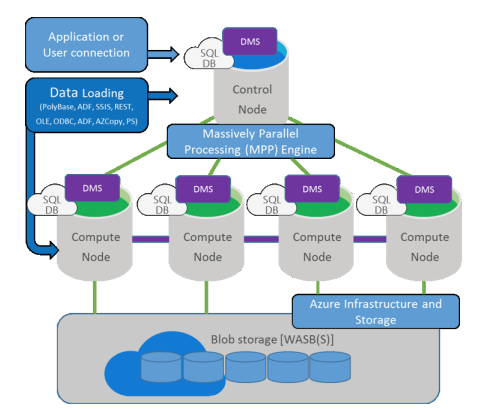
* Parallel queries
* Table partitioning
* Online reindexing
* XML indexes
* CLR integration
* Change tracking for data changes
* Columnstore indexes
* T-SQL windowing functions
* Over 100 new DMVs

# Introducing Azure SQL Data Warehouse

SQL Data Warehouse thêm tính năng massively parallel processing - xử lý song song hàng loạt (MPP) đem lại hiệu quả về chi phí và có sẵn trong Azure. Nó hỗ trợ quy mô petabyte và cũng có thể tăng và giảm quy mô khi khối lượng công việc của bạn thay đổi. Trong khi Cơ sở dữ liệu SQL được thiết kế cho các hệ thống OLTP và kho dữ liệu tổ chức nhỏ, SQL Data Warehouse được xây dựng và tối ưu hóa cho các hệ thống có ít nhất một terabyte dữ liệu, với khả năng mở rộng quy mô vượt xa mức đó. Nó sử dụng các tối ưu hóa hiệu suất tương tự được tích hợp trong APS cho phép các truy vấn mở rộng quy mô và tận dụng lợi thế của việc tính toán song song các truy vấn phân tích, phức tạp.

Có bốn thành phần chính trong kiến ​​trúc SQL Data Warehouse:

* **Control node:** Bạn kết nối với SQL Data Warehouse thông qua Control node, cho dù bạn đang sử dụng các công cụ tải dữ liệu, phát triển hoặc thông minh kinh doanh. Control node là Cơ sở dữ liệu SQL điều phối sự di chuyển dữ liệu và khối lượng công việc tính toán trong hệ thống. Control node nhận các truy vấn và chia chúng thành các đơn vị công việc để chia cho các Compute nodes.
* **Compute nodes:** Compute nodes là Cơ sở dữ liệu SQL riêng biệt cung cấp sức mạnh tính toán cho SQL Data Warehouse. Control node phân phối dữ liệu đồng đều trên các Compute nodes khi tải dữ liệu mới vào kho. Đối với các hoạt động khác, Control node chỉ định hoạt động cho từng cơ sở dữ liệu riêng lẻ dựa trên dữ liệu mà nó chứa. Mỗi Compute nodes trả kết quả từng phần của nó trở lại Control node, nút này tập hợp các kết quả cuối cùng.
* **Storage:** Tất cả dữ SQL Data Warehouse được lưu trữ trong Azure Blob Storage. Dịch vụ lưu trữ dựa trên đối tượng này cung cấp quy mô gần như vô hạn và minh bạch cho phép nhà kho mở rộng quy mô hoạt động lưu trữ tách biệt với hoạt động tính toán. Một lợi ích bổ sung là khả năng tạm dừng kho dữ liệu của bạn và duy trì dữ liệu trong bộ lưu trữ blob. Điều đó có nghĩa là bạn chỉ trả tiền cho các tài nguyên máy tính khi chúng được sử dụng. Hơn nữa, bởi vì lưu trữ blob có khả năng chịu lỗi cao, nó cung cấp các lớp bảo vệ bổ sung cho dữ liệu của bạn.
* **Data movement services:** Dịch vụ di chuyển dữ liệu - Data movement services (DMS) di chuyển dữ liệu giữa các nút một cách minh bạch theo yêu cầu để đáp ứng các truy vấn của người dùng. Mặc dù đây là một dịch vụ nền, bạn có thể nhận thấy các kế hoạch thực thi chứa các hoạt động DMS.



Hình 31: SQL Data Warehouse architecture

1. <https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/introduction-to-memory-optimized-tables?view=sql-server-ver15> [↑](#footnote-ref-1)
2. <https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/creating-and-managing-storage-for-memory-optimized-objects?view=sql-server-2016> [↑](#footnote-ref-2)
3. <https://aboutsqlserver.com/2013/10/15/sql-server-storage-engine-data-pages-and-data-rows/> [↑](#footnote-ref-3)
4. <https://aboutsqlserver.com/2016/09/27/lob-and-row-overflow-storage-in-in-memory-oltp-in-sql-server-2016/> [↑](#footnote-ref-4)
5. <https://aboutsqlserver.com/2013/11/05/sql-server-storage-engine-lob-storage/> [↑](#footnote-ref-5)
6. <https://en.wikipedia.org/wiki/B-tree> [↑](#footnote-ref-6)
7. <https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/transparent-data-encryption?view=sql-server-2016> [↑](#footnote-ref-7)
8. <https://www.sqlshack.com/database-checkpoints-enhancements-sql-server-2016/> [↑](#footnote-ref-8)
9. <https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-procedure-transact-sql?view=sql-server-2016> [↑](#footnote-ref-9)
10. <https://freetuts.net/gioi-thieu-stored-procedure-trong-sql-server-1609.html> [↑](#footnote-ref-10)
11. <https://docs.microsoft.com/en-us/sql/relational-databases/in-memory-oltp/scalar-user-defined-functions-for-in-memory-oltp?view=sql-server-2016> [↑](#footnote-ref-11)
12. <https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-table-transact-sql?view=sql-server-2016> [↑](#footnote-ref-12)
13. <https://docs.microsoft.com/en-us/sql/relational-databases/triggers/dml-triggers?view=sql-server-2016> [↑](#footnote-ref-13)
14. <http://www.kodyaz.com/sql-server-2014/nullable-columns-with-indexes-on-memory-optimized-tables.aspx> [↑](#footnote-ref-14)
15. <https://docs.microsoft.com/en-us/sql/relational-databases/collations/collation-and-unicode-support?view=sql-server-2016> [↑](#footnote-ref-15)
16. <https://butchiso.com/2013/07/character-set-va-collation-trong-mysql.html> [↑](#footnote-ref-16)
17. <https://docs.microsoft.com/en-us/sql/relational-databases/tables/unique-constraints-and-check-constraints?view=sql-server-2016> [↑](#footnote-ref-17)
18. <https://www.mssqltips.com/sqlservertip/5169/sql-server-parallelism-overview/> [↑](#footnote-ref-18)
19. <http://www.infotopie.nl/blog/perf/bottlenecks> [↑](#footnote-ref-19)
20. <https://www.red-gate.com/simple-talk/sql/learn-sql-server/understanding-and-using-parallelism-in-sql-server/> [↑](#footnote-ref-20)
21. <https://www.sqlteam.com/articles/multiple-active-result-sets-mars> [↑](#footnote-ref-21)
22. <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/multiple-active-result-sets-mars> [↑](#footnote-ref-22)
23. <https://docs.microsoft.com/en-us/sql/ssms/visual-db-tools/design-tables-visual-database-tools?view=sql-server-ver15> [↑](#footnote-ref-23)
24. <https://docs.microsoft.com/en-us/sql/relational-databases/security/dynamic-data-masking?view=sql-server-2016> [↑](#footnote-ref-24)
25. <https://quantrimang.com/thiet-lap-sql-server-luon-san-sang-74498> [↑](#footnote-ref-25)
26. <https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/always-on-availability-groups-sql-server?view=sql-server-2016> [↑](#footnote-ref-26)
27. <https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/basic-availability-groups-always-on-availability-groups?view=sql-server-2016> [↑](#footnote-ref-27)
28. <https://docs.microsoft.com/en-us/windows-server/security/group-managed-service-accounts/group-managed-service-accounts-overview> [↑](#footnote-ref-28)
29. <https://docs.microsoft.com/en-us/sql/sql-server/failover-clusters/windows/always-on-failover-cluster-instances-sql-server?view=sql-server-2016> [↑](#footnote-ref-29)
30. <https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-server-diagnostics-transact-sql?view=sql-server-2016> [↑](#footnote-ref-30)
31. <https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/sql-server-always-on-database-health-detection-failover-option?view=sql-server-2016> [↑](#footnote-ref-31)
32. <https://docs.microsoft.com/en-us/powershell/module/msdtc/?view=win10-ps> [↑](#footnote-ref-32)
33. <https://docs.microsoft.com/en-us/sql/relational-databases/databases/tempdb-database?view=sql-server-2016> [↑](#footnote-ref-33)
34. <https://dangvinhcuong.com/2015/02/28/tempdb-trong-microsoft-sql-la-gi-co-loi-ich-gi/> [↑](#footnote-ref-34)
35. <https://docs.microsoft.com/en-us/sql/relational-databases/performance/monitoring-performance-by-using-the-query-store?view=sql-server-2016> [↑](#footnote-ref-35)
36. <https://docs.microsoft.com/en-us/sql/sql-server/stretch-database/stretch-database?view=sql-server-2016> [↑](#footnote-ref-36)
37. <https://docs.microsoft.com/en-us/sql/relational-databases/tables/temporal-tables?view=sql-server-2016> [↑](#footnote-ref-37)
38. <https://docs.microsoft.com/en-us/sql/relational-databases/polybase/polybase-guide?view=sql-server-2016> [↑](#footnote-ref-38)
39. <https://www.microsoft.com/en-us/sql-server/analytics-platform-system> [↑](#footnote-ref-39)
40. <https://dzone.com/articles/sql-and-hadoop> [↑](#footnote-ref-40)