Microsoft

intel

SQL Server 2016 新機能による インメモリ データベースのセキュリティと可用性の向上

- ミッション クリティカル パフォーマンスの提供 -

Edifist Learning Inc.

沖 要知

Microsoft Partner

Microsoft

Gold Learning
Silver Data Analytics
Silver Data Platform
Silver Software Asset Management

はじめに

- コース概要
  - 本セミナーでは、新しい SQL Server で進化したインメモリ テクノロジ、データベースを保護するためのセキュリティ機能、ミッション クリティカルなビジネスデータを保護するための高可用性機能を活用したデータベースソリューション構築のポイントについてデモを交えて、ご紹介します。

- Agenda
  - プラットフォームの変遷
  - SQL Server 2016 への移行
  - SQL Server 2016 新機能導入ガイド

データベース プラットフォームの変遷

- 午前の内容の復習 -

- Intel Xeon プロセッサーの変遷
- Windows Server の変遷
- SQL Server の進化

Intel Xeon プロセッサー E シリーズ ファミリの変遷

- E7 v4 ファミリで 24 TB までの物理メモリ空間が使用可能に

Xeon ファミリの種類	ソケット数	最大メモリ	対象サーバー
Xeon E3	1	32 GB	ワークステーション、エントリー サーバー
Xeon E5	2	1.5 TB	メイン ストリーム サーバー
Xeon E7	4～8	24 TB	ミッション クリティカル サーバー、リアルタイム データ分析

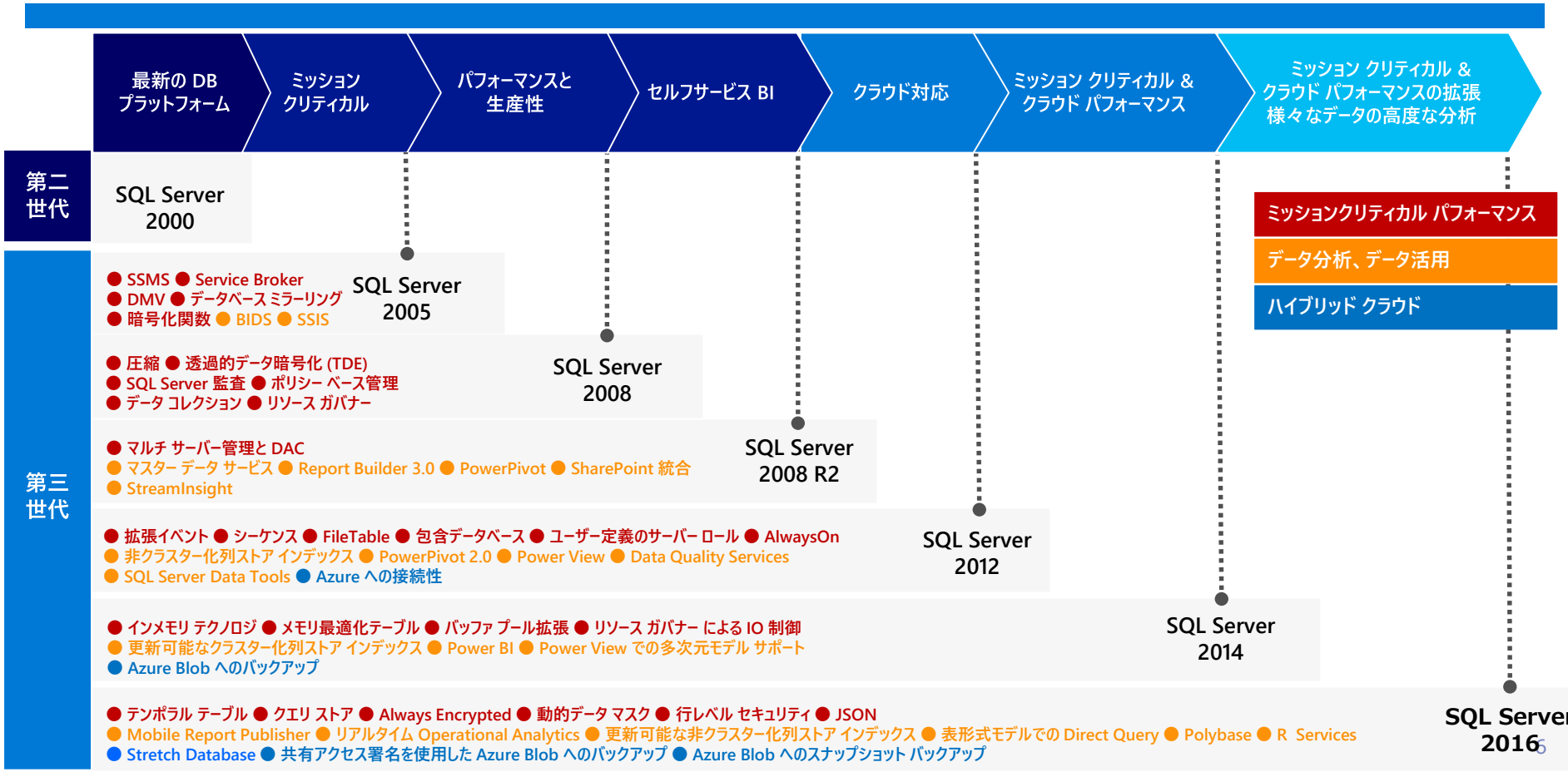
	2010	2011	2012	2013	2014	2016
プロセス ルール マイクロ アーキテクチャ	32 nm Westmere		32 nm Sandy Bridge	22 nm Ivy Bridge	22 nm Haswell	14 nm Broadwell	Skylake
Xeon E5 系	Xeon X5600		Xeon E5-2600	Xeon E5-2600 v2	Xeon E5-2600 v3	Xeon E5-2600 v4	
最大コア数	6		8	12	18	22	
最大メモリ サイズ	288 G		768 G	768 G	1.5 TB	1.5 TB	
最大パス スピード (QPI)	6.4 GT/s		7.2 GT/s	8.0 GT/s	9.6 GT/s	9.6 GT/s	
Xeon E7 系		Xeon E7-8800		Xeon E7-8800 v2	Xeon E7-8800 v3	Xeon E7-8800 v4	
最大コア数		10		15	18	24	
最大メモリ サイズ		4 TB		12 TB	12 TB	24 TB	
最大パス スピード (QPI)		6.4 GT/s		8.0 GT/s	9.6 GT/s	9.6 GT/s	

Microsoft Windows Server の最大メモリ サイズの変遷

- SQL Server 2016 on Windows Server 2016 で 12 TB の物理メモリを使用可能に

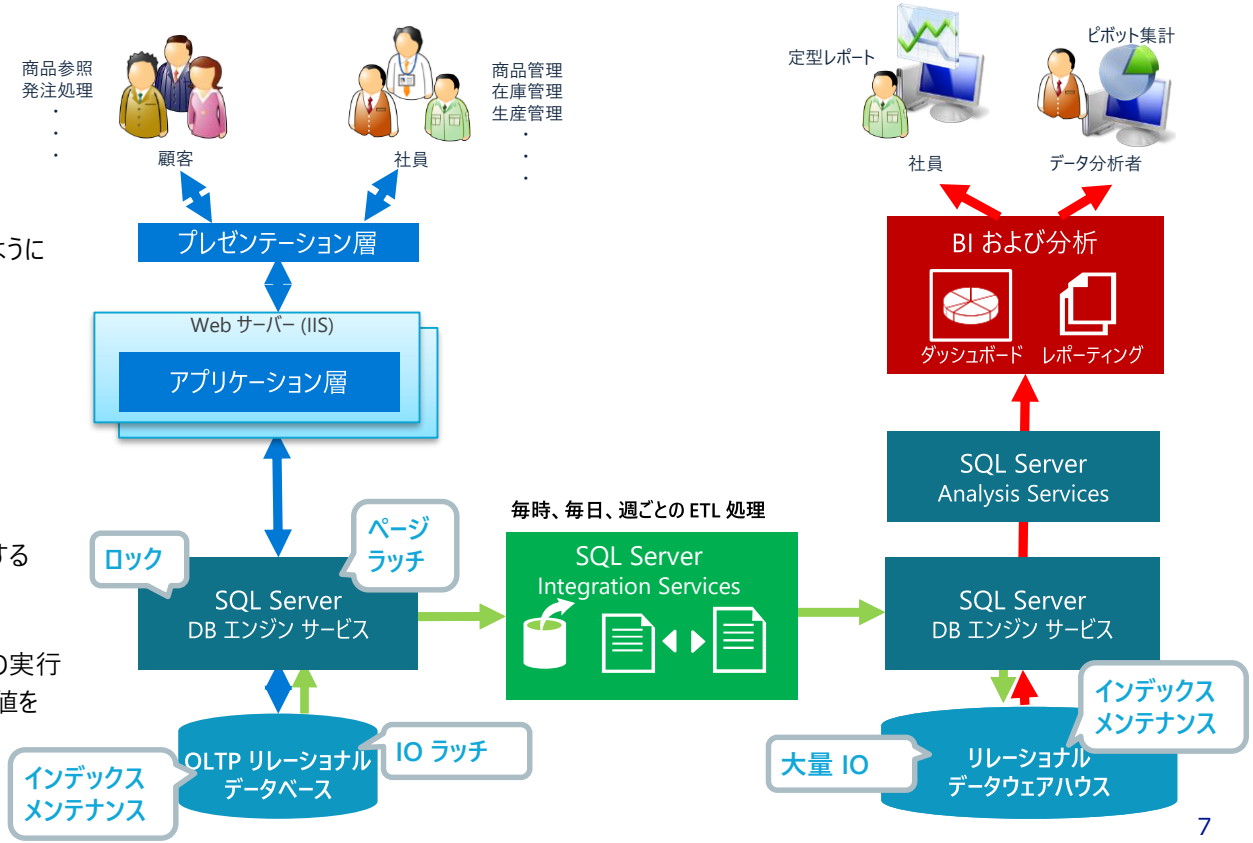
		2008	2009	2012	2013	2016
32 bit	Standard	4 GB				
	Enterprise	32 GB				
	Datacenter	32 GB				
64 bit	Standard	32 GB	32 GB			
	Enterprise	2 TB	2 TB	4 TB	4 TB	12 TB
	Datacenter	2 TB	2 TB	4 TB	4 TB	12 TB

Microsoft SQL Server の進化



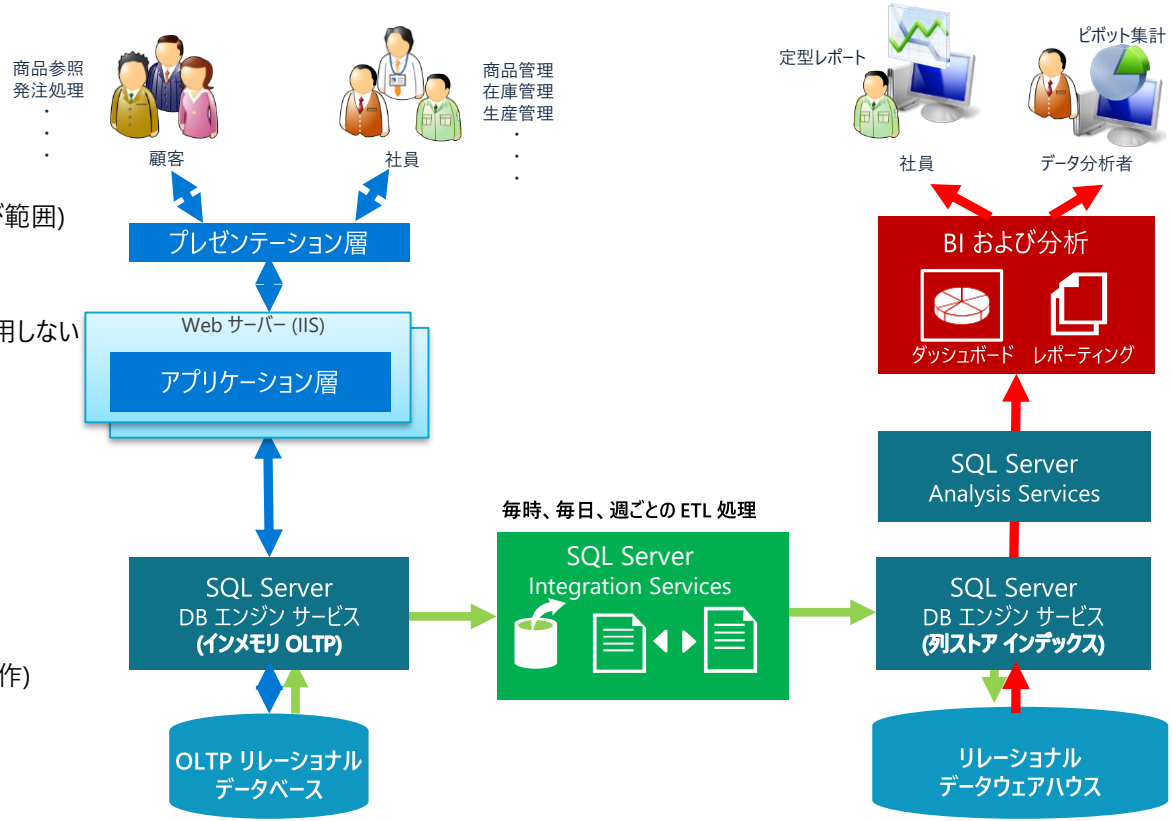
トラディショナルなデータ プラットフォームの課題

- 基幹業務系 (OLTP データベース)
  - 組織での日常業務を支援する技術基盤
  - 絶えず変化するトランザクションの状態を保持 (基本的に履歴は保持しない)
  - トランザクション処理のための最適化
  - 正規化されたテーブル構造 (構造が複雑)
  - 同時実行されるトランザクションを並列処理するように デザインおよびチューニングされる
  - 個々のトランザクションは、短時間で完了し、 アクセスするデータ量は比較的少量
- 情報系 (データウェアハウス)
  - レポーティング、データ分析のためのデータ ソース
  - 定期的に OLTP データベースからロードされる 履歴データを保持し、時間ともにデータ量は増加する
  - ファクト テーブルを中心とするスター スキーマー、 および、スノーフレイク スキーマー構造
  - 一定期間のファクト データを参照する集計クエリの実行
  - アクセスする量が多いため、別途、事前集計した値を 格納している場合もある

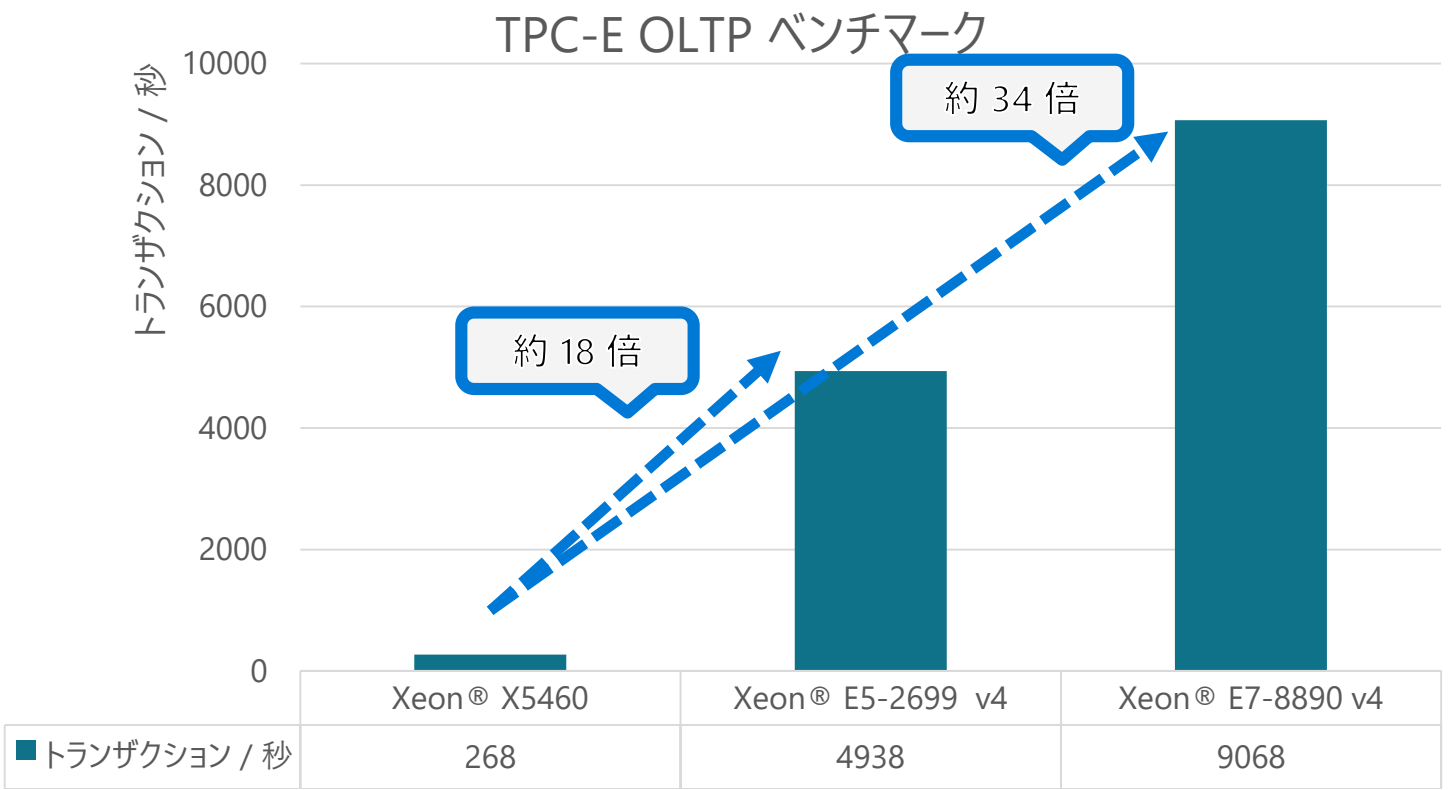


インメモリ テクノロジーによるパフォーマンス向上

- インメモリ OLTP
  - ディスク ベース テーブルから
    メモリ最適化テーブル
  - バッファー プールを使用しない
  - インメモリ データ向けに最適化
  - メモリ内にのみ存在するインデックス (ハッシュおよび範囲)
  - インデックス メンテナンス不要に
  - 永続性を目的としたストリーム ベースのストレージ
  - ロック マネージャー、ラッチ、およびスピン ロックを使用しない
  - マシン語コードにコンパイルされた T-SQL
- 列ストア インデックス
  - B-Tree 形式のインデックスではなく
    列編成されたインデックスで高い圧縮効果
  - 使用するデータのみをメモリにロードすることで
    列データ参照を高速化
  - 並列化されたバッチモードのクエリ処理 (スキャン操作)

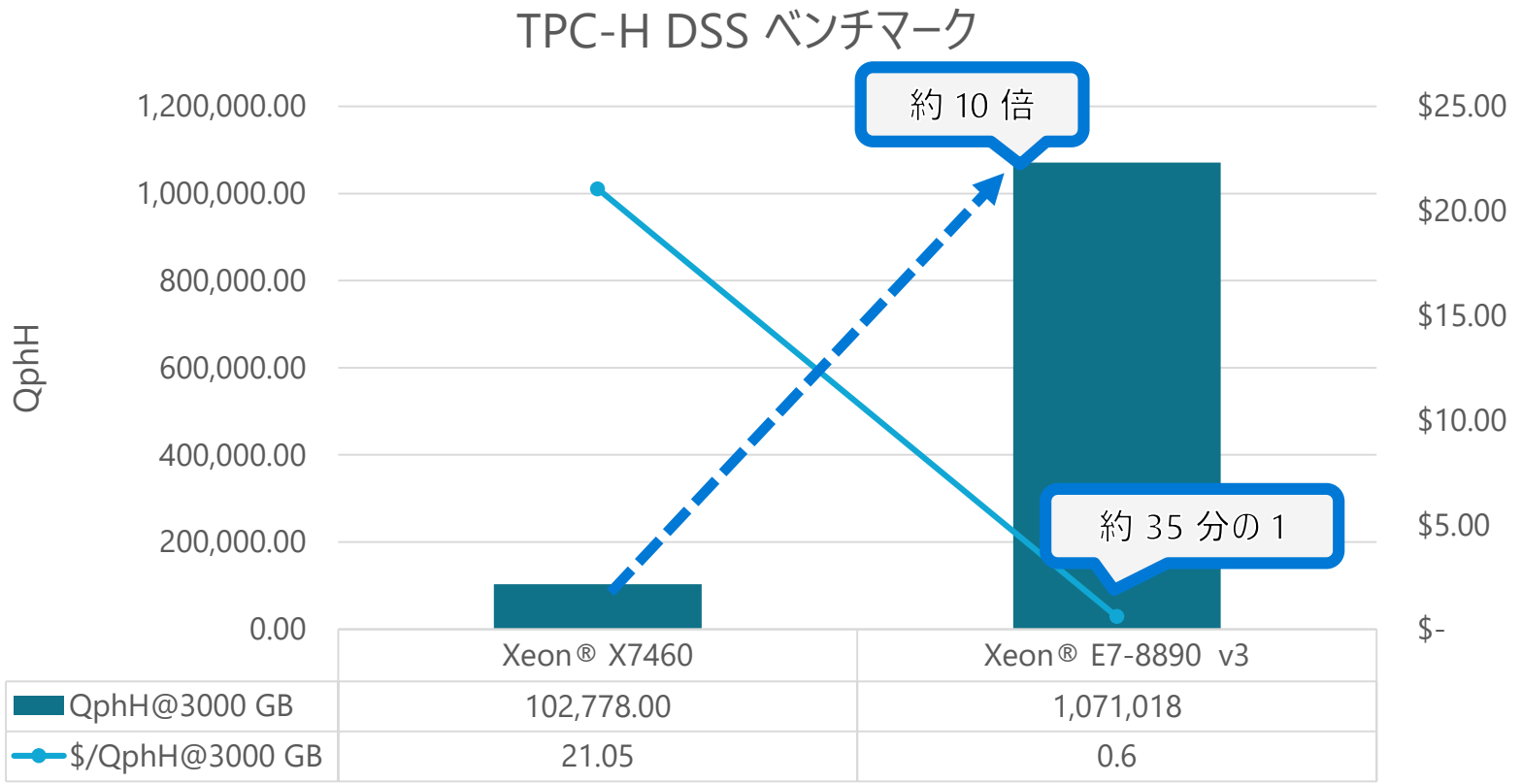


OLTP ベンチマークの比較



出典: Transactional Processing Performance Council (tpc.org)

DSS ベンチマークの比較



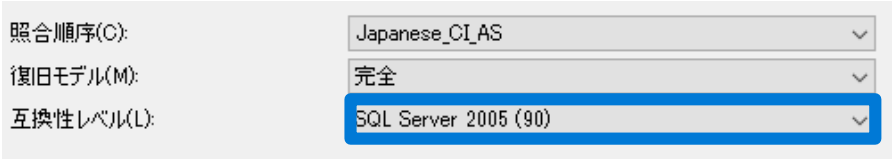
出典: Transactional Processing Performance Council (tpc.org)

SQL Server 2016 への移行

- SQL Server 2016 のエディション
- 以前のバージョンの SQL Server からのデータベース移行

SQL Server 2016 への移行パスと互換性レベル

- データベースのプロパティで互換性レベルを確認



- SQL Server 2005 からはサイド バイ サイドでデータベースをバックアップ、復元して移行後、互換性レベルの変更が必要
  - 互換性レベルが自動で 100 に変更される

バージョン	アップグレード セットアップ	サイド バイ サイド データベース移行	互換性レベル 60	70	80	90	100	110	120	130
SQL Server 2005 SP4 以降		○	○	○	○	○				
SQL Server 2008 SP3 以降	○	○			○	○	○			
SQL Server 2008 R2 SP2 以降	○	○			○	○	○			
SQL Server 2012 SP1 以降	○	○				○	○	○		
SQL Server 2014	○	○					○	○	○	
SQL Server 2016							○	○	○	○

- 互換性レベルの変更による影響は、事前にアップグレード アドバイザーで調査する

参照 https://msdn.microsoft.com/en-us/library/ms143393.aspx

SQL Server 2016 のエディション

- プリンシパル エディション
  - SQL Server 2016 Enterprise
    - SQL Server 2016 の全機能を提供する大規模環境向けエディション
  - SQL Server 2016 Standard
    - 部門や小規模な組織のために基本機能を提供するエディション
- 特別エディション
  - SQL Server 2016 Web
    - 低い TCO で使用できる Web ホスティング企業および Web VAP 向けのエディション
- 無償エディション
  - SQL Server 2016 Developer
    - Enterprise と同等の機能を提供する開発、およびテスト環境でのみで使用可能なエディション
  - SQL Server 2016 Express
    - エントリ レベルの無料のデータベース
    - 学習や、デスクトップおよび小規模サーバー データ ドリブン アプリケーションの構築向き

参照: https://msdn.microsoft.com/ja-jp/library/ms144275.aspx

エディションの比較

			Express	Standard	Enterprise
ライセンス				コア ベース または サーバー + CAL	コア ベース
ミッションクリティカル パフォーマンス	最大コア数	New	4 コア	24 コア	無制限
	インスタンスあたりの最大メモリ		1 GB	128 GB	OS のサポートする最大
	最大サイズ		10 GB	524 PB	524 PB
	基本的な OLTP		●	●	●
	基本的な可用性 (2 ノードの単一データベース フェールオーバー、読み取り不可のセカンダリ		●	●	●
	管理機能 (Management Studio、ポリシー ベースの管理)	New		●	●
	エンタープライズ データ管理 (マスター データ サービス、Data Quality Services)				●
	高度な OLTP (インメモリ OLTP、業務分析)	New			●
	高度な HA (AlwaysOn - マルチ ノード、複数データベースのフェールオーバー、読み取り可能なセカンダリ)				●
セキュリティ	基本的なセキュリティ (行レベル セキュリティ、動的データ マスク、基本的な監査、職掌分散)	New		●	●
	高度なセキュリティ (透過的なデータ暗号化、Always Encrypted)	New			●
データウェアハウス	高度なデータ統合 (あいまい参照変換、あいまいグループ化変換、データ変更キャプチャ)				●
	データ ウェアハウジング (インメモリ列ストア、パーティション分割)	New			●
Business intelligence	プログラミング & 開発者ツール (T-SQL、CLR、データ型、FileTable、JSON)	New	●	●	●
	基本的なデータ統合 (SSIS、組み込みのコネクタ)		●	●	●
	基本的なレポートおよび分析サービス			●	●
	基本的なコーポレート ビジネス インテリジェンス (多次元モデル、基本的な表形式モデル)	New		●	●
	モバイル BI (Datazen)	New			●
	高度なコーポレート ビジネス インテリジェンス (高度な表形式モデル、直接クエリ、インメモリ分析、高度なデータ マイニング)	New			●
高度な分析	“R” との基本的な統合 (R Open への接続、RRE の制限付きの並列処理)	New	●	●	●
	“R” との高度な統合 (RRE の完全な並列処理)	New			●
ハイブリッド クラウド	Stretch Database	New	●	●	●

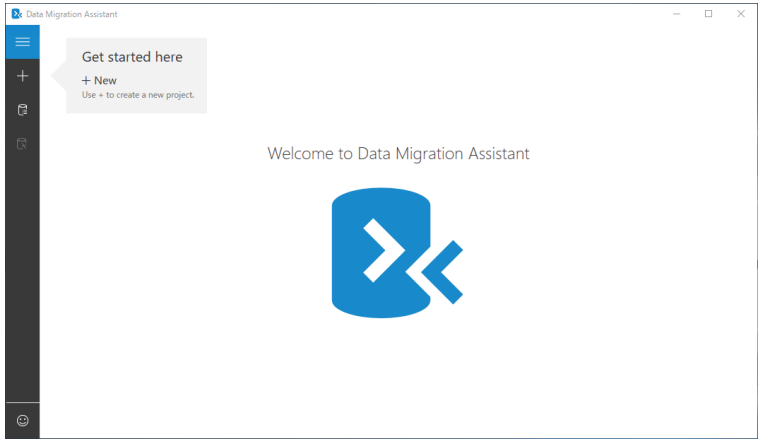
参照: https://msdn.microsoft.com/en-us/library/cc645993.aspx

以前のバージョンの SQL Server からのデータベース移行

- Microsoft Data Migration Assistant
  - アップグレード アドバイザーの後継
  - 変換元: SQL Server 2005、SQL Server 2008、SQL Server 2008 R2、SQL Server 2012、SQL Server 2014、Server 2016
  - 変更先: SQL Server 2012、SQL Server 2014 、SQL Server 2016
- Version 2.0 で 2 種類の機能を提供
  - 移行のアセスメント
  - サイド バイ サイドのデータベース移行

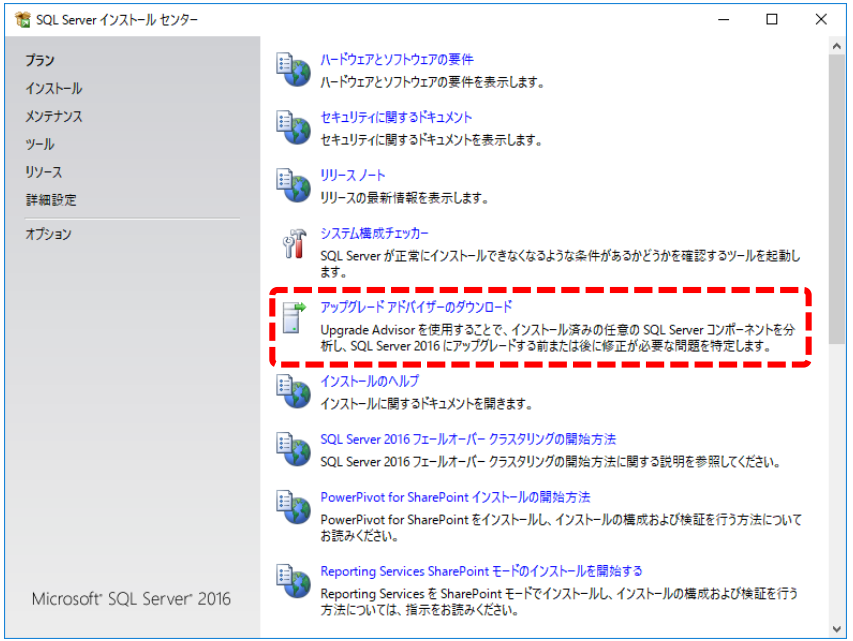
SQL Server 2005
SQL Server 2008
SQL Server 2008 R2
SQL Server 2012
SQL Server 2014
SQL Server 2016

SQL Server 2012
SQL Server 2014
SQL Server 2016



Microsoft Data Migration Assistant のインストール

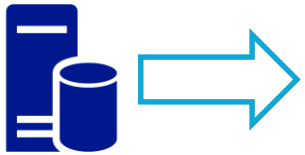
- 前提条件
  - OS (64 Bit)
    - Windows 10、Windows 7、Windows 8、Windows 8.1、Windows Server 2012、Windows 7
  - Microsoft .NET Framework 4.5
    - SQL Server 2016 製品メディア、または、ダウンロード ページ から入手可能
- 最新の Microsoft Data Migration Assistant の入手方法
  - インストール メディアのメニューでは、「アップグレード アドバイザー」と表現されている
  - インストール メディアには同梱されないため、Download Center から、最新の DataMigrationAssistant.msi ファイルを取得する



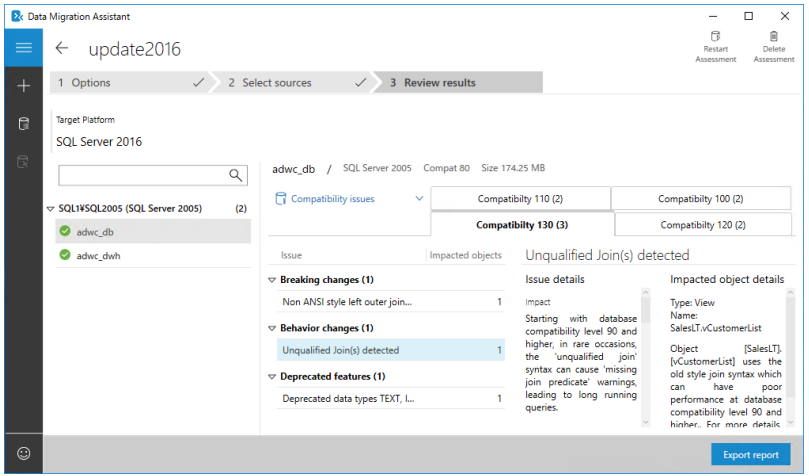
参照：https://msdn.microsoft.com/en-us/library/mt147402.aspx

Microsoft Data Migration Assistant による移行アセスメント

- 以前のバージョンの SQL Server に作成されている
  リレーショナル データベースを分析して修正の必要がある問題のレポート生成:
  - Breaking changes
  - Behavior changes
  - Deprecated features
- SQL Server の移行後、使用が推奨される機能の提案:
  - Performance
  - Security
  - Storage

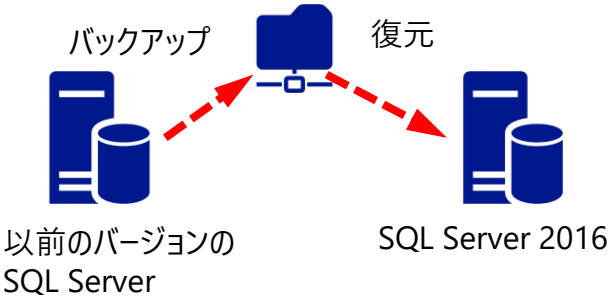


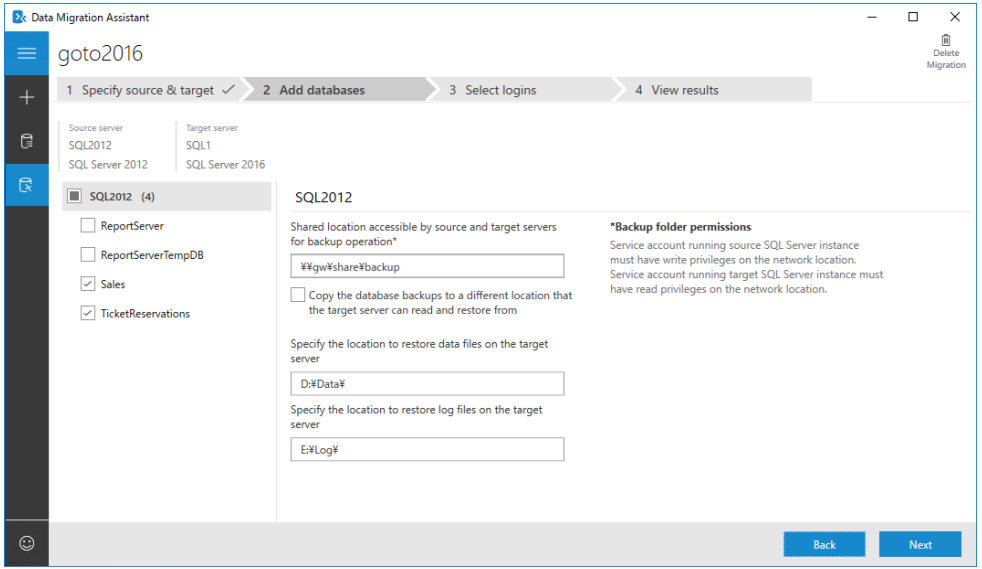
以前のバージョンの SQL Server



Microsoft Data Migration Assistant によるデータベース移行

- 共有フォルダーを介したバックアップ・復元により、データベースとログインの移行を支援





[参考] SQL Server Migration Assistant (SSMA)

- SQL Server 以外のデータベース製品からの移行では、SQL Server Migration Assistant (SSMA) を使用する
  - Microsoft Access、DB2、MySQL、Oracle、Sybase から SQL Server へのデータベース移行を自動化
  - SQL Server Migration Assistant for Access
  - SQL Server Migration Assistant for DB2
  - SQL Server Migration Assistant for MySQL
  - SQL Server Migration Assistant for Oracle
  - SQL Server Migration Assistant for Sybase
- Microsoft SQL Server Migration Assistant (SSMA) のヘルプとサポート
  - MSDN ドキュメントと製品ヘルプ
  - SSMA サポート エイリアス (ssmahelp@microsoft.com)
  - Microsoft プレミア サポート (https://support.microsoft.com/ja-jp)
  - SQL Server コミュニティ フォーラム (https://msdn.microsoft.com/ja-jp/sqlserver/bb671050.aspx)

参考：https://msdn.microsoft.com/ja-jp/library/mt613434.aspx

Microsoft

intel

SQL Server 2016 新機能導入ガイド

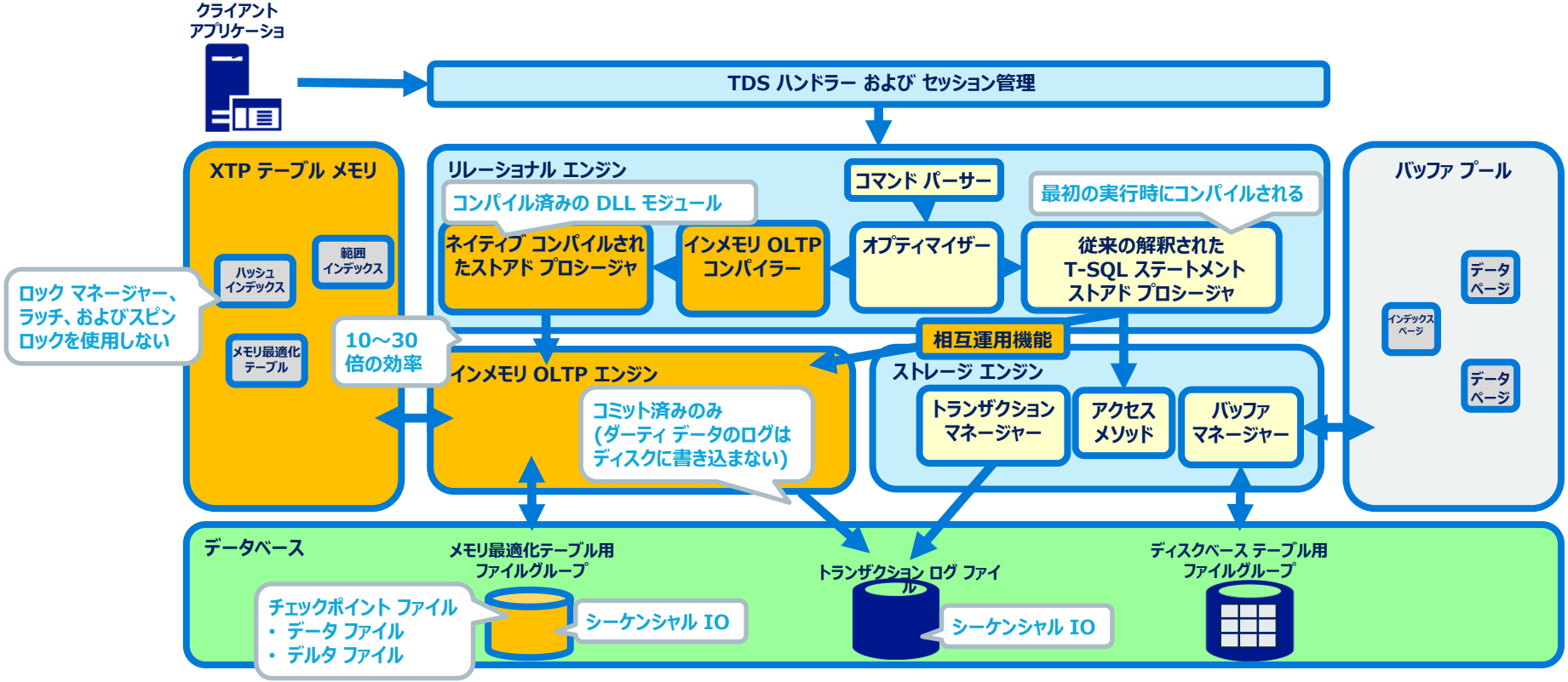
- インメモリ OLTP の強化
- リアルタイム分析を可能にする更新可能な列ストア インデックス
- データベース セキュリティの強化
- データベースの可用性向上

インメモリ OLTP の必要性と利点

利点	高パフォーマンスのデータ操作	スムーズなスケール アップ	効率の良い ビジネス ロジック処理	ハイブリッド エンジンおよび 統合されたエクスペリエンス
インメモリ OLTP の技術的要素	**メイン メモリの最適化** • バッファー プールを使用しない • インメモリ データ向けに最適化 • メモリ内にのみ存在するインデックス (ハッシュおよび範囲) • インデックス メンテナンス不要に • 永続性を目的としたストリーム ベースのストレージ	**高い同時実行性** • マルチバージョン オプティミスティック同時実行制御 (完全な ACID サポート) • コア エンジンはロックなしのアルゴリズムを使用 • ロック マネージャー、ラッチ、およびスピン ロックを使用しない	**マシン語コードにコンパイルされた T-SQL** • C コード ジェネレータおよび VC でマシン語コードにコンパイルされた T-SQL • プロシージャ呼び出しは単なる DLL エントリ ポイント • コンパイル時におけるアグレッシブな最適化	**SQL Server の統合** • 同じ管理しやすさ、管理と開発のエクスペリエンス • 統合されたクエリおよびトランザクション • 統合された HA および バックアップ/復元
背景	**ハードウェアの傾向** メモリ価格の低下 次世代 SSD 接続規格 NVMe	プロセッサー コア数の増加	CPU クロック数の停滞	**ビジネス** TCO

インメモリ OLTP の構成

- メモリ最適化テーブルとネイティブ コンパイルされたストアドプロシージャによる インメモリ OLTP の実現 (SQL Server 2014～)



[参考] インメモリ OLTP データベースへの変更手順

- 既存データベースをインメモリ OLTP の構成する手順
  - メモリ最適化ファイルグループの追加

```
ALTER DATABASE <データベース名>
  ADD FILEGROUP <ファイルグループ名> CONTAINS MEMORY_OPTIMIZED_DATA
```

  - メモリ最適化ファイル グループにデータ ファイルを追加

```
ALTER DATABASE <データベース名>
  ADD FILE (name='<論理ファイル名>', filename='<物理ファイル名>') TO FILEGROUP <ファイルグループ名>
```

  - トランザクション分離レベルの設定

```
ALTER DATABASE <データベース名> SET MEMORY_OPTIMIZED_ELEVATE_TO_SNAPSHOT=ON
```

    - 既定の READ_COMMITED の場合、自動トランザクション モードでのみの実行になるため注意

参照：https://msdn.microsoft.com/ja-jp/library/dn449490(v=sql.120).aspx

[参考] メモリ最適化テーブルの定義

- 2 種類のインデックス
  - ハッシュ インデックス (NONCLUSTERED HASH) / 範囲インデックス (NONCLUSTERED)
- 2 種類のデータの永続化 (DURABILITY) の指定
  - スキーマとデータの永続化 (SCHEMA_AND_DATA) / スキーマのみの永続化 (SCHEMA_ONLY)

```
CREATE TABLE dbo.TicketReservationDetail(
  TicketReservationID bigint NOT NULL
, TicketReservationDetailID bigint IDENTITY(1,1) NOT NULL
, Quantity smallint NOT NULL
, FlightID int NOT NULL
, Comment nvarchar(100) NULL
, CONSTRAINT [PK_TicketReservationDetail] PRIMARY KEY
 NONCLUSTERED HASH (TicketReservationDetailID) WITH (BUCKET_COUNT = 1000000)
) WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA)
GO
```

ハッシュ インデックス

スキーマとデータの永続化を指定

メモリ最適化テーブルとして作成

参照：https://github.com/Microsoft/sql-server-samples/tree/master/samples/features/in-memory/ticket-reservations

[参考] ネイティブ コンパイル ストアドプロシージャの定義

- WITH 句に NATIVE_COMPILATION を指定する
  - セッション設定は作成時に固定となる

```
CREATE PROCEDURE InsertReservationDetails(
   @TicketReservationID int
 , @LineCount int
 , @Comment nvarchar(100)
 , @FlightID int)
WITH NATIVE_COMPILATION, SCHEMABINDING
AS
BEGIN ATOMIC WITH (TRANSACTION ISOLATION LEVEL=SNAPSHOT, LANGUAGE = N'Japanese')
  DECLARE @loop int = 0;
  WHILE (@loop < @LineCount)
  BEGIN
        INSERT INTO dbo.TicketReservationDetail (TicketReservationID, Quantity, FlightID, Comment)
        VALUES(@TicketReservationID, @loop % 8 + 1, @FlightID, @Comment)
        SET @loop += 1
  END
END
GO
```

ネイティブ コンパイルとスキーマ バインドを指定

トランザクション分離レベルの指定

参照：https://msdn.microsoft.com/ja-jp/library/dn169141(v=sql.120).aspx

SQL Server 2014 と 2016 のインメモリ OLTP 機能比較

- インメモリ OLTP の進化

インメモリ OLTP での機能	SQL Server 2014	SQL Server 2016
データベースの最大サイズ	256 GB	OS のメモリサイズ上限 (テーブルはメモリ内に収まる必要がある)
クラスター化列ストア インデックスの作成 (リアルタイム Operational Analytics を可能にする)	×	○
メモリ最適化テーブルでの並列クエリ	×	○ (互換性レベル 130)
透過的暗号化 (TDE) の使用	×	○
LOB データ型 (varbinary(max)、 [n]varchar(max)) の使用	×	○
FOREIEIGN KEY / CHECK / UNIQUE 制約の使用	×	○
ネイティブ コンパイル ストアドプロシージャからユーザー定義スカラー値関数の呼び出し	×	○
インデックス キーに対する BIN2 以外の照合順序の指定	×	○
DML トリガーの使用	×	○
ネイティブ コンパイル ストアドプロシージャでの OUTER JOIN、OR、NOT、UNION [ALL]、 DISTINCT、EXISTS、IN 句の使用	×	○
ALTER PROC / ALTER TABLE ステートメントの使用	DROP / CREATE	○
SSMS テーブル デザイナーの使用	×	○
TRUNCATE TABLE ステートメントの使用	×	× (DELETE を使用)
MERGE ステートメントの使用	×	× (UPDATE、DELETE、INSERT を使用)

列ストア インデックスの構造

- データ ページと 列ストア インデックス ページの比較
  - 行ストア

ページ 1

20050101	8700122	33011
10024	1800	5
50000	20050101	8000998
33011	10024	4500
3	30000	20050101
8700122	12001	10024
1800	8	80000

ページ 2　変更可能

20050102	8700100	2200
10024	1800	
50000	20050102	8000998
33011	10024	4500
3	30000	20050102
8700100	12001	10024
1800	8	80000

- 各ページにキーの順に並べられた行データを格納
- 1 列分のデータがほしい場合でもページ単位でデータを取得するため、不要な列も取得する必要がある
- **小さいデータセットの参照、更新に有利**

…

  - 列ストア

ページ 1 (時間Key)

20050101	20050101	20050101
20050102	20050102	20050102

ページ 2 (製品Key)

8700122	8000998	8700122
8700100	8000998	8700100

ページ 3 (顧客Key)

33011	33011	
22003	33011	

直接の変更は不可

- 各ページは単一の列からのデータを格納
  - 列単位で重複した値が増える傾向がある
- 高い圧縮率
  - ページごとにカージナリティ度を低くすることで、圧縮効率が向上
  - より多くのデータをメモリ内に格納
- 各列は個々にアクセス可能
  - 必要な列だけをフェッチできる
- **大量データ参照時、I/O の劇的な削減が可能**

列ストア編成の種類

- クラスター化列ストア インデックス
  - テーブルに 1 個
    - 行ストアのクラスター化 インデックスがある場合、作成できない
    - すべての行が列編成となる
    - 大きなテーブルでは、パーティションを構成し、パーティション単位で再構築する
- 非クラスター化列ストア インデックス
  - クラスター化インデックスまたはヒープ内の列のサブセットにインデックスを作成
    - クラスター化列ストア インデックスがある場合、作成できない
  - インデックス内の列のコピーを格納する追加ストレージが必要

列ストア インデックスの進化

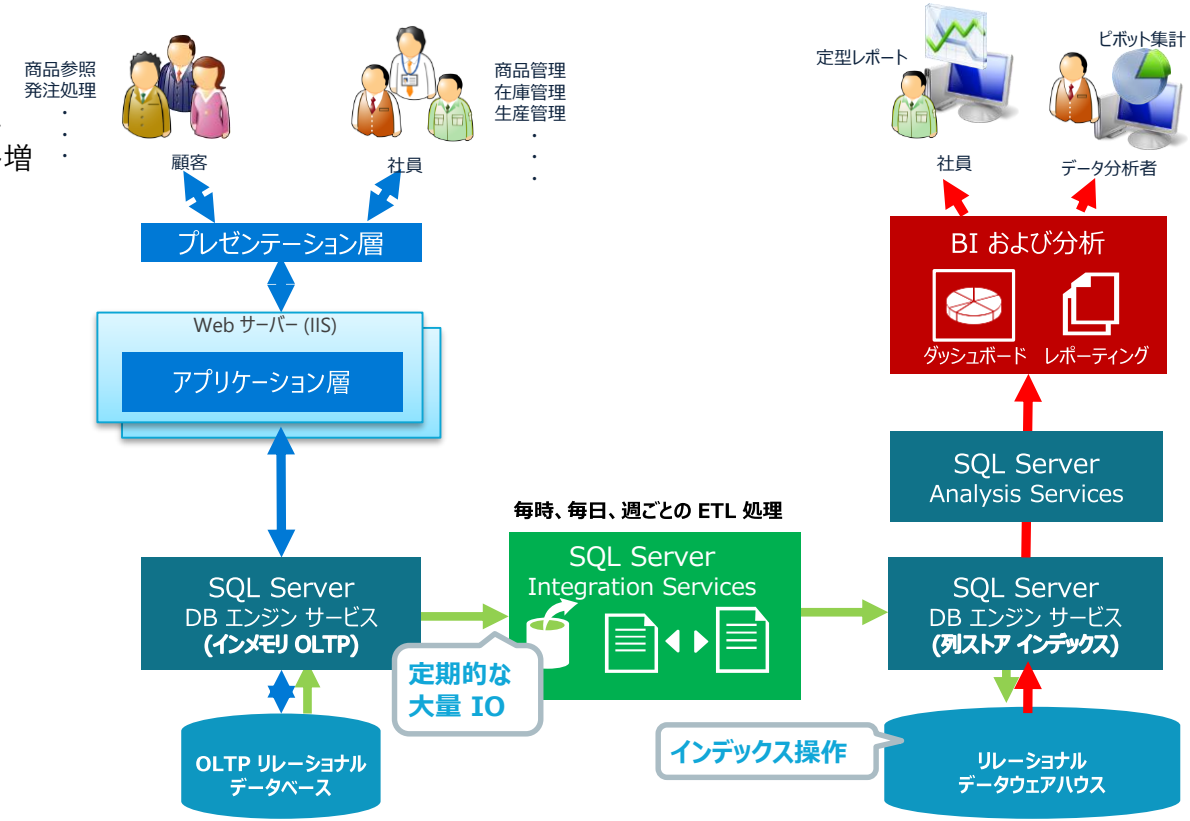
- 更新可能な列ストア インデックスによりリアルタイム分析が可能に

	SQL Server 2012 Enterprise	SQL Server 2014 Enterprise	SQL Server 2016 Enterprise
非クラスター化 列ストア インデックス	• 作成可能に • 更新不可	• 作成可能 • 更新不可	• 作成可能 • 更新可能に • フィルター設定も可能に
クラスター化 列ストア インデックス		• 作成可能に • 更新も可能 • ただし、他の非クラスター化インデックスとの共存は不可	• 作成可能 • 更新可能 • 他の非クラスター化インデックスとの共存も可能に • メモリ最適化テーブルにも作成可能に

- SQL Server 2016 では、どちらのテーブルにも作成が可能
  - ディスク ベース テーブル
  - インメモリ OLTP メモリ最適化テーブル

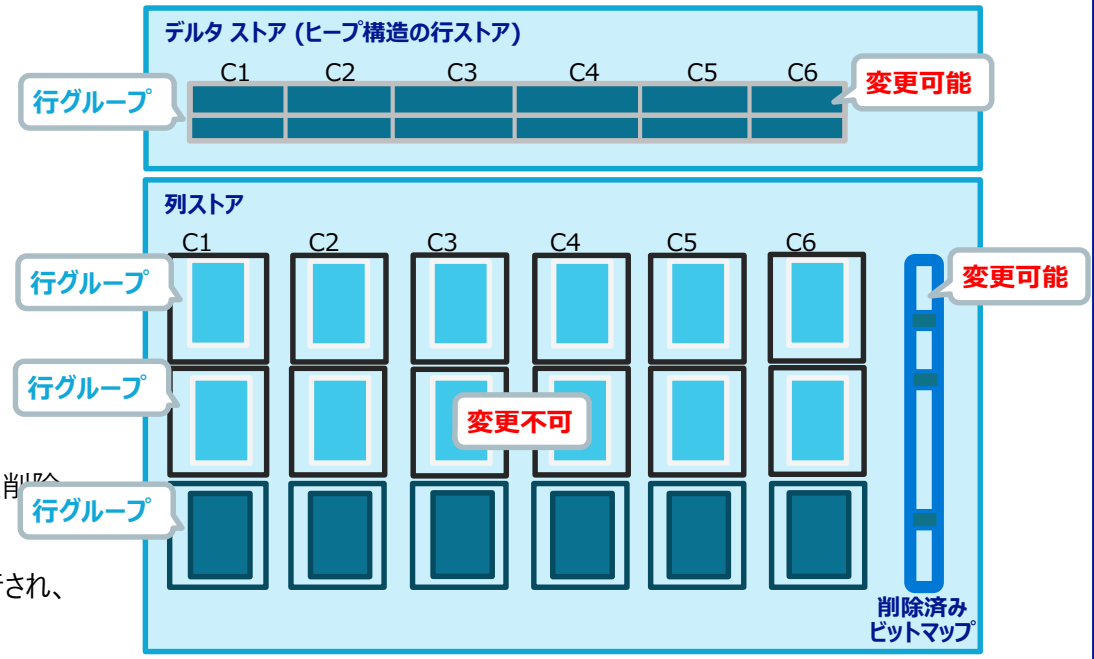
トラディショナルなデータ プラットフォームの課題

- SQL Server 2014 での課題
  - 実装の複雑性
  - ELT 操作による IO 競合とインデックス管理
  - 複数のデータベース サーバー構成によるコスト増
  - 基幹系と情報系のタイム ラグ
  - リアルタイム分析への要求



更新可能な列ストア インデックスに対する DML 操作

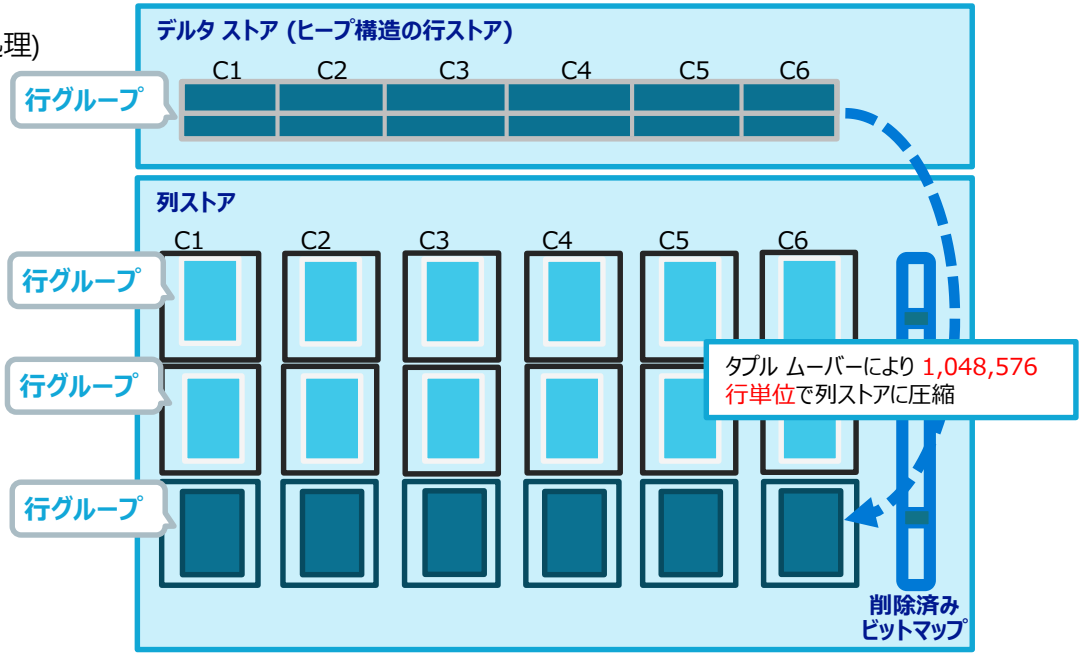
- 更新可能な列ストア インデックスのメカニズム
  - テーブルは行ストアと列ストアで構成される
- NSERT 操作
  - 常にデルタ ストアに挿入される
- DELETE 操作
  - 対象がデルタ ストアにある場合、デルタ ストアから削除
  - 対象が列ストアにある場合、論理削除 (削除済みビットマップにマーク) のみ
- UPDATE 操作
  - 対象がデルタ ストアにある場合、更新操作
  - 対象が列ストアにある場合、デルタ ストアへの挿入と論理削除
- BULK INSERT 操作
  - 10 万行以下のバッチの場合、挿入はデルタ ストアで実行され、10 万行以上の場合、直接、列ストアで実行される
- SELECT 操作
  - 列ストアおよび行ストアのデータを 内部的な UNION 操作で結合する



参照：https://msdn.microsoft.com/ja-jp/library/gg492088(v=sql.120).aspx#performance

タプル ムーバーとインデックス操作

- タプル ムーバー
  - デルタ ストア から 列ストアへのデータ移動 (シングル スレッド処理)
  - セグメントがいっぱい (100 万行) になったデルタ ストアが見つかると “タプル ムーバー” は、データを列編成に変換する
  - 5 分ごとに起動 (トレース フラグで制御可能)
- インデックス操作
  - ALTER INDEX ・・・ REORGANIZE により、タプル ムーバーを任意のタイミングで実行できる
  - ALTER INDEX ・・・ REBUILD により、デルタストアの行ストア データを 列ストアにマージし、テーブルから論理的に削除された行を物理的に削除して列ストアをデフラグできる

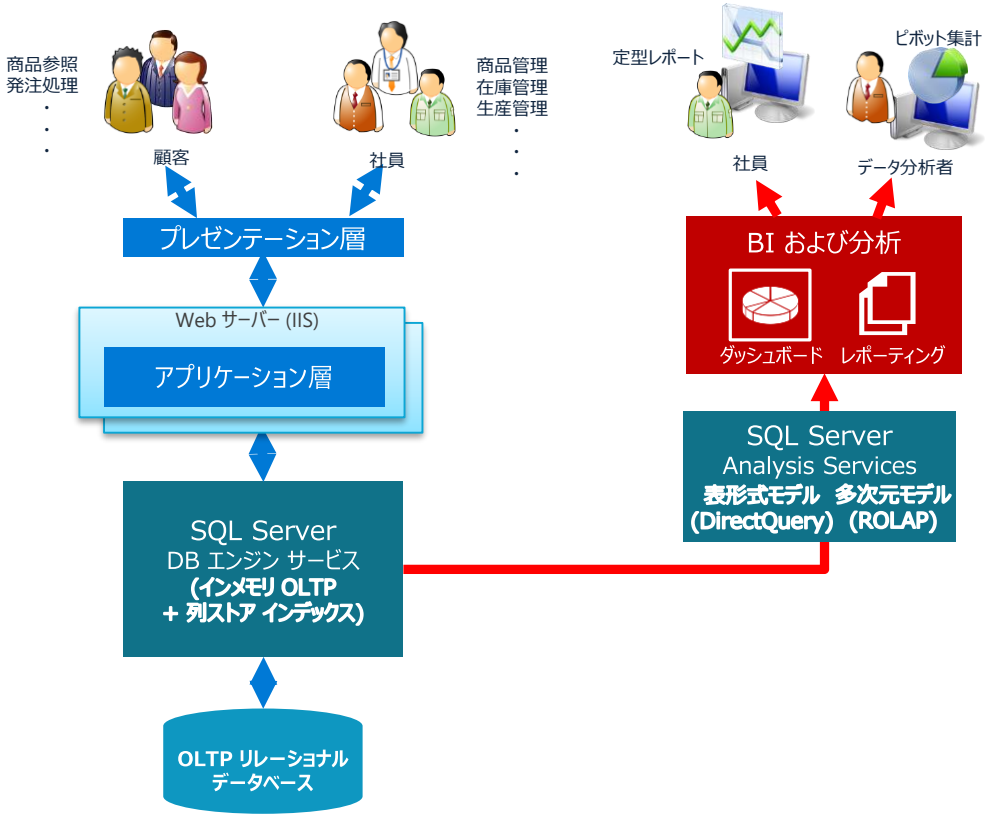


参照：https://msdn.microsoft.com/ja-jp/library/dn589807(v=sql.120).aspx

Operational Analytics による OLTP とデータ分析ストレージの統合

- 使用する機能
  - インメモリ OLTP
    - メモリ最適化テーブル
    - ネイティブコンパイルされたストアド プロシージャ
  - 更新可能な列ストア インデックス
  - テンポラル テーブル

- 利点
  - 遅延がない
  - リレーショナル データウェアハウスが不要
  - ETL 操作が不要



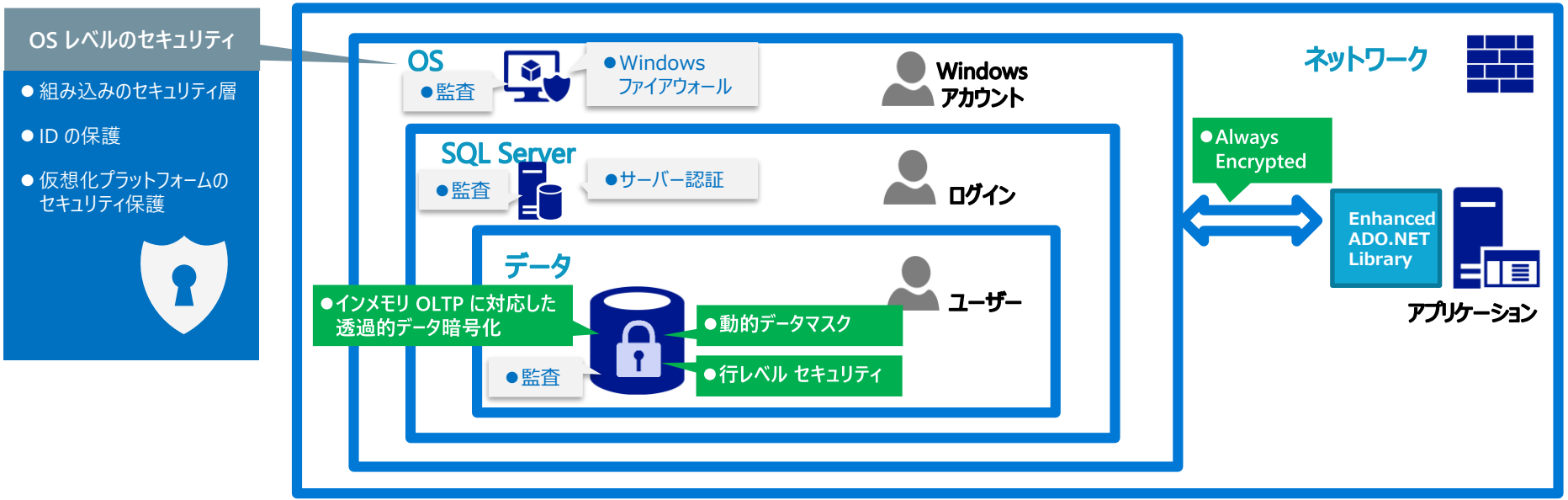
SQL Server 2016 on Windows Server 2016

- Better Together Configurations -

- データベースに対する多層防御の拡張

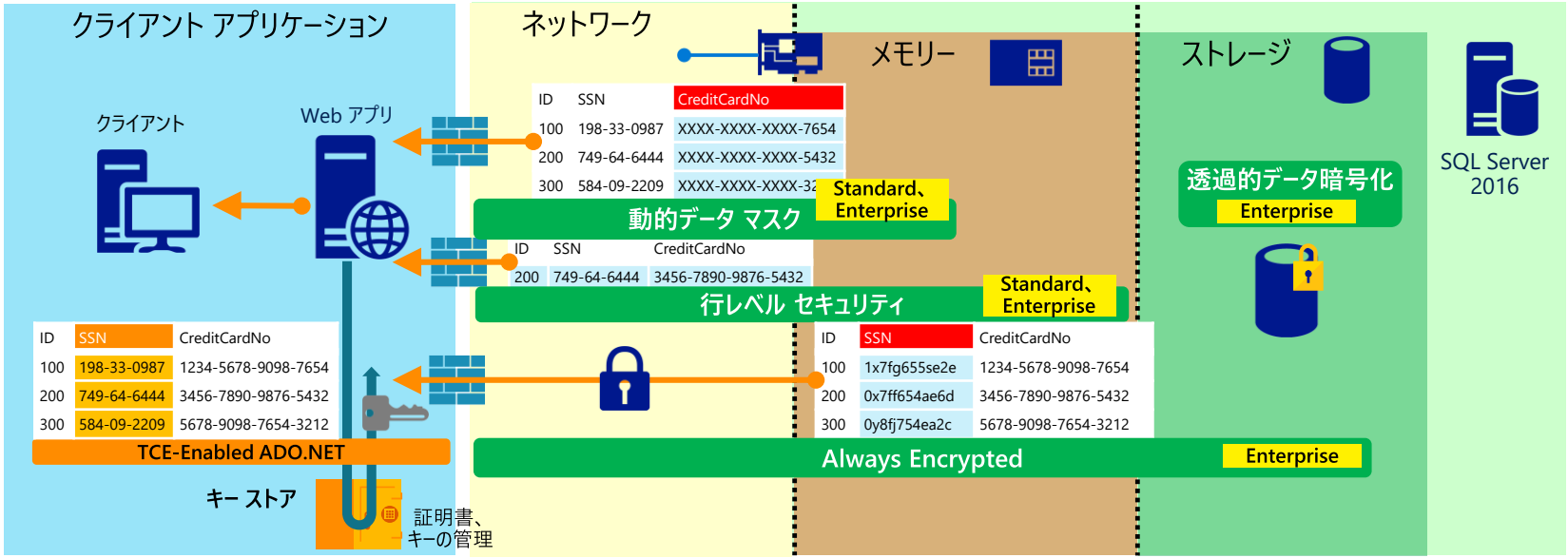
● SQL Server 2016 新機能

● Windows Server 2016 新機能



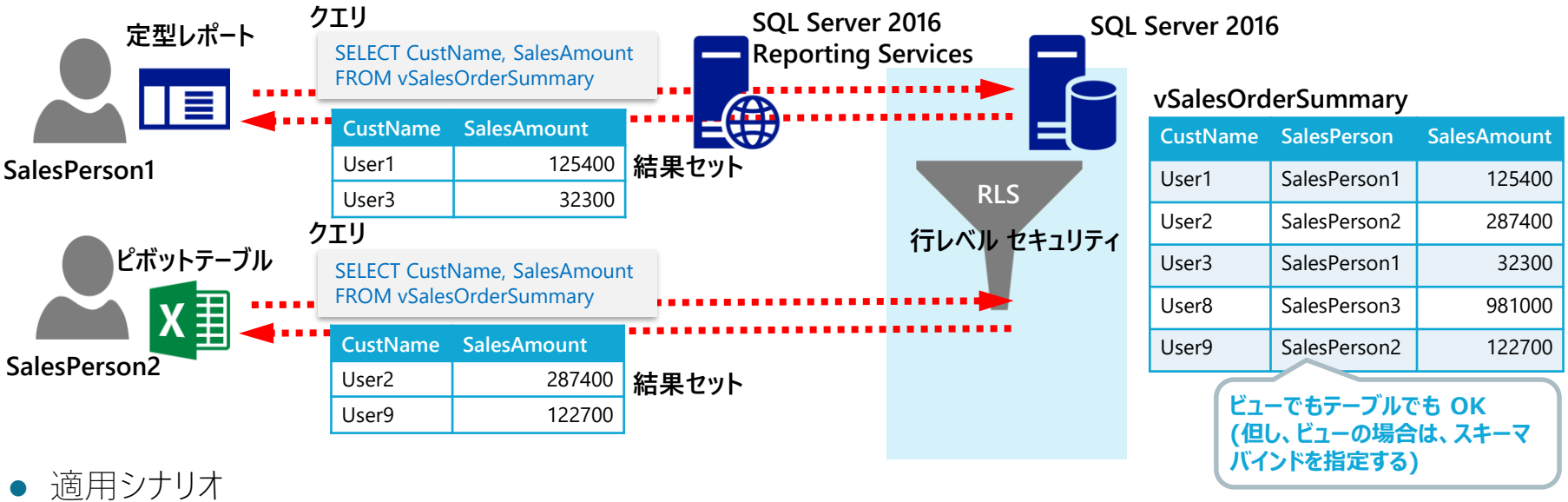
データ分析、データ活用を促進するセキュリティ機能

- 動的データ マスク (DDM)
  - 指定されたマスク フィールド条件で、機微なデータをマスキング
- 行レベル セキュリティ (RLS)
  - ユーザーの ID、ロール、実行コンテキストを基にした行レベルのアクセス制限
- Always Encrypted
  - データを所有する (および表示できる) 人とデータを管理する (ただし、アクセス権を与えない) 人の分離



行レベル セキュリティ (RLS) により、セキュアなレポーティングとセルフサービス BI を実現

- 行レベル セキュリティにより、ユーザーの ID、ロール、実行コンテキストを基にした行レベルのアクセス制限
  - ポリシーとテーブル内のデータでアクセスを制御

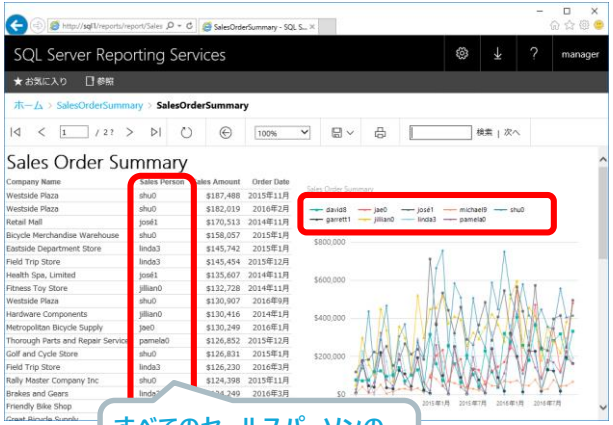


- 適用シナリオ
  - データ分析者の職位やロールに基づく、データ サブセットの抽出とレポート作成
  - 社員の地域やロールに基づく、財務データへのアクセス制限

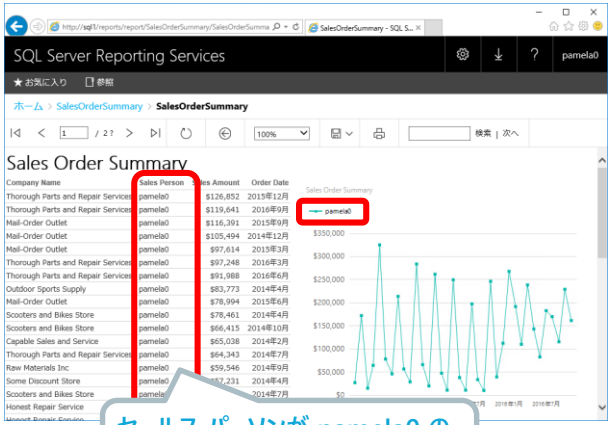
行レベル セキュリティ (RLS) の実装例 (1/3)

- シナリオ
  - すべてのユーザーが共通するレポート コンテンツから、行レベル セキュリティが構成されたデータにアクセスする
  - セールス パーソンの pamela0、david8 は、本人が担当した顧客の売り上げのみが参照できるように設定
  - manager のみ、すべてのデータを参照可能に設定
  - sysadmin 固定サーバー ロールのメンバーであっても、行レベル セキュリティで除外されたユーザー、およびロールは、データにアクセスできない

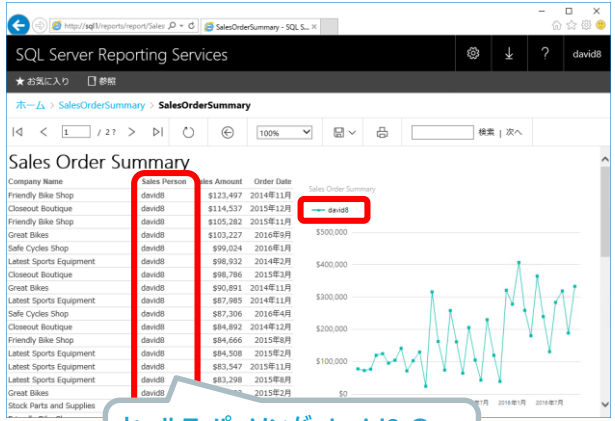
manager が参照した場合のレポート表示



pamela0 が参照した場合のレポート表示



david8 が参照した場合のレポート表示



- ユーザーごとにレポートを分けたり、レポート パラメーターを使用せず実装が可能に！！

行レベル セキュリティ (RLS) の実装例 (2/3)

- ベース テーブル、またはビュー

	CompanyName	SalesPerson	SalesAmount	OrderDate
1	Area Bike Accessories	shu0	12832.9009	2013-11-30
2	Basic Bike Company	david8	9153.6054	2013-11-30
3	Bike Dealers Association	shu0	43362.4196	2013-11-30
4	Capable Sales and Service	pamela0	27510.4109	2013-11-30
5	Fifth Bike Store	david8	3899.6756	2013-11-30
6	First Bike Store	jillian0	6434.0848	2013-11-30
7	Fitness Toy Store	jillian0	38291.2063	2013-11-30
8	General Bike Corporation	garrett1	1481.1742	2013-11-30
9	Grease and Oil Products Company	jillian0	985.018	2013-11-30
10	Great Bikes	david8	48204.0662	2013-11-30
11	Health Spa, Limited	josé1	32474.9324	2013-11-30
12	Historic Bicycle Sales	michael9	6893.2549	2013-11-30

- Windows 認証ログインの作成

```
USE master
CREATE LOGIN [SQL1¥Manager] FROM WINDOWS
CREATE LOGIN [SQL1¥pamela0] FROM WINDOWS
CREATE LOGIN [SQL1¥david8] FROM WINDOWS
```

- データベース ユーザーの作成

```
USE Sales
CREATE USER Manager FROM LOGIN [SQL1¥Manager]
CREATE USER pamela0 FROM LOGIN [SQL1¥pamela0]
CREATE USER david8 FROM LOGIN [SQL1¥david8]
```

行レベル セキュリティ (RLS) の実装例 (3/3)

- 叙述関数の作成

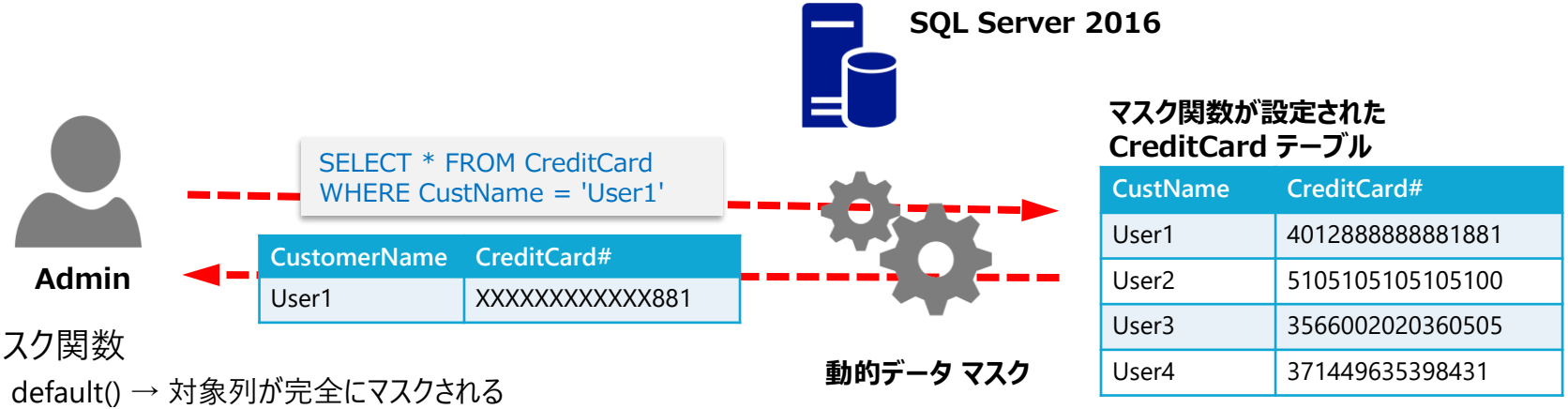
```
CREATE FUNCTION rls.fn_securitypredicate(@SalesPerson AS sysname)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN SELECT 1 AS fn_securitypredicate_result
WHERE @SalesPerson = USER_NAME() OR USER_NAME() = 'Manager';
GO
```

  - ユーザーがテーブル対してクエリを実行するとき、この関数はすべての行に対して動作し、@SalesPerson が USER_NAME() の現在の値と一致しない行を除外する
  - もしくは、USER_NAME() が「Manager」の場合、すべての行を返す
  - クエリ オプティマイザーは、テーブルに対する SELECT、UPDATE、DELETE の各クエリに対して上記 WHERE 述語を追加して実行する
- 叙述関数を SalesLT. vSalesOrderSummary ビューをフィルターする叙述関数としてバインドするセキュリティ ポリシーを作成し、有効化

```
CREATE SECURITY POLICY SalesFilter
ADD FILTER PREDICATE rls.fn_securitypredicate(SalesPerson)
ON SalesLT.vSalesOrderSummary
WITH (STATE = ON)
GO
```

動的データ マスク

- 指定されたマスク フィールド条件で、機微なデータをマスキングする機能



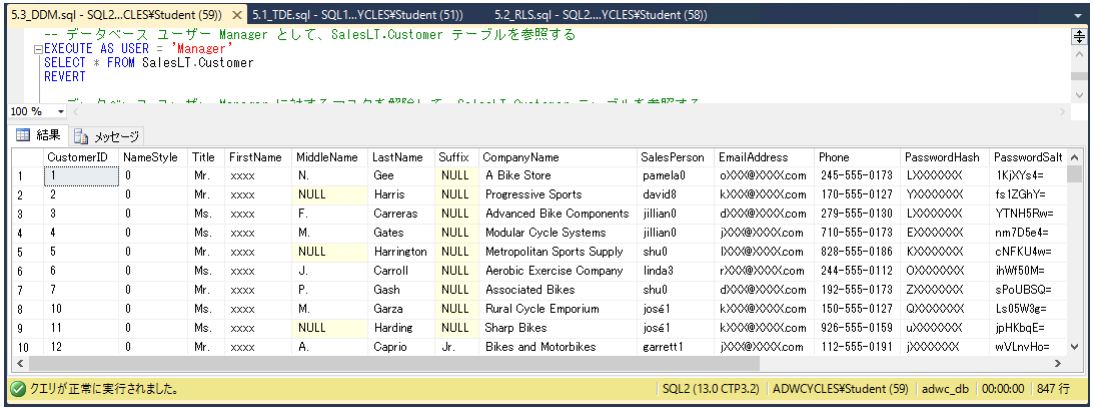
- マスク関数
  - default() → 対象列が完全にマスクされる
  - email() → メール アドレス用のマスク
  - partial(1,"XXXXXXX",0) → 最初の文字以外はマスクされる
  - random(1,5) → ランダムな数値に置き換えられる
- 適用シナリオ
  - 機微なデータの情報漏えいからの保護
  - 特権ユーザーからのアクセス保護

動的データ マスクの実装例

- マスク関数の適用例

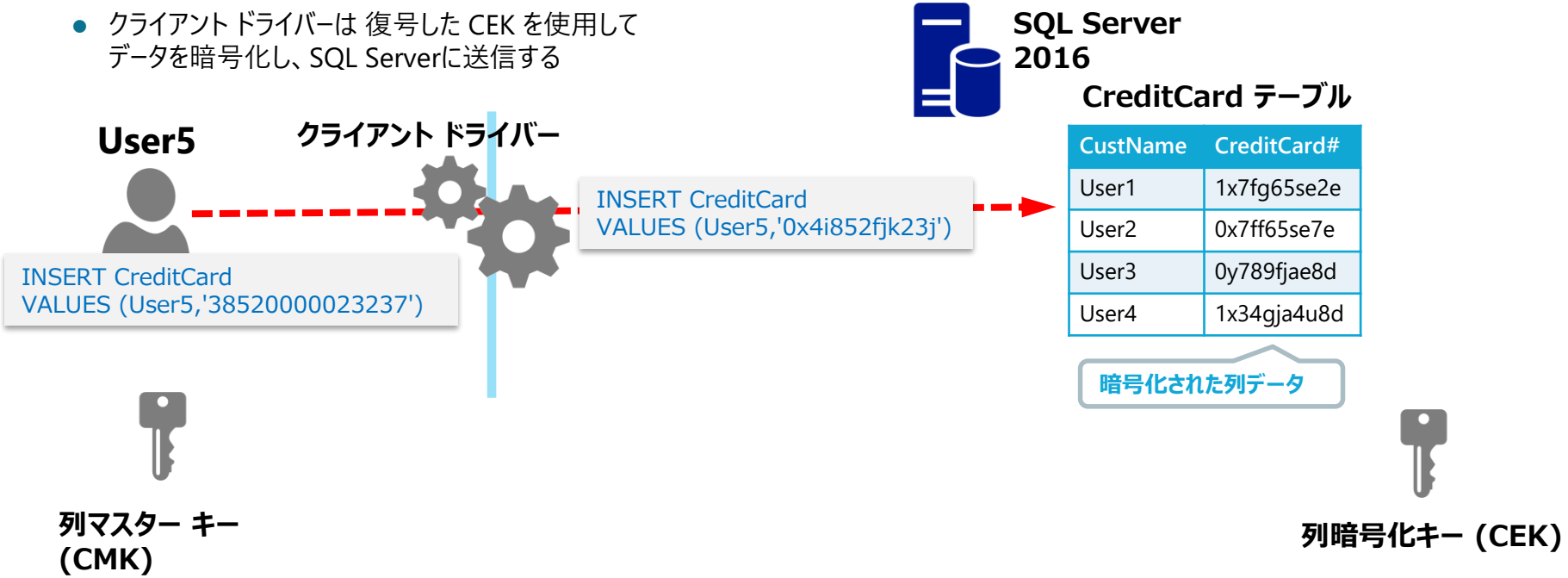
```
ALTER TABLE SalesLT.Customer
ALTER COLUMN FirstName varchar(100) MASKED WITH (FUNCTION = 'default()')
GO
ALTER TABLE SalesLT.Customer
ALTER COLUMN EmailAddress varchar(100) MASKED WITH (FUNCTION = 'email()')
GO
ALTER TABLE SalesLT.Customer
ALTER COLUMN PasswordHash varchar(100) MASKED WITH (FUNCTION ='partial(1,"XXXXXXX",0)')
GO
```

- クエリの結果



Always Encrypted

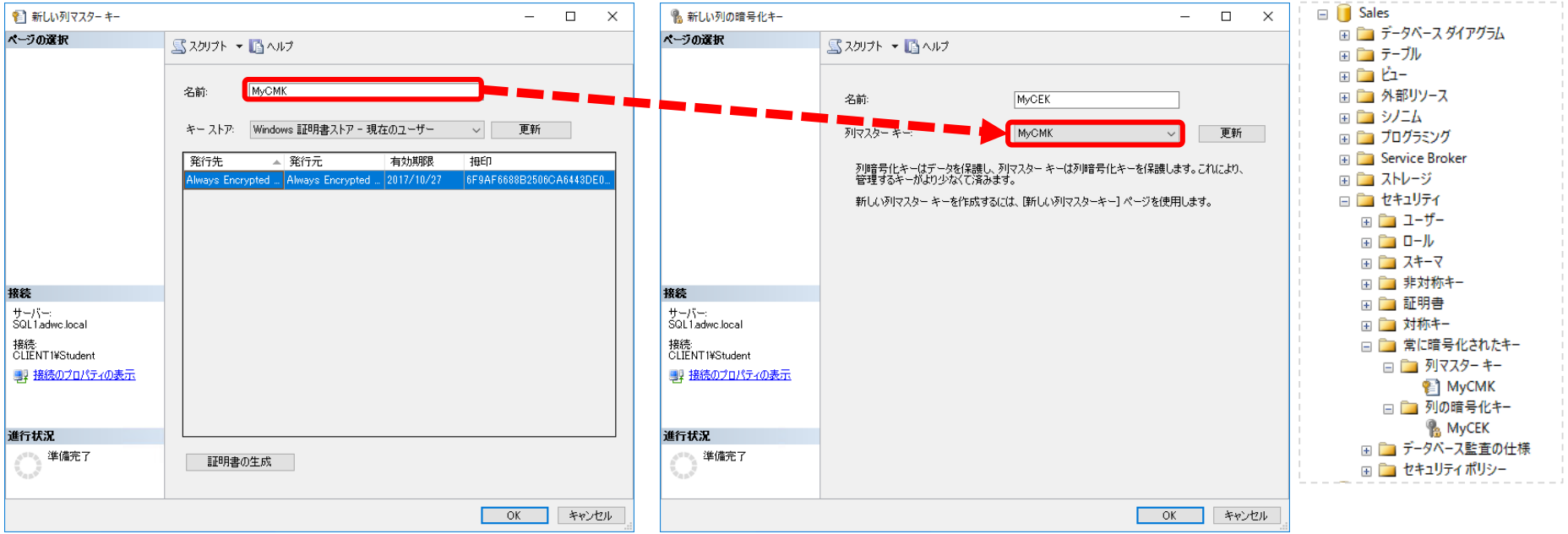
- 通信経路も含め、常に暗号化した状態でデータを操作できる
  - クライアント ドライバーは SQL Server から CEK を取得
  - CEK は、CMK の秘密キーで暗号化されている
  - クライアント ドライバーは 復号した CEK を使用して
    データを暗号化し、SQL Serverに送信する



参照：https://msdn.microsoft.com/en-us/library/mt163865.aspx

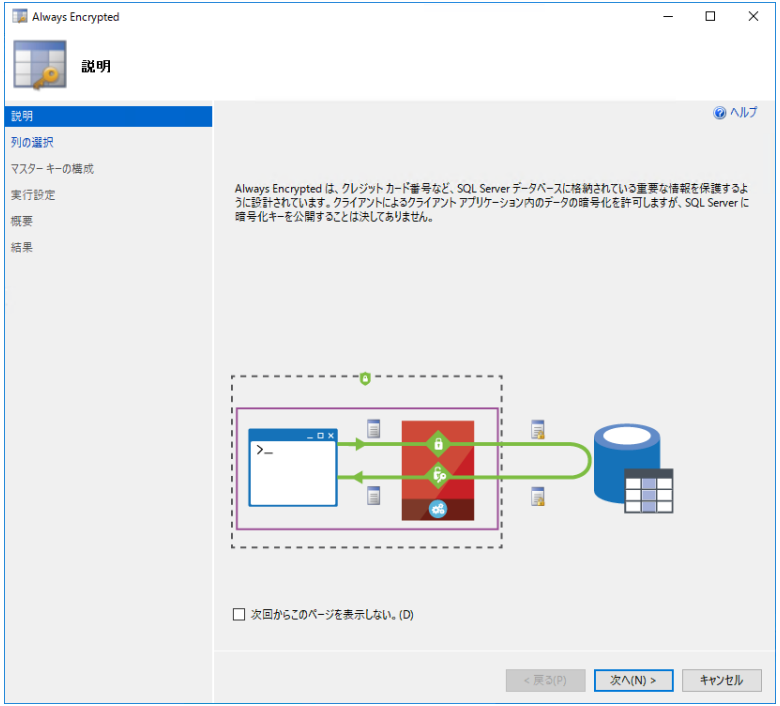
列マスター キーと列の暗号化キーの作成

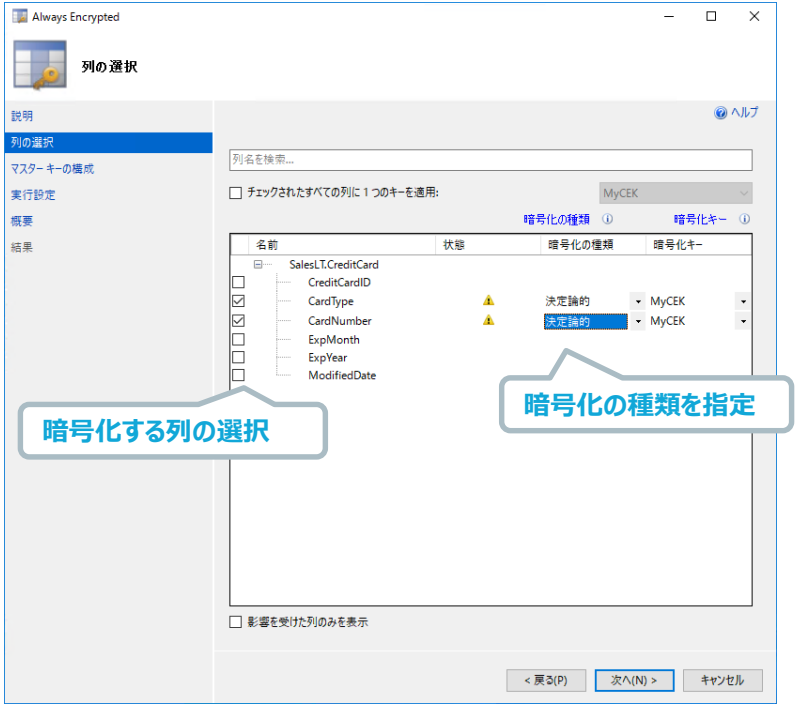
- 列マスター キーを使用して列の暗号化キーを保護



暗号化ウィザードを使用した Always Encrypted の設定

- 対象テーブルを右クリックして [列の暗号化]





Always Encrypted により暗号化されたデータ

- CardType 列と CardNumber 列を「決定論的」を選択して暗号化した例

	CreditCardID	CardType	CardNumber	ExpMonth	ExpYear	ModifiedDate
1	1	SuperiorCard	33332664695310	11	2006	2013-07-29 00:00:00.000
2	2	Distinguish	55552127249722	8	2005	2013-12-05 00:00:00.000
3	3	ColonialVoice	77778344838353	4	2005	2014-01-14 00:00:00.000
4	4	ColonialVoice	77774915718248	7	2006	2013-05-20 00:00:00.000
5	5	Vista	11114404600042	4	2005	2013-02-01 00:00:00.000
6	6	Distinguish	55557132036181	9	2006	2014-04-10 00:00:00.000
7	7	Distinguish	55553635401028	6	2007	2013-02-01 00:00:00.000

平文

結果 メッセージ

	CreditCardID	CardType	CardNumber	ExpMonth	ExpYear	ModifiedDate
1	1	0x019BB914314F94FFFD7D961444B467FF499958562CA9E59E8…	0x01F4A691892C1F26B41AC99FF1F5FD6A4076DC4AC091A115D…	11	2006	2013-07-29 00:00:0
2	2	0x01B02F412E6343A14479BC1BF01B93C7465F29A087CD2C41…	0x0154D3E2138EDC237693AEAF102588FD33516A13222F…			5 00:00:0
3	3	0x0131EA8B79FA9E575D68AA7DF8623D3EA726B50BAAD24BD…	0x01D41C47FDD1574D3BE7524C2F3EACD196E853BF4…			4 00:00:0
4	4	0x0131EA8B79FA9E575D68AA7DF8623D3EA726B50BAAD24BD…	0x0147FD38364E849F3FF001FA6ADE05FF067ECAE216FD2C261F…	7	2006	2013-05-20 00:00:0
5	5	0x0140CE6F84E8F15EED312AE151EBE1ECF13D89F13AC3D114…	0x015DC6979D6015A1C2DD1A3C4646672CEC2BD303C054CC556…	4	2005	2013-02-01 00:00:0
6	6	0x01B02F412E6343A14479BC1BF01B93C7465F29A087CD2C41…	0x018960BA48669A8E1314FE782E6E6D9F6AB3FE0F9B50176C19…	9	2006	2014-04-10 00:00:0
7	7	0x01B02F412E6343A14479BC1BF01B93C7465F29A087CD2C41…	0x01EEA541A6E226DACF4CDD84DDBA495CB9407E1048734E648…	6	2007	2013-02-01 00:00:0

暗号化されたデータ

  - 同じ値は、同じ暗号化コードになる
  - CMK を保持しているクライアントから接続文字列に以下を指定してアクセスすることで平文化される

```
Column Encryption Setting=enabled
```

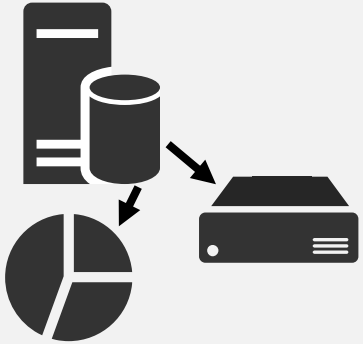
- 暗号化解除は、CMK を保持しているクライアントが列の暗号化ウィザードを再実行して、[暗号化の種類] で「プレーン テキスト」を選択して再設定することで行える

AlwaysOn の使用シナリオ

- 高可用性と拡張性の実現

災害対策

ウォーム セカンダリ レプリカへの手動フェールオーバー



プライマリ レプリカからのオフロード

読み取り可能セカンダリでのロード バランシング (レポーティング、BI)

セカンダリでのデータベース バックアップ

AlwaysOn とは?

- フェールオーバー クラスタリング サービス上に構成される 2 種類のテクノロジーからなる統合された管理環境の総称

フェールオーバー クラスタリング インスタンス (FCI)

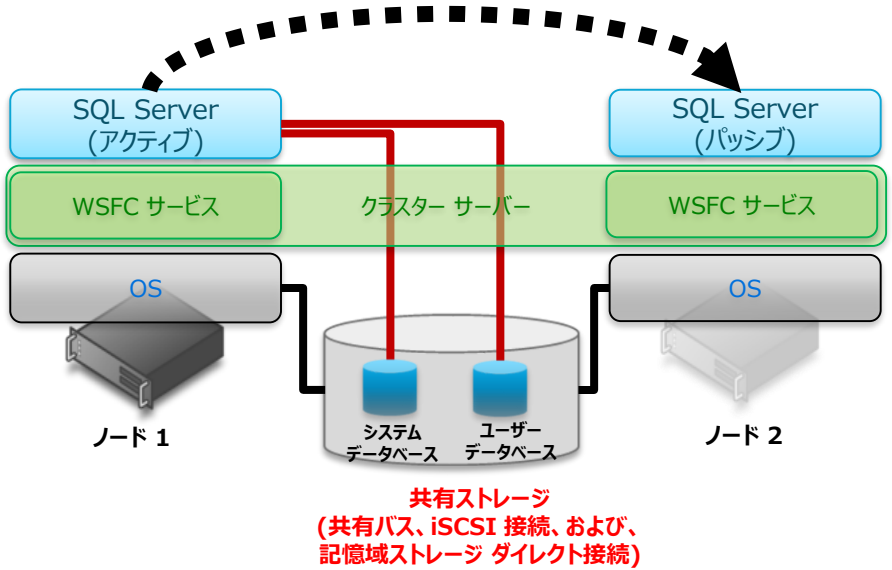
- インスタンス レベルのフェールオーバー
- 共有ストレージ (SAN/SMB)
- 負荷状況に基づく分単位のフェールオーバー時間
- マルチ ノード クラスター
  - パッシブ セカンダリ ノード
  - マルチ サブネット間で構成可能
- 投票権によるフェールオーバーの制御

可用性グループ

- データベース グループのフェールオーバー
- ダイレクト アタッチ ストレージ (共有ディスクは不要)
- 秒単位のフェールオーバー時間
- 複数のセカンダリ レプリカ (8 台まで)
  - アクティブなセカンダリ構成が可能
  - 同期と非同期
- 3 台までの自動フェールオーバー構成

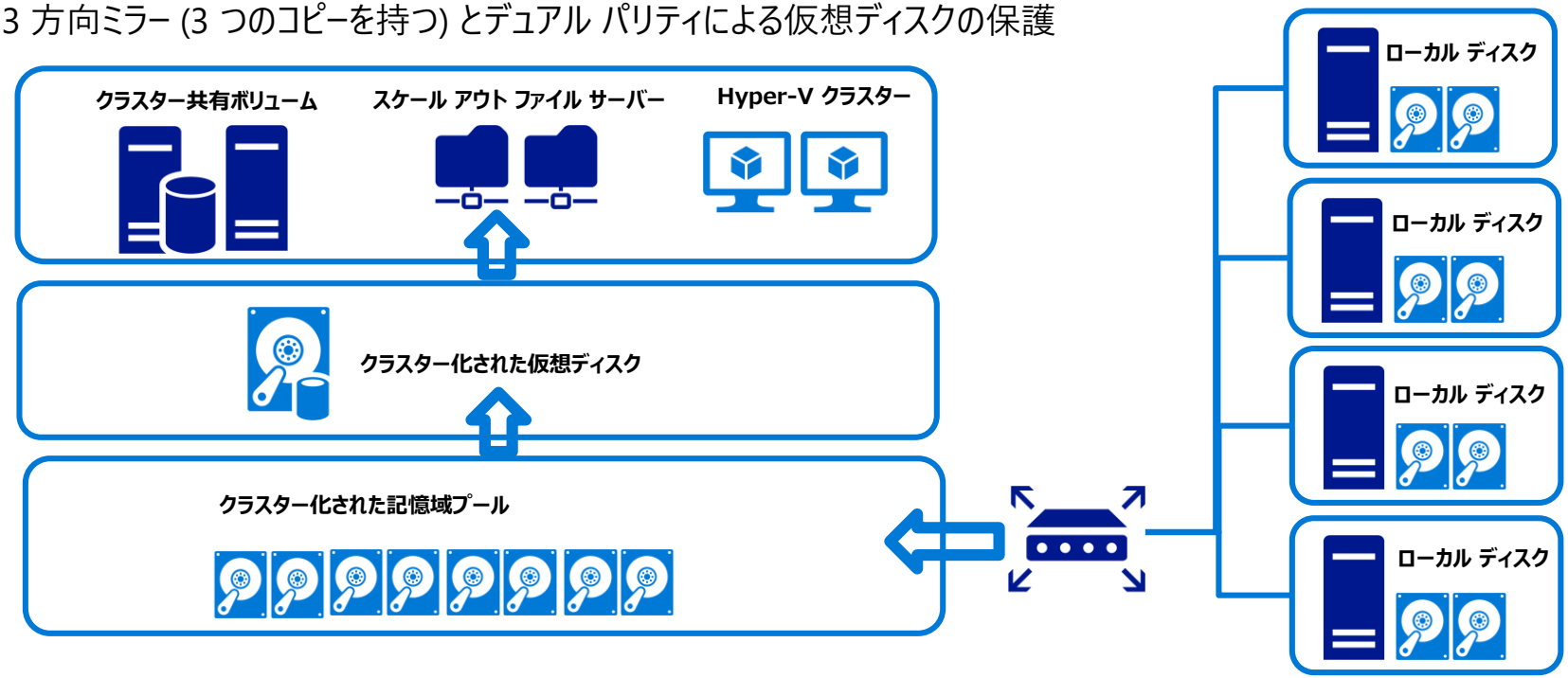
フェール オーバー クラスター インスタンス (FCI)

- インスタンス単位
  - 1 台から 64 台のサーバー ノード (SQL Server 2016 Standard では 2 台まで)
    - 共有ストレージにシステム データベースとユーザーデータベースを配置
    - リソースは単一ノードが所有 (ストレージは Cluster Disk Driver により排他的に制御される)
    - 各ノードで稼働するクラスター サービスによりリソースが監視される
    - 2 系統のネットワーク
  - 利点
    - インスタンス レベルの管理は アクティブ ノードからだけ行えばよい
    - データベース エンジン サービス以外もサポート
    - SQL Server エージェント サービスとの依存関係を保持
  - 欠点
    - 共有ディスクが単一障害点になるため
      RAID 構成などによる保護が必要
    - フェールオーバー時間が分単位



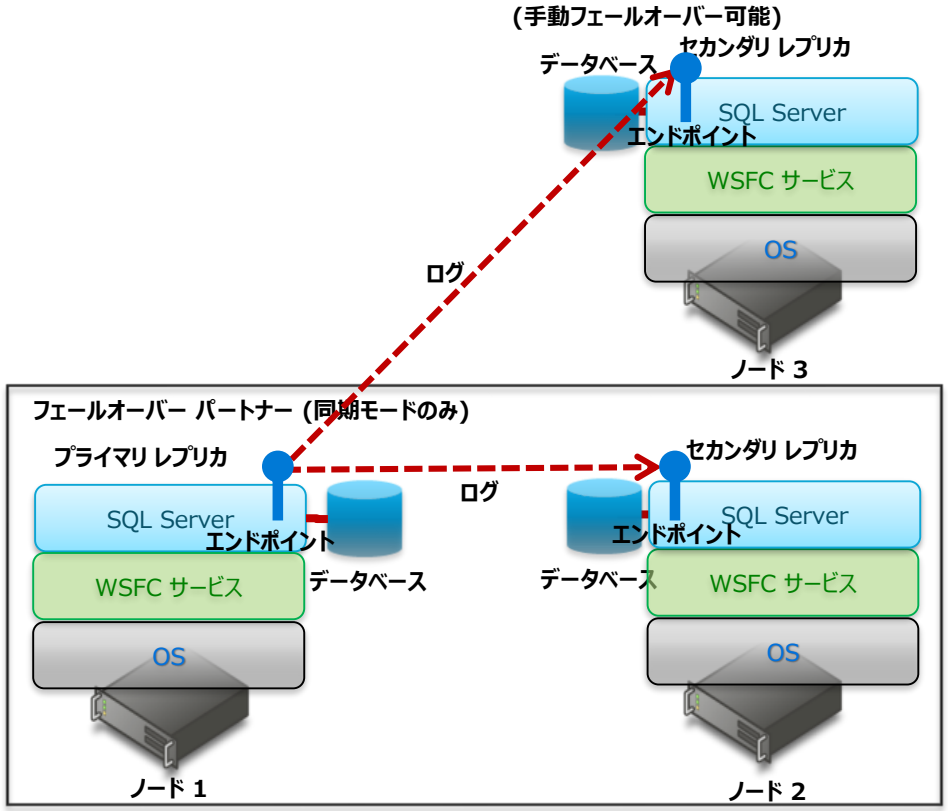
記憶域スペース ダイレクト

- Windows Server 2016 では、記憶域スペース ダイレクトを利用することで、クラスターの各ノードのローカル ディスクをクラスターの共有ボリューム (CSV) として使用可能
  - 3 方向ミラー (3 つのコピーを持つ) とデュアル パリティによる仮想ディスクの保護



AlwaysOn 可用性グループ

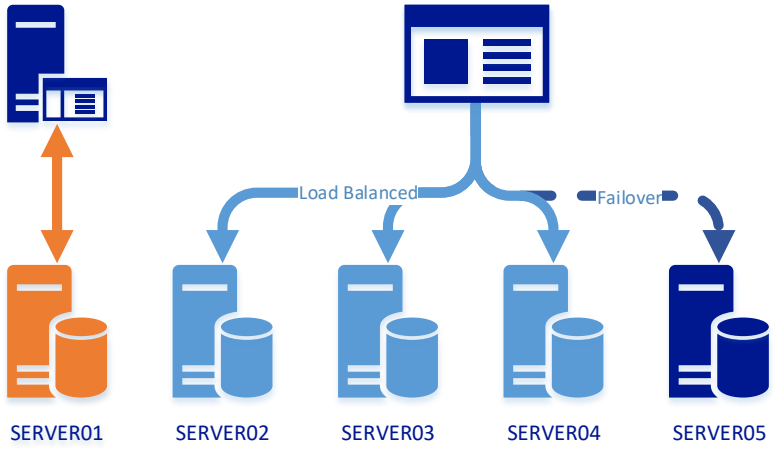
- データベースのグループ単位
  - 8 つまでのセカンダリ レプリカ
  - 3 つまでの自動フェールオーバー
  - 同期コミット モード (3 つまで) と非同期コミット モード
  - エンドポイント通信による暗号化と圧縮
- 利点
  - 可用性向上
    - 瞬時にフェールオーバー
  - 共有ディスクが不要
  - 仮想名を使用したクライアント接続
  - 自動フェールオーバーに監視サーバーが不要
  - 読み取り可能なセカンダリでのロード バランシングが可能
  - セカンダリでバックアップ可能
  - 破損ページの自動修復



SQL Server 2016 AlwaysOn 可用性グループの拡張

- スケーラビリティの向上
  - 自動フェールオーバー ターゲット数の増加
  - 読み取り可能なセカンダリでのロード バランシング
  - ログ転送パフォーマンスの向上
- 管理性の向上
  - AD ドメイン環境に依存しない可用性グループの構成が可能に
  - データベース レベルの正常性検出
  - データベース配置の自動シード処理
  - Azure へのセカンダリ レプリカの展開
  - 分散トランザクション コーディネーター (DTC) のサポート
  - グループの管理されたサービス アカウント (gMSA) のサポート

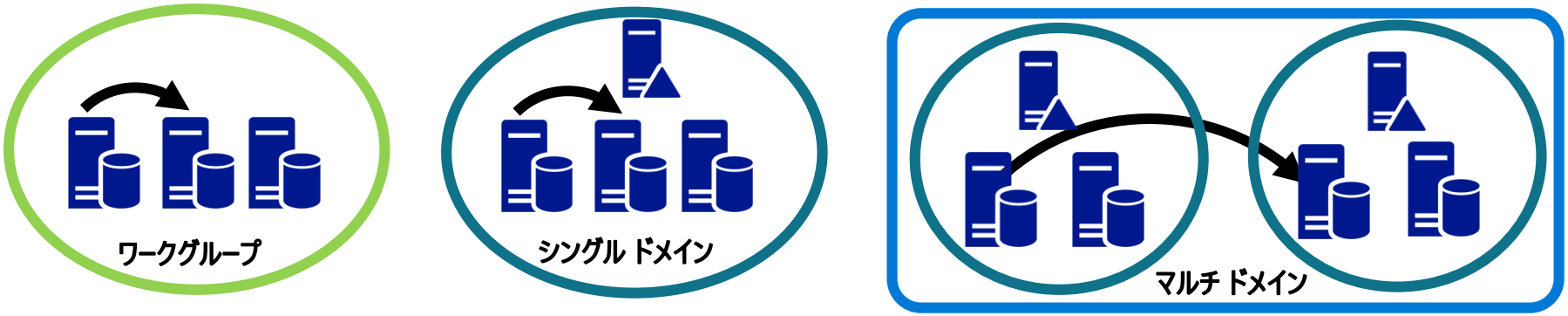
READ_ONLY_ROUTING_LIST = (('SERVER02','SERVER03','SERVER04'),'SERVER05')



読み取り可能なセカンダリでのロード バランシング

Windows Server 2016 により
ドメイン環境に依存しない AlwaysOn 可用性グループが可能に

- Windows Server 2016 よりワークグループ環境、およびマルチドメイン構成での
  フェールオーバー クラスター構成が可能に



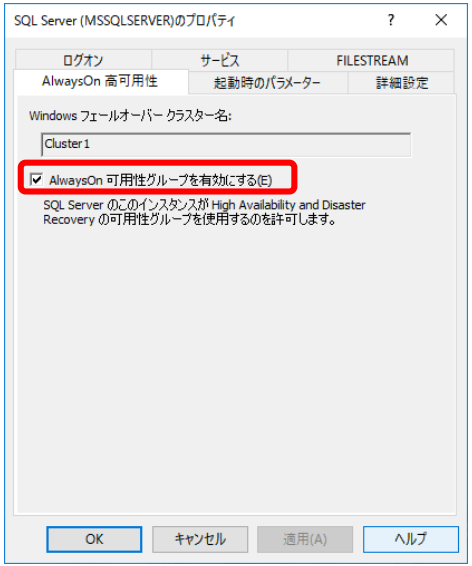
- ワークグループ環境、およびマルチドメイン構成でのフェールオーバー クラスター構成でサポートされるサービス
  - SQL Server
  - ファイル サーバー
  - Hyper-V

参照：https://blogs.msdn.microsoft.com/clustering/2015/08/17/workgroup-and-multi-domain-clusters-in-windows-server-2016/

事前準備

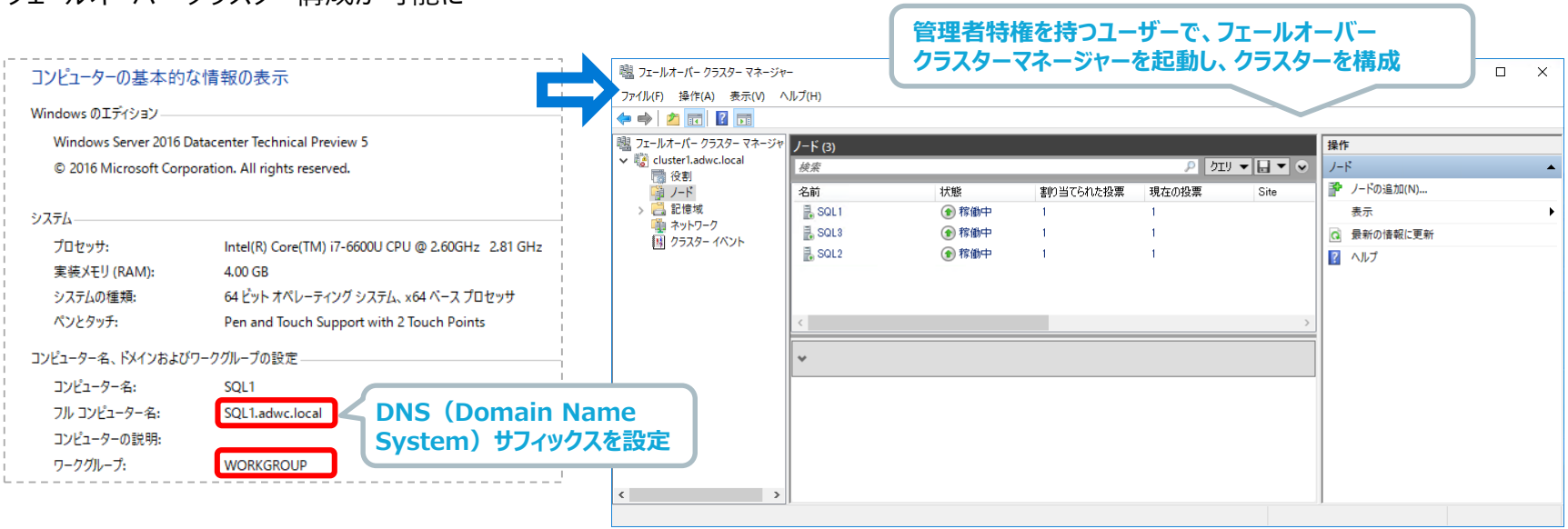
- フェールオーバー クラスタリングのハードウェア要件を満たすサーバー構成を用意
  - https://technet.microsoft.com/ja-jp/library/jj612869.aspx
  - https://technet.microsoft.com/ja-jp/library/jj134244.aspx
- 可用性グループに参加させるすべてのノードで次の作業を行う
  - Windows ファイアウォールを構成
    - SQL Server のポート(既定値 TCP1433) とデータベース同期のためのリスナー ポート (既定値 TCP 5022) の受信許可
  - プライマリ DNS サフィックスを設定
  - フェールオーバー クラスタリング機能をインストールし、ワークグループクラスターを構成
  - SQL Server サービス アカウントとして使用する同じユーザー名とパスワードを持つローカル ユーザーを作成
  - SQL Server 2016 をインストールし、構成マネージャーで AlyaysOn を有効化
    - PowerShell を使用する場合は Enable-SQLAlwaysOn コマンド
- プリンシパル データベースを共有フォルダーにバックアップ

```
USE master
GO
-- 完全バックアップの作成
BACKUP DATABASE Sales TO DISK = '¥¥gw¥share¥backup¥sales.bak' WITH INIT
GO
-- トランザクション ログ バックアップの作成
BACKUP LOG Sales TO DISK = '¥¥gw¥share¥backup¥sales.trn'
GO
```



ワークグループ クラスターの構成

- Windows Server 2016 よりワークグループ環境、およびマルチドメイン構成での
  フェールオーバー クラスター構成が可能に



エンドポイントの作成とアクセス許可

- プリンシパルの作成

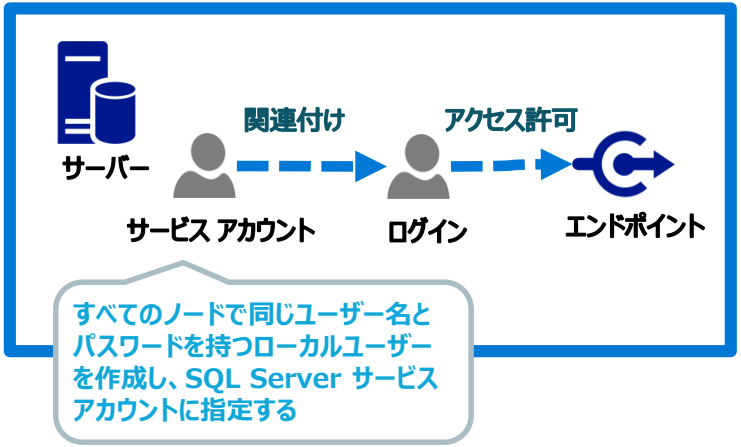
```
USE master
CREATE LOGIN [SQL1¥SQLServer] FROM WINDOWS
```

- エンドポイントの作成

```
CREATE ENDPOINT hadr_Endpoint
 STATE = STARTED
 AS TCP(LISTENER_PORT = 5022)
 FOR DATA_MIRRORING
 (ROLE = ALL
 ,ENCRYPTION = REQUIRED ALGORITHM AES)
 GO
```

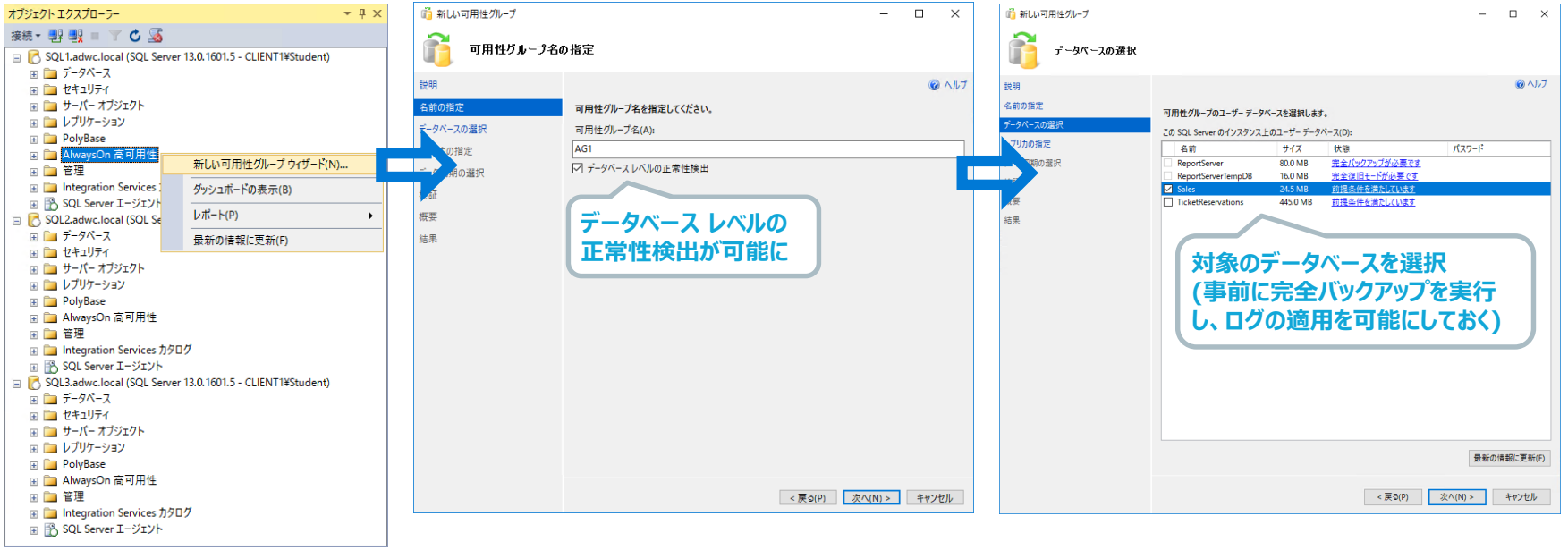
- プリンシパルに対するエンドポイントへのアクセスを許可

```
GRANT CONNECT ON ENDPOINT::hadr_Endpoint TO
[SQL1¥SQLServer]
GO
```



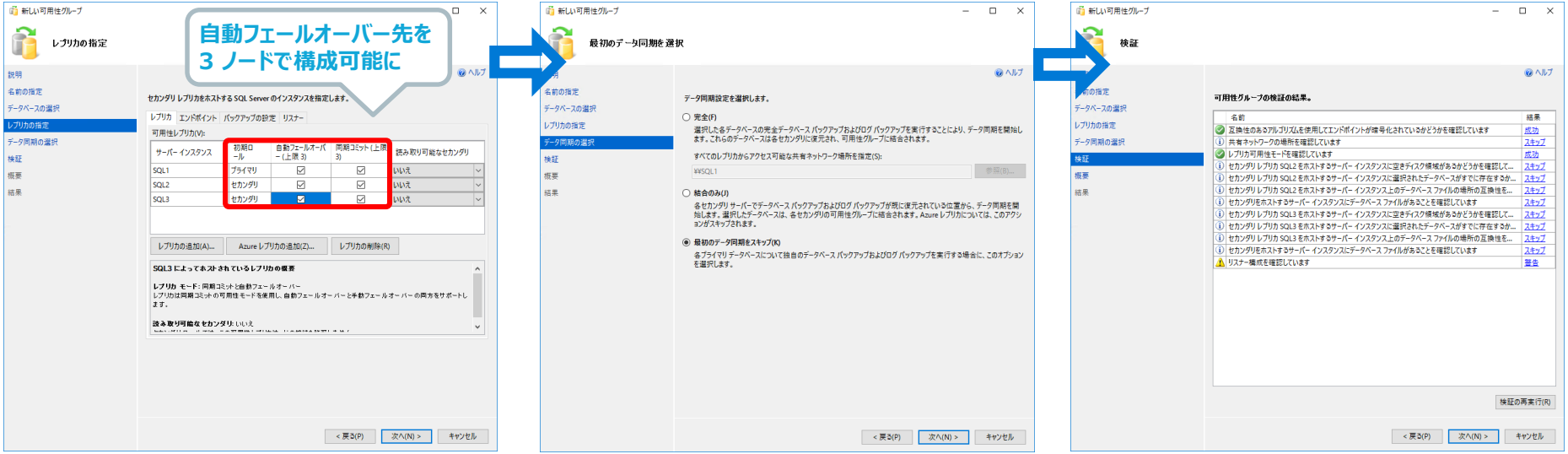
AlwaysOn 高可用性グループの構成

- 高可用性グループ ウィザードの起動



AlwaysOn 高可用性グループの構成

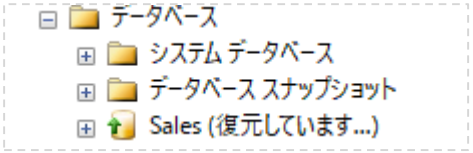
- レプリカの選択と初期同期フォルダーの指定

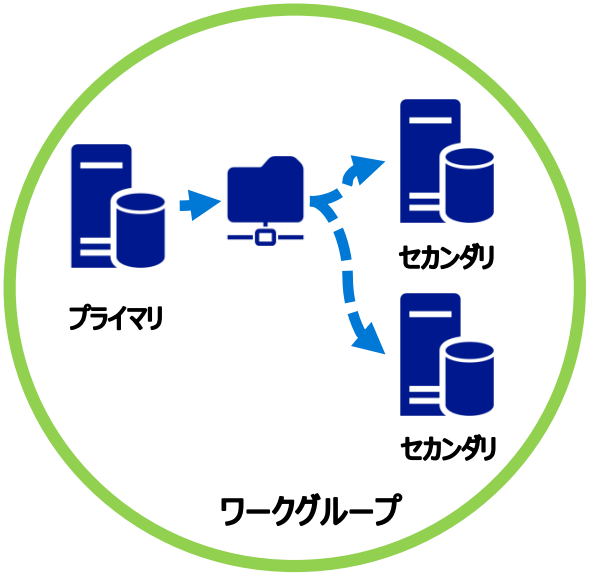


セカンダリにデータベースを復旧なしで復元

- 共有フォルダーに作成したバックアップをセカンダリ側に RESTORE WITH NORECOVERY を使用して復元

```
-- 完全バックアップを復旧なしで復元
RESTORE DATABASE Sales FROM DISK = '¥¥gw¥share¥backup¥sales.bak'
WITH NORECOVERY
GO
-- トランザクション ログ バックアップを復旧なしで復元
RESTORE LOG Sales FROM DISK = '¥¥gw¥share¥backup¥sales.trn'
WITH NORECOVERY
GO
```

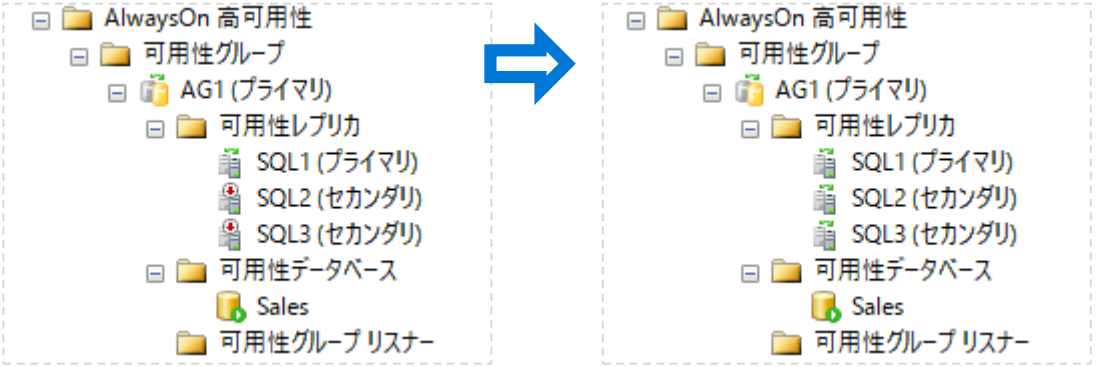




可用性グループへの参加と結合

- セカンダリ側に データベース復元後、T-SQL を使用して、データベースを可用性グループに参加させる

```
-- 可用性グループに参加
ALTER AVAILABILITY GROUP [AG1] JOIN
GO
-- データベースを可用性データベースに移行
ALTER DATABASE Sales SET HADR AVAILABILITY GROUP = AG1
GO
```



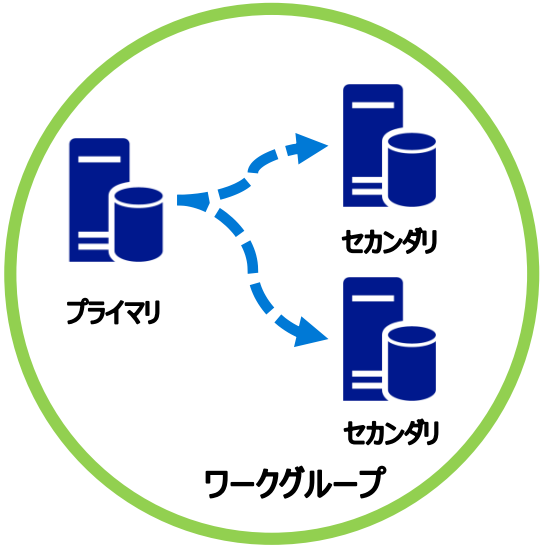
データベース配置の自動シード処理

- 配置先が同じフォルダー構造の場合、セカンダリ レプリカをネットワーク経由で自動的にシード処理可能

```
-- 可用性グループを自動シード モードに設定
ALTER AVAILABILITY GROUP [AG1]
MODIFY REPLICA ON N'SQL1' WITH (SEEDING_MODE= AUTOMATIC)
GO
ALTER AVAILABILITY GROUP [AG1]
MODIFY REPLICA ON N'SQL2' WITH (SEEDING_MODE= AUTOMATIC)
GO
ALTER AVAILABILITY GROUP [AG1]
MODIFY REPLICA ON N'SQL3' WITH (SEEDING_MODE= AUTOMATIC)
GO
```

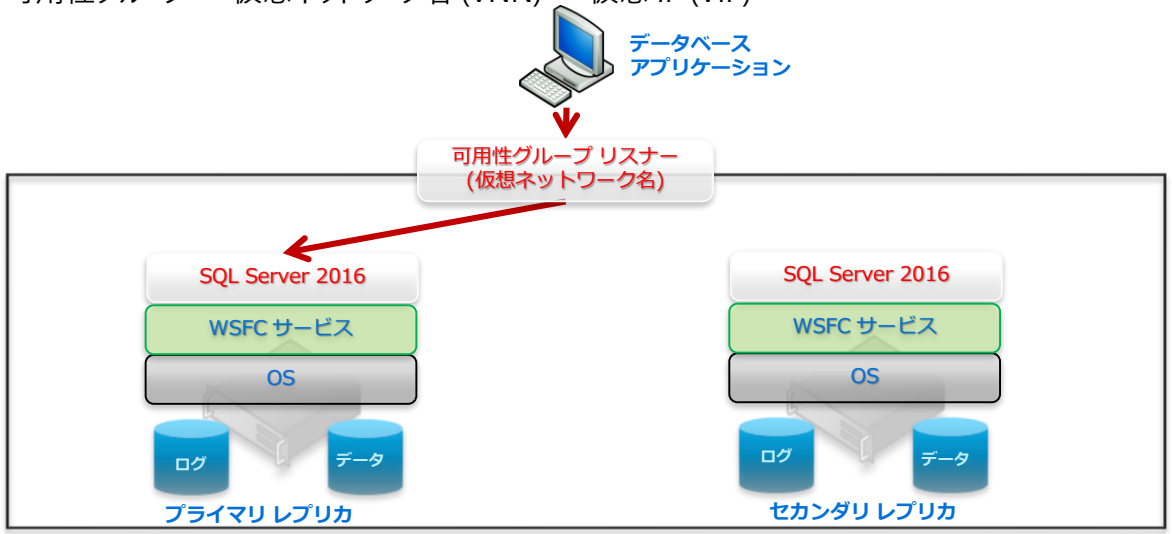
```
-- セカンダリ レプリカで、可用性グループにデータベースの作成を許可
ALTER AVAILABILITY GROUP AG1 GRANT CREATE ANY DATABASE
GO
```

```
-- プライマリ レプリカで、可用性グループへのデータベースの追加
ALTER AVAILABILITY GROUP [AG1] ADD DATABASE TicketReservations
GO
-- 自動シード処理による同期の状態の確認
SELECT * FROM sys.dm_hadr_automatic_seeding
SELECT * FROM sys.dm_hadr_physical_seeding_stats
```



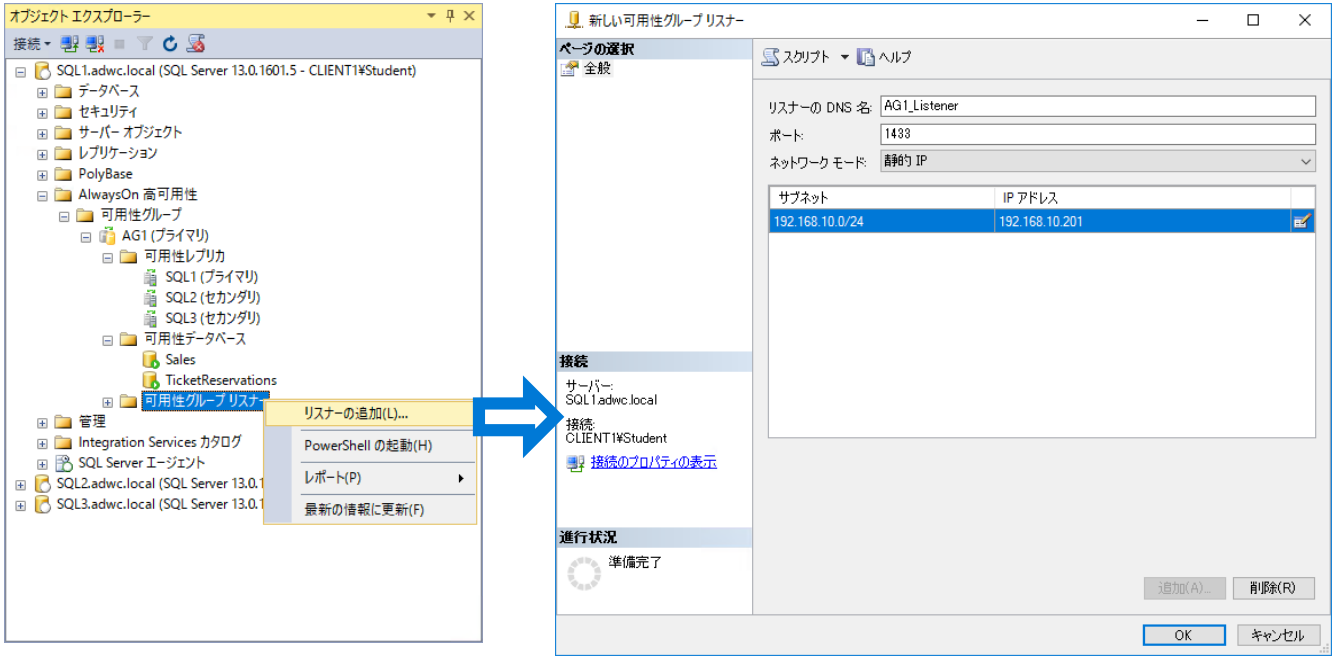
可用性グループ リスナー

- 可用性グループのデータベースへのクライアント接続を提供
  - プライマリ、またはセカンダリのデータベースにアクセスするためのサーバー名を定義
  - 可用性グループ リスナー クライアントは SQL Server インスタンス名を使用しないで接続可能
  - フェールオーバー時、現在のプライマリに接続するため、接続文字列を変更する必要がない
- WSFC のクラスター リソースを使用
  - 依存関係：可用性グループ ← 仮想ネットワーク名 (VNN) ← 仮想 IP (VIP)



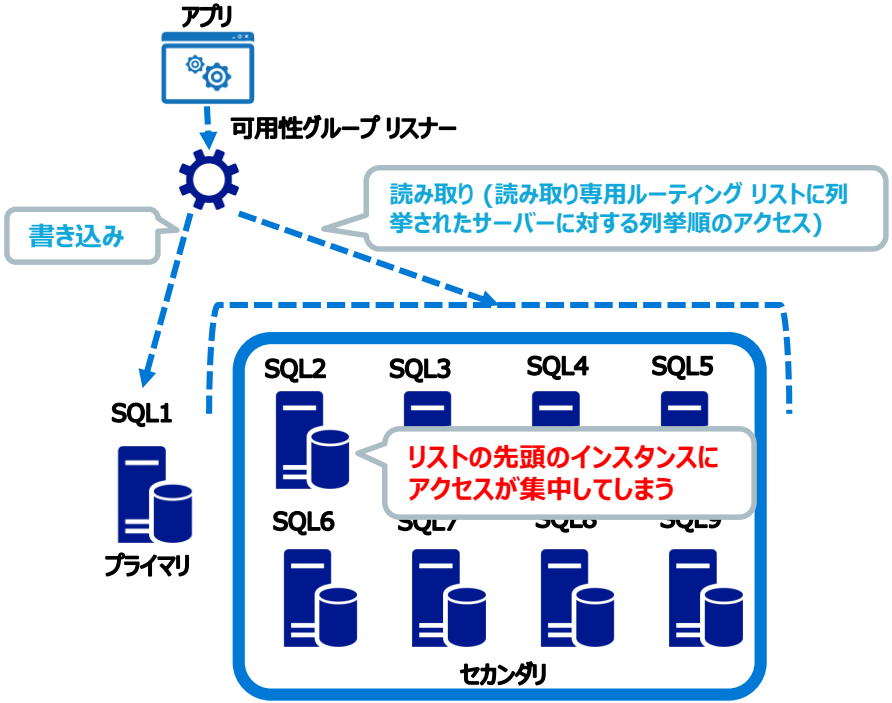
可用性グループ リスナーの作成

- ネットワークモードを静的 IP と動的 IP から選択

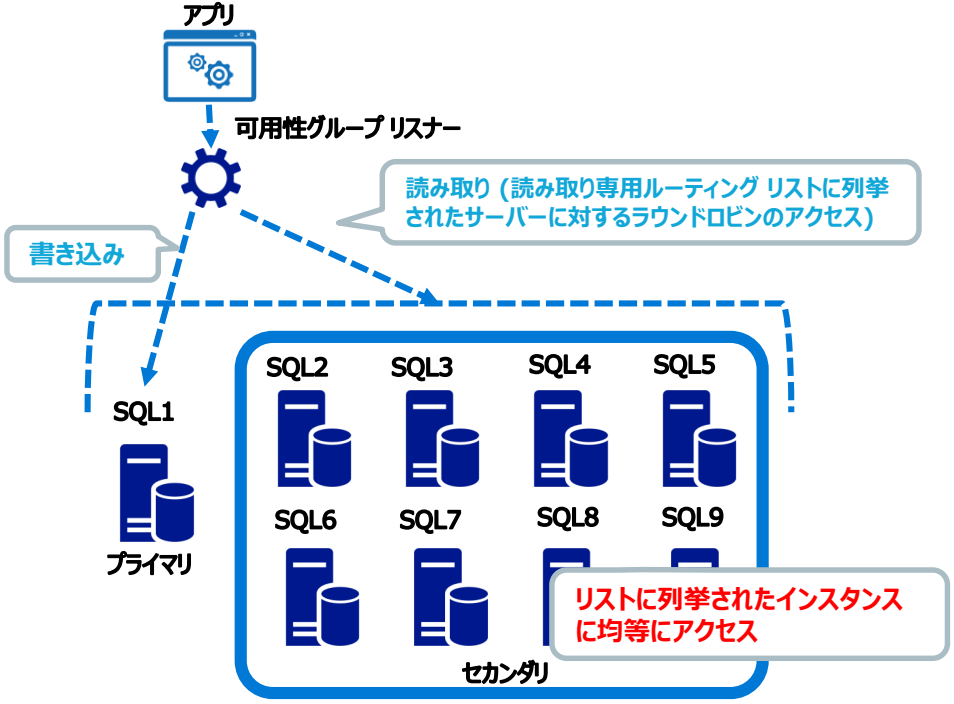


読み取り可能なセカンダリでのロード バランシングの構成

- 可用性グループの読み取り専用ルーティングを構成することで、「ApplicationIntent=ReadOnly」を指定した接続に対して、ラウンドロビンでのアクセスが可能に



SQL Server 2014



SQL Server 2016

読み取り専用アクセスの許可とルーティング URL の指定

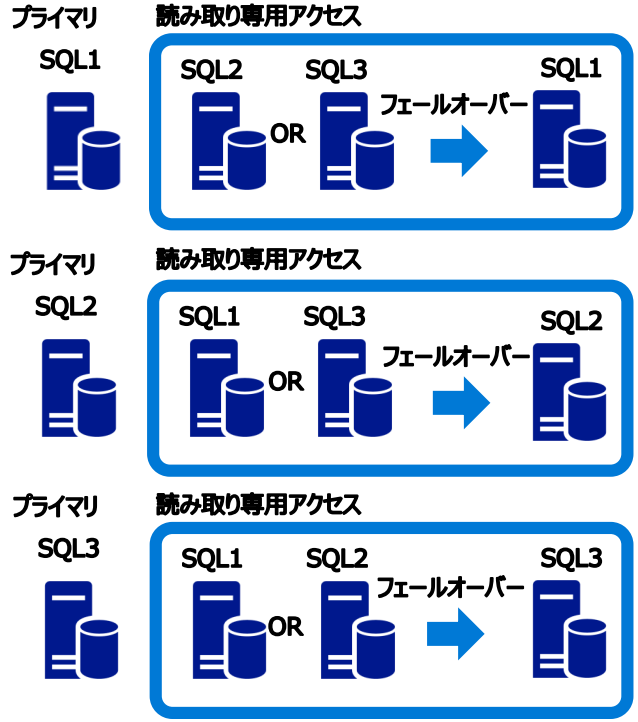
- プライマリ レプリカをホストするサーバー インスタンスに接続して実行

```
-- SQL1 に対する読み取り専用アクセスの許可とルーティング URL の指定
ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON  N'SQL1' WITH
(SECONDARY_ROLE (ALLOW_CONNECTIONS = READ_ONLY))
GO
ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON  N'SQL1' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://SQL1.adwc.local:1433'))
GO
-- SQL2 に対する読み取り専用アクセスの許可とルーティング URL の指定
ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON  N'SQL2' WITH
(SECONDARY_ROLE (ALLOW_CONNECTIONS = READ_ONLY))
GO
ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON  N'SQL2' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://SQL2.adwc.local:1433'))
GO
-- SQL3 に対する読み取り専用アクセスの許可とルーティング URL の指定
ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON  N'SQL3' WITH
(SECONDARY_ROLE (ALLOW_CONNECTIONS = READ_ONLY))
GO
ALTER AVAILABILITY GROUP [AG1]
 MODIFY REPLICA ON  N'SQL3' WITH
(SECONDARY_ROLE (READ_ONLY_ROUTING_URL = N'TCP://SQL3.adwc.local:1433'))
GO
```

読み取り専用ルーティング リストの作成

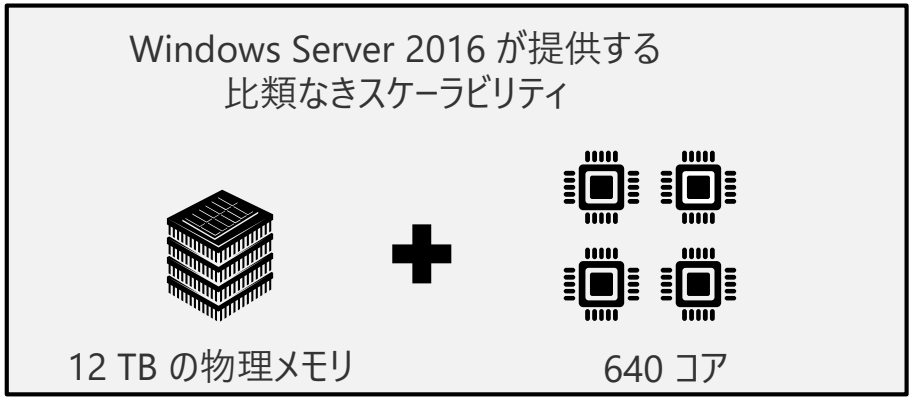
- プライマリ レプリカをホストするサーバー インスタンスに接続して実行

```
-- SQL1 がプライマリの場合の読み取り専用ルーティング リストの作成
ALTER AVAILABILITY GROUP [AG1]
MODIFY REPLICA ON N'SQL1'
WITH (
PRIMARY_ROLE(READ_ONLY_ROUTING_LIST=((N'SQL2',N'SQL3'),N'SQL1')
))
GO
-- SQL2 がプライマリの場合の読み取り専用ルーティング リストの作成
ALTER AVAILABILITY GROUP [AG1]
MODIFY REPLICA ON N'SQL2'
WITH (
PRIMARY_ROLE(READ_ONLY_ROUTING_LIST=((N'SQL1',N'SQL3'),N'SQL2')
))
GO
-- SQL3 がプライマリの場合の読み取り専用ルーティング リストの作成
ALTER AVAILABILITY GROUP [AG1]
MODIFY REPLICA ON N'SQL3'
WITH (
PRIMARY_ROLE(READ_ONLY_ROUTING_LIST=((N'SQL1',N'SQL2'),N'SQL3')
))
GO
```



Better Together Configurations：SQL Server 2016 on Windows Server 2016

- スケールと柔軟性の向上
  - プロセッサーコア数とメモリ サイズの増加
    - 640 コア
    - 12 TB の物理メモリ
  - 高可用性構成の柔軟性
    - 記憶域スペースダイレクト (Storage Spaces Direct) を使用した AlwaysOn FCI 共有ボリュームの構成
    - AD ドメイン環境に依存しない AlwaysOn 可用性グループの構成



ハンズオン トレーニングのご案内

- Microsoft SQL Server 2016 データベース管理者向け講座
  - Microsoft SQL Server 2016 への移行とセキュリティ実装のポイント (1日)
  - Microsoft SQL Server 2016 運用管理 (2日) Coming Soon
- レポーティングおよび、データ分析ソリューションの開発者向け講座
  - Microsoft SQL Server 2016 レポーティング ソリューションの実装と運用管理 (2日) Coming Soon
  - Power BI サービスと Excel を使ったデータ分析環境の構築 (2日)

エディフィスト SQL Server 検索

お気軽に
お問合せください

エディフィストラーニング株式会社
【フリーダイヤル】0120-876-544【電話番号】 03-3282-1311 (代)
【ホームページ】 https://www.edifist.co.jp/

Edifist
エディフィストラーニング株式会社
Microsoft Partner
Microsoft
Gold Learning
Silver Data Analytics
Silver Data Platform
Silver Software Asset Management

SQL Server のご相談は、SQL Direct にお問い合わせください

- 日本のみで提供している Microsoft SQL Server 専用コール センター サービス
  - ご利用に際してのご契約はご不要
  - SQL Server をご検討、ご提案、情報収集されている パートナー企業およびお客様をご支援
  - 日本人スタッフが対応

SQL Server 専用コールセンター
「SQL Direct」
Oh! ゴー ゴー シークル
0120-055-496
月曜日～金曜日 (弊社指定休業日を除く) 9:00~17:30

たとえば、こんなときにお問い合わせください