Microsoft

# SQL Server 2016

## SQL Server 2016 実践シリーズ No.1

SQL Server 2016 への移行とアップグレードの実践

Published: 2016 年 12 月 31 日

有限会社エスキューエル・クオリティ

SQLQuality
データベース コンサルティング

# 目次

## はじめに

「**SQL Server 2016 実践シリーズ**」は、現場のエンジニアの方々が "**即実践**" で利用できるようになるスキルを身につけることを目的としたものです。このドキュメントは、「**SQL Server 2016 への移行とアップグレードの実践**」と題して、移行とアップグレードに関する具体的かつ実践的な方法を説明します。

単なる手順を説明したドキュメントではなく、実際にどのように移行／アップグレードすれば良いのか？ を一番に考えて作成しています。

- 具体的な移行／アップグレード手順は？
- 移行／アップグレードにあたっての注意点は？
- 移行／アップグレードにかかる時間は？ ダウンタイムは？
- ハードウェア リプレースが伴う場合や 32 ビットから 64 ビットに移行する場合は？
- 移行／アップグレードするメリットは？

など、ノウハウを満載したドキュメントになっています。SQL Server での移行／アップグレードは、他のデータベース製品と比べても非常に簡単で、弊社のお客様でも「**こんなに簡単に移行できるとは思わなかった**」、「**アップグレード時の停止時間が見積もりよりも半分で済んだ**」などの声をいただいています。もちろん、移行／アップグレードにあたっては、注意点やポイントもありますので、本ドキュメントではそれらを詳しく説明していきます。

# STEP 1. SQL Server 2016 への移行とアップグレードの概要

第１章では、SQL Server の最新バージョンである **SQL Server 2016** への移行およびアップグレードの概要を説明します。SQL Server 2016 へ移行／アップグレードするメリットや、既に早期導入を行っている企業が、SQL Server 2016 へ移行／アップグレードしたことで、どのような効果が得られたのか（大幅な性能向上など）も説明します。

この章の目次は、次のとおりです。

- ✓ SQL Server 2016 へ移行／アップグレードするメリット
- ✓ SQL Server 2016 による性能向上（早期導入事例）
- ✓ SQL Server 2016 への移行／アップグレードの概要

## 1.1 SQL Server 2016 へ移行／アップグレードするメリット

**旧バージョンの SQL Server**（SQL Server 2005 や 2008、2008 R2、2012、2014）を **SQL Server 2016** へ移行／アップグレードするメリットは、次のとおりです。

### 性能向上

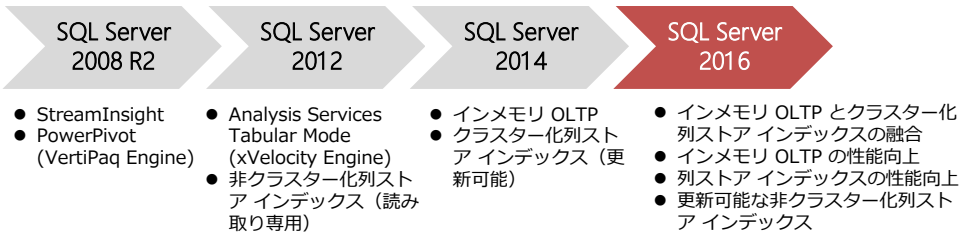
**SQL Server 2016** には、OLTP 向けのインメモリ機能である「**インメモリ OLTP**」や、DWH／BI 向けのインメモリ機能である「**列ストア インデックス**」（カラム指向データベースの SQL Server 実装）、ビッグデータ／データ解析向けのインメモリ機能である Analysis Services の **Tabular Mode**（表形式モード）、エンタープライズ対応でパラレル処理が可能な「**R**」（ScaleR）、**Automatic Soft NUMA**、**Multiple Log Writer**、**INSERT..SELECT のパラレル処理**、**Dynamic Memory Object Scaling**、**TDE**（透過的なデータ暗号化）のインメモリ OLTP 対応と AES-NI 対応、AlwaysOn 可用性グループの**ログ転送性能の向上**など、性能向上を実現できる機能が数多く搭載されています。

実際、SQL Server 2016 の早期導入を行った bwin 社では、インメモリ OLTP を利用して **1 秒あたり 120 万バッチ リクエスト数**を達成、Jack Henry & Associates 社では、**1 秒あたり 130 万予測**を達成（ローン審査におけるスコアリング エンジンとして SQL Server 2016 の **R** を利用）、Moneris Solutions 社では、アプリケーションを一切変更することなく、SQL Server 2016 に変更しただけで **10～20%**の性能向上を実現（ハードウェアは従来と同じものを利用）、FIS 社では、列ストア インデックスを付与しただけで、アプリケーションやクエリを一切変更することなく、**分析クエリ**の性能が **4 倍**、**10 倍**、**20 倍**に向上しています。

StackOverflow 社では、列ストア インデックスを採用して、ログ データを **180GB/日** から **16GB/日** に圧縮（**10 分の 1 以下**に圧縮）、AlwaysOn 可用性グループのローリング アップグレードのおかげで、**11 台**の SQL Server をわずか **48 時間でアップグレードを完了**させています。こうした早期導入事例については、次の項で詳しく説明します。

### SQL Server のインメモリ技術

**SQL Server のインメモリ技術**の歴史は古く、3 つ前のバージョンである SQL Server 2008 R2 から提供された StreamInsight および PowerPivot に始まり、以下のインメモリ技術が提供されています。




*StreamInsight は、現在、Microsoft Azure 上のクラウド サービスである「Stream Analytics」として利用することができます（IoT：Internet of Things データの処理に利用可能）

PowerPivot は、インメモリの BI 機能として SQL Server 2008 R2 から提供され、このエンジンは、SQL Server 2012 からは **xVelocity** エンジンへと名称変更／改良されて、Analysis Services（分析サーバー）でも利用できるようになりました（**Tabular Mode** と呼ばれます）。この **xVelocity** エンジンによって、ビッグデータ／より大量のデータにも対応可能なサーバー版のインメモリ BI 機能を利用できるようになりました。そして、この xVelocity エンジンを RDB に応用したのが「**列ストア インデックス**」です（SQL Server 2012 に登場した列ストア インデックスは読み取り専用での提供）。

SQL Server 2014 からは、列ストア インデックスが更新可能になり「**クラスター化列ストア インデックス**」として提供され、OLTP 処理を高速化するための「**インメモリ OLTP**」機能も提供されました。

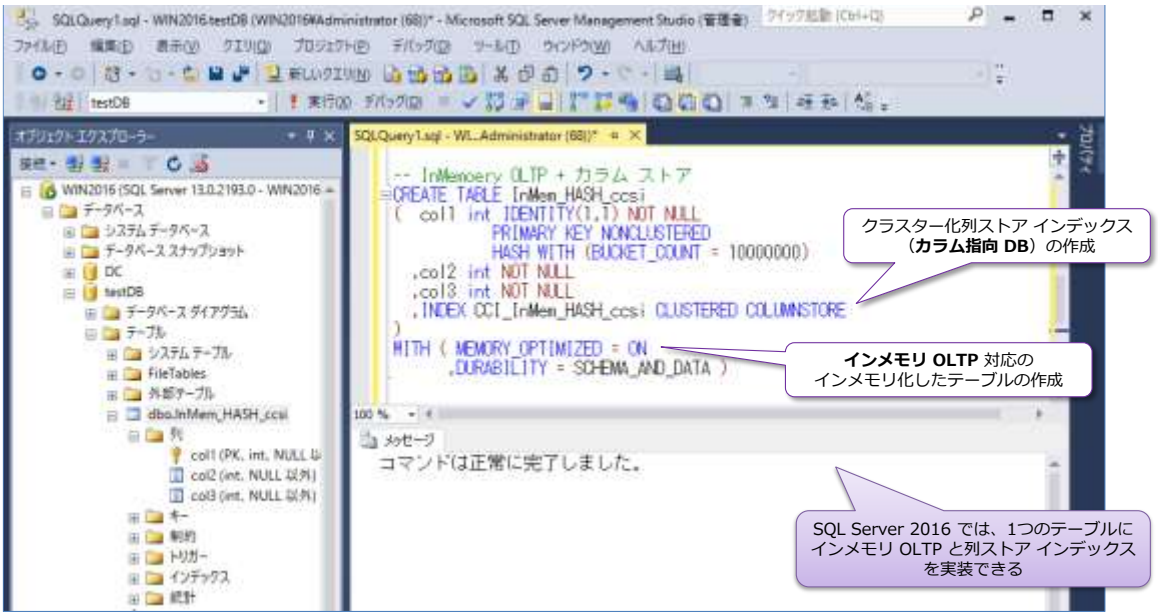
**列ストア インデックス**（Columnstore Index）は、昨今のビッグデータで流行の「**カラム指向データベース**」の SQL Server における実装です。この列ストア インデックスは、基本はインメモリで動作しますが（データ量が物理メモリに載りきる場合はインメモリで動作）、メモリに載りきらないデータがあった場合には、ディスクを利用して動作させることもできます。もし、完全なインメモリでの動作であったとすると、大量のデータを集計処理するために、その分のメモリが必要になり、ハードウェアへの投資が必要になってしまいます。その点、列ストア インデックスであれば、ハイブリッドな動作ができるので、スモール スタートが可能です。

そして今回の SQL Server 2016 では、「**インメモリ OLTP と列ストア インデックスのそれぞれが飛躍的に進化し、かつ両者が融合**」することによって、OLTP もデータ分析も両立できる機能が提供されました。これは、カラム指向データベースを 1 歩進化させて、リレーショナル データベースと融合させ、両者の良いとこどりをしたようなものです。

**SQL Server 2016 のインメモリ機能の進化**

| **インメモリ OLTP の飛躍的な進化** | **列ストア インデックスの飛躍的な進化** |
|---|---|
| ● インメモリ OLTP で列ストア インデックスのサポート（インメモリ OLTP のメモリ最適化テーブルにクラスター化列ストア インデックスを追加可能に）<br>● 最大サイズが 2TB に拡張（大量データに対応）<br>● 64コア以上での性能向上に対応<br>● 並列プランへの対応（メニー コア対応）<br>● TDE（透過的なデータ暗号化）のサポート<br>● ALTER／BIN2 以外の照合順序のサポート（既存環境からの移行しやすさが大幅に向上）<br>● 統計の自動更新のサポート、LOB のサポート | ● 列ストア インデックスの性能向上（バッチ モードの性能向上、集計のプッシュダウン、更新性能の向上、ウィンドウ関数のバッチ モード対応など）<br>● 非クラスター化列ストア インデックスが更新可能に<br>● 非クラスター化列ストア インデックスでフィルター列が利用可能に<br>● クラスター化列ストア インデックスでの追加の B-tree インデックスのサポート<br>● 主キー／外部キー制約のサポート<br>● 可用性グループでの読み取り可能セカンダリのサポート<br>● COMPRESS_DELAY（遅延圧縮）のサポート |
| **データベース エンジンの進化** | |
| ● Automatic Soft NUMA（メニーコア CPU での動作の最適化）<br>● Multiple Log Writer（ログ書き込み性能の向上）<br>● INSERT..SELECT のパラレル処理<br>● Dynamic Memory Object Scaling（CMemThread Wait の抑制）<br>● TDE の AES-NI 対応による性能向上<br>● AlwaysOn可用性グループでのログ転送性能の向上 | **CU1** での性能向上<br>・統計更新の性能向上<br>・列ストア インデックスの DMV 性能の向上<br>**CU2** での性能向上<br>・Spinlock Wait の抑制 |

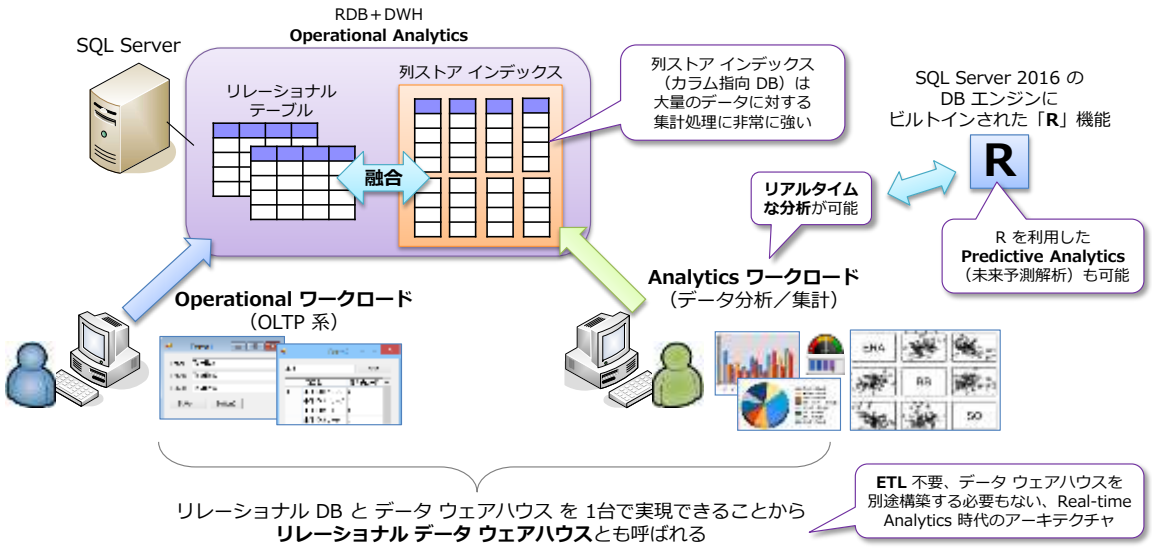
SQL Server 2016 では、OLTP に強い「**インメモリ OLTP**」と、大量のデータに対する分析／集計に強い「**カラム指向**」を同じテーブルに実装できるようになりました。具体的には、次のように **CREATE TABLE** ステートメントを実行するだけで、両者を同時に利用することができます。




SQL Server 2014 までは、別々のテーブルとして実装することはできましたが、上の画面のように 1 つのテーブルで両者を実装することはできませんでした。

SQL Server 2016 における OLTP とカラム指向の融合は、これからのデータベースのあり方を変えたとも言える大きな進化で、これによって、**リアルタイム分析**（リレーショナル データベースに蓄積したデータに対して、直接データ分析を行うこと）も、性能面で（今までのような）ストレスを感じることなくできるようになります。このような構成は **Operational Analytics** や **HTAP**（Hybrid Transaction and Analytics Processing）とも呼ばれています。

**SQL Server 2016 での Operational Analytics（リレーショナル DB とカラム指向の融合）**




データ ウェアハウスを構築しなくても、データ分析システムを構築できることから、リレーショナル データベース（RDB）とデータ ウェアハウス（DWH）を組み合わせた造語で「**リレーショナル データ ウェアハウス**」と呼ばれることもあります。

SQL Server 2016 からは、オープンソースの統計解析向けのプログラミング言語である「**R**」もビルトイン（データベース エンジンに統合）されているので、これを組み合わせれば、昨今の流行の

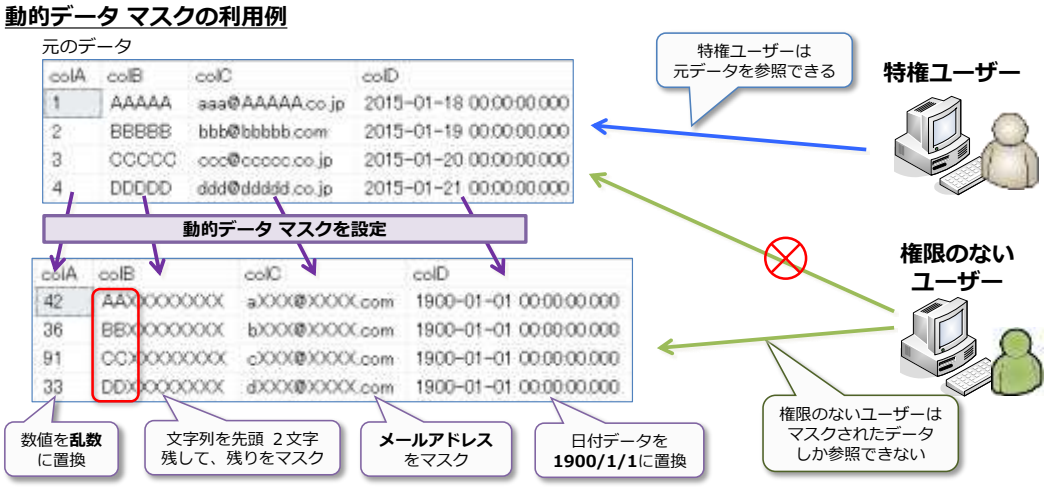
**Predictive Analytics** も可能になります。この SQL Server 2016 に統合された R は、エンタープライズ向けの R として定評のあった「**Revolution R**」を買収したものなので、性能面でも大きなメリットがあります（後述の早期導入事例での PROS 社では、従来の 100 倍の性能向上を実現しています）。

## セキュリティの強化

SQL Server 2016 には、**セキュリティを向上**させることができる機能が数多く提供されているので、これも SQL Server 2016 へ移行／アップグレードするメリットの 1 つです。その主なものは、次のとおりです。

- **動的データ マスク（Dynamic Data Masking）**

  顧客情報（クレジット カード番号やマイナンバーなど）や機密情報をマスク（別の値に置換）して、情報漏洩を防止できる機能




- **行レベル セキュリティ（Row Level Security）**

  行レベルのアクセス制御を実現できる機能で、ユーザーごとに、参照できる行データを制限することができる

- **Always Encrypted による列データの暗号化**

  ネットワーク上を流れるデータも、データベース内に格納されるデータも、**すべて暗号化**して格納できる機能（列データを暗号化して、アプリケーションも透過的に利用可能）

- **TDE（透過的なデータ暗号化）の性能向上、インメモリ OLTP 対応**

  Intel の AES-NI（AES 暗号化の処理を高速化するための命令セット）に対応して、TDE の性能を向上。インメモリ OLTP を利用している場合にも TDE が利用可能に

これらの機能の利用方法については、SQL Server 2016 自習書シリーズの No.1「**SQL Server 2016 の新機能の概要**」編で詳しく説明しているので、こちらもぜひご覧いただければと思います。

## SQL Server 2005 と Windows Server 2003／2003 R2 のサポート終了

**SQL Server 2005** や **SQL Server 2008** を動作させる **OS** としては、**Windows Server 2003** や **Windows Server 2003 R2** を利用しているという場合が多いのではないでしょうか。弊社のお客様にもたくさんいらっしゃいました。

しかし、**Windows Server 2003／2003 R2** は **2015 年 7 月 15 日**にサポート期限が切れ、**SQL Server 2005** は **2016 年 4 月 12 日**にサポート期限が切れています（延長サポートの終了）。このため、今後、もし新たな脆弱性が見つかったとしても修正プログラムが提供されなくなっています。なお、SQL Server 2008 と SQL Server 2008 R2 については、2014 年 7 月 8 日にメインストリーム サポートは終了して、現在は延長サポート期間に入っています。

これは、**OS** や **SQL Server** を移行／アップグレードする良い機会になります。「**OS が変わると、SQL Server の動作が心配**」という方もいらっしゃると思いますが、SQL Server は、**SQL Server だけで完結した製品**なので、OS の影響をほとんど受けません（OS が変わったとしても、SQL Server の動作にはほとんど影響がありません。影響があるのは NTFS アクセス許可ぐらいです）。弊社のお客様でも、SQL Server をアップグレードするタイミングで OS を変更することが多いのですが、該当マシンを **SQL Server 専用**として利用している場合には、今まで（SQL Server で）問題が起きたことはありません。

また、SQL Server 2016 は、**X64**（64 ビット）環境のみのサポートになるので、**32 ビットから 64 ビット環境へ移行**するという場合もあると思いますが、**SQL Server 専用**のマシンであれば、この影響もほとんどありません。SQL Server の機能は、32 ビットでも 64 ビットでも全く同じように動作させることができ、むしろ **64 ビット化することによる性能メリット**が（必ずといっても過言ではないほど）得られます。弊社のお客様でも、32 ビットを 64 ビット環境に移行することで、確実に性能向上しています。

これは最近のハードウェア性能の向上による影響も大きく、数年前に比べて**メモリが非常に低価格**になっていたり、**CPU コア数**の増加や、ファイバ チャネルなど**ストレージ接続**部分での高速化、**SSD／フラッシュ ストレージ**の低価格化など、数年前と同じ金額で、（当時と比べると）非常に高性能なマシンを購入することができます。例えば、メモリを **128GB** 搭載したマシンでも、エントリ レベルのサーバーであれば 200 万円以下で購入できてしまいます。

過去に、32 ビットから 64 ビット環境への移行時に問題となりえたのは、**IIS**（Web サーバー）上で **ASP／ASP.NET** を動作させていた場合で、この場合は（32 ビットの）COM コンポーネントを動作させるために、いくつかの修正が必要になりました（SQL Server のデータベースに関しては修正の必要はありませんでした）。

## SQL Server 2005 から SQL Server 2016 へのデータベースの移行も可能

詳しくは第 5 章の「**データベースの移行手順**」で説明しますが、最も基本となる**データベース エンジン**部分に関しては、**SQL Server 2005** から **SQL Server 2016** との間では大きな変化はない

ので、SQL Server 2005 上で取得したデータベースのバックアップを SQL Server 2016 上にリストアしても、多くの場合は何の問題もなく動作します。実際、弊社でも、今のところ 4 社のお客様のデータベースを移行して検証していますが、特に問題は発生していません（ストアド プロシージャやアプリケーションは問題なく動作し、OS を変更した影響もありません）。また、前述したような SQL Server 2016 からの新機能が利用できることによって、多くの処理で性能向上を実現することができるので、移行するメリットがあります。

## SQL Server 2005 からの移行メリット

**SQL Server 2005** から **SQL Server 2016** への移行を行う場合には、**SQL Server 2008** 以降で提供された機能も大きなメリットになります。SQL Server 2008 以降、たくさんの性能向上に関する機能が提供されていますが、それらをまとめると次のようになります。

| バージョン | 性能向上を実現できる新機能 |
|---|---|
| SQL Server 2008 | ● データ圧縮<br>● バックアップ圧縮<br>● パーティション環境でのパラレル処理<br>● スタージョインの性能向上<br>● データベース ミラーリングでのログ転送時の圧縮 |
| SQL Server 2008 R2 | ● Unicode 圧縮<br>● StreamInsight（インメモリのイベント処理機能）<br>● PowerPivot（インメモリの BI 機能） |
| SQL Server 2012 | ● メモリ マネージャーの改良（ラージ オブジェクトへの対応）<br>● 列ストア インデックス（読み取り専用）<br>● 15,000 パーティションのサポート<br>● AlwaysOn 可用性グループ（データベース ミラーリングの進化版）<br>● Analysis Services のインメモリ化（インメモリ BI のサーバー版）<br>● クラスター（WSFC）環境での tempdb のローカル配置 |
| SQL Server 2014 | ● インメモリ OLTP（OLTP 向けのインメモリ機能）<br>● 更新可能な列ストア インデックス（クラスター化列ストア インデックス）<br>● バッファ プール拡張（SSD をバッファ プールの一部として利用可能）<br>● Delayed Durability（トランザクション ログへの遅延書き込みが可能）<br>● SELECT INTO のパラレル処理<br>● 新しい基数推定（CE）のアルゴリズム |

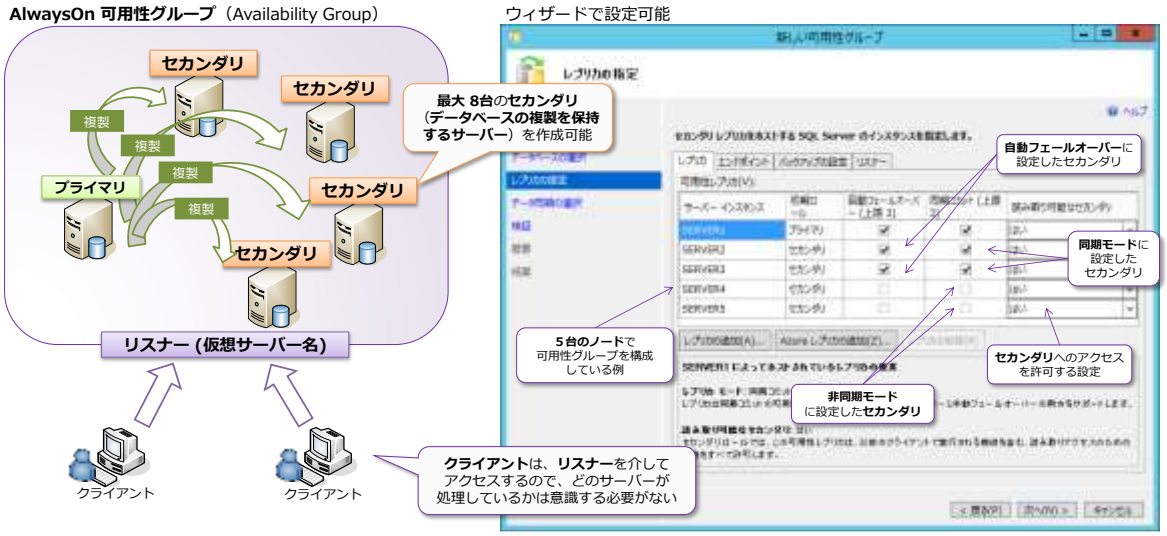
この中でも、**SQL Server 2008** から提供された「**データ圧縮**」や「**バックアップ圧縮**」は、大きな性能向上を実現できる機能なので、弊社のお客様でもたくさん利用されています。

SQL Server 2014 で提供された「**SELECT INTO のパラレル処理**」や「**Delayed Durability**」に関しても、弊社のお客様環境で大きな成果を得られています。SQL Server 2016 からは、SELECT INTO だけでなく、**INSERT..SELECT** についてもパラレル処理に対応したので、これらは多くの方々にメリットをもたらすと思います。

性能面の他にも、SQL Server 2008 以降では、**セキュリティ強化**や**可用性の向上**、機能面での強化（Transact-SQL など）などを実現するための多くの新機能が提供されています。その主なものは、次のとおりです。

| バージョン | 役立つ新機能 |
|---|---|
| SQL Server 2008 | ● SQL Server Audit による監査（コンプライアンスの実現が可能）<br>● 透過的なデータ暗号化<br>● リソース ガバナーによるリソース調整<br>● パフォーマンス データ コレクションによる性能監視<br>● FileStream データ型<br>● 新しい日付データ型（date／time／datetime2／datetimeoffset）<br>● MERGE ステートメント |
| SQL Server 2012 | ● AlwaysOn 可用性グループによる可用性の向上<br>● クラスター（WSFC）での SMB やマルチ サブネット構成のサポート<br>● Windows Server Core 対応<br>● T-SQL の強化（SEQUENCE、EOMONTH、CONCAT、THROW、IIF、FORMAT、OFFSET..FETCH、CHOOSE などのサポート） |
| SQL Server 2014 | ● AlwaysOn 可用性グループで 8台のセカンダリが作成可能<br>● クラスター（WSFC）の CVS（クラスター共有ボリューム）サポート<br>● リソース ガバナーの強化（I/O 制限など）<br>● インデックス再構築やパーティション Switch 時のロック優先度変更 |
| SQL Server 2016 | ● AlwaysOn 可用性グループの強化<br>自動フェールオーバーの台数が 2から 3 に増加、ログ転送性能の向上、ラウンドロビン レプリカ、TDE のサポート、DTC のサポート、ワークグループや Standard エディションでも利用可能<br>● トレースフラグ 1117、1118、4199 や MAXDOP、CE（基数推定）を DB 単位で設定可能に<br>● T-SQL の強化（DROP .. IF EXISTS、STRING_SPLIT、COMPRESS、FORMATMESSAGE、bcp と Bulk Insert で Unicode 対応 など）<br>● TRUNCATE TABLE でパーティション指定が可能に<br>● ライブ クエリ統計、クエリ ストアによる性能監視、プラン固定<br>● メモリ最適化テーブルへのレプリケーション |

**SQL Server 2012** からは、**AlwaysOn 可用性グループ**機能が提供されたことも大きな進化です。これは、データベース ミラーリングを進化させたような機能で、サーバー間での**データベースの複製**ができることはもちろん、セカンダリ（複製）側のデータを**リアルタイムに参照**できたり、**セカンダリを複数台**（SQL Server 2012 では最大 4 台、SQL Server 2014 からは最大 8 台）構成できたり、セカンダリ側でデータベースのバックアップを取得したりすることもできます。データベースの複製による可用性の向上だけでなく、セカンダリを有効活用することもできるようになっています。また、**クラウド DR**（セカンダリを Microsoft Azure 上に配置）も実現できます。




これらの SQL Server 2008 以降に提供された新機能については、**SQL Server 2014 自習書シ**

**リーズ**の「**SQL Server 2005 ユーザーのための SQL Server 2014**」で詳しく説明しているので、こちらもぜひご覧いただければと思います（ステップ バイ ステップ形式で簡単に試せるようにしています）。

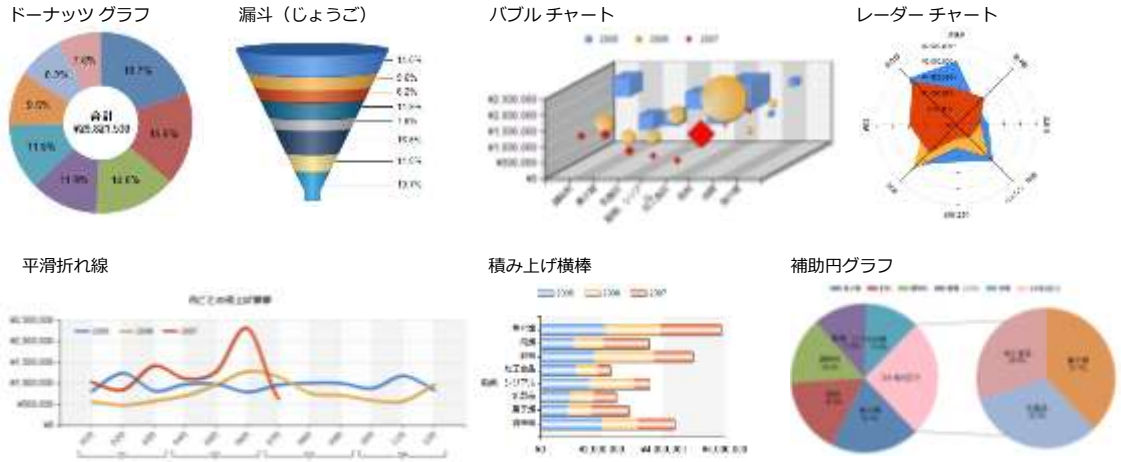
## SQL Server 2008 以降の DWH／BI 機能

**SQL Server 2005** から SQL Server 2016 へ移行する場合には、**SQL Server 2008** 以降で提供された以下の DWH／BI 機能を利用できることも大きなメリットになります。

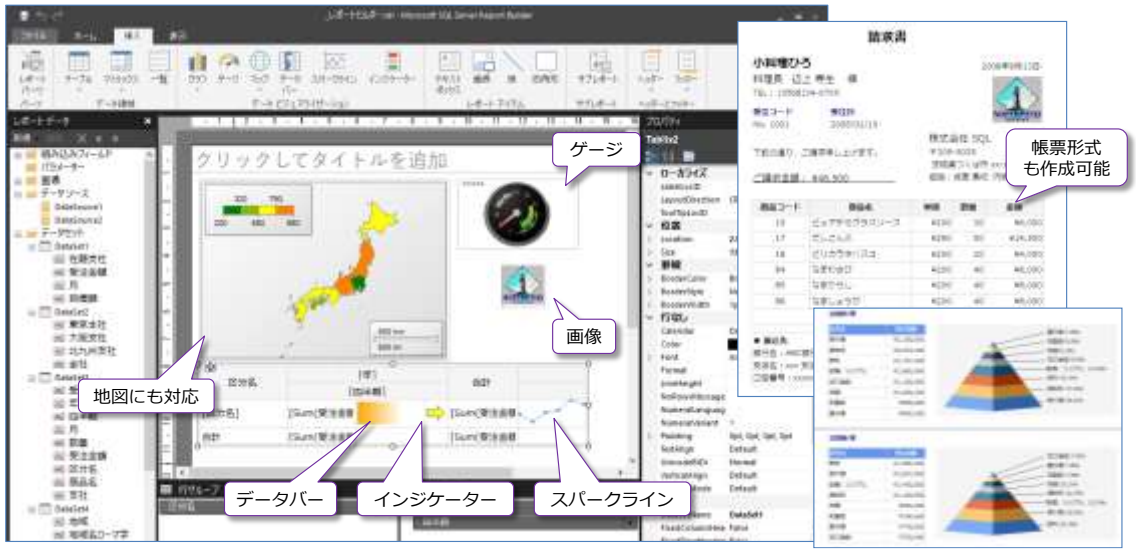
| バージョン | DWH／BI 関連の新機能 |
|---|---|
| SQL Server 2008 | ● データ圧縮<br>● パーティション環境でのパラレル処理、スタージョインの性能向上<br>● データ プロファイル タスク<br>● **Spatial データ型**対応（地図データの格納）<br>● **Reporting Services の大幅強化**（多彩なグラフ表現、IIS 非依存） |
| SQL Server 2008 R2 | ● PowerPivot（インメモリの BI 機能）<br>● MDS（マスター データ サービス）によるマスター データ管理<br>● Reporting Services の地図対応（.shp ファイルや Bing マップ対応） |
| SQL Server 2012 | ● 列ストア インデックス（読み取り専用）<br>● 15,000 パーティションのサポート<br>● **Analysis Services のインメモリ化**（**インメモリ BI** のサーバー版）<br>● DQS（Data Quality Services）によるデータ品質の管理<br>● 分析関数の提供（LEAD、LAG、CUME_DIST、OVER 句の拡張など）<br>● Power View によるデータ分析レポートの作成<br>● MDS の Excel アドイン<br>● Integration Services での CDC（変更データ キャプチャ）サポート<br>● Apache Hadoop への対応（Hadoop Connector の提供） |
| SQL Server 2014 | ● **クラスター化列ストア インデックス**（更新可能な列ストア インデックス）<br>● Power View の Analysis Services 多次元モデルへの対応 |
| SQL Server 2016 | ● インメモリ OLTP とクラスター化列ストア インデックスの融合<br>● **R 統合**（SQL Server R Services）<br>● **Reporting Services の大幅強化**<br>パラメーター ボックスのカスタマイズ、新しいグラフのサポート（サンバースト、ツリーマップ）、PowerPoint レンダリング、印刷コントロールの変更、iframe での埋込対応、Datazen 統合（モバイル レポート、新しいレポート マネージャー）など<br>● **Analysis Services の強化**<br>Tabular Model でのパーティションの並列処理のサポート、DAX エンジンの性能向上、DirectQuery の新しい実装と大幅な性能向上、計算テーブルのサポート、新しい DAX 関数（50以上）、双方向フィルターのサポート、スクリプト モデルの変更 など<br>● **Integration Services の強化**<br>Execute SQL タスクで R スクリプトのサポート、Azure Feature Pack、Hadoop（HDFS）のサポート、データ フローでのエラー時の列名のサポート、制御フローのテンプレート作成、AlwaysOn 可用性グループでの SSIS カタログ DB のサポート、AutoAdjustBufferSize プロパティ など<br>● PolyBase で **Hadoop** アクセス（HDFS） |

**SQL Server 2005** と比較すると、**Reporting Services** が特に大きく進化しています。レポートが非常に作りやすくなって、かつ多彩なグラフ表現ができるようになったことで、弊社のお客様にも大変好評です。Excel と同じようなレポートを作成することも可能で、弊社のお客様では、今まで Excel で（人手で）管理していたワークシート（分析レポート）を、Reporting Services に移行することで分析レポートの作成を自動化する、といった使い方をしていたりします。

Reporting Services の多彩なグラフ表現（Excel と同じようなグラフ／レポートを作成可能）



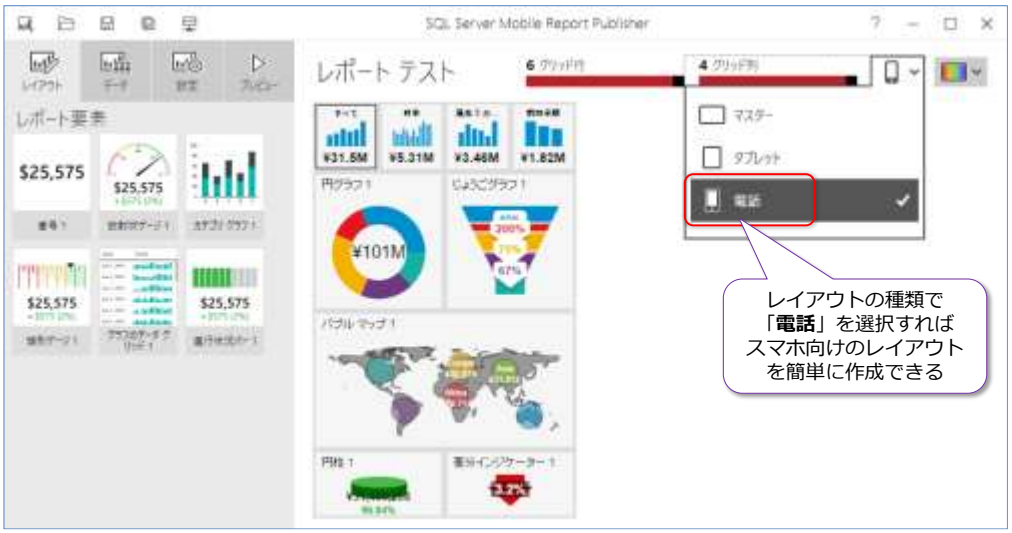

Reporting Services のレポート ビルダーでレポートを作成しているときの様子（フリー フォーマットでレポート作成が可能）




**Reporting Services** は、**SQL Server 2016** でも大幅に強化されています。特に、Datazen 統合によって、次のような見栄えの良いレポートを簡単に作成できるようになり、かつ**モバイル レポート**（スマートフォンやタブレット端末など向けのレポート）にも対応しました。

モバイル レポート パブリッシャー ツールを利用してモバイル レポートを作成している様子

スマホ向けのレイアウトを作成




スマホからモバイル レポートにアクセスしているときの様子


このモバイル レポートは、**HTML5** ベースなので、HTML5 に対応した Web ブラウザーであれば、スマホでもタブレットでも、どんな端末からでも同じようにレポートを参照することができます。SQL Server 2016 では、こうした見栄えの良いモバイル レポートを簡単に作成できるようになりました。これらの詳細については、SQL Server 2016 自習書シリーズの No.3「**Reporting Services の新機能**」編で説明しているので、こちらもぜひご覧いただければと思います。

**SQL Server 2016** では、**Analysis Services** も大幅に強化されていることも大きな移行メリットです。Analysis Services は、**SQL Server 2012** のときに**インメモリの BI エンジン**として **Tabular Mode**（表形式モード）が提供されて、この Tabular Mode が SQL Server 2016 でさらに進化しました（パーティションの並列処理のサポートや DirectQuery の大幅な性能向上など）。詳細については、SQL Server 2016 自習書シリーズの No.4「**Analysis Services の新機能**」編で説明しているので、こちらもぜひご覧いただければと思います。

## 1.2 SQL Server 2016 による早期導入事例（性能向上など）

既に SQL Server 2016 を早期導入／検証を行っている企業は、次のようにたくさんあります。

| 社名 | システム／サービスの特徴など | SQL Server 2016 の採用効果など |
|---|---|---|
| bwin | オンライン ゲームの提供。<br>1日あたり **1900万**ベット、<br>1日あたり **25万**以上のアクティブ ユーザー数 | **120万バッチ要求数/sec** を達成。<br>従来の Memcached や Cassandra、Microsoft AppFabric などを利用して構築していた独自のキャッシュ システムを、SQL Server 2016 のインメモリ OLTP を用いた ASP.NET SessionState に変更。<br>従来のシステムでは **19ノード**で **15万バッチ要求数/秒**だったところを、SQL Server 2016 では、わずか **1台**で **120万バッチ要求数/秒**を実現 |
| StackOverflow | 開発者向け Q&A サイトの提供。<br>年間ページビュー数は **85億**<br>毎年 **25%** のトラフィック増 | クラスター化列ストア インデックスを利用して 180GB/日 のログ データを 16GB/日 に圧縮（**10分の 1以下に圧縮**）。<br>ログ データを保存する仕組みとして理想的な技術。<br>AlwaysOn可用性グループの**ローリング アップグレード**のおかげで、**11台**の SQL Server を **48時間でアップグレード完了**。可用性グループは、ニューヨークにプライマリとセカンダリ（読み取り専用）、コロラドに DR 用のセカンダリを配置 |
| Jack Henry & Associates | 1万を超える**金融機関**向けに300以上のサービスと製品を提供、**ローン審査**（不良債権になる確率の予測など）。 | **1秒あたり 130万予測** を達成。<br>ローン予測（ローンが返済されるかどうか）において、<br>金利やローン条件、会員のクレジット スコアなどの変化要因を用いた What-If 予測での**スコアリング エンジン**として SQL Server 2016 にビルトインされた **R** を採用。リアルタイム予測分析（Predictive Analytics）を実現 |
| Heartland Bank | ニュージーランドの**銀行業** | 従来の SAS ベースのシステムを SQL Server 2016 と **Microsoft R** をベースにした Microsoftプラットフォームに置換。大手銀行や PayPal のようなサービスと競争するために、**投資予測**や**延滞分析**、**仲介／ブローカーのパフォーマンス分析**などを R プラットフォームに移行。<br>クレジットカード スコア（スコアリング エンジン）に関しても移行を検討中 |
| Fidelity National Information Services (FIS) | **金融機関**向けサービスを提供 | 非クラスター化列ストア インデックス（NCCI）を付与するだけで（アプリやクエリを変更することなく）、**分析クエリ**の性能が **4倍**、**10倍**、**20倍**に向上。<br>OLTP テーブルに NCCI を追加するだけだったので、ETL ルーチンを追加したり、DB の再設計、トランザクション処理の変更をする必要はなかった。 |
| Moneris Solutions Corporation | 北米の大手**決済処理**事業者 | アプリを変更することなく、SQL Server 2016 に変更しただけで、**10～20%**の性能向上を実現（ハードウェアは従来と同じものを利用）。**Always Encrypted** を利用すれば、システム上の全ての情報を保護にできるので、法令遵守だけでなく、**最先端のセキュリティ**を提供できる。インメモリ OLTP と Always Encrypted によってスピードとセキュリティの強化、AlwaysOn 可用性グループの読み取り専用セカンダリによって負荷分散も実現できる |
| PROS Holdings | Revenue Management を提供 | SQL Server 2016 にビルトインされた **R** を採用して、<br>従来のシステムよりも **100倍**速い性能を実現 |
| Derivco | UK のオンライン ゲーム提供 | AlwaysOn可用性グループを利用して、より効率的なバックアップと DR（災害復旧）を実現。まったく調整することなく、SQL Server 2016 にアップグレードしただけで、サポートできる同時プレーヤー数が **25%** も向上 |
| Infosys Limited | インドのITコンサルティング業 | SQL Server 2012 で構築していた EDW（エンタープライズ データ ウェアハウス）を、SQL Server 2016 へ移行（製品出荷後 2ヶ月半という目標で実施）。AlwaysOn可用性グループの**分散可用性グループ**（DAG）を利用したクラスターを構成して、リアルタイムにデータを利用できるようになったことで、ETL の実装を廃止。今後は、列ストア インデックスを利用したリアルタイム アナリティクスの実装を予定。 |
| Mediterranean Shipping Company (MSC) | 世界有数の**コンテナ運送**企業<br>米国だけでも 1日 5億 5,000 万件のトランザクション数 | SQL Server 2016 の Analysis Services（Tabular Model）と Power BI で、従来のレポート作成に時間がかかっていた手間を軽減。従業員はあらゆるデバイスの任意のブラウザから PowerBI を利用することができ、ほぼリアルタイムにギガバイトのデータに対してクエリを実行可能（以前のレポートだと、メールに添付された Excel ファイルを開くなどが必要だった）。SQL Server 2016 の早期検証で、クエリが **30～50%** 性能向上することを確認。 |
| Danske Fragtmand | デンマーク全域で数千台のトラックを使用して貨物を輸送する**物流業者** | 物流データをフラッシュ ドライブ上の SQL Server に保存。<br>現在、データセンターには 160TB ものフラッシュ ドライブがあり、250万回の IOPS と 2GB 以上のスループットを達成。<br>オールフラッシュの Windows Server 2016 と SQL Server 2016 を組み合わせたテストでは、BI クエリの 1つは従来のインフラよりも **9,521倍**も速く実行できることを確認 |
| Meijer | 米国の**大手スーパーマーケット**（スーパーセンター）チェーン<br>約 300店舗を展開 | 約 80万の商品点数に対して、SQL Server 2016 のインメモリ テクノロジーを利用して、Analysis Services（SSAS）のキューブに **200億件**分のデータを格納。このデータに対して Power BI からアクセスして、リアルタイムに近いデータ分析を実現 |
| CMC Markets | UK の**オンライン トレード**の世界的なプロバイダー。過去3年間で急成長し、57,000人以上のアクティブなクライアント | 8年前に SQL Server 2005 を採用。他のデータベース テクノロジを経験した後、SQL Server 2016 を採用して、パフォーマンスの低下を招くことなく、機能開発の速度を向上させて、顧客に新しい製品をより早く導入できるようになった |
| MedPoint Digital, Inc. | **医療業界**向けの Webベースのコミュニケーション・コラボレーションソフトウェアを提供 | Microsoft Azure 上の仮想マシン（IaaS）を利用して、<br>SQL Server 2012 から SQL Server 2016 にアップグレードして<br>現在のワークロード性能の最適化。**Always Encrypted** 機能によってクライアント データを安全に保つことが可能 |
| Saxo Bank | デンマークの**オンライン トレード**の世界的なプロバイダー | クラスター化列ストア インデックスを利用することで、<br>ストレージ容量を大幅に削減（**80%以下**のディスク容量で済んだ）。クエリ性能は、列ストアを利用したことで **10倍**に |
| M-Files | SaaS EIM のベンダー | もっとも重要な**分析クエリ**で **10倍**の性能向上<br>そして、このクエリの実行が OLTP ワークロードに影響を与えない |

## bwin 社では 120 万バッチ リクエスト数/秒を達成

**bwin** 社（欧州サッカーでの大手スポンサーとしてもお馴染みの、オンライン ゲームなどを提供している会社）では、SQL Server 2016 の**インメモリ OLTP** 機能を採用／移行することで、**1 秒あたり 120 万件**もの **Batch Request** を**処理**をできるようになりました。同社は、1 日あたりのベット数が **1,900 万**、1 日あたり **25 万**以上ものアクティブ ユーザーが訪れる人気サイトです。

**bwin** 社では、従来 Memcached や Cassandra、Microsoft AppFabric などを利用して構築していた**独自のキャッシュ システム**がありましたが、これを SQL Server 2016 のインメモリ OLTP を用いた ASP.NET SessionState ベースのものに変更しました。これによって、従来のシステムでは **19 ノード**で **15 万 Batch Request/秒**が限界だったところを、SQL Server 2016 では、わずか **1 台**で **120 万 Batch Request/秒**を実現できるようになりました。

こうした bwin 社でのキャッシュ システムに関する詳細は、以下の米マイクロソフトの SQL Server Customer Advisory Team（SQL CAT）チームの Blog に記載されています。

How bwin is using SQL Server 2016 In-Memory OLTP to achieve unprecedented performance and scale
https://blogs.msdn.microsoft.com/sqlcat/2016/10/26/how-bwin-is-using-sql-server-2016-in-memory-oltp-to-achieve-unprecedented-performance-and-scale

SQL Server Customer Advisory Team

How bwin is using SQL Server 2016 In-Memory OLTP to achieve unprecedented performance and scale

★★★★★

A customer blog

October 26, 2016 by Mike Weiner - SQLCAT // 3 Comments

Share 83 63 93

*Written by: Biljana Lazic (bwin – Senior DBA) and Rick Kutschera (bwin – Engineering Manager). Reviewed by: Mike Weiner (SQLCAT)*

bwin (part of GVC Holdings PLC) is one of Europe's leading online betting brands, and is synonymous with sports. Having offices situated in various locations across Europe, India, and the US, bwin is a leader in several markets including Germany, Belgium, France, Italy and Spain.

To be able to achieve our goals in these increasingly competitive markets, bwin's infrastructure is constantly being pushed to stay on top of today's – and sometimes even tomorrow's – technology demands. With around 19 million bets and over 250 000 active users per day, our performance and scale requirements are extraordinary. In this blog, we will discuss how we have adopted In-Memory OLTP with SQL Server 2016 to meet these demands.

Our Caching Systems:

For years we've depended on Microsoft AppFabric and other distributed cache systems such as Cassandra or Memcached, as one of the central pieces in our architecture, in order to meet the demanding requirements on our systems. All major components, including

このブログ記事の内容を要約すると、次のようになります。

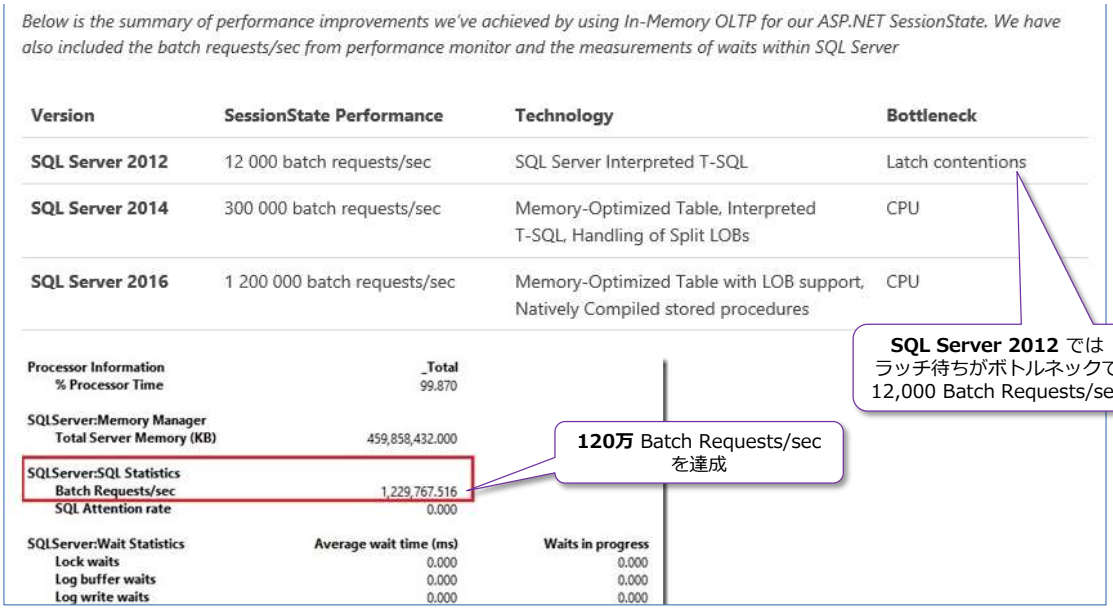
bwin 社では、従来の Memcached や Cassandra、Microsoft AppFabric などを利用して構築していた独自のキャッシュ システムでは、トランザクション量の増大によって、キャッシュ システムのワークロードが追いつかないという事態になり、ノード数を増やしても、安定性の問題に直面

して、可用性を落とすことにもなって、さらには、セットアップとメンテナンスの苦労もあったため、いろいろな部門にとっての作業負荷が増えていました。

そこで、これらの問題を解決するべく、キャッシュ システムとして採用したのが **ASP.NET** の **SessionState** です。SessionState は、元々利用していたキャッシュ システムでしたが、SQL Server 2012 で動作させていた SessionState は、ラッチ待ちのボトルネックによって 1 秒あたり 12,000 バッチ リクエスト数が限界でした。SQL Server 2014 が登場したときには、世界で一番最初にインメモリ OLTP を導入して、30 万 バッチ リクエスト数/秒を達成することができました。このときのボトルネックは、LOB（ラージ オブジェクト）データの Split 処理だったそうです。

SQL Server 2014 のインメモリ OLTP では、**varbinary(max)** や **varchar(max)** などの **LOB** に対応していなかったため、LOB データは分割処理する必要がありました（こうした処理を実現するためにネイティブ コンパイル ストアド プロシージャ化をすることもできていませんでした）。

SQL Server 2016 からは、インメモリ OLTP で **LOB がサポート**されたことによって、**ネイティブ コンパイル ストアド プロシージャ化も可能**になり、**1 秒あたり 120 万バッチ リクエスト数**という **4 倍**もの性能向上を実現することができました。




＊上の画面ショットは、ブログ記事「How bwin is using SQL Server 2016 In-Memory OLTP to achieve unprecedented performance and scale」より引用。吹出しは筆者が追加

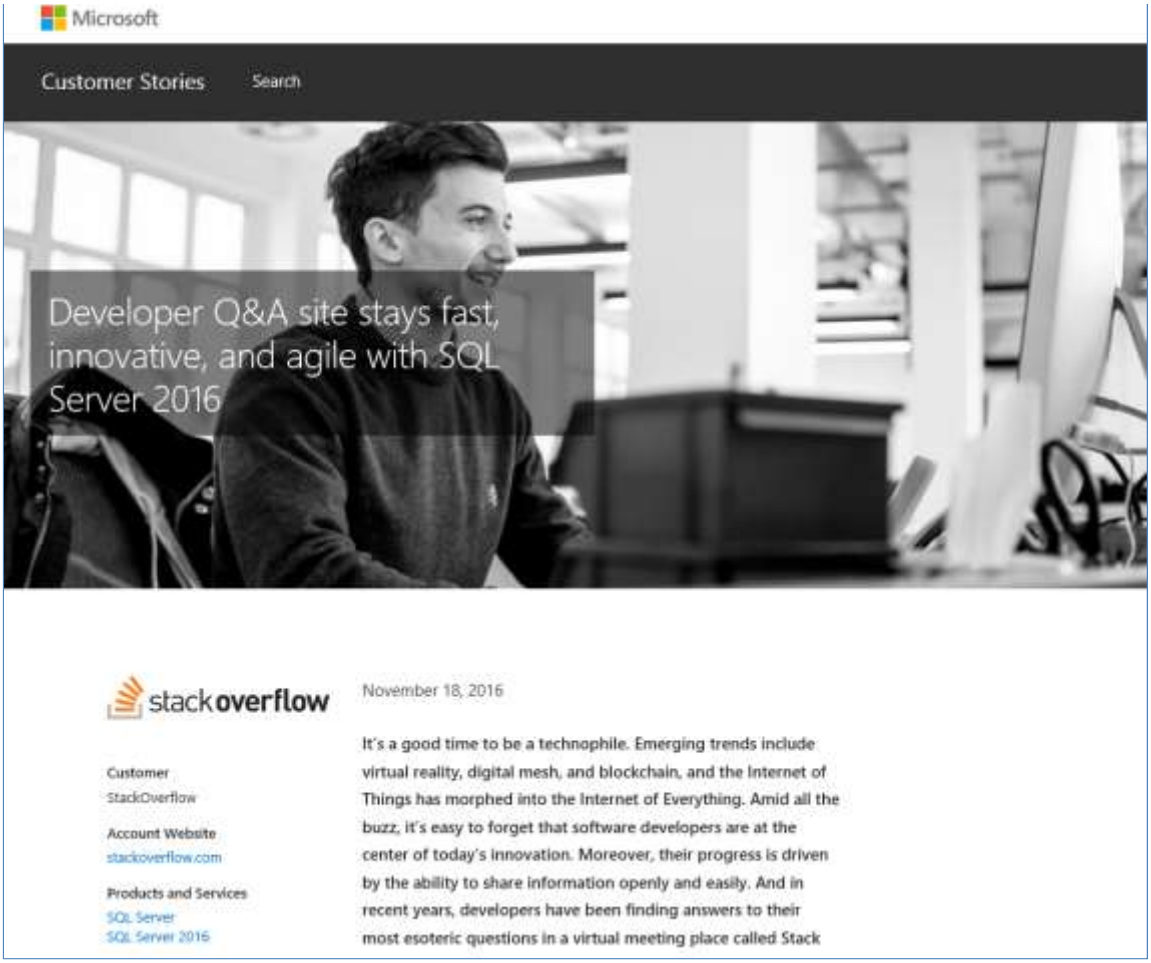
この 120 万バッチ リクエスト数を達成するにあたっては、途中 80 万バッチ リクエスト数/秒という壁（Spinlock の問題）がありましたが、これは CU2（SQL Server 2016 の累積的な更新プログラム 2）によって性能改善されました。

この 120 万バッチ リクエスト数という性能を得られたことで、キャッシュ システムに採用できるという確信を得られたとのことです。

## StackOverflow 社では列ストアの採用、AG のローリング アップグレード

StackOverflow は、私たちもよく利用している開発者のための Q&A コミュニティを運用している会社です。C# や VB、ASP.NET などプログラミングに関することなどで、検索エンジンで検索すると必ずといって良いほど上位に結果が表示される有名サイトです。年間ページビュー数は **85億**にも上り、毎年 **25%** のトラフィック増に直面しているそうです。

StackOverflow 社の詳細については、以下の米マイクロソフトの事例（Customer Stories）サイトに記載されています。
https://customers.microsoft.com/en-us/story/developer-qa-site-stays-fast-innovative-and-agile-with-sql-server-2016




この事例記事を要約すると、次のようになります。

この記事の中で、StackOverflow 社の Nick Craver 氏は「クラスター化列ストア インデックスを利用することで、1 日あたり **180GB** のデータを約 **16GB** に圧縮、**10 分の 1 以下に圧縮**ができた」と語っています。

また、同社では **SSD** を利用していますが、もし 180GB／日 のデータ サイズであったとすると、6 ヶ月分のデータを格納するために、**30TB** もの（非常に効果な）SSD を購入しなければならなくなります（例えば、2TB の PCI-e SSD を購入するだけでも数百万円のコストがかかるので、コ

ストおよび容量の両面から HDD を利用するのが現実的になってしまいます)。

一方、クラスター化列ストア インデックスを利用して **16GB／日** に圧縮できるのであれば、**6 ヶ月分のデータ**を格納したとしても、**約 2.9TB** で済みます。Nick Craver 氏は「SQL Server 2016 の列ストア インデックスを利用して、6 ヶ月分のログ データを SSD に格納できるのは大きなアドバンテージです。また、単なる圧縮ではなく、**実際のスピードも非常に優れて**います。**ログ データを保存する仕組みとして理想的な技術**です」と付け加えています。

蓄積したログ データは、**トラフィック パターンを分析**するために利用して、サイトを強化するための Insight（洞察）を得ています。例えば、「現在、会社内にリアルタイムに近いデータ パイプがあるので、StackOverflow のすべてのヒット数は**マップ**（**地図**）上に表示されます。これによって、ロサンゼルスにいる Ruby 開発者の数や、ボットネットをブロックする方法、または**より良い広告を出す方法**を理解できます。また、より良い質問をどのように提示して、より反応する可能性が高いかどうかなど、履歴データをすぐに参照できるということは、シミュレーションを迅速に再実行できることを意味しています」。

StackOverflow 社では、**AlwaysOn 可用性グループ**を利用して、**ニューヨーク**にプライマリとセカンダリ（読み取り専用）、**コロラド**に DR（災害対策）用のセカンダリを配置しています。SQL Server 2016 へのアップグレードにあたっては、AlwaysOn 可用性グループの**ローリング アップグレード**機能のおかげで、**11 台**の SQL Server を、わずか **48 時間でアップグレードを完了**させることができたそうです。

## Jack Henry & Associates 社では 130 万予測/秒を達成（R によるローン予測）

1 万を超える金融機関向けに 300 以上のサービスと製品を提供している **Jack Henry & Associates** 社では、ローン申請における**スコアリング・エンジン**として SQL Server 2016 およびビルドインされた **R**（**ScaleR**）を利用して、インテリジェントな **EDW**（エンタープライズ データ ウェアハウス）を構築しています。

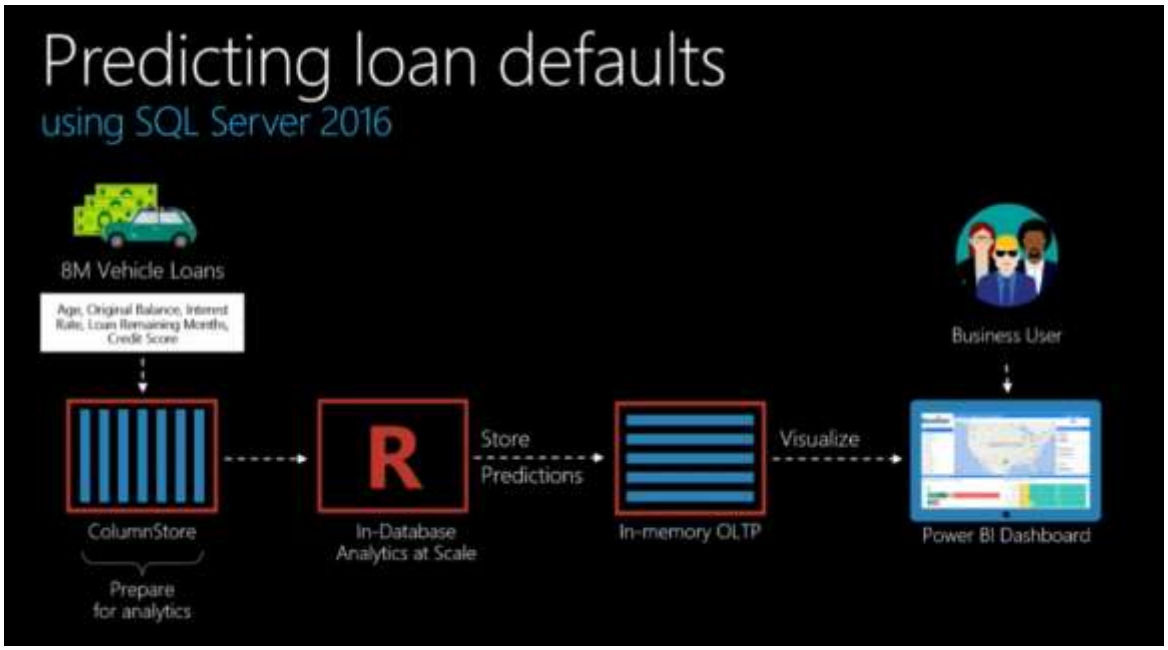
昨年開催された PASS Summit 2016 の基調講演では、以下のスライドが紹介されていました。


*上の画面は、PASS Summit 2016 での基調講演より引用

**Jack Henry & Associates** 社では、**1 秒あたり 130 万**もの**ローン申請**（ローンが返済されるか

どうかのローン予測）を処理できたという事例です。

これの具体的な構成については、次のように紹介されていました。




*上の画面は、PASS Summit 2016 での基調講演より引用

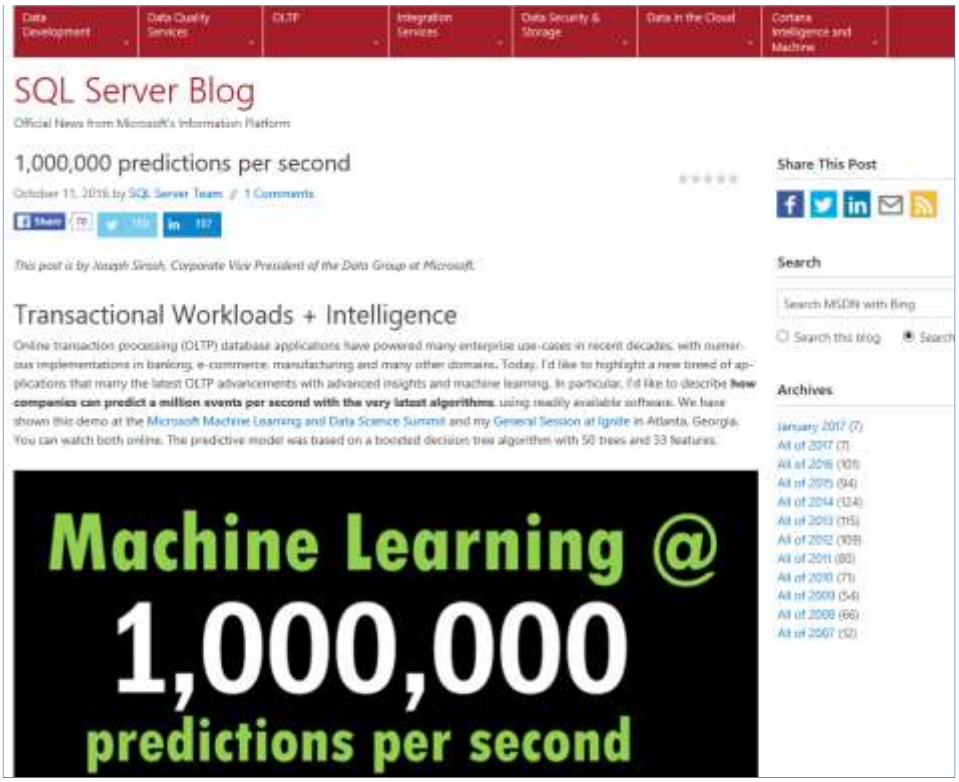
SQL Server 2016 の**列ストア インデックス**と**インメモリ OLTP**、SQL Server にビルトインされた **R**（**ScaleR**）を利用して、ローン予測を処理しています。

ローンが返済されるかどうか（債務不履行の可能性）を判断するには、金利やローン条件、会員のクレジット スコアなどの変化要因が影響し、不良債権になる確率を予測するインテリジェントな方法が必要になります。例えば、金利の上昇をシナリオに **What-If 分析**を行ったりするなどが必要です。

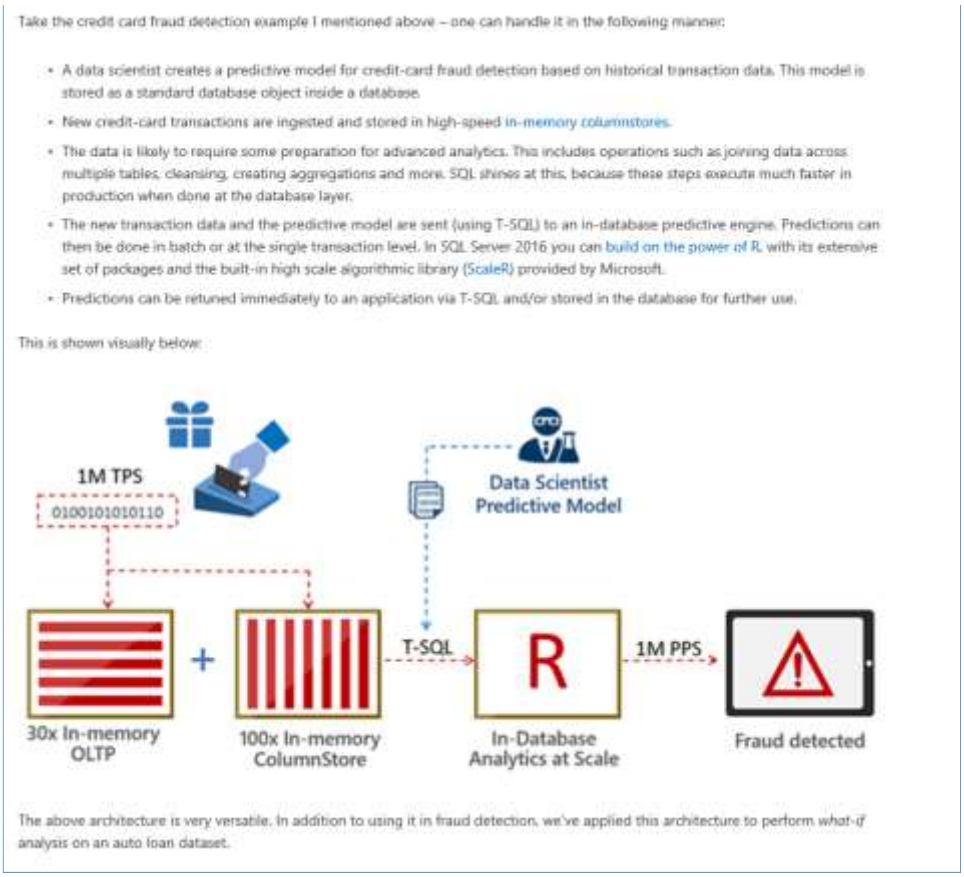
SQL Server 2016 にビルトインされた R では、ローンのスコアリング モデルを、ストアド プロシージャで呼び出して、パラレル スレッドで処理することができます。このような**リアルタイム予測分析**（Real-Time Predictive Analytics）は、ビジネスの柔軟性を飛躍的に高め、高い収益性をもたらす重要な要素になります。

こうしたスコアリング エンジン（予測分析）に関しては、以下の米マイクロソフトの SQL Server チームの Blog の記事が参考になります。

1,000,000 predictions per second
https://blogs.technet.microsoft.com/dataplatforminsider/2016/10/11/1000000-predictions-per-second/




このブログ記事では、**クレジット カード取引**の場合の例も、次のように紹介されています。




この記事を要約すると、次のようになります。

金融サービス企業では、クレジット カードでの取引が正当なものであるか、**不正使用**（**Credit Card**

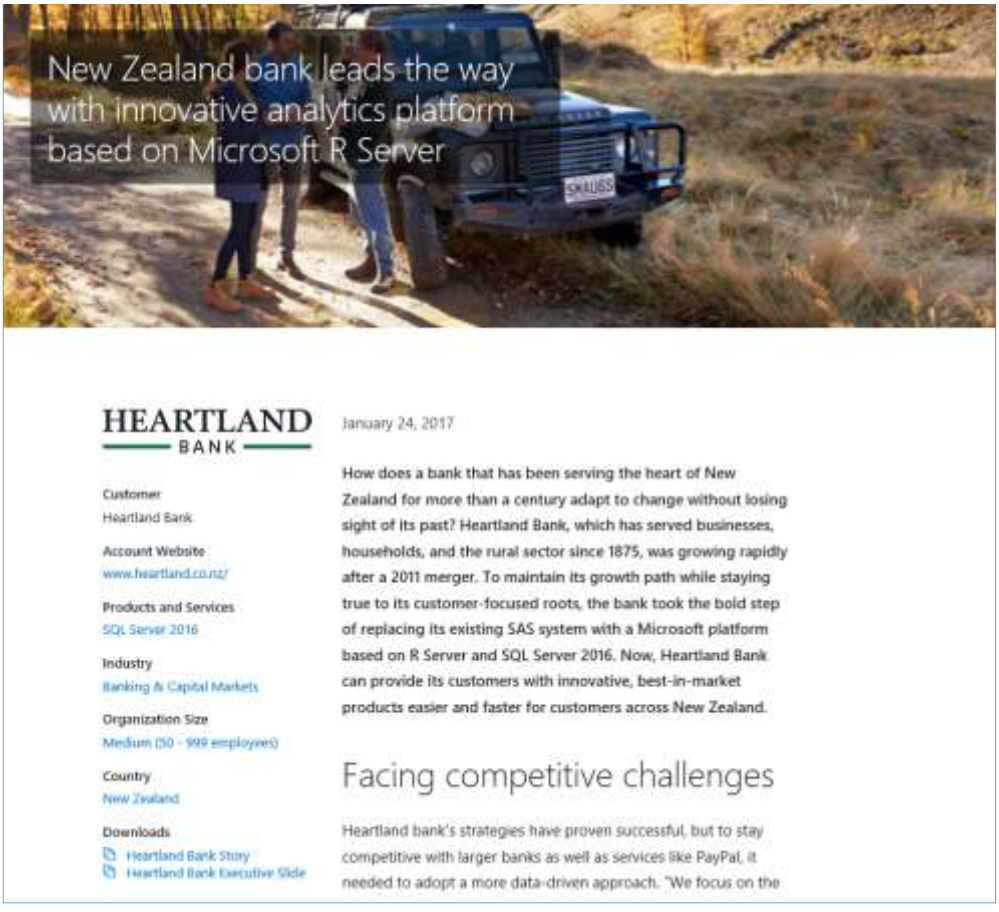
**Fraud**）なのかどうかを判断する必要があります。この判断には**予測モデル**（predictive model）がよく利用されます。

1 秒あたりのトランザクション数（TPS）が増えれば、1 秒あたりの予測数（PPS）も増えます。例えば、昨年の VISA ネットワークでの 56,000TPS、年間取引量は 10 億 100 以上、予測と意志決定には、OLTP と高速な予測エンジンを備えた強力なプラットフォームが必要になります。例えば、顧客数の増大を想定して、1 秒あたり 100 万予測（100 万 PPS）を達成できるプラットフォームにできないか、これを実現するための SQL Server 2016 における設計方法が、前掲の図のインメモリ OLTP と列ストア インデックス、R の組み合わせです。

## Heartland Bank では SAS を SQL Server 2016＋R に置換

**Heartland Bank** は、ニュージーランドの銀行業ですが、従来の SAS ベースのシステムを **SQL Server 2016** と **Microsoft R** をベースにした Microsoft プラットフォームに置き換えています。大手銀行や PayPal のようなサービスと競争するために、**投資予測**や**延滞分析**、**仲介／ブローカーのパフォーマンス分析**などを R プラットフォームに移行して、**クレジットカード スコア**（スコアリング エンジン）に関しても移行を検討中です。

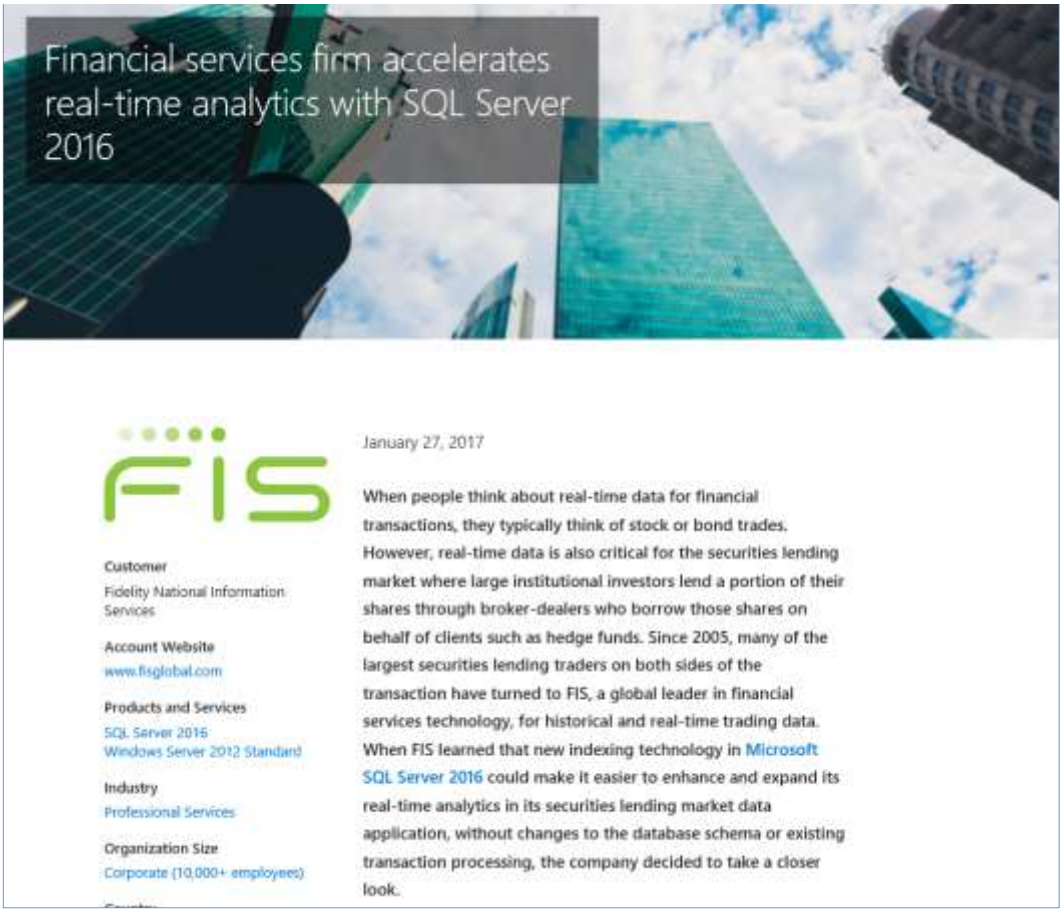
Heartland Bank の詳細については、以下の米マイクロソフトの事例（Customer Stories）サイトに記載されています。
https://customers.microsoft.com/en-us/story/heartlandbank



## FIS 社では、NCCI を付与するだけで分析クエリの性能が 4、10、20 倍に向上

**金融機関**向けにサービスを提供している **FIS**（Fidelity National Information Services）社では、SQL Server 2016 の非クラスター化列ストア インデックス（NCCI）を付与するだけで、アプリケーションやクエリを一切変更することなく、**分析クエリ**の性能が **4 倍**、**10 倍**、**20 倍**に向上することを確認しました。OLTP テーブルに NCCI を追加するだけだったので、ETL ルーチンを追加したり、DB を再設計したり、トランザクション処理の変更をしたりする必要はありませんでした。

FIS 社の詳細については、以下の米マイクロソフトの事例（Customer Stories）サイトに記載されています。
https://customers.microsoft.com/en-us/story/financial-services-firm-accelerates-real-time-analytics-with-sql-server

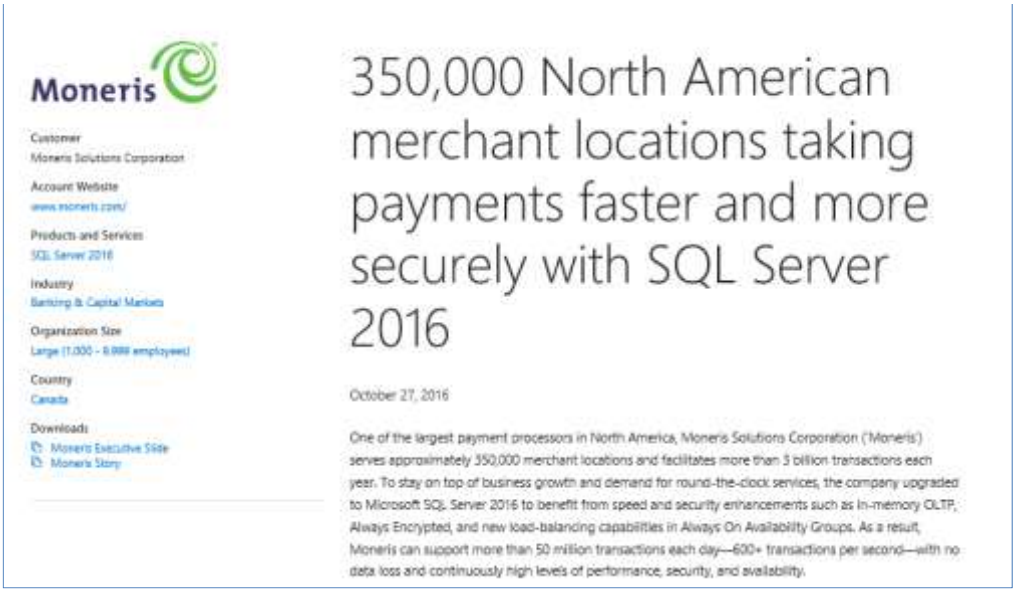



## Moneris Solutions ではアプリを変更することなく 10～20%の性能向上を実現

北米最大の**決済処理**事業者の 1 つである **Moneris Solutions Corporation** では、アプリケーションを一切変更することなく、SQL Server 2016 に変更しただけで、**10～20%**の性能向上を実現しました（ハードウェアは従来と同じものを利用）。

**Always Encrypted** を利用すれば、システム上の全ての情報を保護にできるので、法令遵守だけでなく、**最先端のセキュリティ**を提供できます。インメモリ OLTP と Always Encrypted によってスピードとセキュリティの強化、AlwaysOn 可用性グループの読み取り専用セカンダリによって負荷分散も実現可能です。

Moneris Solutions 社の詳細については、以下の米マイクロソフトの事例（Customer Stories）サイトに記載されています。
https://customers.microsoft.com/en-us/story/moneris




## PROS では 100 倍の性能向上（R を採用）

Revenue Management を提供している **PROS Holdings** 社では、SQL Server 2016 にビルトインされた **R** を採用して、従来のシステムよりも **100 倍**速い性能を実現しています。これについては、昨年開催された PASS Summit 2016 の基調講演で、次のように紹介されていました。


*上の画面は、PASS Summit 2016 での基調講演より引用

PROS 社の事例については、次の URL で動画を参照することもできます。

PROS Featured in Microsoft SQL Server 2016 Launch
http://info.pros.com/SQL-2016_Watch-Now-View.html

## Derivco ではアップグレードしただけで同時プレーヤー数が 25%も向上

オンライン ゲームを提供している UK の **Derivco** 社では、**AlwaysOn 可用性グループ**を利用して、より効率的なバックアップと **DR**(**災害復旧**)を実現し、まったく調整することなく、SQL Server 2016 にアップグレードしただけで、サポートできる同時プレーヤー数が **25%** も向上しました。

Derivco 社の詳細については、以下の米マイクロソフトの事例 (Customer Stories) サイトに記載されています。
https://customers.microsoft.com/en-us/story/when-an-online-gaming-company-doesnt-want-to-risk-its-future-it-bets-on-sql-server-2016


## Meijer では Analysis Services と Power BI でリアルタイムに近いデータ分析

米国の大手スーパーマーケット チェーン (約 300 店舗を展開) の **Meijer** 社では、約 80 万の商品点数に対して、SQL Server 2016 のインメモリ テクノロジーを利用して、Analysis Services (SSAS) のキューブに **200 億件**分のデータを格納、このデータに対して Power BI からアクセスして、リアルタイムに近いデータ分析を実現しています。

Meijer 社の詳細については、以下の米マイクロソフトの事例 (Customer Stories) サイトに記載されています。
https://customers.microsoft.com/en-us/story/top-supermarket-chain-gains-insight-and-boosts-profitability-with-microsoft-power-bi

## その他の事例

その他の事例に関しては、以下の表に URL をまとめました。

| 社名 | システム/サービスの特徴など | SQL Server 2016 の採用効果など |
|---|---|---|
| Infosys Limited | インドの<br>ITコンサルティング業 | SQL Server 2012 で構築していた EDW（エンタープライズ データ ウェアハウス）を、SQL Server 2016 へ移行（製品出荷後 2ヶ月半という目標で実施）。AlwaysOn可用性グループの**分散可用性グループ**（DAG）を利用したクラスターを構成して、リアルタイムにデータを利用できるようになったことで、ETL の実装を廃止。今後は、列ストア インデックスを利用したリアルタイム アナリティクスの実装を予定。<br>https://customers.microsoft.com/en-us/story/infosyssql |
| Mediterranean Shipping Company (MSC) | 世界有数の**コンテナ運送**企業<br>米国だけでも 1日 5億 5,000 万件のトランザクション数 | SQL Server 2016 の Analysis Services（Tabular Model）と Power BI で、従来のレポート作成に時間がかかっていた手間を軽減。従業員はあらゆるデバイスの任意のブラウザから PowerBI を利用することができ、ほぼリアルタイムにギガバイトのデータに対してクエリを実行可能（以前のレポートだと、メールに添付された Excel ファイルを開くなどが必要だった）。SQL Server 2016 の早期検証で、クエリが **30～50%** 性能向上することを確認。<br>https://customers.microsoft.com/en-us/story/getting-the-worlds-goods-where-they-need-to-go |
| Danske Fragtmand | デンマーク全域で数千台のトラックを使用して貨物を輸送する**物流業者** | 物流データをフラッシュ ドライブ上の SQL Server に保存。<br>現在、データセンターには 160TB ものフラッシュ ドライブがあり、250万回の IOPS と 2GB 以上のスループットを達成。オールフラッシュの Windows Server 2016 と SQL Server 2016 を組み合わせたテストでは、BI クエリの 1つは従来のインフラよりも **9,521倍**も速く実行できることを確認<br>https://customers.microsoft.com/en-us/story/danske-fragtmaend-innovates-with-sql-server-on-storage-replica |
| CMC Markets | UK の**オンライン トレード**の世界的なプロバイダー。過去3年間で急成長、57,000人以上のアクティブなクライアント | 8年前に SQL Server 2005 を採用。他のデータベース テクノロジを経験した後、SQL Server 2016 を採用して、パフォーマンスの低下を招くことなく、機能開発の速度を向上させて、顧客に新しい製品をより早く導入できるようになった<br>https://customers.microsoft.com/en-us/story/because-a-second-is-too-long-to-wait-this-financial-services-firm-is-updating-its-trading-software |
| MedPoint Digital, Inc. | **医療業界**向けの Webベースのコミュニケーション・コラボレーションソフトウェアを提供 | Microsoft Azure 上の仮想マシン（IaaS）を利用して、<br>SQL Server 2012 から SQL Server 2016 にアップグレードして<br>現在のワークロード性能の最適化。**Always Encrypted** 機能によってクライアント データを安全に保つことが可能<br>https://customers.microsoft.com/en-us/story/medpointdigital |
| Saxo Bank | デンマークの**オンライン トレード**の世界的なプロバイダー | クラスター化列ストア インデックスを利用することで、<br>ストレージ容量を大幅に削減（**80%以下**のディスク容量で済んだ）。クエリ性能は、列ストアを利用したことで **10倍**に<br>https://blogs.technet.microsoft.com/dataplatforminsider/2016/06/01/sql-server-2016-is-generally-available-today/ |
| M-Files | SaaS EIM のベンダー | もっとも重要な**分析クエリ**で **10倍**の性能向上<br>そして、このクエリの実行が OLTP ワークロードに影響を与えない<br>https://blogs.technet.microsoft.com/dataplatforminsider/2016/06/01/sql-server-2016-is-generally-available-today/ |

## SQL Server 2016 の参考情報

SQL Server 2016 に関しては、以下のブログ記事もお勧めです。

Migrating SAP workloads to SQL Server just got 2.5x faster
https://blogs.msdn.microsoft.com/sql_server_team/migrating-sap-workloads-to-sql-server-just-got-2-5x-faster/

SQL Server 2016 – It Just Runs Faster Announcement
https://blogs.msdn.microsoft.com/psssql/2016/02/23/sql-2016-it-just-runs-faster-announcement/

SQL Server 2016 is generally available today
https://blogs.technet.microsoft.com/dataplatforminsider/2016/06/01/sql-server-2016-is-generally-available-today/

## SQL Server 2016 は TPC-H ベンチマークの 10TB でワールド レコード

**TPC-H** は、データ ウェアハウス／意志決定支援システム向けの**ベンチマーク テスト**として有名なものですが、SQL Server 2016 は、**10TB**（テラ バイト）規模のベンチマーク テストで**世界記録**（ワールド レコード）を更新しました。執筆時点（2016 年 12 月）では、Non-Clustered の 10TB 部門では、上位 1、2、3 位を SQL Server 2016、4 位～8 位を SQL Server 2014 が獲得して、SQL Server だけで**上位 1 位～8 位を独占**しています（以下の URL で最新のベンチマーク結果を参照できます）。

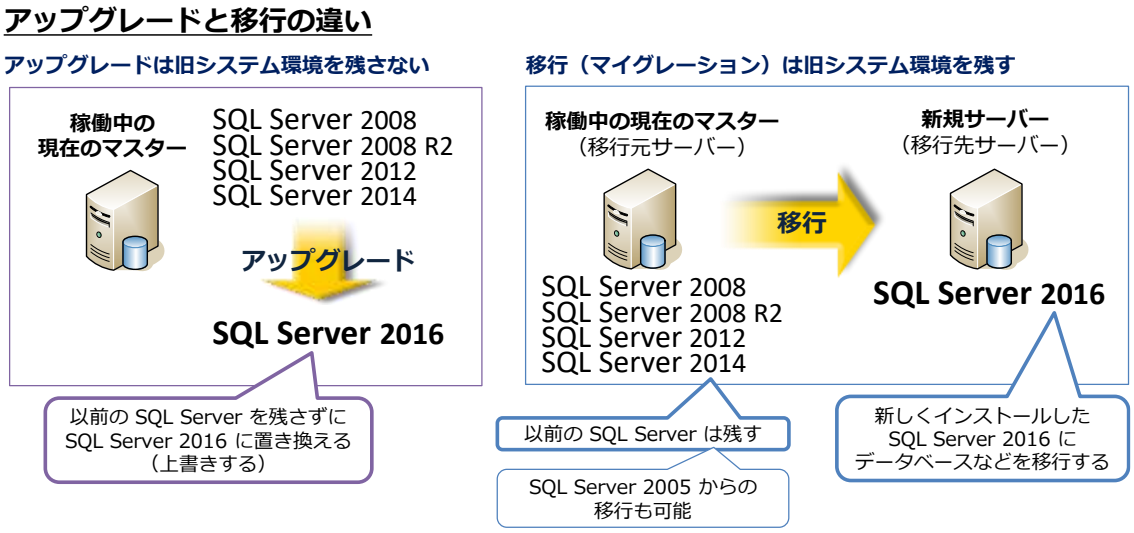
TPC-H - Top Ten Performance Results - Non-Clustered
http://www.tpc.org/tpch/results/tpch_perf_results.asp?resulttype=noncluster

この世界記録に関しては、SQL Server チームの Blog にも投稿されているので、以下の記事が参考になると思います。

SQL Server Team Blog：
SQL Server 2016 posts world record TPC-H 10 TB benchmark
http://blogs.technet.microsoft.com/dataplatforminsider/2016/07/18/sql-server-2016-posts-world-record-tpc-h-10-tb-benchmark/

## 1.3 SQL Server 2016 へのアップグレードと移行の概要

「**アップグレード**」(Upgrade) と「**移行**」(マイグレーション : Migration) は、どちらも似たような用語ですが、厳密には、旧システム環境を残すかどうか (サイド バイ サイドで、以前のバージョンの SQL Server と SQL Server 2016 を並列実行できるようにするかどうか) という違いがあります。次のように、**アップグレード**の場合は旧システム環境を残さず、**移行**の場合は旧システム環境を残します。




**アップグレード**は、旧システム環境を残さずに、**以前のバージョンの SQL Server** (SQL Server 2008 や 2008 R2／2012／2014) を、**SQL Server 2016** に完全に置き換えます (上書きします)。なお、同一マシン内でのアップグレードは、「**インプレース アップグレード**」とも呼ばれています。

一方、**移行** (マイグレーション) では、旧システム環境 (以前の SQL Server) とは別に、**新しい環境** (SQL Server 2016) を作成して、そこにデータベースや各種機能を移行します。

### アップグレード パス

**SQL Server 2016** は、以下のバージョンの **SQL Server** からの**アップグレード** (インプレース アップグレード) を行うことができます。

- **SQL Server 2008 SP4** 以降
- **SQL Server 2008 R2 SP3** 以降
- **SQL Server 2012 SP2** 以降
- **SQL Server 2014 RTM** 以降

SQL Server 2016 からは、**SQL Server 2005** からのインプレース アップグレードがサポートされなくなりました (第 5 章で紹介するデータベースの移行は可能です)。

**SQL Server 2008** は **SP4**、**SQL Server 2008 R2** は **SP3**、**SQL Server 2012** は **SP2** 以

降を適用しておくことで、インプレース アップグレードを行うことができます。

なお、古いバージョンの SQL Server 上で、ユーザー データベースの互換性レベルが **90**（SQL Server 2005 レベル）以下のものを利用している場合は、アップグレード時に、そのデータベースの互換性レベルが **100**（SQL Server 2008 レベル）に自動的に上がりますが、詳しくは第 3 章で説明します。

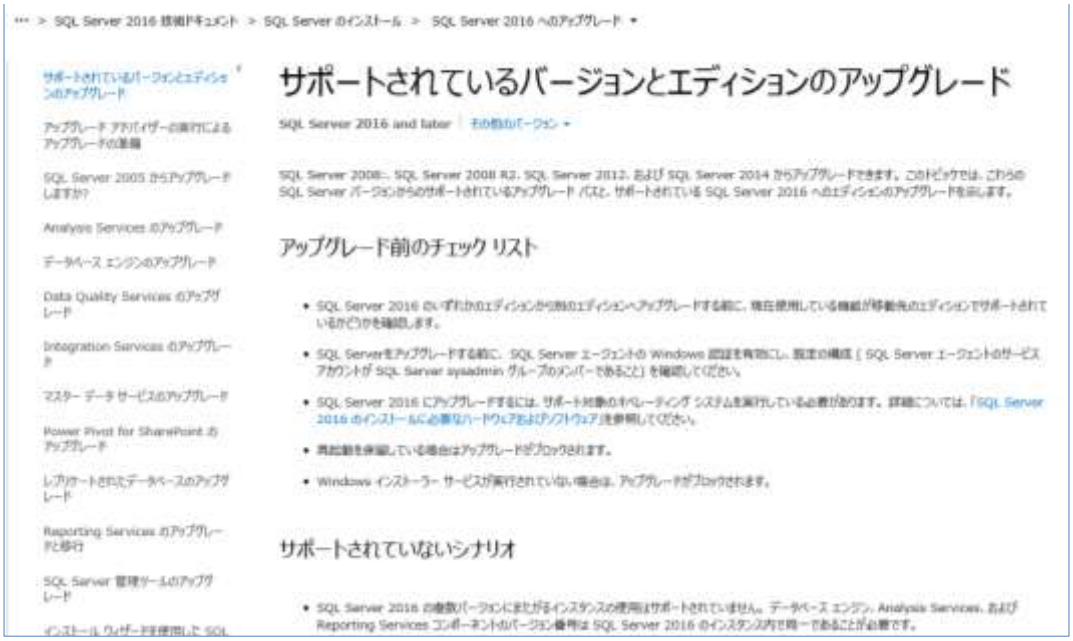
アップグレード時に**エディションを変更する**場合は、基本的には**上位互換**（**Standard** エディションを **Enterprise** エディションへ変更するなど、上位のエディションへの変更）であれば問題ありません。一方、下位のエディションへの変更（**Enterprise** エディションを **Standard** エディションへ変更するなど）はサポートされていません。

**クロス プラットフォーム**（**32 ビット**から **64 ビット**への変更）に関しては、インプレース アップグレードがサポートされていないので、SQL Server 2016 の新規インストールを行って、**データベースの移行**（**マイグレーション**）を行う必要があります。**データベースの移行**であれば、**32 ビット**も **64 ビット**も関係ありません（移行の詳細は、第 5 章で説明します）

なお、**Microsoft Azure** などのクラウド環境への変更を行う場合にも、第 5 章の「**移行**」の手順で行うことができます。完全に別のサーバー（同じ名前の別サーバーではなく、異なる名前の別サーバー）への変更を行う場合には、移行を利用するようにします（移行の手順も簡単です）。

その他のアップグレード条件に関しては、オンライン ブックの以下のトピックに記載されているので、一読しておくことをお勧めします。

SQL Server 2016 へのアップグレード
サポートされているバージョンとエディションのアップグレード
http://msdn.microsoft.com/ja-jp/library/ms143393.aspx




注意： このドキュメント内の **Service Pack（SP）の条件**に関しては、（執筆時点では）記述が間違っているので、本文中で説明した SP（SQL Server 2008 なら **SP4**、2008 R2 なら **SP3**、2012 なら **SP2**）に置き換えて読み進めてください。

## 移行可能なデータベース

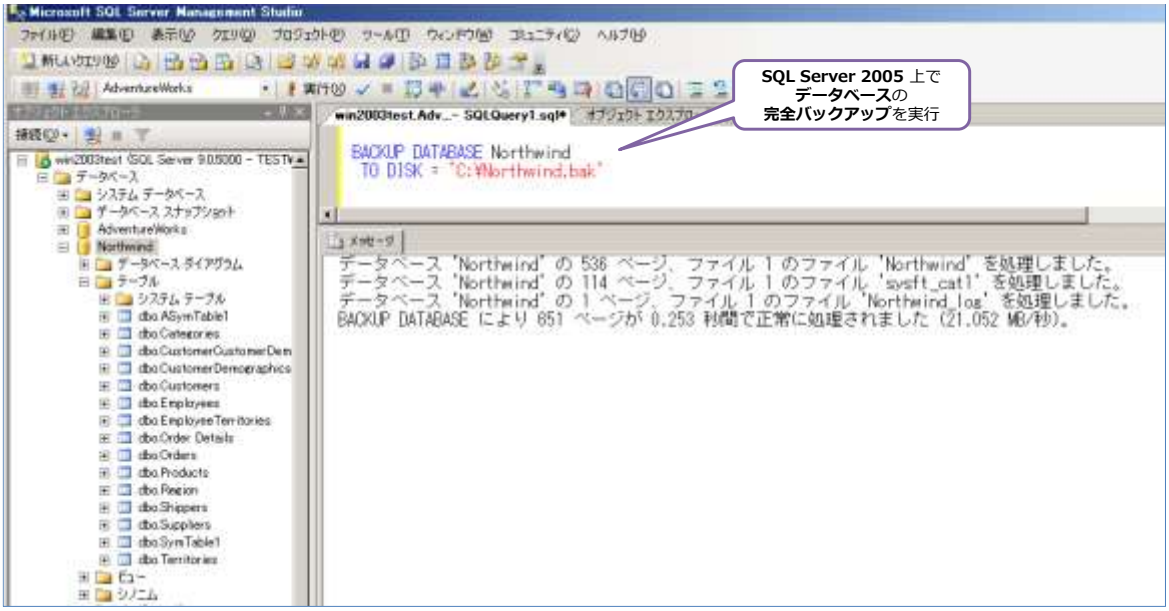
**SQL Server 2016** への**移行**（マイグレーション）が可能なデータベースは、次のとおりです。

- **SQL Server 2005** 上のデータベース
- **SQL Server 2008** 上のデータベース
- **SQL Server 2008 R2** 上のデータベース
- **SQL Server 2012** 上のデータベース
- **SQL Server 2014** 上のデータベース

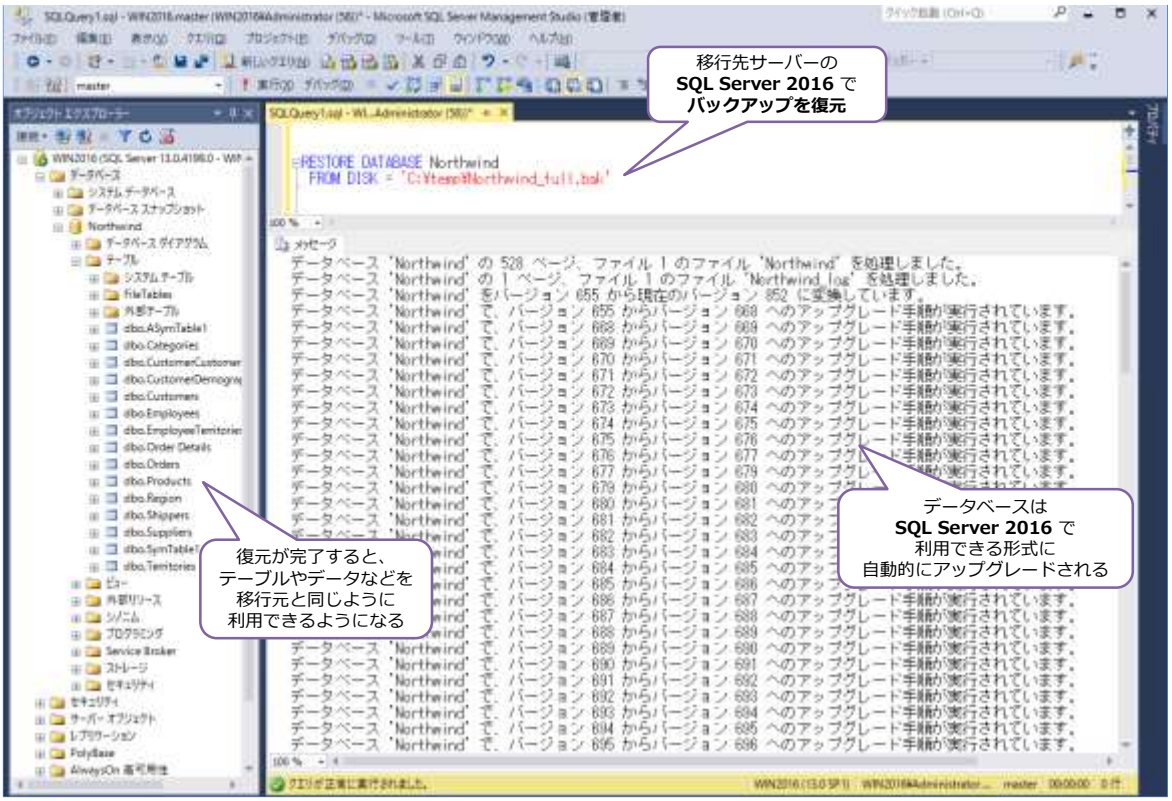
インプレース アップグレードは SQL Server 2008 以上が対象でしたが、移行（マイグレーション）であれば、**SQL Server 2005** のデータベースを SQL Server 2016 に移行することもできます。

**データベースの移行**は、**標準のバックアップと復元**機能を利用することで、簡単に行うことができます。移行元の SQL Server 上でデータベースを**完全バックアップ**して、それを **SQL Server 2016** 上で**復元**（リストア）するだけで、データベースの移行が完了です（2 つのステートメントを実行するだけで完了します）。

例えば、**SQL Server 2005** 上でデータベースの**完全バックアップ**を実行する場合は、次のように **BACKUP DATABASE** ステートメントを実行します。




このバックアップ ファイル（**.bak**）を、移行先となる **SQL Server 2016** 上で**復元**すれば、データベースの移行が完了です。復元は、次のように **RESTORE DATABASE** ステートメントを実行するだけです。




このように、**SQL Server 2005** 上で取得したバックアップは、**SQL Server 2016** 上にも簡単に復元することができます。これだけの操作で、**テーブル内のデータ**が復元されることはもちろん、インデックスやビュー、ストアド プロシージャ、制約、トリガー、ユーザー定義関数、SQL CLR オブジェクト、フルテキスト インデックス、暗号化のための対称キー／非対称キー、データ パーティション、データベースの設定オプションなど、データベース内のすべてのオブジェクトを復元できます。

**SQL Server 2005** のデータベースを復元した場合には、**データベースの互換性レベル**が **100**（SQL Server 2008 レベル）に自動的に上がったり、データベース ユーザーとオブジェクト権限に関しては、元の環境と同じように利用するためには追加手順が必要になったりしますが、ほとんどのオブジェクトを移行元と同じように利用することができます（追加手順が必要なオブジェクトについては、第 5 章で詳しく説明しています）。

このように SQL Server は、データベースの移行を非常に簡単に行うことができ、SQL Server 2005 や 2008、2008 R2、2012、2014 で利用していたデータベースを、SQL Server 2016 上に簡単に復元／移行することができます。

バックアップと復元機能を利用すれば、移行元が 32 ビットで、移行先が 64 ビットでも、移行元と移行先の OS が違っても、移行先がクラウド（Mirosoft Azure 上の仮想マシン）であったとしても、関係なくデータベースの移行を行うことができます。

既存の SQL Server が 32 ビットの場合や、SQL Server 2005 を利用している場合には、インプレース アップグレードがサポートされていないので、バックアップと復元機能を利用して、移行をするようにします。

## SQL Server 2016 をインストール可能な OS

**SQL Server 2016** は、以下の **OS** にインストールすることができます（アップグレード時も条件は同じです）。

- **Windows Server 2012（X64）**
- **Windows Server 2012 R2（X64）＋ KB 2919355**
- **Windows Server 2016（X64）**

＊ Developer エディションの場合は、Windows 8、Windows 8.1、Windows 10 にインストールすることも可能

以前の SQL Server との大きな違いは、**SQL Server 2016** が **X64**（64 ビット版）の OS のみでサポートされるようになったことと、**Windows Server 2012** 以降の OS が必要になることです。SQL Server 2014 のときは Windows Server 2008 SP2 以降の OS がサポートされていましたが、SQL Server 2016 から変更になりました。

したがって、SQL Server 2008 や 2008 R2、2012、2014 を、**Windows Server 2003** や **2003 R2**、**2008**、**2008 R2** などで動作させている場合には、OS を **Windows Server 2012** 以降にアップグレードしてから、SQL Server をアップグレードする必要があります（詳しくは、第３章で説明します）。

また、OS を Windows Server 2012 以上にアップグレードする場合は、SQL Server の Service Pack 要件が次のようにあります。

**SQL Server を動作させるために必要となる Service Pack**

| | Windows Server 2012 | Windows Server 2012 R2 | Windows Server 2016 |
|---|---|---|---|
| SQL Server 2008 | SP3 | SP3 | 未サポート |
| SQL Server 2008 R2 | SP1 | SP2 | 未サポート |
| SQL Server 2012 | RTM | SP1 | SP2 |
| SQL Server 2014 | RTM | RTM | SP1 |

**SQL Server 2016 にアップグレードするための Service Pack 要件（前掲）**

| | |
|---|---|
| SQL Server 2008 | SP4 |
| SQL Server 2008 R2 | SP3 |
| SQL Server 2012 | SP2 |
| SQL Server 2014 | RTM |

SQL Server 2008 を利用している場合は、Windows Server 2012 にアップグレードする前に、SP3 以上、SQL Server 2016 にアップグレードする前に SP4 以上を適用しておく必要があります。また、Windows Server 2016 からは、SQL Server 2008 および 2008 R2 が未サポートになりました。これについては、以下の KB 2681562 が参考になると思います。

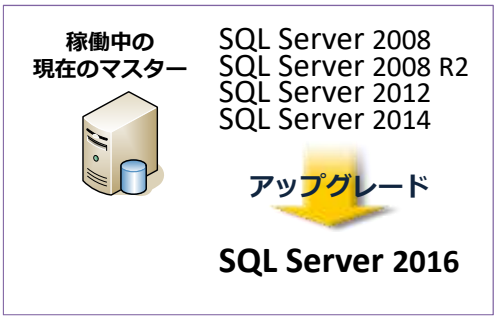
Using SQL Server in Windows 8 and later versions of Windows operating system
http://support.microsoft.com/en-us/kb/2681562

## 1.4 アップグレードと移行の主なケース

アップグレードと移行は、次のように 4 つのケースに分類できます。

- **ケース１　同一マシンでのアップグレード（インプレース アップグレード）**
  単純なアップグレード（以前の SQL Server を上書き）
- **ケース２　新規サーバー（別マシン）へのアップグレード**
  ケース１との違いは別マシンかどうか（ハードウェア リプレースによって新規サーバーを購入した場合のアップグレード）
- **ケース３　新規サーバー（別マシン）への移行（マイグレーション）**
  アップグレードではなく、別マシンへの移行
- **ケース４　同一マシンでの移行（別インスタンスをサイド バイ サイドで実行）**
  ケース２との違いは別マシンかどうか（ケース４ は同じマシン内）

### ケース１　同一マシンでのアップグレード（インプレース アップグレード）




これは、**単純なアップグレード**（インプレース アップグレード）で、以前のバージョンの SQL Server を SQL Server 2016 に完全に置き換える（上書きする）ものです。

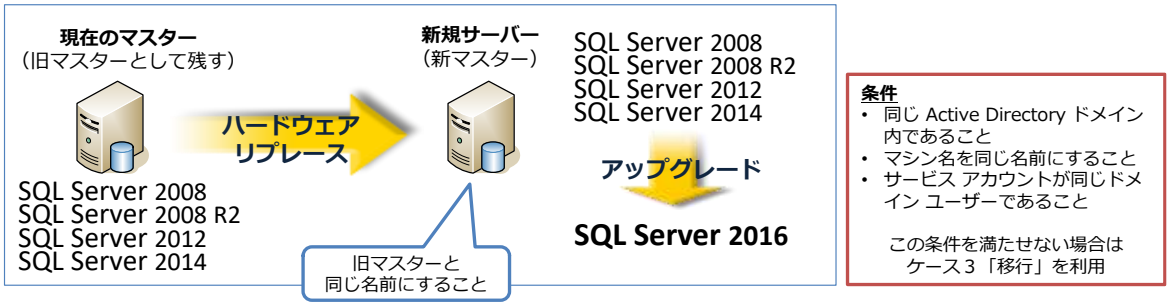
この方法のメリットは、以前のバージョンで利用していた機能をほとんどすべてそのまま利用できることです。データベースを以前と変わらずに利用できることはもちろん、ログイン アカウントや、ジョブ、警告、暗号化、リンク サーバー、メンテナンス プラン（保守計画）、リソース ガバナー、SQL Server Audit（監査）、データベース メール、サーバーの構成オプション、Reporting Services、Analysis Services など、以前の SQL Server で設定／利用していた機能を、そのまま SQL Server 2016 上でも利用することができます。

また、レプリケーションやログ配布、AlwaysOn 可用性グループ、データベース ミラーリングといったサーバー間の連携機能や、WSFC（Windows Server フェールオーバー クラスタリング）上の SQL Server インスタンスに関しても、アップグレードをすることができます。ログ配布や AlwaysOn 可用性グループ、データベース ミラーリングに関しては、セカンダリを先にアップグレードすることで、ローリング アップグレードも可能です（ローリング アップグレードによって、セカンダリをアップグレード中でも、プライマリを利用することができるので、ダウンタイムを最小限に抑えることができます）。

一方で、この方法のデメリットは、アップグレード後に、旧システム環境が利用できなくなってしまう(以前のバージョンの SQL Server が完全に利用できなくなってしまう)ことです。これだと、万が一アップグレードに失敗してしまった場合には、元の環境に戻すのが大変になり、失敗時は、(最悪は) OS のインストールからやり直して、バックアップからすべてを復元しなければならない場合があります。

したがって、このケースを利用する場合は、万が一のアップグレード失敗時に備えて、元の環境でしっかりとバックアップを取得しておくこと、元へ戻す手順(ロールバック手順)をしっかりと計画しておくことが重要になります。また、次の「**ケース 2**」のように、ハードウェア リプレースを伴うアップグレードの場合であれば、元の環境をそのまま残しておくことができるので、万が一の失敗時にもすぐに元の環境に戻せるようになります。

## ケース 2　新規サーバー(別のマシン)へのアップグレード




ケース 1 は、**同一マシン**でのアップグレードでしたが、このケースは**別マシン**を利用したアップグレードです。ハードウェアの保守切れや老朽化などによって、**新規サーバーの購入**(**ハードウェア リプレース**)を行って、そのマシンへ旧システム環境を**丸ごと複製**して、それを **SQL Server 2016** へアップグレードする方法です。

このケースは、弊社のお客様に多く、**現在のマスター**と**同じ名前の新規サーバー**を、**同じ Active Directory ドメイン内**に構築したい場合には、この方法が一番簡単で確実になります。ただし、この方法は、あくまでも同じ名前の新規サーバーを同じ Active Directory ドメイン内に作成できること、サービス アカウントが同じドメイン ユーザーであることが条件になるので、異なるドメインやワークグループ環境、違う名前の新規サーバーを構築する場合には、次の「**ケース 3　別マシンへの移行**」(マイグレーション)を利用する必要があります。

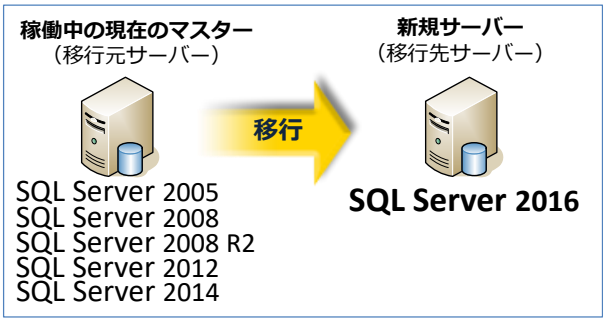
ケース 1 では、同一マシンでのアップグレードによって、元の環境を残せないのがデメリットであると説明しましたが、このケースでは、旧システム環境をそのまま残すことができるのがメリットです。具体的な手順は後述しますが、旧システム環境が SQL Server 2008 の場合は、SQL Server 2008 が全く同じように動作するように、**丸ごと複製**します(システム データベースを移動するなどして、各種の設定を**完全複製**し、**マシン名も同じ**にして、**同じ Active Directory ドメイン内**に参加するようにします)。丸ごとの複製が完了したら、旧システム環境をネットワークから切り離して、新規サーバー(新マスター)を **SQL Server 2016** へ**アップグレード**します。これによって、万が一アップグレードの失敗があったとしても、旧システム環境が残っているので、すぐに元の環

境に戻せるようになります。

過去の弊社のお客様では、この方法（ケース２）を利用してアップグレードを行ったときに、ハードウェア リプレースだけでなく、データセンターの変更（別のデータセンター内のハードウェアに変更）も行ったのですが、当日の作業で、SQL Server へのアップグレードは完了したものの、（データセンター側の）ネットワーク機器の設定ミスによって、ネットワークの移行が失敗してしまうということがありました。このままでは予定停止時間をオーバーしてしまって、翌朝のサービス インには間に合わなくなってしまうことから、この日は残っている旧システム環境を利用して元に戻して、アップグレードを断念したということがありました（後日、再度作業を行って、今度は無事にアップグレードが完了しました）。

このように、万が一に備えて、旧システム環境がそのまま残っていることは、大変便利です。もし、旧システム環境が残っておらず、OS をイチからインストールし直さなければならなかったとすると、とても停止時間内には元に戻すことはできませんでした。

## ケース３　新規サーバー（別マシン）への移行（マイグレーション）




このケースは、**新規サーバー**（**ハードウェア リプレース**など）を利用して、そのマシンへ **SQL Server 2016** をインストールし、データベースや各種の設定を**移行**（マイグレーション）する方法です。弊社のお客様で一番多いのがこのケースです。ハードウェア リプレースを機会に、32 ビットから 64 ビット環境へ移行したり、SSD やフラッシュ ストレージなどの高速ストレージ環境へ移行したりするというお客様も多くいらっしゃいます。

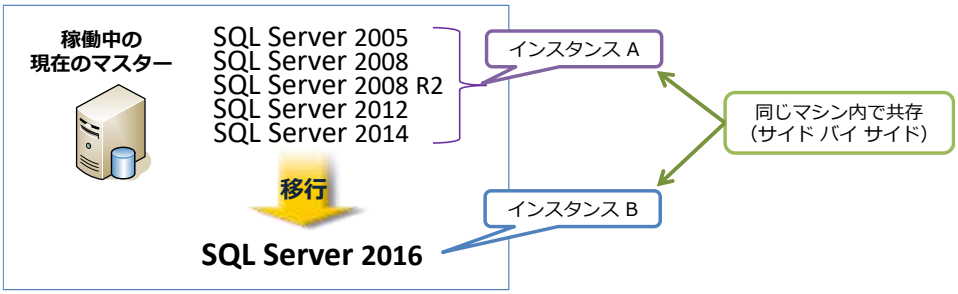
こうしたマシンをまたがった場合でも、SQL Server では簡単に移行を行うことができます。データベースの移行は、**標準のバックアップと復元**機能で行うことができ、バックアップと復元機能を利用すれば、別マシンであっても簡単にデータベースを複製することができ、32 ビットから 64 ビットへ変更したとしても、移行先の OS が変わったとしても、オンプレミスからクラウドに変わったとしても、簡単に移行することができます。SQL Server 2005 上で取得したデータベースのバックアップを、SQL Server 2016 上に復元することもできます。

各種の設定（ログイン アカウントやジョブ、警告、リンク サーバーなどの設定）に関しても、**スクリプト生成**機能を利用することで、簡単に移行することができます。

ケース１ やケース２ との大きな違いとしては、サーバー管理に関する機能（リソース ガバナーや

SQL Server Audit、パフォーマンス データ コレクションなど）や、サーバー間の連携機能（ログ配布やレプリケーション、AlwaysOn 可用性グループ、データベース ミラーリングなど）に関しては、再設定をする必要がある点です。

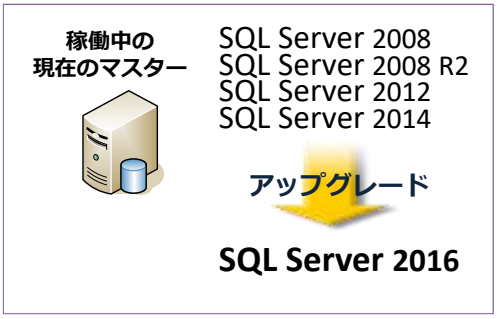
## ケース４　同一マシンでの移行（別インスタンスをサイド バイ サイドで実行）




このケースでは、**同じマシン内**に、**新しいインスタンス**として、**SQL Server 2016** をインストールし、旧システム環境（旧バージョンの SQL Server インスタンス）からデータベースや各種の設定を**移行**（マイグレーション）します。ケース３ との違いは、同じマシンかどうかだけになります。

以降では、これらのケース 1～3 での具体的な手順を説明します（ケース４ に関しては、ケース３ とほとんど同じなので省略します）。

## 1.5 ケース１「同一マシンでのアップグレード」の手順概要




ケース１ でのアップグレード手順は、第３章で詳しく説明しますが、概要は次のとおりです。

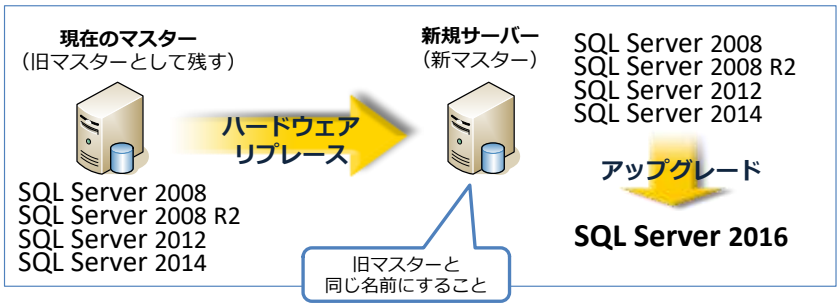
1. **Data Migration Assistant** による事前チェックを行う
2. **OS** に **Windows Server 2003** や **2003 R2**、**2008**、**2008 R2** を利用している場合は、**Windows Server 2012** 以上に**アップグレード**する

   OS のアップグレードにあたっては、SQL Server を動作させるための Service Pack 要件を確認する（例えば、Windows Server 2012 にアップグレードする場合に、SQL Server 2008 なら SP3、SQL Server 2008 R2 なら SP1 が必要など）
3. **SQL Server 2016** への**アップグレード要件**を確認する（アップグレード可能な **Service Pack** を確認。例えば、SQL Server 2008 なら SP4、SQL Server 2008 R2 なら SP3 が必要など）
4. **SQL Server 2016** への**アップグレード インストール**を行う
5. **SQL Server 2016** の**最新の修正プログラム**（CU や Service Pack）をインストールする
6. **Management Studio**（管理ツール）の最新版をダウンロードして、インストールする（オプション）
7. **統計**（Statistics）を更新する
8. **データベースの互換性レベルを 130** へ上げる（オプション）

   互換性レベルの影響については**クエリ ストア**機能を利用することで簡単にチェック可能（実行プランの比較や、プラン強制もできる）。
9. **BIDS**（Business Intelligence Development Studio）や **SSDT-BI**（SQL Server Data Tools - Business Intelligence）を利用している場合は、**SSDT** の最新版をダウンロードして、インストールする（オプション）

前述したように、この方法のデメリットは、アップグレード後に、旧システム環境が利用できなくなってしまう（以前のバージョンの SQL Server が完全に利用できなくなってしまう）ことです。したがって、万が一の失敗時に備えて元の環境へ戻す手順（ロールバック手順）をしっかりと計画しておくことが重要になります。

## 1.6 ケース２「新規サーバーへのアップグレード」の手順概要




ケース２ でのアップグレード手順は、第４章で詳しく説明しますが、概要は次のとおりです。

1. **現在のマスター環境**を**丸ごと新規サーバーへ複製**する

   - 現在のマスターで**オフライン バックアップ**（全データベース）を取得する
     （ユーザー データベースに関しては、オンライン バックアップでも可）
   - 現在のマスターを停止して、**ネットワークから切り離す**（旧マスターとなる）
   - **新規サーバー**に **OS** をインストールし、マシン名を**旧マスターと同じ名前**へ変更する
     （インストールする **OS** は、旧マスターと異なるものでも可）
   - **新規サーバー**を **Active Directory ドメインへ参加**させる
   - 新規サーバーへ旧マスターと**同じバージョンの SQL Server** をインストールする
   - 旧マスターの SQL Server にインストール済みの**修正プログラム**を、新規サーバーにもインストールする
   - 新規サーバーの **SQL Server を停止**する
   - 旧マスターで取得した**オフライン バックアップを上書きコピー**する（復元する）
     ユーザー データベースをオンライン バックアップで取得している場合は、SQL Server の起動後に、該当データベースを復元する
   - 新規サーバーの **SQL Server を起動**する
   - **レジストリ**に格納されている情報を再設定する（TCP ポート番号や起動時パラメーターでのトレースフラグ設定などのうち、旧マスターで設定を変更しているものがある場合はそれらを再設定する）
   - **OS の設定**で、旧マスターで変更しているものがある場合は、それらを再設定する
     （フォルダー構成や、NTFS アクセス許可、ユーザーの権利、共有フォルダーなど）

   以上で旧マスターとまったく同じ環境を丸ごと新規サーバー上で動作させることができます。

2. **Data Migration Assistant** による事前チェックを行う

3. **SQL Server 2016** への**アップグレード要件**を確認する（アップグレード可能な Service Pack を確認／インストールする）

   SQL Server 2008 なら SP4、SQL Server 2008 R2 なら SP3、SQL Server 2012 なら SP2 が必要。OS に Windows Server 2003 や 2003 R2、2008、2008 R2 を利用している場合は、Windows Server 2012 以上にアップグレードする。

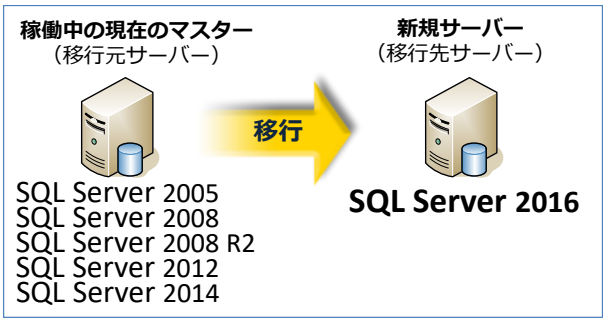
4. **新規サーバー**を **SQL Server 2016 へアップグレード**する
5. **SQL Server 2016** の**最新の修正プログラム**（CU や Service Pack など）をインストールする
6. **Management Studio**（管理ツール）の最新版をダウンロードして、インストールする（オプション）
7. **統計**（Statistics）を更新する
8. **データベースの互換性レベルを 130** へ上げる（オプション）

   互換性レベルの影響については**クエリ ストア**機能を利用することで簡単にチェック可能（実行プランの比較や、プラン強制もできる）。
9. **BIDS**（Business Intelligence Development Studio）や **SSDT-BI**（SQL Server Data Tools - Business Intelligence）を利用している場合は、**SSDT** の最新版をダウンロードして、インストールする（オプション）

このケースでは、万が一アップグレードの失敗があったとしても、旧システム環境が残っているので、すぐに元の環境に戻すことができます。**現在のマスター**と**同じ名前の新規サーバー**を構築したい場合には、この方法が一番簡単で確実です。

このケースでは、後述のケース3 とは違ってシステム データベース関連のオブジェクトをそのまま動作させられることもメリットです。

## 1.7 ケース３「新規サーバーへの移行」（マイグレーション）の手順概要




ケース３ での移行（マイグレーション）手順は、第５章で詳しく説明しますが、手順の概要は次のとおりです。

1. **Data Migration Assistant** による事前チェックを行う
2. **新規サーバー**へ **Windows Server 2012**（X64 版）以上の OS をインストールする（移行元と同じ OS である必要はありません）
3. **Active Directory ドメイン**環境の場合は、**新規サーバー**をドメインに参加させる
4. **SQL Server 2016** を**インストールするためのソフトウェア要件**を確認する

   OS は、Windows Server 2012 以降の X64 版をサポート、Windows Server 2012 R2 の場合は KB 2919355 が必要など。データベース メール機能を利用している場合は、.NET Framework 3.5 SP1 をインストールしておく必要がある
5. **新規サーバー**へ **SQL Server 2016** をインストールする
6. **新規サーバー**へ **SQL Server 2016** の**最新の修正プログラム**（CU や Service Pack など）をインストールする
7. **Management Studio**（管理ツール）の最新版をダウンロードして、インストールする（オプション）
8. **移行元サーバー**（SQL Server 2005／2008／2008 R2／2012／2014）の**データベース**を、**新規サーバー**（SQL Server 2016）へ移行する（バックアップと復元機能を利用）
9. **統計**（Statistics）を更新する
10. **フルテキスト インデックス**を再構築する（フルテキスト検索機能を利用している場合）
11. **データベースの互換性レベルを 130** へ上げる（オプション）

    互換性レベルの影響については**クエリ ストア**機能を利用することで簡単にチェック可能（実行プランの比較や、プラン強制もできる）。
12. **データベースの所有者**を確認／設定する（∵所有者が空の場合には、**データベース ダイアグラム**と後述の **SQL CLR オブジェクト**が動作しないため）

13. **SQL CLR オブジェクト**の権限セットで「**UNSAFE**」または「**外部**」を利用している場合は、**TRUSTWORTHY** オプションを有効化する
14. **システム データベース**関連のオブジェクトを移行する（ログイン アカウントやサーバー ロール、tempdb の設定、リンク サーバー、ユーザー定義エラー、構成オプション、データベース メール、ジョブ、警告、オペレーターなどのうち、移行元で設定を変更／利用しているものがある場合は、それらを移行する）
15. **レジストリ**に格納されている情報を再設定する（サービスの自動起動やサービス アカウント、認証モード、TCP ポート番号、起動時パラメーターでのトレースフラグの設定などのうち、移行元で設定を変更しているものがある場合は、それらを再設定する）
16. **OS の設定**で、移行元で変更しているものがある場合は、それらを再設定する（フォルダー構成や、NTFS アクセス許可、ユーザーの権利、共有フォルダーなど）
17. **メンテナンス プラン**（保守計画）を利用している場合は、メンテナンス プランを再作成する
18. **BIDS** または **SSDT-BI** を利用している場合は、**SSDT** の最新版をダウンロードして、インストールする（オプション）
19. **Integration Services** を利用している場合は、SSIS パッケージを移行する
20. **SQL Server Audit** や**リソース ガバナー**、**パフォーマンス データ コレクション**などのサーバー管理機能を利用している場合は、これらを再設定する
21. **レプリケーション**や**ログ配布**、**可用性グループ**、**データベース ミラーリング**などのサーバー間の連携機能を利用している場合は、これらを再設定する

前述したように、新規サーバーへの**データベースの移行**は、**標準のバックアップと復元**機能を利用して、簡単に行うことができます。データベース ダイアグラムや SQL CLR オブジェクトを利用している場合は、追加の作業が必要になりますが、これらはコマンドを 1 つ 2 つ実行するだけなので簡単な作業です。

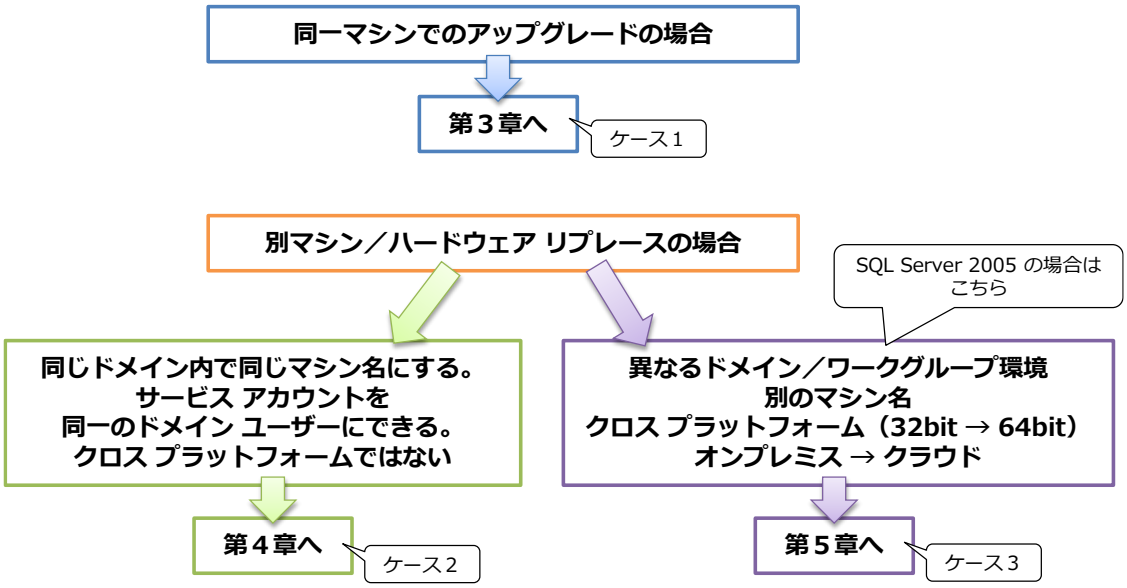
**各種の設定**（**システム データベース**や**レジストリ**に格納されている設定、**メンテナンス プラン**など）は、移行元で設定を変更していたり、該当オブジェクトを利用したりしている場合には、再設定／再作成が必要になります。**システム データベース**に格納されているものに関しては、ほとんどのものが GUI 操作で**スクリプト生成**することができるので、簡単に移行することができます。**レジストリ**に格納されているものに関しては、SQL Server はレジストリをほとんど利用していないので、こちらも再設定は簡単です。

この移行方法を利用するメリットは、**OS** を簡単に変更できること（**Windows Server 2003／2003 R2** や **Windows Server 2008／2008 R2** から **Windows Server 2012** や **2012 R2**、**Windows Server 2016** へ変更するなど）、クロス プラットフォーム（移行元が **32 ビット**で、移行先が **64 ビット**など）でも関係がないこと、**Microsoft Azure** などの**クラウド環境**であったとしてもデータベースの移行が行えること、**SQL Server 2005 からの移行**にも対応していることです。

## 1.8 第 2 章以降の内容について

第 2 章以降では、ケース 1～3 の具体的な手順を説明します。

- 第 2 章　Data Migration Assistant の利用方法
- 第 3 章　ケース 1　同一マシンでのアップグレード
- 第 4 章　ケース 2　新規サーバー（別マシン）へのアップグレード
- 第 5 章　ケース 3　新規サーバー（別マシン）への移行




なお、Reporting Services や Analysis Services を移行／アップグレードする手順については、各章（3～5 章）の後半で説明しています。

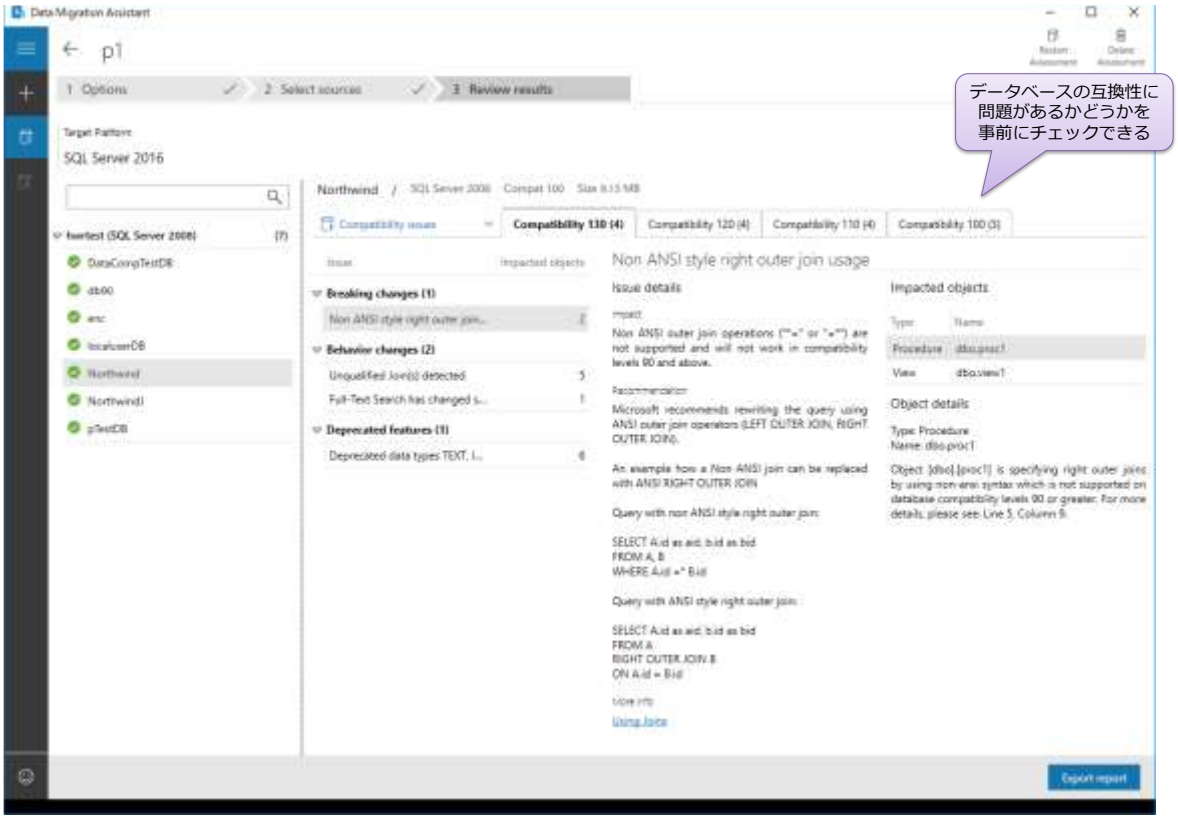
# STEP 2. Data Migration Assistant による事前チェック

この章では、**Data Migration Assistant**（旧アップグレード アドバイザー）を利用したデータベースの事前チェック（アップグレード時の互換性問題の事前チェック）を行う方法について説明します。

- ✓ Data Migration Assistant とは
- ✓ Data Migration Assistant のダウンロード／インストール
- ✓ Data Migration Assistant の利用方法

## 2.1 Data Migration Assistant の概要

**Data Migration Assistant** は、**SQL Server 2008** や **2008 R2／2012／2014** を **SQL Server 2016** へ移行／アップグレードした際に発生する問題を事前にチェックすることができるツールです（SQL Server 2014 までは**アップグレード アドバイザー**として提供されていました）。

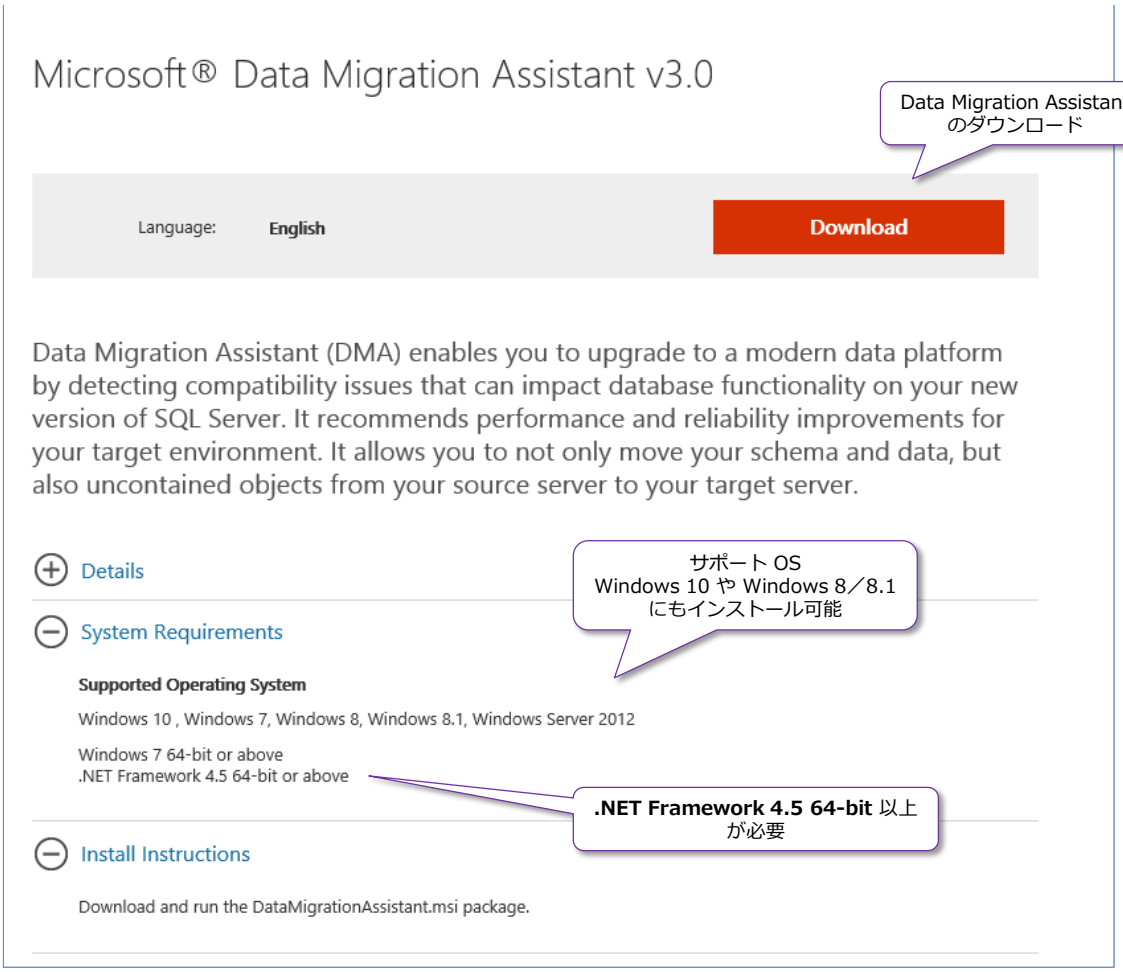



Data Migration Assistant は、アップグレード前の SQL Server に対して実行して、データベースをアップグレードしたときの問題点（互換性に関する問題）を簡単に見つけることができます。このツールは、クライアント OS（Windows 10 や Windows 8／8.1）にインストールして実行することもできるので、気軽にデータベースの互換性をチェックすることができます。

## 2.2 Data Migration Assistant のダウンロード／インストール

**Data Migration Assistant** は、次の URL からダウンロードすることができます。

Microsoft Data Migration Assistant v3.0（執筆時点では英語版のみ）
http://www.microsoft.com/en-us/download/details.aspx?id=53595



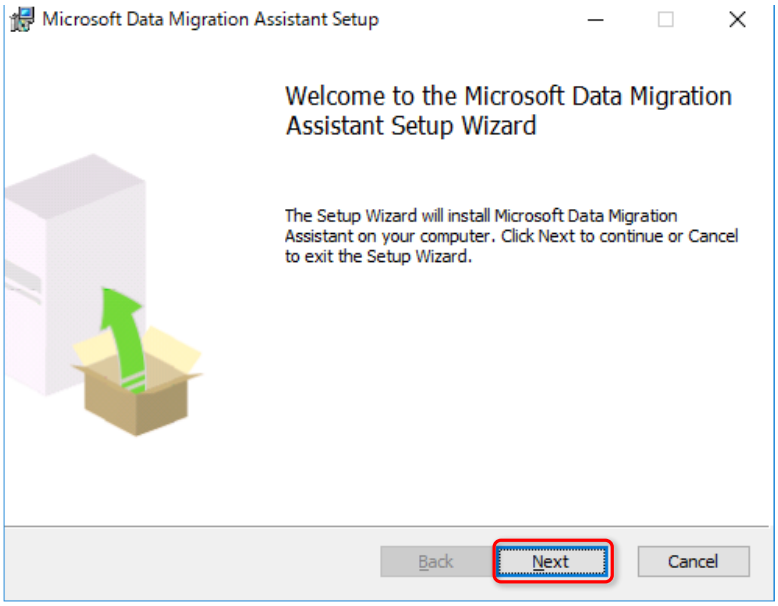

このツールは、Windows 10 や Windows 8／8.1 などのクライアント OS にもインストールすることができます。

インストールに必要なコンポーネントは、**.NET Framework 4.5 64-bit** 以上になります。

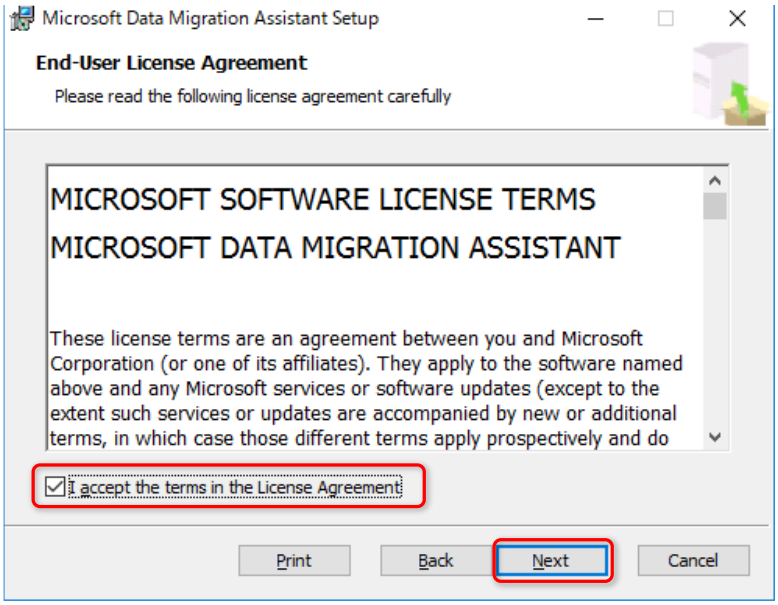
### Data Migration Assistant のインストール

**Data Migration Assistant** のインストール方法は、次のとおりです。

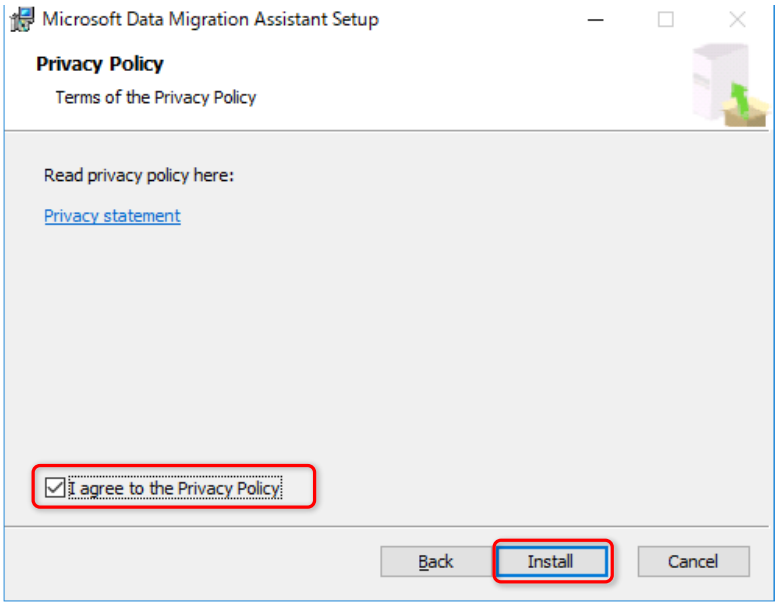
まずは、ダウンロードしたインストーラー（**DataMigrationAssistant.msi**）を実行して、次のようにインストール ウィザードが開始されたら、［**Next**］ボタンをクリックします。



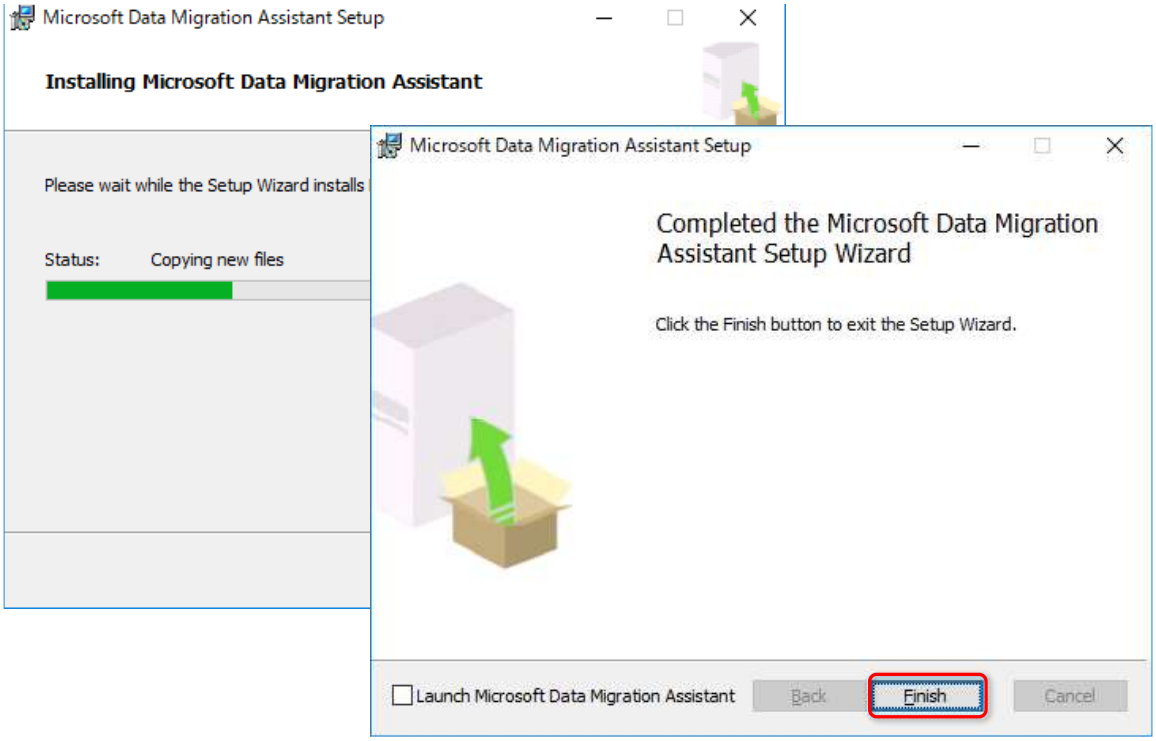

次の［**End-User License Agreement**］（**使用許諾契約書**）ページでは、ライセンス条項の内容を確認した上で、［**I accept the terms in the License Agreement**］をチェックして、［**Next**］ボタンをクリックします。




次の［**Privacy Policy**］ページでは、プライバシー ポリシーの内容を確認した上で、［**I agree to the Privacy Policy**］をチェックして、［**Install**］ボタンをクリックします。




これでインストールが始まって、インストール中は次のように進行状況が表示されます。



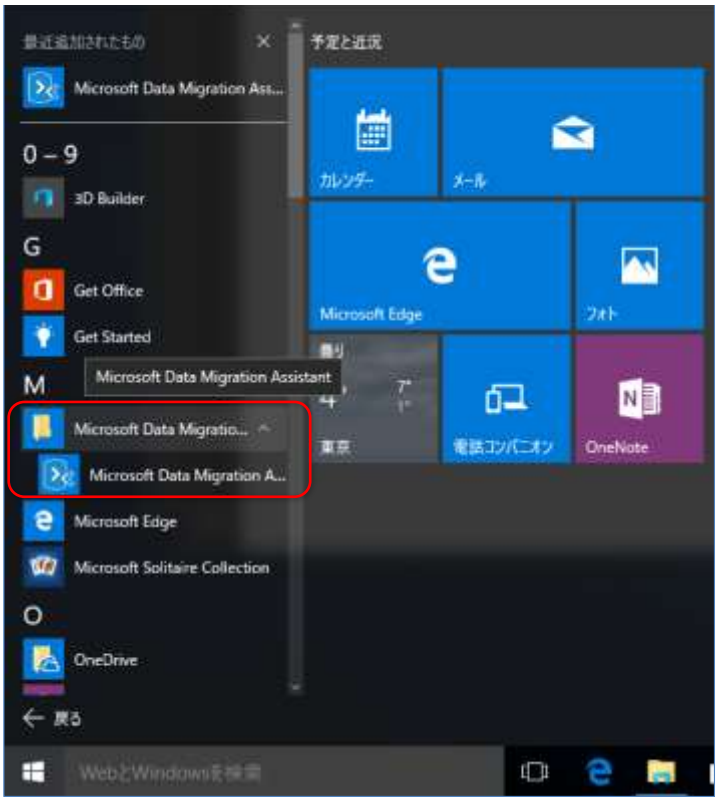

インストールが正常に完了したら、[**Finish**] ボタンをクリックして、ウィザードを終了します。

以上で Data Migration Assistant のインストールが完了です。

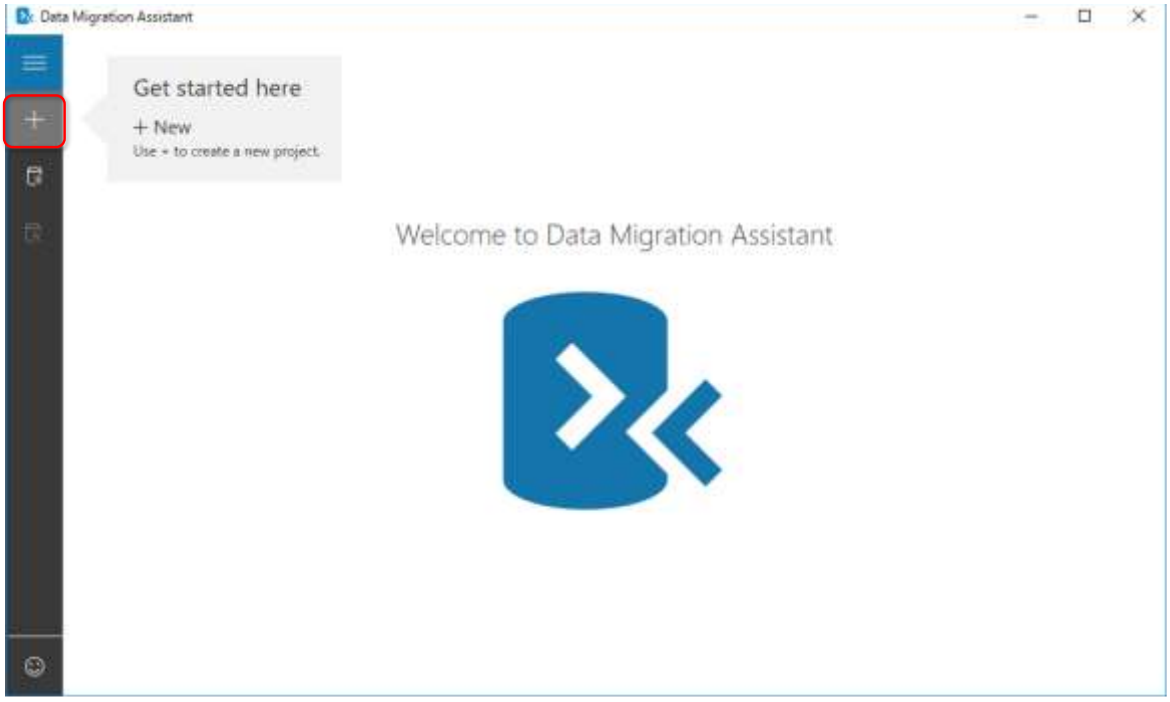
## 2.3 Data Migration Assistant の利用手順

**Data Migration Assistant** を利用する手順は、次のとおりです。

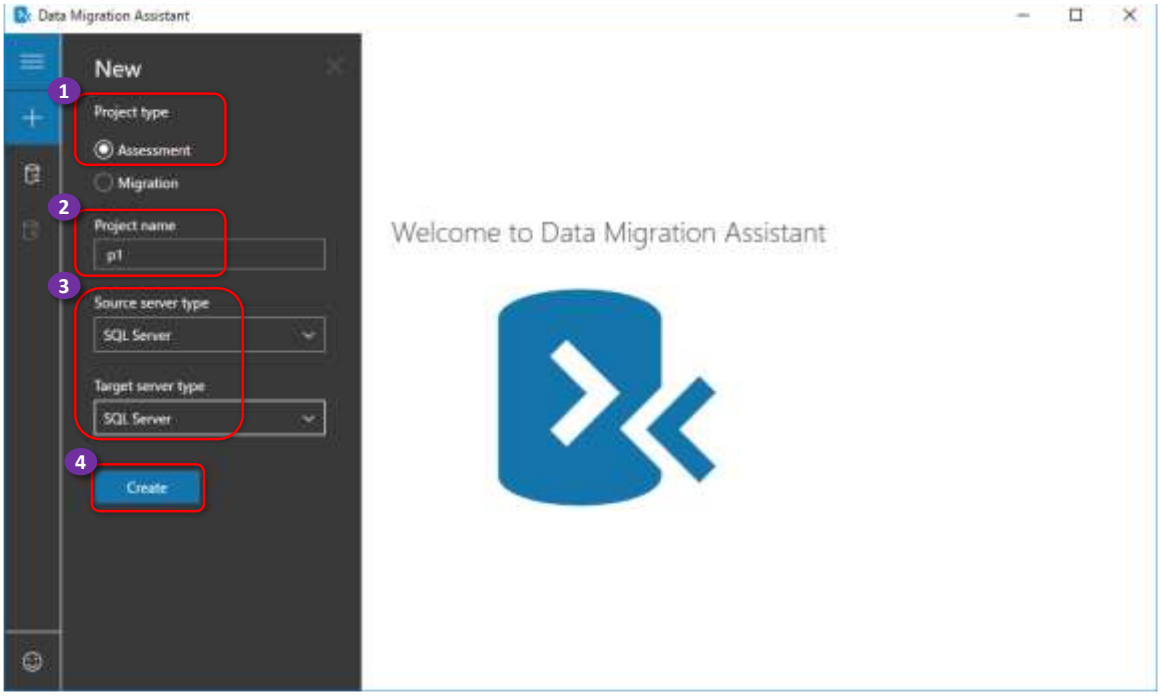
**Data Migration Assistant** を起動するには、スタート メニューで［**Microsoft Data Migration Assistant**］グループから「**Microsoft Data Migration Assistant**」をクリックします。




Data Migration Assistant が起動したら、画面左上の［**+**］（New）ボタンをクリックします。

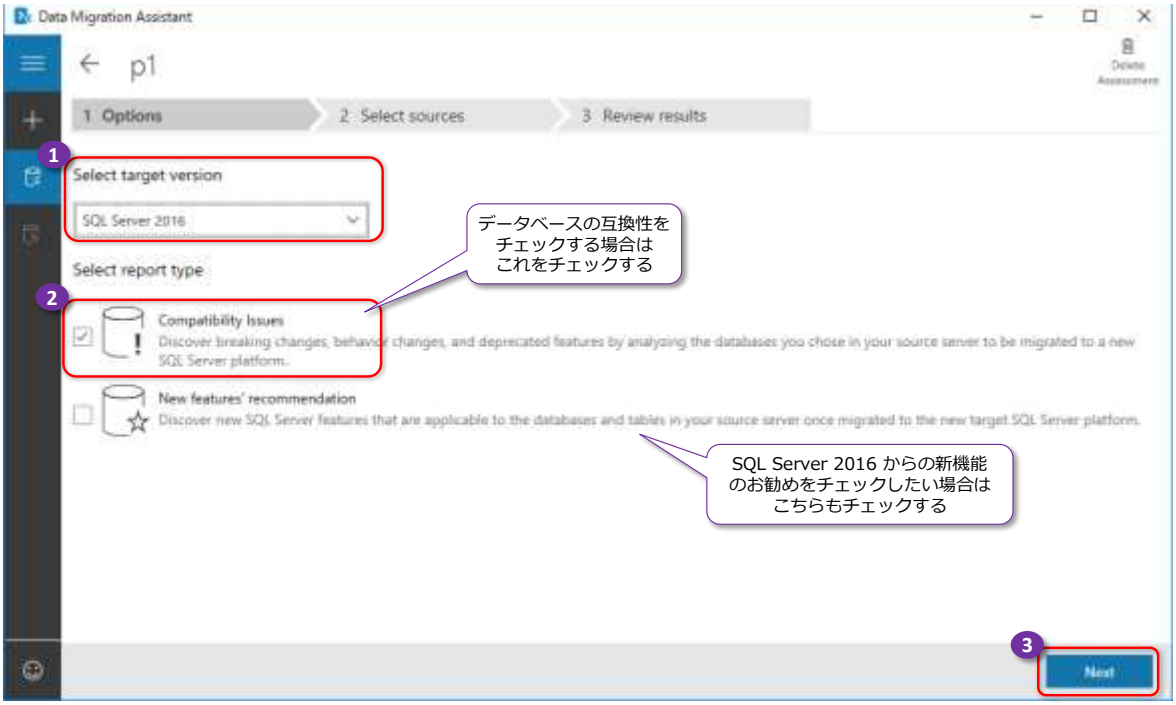


［**New**］ブレードが表示されたら、［**Project type**］で「**Assesment**」（アセスメント）、［**Project name**］に任意のプロジェクト名（画面は **p1**）を入力します。




［**Source server type**］と［**Target server type**］には、どちらも「**SQL Server**」を選択して、［**Create**］をクリックします。これで、データベースの互換性をチェックできるプロジェクトを作成することができます。

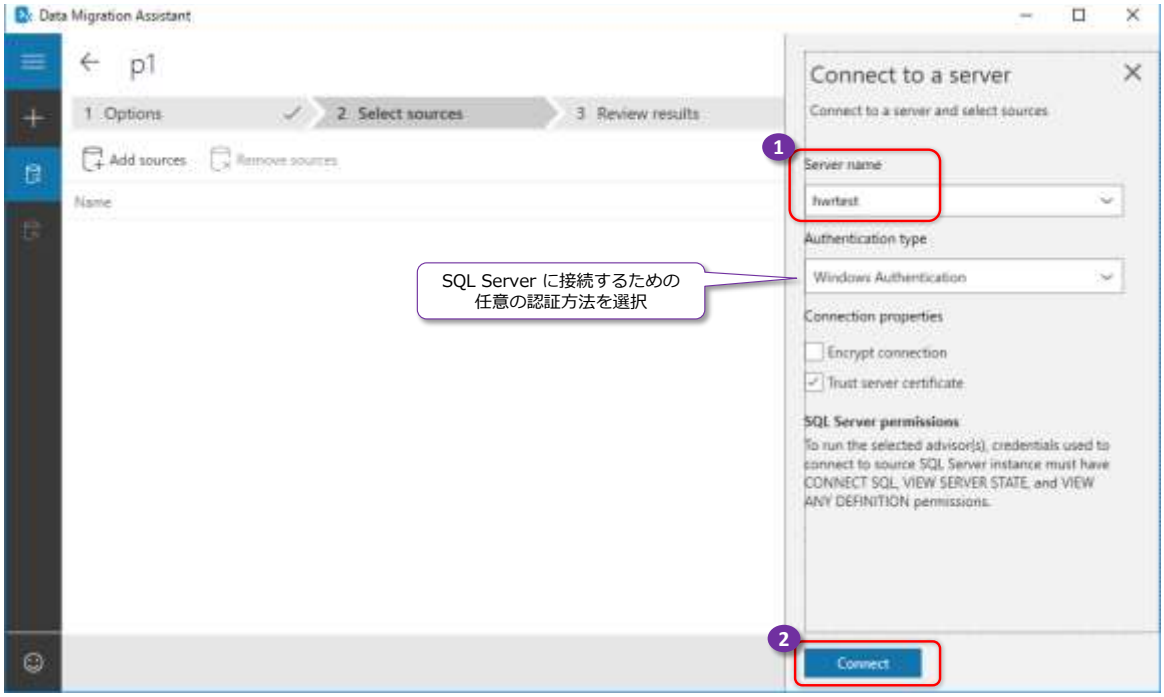
次に［**Options**］ページが表示されるので、［**Select target version**］で「**SQL Server 2016**」を選択します。



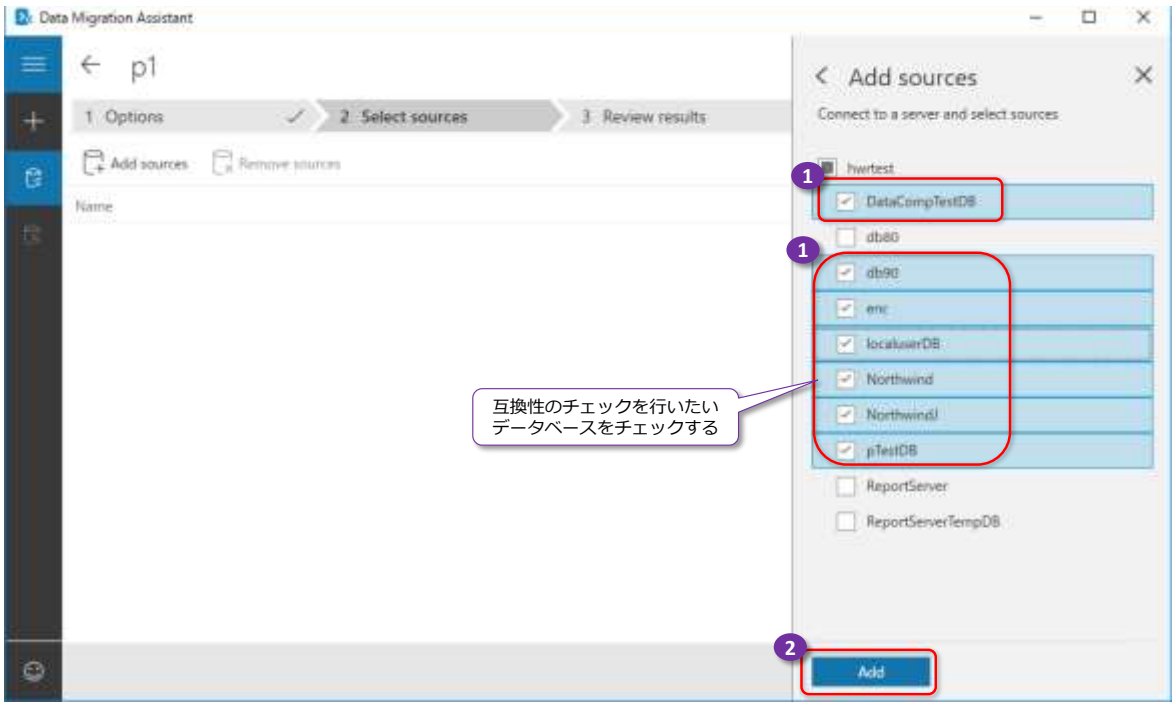

［**Select target type**］では、「**Compatibility issues**」（互換性に関する問題）を選択して、［**Next**］

ボタンをクリックします。

［**Connect to a server**］ページが表示されたら、［**Server name**］で接続先となる SQL Server（アップグレード前の SQL Server）の名前を入力（画面は、SQL Server 2008 を動作させている **hwrtest** という名前のマシンを指定）して、［**Connect**］ボタンをクリックします。

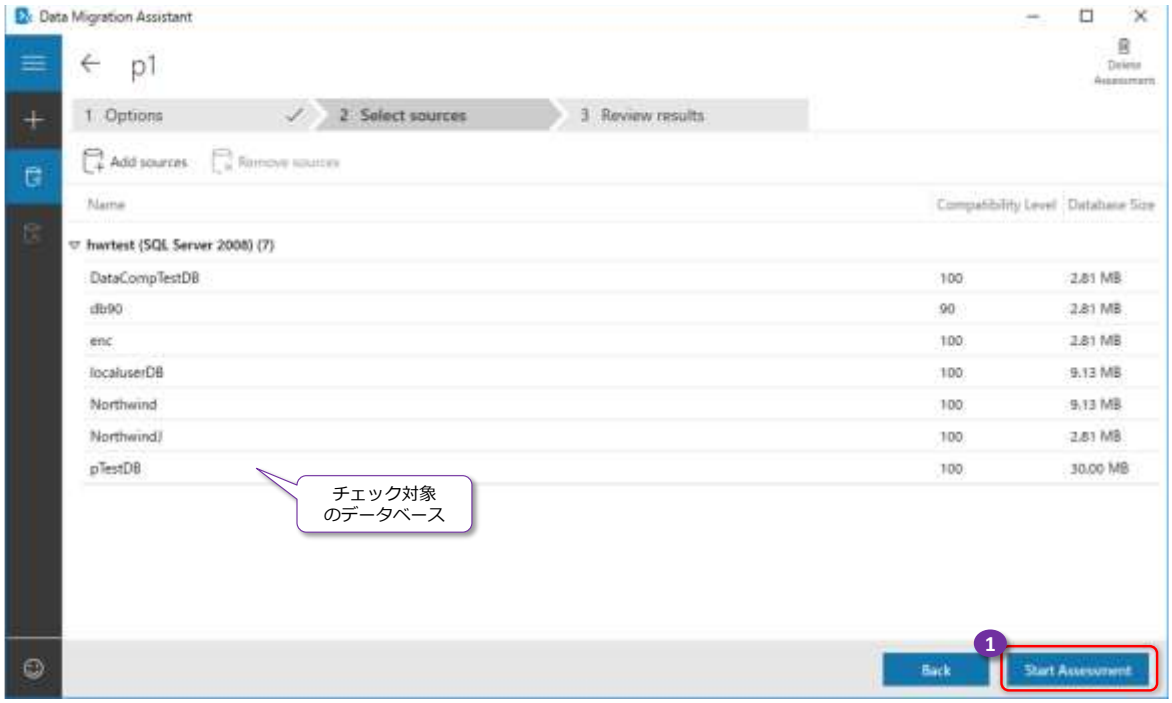



次に［**Add sources**］ページが表示されると、接続した SQL Server 上のデータベースが一覧されるので、互換性のチェックを行いたいデータベースをチェックして、［**Add**］ボタンをクリックします。

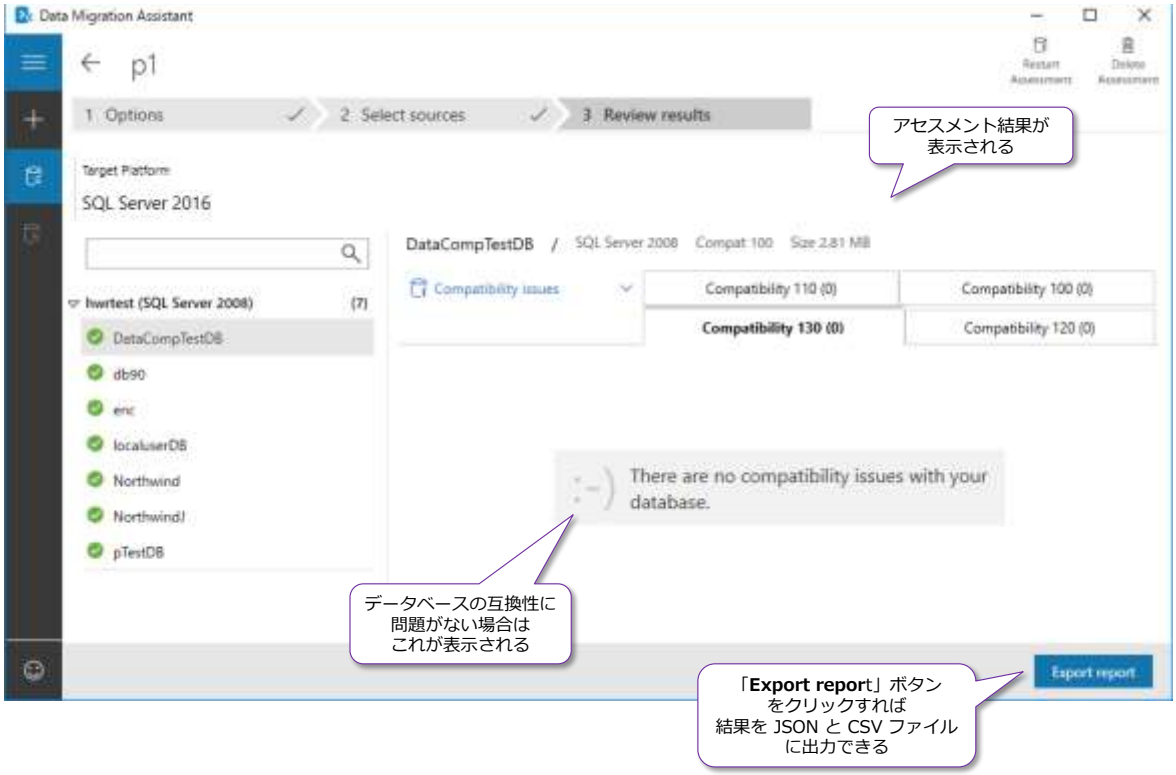



次に、選択したデータベースの一覧が表示されるので、［**Start Assesment**］（アセスメントの開

始）ボタンをクリックします。

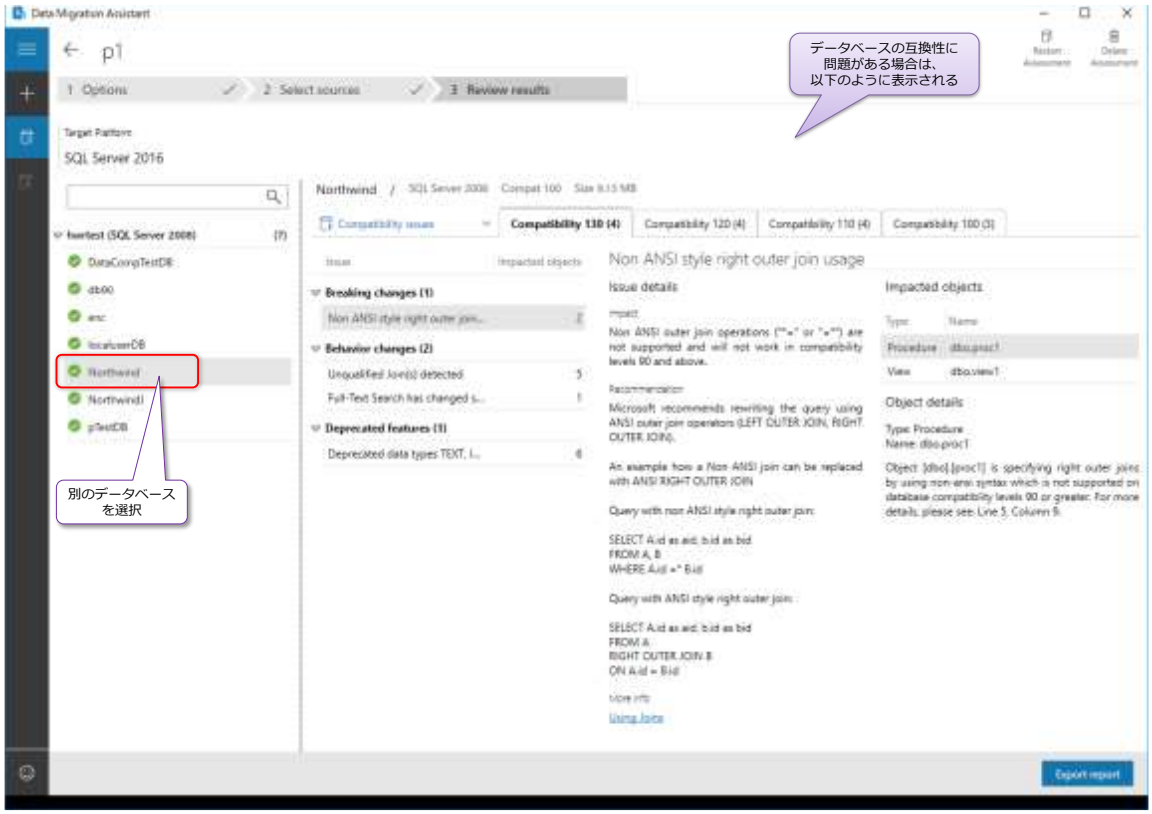



これでデータベースのチェックが始まり、チェックが完了すると、次のように結果が表示されます。

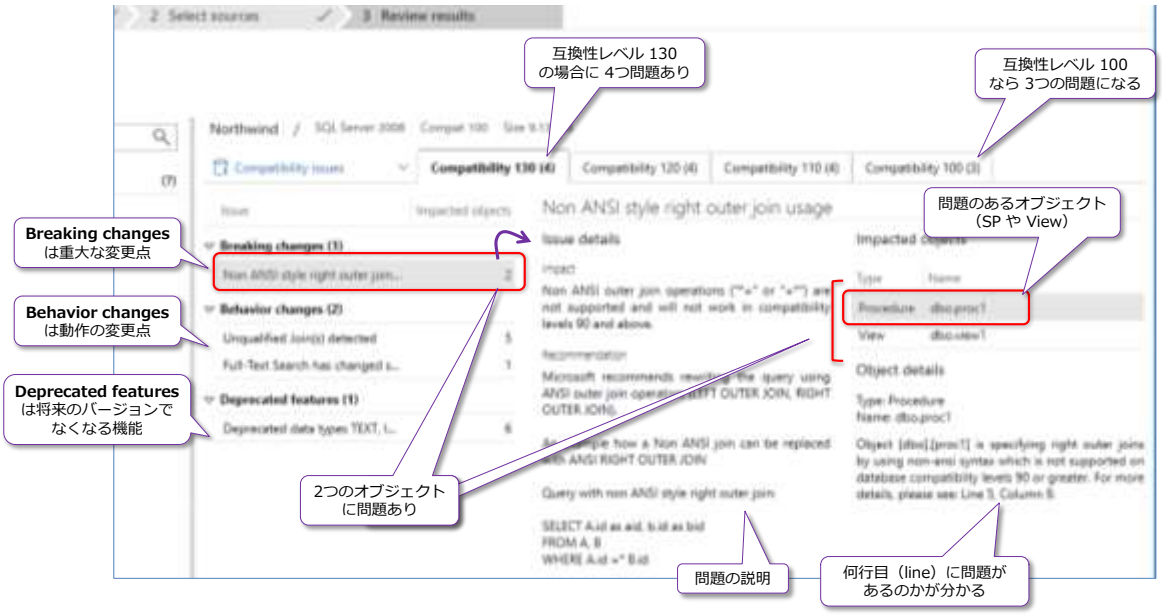



データベースの互換性に問題がない場合は、「**There are no compatibility issues with your database**」と表示されます。

互換性に問題がある場合は、次のように表示されます。




これをズームして、詳しく説明すると、以下のようになります。




問題がある場合は、「**Compatibility 130 (4)**」などのタブが表示されて、これは互換性レベル 130 の場合だと 4 つの問題があることが分かり、その 4 つの詳細が「**Breaking changes**」や「**Behavior changes**」などにリストされています。

**Breaking changes** は、"**重大な変更点**" で、SQL Server 2016（130 レベル）では、未サポートの機能になるので、修正しないと動作しないというものになります。画面は、「**=***」などの外部結合（互換性レベル 90 まで利用できた古い記述）が利用されているストアド プロシージャおよびビューがある場合のメッセージです。SQL Server 2016 からは、**100**（SQL Server 2008 レ

ベル）以上の互換性レベルがサポートされるので、この外部結合を修正する必要があるというメッセージです。また、該当オブジェクトを選択すれば、「**Line 5. Column 9**」のように 5 行目の 9 列目に該当箇所があることを示してくれているので、どこを修正すれば良いのかを簡単に確認することができます。

**Behavior changes** は、"**動作の変更点**" で、SQL Server 2016 で動作が変わったもの（例えば、フルテキスト検索機能が SQL Server 2008 以降で大きく変わっていることや、JOIN を利用しない結合の記述：カンマでテーブルを列挙する結合演算を利用した場合に missing join predicate 警告が発生する場合があるように変わったなど）がリストされています。これらは、SQL Server 2016 にアップグレードした後も、そのまま利用することができますが、気を付けたほうが良い／変更したほうが良いリストになっています。

**Deprecated features** は、"**将来のバージョンでなくなる機能**" で、SQL Server 2016 では利用できますが、SQL Server 2016 の次のバージョン以降ではサポートされなくなる予定のもの（例えば、image や text データ型などはサポートされなくなり、varbinary(max) や varchar(max) を代わりに利用する必要があるなど）です。これも、SQL Server 2016 にアップグレードした後にそのまま利用することができますが、将来を見据えると変更したほうが良いリストになっています。

このように、Data Migration Assistant を利用すれば、データベースをアップグレードしたときの問題点を簡単に見つけることができるので大変便利です。

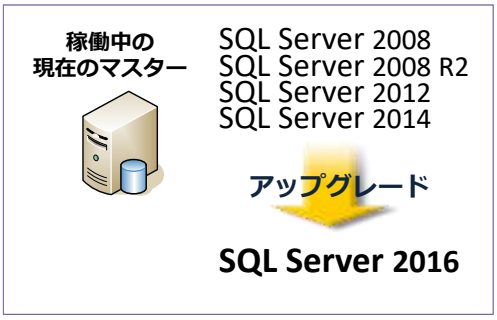
# STEP 3. ケース1 ：同一マシンでの SQL Server 2016 へのアップグレード

この章では、このドキュメントで扱う 3 つのケースのうちの「**ケース 1　同一マシンでのアップグレード**」（**インプレース アップグレード**）について説明します。

- ✓ SQL Server 2016 へのアップグレード要件
- ✓ SQL Server 2016 へのアップグレード インストール
- ✓ SQL Server 2016 の最新の修正プログラムのインストール
- ✓ Management Studio／SSDT ツールのインストール
- ✓ コマンドライン ツールのパス、オンライン ブックの参照
- ✓ 統計の更新
- ✓ データベースの互換性レベルを 130 へ上げる。
  クエリ ストアで互換性レベルの影響（実行プランの変化）をチェックする
- ✓ 提供が中止された機能、動作変更があった機能
- ✓ レプリケーション／ログ配布／DBM／可用性グループのアップグレード
- ✓ Integration Services のアップグレード
- ✓ データ コレクションのアップグレード
- ✓ Reporting Services、Analysis Services のアップグレード

## 3.1 ケース 1「同一マシンでのアップグレード」

この章では、このドキュメントで扱う 3 つのケースのうちの「**ケース 1　同一マシンでのアップグレード**」（インプレース アップグレード）について説明します。




### アップグレード手順の概要

このケースでのアップグレード手順の概要は、次のとおりです。

1. **Data Migration Assistant** による事前チェックを行う
2. **OS** に **Windows Server 2003** や **2003 R2**、**2008**、**2008 R2** を利用している場合は、**Windows Server 2012** 以上に**アップグレード**する

   OS のアップグレードにあたっては、SQL Server を動作させるための Service Pack 要件を確認する（例えば、Windows Server 2012 にアップグレードする場合に、SQL Server 2008 なら SP3、SQL Server 2008 R2 なら SP1 が必要など）
3. **SQL Server 2016** への**アップグレード要件**を確認する（アップグレード可能な **Service Pack** を確認。例えば、SQL Server 2008 なら SP4、SQL Server 2008 R2 なら SP3 が必要など）
4. **SQL Server 2016** への**アップグレード インストール**を行う
5. **SQL Server 2016** の**最新の修正プログラム**（CU や Service Pack）をインストールする
6. **Management Studio**（管理ツール）の最新版をダウンロードして、インストールする（オプション）
7. **統計**（Statistics）を更新する
8. **データベースの互換性レベルを 130** へ上げる（オプション）
9. **BIDS**（Business Intelligence Development Studio）や **SSDT-BI**（SQL Server Data Tools - Business Intelligence）を利用している場合は、**SSDT** の最新版をダウンロードして、インストールする（オプション）

この方法のメリットは、以前のバージョンで利用していた機能をほとんどすべてそのまま利用できることです。**データベース**を以前と変わらずに利用できることはもちろん、**ログイン アカウント**や、**ジョブ**、**警告**、**暗号化**、**リンク サーバー**、**メンテナンス プラン**（保守計画）、**リソース ガバナー**、**監査**（SQL Server Audit）、**データベース メール**、**サーバーの構成オプション**など、以前の SQL Server で設定／利用していた機能を、そのまま SQL Server 2016 上でも利用することができます。

また、**レプリケーション**や**ログ配布**、**データベース ミラーリング**、**可用性グループ**といったサーバー間の連携機能や、**WSFC**（Windows Server フェールオーバー クラスタリング）上の SQL Server インスタンスに関しても、アップグレードをすることができます。ログ配布／データベース ミラーリング／可用性グループに関しては、**セカンダリ**を先にアップグレードすることで、**ローリング アップグレード**も可能です（ローリング アップグレードによって、セカンダリをアップグレード中でも、プライマリを利用することができるので、ダウンタイムを最小限に抑えることができます）。

**Integration Services** や **Reporting Services**、**Analysis Services** を利用している場合でも、アップグレード インストールをすることによって、以前のバージョンで利用していた Integration Services パッケージや Reporting Services レポート、Analysis Services の多次元キューブを、アップグレード後もそのまま利用することができます（これらの機能については、この章の後半で説明します）。

一方で、この方法のデメリットは、アップグレード後に、旧システム環境が利用できなくなってしまう（以前のバージョンの SQL Server が完全に利用できなくなってしまう）ことです。これだと、万が一アップグレードに失敗してしまった場合には、元の環境に戻すのが大変になり、失敗時は、（最悪は）OS のインストールからやり直して、バックアップからすべてを復元しなければならない場合があります。

したがって、このケースを利用する場合は、万が一のアップグレード失敗時に備えて、元の環境でしっかりとバックアップを取得しておくこと、元へ戻す手順（ロールバック手順）をしっかりと計画しておくことが重要になります。また、次の章で説明する「**ケース２ 新規サーバーへのアップグレード**」のように、ハードウェア リプレースを伴うアップグレードの場合であれば、元の環境をそのまま残しておくことができるので、万が一の失敗時にもすぐに元の環境に戻せるようになります。

以降では、ケース１ でのアップグレード手順について、１つ １つの手順を詳しく説明します。

**Note： 32 ビットから 64 ビット環境へのアップグレードの場合**

前述したように、**アップグレード**（インプレース アップグレード）は、**クロス プラットフォーム**をサポートしていないので、32 ビットから 64 ビット環境への変更を行う場合には、第５章で説明する「**移行**」（マイグレーション）の手順を利用する必要があります。

## 3.2 SQL Server 2016 のアップグレード要件

**SQL Server 2016 へのアップグレード**を行うにあたっては、**SQL Server 2016 へのアップグレード要件**（**アップグレード パス**と**インストール要件**）を把握しておく必要があります。

### SQL Server 2016 へのアップグレード パス

第 1 章で説明したように、**SQL Server 2016** は、以下のバージョンの **SQL Server** からの**アップグレード**を行うことができます。

- **SQL Server 2008 SP4** 以降
- **SQL Server 2008 R2 SP3** 以降
- **SQL Server 2012 SP2** 以降
- **SQL Server 2014 RTM** 以降

SQL Server 2016 からは、**SQL Server 2005** からのインプレース アップグレードがサポートされなくなりました（第 5 章で紹介するデータベースの移行は可能です）。

**SQL Server 2008** は **SP4**、**SQL Server 2008 R2** は **SP3**、**SQL Server 2012** は **SP2** 以降を適用しておくことで、インプレース アップグレードを行うことができます。

なお、古いバージョンの SQL Server 上で、ユーザー データベースの互換性レベルが **90**（SQL Server 2005 レベル）以下のものを利用している場合は、アップグレード時に、そのデータベースの互換性レベルが **100**（SQL Server 2008 レベル）に自動的に上がります（詳しくは後述します）。

**クロス プラットフォーム**（**32 ビット**から **64 ビット**への変更）に関しては、インプレース アップグレードがサポートされていないので、**データベースの移行**（マイグレーション）を利用する必要があります。移行方法については、第 5 章で詳しく説明しますが、**移行**であれば、**32 ビット**も **64 ビット**も関係なく行うことができます。

なお、**Microsoft Azure** などのクラウド環境への変更を行う場合にも、第 5 章の「**移行**」の手順で行うことができます。完全に別のサーバー（同じ名前の別サーバーではなく、異なる名前の別サーバー）への変更を行う場合には、移行を利用するようにします（移行の手順も簡単です）。

## SQL Server 2016 をインストール／アップグレードするための OS 要件

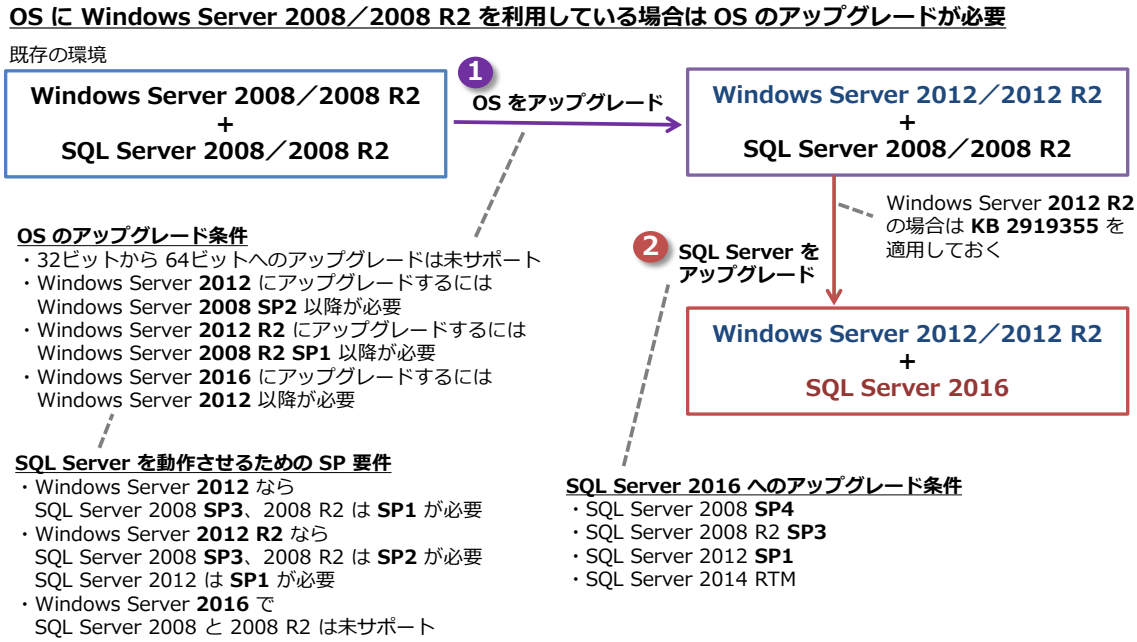
**SQL Server 2016** をインストール（またはアップグレード）するには、以下の **OS** を利用する必要があります。

- **Windows Server 2012**（X64）
- **Windows Server 2012 R2**（X64）＋ **KB 2919355**
- **Windows Server 2016**（X64）

＊ Developer エディションの場合は、Windows 8、Windows 8.1、Windows 10 にインストールすることも可能

以前の SQL Server との大きな違いは、**SQL Server 2016** が **X64**（64 ビット版）の OS のみでサポートされるようになったことと、**Windows Server 2012** 以降の OS が必要になることです。SQL Server 2014 のときは Windows Server 2008 SP2 以降の OS がサポートされていましたが、SQL Server 2016 から変更になりました。

したがって、SQL Server 2008 や 2008 R2、SQL Server 2012 を **Windows Server 2008** や **2008 R2** などで動作させている場合には、次のように OS を **Windows Server 2012** 以上にアップグレードしてから、SQL Server をアップグレードする必要があります。




また、SQL Server 2016 は、X64（64 ビット版）の OS のみがサポートされるので、32 ビット環境の OS を利用していた場合には、OS としても、SQL Server としてもインプレース アップグレードを行うことができません。この場合には、第 5 章で説明する「**移行**」の手順を利用する必要があります。

OS に **Windows Server 2003** や **2003 R2** を利用している場合は、Windows Server 2012 以降に直接アップグレードすることはできないので（∵Windows Server 2012 へのアップグレードには、Windows Server 2008 SP2 以降が必須のため）、Windows Server 2008 や 2008 R2

にアップグレードしてから、Windows Server 2012 以降にアップグレードする必要があります。したがって、Windows Server 2003 や 2003 R2 を利用している場合には、これを機会に OS の新規インストールでの「**移行**」（インプレース アップグレードではなく、移行の実施）を検討するのも 1 つの方法になります（移行方法については、第 5 章で説明しています）。

**Windows Server**（OS）の**アップグレード要件**は、次のようになっています。

**Windows Server のアップグレード要件**

| | アップグレード先 | | |
|---|---|---|---|
| | **Windows Server 2012** | **Windows Server 2012 R2** | **Windows Server 2016** |
| Windows Server 2003／2003 R2 | × 不可 | × 不可 | × 不可 |
| Windows Server 2008 | **SP2** 以降が必要 | × 不可 | × 不可 |
| Windows Server 2008 R2 | **SP1** 以降が必要 | **SP1** 以降が必要 | × 不可 |
| Windows Server 2012 | - | RTM で OK | RTM で OK |
| Windows Server 2012 R2 | - | - | RTM で OK |

**Windows Server 2012** へのアップグレードには、Windows Server **2008 SP2**、**Windows Server 2012 R2** へのアップグレードには、Windows Server **2008 R2 SP1** 以降、**Windows Server 2016** へのアップグレードには Windows Server **2012** 以上が必要になります。

OS を Windows Server 2012 以上にアップグレードする前には、SQL Server の **Service Pack 要件**（以下）も満たしておく必要があります。

**SQL Server を動作させるために必要となる Service Pack**

| | **Windows Server 2012** | **Windows Server 2012 R2** | **Windows Server 2016** |
|---|---|---|---|
| SQL Server 2008 | SP3 | SP3 | 未サポート |
| SQL Server 2008 R2 | SP1 | SP2 | 未サポート |
| SQL Server 2012 | RTM | SP1 | SP2 |
| SQL Server 2014 | RTM | RTM | SP1 |

**SQL Server 2016 にアップグレードするための Service Pack 要件（前掲）**

| | |
|---|---|
| SQL Server 2008 | SP4 |
| SQL Server 2008 R2 | SP3 |
| SQL Server 2012 | SP2 |
| SQL Server 2014 | RTM |

**Windows Server 2012** にアップグレードする前には、**SQL Server 2008** を利用している場合には **SP3** 以上（SQL Server 2016 にアップグレードするためには **SP4** 以上）、**SQL Server 2008 R2** の場合には **SP1** 以上（SQL Server 2016 にアップグレードするためには **SP3** 以上）を適用しておく必要があります。

したがって、SQL Server 2008 の SP1 や SP2、SQL Server 2008 R2 の RTM などで運用している環境を SQL Server 2016 にアップグレードする場合には、SQL Server 2016 へのアップグレード要件である SP（SQL Server 2008 は SP4、SQL Server 2008 R2 は SP3）を、OS のアップグレードする前に適用しておくと、二度手間にならずに済みます（SP を適用する時間の短

縮＝アップグレード時のダウンタイムの削減に繋がります）。

## SQL Server 2016 をインストールするためのソフトウェア要件

**SQL Server 2016** をインストール（およびアップグレード）するための**ソフトウェア要件**は、次のとおりです。

| | 要件 |
|---|---|
| OS | Windows Server 2012 以降 または<br>Windows Server 2012 R2 ＋ KB 2919355 以降 または<br>Windows Server 2016 以降<br>ただし、X64（64ビット版）の OS のみがサポートされます。 |
| ソフトウェア | ・.NET Framework 4.6（事前インストールは不要で、SQL Server 2016 のインストール時に自動的にインストールされます） |

* Developer エディションの場合は、Windows 8、Windows 8.1、Windows 10 にインストールすることも可能です。
* SQL Server 2016 からは、Management Studio と SSDT（SQL Server Data Tools）は、別途ダウンロード／インストールする形に変更されました。
Management Studio は、32ビット版の Windows 7（SP1 以降）や Windows 8 にインストールすることもできます。
Management Studio を Windows Server 2012 または Windows 7／8／8.1 にインストールする場合には KB 2862966 の Security Update が必要になります。

SQL Server 2016 からは、**.NET Framework 4.6** が必須コンポーネントになりましたが、これは SQL Server 2016 のインストール時に自動的にインストールされるので、事前にインストールしておく必要はありません。

OS に **Windows Server 2012 R2** を利用する場合は、**KB**（Knowledge Base：サポート技術情報）の **2919355** を適用しておく必要があります。この KB は、次の URL からダウンロードすることができます。

Windows Server 2012 R2 Update（KB2919355）
http://www.microsoft.com/ja-jp/download/details.aspx?id=42334

**Note： SQL Server と .NET Framework のバージョン**

SQL Server の過去のバージョンと .NET Framework のバージョン（SQL Server を動作させるための必須条件となる .NET Framework）の関係は、次の表のとおりです。

| SQL Server | .NET Framework のバージョン |
|---|---|
| SQL Server 2005 | .NET Framework 2.0 |
| SQL Server 2008 | .NET Framework 3.5 SP1 |
| SQL Server 2008 R2 | .NET Framework 3.5 SP1 |
| SQL Server 2012 | .NET Framework 3.5 SP1 + 更新プログラム<br>.NET Framework 4／4.5 |
| SQL Server 2014 | .NET Framework 3.5 SP1 + 更新プログラム<br>.NET Framework 4／4.5 |
| SQL Server 2016 | .NET Framework 4.6（事前インストールは不要で、SQL Server 2016 のインストール時に自動的にインストールされる） |

## Management Studio と SSDT はダウンロード提供に変更

SQL Server 2016 からは、**Management Studio**（管理ツール）と **SSDT**（SQL Server Data Tools：従来の Business Intelligence Development Studio）は、別途ダウンロードして、インストールする形に変更されました（ダウンロード URL やインストール方法については後述します）。

OS に **Windows Server 2012**（R2 ではない 2012）を利用する場合には、**Management Studio** を利用するために、**KB 2862966** の Security Update が必須になります。

## 3.3 SQL Server 2016 へのアップグレード要件のまとめ

**SQL Server 2016 へのインプレース アップグレード要件**をまとめると、次のようになります。

**インプレース アップグレードできる SQL Server のバージョン**

- **SQL Server 2008 SP4** 以降
- **SQL Server 2008 R2 SP3** 以降
- **SQL Server 2012 SP2** 以降
- **SQL Server 2014 RTM** 以降

アップグレード前の **SQL Server** には **SP**（**Service Pack**）をインストールしておく必要があり、**SQL Server 2008** では **SP4**、**SQL Server 2008 R2** では **SP3**、**SQL Server 2012** では **SP2** 以降が必須になります（**SQL Server 2014** の場合は SP は必須ではありません）。

**SQL Server 2016 をインストール可能な OS（64 ビット版のみ）**

- **Windows Server 2012**（X64）
  Management Studio をインストールする場合は **KB 2862966** が必須
- **Windows Server 2012 R2**（X64）+ **KB 2919355**
- **Windows Server 2016**（X64）

* Developer エディションの場合は、Windows 8、Windows 8.1、Windows 10 にインストールすることも可能

SQL Server 2016 は、**Windows Server 2012** 以降の OS がサポートされるので、**Windows Server 2003** や **2003 R2**、**2008**、**2008 R2** を利用している場合には、SQL Server をアップグレードする前に、OS を Windows Server 2012 以降にアップグレードする必要があります。また、Windows Server 2012 を利用する場合は、SQL Server 2008 なら **SP3**、SQL Server 2008 R2 なら **SP1**、Windows Server 2012 R2 を利用する場合は、SQL Server 2008 なら **SP3**、SQL Server 2008 R2 なら **SP2**、SQL Server 2012 なら **SP1** が必須になります。

**インプレース アップグレードができないケース**

- **SQL Server 2005** を利用している場合
- **32 ビット版**から **64 ビット版**へのアップグレード
- **下位エディション**へのアップグレード（Enterprise から Standard へアップグレードするのは不可など）
  なお、上位エディションへのアップグレードは可能（Standard から Enterprise にアップグレードするなどは可能）

これらのインプレース アップグレードができないケースでは、第 5 章で説明する「移行」の手順を利用することで、移行を行うことができます。

## 3.4 SQL Server 2016 へのアップグレード インストールの手順

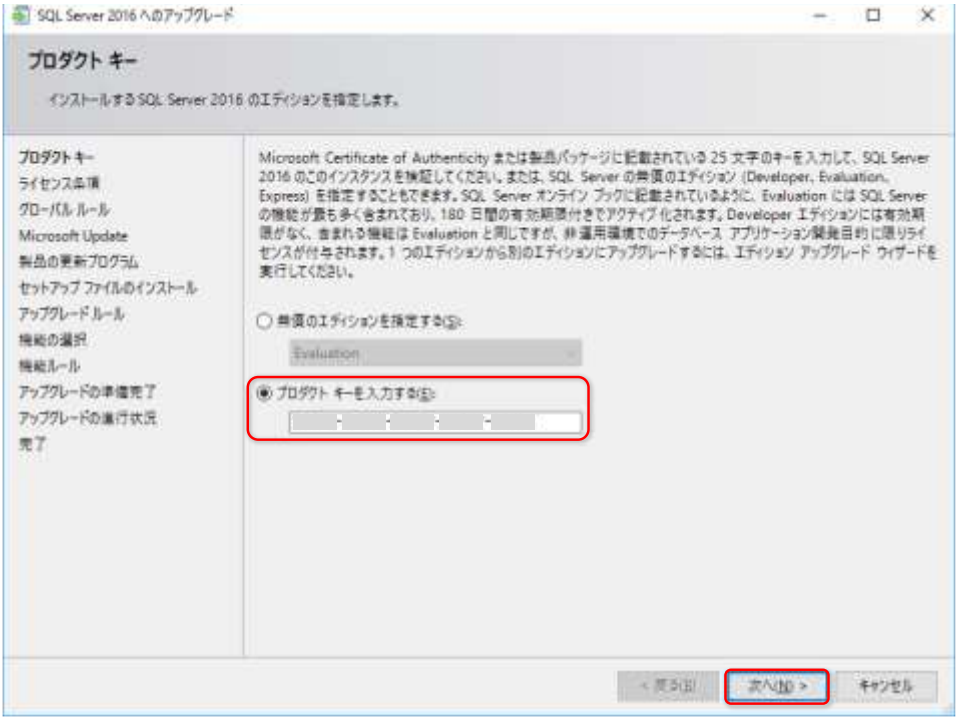
SQL Server 2016 へのアップグレード インストールを行う手順は、次のとおりです。

まずは、SQL Server 2016 のインストール メディアから「**setup.exe**」を実行します。これにより、次のように［**SQL Server インストール センター**］ページが表示されます。


このページでは、次のように［**インストール**］をクリックして、「**以前のバージョンの SQL Server からのアップグレード**」をクリックします。

これで、アップグレード ウィザードが起動して、SQL Server 2016 へのアップグレード インストールをできるようになります。

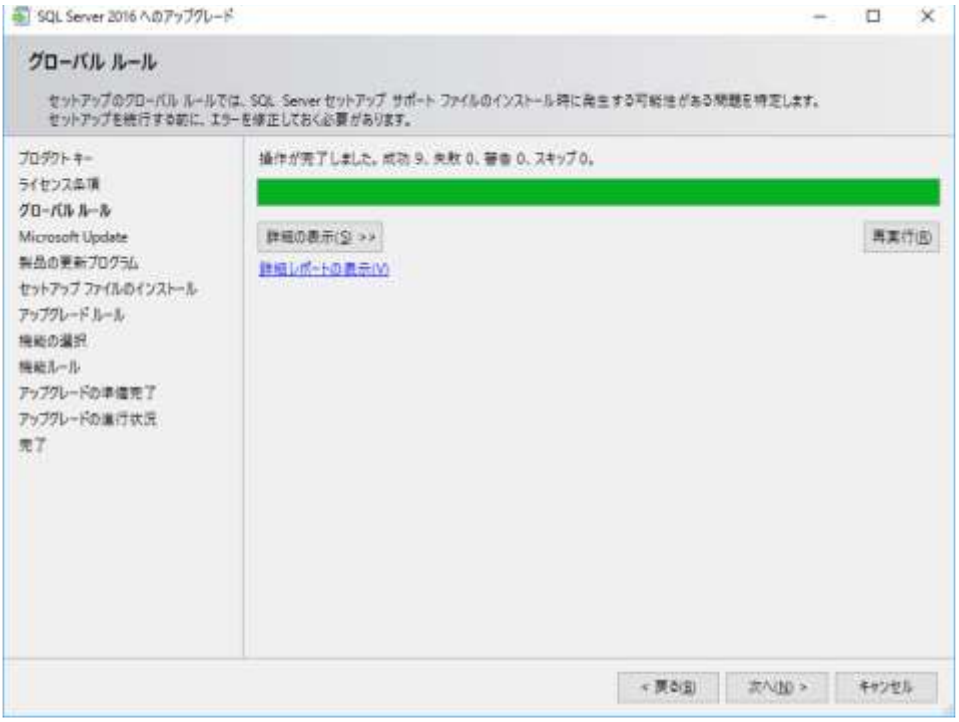
［**プロダクト キー**］ページが表示されたら、プロダクト キーを入力して、［**次へ**］ボタンをクリックします。




次の［**ライセンス条項**］ページでは、ライセンス条項の内容を確認した上で、［**ライセンス条項に同意します。**］をチェックして、［**次へ**］ボタンをクリックします。

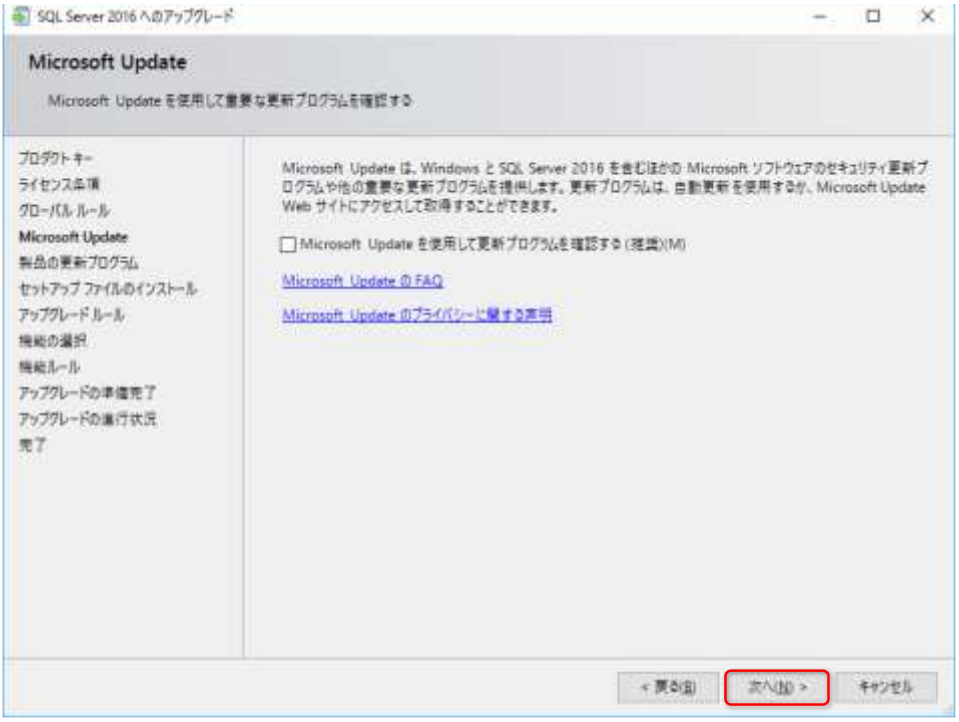

次の［**グローバル ルール**］ページでは、アップグレード要件を満たしているかどうかがチェックさ

れます。

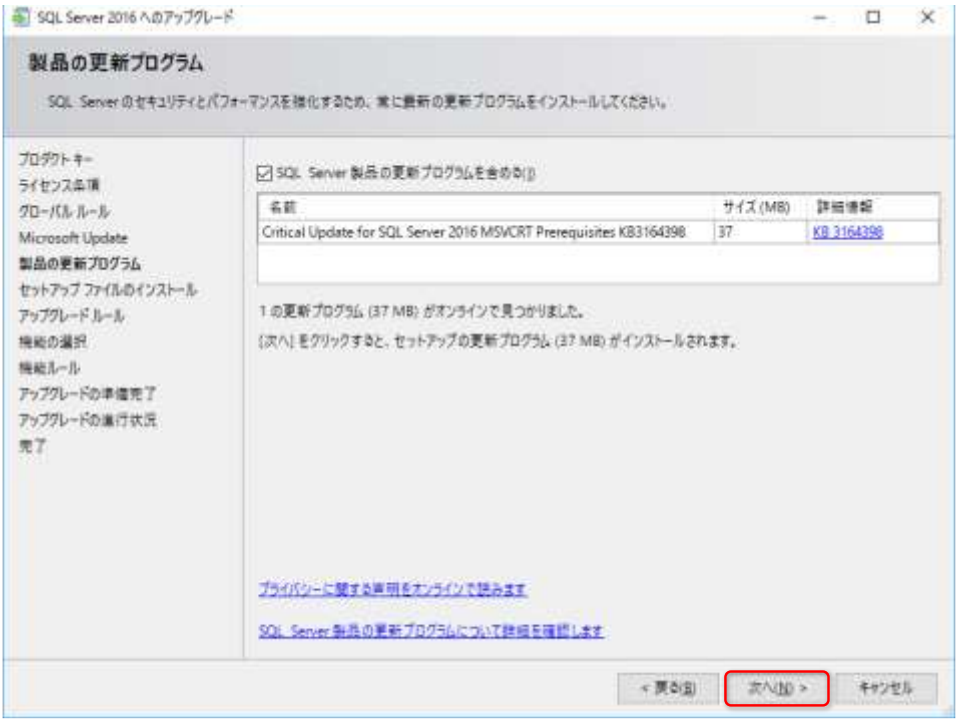



問題がない場合は、自動的に次のページへ進み、問題がある場合（警告や失敗がある場合）にはそれらが提示されます。

次の［**Microsoft Update**］ページでは、Microsoft Update を利用して、更新プログラムを確認するかどうかを設定して、［**次へ**］ボタンをクリックします。

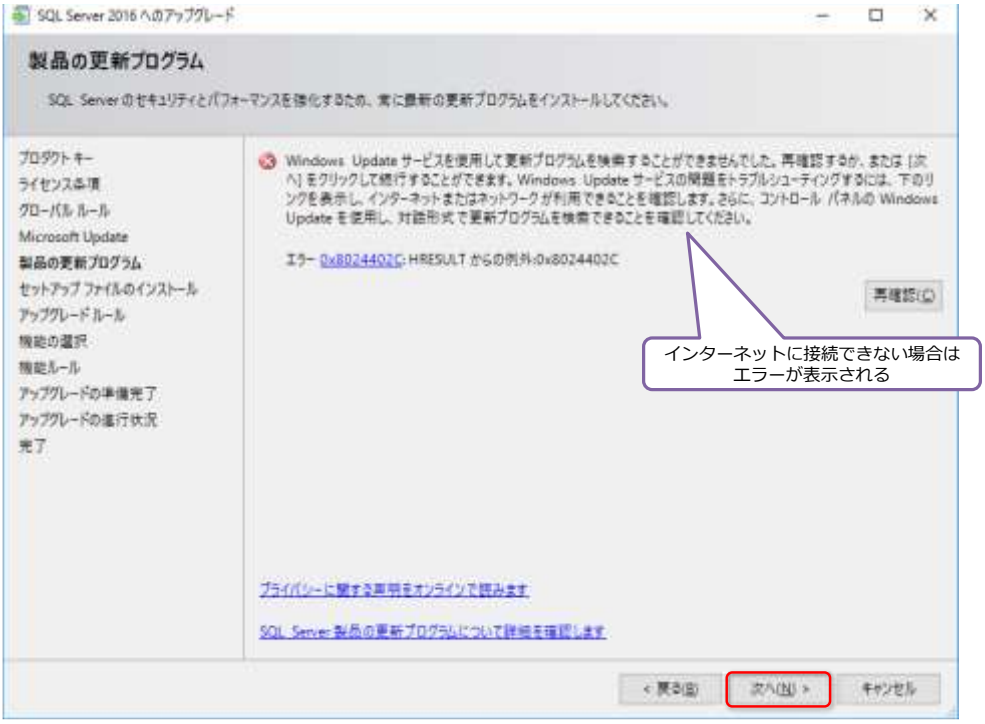



次の［**製品の更新プログラム**］ページは、SQL Server に関する更新プログラムがある場合に表示されます。




［**SQL Server 製品の更新プログラムを含める**］をチェックした場合は、インターネット経由で更新プログラムをダウンロードすることになるので、その分インストールにかかる時間は長くなりますが、更新プログラムがある場合は、更新プログラムを含めておくことをお勧めします。

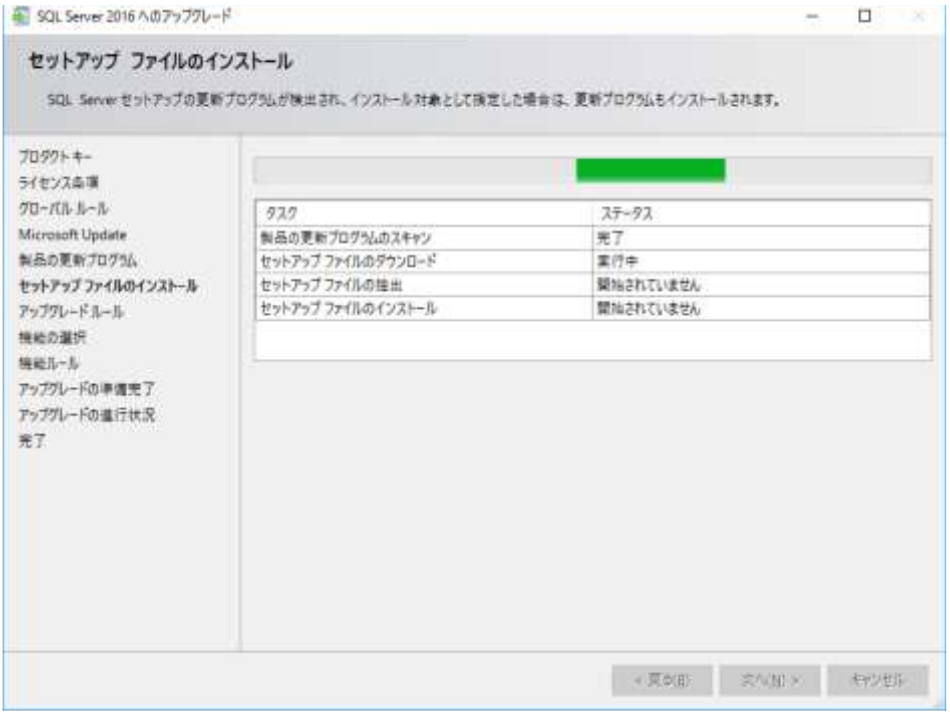
なお、インターネットに接続できない環境の場合には、次のようにエラー ページが表示されます。



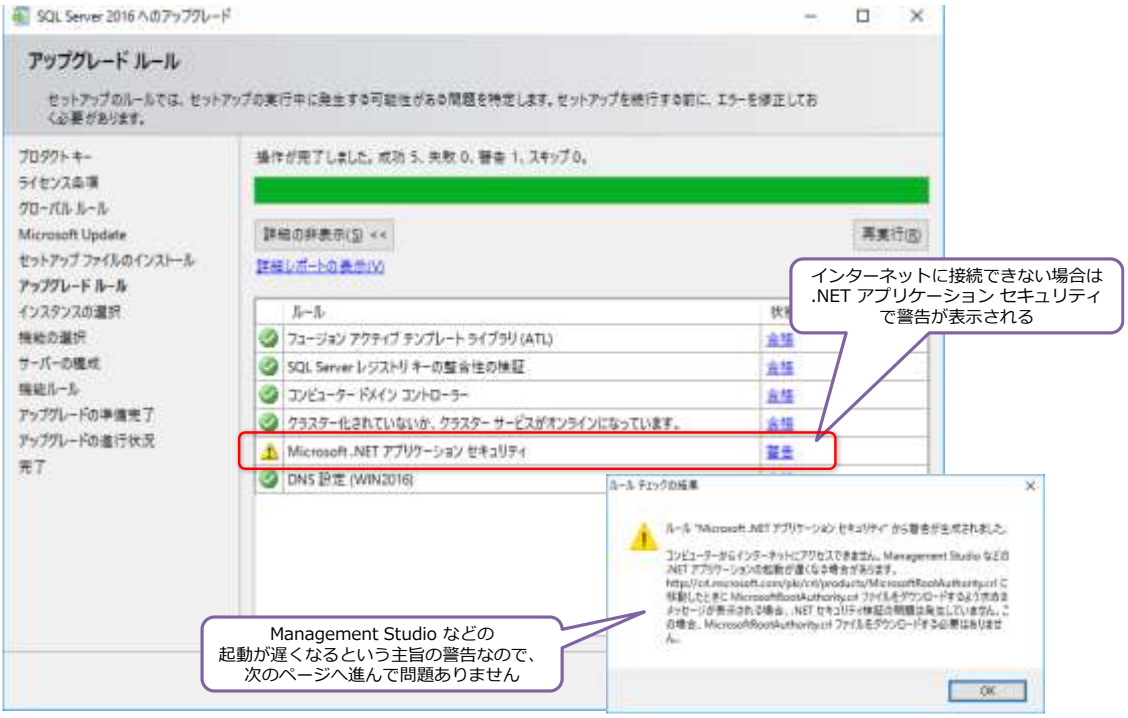

このエラーは、更新プログラムが取得できないという主旨のメッセージなので、［**次へ**］ボタンをクリックして、次のページへ進んで問題ありません。

次の［**セットアップ ファイルのインストール**］ページでは、セットアップに必要なファイルがイン

ストールされます。



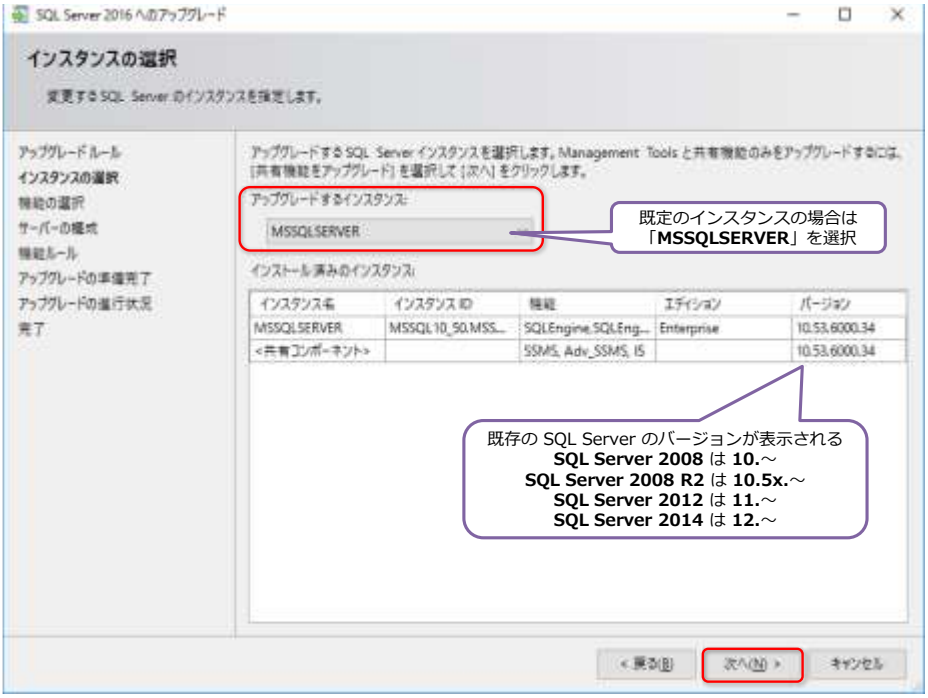

次の [**アップグレード ルール**] ページでは、アップグレード要件を満たしているかどうかが再度チェックされますが、問題がない場合は、自動的に次のページへ進みます。




インターネットに接続できない環境の場合には、、Management Studio などの起動が遅くなるという主旨の警告が表示されますが、[**次へ**] ボタンをクリックして、次のページへ進んで問題ありません。

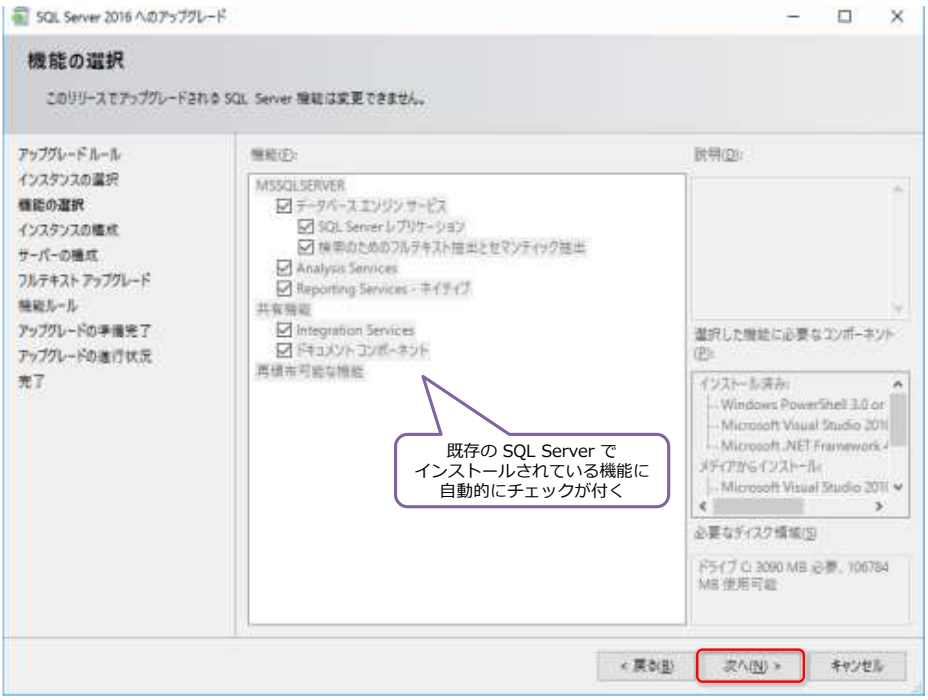
## アップグレードするインスタンスの選択

次の[**インスタンスの選択**]ページでは、[**アップグレードするインスタンス**]で **SQL Server 2016** へアップグレードする既存の SQL Server のインスタンスを選択します。

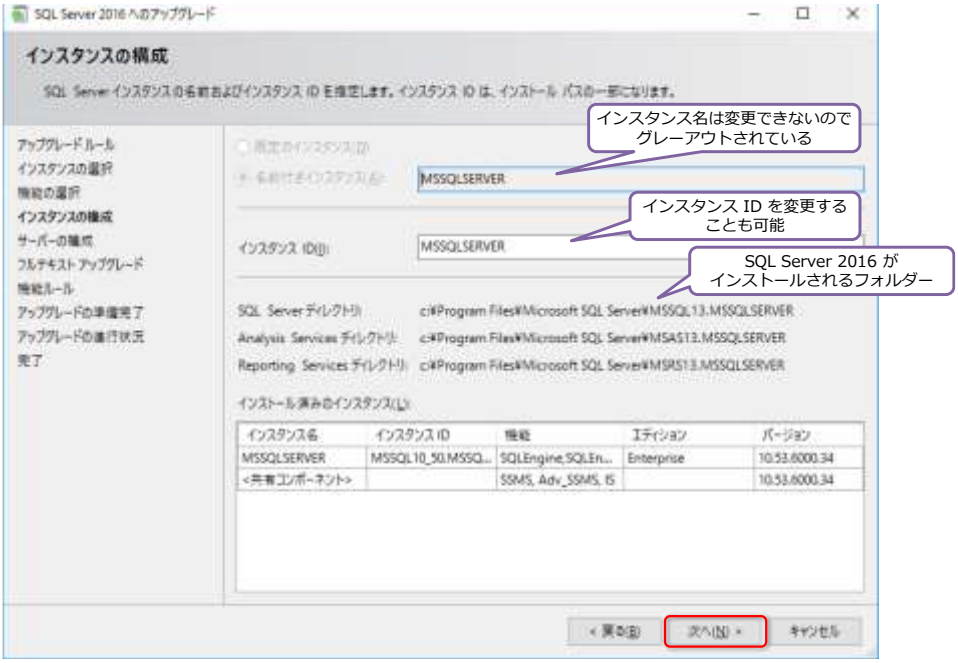



既定のインスタンスをアップグレードする場合は、[**アップグレードするインスタンス**]で「**MSSSQLSERVER**」を選択して、[**次へ**]ボタンをクリックします。

次の［**機能の選択**］ページでは、選択したインスタンスにインストールされている機能に、自動的にチェックが付きます（アップグレード対象になります）。

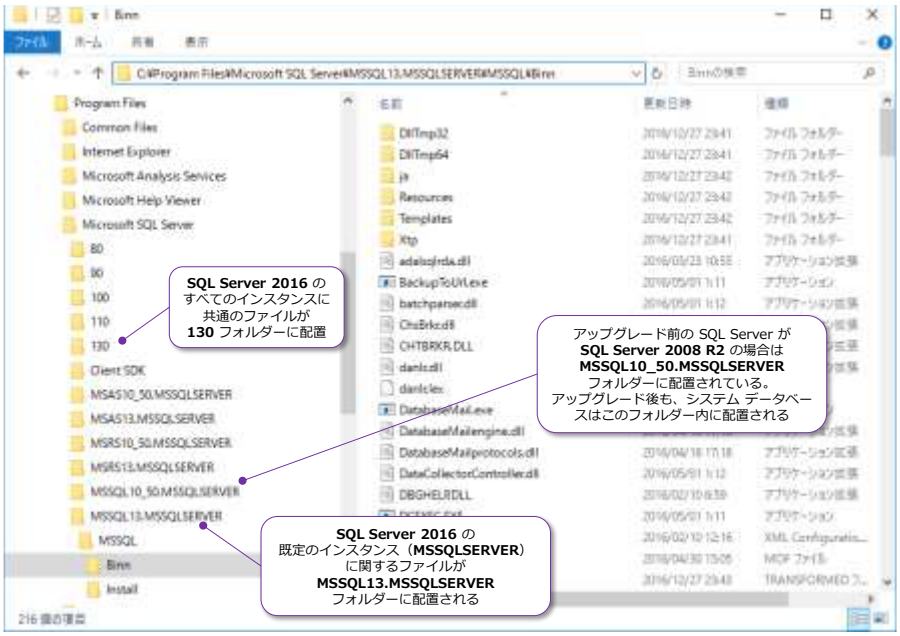


次の［**インスタンスの構成**］ページでは、［**インスタンス ID**］や、インストール先となるフォルダー（ディレクトリ）が提示されます。



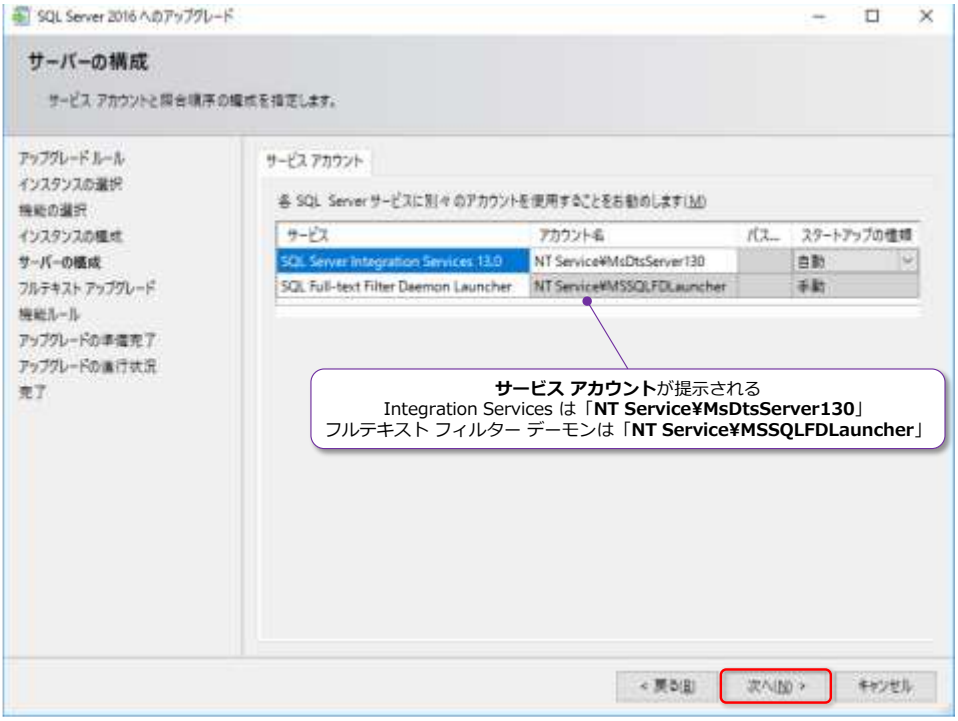

アップグレード時は、［**インスタンス名**］は変更することができないので、インスタンス名はグレーアウトされています（既定のインスタンスの場合は **MSSQLSERVER**）。**インスタンス ID** は、内部的にインスタンスを識別するための ID で、既定値はインスタンス名と同じものになります。これは変更することもできますが、インスタンス名と同じにしておいた方が分かりやすいので、変更しないことをお勧めします（インスタンス ID を変更した場合は、インストール先のフォルダーなどがその ID に変更されます）。

**SQL Server 2016** は、［**SQL Server ディレクトリ**］に表示されたフォルダー（**C:¥Program Files¥Microsoft SQL Server¥MSSQL.13¥インスタンス ID**）にインストールされて、インストール後のフォルダー構成は、次のようになります。



システム データベースなどは、アップグレード前と同じフォルダーを利用しますが、SQL Server 2016 のインスタンスに関する情報は、[**SQL Server ディレクトリ**] に表示されたフォルダーにインストールされて、各種ツールなど、すべてのインスタンスに共通のファイルが「**130**」フォルダー配下に格納されます。

次の [**サーバーの構成**] ページの [**サービス アカウント**] タブでは、**Integration Services** の**サービス アカウント**を変更することができます（既存の SQL Server に Integration Services をインストールしている場合）。

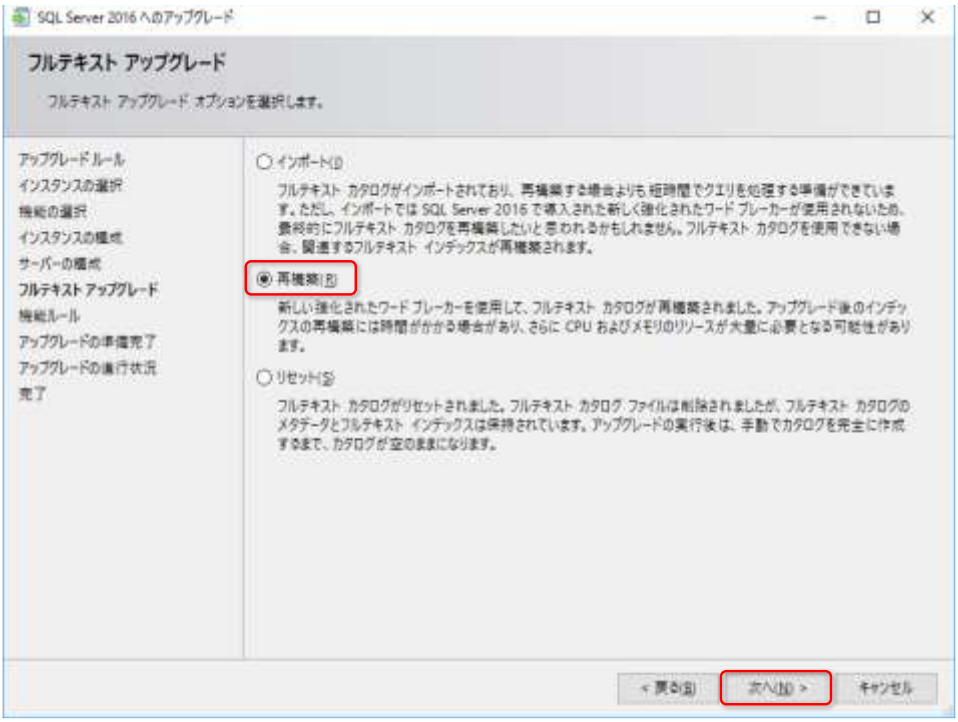



**Integration Services** の**サービス アカウント**には、「**NT Service¥MsDtsServer130**」アカウント（ローカルの内部アカウント）が提示されます。既存の SQL Server で**フルテキスト検索**機能を利用している場合は、**フルテキスト フィルター デーモン サービス**のサービス アカウントとして「**NT Service¥MSSQLFDLauncher**」が提示されますが、このアカウントは変更することはできません（このアカウントを利用します）。

ここには表示されない「**SQL Server データベース エンジン**」や「**SQL Server エージェント**」のサービス アカウントに関しては、既存の SQL Server の設定がそのまま引き継がれます。

## フルテキスト インデックスの再構築

既存の SQL Server で、**フルテキスト検索**機能を利用している場合は、次のように [**フルテキスト アップグレード**] ページが表示されます。

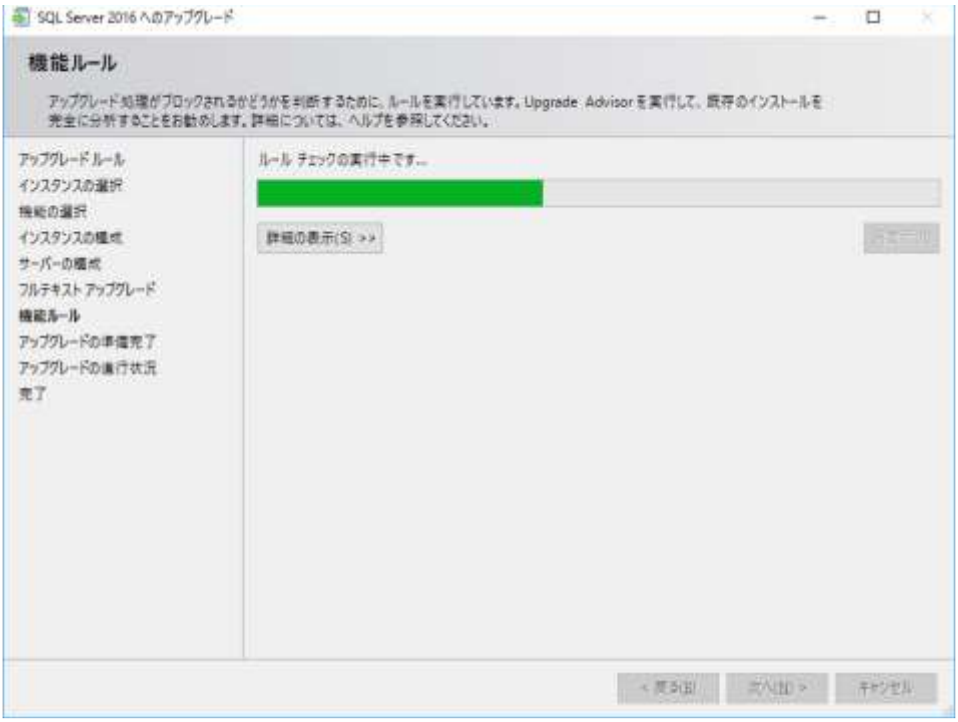



フルテキスト検索機能は、SQL Server 2016 で内部アーキテクチャが変更されているので、このページでは［**再構築**］を選択することで、新しいアーキテクチャに対応させた形にフルテキスト インデックスを再作成することができます。ただし、この再構築は、イチからフルテキスト インデックスを作成するのと同じだけの時間がかかるので、テーブル サイズが大きい場合には、実行時間がかかってしまいます。

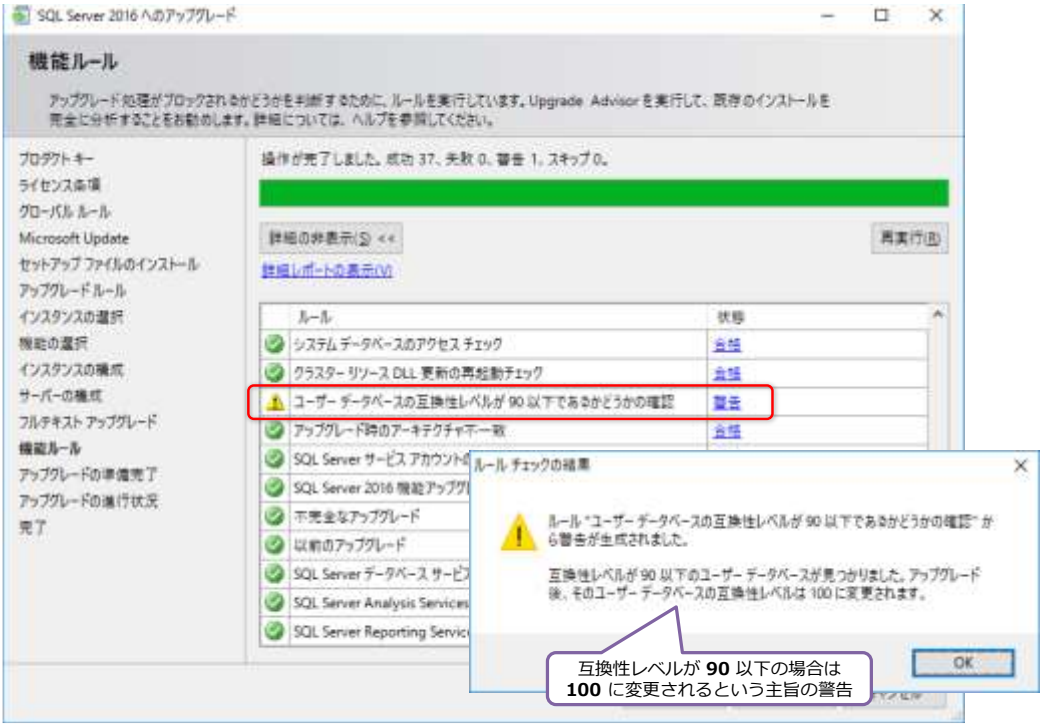
これを避けるには、［**インポート**］を選択して、既存の SQL Server のフルテキスト インデックスをインポート（そのまま取り込み）するようにします。これで、既存の SQL Server 上で作成されたフルテキスト インデックスをそのまま SQL Server 2016 に取り込むことができ、（暫定的に）利用することができます。この場合は、アップグレード後に、手動で再構築を行うことで、SQL Server 2016 の新しいアーキテクチャへ対応させることができます（手動で再構築を行う手順については、第５章で説明しています）。

## アップグレード前の最終チェック

次の［**機能のルール**］ページでは、アップグレードにあたっての条件を満たしているかどうかの最終チェックが行われます。



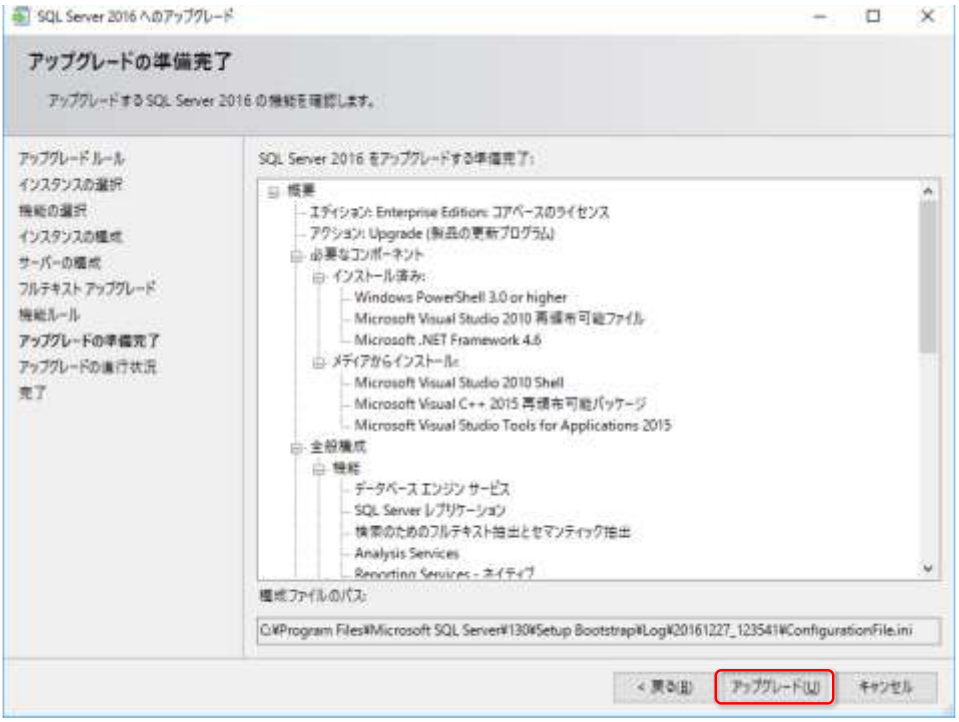

既存の SQL Server 環境に、互換性レベルが **90**（**SQL Server 2005** レベル）以下のデータベースが存在する場合は、次のように**データベースの互換性レベル**に関する警告が表示されます。



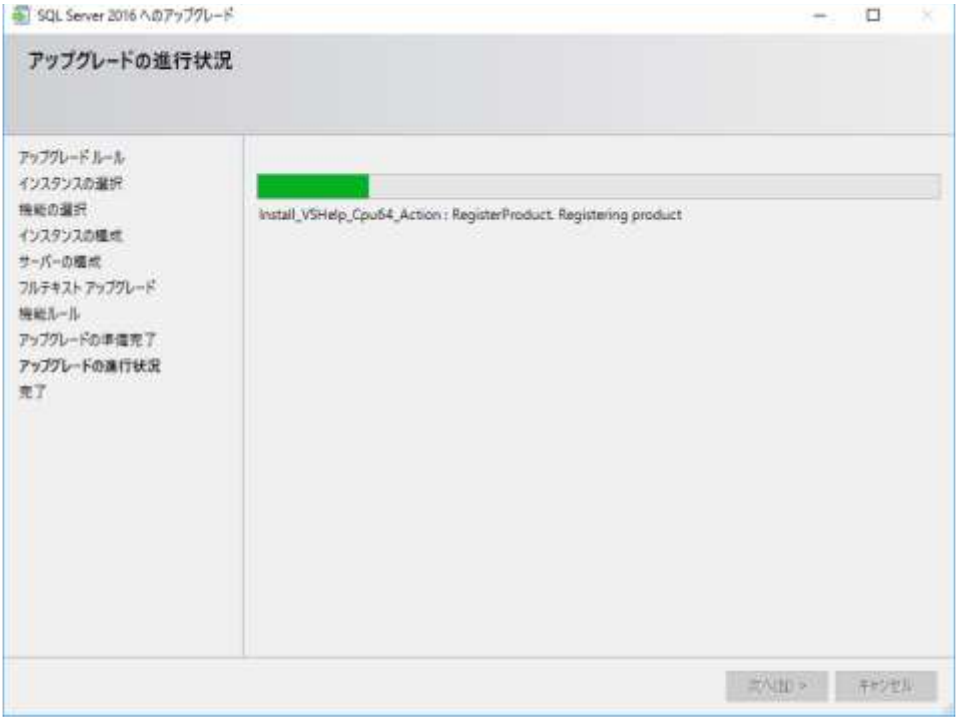

**SQL Server 2005** レベルのユーザー データベースは、SQL Server 2016 ではサポートされていないので、SQL Server 2016 へのアップグレードによって、互換性レベルが **100**（SQL Server 2008 レベル）に自動変更されるという主旨の警告になります（互換性レベルについては後述します）。

## アップグレードの実行

次の［**アップグレードの準備完了**］ページでは、［**アップグレード**］ボタンをクリックすることで、アップグレードが開始されます。

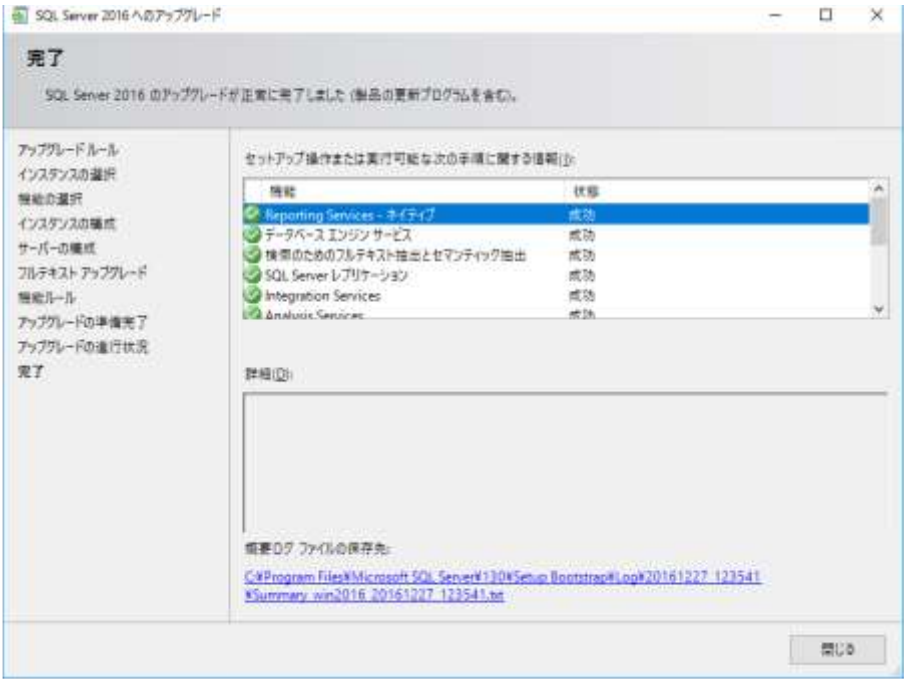



アップグレード中は、次のように進行状況が表示されます。



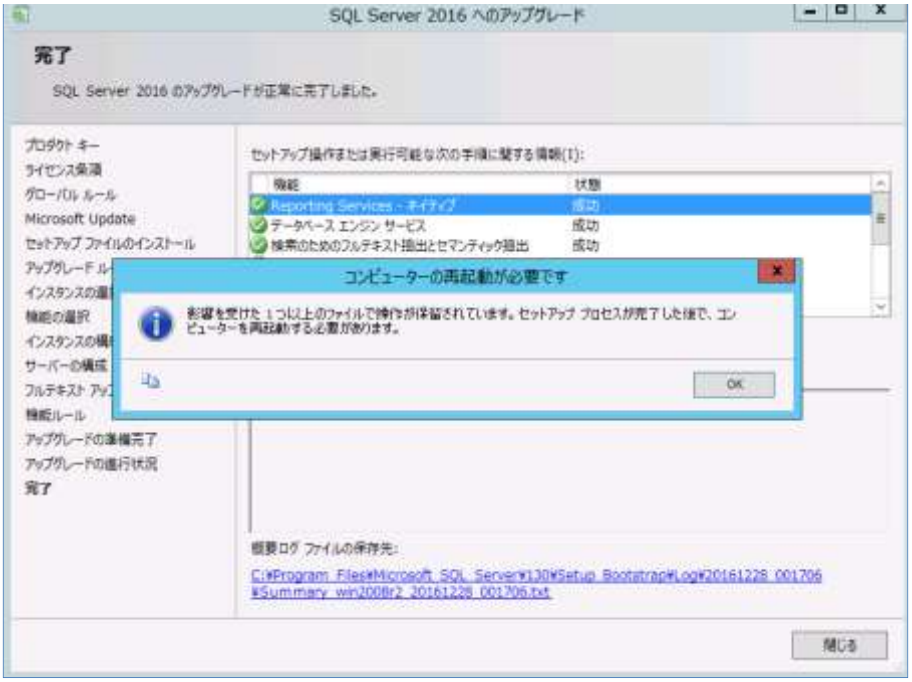

アップグレードにかかる時間は、以前のバージョンでインストールされていた機能の種類や、フルテキスト インデックスを再構築するかどうか、ハードウェア環境（SSD などの高速ストレージかどうか）などによって、大きく前後しますが、15 分～1 時間 30 分程度で完了します。

アップグレードが完了すると、次のように［**完了**］ページが表示されます。



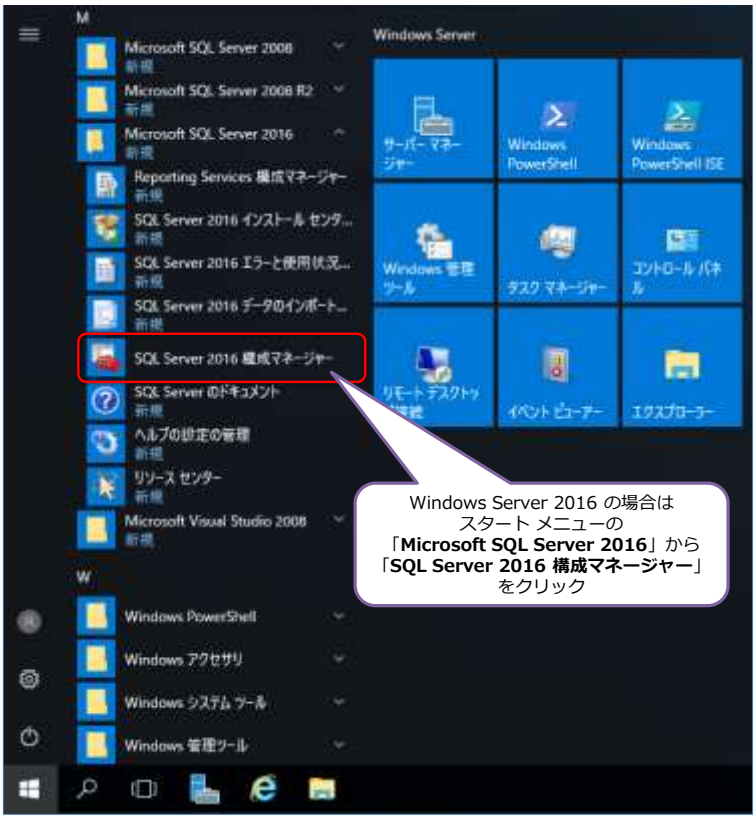

以上で SQL Server 2016 へのアップグレードが完了です。

OS に Windows Server 2012 や 2012 R2 を利用している場合は、アップグレードの完了に**OS の再起動**が必要になる場合があり、その場合は、次のように表示されます。



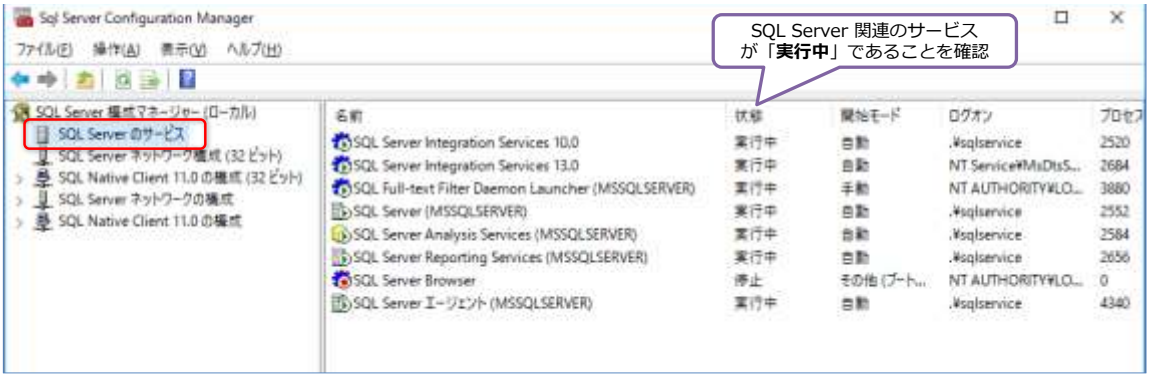
## SQL Server 構成マネージャーでサービスの確認

アップグレード完了後は、「**SQL Server 2016 構成マネージャー**」ツールを利用して、サービスが正常起動しているかどうかを確認します。Windows Server 2016 の場合は「**スタート**」メニュー、Windows Server 2012／2012 R2 の場合はスタート画面の「**Microsoft SQL Server 2016**」グループから「**SQL Server 2016 構成マネージャー**」をクリックします。



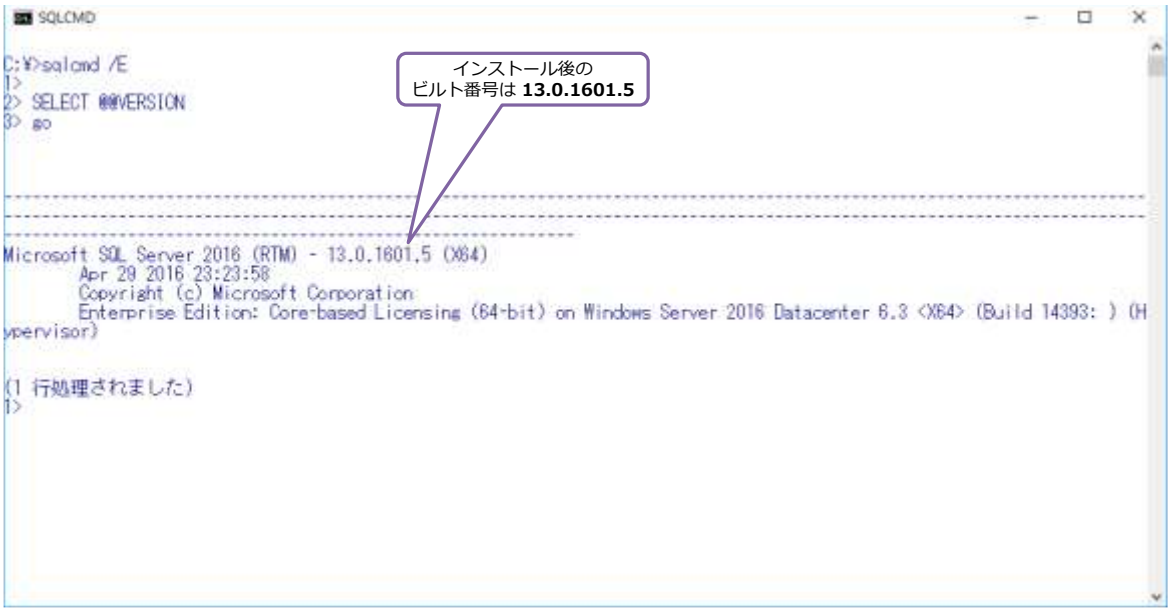

Windows Server 2012／2012 R2 の場合はスタート画面から起動

**SQL Server 2016 構成マネージャー**が起動したら、SQL Server サービスや、SQL Server エージェント サービスなど、以前のバージョンで利用していた SQL Server 関連のサービスが「**実行中**」になっていることを確認します。




## SQL Server のビルド番号の確認

SQL Server 2016 の製品版（RTM）にアップグレードした後のビルド番号は「**13.0.1601.5**」です（アップグレード時に、製品更新プログラムを含めている場合には、これよりも高いバージョンになる場合があります）。これを確認するには、次のように **sqlcmd** ツールを利用して、「**SELECT @@VERSION**」を実行します（後述の Management Studio のインストールを実行した後であれば、Management Studio から確認できます）。

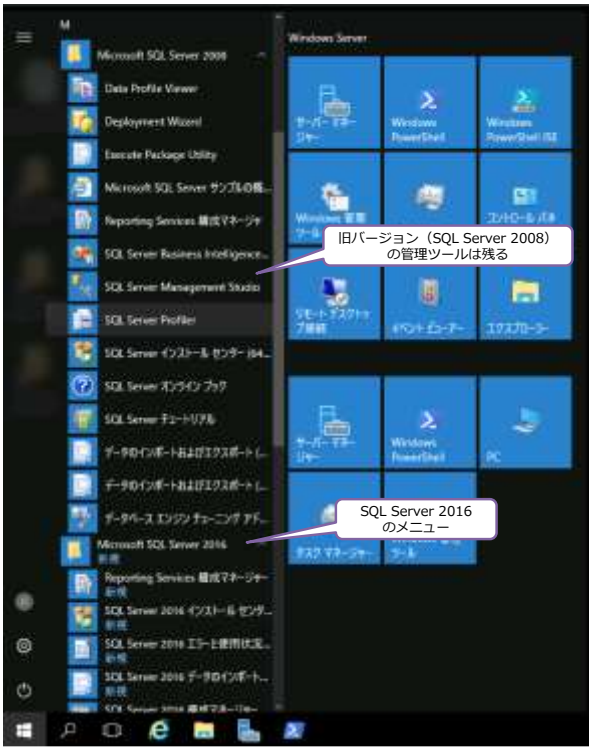



なお、このバージョン番号は、後述の CU4 を適用すると「**13.0.2193.0**」、**SP1**（Service Pack 1）の場合は、「**13.0.4001.0**」、SP1+CU1 の場合は「**13.0.4411.0**」に変わります。

## 旧バージョンの管理ツールは削除されない

SQL Server 2016 へのアップグレードが完了しても、**旧バージョンの管理ツール**は削除されずに残っています（サイド バイ サイドで、旧バージョンと SQL Server 2016 の両方のツールを利用できますが、旧バージョンのツールから SQL Server 2016 へ接続できるかどうかは、ツールのバージョンによって異なります）。

Windows Server 2016 環境で、SQL Server 2008 を SQL Server 2016 へアップグレードした場合




Windows Server 2012 R2 環境で、SQL Server 2008 を SQL Server 2016 へアップグレードした場合

## 3.5 SQL Server 2016 へのアップグレード後の作業

アップグレードが完了した後は、ほとんどすべての機能が、以前の SQL Server と同じように利用することができます。

アップグレード後の主な作業は、次のようになります（次項以降で詳しく説明します）。

- **SQL Server 2016** の**最新の修正プログラム**（SP1 や CU）のインストール
- 最新版の **Management Studio** のダウンロードとインストール（オプション）
- **オンライン ブック**（Books Online）の参照方法
- **コマンドライン ツール**（sqlcmd や bcp ツールなど）のパスについて（旧バージョンが既定になる）
- 最新版の **SSDT**（SQL Server Data Tools）のダウンロードとインストール（**BIDS** や SSDT-BI を利用していた場合）
- 統計の更新
- **互換性レベルを 130** へ上げる（オプション）。
  クエリ ストアで互換性レベルの影響（実行プランの変化）をチェックする

## 3.6 SQL Server 2016 の修正プログラムのインストール

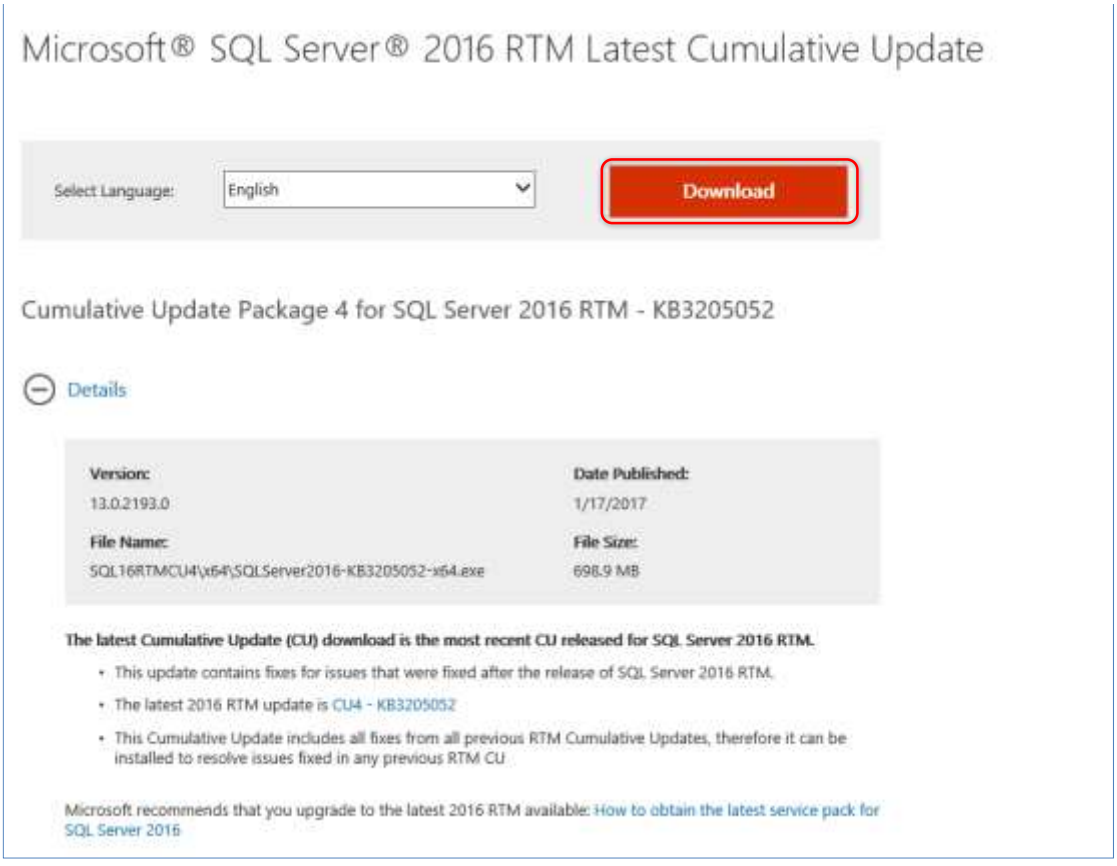
**SQL Server 2016** の製品出荷後には、**修正プログラム**がリリースされているので、アップグレード後は、これもインストールしておくことをお勧めします。**CU2**（累積的な更新プログラム２）には、性能向上に関する修正が入っているので、できる限り最新の修正プログラムを適用しておくことをお勧めします。また、**Reporting Services** を利用している場合は、SP1 に対する重要な更新プログラム（KB 3207512）が提供されているので、これを含んだ **CU4** または **SP1＋CU1** 以上を適用しておくことをお勧めします。

なお、本ドキュメントの執筆時点（2017 年 1 月）では、**CU4** と **SP1**（Service Pack 1）＋**SP1 に対する CU1** が最新の修正プログラムですが、今後 **SP2** や **CU5** など、新しい修正プログラムが提供された場合には、それらをインストールしておくことをお勧めします。

### SQL Server 2016 CU4 のダウンロード／インストール

**SQL Server 2016** の **CU4**（KB 3205052）は、次の URL からダウンロードすることができます。

Microsoft SQL Server 2016 RTM Latest Cumulative Update
https://www.microsoft.com/en-us/download/details.aspx?id=53338

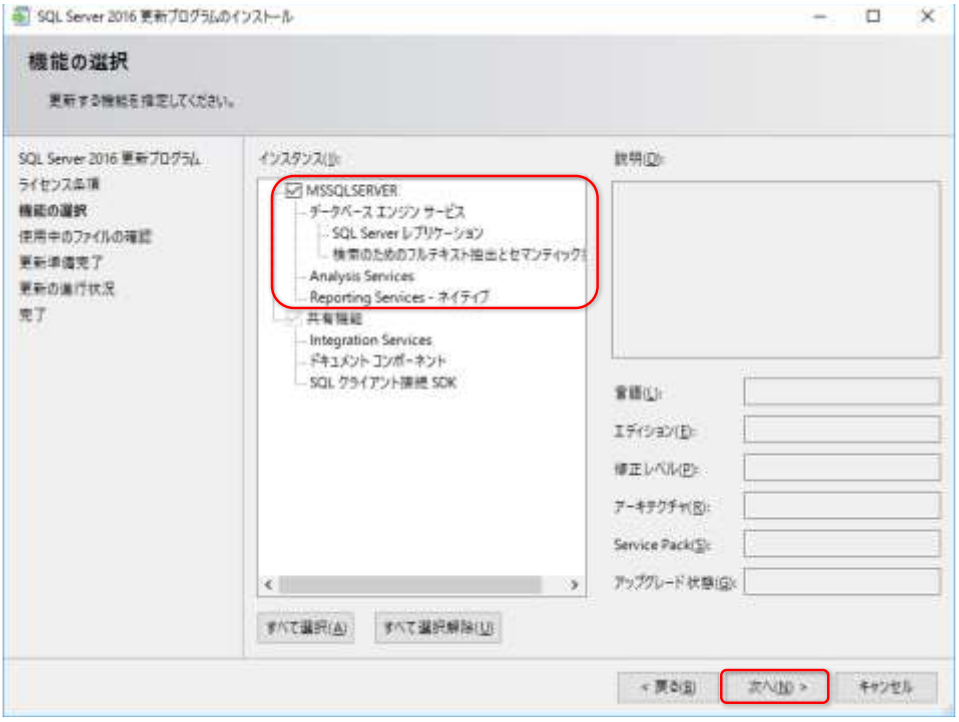



「**Download**」ボタンをクリックすると、「**SQLServer2016-KB3205052-x64.exe**」ファイルをダウンロードすることができます。これを実行すると、次のように CU4 をインストールするこ

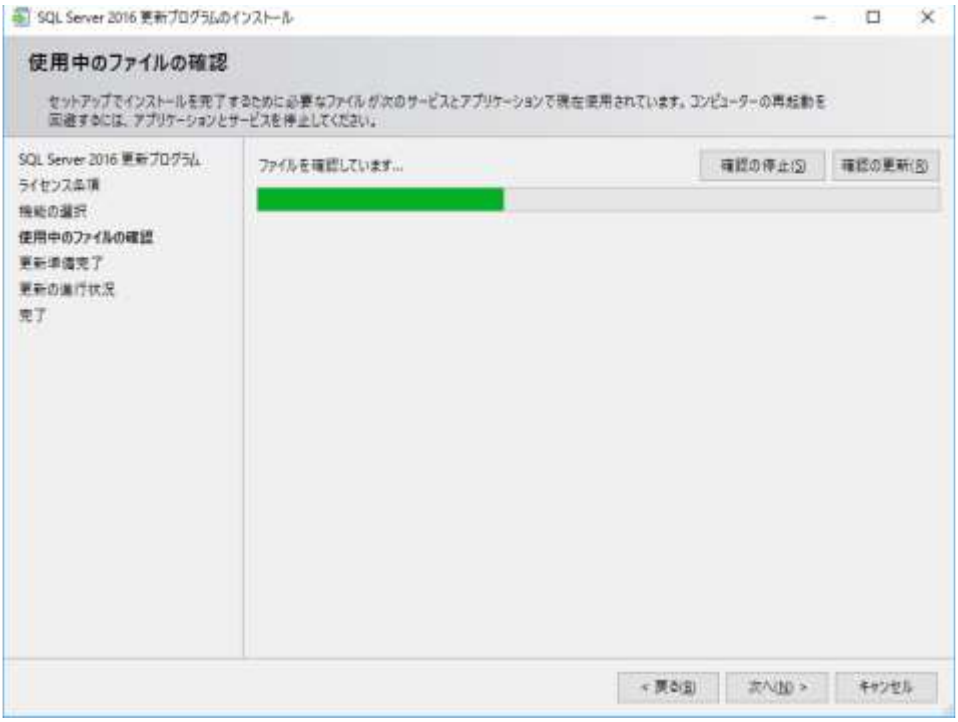
とができます。

最初の［**ライセンス条項**］ページでは、ライセンス条項の内容を確認した上で、［**ライセンス条項に同意します。**］をチェックして、［**次へ**］ボタンをクリックします。

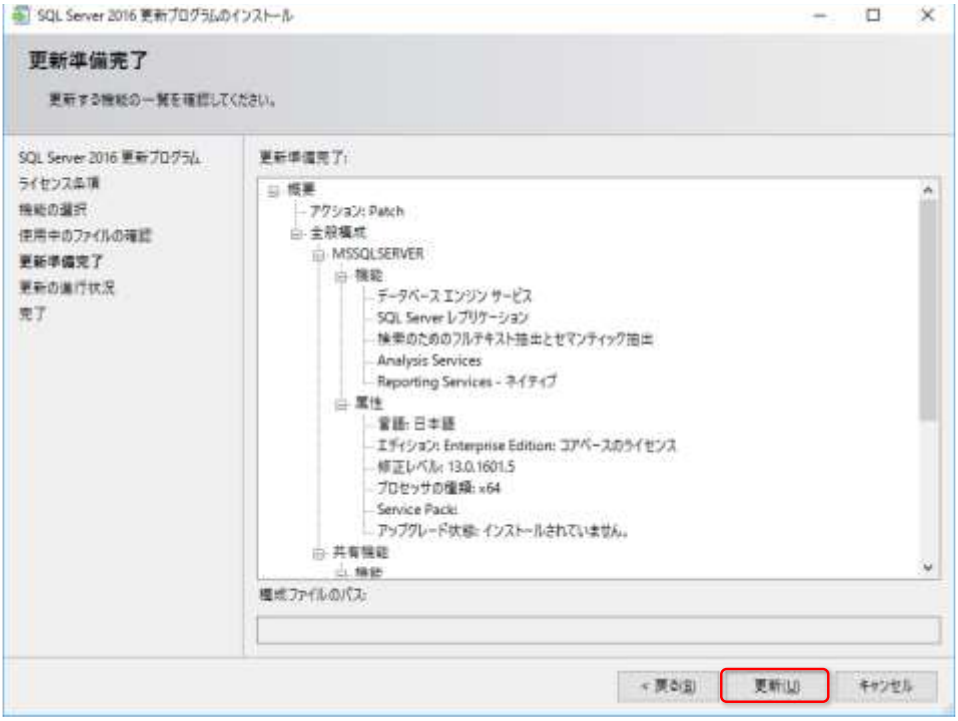

次の［**機能の選択**］ページでは、［**MSSQLSERVER**］（既定のインスタンスの場合）をチェックして、［**次へ**］ボタンをクリックします。



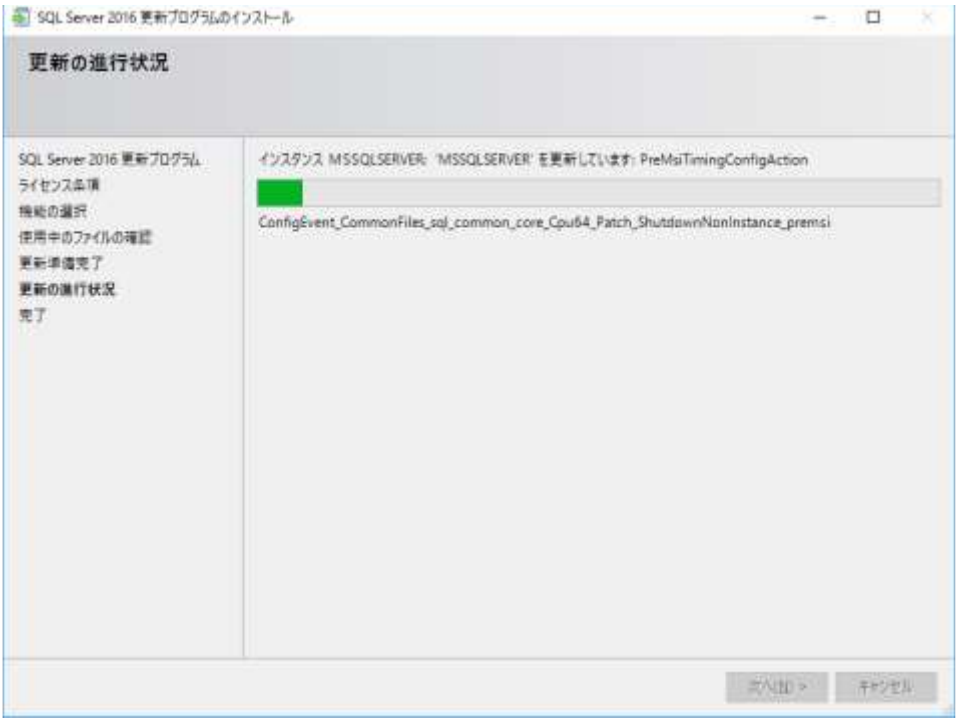

次の［**使用中のファイルの確認**］ページでは、使用中のファイルがあるかどうかがチェックされます。




次の［**更新準備完了**］ページでは、［**更新**］ボタンをクリックします。

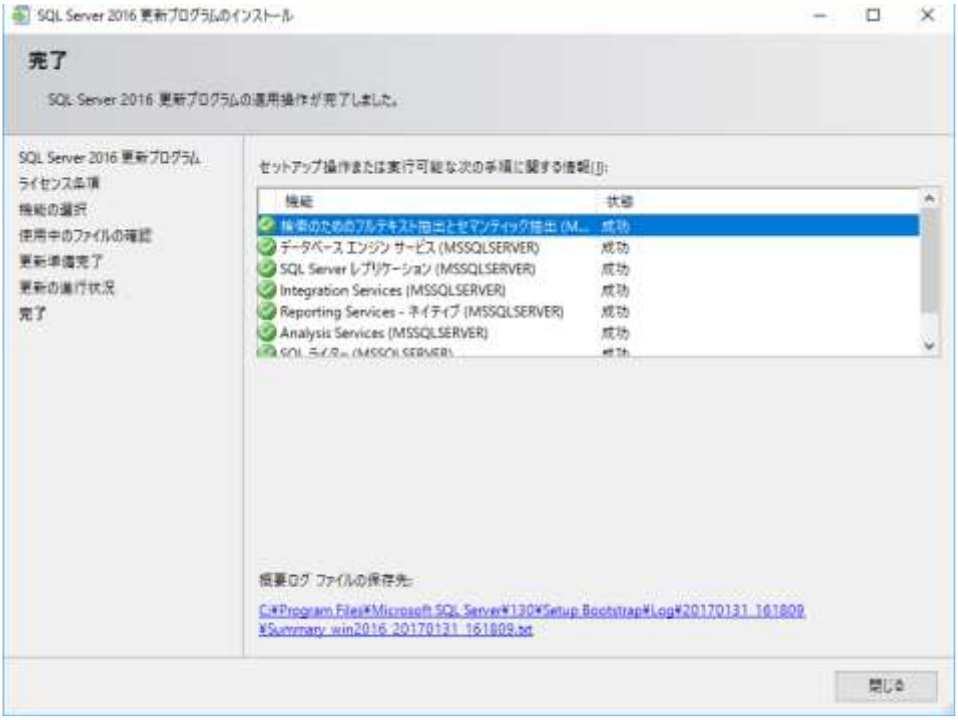



これで **SP1** のインストールが始まって、インストール中は、次のように［**インストールの進行状況**］ページが表示されます。




インストールにかかる時間は、環境によって大きく変化しますが、10 分～1 時間くらいです。

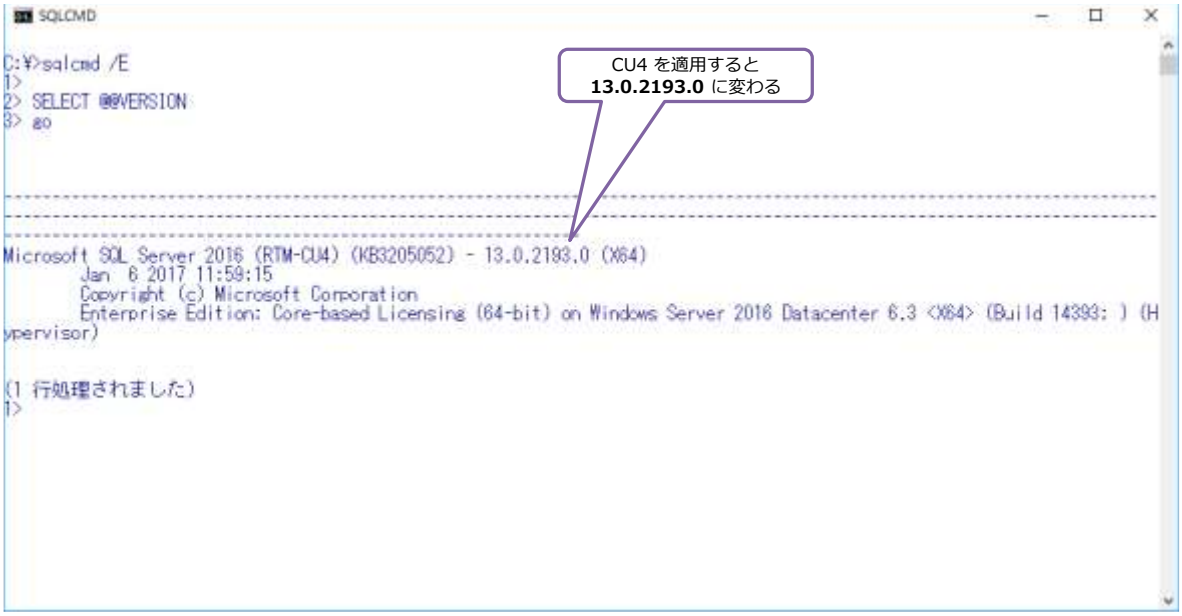
インストールが完了すると、次のように［**完了**］ページが表示されます。




Reporting Services やパフォーマンス データ コレクション機能を利用している場合には、インストール完了後に OS を再起動しておくことをお勧めします。

## CU4 がインストールされたことの確認 ～13.0.2193.0～

CU4 のインストールが完了すると、SQL Server のビルド番号は「**13.0.2193.0**」に変わります（CU4 をインストールする前は **13.0.1601.5**）。これを確認するには、次のように **sqlcmd** ツールを利用して、「**SELECT @@VERSION**」を実行します（後述の Management Studio のインストールを実行した後であれば、Management Studio から確認できます）。




## SQL Server 2016 SP1 のをインストールする場合

**CU4** は、SP1 以降に提供されたものなので、SP1 の修正内容を含んでいますが、**SP1** をダウンロードしたい場合は、次の URL から行えます。

Microsoft SQL Server 2016 Service Pack 1（SP1）
http://www.microsoft.com/ja-JP/download/details.aspx?id=54276

SP1 を適用した場合は、SQL Server のビルド番号は「**13.0.4001.0**」に変わります。

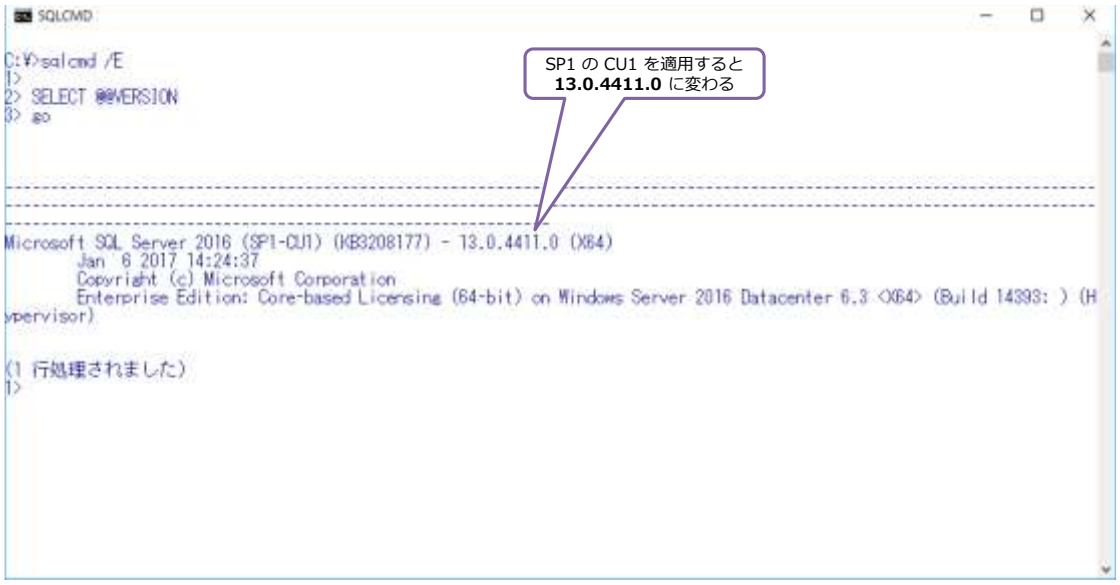

## SP1+CU1 のインストール

SP1 をインストールしている場合は、SP1 に対する CU1 が提供されているので、これもインストールしておくことをお勧めします（SP1+CU1 は、RTM+CU4 に相当する修正になります）。

SP1＋CU1 は、次の URL からダウンロードできます。

Microsoft SQL Server 2016 SP1 の最新の累積的な更新プログラム
http://www.microsoft.com/ja-JP/download/details.aspx?id=54613

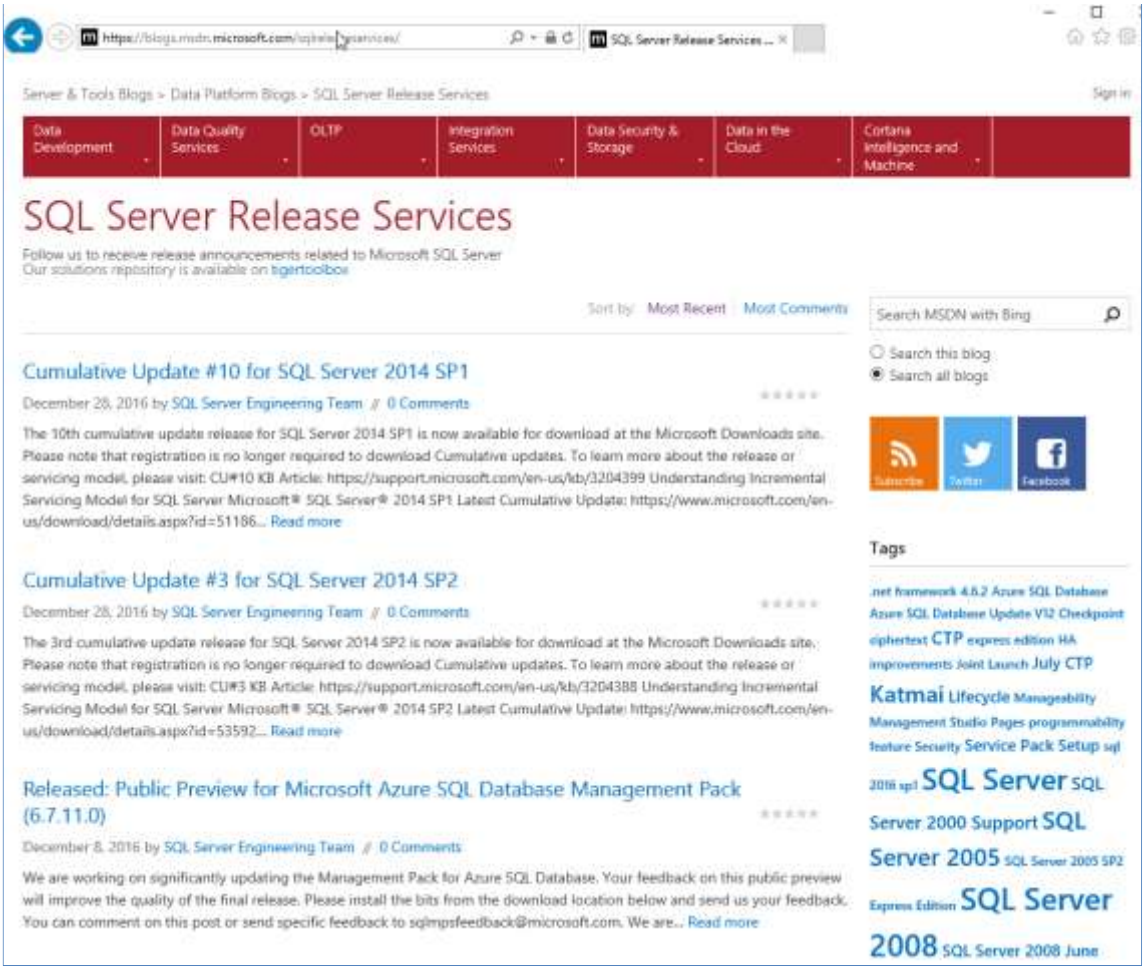
SP1＋CU1 を適用した場合は、SQL Server のビルド番号は「**13.0.4411.0**」に変わります。



## 最新の修正プログラムに関する情報 ～Release Services ブログ～

SQL Server に関する最新版の修正プログラムは、以下の SQL Server Release Services ブログで紹介されているので、こちらも定期的にチェックしておくことをお勧めします。

SQL Server Release Services
http://blogs.msdn.microsoft.com/sqlreleaseservices/

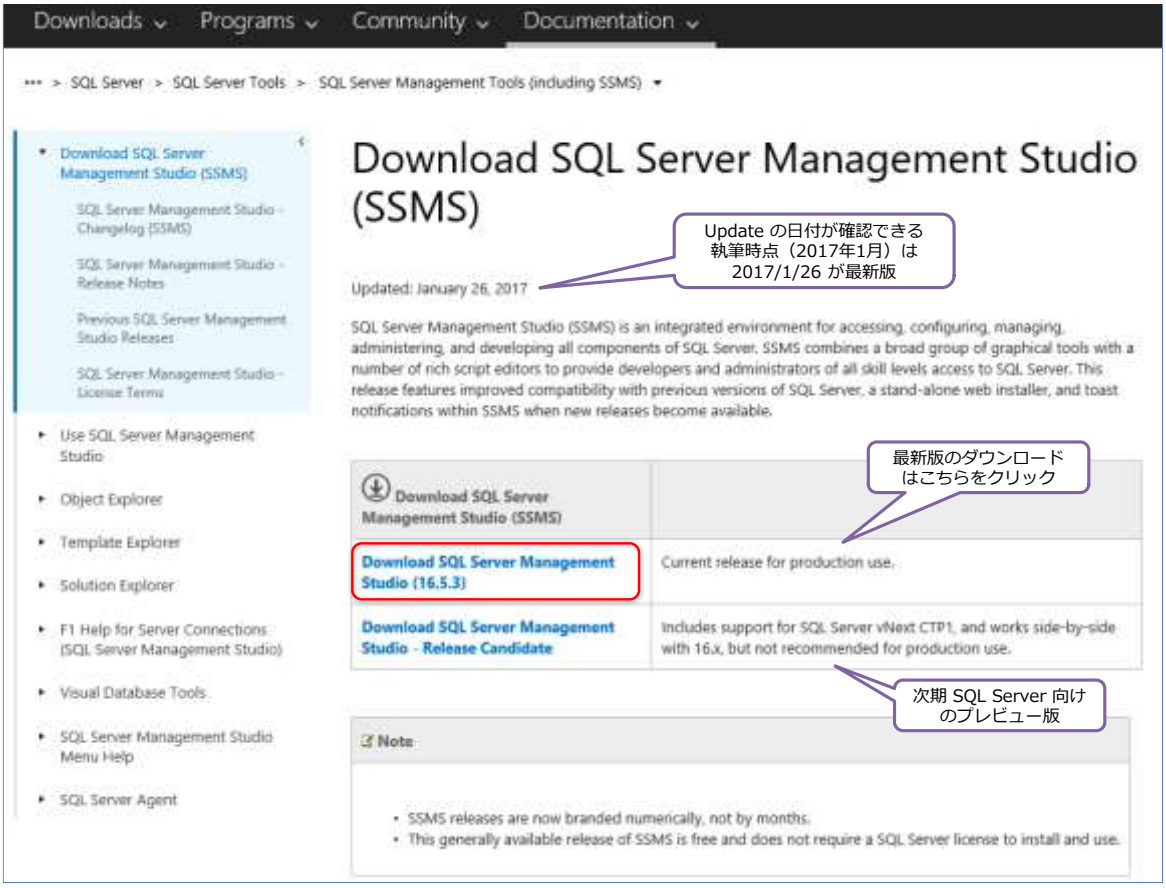


## 3.7 最新版の Management Studio のダウンロードとインストール

SQL Server 2016 からは、Management Studio（SQL Server の管理ツール）がダウンロード版の提供のみに変更されました。これは、クラウド（Microsoft Azure の提供する各種サービス）のアップデートにイチ早く対応したり、製品へのフィードバックをイチ早く反映させるためです。執筆時点（2017 年 1 月）では、2016 年 6 月に提供された RTM（製品）版を皮切りに、7 月、8 月、9 月に 2 回、10 月、12 月、2017 年 1 月と、約 1 ヶ月ごとに Update 版の Management Studio が提供されています（今後も定期的な間隔での最新版の提供が予定されています）。

最新版の Management Studio は、次の URL からダウンロードすることができます。

Management Studio の最新版のダウンロード
http://msdn.microsoft.com/en-us/library/mt238290.aspx




この Web サイトは、英語のページですが、日本語環境であれば、日本語版のインストーラーである「**SSMS-Setup-JPN.exe**」ファイルのダウンロードが始まるので、「**-JPN**」が付くことを確認しておいてください。

| **Note： Windows Server 2012 の場合は KB 2862966 を適用しておく必要あり** |
| --- |
| OS に **Windows Server 2012**（R2 ではない 2012）を利用している場合には、Management Studio をインストールするためには、事前に **KB 2862966** の Security Update を適用しておく必要があります。 |

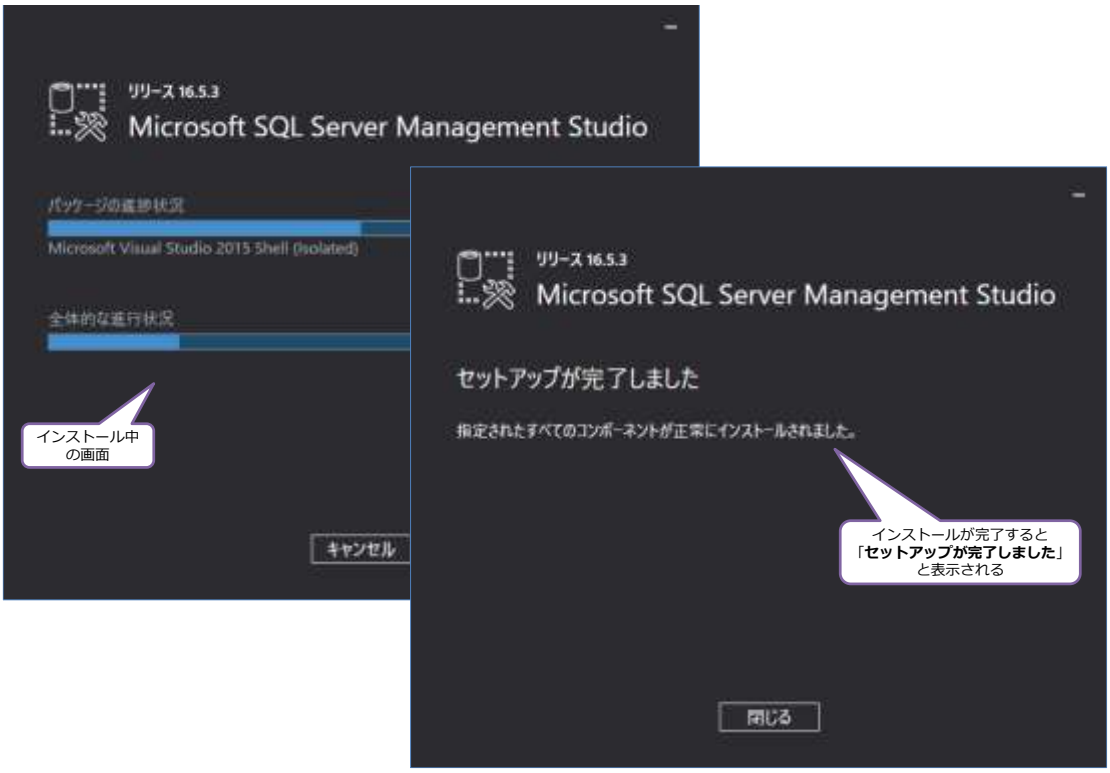
**KB 2862966** は、次の URL からダウンロードすることができます。

Windows Server 2012 用セキュリティ更新プログラム (KB2862966)
http://www.microsoft.com/ja-JP/download/details.aspx?id=39884


## Management Studio のインストール

Management Studio をインストールするには、ダウンロードした **SSMS-Setup-JPN.exe** ファイルをダブルクリックして実行します。次のようにインストーラーが起動したら、[**インストール**] ボタンをクリックします。

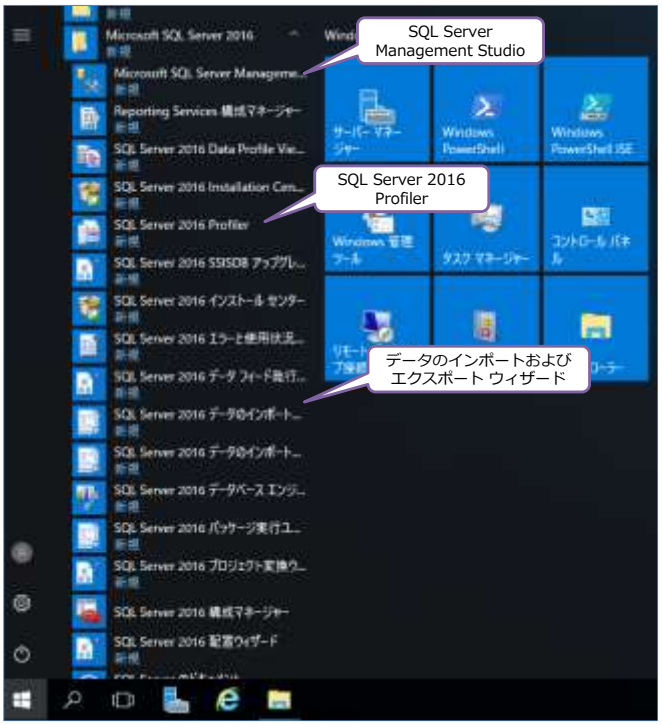
Management Studio のインストール中は、次のように進行状況が表示されます。



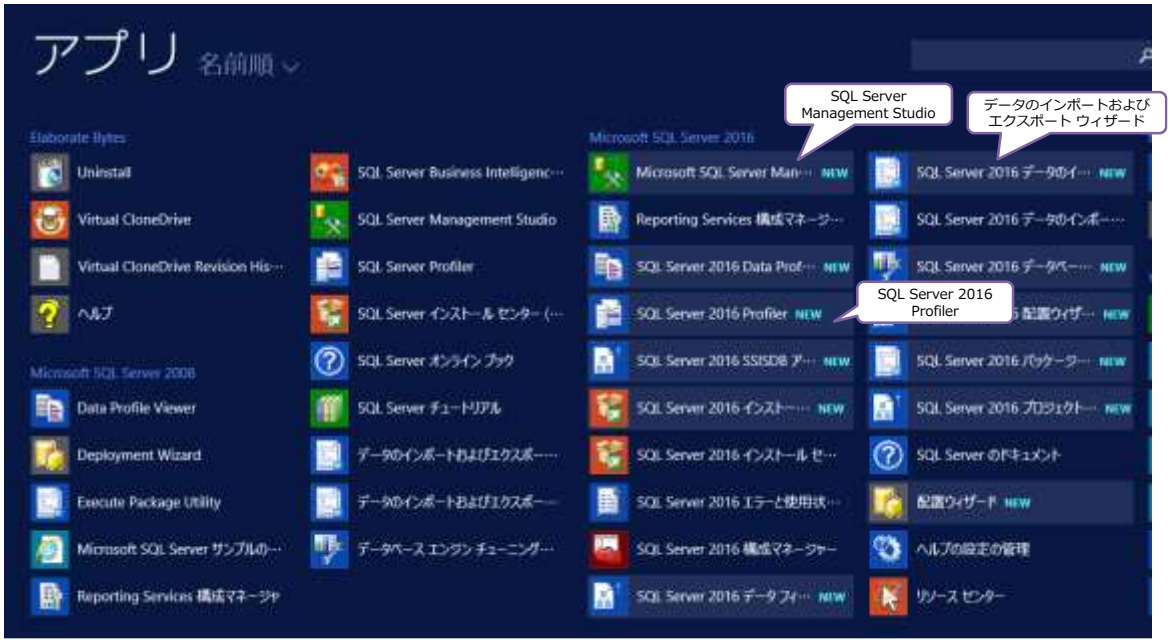

「**セットアップが完了しました**」と表示されれば、Management Studio のインストールが完了です（環境によっては、完了後に再起動が促される場合があります）

Management Studio のインストールが完了すると、次のように［**スタート**］メニューに Management Studio や Profiler（プロファイラー）、インポート／エクスポート ウィザードなどが追加されます。

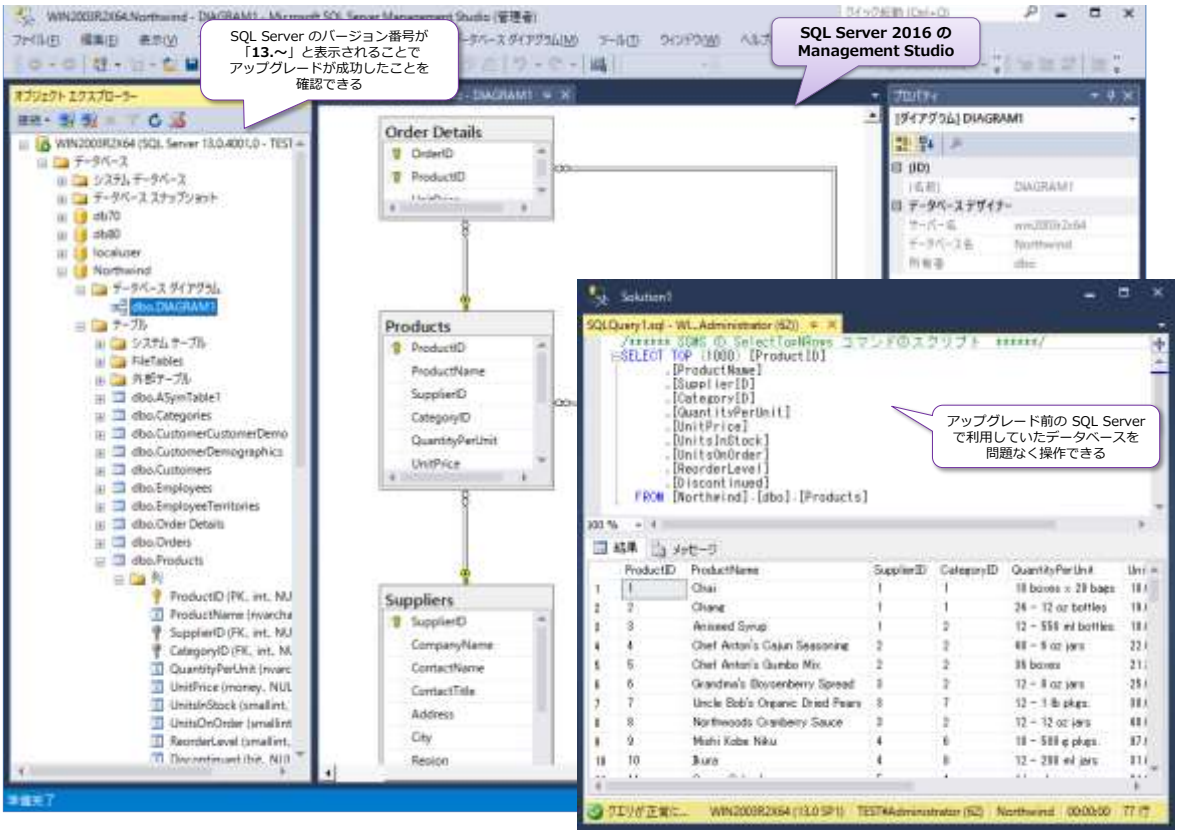
Windows Server 2016 環境の場合の「スタート」メニュー



Windows Server 2012 R2 環境の場合の「スタート」画面



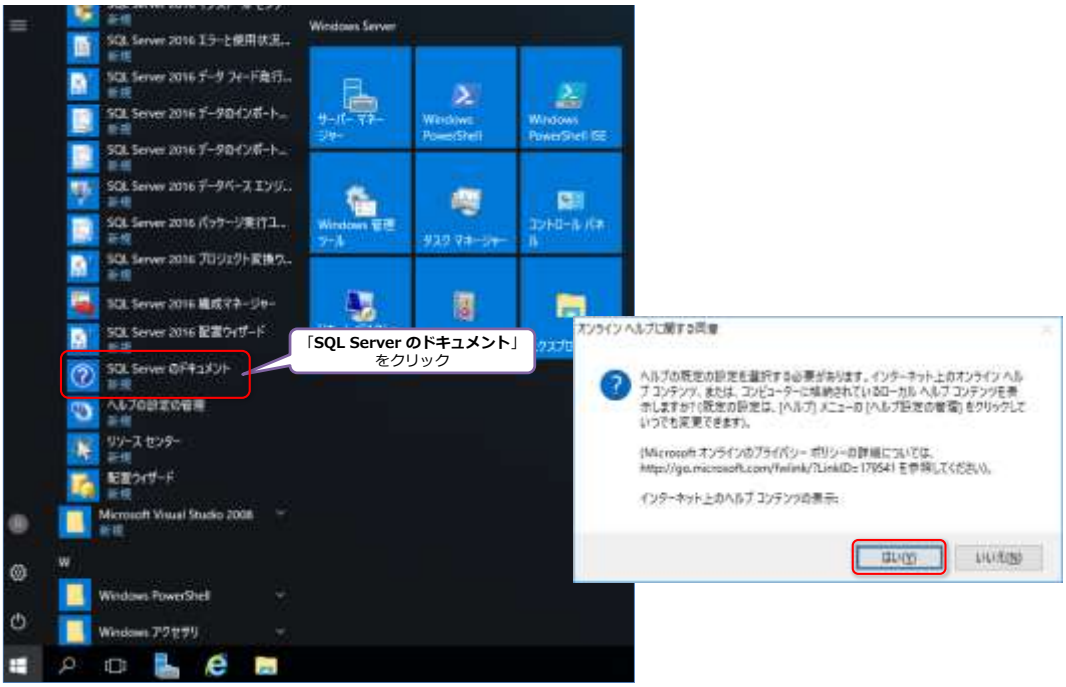

Management Studio を起動すると、アップグレード前の環境と同じようにデータベースが利用できることを確認できます。



## 3.8 オンライン ブック（Books Online）の参照方法

**SQL Server のヘルプ**である「**オンライン ブック**」（BOL : Books Online）は、SQL Server 2016 からは、ローカル マシンにはインストールされず、オンライン版のみの提供に変わりました。

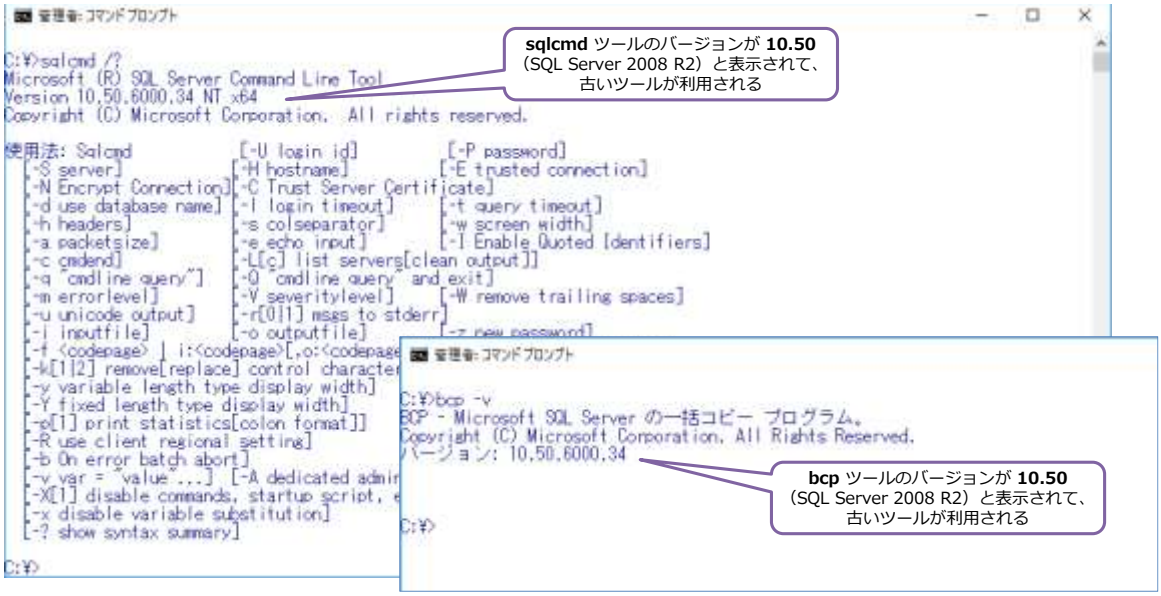
オンライン ブックを起動するには、スタート メニューの［**Microsoft SQL Server 2016**］グループから［**SQL Server のドキュメント**］をクリックします（Windows Server 2012／2012 R2 の場合は、［**スタート**］画面から起動します）。




［**オンライン ヘルプに関する同意**］ダイアログが表示されたら、内容を確認した上で［**はい**］をクリックすることで、オンライン版のヘルプを参照できるようになります。

## 3.9 コマンドライン ツール（sqlcmd、bcp）のパスについて

SQL Server 2016 へアップグレードすると、**sqlcmd** や **bcp** ツールなどの**コマンドライン ツール**は、既定では旧バージョンのツールが実行されます。これは次のような状況です。

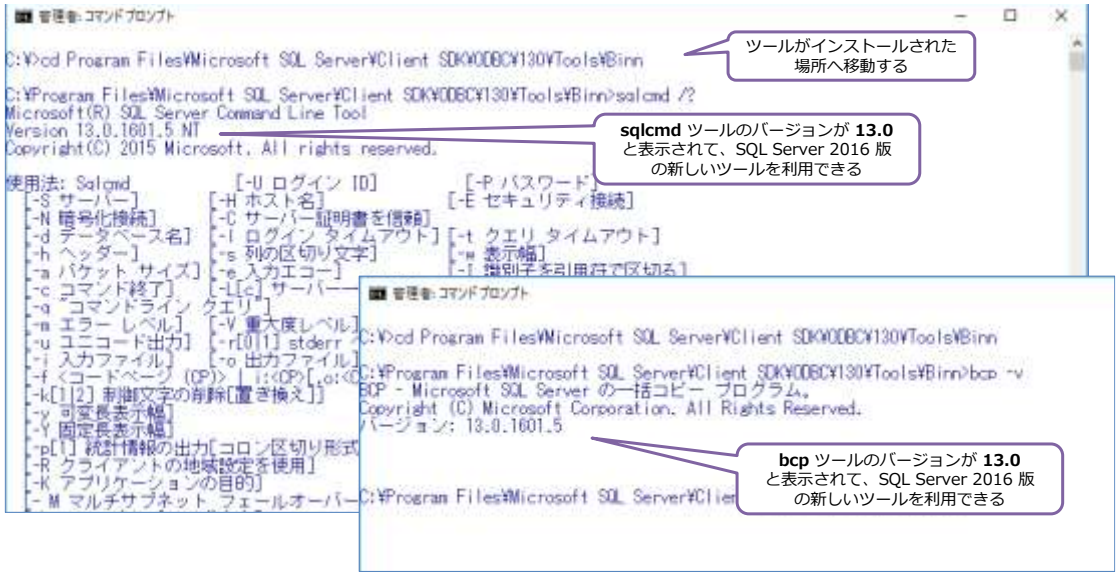



パスを指定しないでコマンドライン ツールを実行すると、アップグレード前のバージョンのツールが実行されてしまいます。

**sqlcmd** と **bcp** ツールの SQL Server 2016 版は、次のパスにインストールされています。

**C:¥Program Files¥Microsoft SQL Server¥Client SDK¥ODBC¥130¥Tools¥Binn**

したがって、このパスへ移動して（**cd** コマンドで移動して）、ツールを実行すれば、SQL Server 2016 のツールを利用することができます。

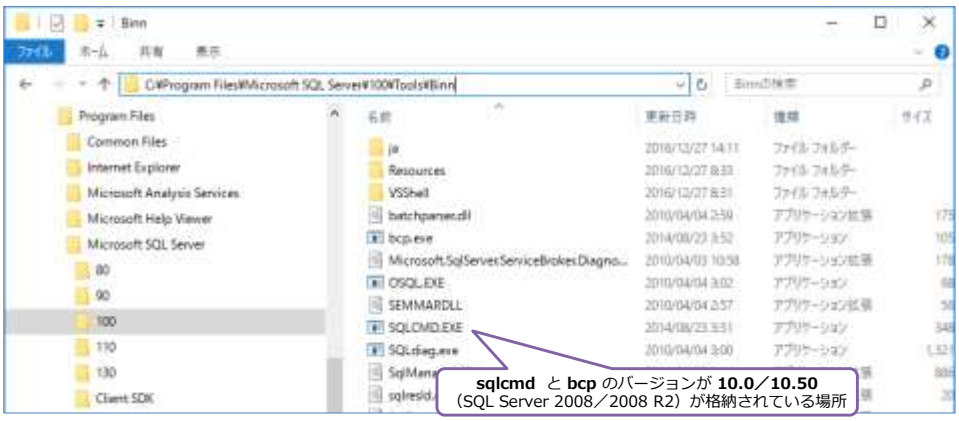


**Note： PATH（環境変数）を変更する**

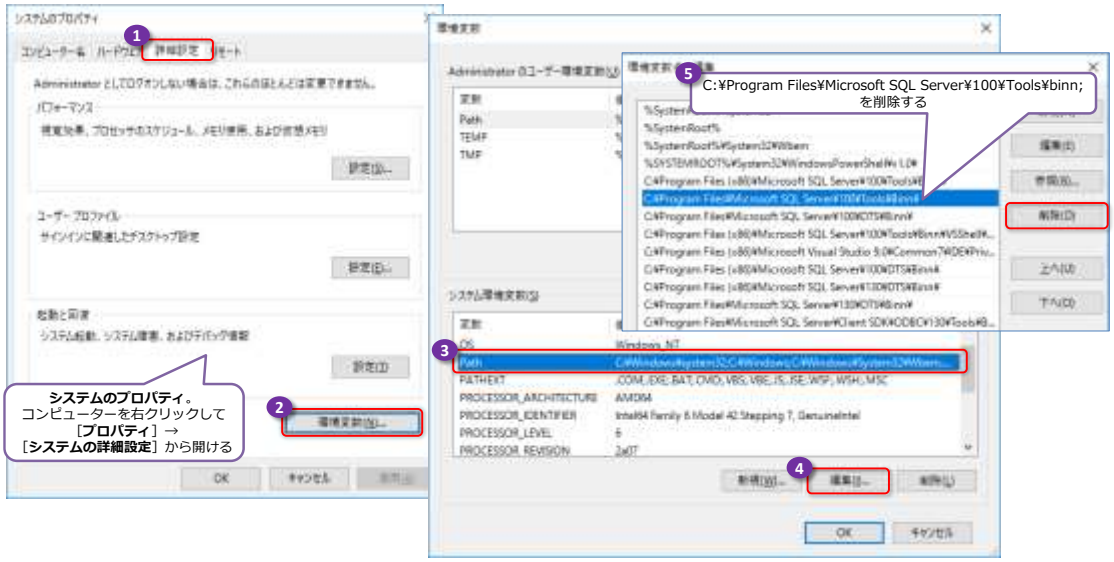
古いバージョンのコマンドライン ツールが実行されてしまう理由は、環境変数の「**PATH**」の設定にあるので、これを変更してしまうという方法もあります。


SQL Server 2008／2008 R2 からのアップグレードの場合は、**sqlcmd** と **bcp** ツールの古いバージョンが以下のパスに格納されています。

**C:¥Program Files¥Microsoft SQL Server¥100¥Tools¥binn**




したがって、このパスを環境変数（システム環境変数）の「**PATH**」から削除すれば、パスを省略した場合に、SQL Server 2016 のツールを実行できるようになります。

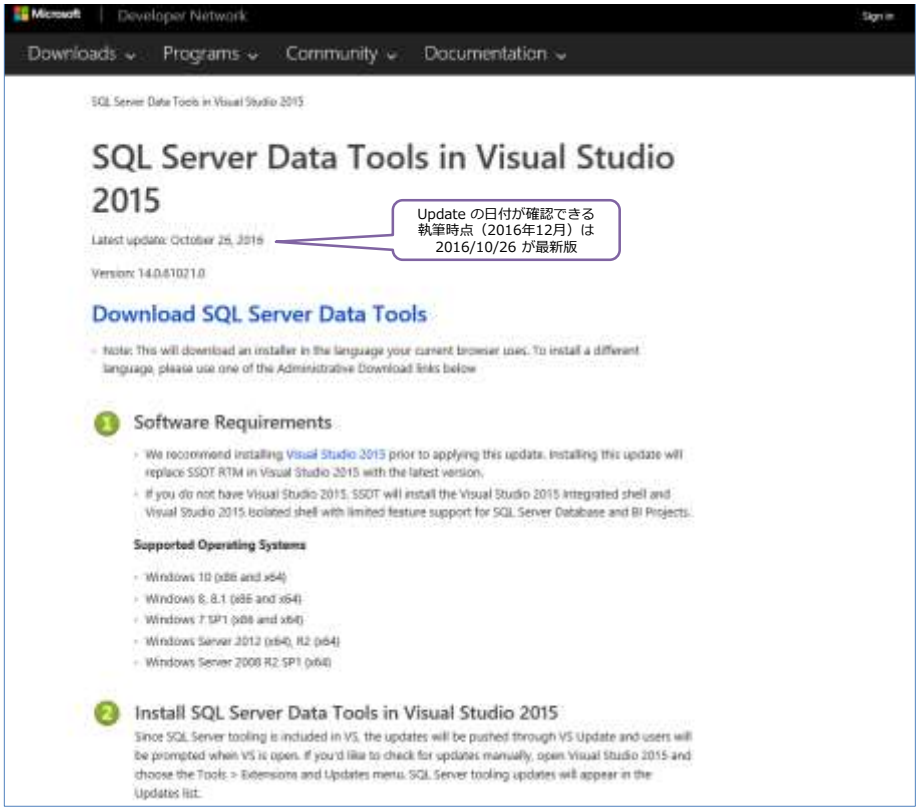


## 3.10 SQL Server Data Tools（SSDT）のインストール（旧 BIDS）

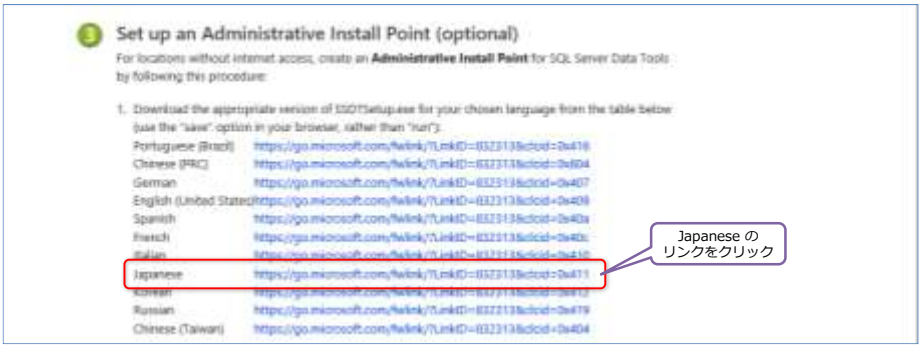
SQL Server 2008 や 2008 R2 での **Business Intelligence Development Studio**（**BIDS**）ツールは、SQL Server 2012 からは **SQL Server Data Tools**（**SSDT**）に名称変更されました。**SSDT** は、SQL Server 2012／2014 のときには、.NET 開発者のための **SSDT** と BI プロジェクトのための（従来の BIDS に相当する）**SSDT-BI**（**SQL Server Data Tools - Business Intelligence**）の 2 種類が存在していましたが、SQL Server 2016 からは、この 2 つが一本化されて、「**SSDT**」のみの提供に変わりました。

### SSDT の最新版のダウンロードとインストール

SSDT の最新版は、次の URL からダウンロードすることができます。
http://msdn.microsoft.com/en-us/mt186501




:

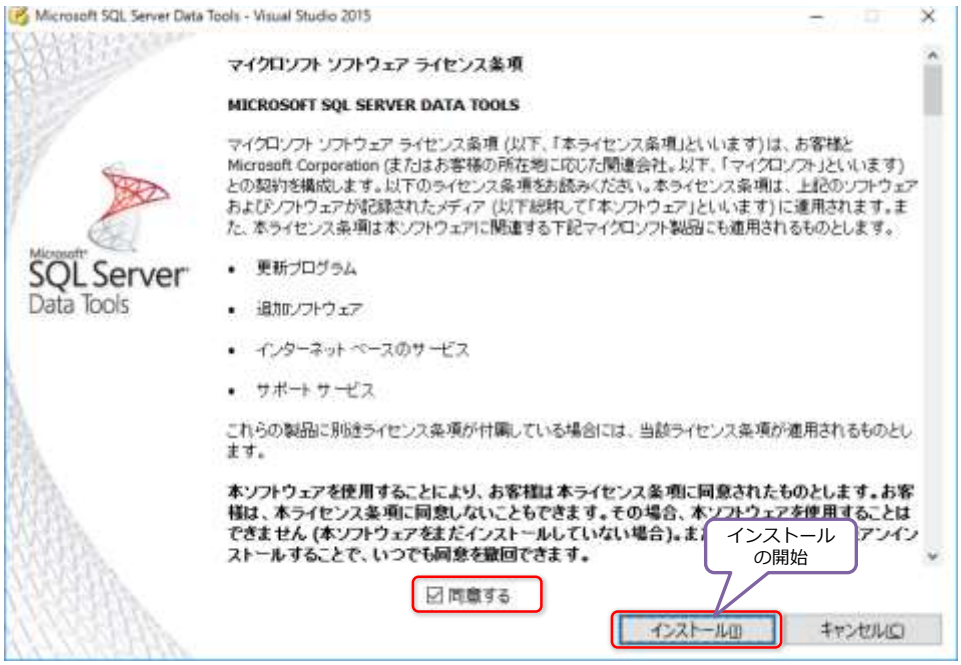



この Web サイトは英語のページですが、**Japanese** のリンクをクリックすれば、日本語版のイン

ストーラー（**SSDTSetup.exe**）をダウンロードすることができます。

ダウンロードした **SSDTSetup.exe** ファイルをダブル クリックすると、次のようにインストールを開始することができます。


次の［**ライセンス条項**］ページでは、ライセンス条項の内容を確認した上で、［**同意する**］をチェックして、［**インストール**］ボタンをクリックします。

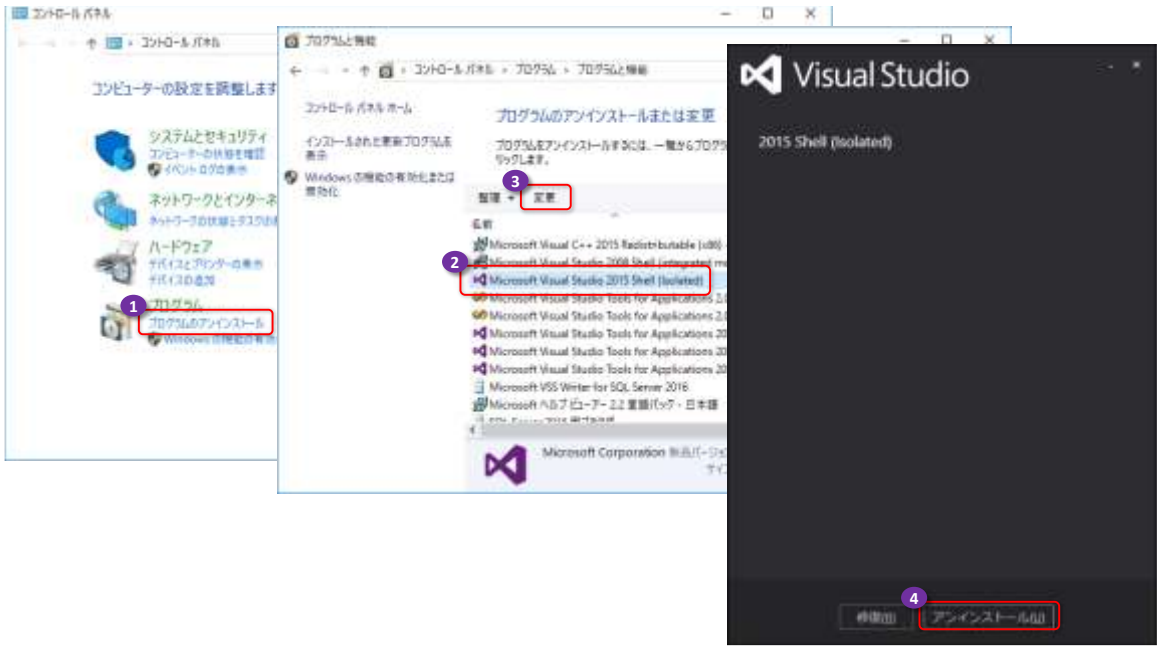



これで SSDT のインストールが開始されて、インストール中は、次のように進行状況が表示されます（最初にファイルのダウンロードが行われるので、数十分ぐらい時間がかかります）。


「**セットアップに成功しました**」と表示されれば、SSDT のインストールが完了です（環境によっては、完了後に再起動が促される場合があります）

## SSDT のインストールに失敗する場合

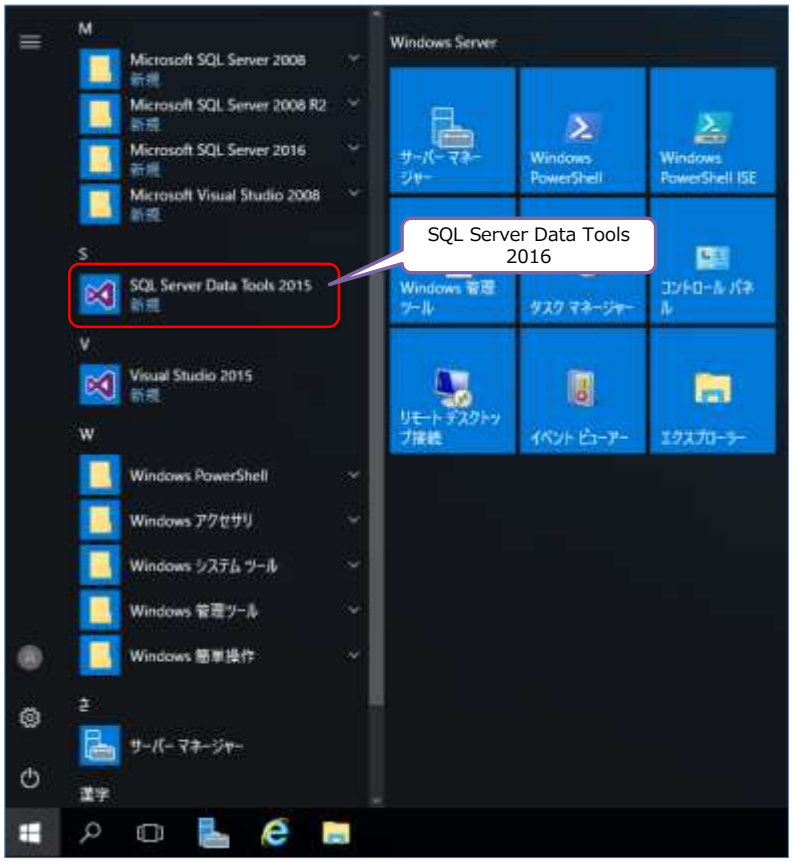
なお、SSDT のインストールに失敗してしまった場合には、次のように［**コントロール パネル**］の［**プログラムのアンインストール**］から［**Microsoft Visual Studio 2015 Shell (Isolated)**］を選択して、［**変更**］ボタンをクリックし、Visual Studio 2015 Shell (Isolated) を**アンインストール**してから、再度セットアップを実行してみてください。



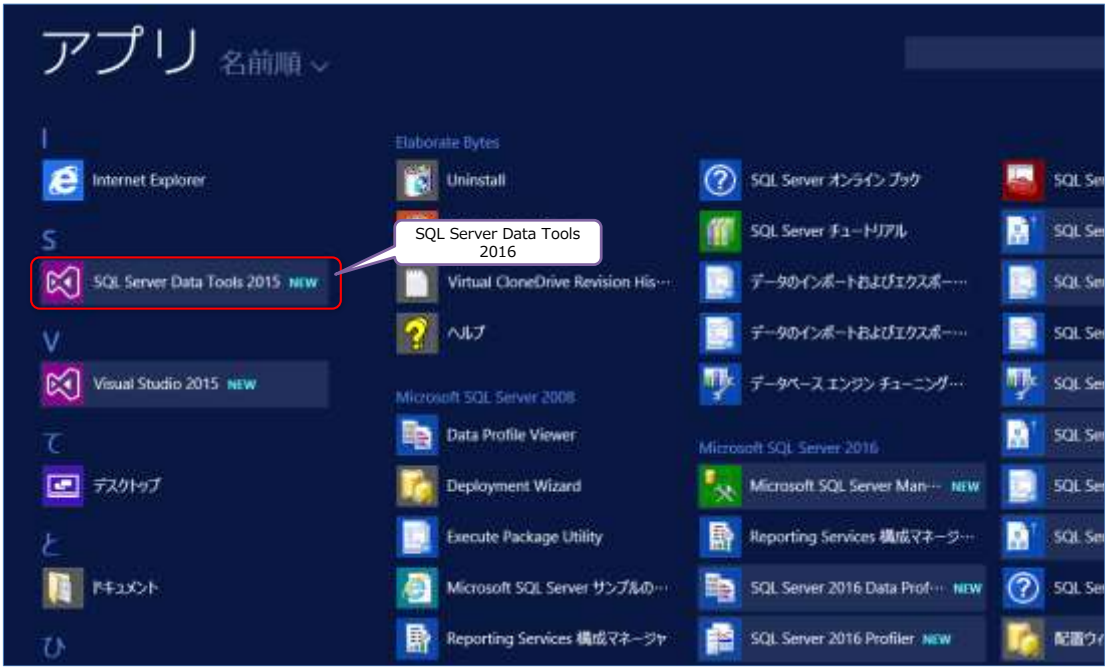
## SSDT の起動 ～SQL Server Data Tools 2016～

SSDT のインストールが完了した後は、[**スタート**] メニューに [**SQL Server Data Tools 2016**] が追加されるので、これをクリックして起動することができます。

Windows Server 2016 環境の場合の「スタート」メニュー



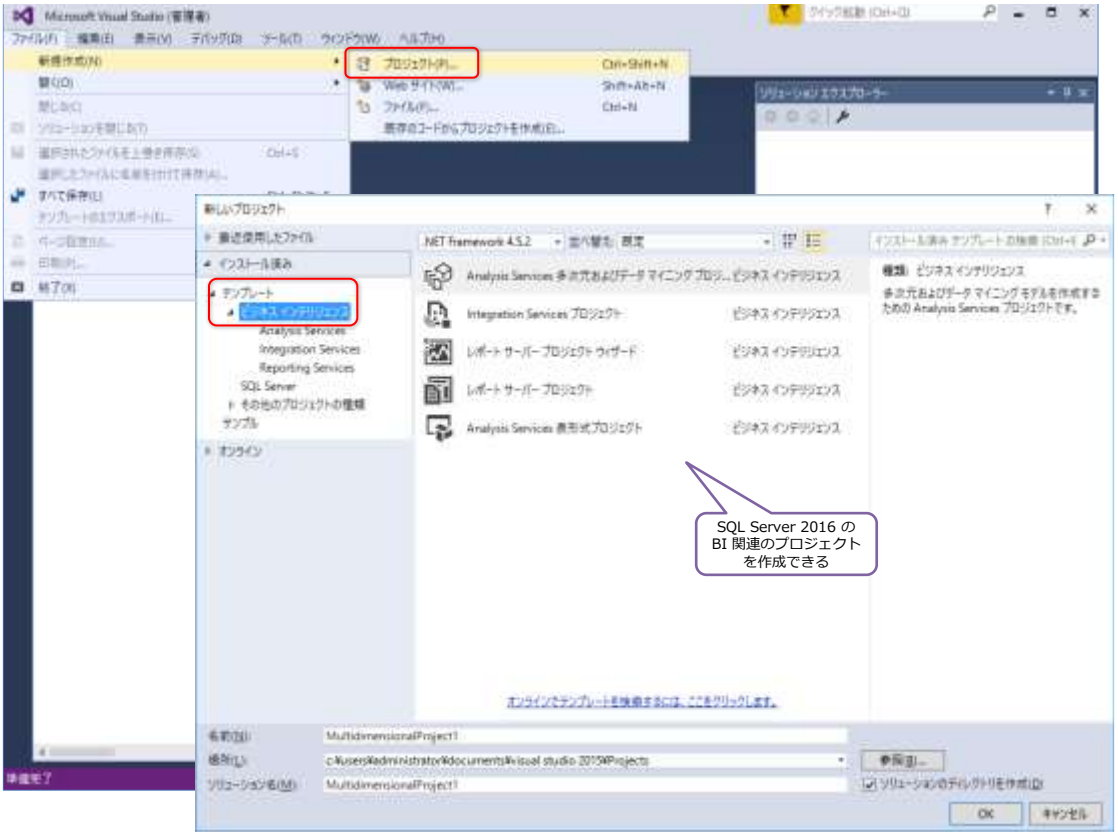

Windows Server 2012 R2 環境の場合の「スタート」画面



SSDT は、Visual Studio 2015 のシェルに統合されているので、起動すると、次のように Visual Studio 2015 が起動されます。



Visual Studio 2015（SSDT）が起動したら、次のように［**ファイル**］メニューの［**新規作成**］から［**プロジェクト**］をクリックして、［**テンプレート**］で［**ビジネス インテリジェンス**］を選択することで、SQL Server 2016 の BI 関連のプロジェクト（Analysis Services や Integration Services、Reporting Services など）を作成できるようになります。

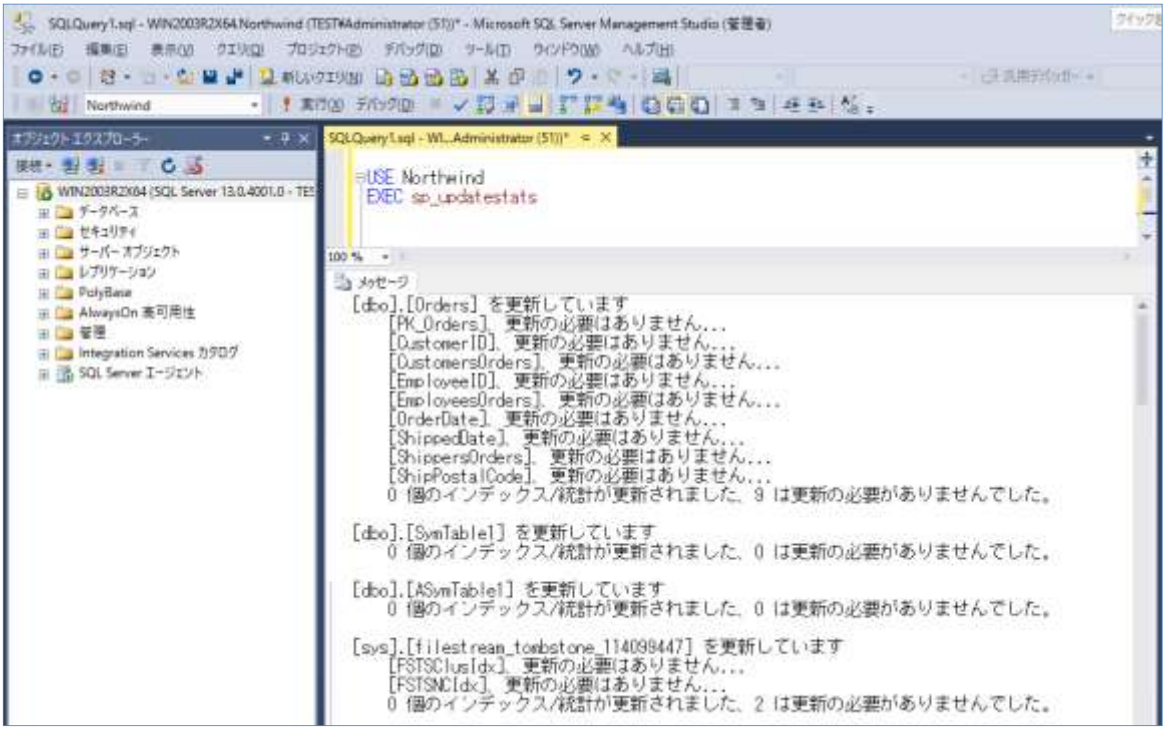


## 3.11 統計の更新

SQL Server 2016 へアップグレードをした後には、**統計**（Statistics）を更新しておくことがお勧めになります。**統計**は、クエリ オプティマイザーが最適な実行プランを選択するために、内部利用しているデータの分布状況です。アップグレード後にも、最適な実行プランが選択されるようにするためには、統計を最新の状態へ更新しておくようにします。

統計を更新するには、次のように **sp_updatestats** システム ストアド プロシージャを実行します。

```
USE データベース名
EXEC sp_updatestats
```




統計の更新は、テーブル サイズが大きい場合には、実行に時間がかかる場合があります。

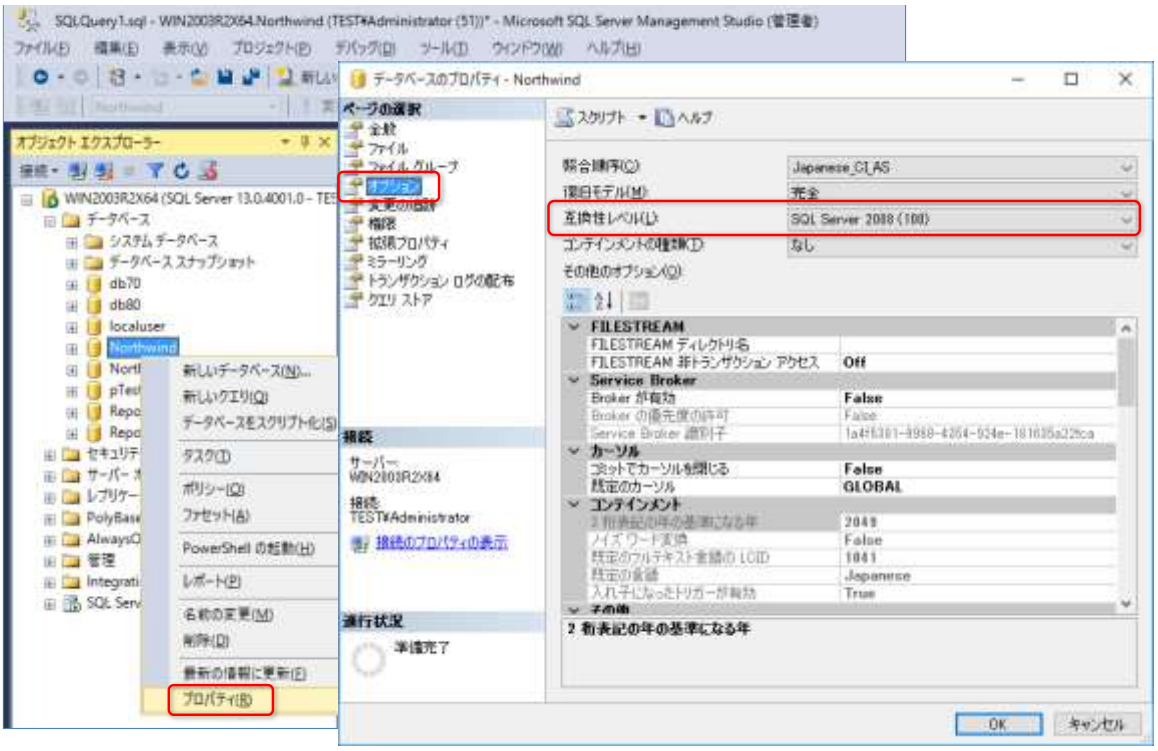
統計の更新後は、実際に利用しているクエリの「**推定実行プラン**」や、可能であればクエリ実行時間などを確認して、アップグレード後にも問題なく、クエリが処理できているかどうかを確認しておくことをお勧めします（後述の**クエリ ストア**機能を利用して、実行プランを確認することもできます）。

以前のバージョンでの実行プランと同じ実行プランを利用したい場合には、「**プランガイド**」機能を利用するという方法もあります。また、SQL Server 2016 からは、後述の**クエリ ストア**機能を利用することで、実行プランの比較とプラン強制（プランガイドと同じ機能）を行うこともできます。これについてはデータベースの互換性レベルのところで詳しく説明します。

## 3.12 データベースの互換性レベルを 130 へ上げる（オプション）

アップグレード前の SQL Server で**データベースの互換性レベル**が「**90**」（SQL Server 2005 レベル）以下であったものは、**SQL Server 2016** にアップグレードした後に、自動的に「**100**」（SQL Server 2008 に相当するレベル）に変更されています。

データベースの互換性レベルを確認するには、次のようにユーザー データベースの［**プロパティ**］を開いて、［**オプション**］ページを表示します。




SQL Server 2016 では、互換性レベル **100** 以降がサポートされています。

### SQL Server 2016 で利用できるデータベースの互換性レベル

**SQL Server 2016** で利用できる**データベースの互換性レベル**は、次の 4 つです。

- **100**（SQL Server 2008／2008 R2 レベル）
- **110**（SQL Server 2012 レベル）
- **120**（SQL Server 2014 レベル）
- **130**（SQL Server 2016 レベル）

SQL Server 2008／2008 R2 をアップグレードすると **100**、SQL Server 2012 をアップグレードすると **110**、SQL Server 2014 をアップグレードすると **120** が利用されます。

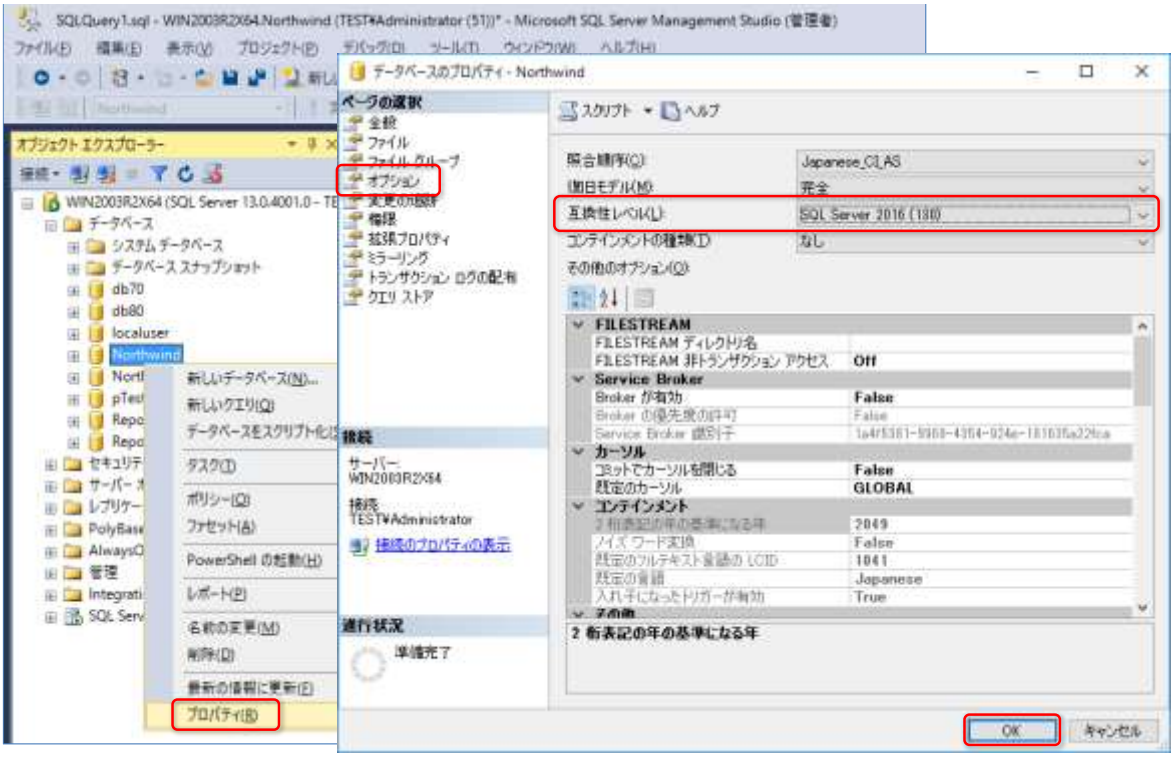
互換性レベルは、**100** や **110**、**120** のまま利用しても問題はありませんが、より良い性能を考慮するのであれば、**130**（SQL Server 2016 レベル）に上げておくことがお勧めになります。**130**

に変更すれば、SQL Server 2016 からの新機能である「**INSERT .. SELECT の並列処理**」や「**列ストア インデックスでのバッチ モードの性能向上**」、「**インメモリ OLTP での並列処理**」などを利用することができるからです。

互換性レベルの **100**、**110**、**120** および **130** では、細かい修正はありますが（詳しくは後述します）、基本的な Transact-SQL ステートメントであれば同じように利用できるので、多くの環境では問題が出ないと思います（弊社のお客様では、今のところ 4 社ほど SQL Server 2008 から SQL Server 2016 へのデータベースの移行を試していますが、何の問題もなく、ストアド プロシージャやアプリケーションを実行することができています）。

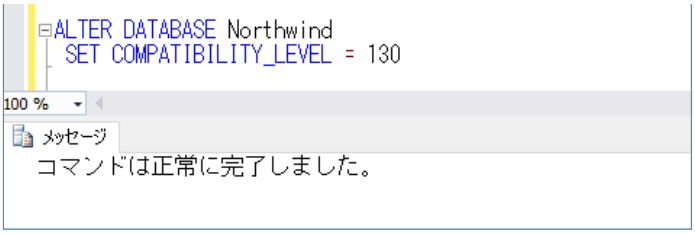
## データベースの互換性レベルを 130 へ上げる

互換性レベルを 130 へ上げるには、次のようにデータベースのプロパティの［**オプション**］ページで、［**互換性レベル**］に［**SQL Server 2016 (130)**］を選択して、［**OK**］ボタンをクリックします（互換性レベルを変更する前に、現在の実行プランの保存や、性能チェックを行いたい場合は、次の項で説明する**クエリストア**機能を利用してから、互換性レベルを変更してください）。




Transact-SQL ステートメントを利用して互換性レベルを変更したい場合には、次のように **ALTER DATABASE** ステートメントを実行します。

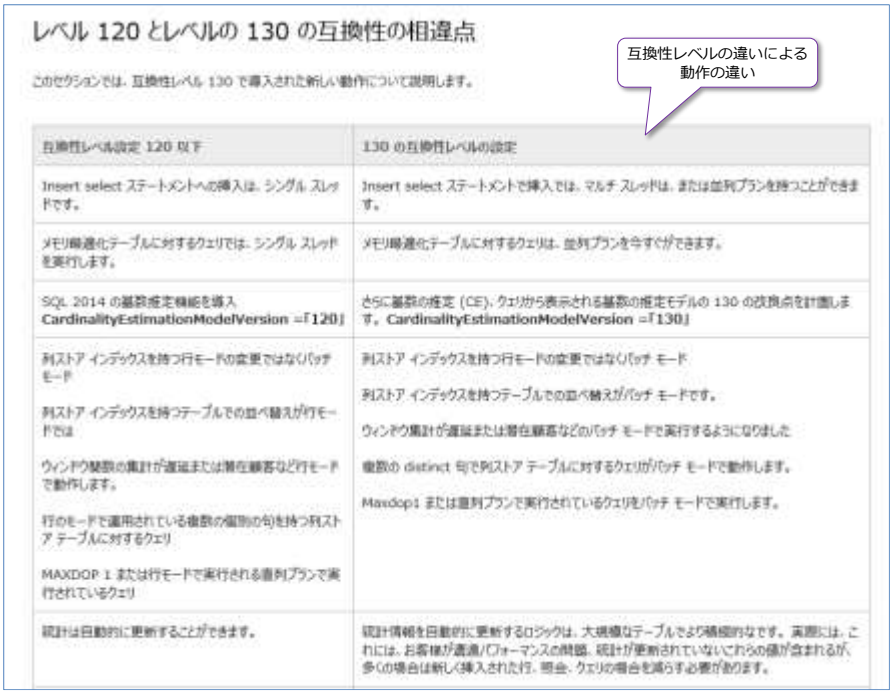
```
ALTER DATABASE データベース名
 SET COMPATIBILITY_LEVEL = 130
```




## 互換性レベルの違い

互換性レベルの違い（動作に変更のあったステートメント）に関しては、オンライン ブックの以下のトピックに記載されています。

ALTER DATABASE 互換性レベル
http://msdn.microsoft.com/ja-jp/library/bb510680.aspx

このトピックの主なものを取り上げると、次のようになります（関係ありそうな部分は、オンライン ブックを参照しておくことをお勧めします）。

**130 との相違点（120 以下と 130 との差）**

- **INSERT .. SELECT** が並列操作になるかどうか（130 では並列になる）
- **インメモリ OLTP** のメモリ最適化テーブルを並列処理できるかどうか（130 では並列になる）
- **列ストア インデックス**の動作の違い（性能に関する動作の違いが発生。130 にすることで列ストア インデックスの性能が向上）
- SQL Server 2014 から提供された**基数見積**（CE: Cardinality Estimation）を利用して、実行プランを選択する（これについては、データベースごとに設定を変更することができるようになったので、詳しくは後述）
- 統計の自動更新の動作の違い
- **暗号化アルゴリズム**は SHA2_256 と SHA2_512 のみが許可される（MD2 と MD4、MD5、SHA、SHA1 は許可されない）

**120 との相違点（110 以下と 120 との差）**

- SQL Server 2014 からの**新しいクエリ オプティマイザー**が利用されるかどうか（120 では利用される、100／110 では利用されない）
- **date 型**の文字列値で、言語設定が無視されるかどうか（120 では無視しない）
- **再帰 CTE**（共通テーブル式）で重複する列名を使用できるかどうか（120 では使用できない）
- **SELECT INTO** が並列操作になるかどうか（120 では並列になる）

**110 との相違点（100 以下と 110 以上との差）**

- **SQL CLR**（**CLR 統合**）の動作の違い（100 と 110 以降で動作が異なる。Unicode 5.1 のサポートや、System.TimeSpan 値の扱いなど）
- **XQuery 関数**の string-length および substring の動作が 100 と 110 以降で異なる（サロゲート ペアの場合の文字カウントの仕方が異なる）
- **PIVOT** が**再帰 CTE** で許可されるかどうか（110 以降では許可されない）
- **RC4** または **RC4_128** での暗号化がサポートされるかどうか（110 以降ではサポートされない）
- **time** および **datetime2** データ型の**計算列**で、CAST または CONVERT で日付／時刻スタイルを省略したときの動作の違い（110 以降では既定のスタイルが 121 になる）
- **パーティション ビュー**で参照するリモート テーブルが **smalldatetime** 型の列が datetime にマップされるかどうか（110 以降では、smalldatetime にマップされる）
- **SOUNDEX 関数**では動作の違い（110 以降では、大文字の H または大文字の W があ

る場合に、右側の子音は無視されるなど）

**100 との相違点（90 以下と 100 以上との差）**

- **複数ステートメントのテーブル値関数**が作成されるときの QUOTED_IDENTIFER 設定の違い（100 以降ではセッションの設定が利用される）
- **パーティション関数**での datetime および smalldatetime リテラルが言語設定を利用するかどうか（100 以降では利用する）
- INSERT または SELECT INTO ステートメントでの **FOR BROWSE 句**（100 以降では許可される）
- **OUTPUT 句**でのフルテキスト述語（100 以降では使用できない）
- フルテキスト インデックスでの **STOPLIST**（100 以降でサポートされる）
- **MERGE** が予約されたキーワードかどうか（100 以降では予約されたキーワードになる）
- INSERT ステートメントでの **<dml_table_source>**、**OUTPUT 句**（100 以降でサポートされる）
- **DBCC CHECKDB** での **NOINDEX** の動作の違い
- データ操作言語 (DML) ステートメントで **OUTPUT 句**を使用した場合の実行時エラーの動作の違い
- **CUBE** および **ROLLUP** が予約されたキーワードかどうか（100 以降では GROUP BY 句内では予約されたキーワードになる）
- **XML** の **anyType 型**が厳密な検証かどうか（100 以降では緩やかな検証）
- 特殊な属性 **xsi:nil** および **xsi:type** が変更可能かどうか
- **XML** 定数文字列値を **datetime** 型に変換するユーザー定義関数が "**決定的**" とマークされるかどうか
- **XML** の union 型と list 型が完全にサポートされているかどうか
- **xQuery** メソッドでの SET オプションが検証されるかどうか
- **XML** 属性値で行末文字（復帰と改行）を含む場合の動作の違い
- ROWGUIDCOL または IDENTITY を**制約**として指定できるかどうか（100 以降では指定できない）
- UPDATE T1 SET @v = column_name = <expression> のような**双方向代入**を行った場合の動作の違い
- 変数代入のステートメントを **UNION** で連結する場合の動作の違い
- **ODBC 関数 {fn CONVERT()}** での日付への変換時の動作の違い
- **ODBC 関数 {fn CURDATE()}** の動作の違い
- DATEPART 関数での**無効な datetime リテラル**が与えられた場合の動作（例えば、100 以降では、**'2007/05-30'** はエラー 241 が返り、90 では正常にコンパイルされる）

いずれも細かい修正点ばかりで、XML や再帰 CTE、OUTPUT 句、フルテキスト検索、ODBC など、関係ないものが多いのではないでしょうか。弊社のお客様のデータベースでは、上記に該当するステートメントはほとんどなく、**INSERT .. SELECT** および **SELECT INTO の並列処理**のみが該当しました。INSERT .. SELECT と SELECT INTO のパラレル処理に関しては、非常に便利な機能で、これは互換性レベルを 130 にしないと利用できないので、ぜひ 130 へ上げることを検討してみることをお勧めします。また、130 レベルを利用することで、インメモリ OLTP と列ストア インデックスの性能向上を実現することができるので、130 レベルを利用することをお勧めします。

## Data Migration Assistant で事前に互換性レベルの違いを簡単に確認可能

第 2 章で紹介した **Data Migration Assistant**(旧アップグレード アドバイザー)を利用すれば、アップグレード前の SQL Server に対して、SQL Server 2016 にアップグレードしたときの互換性レベルの問題点（130 や 120 を利用したときにどういった問題が発生するのか）を簡単に見つけることができます。

## クエリ ストアで互換性レベルの違いを確認

詳しくは次項で説明しますが、互換性レベルの違いについては、SQL Server 2016 から提供された「**クエリ ストア**」機能を利用することで、期待した動作になるかどうかをチェックできるので、お勧めです。特に互換性レベルを 130 に上げたことによる性能向上や予期せぬ性能低下（想定外の実行プランが選択されたことによる性能低下など）を確認することができます。

## 互換性レベル 80（SQL Server 2000 レベル）を利用していた場合

アップグレード前の SQL Server で、**互換性レベル 80**（**SQL Server 2000** との互換性があるレベル）を利用していたデータベースは、SQL Server 2016 にアップグレードしたタイミングで、自動的に **100**（SQL Server 2008 レベル）に変更されます。

この場合は、互換性レベル **80** では利用できていたものが、**100** 以降では利用できないということが発生します。例えば、外部結合演算子の「***=**」と「**=***」は、80 では利用できますが、100 以降では利用できないなどです。**80**（および SQL Server 2000）に関しては、SQL Server 2012 から未サポートになっているので、オンライン ブックの以下のトピックが参考になります。

SQL Server 2016 で廃止されたデータベース エンジンの機能
- SQL Server 2012 で廃止されたデータベース エンジンの機能
http://msdn.microsoft.com/ja-jp/library/ms144262.aspx


SQL Server 2008 R2 のオンライン ブックでの

ALTER DATABASE 互換性レベルの「**互換性レベル 80 とレベル 90 の相違点**」
http://msdn.microsoft.com/ja-jp/library/bb510680(v=sql.105).aspx

互換性レベル 80 とレベル 90 の相違点

ここでは、互換性レベル 90 に導入された新しい動作について説明します。

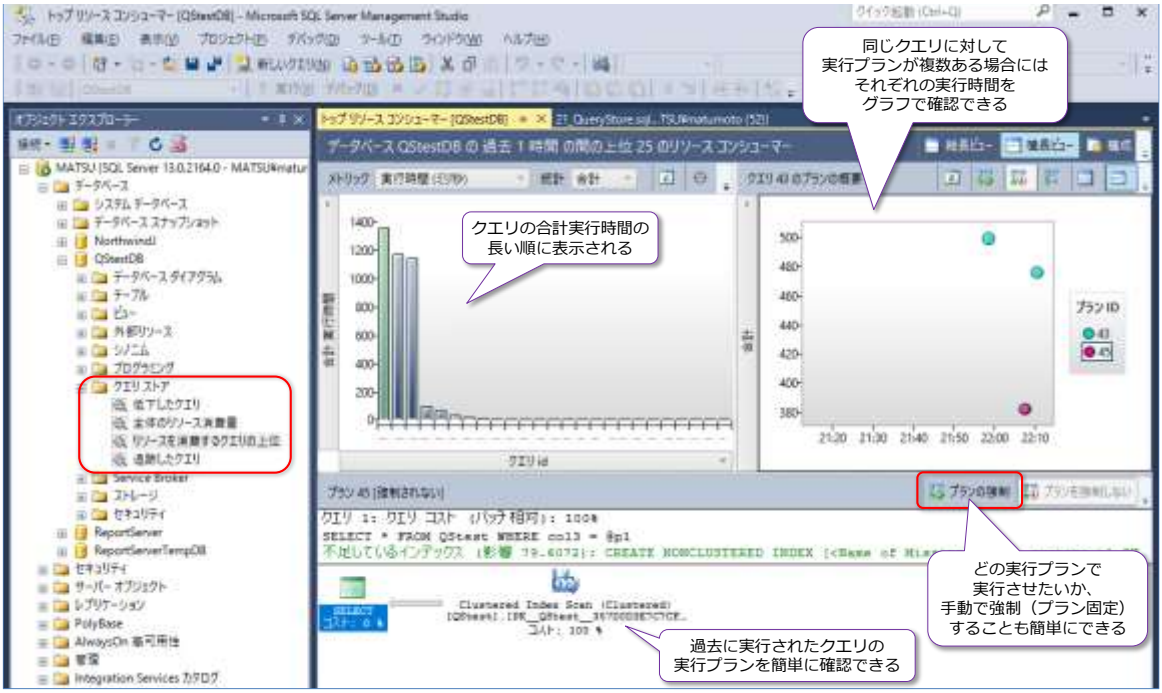
互換性レベル 90 では、動作が次のように変更されます。

SQL Server 2008 R2 のオンライン ブック

| 互換性レベル設定 80 | 互換性レベル設定 90 | 影響度 |
|---|---|---|
| FROM 句のロック ヒントに対して、WITH キーワードは常に省略可能です。 | いくつかの例外を除き、テーブル ヒントは、FROM 句で WITH キーワードを使って指定した場合にのみサポートされます。詳細については、「FROM (Transact-SQL)」を参照してください。 | 高 |
| 警告メッセージでは、外部結合の演算子 *= および =* がサポートされます。 | これらの演算子はサポートされません。OUTER JOIN キーワードを使用してください。 | 高 |
| ORDER BY リストにある列参照を SELECT リスト内に定義されている列にバインドする場合、列のあいまいな部分は無視され、場合によっては列プレフィックスも無視されます。これにより、結果セットが予期しない順序で返されることがあります。<br><br>たとえば、ORDER BY 句に含まれる 1 つの列が 2 つの要素 (<table_alias>.<column>) で表され、この列が SELECT リストの中で列への参照として使用されている場合、ORDER BY 句は受け付けられますが、テーブルの別名は無視されます。次のクエリを考えてみます。<br><br>SELECT c1 = -c1 FROM t_table AS x ORDER BY x.c1<br><br>このクエリを実行するとき、ORDER BY で列のプレフィックスは無視されます。指定したソース列 (x.c1) に対しては期待どおりに並べ替え操作が行われず、代わりにクエリ内で定義されている派生列 c1 に対してソートが実行されます。このクエリの実行プランでは、まず派生列の値が計算され、次にその計算値が並べ替えられます。 | 列のあいまいな部分に関してエラーが発生します。ORDER BY 句に列のプレフィックスが指定されている場合、SELECT リストで定義されている列にバインドするときに、列のプレフィックスは無視されません。<br><br>次のクエリを考えてみます。<br><br>SELECT c1 = -c1 FROM t_table AS x ORDER BY x.c1<br><br>このクエリを実行するとき、ORDER BY 句内の列のプレフィックスは無視されません。指定したソース列 (x.c1) に対し、期待どおりに並べ替え操作が行われます。このクエリの実行プランでは、t_table から返された行を並べ替え操作で整列させた後、SELECT リストで定義されている派生列 c1 の値を計算するようになっています。この場合、結果の順序は SQL Server 2000 で返される結果と異なる可能性があります。SQL Server 2000 と同じ結果の順序にするには、ORDER BY 句の列プレフィックスを削除してください。 | 中 |
| 異なるデータ型に対する UNION の INSERT SELECT では、UNION の各分岐は、INSERT の対象列のデータ型に直接キャストされます。互換性のない型を変換したために自己結合が失敗したとしても、UNION の結果の型へのブランチが変換されることはないため、INSERT SELECT によって UNION が正常に行われます。 | UNION の結果の型は INSERT SELECT とは無関係に派生します。UNION の各ブランチは UNION の結果の型にキャストされた後、INSERT の対象列の型にキャストされます。UNION に互換性のない型がある場合は、最初のキャストでエラーが発生する可能性があります。互換性レベル 90 で実行するには、INSERT SELECT 内で使用されている互換性のないデータ型の組合をすべて修正する必要があります。 | 中 |

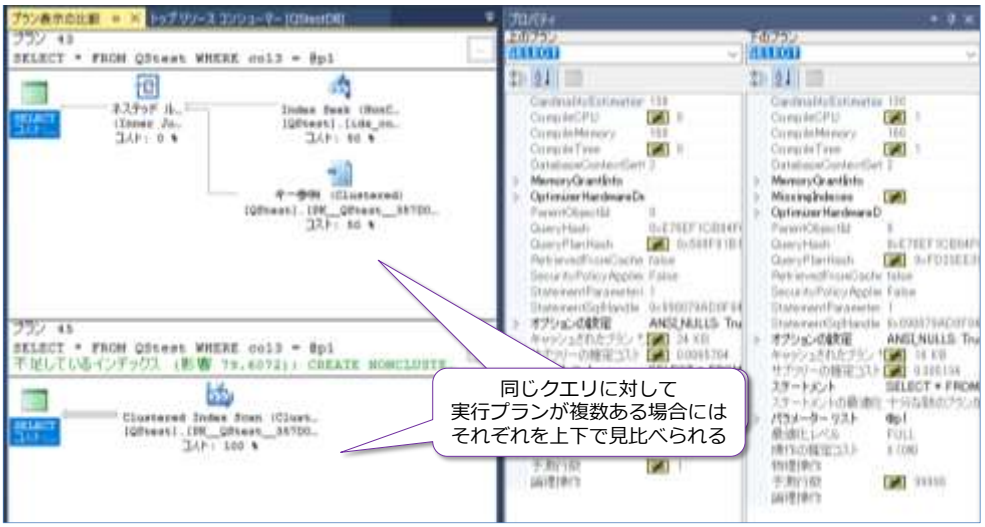
## 3.13 クエリストアで互換性レベルの違いによる実行プランの比較

互換性レベルを変更したことによる性能調査（実行プランの比較）には、SQL Server 2016 から提供された「**クエリ ストア**」（Query Store）機能を利用するのがお勧めです。**クエリ ストア**は、Store（蓄積）という名のとおり、**クエリの実行履歴**／**実行プラン**を保存できる機能です。




これまでのバージョンでも **query_stats** という動的管理ビュー（DMV）を参照すれば、クエリの実行履歴を確認することができましたが、この情報は、SQL Server を再起動したり、プロシージャ キャッシュのサイズが圧迫された場合には、参照できない（クリアされてしまう）という性質のものでした。これに対して、クエリ ストアであれば、クエリの実行履歴を永続化することができるので、SQL Server を再起動しても、過去に遡ってクエリの実行履歴を確認することができます。

クエリ ストアでは、クエリの実行履歴だけでなく、**実行プラン**も記録しておくことができ、かつ異なる実行プランで実行された場合には、その複数の実行プランを比較することができます。



同じクエリに対して、複数の実行プランがある場合には、どちらを利用させたいのかを指定（データベース管理者によるプラン強制／固定）することもできるので、互換性レベルを変更する前後で、性能が良いほうの実行プランを強制利用させる、といった使い方ができます。

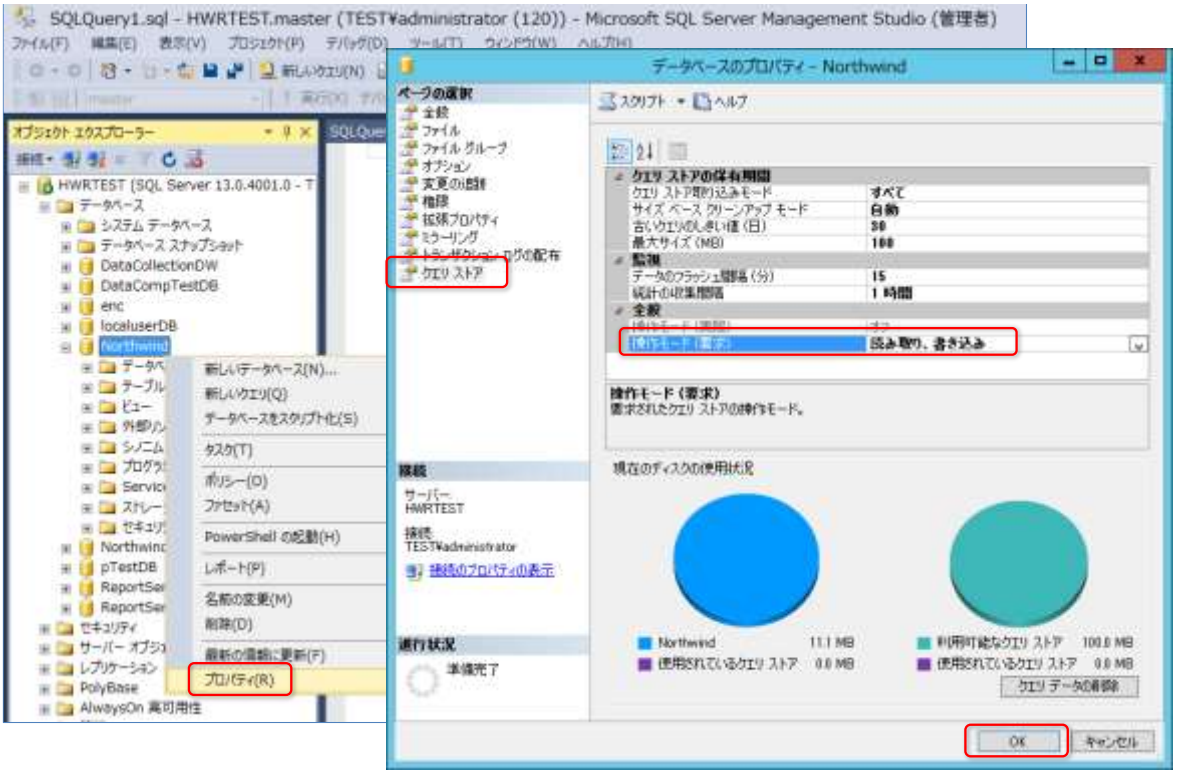
## 互換性レベルの違いを評価する手順

互換性レベルを変更するにあたっては、次のようにクエリ ストアを利用するのがお勧めです。

1. 互換性レベルはアップグレードした直後のものを利用する
   （SQL Server 2008／2008 R2 なら **100**、SQL Server 2012 なら **110**、SQL Server 2014 なら **120** をそのまま利用しておく）
2. データベースで**クエリ ストアを有効化**する
3. アプリケーションを実行して、データベースに対してクエリを発行する、あるいは、アプリケーションから発行されるクエリ（トレースで取得したものや .sql ファイルなど、ワークロードをスクリプト化したもの）をデータベースに対して実行する
4. 互換性レベルを **130**（SQL Server 2016 レベル）に上げる
5. 手順 3 と同じクエリ（ワークロード）をデータベースに対して実行する
6. クエリ ストアを参照して、実行プランの違いを確認する
7. 複数の実行プランがあるものは、最適な実行プランを利用するように、必要に応じてプラン強制を実施する

## クエリ ストアの有効化

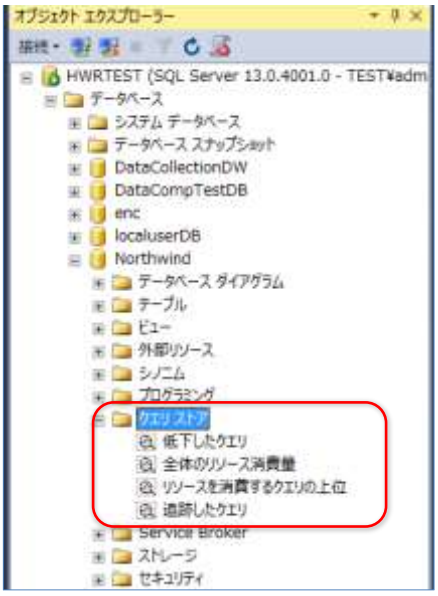
クエリ ストアを有効化するには、次のようにデータベースを右クリックして、[**プロパティ**] をクリックします。



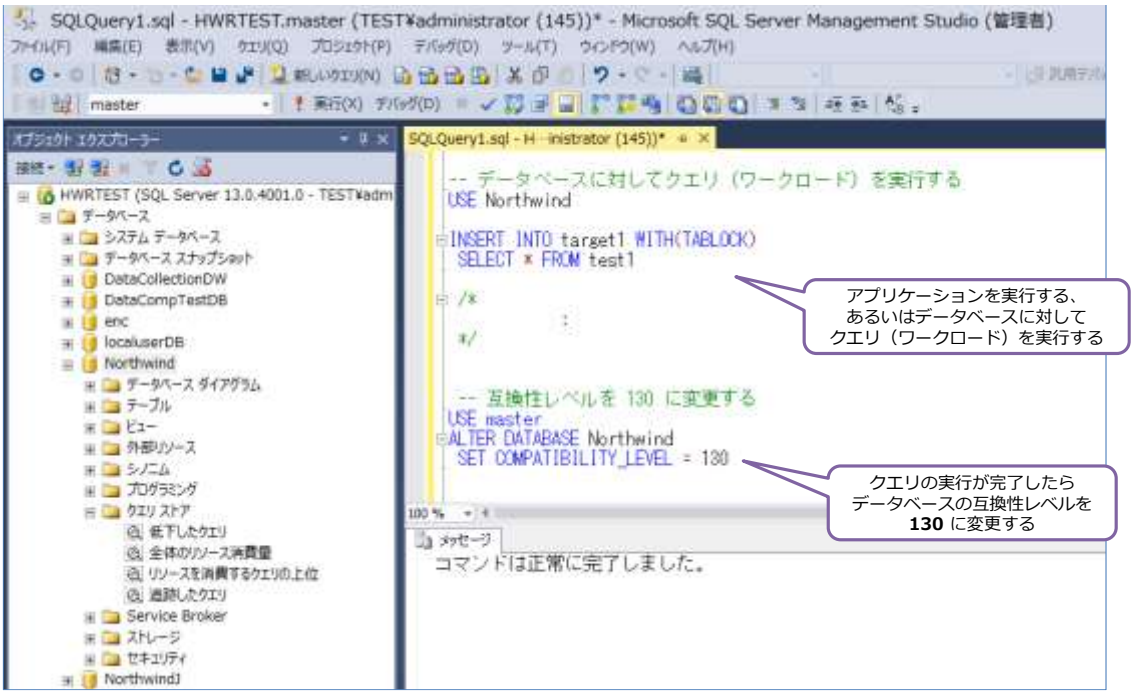

［**データベースのプロパティ**］ダイアログでは、［**クエリ ストア**］ページを開いて、［**操作モード (要求時)**］を［**読み取り/書き込み**］に変更し、［**OK**］ボタンをクリックします。

以上でクエリ ストアの設定が完了です。これでクエリ ストアが有効になって、クエリの実行履歴が自動的に記録されていくようになります。

クエリ ストアを有効化すると、次のようにデータベース内に［**クエリ ストア**］フォルダーが作成されて、記録されたクエリを確認することができます。

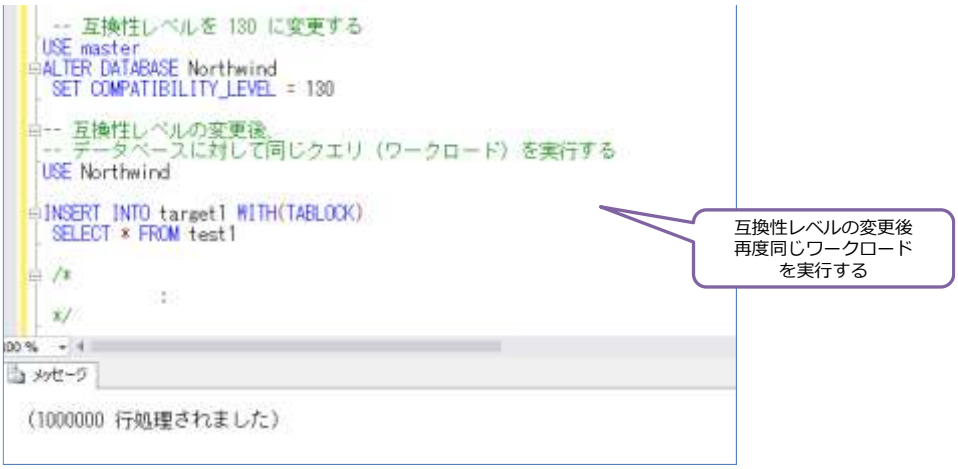



クエリ ストアの設定後は、アプリケーションを実行して、データベースに対してクエリを発行、あるいは、アプリケーションから発行されるクエリ（トレースで取得したものや .sql ファイルなど、ワークロードをスクリプト化したもの）をデータベースに対して実行します。

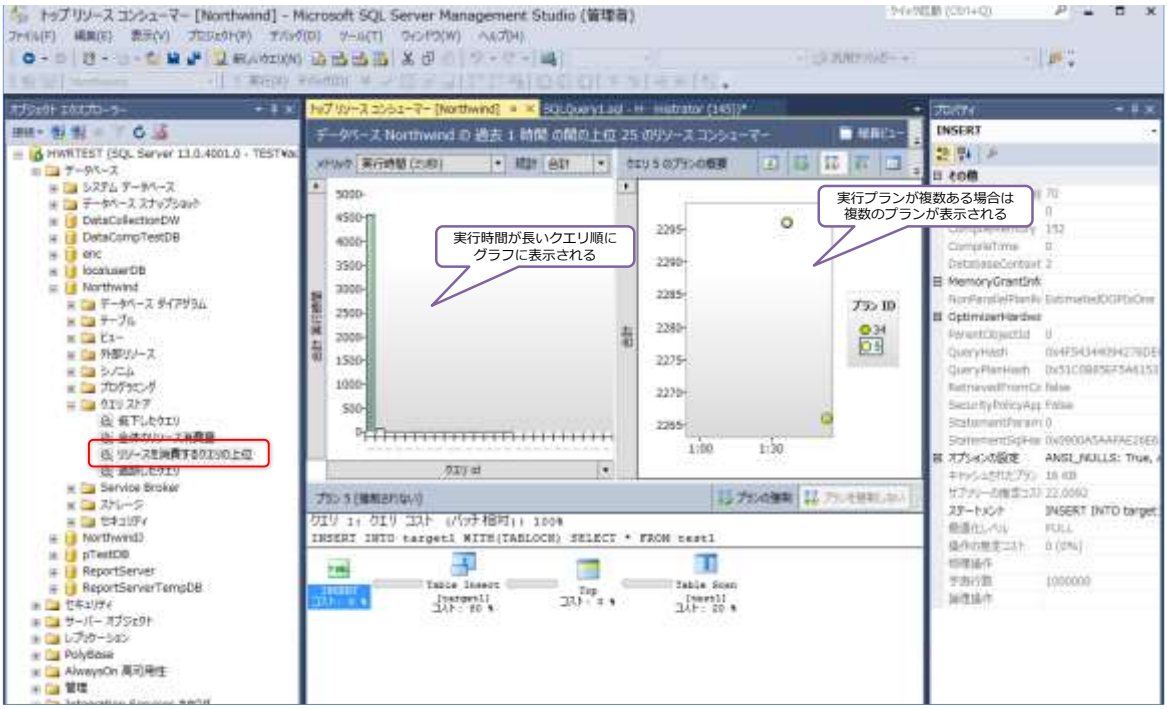



実行が完了したら、データベースの互換性レベルを **130**（SQL Server 2016 レベル）に変更します。

互換性レベルを変更したら、互換性レベルを変更する前と同じクエリ（ワークロード）をデータベースに対して実行します。




実行が完了したら、データベース内の［**クエリ ストア**］フォルダーを展開して、［**リソースを消費するクエリの上位**］をクリックします。



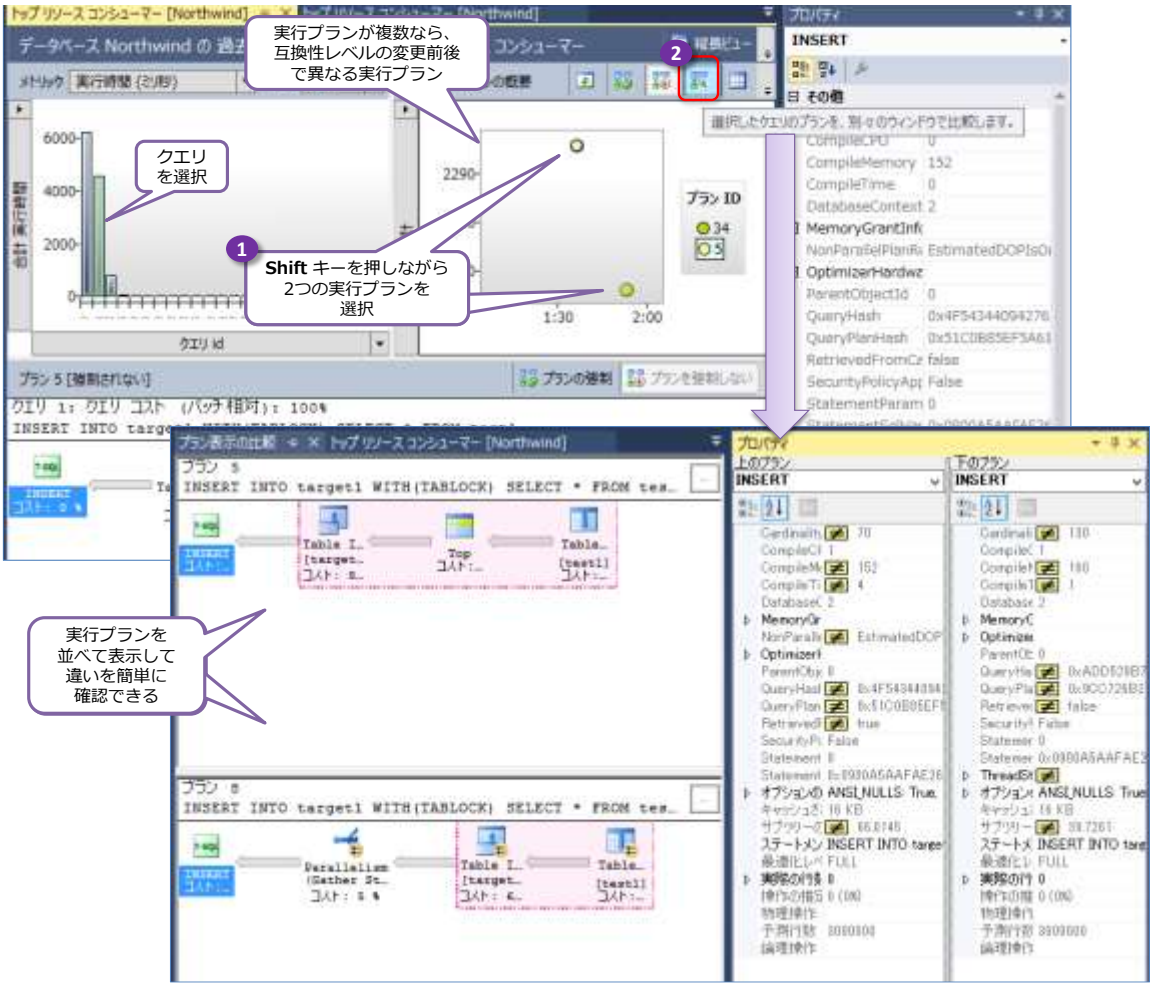

左側のグラフは、**合計実行時間**の長い順にクエリが表示されて、右側のグラフには、選択したクエリの**実行プラン**が表示されます。右側のグラフの **Y 軸**は、**合計実行時間**になっていて、実行プランごとの合計実行時間（各実行プランで実行されたクエリの合計の実行時間）で、どの実行プランが速く実行できたのかを一目瞭然で分かるようになっています。

この実行プランのグラフに、**複数の実行プラン**が表示されているということは、**互換性レベルを変更する前後**で**異なる実行プラン**で実行されていたということにもなります。

これに対して、次のように **1 つの実行プラン**しか表示されない場合は、互換性レベルを変更しても全く同じ実行プランが利用されたということが分かります。


複数の実行プランがある場合（互換性レベルの影響があった場合）は、次のように **Shift** キーを押しながら **2 つのプラン ID** をクリックして選択して、ツールバーの［**選択したクエリのプランを、別々のウィンドウで比較します**］をクリックすることで、実行プランを並べて比較することもでき

ます。




このように並べて表示することで、どこが異なるのかを非常に分かりやすく確認することができます。これで、合計実行時間や、内部的な実行プランを確認できるので、このうち、効率の良い実行プランを確実に利用させるようにするには、実行プランのグラフの上部に［**プランの強制**］ボタンがあるので、これをクリックするだけで行うことができます。

このように、クエリ ストア機能を利用すれば、互換性レベルの違いによる性能差を簡単に確認することができるので、ぜひ利用してみてください。

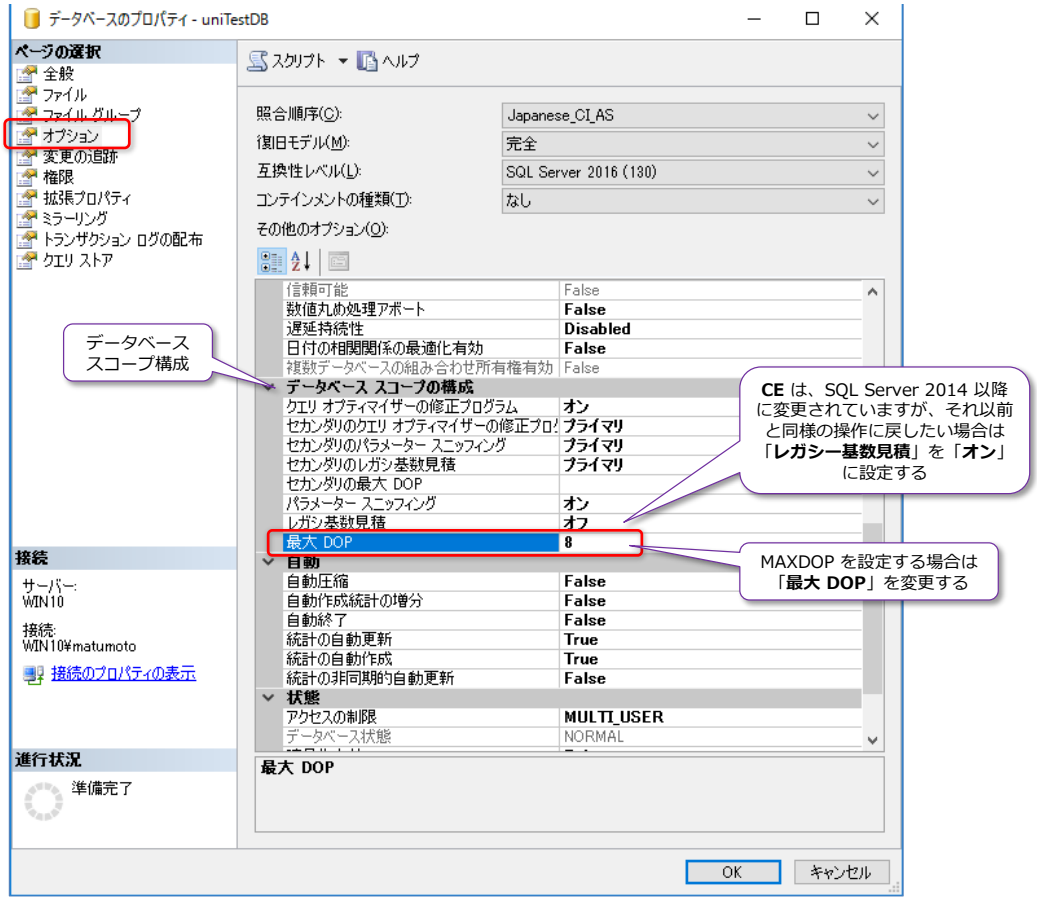
なお、その他のクエリ ストアの利用方法については、SQL Server 2016 自習書シリーズの No.1「**SQL Server 2016 からの新機能の概要**」編でも説明しているので、こちらもぜひご覧いただければと思います。

その他、互換性レベルに応じた実行プランについては、オンライン ブックの以下のトピックも参考になると思います。

クエリ プランの移行
http://msdn.microsoft.com/ja-jp/library/bb895281.aspx

## データベース単位での動作変更（データベース スコープ構成）

SQL Server 2016 からは、いくつかのクエリ プロセッサの動作をデータベース単位で変更できるようになっています。これは「**データベース スコープ構成**」と呼ばれて、次のようにデータベースのプロパティの「**オプション**」ページで設定できます。




データベース スコープ構成では、**MAXDOP**（並列クエリ時の利用する CPU コア数の制限）や **CE**（基数見積）の設定を変更できるので便利です。**CE** は、**SQL Server 2014**（互換性レベル **120**）から変更された新しいアーキテクチャを利用する場合は「**レガシ基数見積**」を「**オフ**」、SQL Server 2012 以前と同じ CE を利用したい場合は「**オン**」に設定します。

データベース スコープ構成は、**ALTER DATABASE SCOPED CONFIGURATION** ステートメントを利用して、次のように設定することもできます。

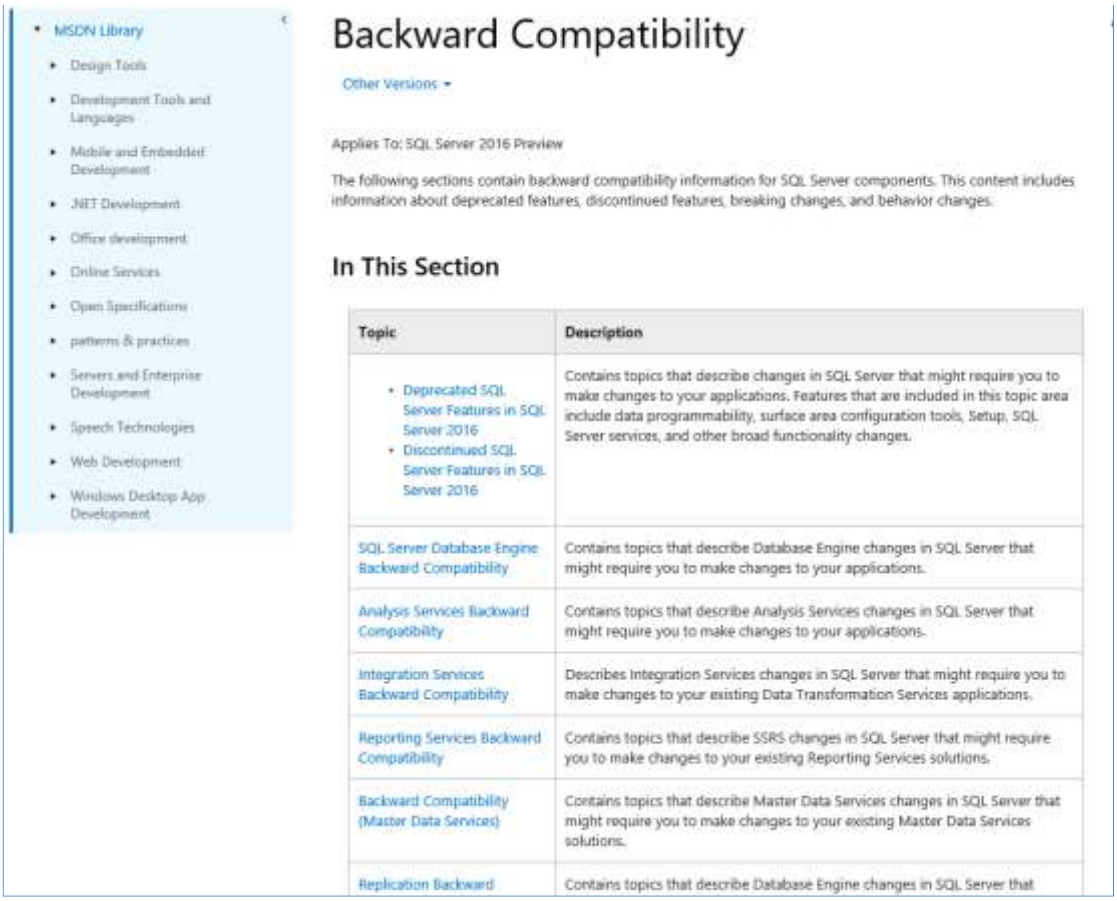
```
-- 例：レガシ基数見積 をオンに設定する場合
USE データベース名
ALTER DATABASE SCOPED CONFIGURATION
 SET LEGACY_CARDINALITY_ESTIMATION = ON
```

これらの設定を変更した場合も、**クエリ ストア**機能を利用すれば、設定前後での実行プランを簡単に比較することができるので、その設定が妥当なのかどうかをチェックできて、大変便利です。

## 3.14 SQL Server 2016 で提供が中止された機能

SQL Server 2005 以降の SQL Server と SQL Server 2016 との間では、データベース エンジンの基本部分に関して大きな変化はなく、ほとんどの機能をそのまま利用することができます。変更点に関しては、SQL Server 2016 のオンライン ブックの以下のトピックにまとまっています。

Backward Compatibility（旧バージョンとの互換性）
http://msdn.microsoft.com/en-us/library/cc280407.aspx




このトピックでは、さらに次のトピックに分かれています。

- **Deprecated SQL Server Features in SQL Server 2016**
  将来の SQL Server（SQL Server 2016 の次のバージョン以降）で廃止される予定の機能（非推奨の機能）
- **Discontinued SQL Server Features in SQL Server 2016**
  SQL Server 2016 で廃止された機能
- **SQL Server Database Engine Backward Compatibility**
  SQL Server データベース エンジンの旧バージョンとの互換性
- **Analysis Services Backward Compatibility**
- **Integration Services Backward Compatibility**
- **Reporting Services Backward Compatibility**

- **Backward Compatibility (Master Data Services)**
- **Replication Backward Compatibility**

Analysis Services や Integration Services、Reporting Services、マスター データ サービス、レプリケーション機能を利用している場合は、それぞれのトピックを参照しておくことをお勧めします。

SQL Server 全体や、データベースに関連する部分は、最初の 3 つ（**Deprecated SQL Server Features in SQL Server 2016**、**Discontinued SQL Server Features in SQL Server 2016**、**SQL Server Database Engine Backward Compatibility**）になるので、ここではこれらについて重要な順に説明します。

## SQL Server 2016 で提供が中止された機能（Discontinued）

Discontinued Database Engine Functionality in SQL Server 2016
SQL Server 2016 で廃止されたデータベース エンジンの機能
http://msdn.microsoft.com/ja-jp/library/ms144262.aspx


SQL Server 2016 で提供されなくなった機能は、次の 3 つのみです。

- 32 ビット版のインストーラーの廃止（**X64 版のみ**の提供）
- 互換性レベル **90**（SQL Server 2005 レベル）の廃止（SQL Server 2014 から廃止）
- **ActiveX** サブシステムの廃止（SQL Server Agent ジョブでの ActiveX スクリプトの実行不可）

互換性レベル 90 は廃止されたので、アップグレード時には、90 以下のデータベースは 100 に自動的にアップグレードされています。また、5 章の「移行」で紹介するバックアップ／復元機能を利用した場合は、90 以下のデータベースは、復元時に 100 に自動的にアップグレードされます。

ActiveX に関しては、SQL Server Agent ジョブで「**ActiveX スクリプト**」を利用している場合に影響します。もし、以前のバージョンで ActiveX スクリプトで記述したジョブがある場合は、SQL Server 2016 からは ActiveX スクリプトを実行できなくなっているので、代わりに PowerShell などを利用するように変更する必要があります。

## SQL Server 2014 以前のバージョンで提供が中止された機能

**SQL Server 2014 以前のバージョン**で提供が中止された機能は、次のとおりです。

**SQL Server 2014 で提供が中止されたもの**

- 互換性レベル「**90**」（SQL Server 2005 相当の互換性レベル）の提供中止

**SQL Server 2012 で提供が中止されたもの**

- **Active Directory Helper** サービスの提供中止
  （sp_ActiveDirectory_Obj、sp_ActiveDirectory_SCP、sp_ActiveDirectory_Start）
- 互換性レベル「**80**」（SQL Server 2000 相当の互換性レベル）の提供中止
- BACKUP ステートメントでの **WITH PASSWORD** の提供中止
- RESTORE ステートメントでの **WITH DBO_ONLY** の提供中止
- 構成オプションの「**user instance timeout**」と「**user instances enabled**」の提供中止
- **VIA** プロトコルの提供中止
- トリガーでの **WITH APPEND** 句の提供中止
- **sp_dboption** の提供中止（ALTER DATABASE を利用するように変更）
- **SQL Mail** の提供中止（データベース メールを利用するように変更）
- 32 ビットの **AWE** (Address Windowing Extensions) と 32 ビットのホット アド メモリ サポートの提供中止（代わりに X64 環境を利用するように変更）
- **DATABASEPROPERTY** の提供中止（代わりに DATABASEPROPERTYEX を利用するように変更）
- **SQL-DMO** の提供中止（代わりに SMO を利用するように変更）
- **FASTFIRSTROW** ヒントの提供中止（代わりに OPTION (FAST n) を利用するように変更）
- **sp_addserver** によるリモート サーバー機能の提供中止（代わりにリンク サーバーを利用するように変更）
- **sp_dropalias** の提供中止
- **PWDCOMPARE** の提供中止
- SMO での **Service Broker** のプログラミングの提供中止
- **SET DISABLE_DEF_CNST_CHK** の提供中止

- **sys.database_principal_aliases** の提供中止
- **RAISERROR integer 'string'** という形式で利用する RAISERROR の提供中止（RAISERROR(...) 構文を利用するように変更）
- **COMPUTE／COMPUTE BY** の提供中止（代わりに ROLLUP を利用するように変更）
- ***=** および **=*** の提供中止（代わりに OUTER JOIN を利用するように変更）
- **XEvents**（拡張イベント）の変更

**SQL Server 2008／2008 R2 で提供が中止されたもの**
SQL Server 2008 R2 の該当ヘルプより
http://msdn.microsoft.com/ja-jp/library/ms144262(v=sql.105).aspx

- 互換性レベル **60**、**65**、**70** の提供中止（SQL Server 7.0 以前の互換性レベル）
- **sp_addalias** の提供中止
- **登録済みサーバー API** の提供中止
- **DUMP／LOAD** ステートメントの提供中止
- BACKUP LOG **WITH TRUNCATE_ONLY** および **NO_LOG** の提供中止（代わりにログ バックアップを実行するように変更）
- **DBCC CONCURRENCYVIOLATION** の提供中止
- **グループ**機能（sp_addgroup、sp_changegroup、sp_dropgroup、sp_helpgroup）の提供中止（代わりにロールを利用するように変更）
- **Northwind** と **pubs** サンプル データベースの提供中止（代わりに AdventureWorks を提供）
- Reporting Services の **Itanium プラットフォーム**でのサポートの提供中止
- xpress エディションでの **SQL-DMO** の提供中止
- **Web Assistant** 機能の提供中止
- **セキュリティ構成ツール**の提供中止
- **自動インストール**（無人セットアップ）でのいくつかのパラメーター変更

以上が提供が中止された機能になりますが、皆さんのシステムに該当しそうなものは、「**互換性レベル 90 の提供中止**」、「**32 ビット AWE の提供中止**」（代わりに X64 を利用）「**BACKUP LOG WITH TRUNCATE_ONLY の提供中止**」（代わりにログ バックアップを利用）あたりではないでしょうか？

## 推奨されない機能（将来のバージョンで提供されなくなる機能）

Deprecated Database Engine Features in SQL Server 2016
SQL Server 2016 データベース エンジンの非推奨機能
http://msdn.microsoft.com/ja-jp/library/ms143729.aspx

SQL Server 2016 データベース エンジンの非推奨機能

SQL Server 2016 and later

適用対象: SQL Server (2016 以降) Azure SQL Database Azure SQL Data Warehouse Parallel Data Warehouse

このトピックでは、SQL Server データベース エンジン でまだ使用できるものの、非推奨となった SQL Server 2016の機能について説明します。これらの機能は SQL Serverの今後のリリースで削除される予定です。非推奨機能を新しいアプリケーションで使用しないでください。

非推奨機能の使用は、SQL Server Deprecated Features オブジェクトのパフォーマンス カウンターおよびトレース イベントを使用して監視できます。詳細については、「SQL Server オブジェクトの使用」を参照してください。

これらのカウンターの値は、次のステートメントを実行して入手することもできます。

```
SELECT * FROM sys.dm_os_performance_counters
WHERE object_name = 'SQLServer:Deprecated Features';
```

SQL Server の次のバージョンでサポートされない機能

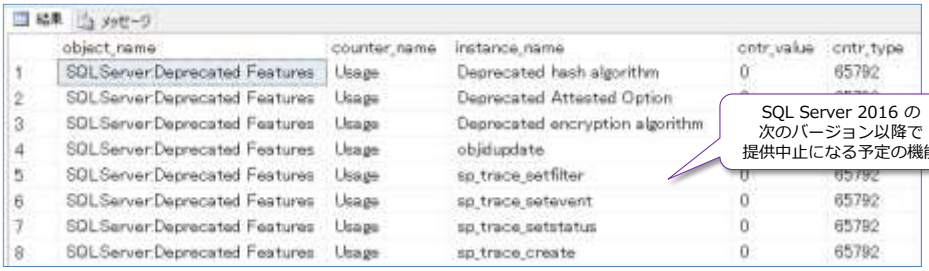
以下の SQL Server データベース エンジン 機能は、SQL Serverの次のバージョンではサポートされません。新規の開発作業ではこれらの機能を使用しないようにし、現在これらの機能を使用しているアプリケーションはできるだけ早く修正してください。機能名 の値は、トレース イベントには ObjectName として表示され、パフォーマンス カウンターおよび sys.dm_os_performance_counters にはインスタンス名として表示されます。機能 ID の値は、トレース イベントに ObjectId として表示されます。

| カテゴリ | 非推奨機能 | 代替 | 機能名 | 機能 ID |
|---|---|---|---|---|
| バックアップと復元 | RESTORE { DATABASE \| LOG } WITH {MEDIA}PASSWORD はこれまでどおり非推奨です。BACKUP { DATABASE \| LOG } WITH PASSWORD および BACKUP { DATABASE \| LOG } WITH MEDIAPASSWORD は廃止されました。 | [SQL] : | BACKUP DATABASE または LOG WITH PASSWORD<br><br>BACKUP DATABASE または LOG WITH MEDIAPASSWORD | 104<br><br>103 |

このページでリストされている機能は、SQL Server 2016 では引き続き利用することができますが、次の SQL Server 以降で提供が停止される予定の機能の一覧になっています。したがって、今後のアプリケーション開発においては、これらの機能はなるべく利用しないほうが、今後のバージョンアップ時に問題が発生しなくて済むようになるので、できる限り利用しないことをお勧めします。

停止される予定の機能は、以下のようにクエリを実行して確認することもできます。

```
SELECT * FROM sys.dm_os_performance_counters
 WHERE object_name = 'SQLServer:Deprecated Features'
```



結果 メッセージ

| | object_name | counter_name | instance_name | cntr_value | cntr_type |
|---|---|---|---|---|---|
| 1 | SQLServer:Deprecated Features | Usage | Deprecated hash algorithm | 0 | 65792 |
| 2 | SQLServer:Deprecated Features | Usage | Deprecated Attested Option | | |
| 3 | SQLServer:Deprecated Features | Usage | Deprecated encryption algorithm | | |
| 4 | SQLServer:Deprecated Features | Usage | objidupdate | | |
| 5 | SQLServer:Deprecated Features | Usage | sp_trace_setfilter | 0 | 65792 |
| 6 | SQLServer:Deprecated Features | Usage | sp_trace_setevent | 0 | 65792 |
| 7 | SQLServer:Deprecated Features | Usage | sp_trace_setstatus | 0 | 65792 |
| 8 | SQLServer:Deprecated Features | Usage | sp_trace_create | 0 | 65792 |

SQL Server 2016 の次のバージョン以降で提供中止になる予定の機能

SQL Server 2016 の次のバージョンでは、互換性レベル 110（SQL Server 2008 R2 レベル）

や SQL Server 2008／2008 R2 からのアップグレード、sqlmaint ユーティリティの廃止などが予定されています。

## 重大な変更（Breaking Changes）

**SQL Server Database Engine Backward Compatibility**（SQL Server データベース エンジンの旧バージョンとの互換性）トピックでは、次のように 3 つのサブトピックがあります。

http://msdn.microsoft.com/ja-jp/library/ms143532.aspx


非推奨機能（**Deprecated**）と廃止された機能（**Discontined**）のトピックは、前掲のものと同じリンクになっています。重大な変更（**Breaking Changes**）のトピックについては、ここで説明します。

Breaking Changes to Database Engine Features in SQL Server 2016
SQL Server 2016 におけるデータベース エンジン機能の重大な変更
http://msdn.microsoft.com/en-us/library/ms143179.aspx

このページで紹介されている重大な変更は、以下の 4 点です。

- sys.dm_io_virtual_file_stats の sample_ms 列のデータ型が int から bigint に変更
- sys.fn_virtualfilestats の TimeStamp 列のデータ型が int から bigint に変更
- 暗号化アルゴリズムの MD2、MD4、MD5、SHA、SHA1 は非推奨。互換性レベル 130 では、SHA2_256 と SHA2_512 のみが許可される
- 互換性レベル 130 では、datetime から datetime2 への暗黙的な変換では、ミリ秒が考慮されるように変更されたので、明示的な変換を利用することを推奨

## SQL Server 2014 以前のバージョンでの重大な変更

**SQL Server 2014** 以前のバージョンでの**重大な変更**（**Breaking Changes**）トピックに記載されている内容は、次のとおりです。

**SQL Server 2014 での重大な変更点**

- 特になし

**SQL Server 2012 での重大な変更点**

- **NEXT** がキーワードに変更されたので列名やテーブル名に NEXT を利用している場合はエラーが発生する可能性がある（∵シーケンスのサポート）
- **PIVOT** は互換性レベルが 110 の場合、**再帰 CTE** で許可されない
- **アプリケーション ロール**（sp_setapprole と sp_unsetapprole）の動作の変更
- EXECUTE AS のクッキーの **OUTPUT** パラメーターの動作の変更
- sys.**fn_get_audit_file** 関数で 2 つの追加列がある
- **WITHIN** が予約キーワードに変更された
- time および datetime2 データ型の**計算列**で、CAST または CONVERT で**日付／時刻スタイルを省略**したときの動作の違い（110 以降では既定のスタイルが 121 になる）
- ALTER TABLE で 4 部構成（**server.database.schema.table**）のテーブル指定での動作が変更
- **FOR BROWSE／SET NO_BROWSETABLE ON** の動作の変更
- **SOUNDEX** 関数では動作の変更
- **DML** ステートメントが失敗した場合の RowCount 動作の変更
- **SERVERPROPERTY** 関数の動作の変更
- **CREATE LOGIN WITH PASSWORD = 'password' HASHED** の動作の変更
- datetimeoffset で CAST／CONVERT で**日付／時刻スタイルを省略**したときの動作の変更
- **動的管理ビュー**の変更（sys.dm_exec_requests、sys.dm_os_memory_cache_counters、sys.dm_os_memory_cache_entries、sys.dm_os_memory_clerks、sys.dm_os_workers、sys.dm_os_memory_nodes、sys.dm_os_memory_objects、sys.dm_os_sys_info）

- **カタログ ビュー**の変更（sys.data_spaces、sys.partition_schemes、sys.filegroups、sys.partition_functions）
- geometry、geography、および hierarchyid での **Microsoft.SqlServer.Types.dll** の変更
- **XQuery** 関数のサロゲート対応に伴う動作の変更

SQL Server 2012 の該当ヘルプより
http://msdn.microsoft.com/ja-jp/library/cc707784(v=sql.110).aspx

- **XEvents**（拡張イベント）の変更
  resource_monitor_ring_buffer_record／memory_node_oom_ring_buffer_recorded で single_pages_kb、multiple_pages_kb が削除、target_kb、pages_kb を追加
- **Sync Framework** が含まれなくなった（別途インストールが必要）

**SQL Server 2008／2008 R2 での重大な変更点**

- **自動インストール**時に /IAcceptSQLServerLicenseTerms が必須になった
- **SMO** の変更（SmoEnum.dll が削除されているなど）
- **新しい照合順序**の追加（Windows Server 2008 が提供する照合順序に完全対応した新しい照合順序を 80 個導入）
- **CLR 統合**（SQL CLR）の動作の変更
- **動的管理ビュー**の変更（sys.dm_os_sys_info、sys.dm_exec_query_resource_semaphores、sys.dm_exec_query_memory_grants）
- Windows 認証での**ログイン エラー時のエラー番号**の変更（SQL Server 2005 では 18452、2008 では 18456）
- **プラン表示**の XML スキーマの変更
- **DDL イベント** ALTER_AUTHORIZATION_DATABASE の動作の変更
- CONVERT 関数での**無効なスタイル**が渡されたときの動作の変更
- **アセンブリ**に対する EXECUTE 権限の許可、拒否、および取り消しを行うことはできなくなった
- **システム型**に対する権限の許可、拒否、および取り消しを行うことができなくなった
- **GROUP BY** リストに使用される式に**サブクエリ**を含めることができなくなった
- **OUTPUT 句**の動作の変更
- **precompute rank** サーバー レベル オプションがサポートされなくなった
- スナップショット分離で **READPAST** ヒントを指定することができなくなった
- **sp_helpuser** の動作の変更
- **透過的なデータ暗号化**（TDE）がサポートされるようになった
- **XQuery** の動作の変更
- **SSL** を使用した SQL Server Native Client 接続の動作の変更

以上が重大な変更があったものになりますが、皆さんのシステムに該当するものは少ないのではないでしょうか。

## SQL Server 2014 以前のバージョンでの動作の変更

**SQL Server 2014** 以前のバージョンでの**動作の変更**（**Behavior Changes**）トピックに記載されている内容は、次のとおりです。

**SQL Server 2014 での動作の変更**

- **XML** ドキュメントで **4020** 文字を超えた場合の動作の変更

**SQL Server 2012 での動作の変更**

SQL Server 2012 の該当ヘルプより
http://msdn.microsoft.com/ja-jp/library/cc707785(v=sql.110).aspx

- **マルチサブネット フェールオーバー クラスター**がサポートされる
- フェールオーバー クラスターでの **SQL Server エラー検出方法**の変更
- **SQLDescribeCol** の動作の変更
- **CLR ユーザー定義関数**と **CLR ユーザー定義型の決定的メソッド**でたたみ込みが可能になった
- 空間型の STEnvelope() メソッドの動作の変更
- **LOG** 関数で省略可能なパラメーターの追加
- パーティション インデックス操作中の**統計計算**の変更
- **XML value** メソッドによるデータ型変換の変更
- **XML** モードでの sqlcmd.exe の動作変更
- **DBCC CHECKIDENT** のメッセージの変更
- **XML** データ型に対する **exist()** 関数の動作の変更

**SQL Server 2008／2008 R2 での動作の変更**

SQL Server 2008 の該当ヘルプより
http://msdn.microsoft.com/ja-jp/library/ms143359(v=sql.100).aspx

- **ジョブのスクリプト生成**での動作の変更（@schedule_uid）
- **access check result cache** オプションの変更
- **フルテキスト検索**のアーキテクチャの変更（データベース エンジンに統合）
- **ループバック リンク サーバー**での動作の変更
- **プラン ガイド**の動作の変更
- **パーティション テーブル／インデックス**でのクエリ処理方法の変更（クエリ プロセッサの改良）
- **REPLACE** 関数で末尾のスペースが切り捨てられるかどうかの動作の違い（SQL Server 2005 では切り捨て、2008 では切り捨てない）
- **Resource** データベースが作成される場所の変更
- **tempdb** データベースでの PAGE_VERIFY オプションの既定値が **CHECKSUM** に変更
- **INSERT .. SELECT** での最小ログ記録のサポート

- **XML** データ型での動作変更（lax 検証、xs:anyType 要素など）

これらも皆さんのシステムに該当するものは少ないのではないでしょうか。

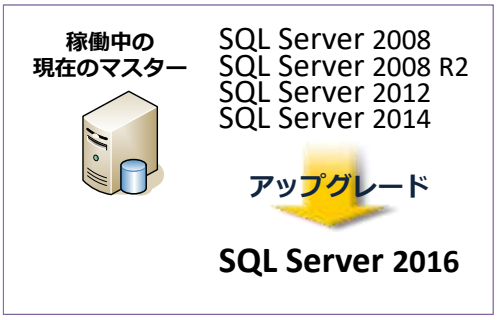
**フルテキスト検索**機能を利用している場合には、SQL Server 2008 や SQL Server 2012 で大きな動作変更があるので、以下のトピックも一読しておくことをお勧めします。

フルテキスト検索における旧バージョンとの互換性
http://msdn.microsoft.com/ja-jp/library/ms143544.aspx

フルテキスト検索の重大な変更
http://msdn.microsoft.com/ja-jp/library/ms143709.aspx

## 3.15 ケース1「同一マシンでのアップグレード」のまとめ

この章で説明した「**ケース1　同一マシンでのアップグレード**」についてまとめると、次のようになります。




### アップグレード手順

このケースでのアップグレード手順は、次のとおりです。

1. **Data Migration Assistant** による事前チェックを行う

   Data Migration Assistant は、Windows 10 や 8、8.1 などのクライアント OS にインストールして、リモート マシンからチェックすることも可能。

2. **OS** に Windows Server 2003 や 2003 R2、2008、2008 R2 を利用している場合は、**Windows Server 2012** 以上にアップグレードする

   SQL Server 2016 は、Windows Server 2012（X64）、Windows Server 2012 R2（X64）＋ KB 2919355、Windows Server 2016（X64）でサポートされている。

   OS のアップグレードにあたっては、SQL Server を動作させるための Service Pack 要件を確認する(例えば､Windows Server 2012 にアップグレードする場合には､SQL Server 2008 なら SP3、SQL Server 2008 R2 なら SP1 が必要など)

3. **SQL Server 2016** への**アップグレード要件**を確認する（アップグレード可能な Service Pack を確認／インストールする）

   SQL Server 2008 は SP 4 、SQL Server 2008 R2 は SP 3 、SQL Server 2012 は SP2 が必要。SQL Server 2014 の場合は SP は必要ない。

4. **SQL Server 2016** への**アップグレード インストール**を行う

   アップグレード中は、.NET Framework 4.6 が自動的にインストールされる。

5. **SQL Server 2016** の**最新の修正プログラム**（CU や Service Pack）をインストールする

   CU2 には、性能向上に関する修正が入っているので、できる限り最新の修正プログラムを適用しておくことをお勧めします。

Reporting Services を利用している場合は、SP1 に対する重要な更新プログラムがあるので、CU4（SP1＋CU1）以上を適用しておくことをお勧めします。

6. **Management Studio**（管理ツール）の最新版をダウンロードして、インストールする（オプション）

7. **統計**（Statistics）を更新する

   **sp_updatestats** を実行して、統計を更新する

8. **データベースの互換性レベルを 130** へ上げる（オプション）

   既定では、SQL Server 2008／2008 R2 からのアップグレードの場合は、**100**、SQL Server 2012 からの場合は **110**、SQL Server 2014 からの場合は **120** になる。互換性レベル **90**（SQL Server 2005 レベル）以下のデータベースがある場合は、アップグレード時に **100**（SQL Server 2008 レベル）に自動的に上がる。互換性レベルは **130** を利用することで、各種の性能向上機能を利用できるので、130 に上げるのがお勧め。

   互換性レベルの影響については**クエリ ストア**機能を利用することで簡単にチェック可能（実行プランの比較や、プラン強制もできる）。

9. **BIDS**（Business Intelligence Development Studio）や **SSDT-BI**（SQL Server Data Tools - Business Intelligence）を利用している場合は、**SSDT** の最新版をダウンロードして、インストールする（オプション）

   SSDT は、Windows 10 や 8、8.1 などのクライアント OS にインストールすることもできる。

10. **コマンドライン ツール**のパスの変更（オプション）

    sqlcmd や bcp、後述の dtexec などのコマンドライン ツールは、既定で古いバージョンのツールが利用されるので、SQL Server 2016 バージョンのツールを利用するには、パスを指定して実行する必要がある。あるいは環境変数の **PATH** を変更する

ケース 1 のメリットは、以前のバージョンで利用していた機能をほとんどすべてそのまま利用できることです。**データベース**を以前と変わらずに利用できることはもちろん、**ログイン アカウント**や、**ジョブ**、**警告**、**暗号化**、**リンク サーバー**、**メンテナンス プラン**（保守計画）、**リソース ガバナー**、**監査**（SQL Server Audit）、**データベース メール**、**サーバーの構成オプション**など、以前の SQL Server で設定／利用していた機能を、そのまま SQL Server 2016 上でも利用することができます（パフォーマンス データ コレクションだけは例外で、追加の手順が必要になりますが、これについては後述します）。

また、**レプリケーション**や**ログ配布**、**データベース ミラーリング**、**可用性グループ**といったサーバー間の連携機能や、**WSFC**（Windows Server フェールオーバー クラスタリング）上の SQL Server インスタンスに関しても、アップグレードをすることができます。ログ配布／データベース ミラーリング／可用性グループに関しては、**セカンダリ**を先にアップグレードすることで、ロー

リング アップグレードも可能です（後述します）。

**Integration Services** や **Reporting Services**、**Analysis Services** を利用している場合でも、アップグレード インストールをすることによって、以前のバージョンで利用していた Integration Services パッケージや Reporting Services レポート、Analysis Services の多次元キューブを、アップグレード後もそのまま利用することができます（後述します）。

一方で、この方法のデメリットは、アップグレード後に、旧システム環境が利用できなくなってしまう（以前のバージョンの SQL Server が完全に利用できなくなってしまう）ことです。これだと、万が一アップグレードに失敗してしまった場合には、元の環境に戻すのが大変になり、失敗時は、（最悪は）OS のインストールからやり直して、バックアップからすべてを復元しなければならない場合があります。

したがって、このケースを利用する場合は、万が一のアップグレード失敗時に備えて、元の環境でしっかりとバックアップを取得しておくこと、元へ戻す手順（ロールバック手順）をしっかりと計画しておくことが重要になります。また、次の章で説明する「**ケース 2 新規サーバーへのアップグレード**」のように、ハードウェア リプレースを伴うアップグレードの場合であれば、元の環境をそのまま残しておくことができるので、万が一の失敗時にもすぐに元の環境に戻せるようになります。

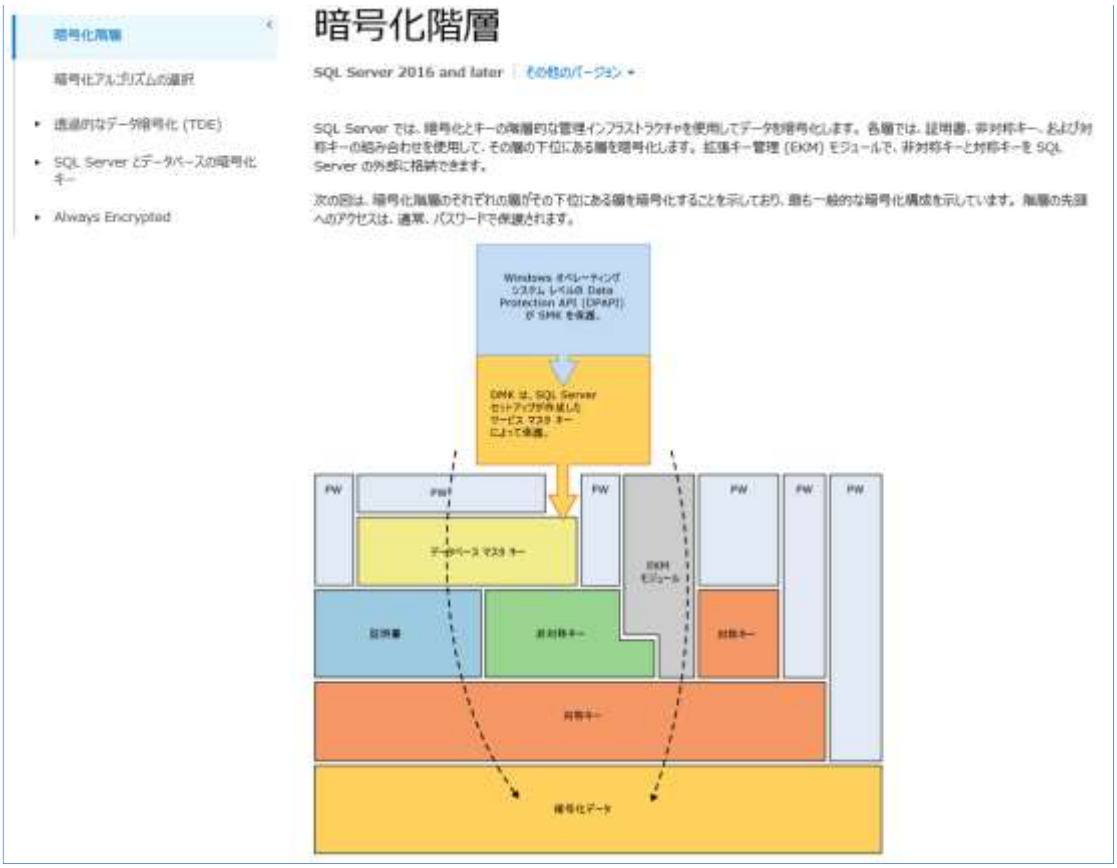
## 3.16 アップグレード前に実行しておくべき作業（バックアップ）

繰り返しになりますが、このケースでは、アップグレード後に、旧システム環境が利用できなくなってしまう（以前のバージョンの SQL Server が完全に利用できなくなってしまう）ことになるので、万が一のアップグレードの失敗に備えて、アップグレード前にはバックアップをしておくことが重要です。SQL Server に関しては、**サービス マスター キー**と**全データベースのオフライン バックアップ**（ユーザー データベースに関してはオンライン バックアップでも可）を取得しておくことがお勧めになります。

### サービス マスター キーのバックアップ

**サービス マスター キー**は、SQL Server の**各種の暗号化**における最も重要なキーとなるものです。これについては、オンライン ブックの以下のトピックを一読しておくことをお勧めします。

暗号化階層
http://msdn.microsoft.com/ja-jp/library/ms189586.aspx




サービス マスター キーは、**リンク サーバー**のセキュリティ設定（リモート ログインのパスワードなど）や、**データベース メール**でのメール サーバーへのログイン パスワード、**TDE**（透過的なデータ暗号化）で利用する**データベース マスター キー**などで利用され、（内部的なデータを含めた）**各種のデータを暗号化**するための最も重要なキーになります。

サービス マスター キーをバックアップするには、次のように **BACKUP SERVICE MASTER**

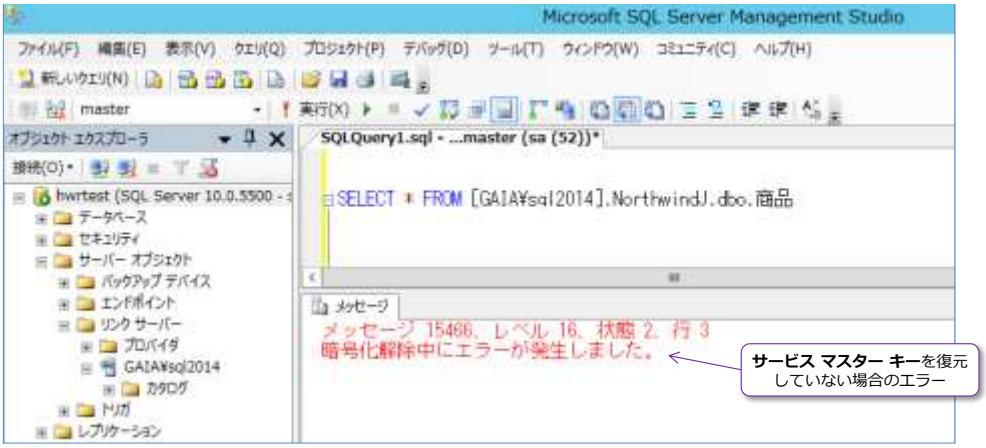
**KEY** ステートメントを実行します。

```
USE master
BACKUP SERVICE MASTER KEY TO FILE = 'C:¥smk.bak'
 ENCRYPTION BY PASSWORD = 'XXXXXXXX'
```

**TO FILE** で指定するファイル名は、皆さんの環境に合わせて変更してください。また、**ENCRYPTION BY PASSWORD** で指定するパスワードは、複雑なパスワード（大文字と小文字、@などの特殊文字を含むなど）にする必要があり、復元時に必要となるパスワードです（復元は、**RESTORE SERVICE MASTER KEY** ステートメントで行うことができます）。

ただし、バックアップしたサービス マスター キーを復元できるのは、SQL Server サービスの**サービス アカウント**が同じ場合のみになるので、OS を入れ替えたりした場合を考慮すると、サービス アカウントを **Active Directory のドメインのユーザー**にしておく必要があります（∵OS を入れ替えると、**Windows のローカル ユーザー**には、異なる Security ID が割り当てられてしまうため、ローカル ユーザーをサービス アカウントにしていると、OS 入れ替え時にサービス マスター キーを復元できなくなってしまいます）。

もし、サービス マスター キーを復元できない場合は、このキーを利用して暗号化された機能が次のように利用できなくなります。




これは、SQL Server 2008 でリンク サーバーを利用しているときのエラーですが、「**暗号化解除中にエラーが発生しました**」と表示されます。

データベース メール機能を利用していて、メールサーバーへのログイン情報を設定している場合には、これと同じエラーで、利用できなくなります。

**TDE**（透過的なデータ暗号化）を利用している場合は、**データベース マスター キー**が、サービス マスター キーに依存しているので、TDE が利用できなくなります。

こうした状況を回避するための方法（どうしてもサービス マスター キーを復元できない状況への対処方法）として、サービス マスター キーを作成し直す（**REGENERATE** する）ということもできます。これは、次のように **ALTER SERVICE MASTER KEY** ステートメントを実行します。

```
USE master
```

```
ALTER SERVICE MASTER KEY REGENERATE
```

暗号化が関連する機能や、パスワードの設定を含むような機能の動作で、エラーになる場合は、サービス マスター キーをぜひ確認してみてください。「**暗号化解除中にエラー**」がキーワードです。

もし、上記のステートメントがエラーになる場合は、次のように **FORCE** オプションを付けて、強制的に作成し直すことも検討してみてください。

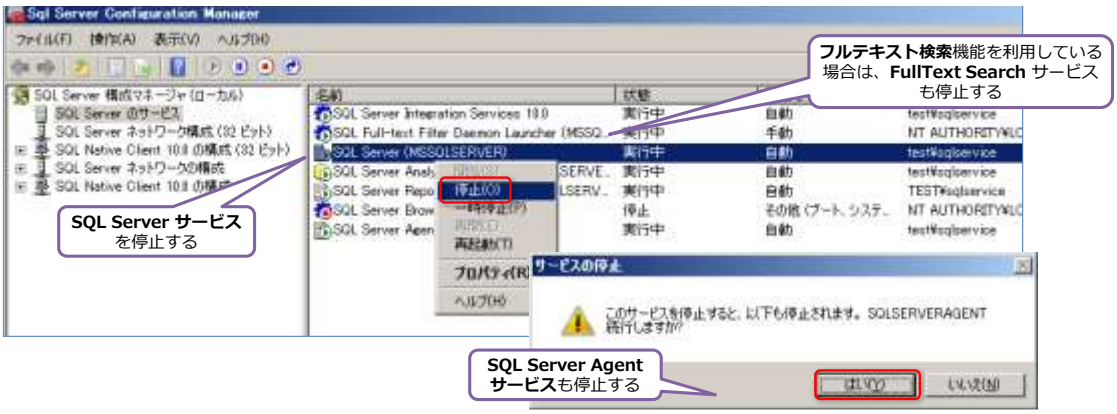
```
USE master
ALTER SERVICE MASTER KEY FORCE REGENERATE
```

ただし、このように、**FORCE** を利用した場合は、暗号設定が失われる可能性があるので、暗号化が関連する機能（リンク サーバーのセキュリティ設定やデータベース メールでのメールサーバーへのログイン情報など）の再設定が必要になる場合があります。**データベース マスター キー**や**証明書**を利用している場合には、それらも旧マスターでバックアップしておくことをお勧めします。

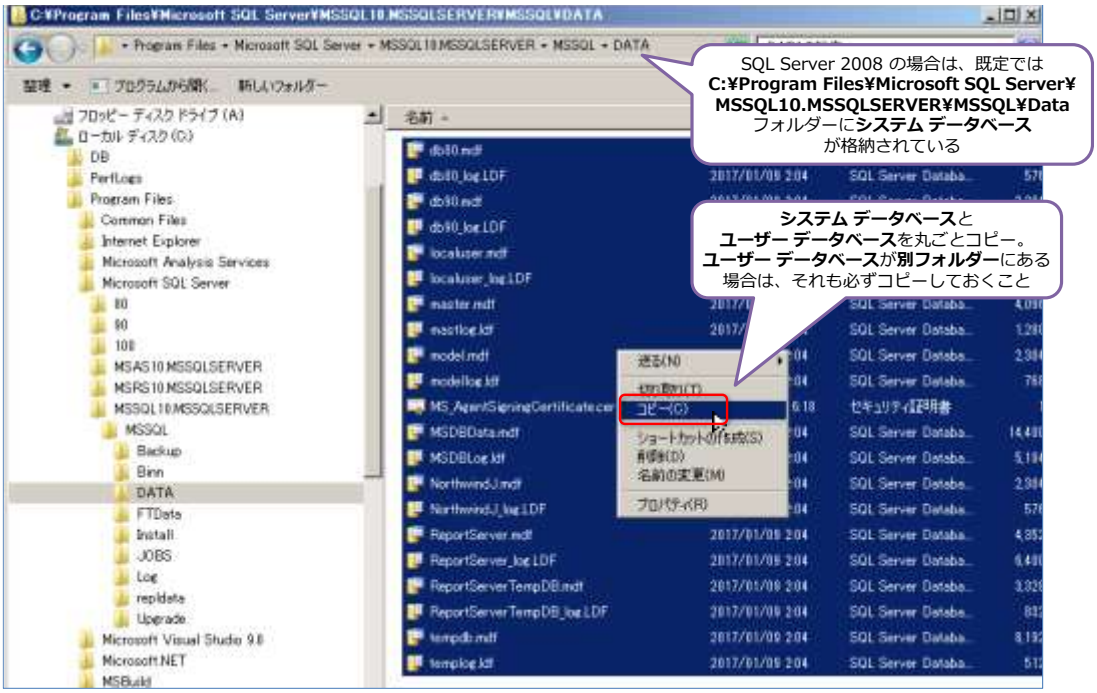
## 全データベースのオフライン バックアップを取得しておく

アップグレード前には、**全データベースのオフライン バックアップ**（ユーザー データベースに関してはオンライン バックアップでも可）を取得しておくことがお勧めになります。**master** や **msdb** などの**システム データベース**のオフライン バックアップは、次のように行います。

まず、オフライン バックアップを取得するために **SQL Server サービス**と **SQL Server Agent サービス**を停止します（**フルテキスト検索**機能を利用している場合は、**SQL Full-text Filter Daemon Launcher** サービスも停止します）。




SQL Server サービスの停止が完了したら、**すべてのデータベースのデータ ファイル**（**.mdf**）と**トランザクション ログ ファイル**（**.ldf**）を、ファイル コピーで取得します。




**SQL Server 2008** の場合は、既定では**システム データベース**は、次のフォルダーに格納されています。

**C:¥Program Files¥Microsoft SQL Server¥MSSQL10.MSSQLSERVER¥MSSQL¥DATA**

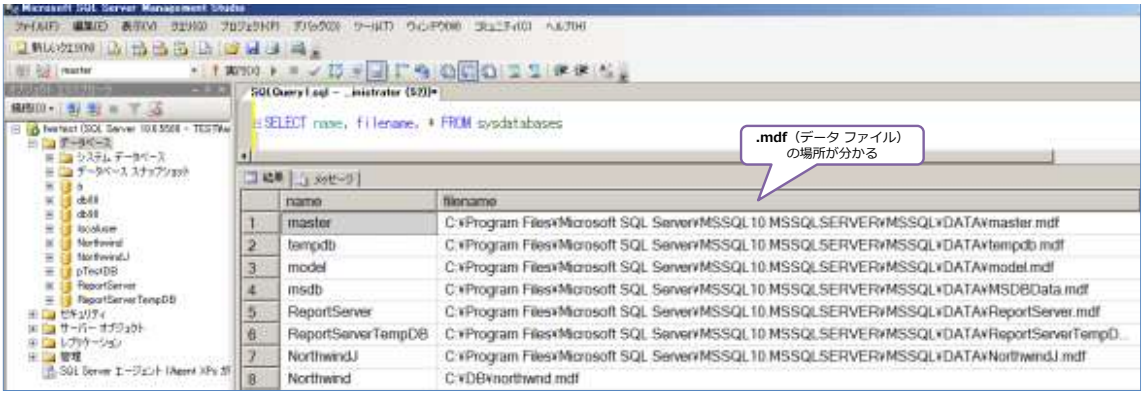
**SQL Server 2008 R2／2012／2014** の場合は、既定では次のフォルダーになります。

SQL Server 2008 R2 の場合
**C:¥Program Files¥Microsoft SQL Server¥MSSQL10_50.MSSQLSERVER¥MSSQL¥DATA**

SQL Server 2012 の場合
**C:¥Program Files¥Microsoft SQL Server¥MSSQL11.MSSQLSERVER¥MSSQL¥DATA**

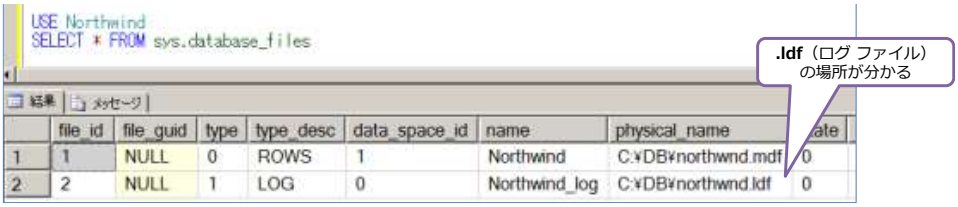
SQL Server 2014 の場合
**C:¥Program Files¥Microsoft SQL Server¥MSSQL12.MSSQLSERVER¥MSSQL¥DATA**

システム データベースを含めた、ユーザー データベースが格納されているフォルダーは、次のように **sysdatabases** 互換ビューの **filename** 列を参照することで確認することができます。




ここにリストされるデータベースのファイル（**.mdf**）とそれに対応したログ ファイル（**.ldf**）をす

べてコピーするようにします。このビューでは、**.ldf** ファイルの場所が分からないので、これを確認したい場合には、次のように **sys.database_files** システム ビューの **physical_name** 列を参照するようにします。




| | file_id | file_guid | type | type_desc | data_space_id | name | physical_name | ate |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | NULL | 0 | ROWS | 1 | Northwind | C:¥DB¥northwnd.mdf | 0 |
| 2 | 2 | NULL | 1 | LOG | 0 | Northwind_log | C:¥DB¥northwnd.ldf | 0 |

以上で、アップグレード前のバックアップ作業が完了です。

万が一の際に、システム データベースをオフライン バックアップから復元する際には、現在と全く同じ構成の OS／SQL Server であることが重要です。**OS の場合**は、参加するドメイン、マシン名、ドライブ／フォルダー構成、NTFS アクセス許可、共有フォルダー構成、**SQL Server の場合**は、インストール先のパスや照合順序、サービス アカウントを同じ構成にしておくこと、同じ修正プログラム（Service Pack や CU）をインストールしておくことが重要になります。

フォルダー構成やインストール先のパスが変わってしまうと、オフライン バックアップを元に戻しても、SQL Server が正しく起動しなくなってしまうので注意してください。このあたりの OS を入れ替えた場合の注意点は、次の章で説明するハードウェア リプレース時の丸ごと複製の作業と全く同じ考え方、対処方法になるので、次の章もぜひご覧いただければと思います。

もし、サービス アカウントに **Windows のローカル ユーザー**を設定している場合は、OS を入れ替えることによって、同じ名前のユーザーを作成しても、異なる Security ID（SID）が割り当てられてしまうので、（内部的には）別のユーザーとして見なされてしまいます。こうなると、サービス マスター キーでエラーが出ることになるので、こういった場合には、前述の **ALTER SERVICE MASTER KEY** を利用してサービス マスター キーを強制的に作成し直す（**REGENERATE** する）といったことも必要になってきます。

また、**Windows のローカル ユーザー**を利用している場合には、特にセキュリティ関連の設定で影響が出てくるところがいくつかあるので、これらについては、次の章もぜひご覧いただければと思います。

なお、OS の再インストールが発生するような事態を避けるには、サードパーティ製のイメージ バックアップ ツール（ストレージ レベルで丸ごとバックアップ可能なソフト）を利用したり、仮想マシン環境であれば仮想マシンを丸ごとバックアップしておいたりするのも 1 つの方法です。これであれば、同じ構成の OS／SQL Server を簡単に戻せます。

いずれにしても、同一マシンでアップグレードを行う場合は、万が一の失敗時における、ロールバックにかかる時間も考慮して、アップグレード時間を見積もっておくことが重要になります。

## 3.17 アップグレードにかかる時間の見積もり

SQL Server 2016 へのアップグレードにかかる時間は、環境によって大きく異なりますが（特に OS をアップグレードするかどうか）、次のように見積もることができます。

### OS に Windows Server 2008／2008 R2 を利用している場合

OS に **Windows Server 2008／2008 R2** を利用している場合は、SQL Server 2016 にアップグレードするためには、OS を **Windows Server 2012** 以上にアップグレードする必要があるので、その分作業時間がかかります。作業時間の見積もりは、次のとおりです。

| | アップグレード作業 | 当日の作業時間の見積もり | 説明 |
|---|---|---|---|
| **0** | アップグレード前のバックアップ作業 | 大きく変動 | : |
| **1** | Windows Server 2008 の **SP2**、Windows Server 2008 R2 の **SP1** がインストールされていない場合は、**SP1** をインストールする | 20分～1時間 | ∵Windows Server 2008 SP2 は、Windows Server 2012 にアップグレードするための必須条件、Windows Server 2008 R2 SP1 は Windows Server 2012／2012 R2 にアップグレードするための必須要件であるため。<br>作業時間は、ハードウェア環境によって作業時間が前後する。OS の再起動に 10分程度かかるサーバー機もあることに注意 |
| **2** | SQL Server を動作させるための Service Pack（SP）要件を確認して、SP がインストールされていない場合は、SP をインストールする | 10分～1時間 | OS を Windows Server 2012 や 2012 R2 にアップグレードする前に、SQL Server を動作させるための Service Pack 要件を確認して、要件を満たしていない場合には、Service Pack を適用する。<br>Windows Server 2012 なら<br>SQL Server 2008 は **SP3**、2008 R2 は **SP1** が必須<br>Windows Server 2012 R2 なら<br>SQL Server 2008 は **SP3**、2008 R2 は **SP2**、2012 は **SP1** 必須 |
| **3** | OS を Windows Server 2008／2008 R2 から Windows Server 2012／2012 R2 にアップグレードする | 20分～1時間 | Windows Server 2012 R2 は **with Update のパッケージ**を利用することで、OS と KB 2919355（SQL Server 2016 の必須要件）のアップグレードを同時に行うことが可能。with Update ではない場合は、OS のインストール後に KB 2919355 を適用する必要がある。<br>作業時間は、ハードウェア環境によって作業時間が前後する。OS の再起動に 10分程度かかるサーバー機もあるため、機種によっては 1 時間以上になることも。 |
| **4** | Data Migration Assistant による互換性のチェックを行う | - | : |
| **5** | SQL Server 2016 のアップグレードに必要な SP がインストールされていない場合は、SP をインストールする | 10分～1時間 | SQL Server 2008 は **SP4**<br>SQL Server 2008 R2 は **SP3**<br>SQL Server 2012 は **SP2** が必須条件。<br>SQL Server 2014 の場合は SP なしでも大丈夫 |
| **6** | SQL Server 2016 へのアップグレード インストールを行う | 15分～1時間30分 | : |
| **7** | SQL Server 2016 の最新の修正プログラム（CU や SP）をインストールする | 15分～1時間30分 | SP1 以降を適用しておくことをお勧めします。<br>Reporting Services を利用する場合は、SP1 に対する重要な更新プログラム（KB3207512）があるので CU4（SP1+CU1）以上を推奨 |
| **8** | Management Studio の最新版をインストールする（オプション） | 10分～1時間 | 事前にインストーラーをダウンロードしておくことで<br>インストール時間を短縮できる |
| **9** | 統計（Statistics）を更新する | 数分～ | : |
| **10** | データベースの互換性レベルを 130 へ上げる（オプション） | 数分 | 必要に応じて、クエリストア機能を利用して、<br>互換性レベルの影響（特に性能面）をチェックする |
| **11** | 動作チェックを行う | 1時間～ | : |
| **12** | SSDT のインストール（オプション） | 10分～1時間 | 事前にダウンロード版のインストーラー（iso）をダウンロードしておくことで、インストール時間を短縮できる |
| **99** | OS を Windows Server 2016 へアップグレードする（オプション） | 20分～2時間 | : |
| **99** | 万が一の失敗時のロールバック作業 | 大きく変動 | : |

**Windows Server 2008** を利用している場合は、Windows Server 2012 にアップグレードするために、Windows Server 2008 **SP2** が必須要件になります（Windows Server 2008 は Windows Server 2012 R2 にはアップグレードできません）。

**Windows Server 2008 R2** を利用している場合は、Windows Server 2012 または Windows Server 2012 R2 にアップグレードするために、Windows Server 2008 R2 **SP1** が必須要件になります（Windows Server 2008 R2 は Windows Server 2016 にはアップグレードできません）。

なお、Windows Server 2016 にアップグレードするには、Windows Server 2012／2012 R2 が必要になります。

こうした **Windows Server**（OS）の**アップグレード条件**をまとまると、次のようになります。

**Windows Server のアップグレード要件**

| | アップグレード先 | | |
|---|---|---|---|
| | **Windows Server 2012** | **Windows Server 2012 R2** | **Windows Server 2016** |
| Windows Server 2003／2003 R2 | × 不可 | × 不可 | × 不可 |
| Windows Server 2008 | **SP2** 以降が必要 | × 不可 | × 不可 |
| Windows Server 2008 R2 | **SP1** 以降が必要 | **SP1** 以降が必要 | × 不可 |
| Windows Server 2012 | - | RTM で OK | RTM で OK |
| Windows Server 2012 R2 | - | - | RTM で OK |

各詳細については、以下のドキュメントが参考になります。

Windows Server 2012 の評価版と更新プログラムのオプション
http://msdn.microsoft.com/ja-jp/library/jj574204(v=ws.11).aspx

Windows Server 2012 R2 のアップグレード オプション
http://technet.microsoft.com/ja-jp/library/dn303416.aspx


Windows Server の以前の製品版から Windows Server 2012 R2 へのアップグレード

以下の表は、既にライセンスを取得している (つまり評価版でない) Windows オペレーティング システムを Windows Server 2012 R2 のどのエディションにアップグレードできるかを簡単にまとめたものです。

サポートされるパスに関しては、次の一般的なガイドラインが適用されます。

- 32 ビット アーキテクチャから 64 ビット アーキテクチャへの一括アップグレードはサポートされません。すべてのエディションの Windows Server 2012 R2 が 64 ビットのみです。
- 1 つの言語から別の言語への一括アップグレードはサポートされません。
- 1 つのビルドの種類から別のビルドの種類への一括アップグレード (fre から chk へなど) はサポートされません。
- サーバーがドメイン コントローラーの場合は、http://technet.microsoft.com/library/hh994618.aspx を参照して重要な情報をご確認ください。
- Windows Server 2012 R2 のプレリリース版からのアップグレードはサポートされません。Windows Server 2012 R2 のクリーン インストールを実行してください。
- Server Core インストールから Windows Server 2012 R2 のフル インストール モードへの切り替え (およびその逆方向の切り替え) を 1 回の操作で行うアップグレードはサポートされていません。ただし、いったんアップグレードが完了すると、Windows Server 2012 R2 では、Server Core モードとフル インストール モードを自由に切り替えることができます。各インストール オプションの内容、各オプション間での変換方法、新しい最小サーバー インターフェイスとオンデマンド機能の使用方法については、http://technet.microsoft.com/library/hh831786 を参照してください。

左の列に現在のバージョンがない場合、このリリースの Windows Server 2012 R2 へのアップグレードはサポートされません。

右の列に複数のエディションが表示されている場合、同じ開始バージョンから各エディションにアップグレードできます。

Windows Server 2012 R2 へのアップグレード要件

| 使用している Windows オペレーティング システム | アップグレード先のエディション |
|---|---|
| Windows Server 2008 R2 Datacenter SP1 | Windows Server 2012 R2 Datacenter |
| Windows Server 2008 R2 Enterprise SP1 | Windows Server 2012 R2 Standard または Windows Server 2012 R2 Datacenter |
| Windows Server 2008 R2 Standard SP1 | Windows Server 2012 R2 Standard または Windows Server 2012 R2 Datacenter |
| Windows Web Server 2008 R2 SP1 | Windows Server 2012 R2 Standard |
| Windows Server 2012 Datacenter | Windows Server 2012 R2 Datacenter |
| Windows Server 2012 Standard | Windows Server 2012 R2 Standard または Windows Server 2012 R2 Datacenter |

Windows Server 2016 のアップグレード オプションと変換オプション
http://technet.microsoft.com/ja-jp/windows-server-docs/get-started/supported-upgrade-paths

OS を Windows Server 2012 以上にアップグレードする前には、SQL Server の **Service Pack 要件**（以下）も満たしておく必要があります。

**SQL Server を動作させるために必要となる Service Pack**

| | Windows Server 2012 | Windows Server 2012 R2 | Windows Server 2016 |
|---|---|---|---|
| SQL Server 2008 | SP3 | SP3 | 未サポート |
| SQL Server 2008 R2 | SP1 | SP2 | 未サポート |
| SQL Server 2012 | RTM | SP1 | SP2 |
| SQL Server 2014 | RTM | RTM | SP1 |

**SQL Server 2016 にアップグレードするための Service Pack 要件（前掲）**

| | |
|---|---|
| SQL Server 2008 | SP4 |
| SQL Server 2008 R2 | SP3 |
| SQL Server 2012 | SP2 |
| SQL Server 2014 | RTM |

**Windows Server 2012** にアップグレードする前には、**SQL Server 2008** を利用している場合には **SP3** 以上（SQL Server 2016 にアップグレードするためには **SP4** 以上）、**SQL Server 2008 R2** の場合には **SP1** 以上（SQL Server 2016 にアップグレードするためには **SP3** 以上）を適用しておく必要があります。

したがって、SQL Server 2008 の SP1 や SP2、SQL Server 2008 R2 の RTM などで運用している環境を SQL Server 2016 にアップグレードする場合には、SQL Server 2016 へのアップグレード要件である SP（SQL Server 2008 は SP4、SQL Server 2008 R2 は SP3）を、OS のアップグレードする前に適用しておくと、二度手間にならずに済みます（SP を適用する時間の短縮＝アップグレード時のダウンタイムの削減に繋がります）。

なお、Windows Server 2016 では、SQL Server 2008 および 2008 R2 は未サポートになっています。これらの Windows Server と SQL Server の対応状況については、以下の KB 2681562 が参考になります。

Using SQL Server in Windows 8 and later versions of Windows operating system
http://support.microsoft.com/en-us/kb/2681562

## OS に Windows Server 2012／2012 R2 を利用している場合

OS に Windows Server 2012／2012 R2 を利用している場合は、次のようになります。

アップグレード時間の見積り

| | アップグレード作業 | 当日の作業時間の見積もり | 説明 |
|---|---|---|---|
| **0** | アップグレード前のバックアップ作業 | 大きく変動 | : |
| **1** | Windows Server 2012 R2 を利用している場合は、**KB 2919355**（SQL Server 2016 の必須要件）をインストールする | 20分～1時間 | : |
| **2** | Data Migration Assistant による互換性のチェックを行う | - | : |
| **3** | SQL Server 2016 のアップグレードに必要な SP がインストールされていない場合は、SP をインストールする | 10分～<br>1時間 | SQL Server 2008 は **SP4**<br>SQL Server 2008 R2 は **SP3**<br>SQL Server 2012 は **SP2** が必須条件。<br>SQL Server 2014 の場合は SP なしでも大丈夫 |
| **4** | SQL Server 2016 へのアップグレード インストールを行う | 15分～<br>1時間30分 | : |
| **5** | SQL Server 2016 の最新の修正プログラム（CU や SP）をインストールする | 15分～<br>1時間30分 | SP1 以降を適用しておくことをお勧めします。<br>Reporting Services を利用する場合は、SP1 に対する重要な更新プログラム（KB3207512）があるので CU4（SP1+CU1）以上を推奨 |
| **6** | Management Studio の最新版をインストールする（オプション） | 10分～1時間 | 事前にインストーラーをダウンロードしておくことで<br>インストール時間を短縮できる |
| **7** | 統計（Statistics）を更新する | 数分～ | : |
| **8** | データベースの互換性レベルを 130 へ上げる（オプション） | 数分 | 必要に応じて、クエリストア機能を利用して、<br>互換性レベルの影響（特に性能面）をチェックする |
| **9** | 動作チェックを行う | 1時間～ | : |
| **10** | SSDT のインストール（オプション） | 10分～1時間 | 事前にダウンロード版のインストーラー（iso）をダウンロードしておくことで、インストール時間を短縮できる |
| **99** | OS を Windows Server 2016 へアップグレードする（オプション） | 20分～2時間 | : |
| **99** | 万が一の失敗時のロールバック作業 | 大きく変動 | : |

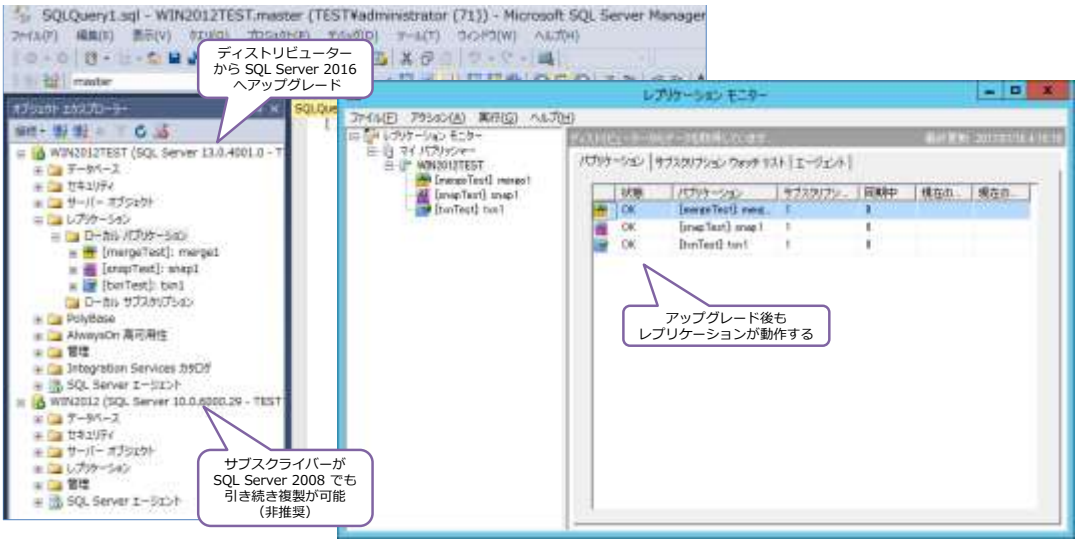
Windows Server 2012 R2 を利用している場合は、SQL Server 2016 をインストールするための必須要件である **KB 2919355** を適用しておくことが必要になります。なお、この KB の適用には OS の再起動も伴うので、サーバー機の場合は OS の再起動に数分の時間がかかるものも多いので、そういった時間も考慮しておく必要があります。

## 3.18 レプリケーション、ログ配布、DBM などのアップグレード

**レプリケーション**や**ログ配布**、**データベース ミラーリング**、**可用性グループ**といったサーバー間の連携機能や、**WSFC**（Windows Server フェールオーバー クラスタリング）上の SQL Server インスタンスを利用している場合も、SQL Server 2016 へのアップグレード インストールによって、アップグレード後にも設定が引き継がれます。**レプリケーション**に関しては**ディストリビューター**からアップグレードし、**ログ配布**や**データベース ミラーリング**、**可用性グループ**、**WSFC のインスタンス**に関してはセカンダリ（スタンバイ）側からアップグレードするようにします。

### レプリケーションのアップグレードの場合

**レプリケーション**に関しては、**ディストリビューター**として構成しているマシン（既定ではパブリッシャーと同じマシン）から SQL Server 2016 へアップグレードすることで、基本的にはレプリケーション構成をそのままアップグレードすることができます（レプリケーション機能は、下位バージョンへのレプリケートも可能なため、サブスクライバーが SQL Server 2008 や SQL Server 2008 R2 でも、引き続き複製することができます）。




ただし、**SQL Server 2008／2008 R2** へのレプリケーションに関しては、SQL Server 2016 では**非推奨機能**になっています（∵将来のバージョンで削除される予定の機能のため）。

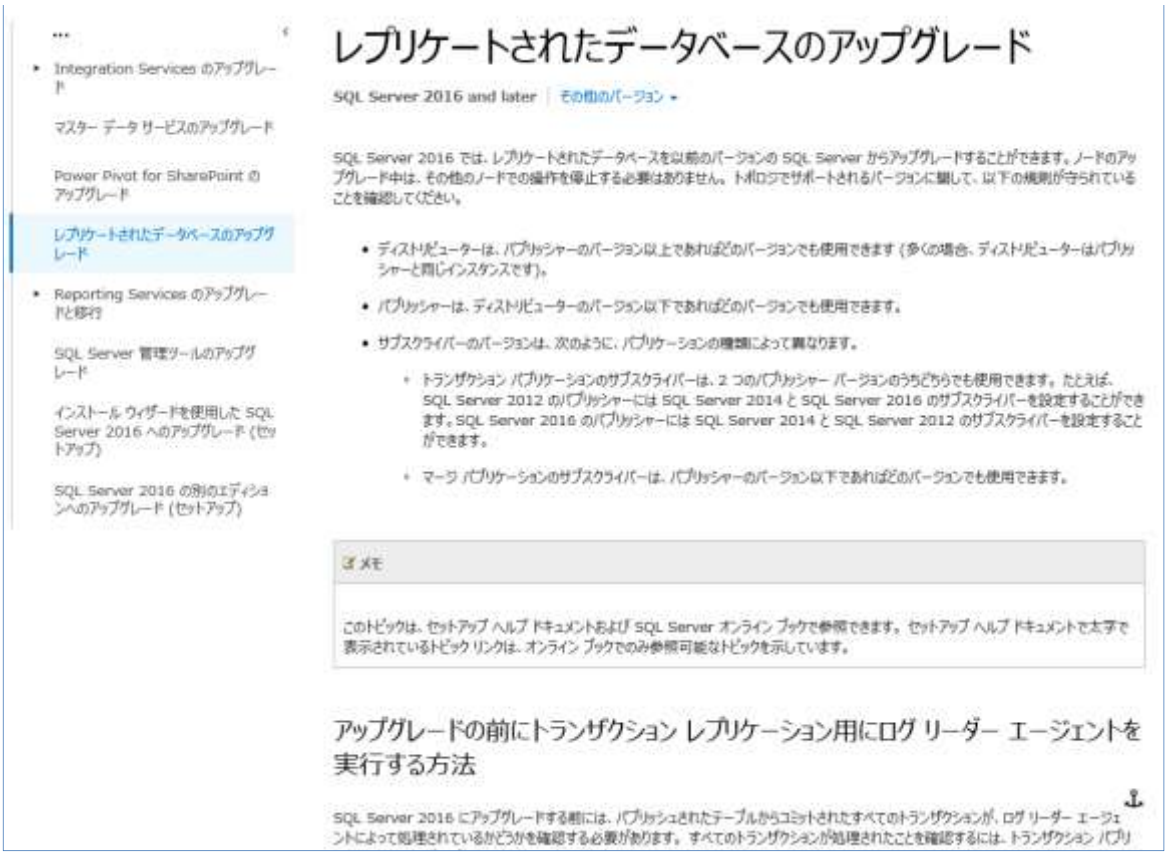
SQL Server レプリケーションの非推奨機能
http://msdn.microsoft.com/ja-jp/library/ms143550.aspx

レプリケーションに関しては、以下のトピックも一読しておくことをお勧めします。

レプリケーションの旧バージョンとの互換性
http://msdn.microsoft.com/ja-jp/library/ms143323.aspx

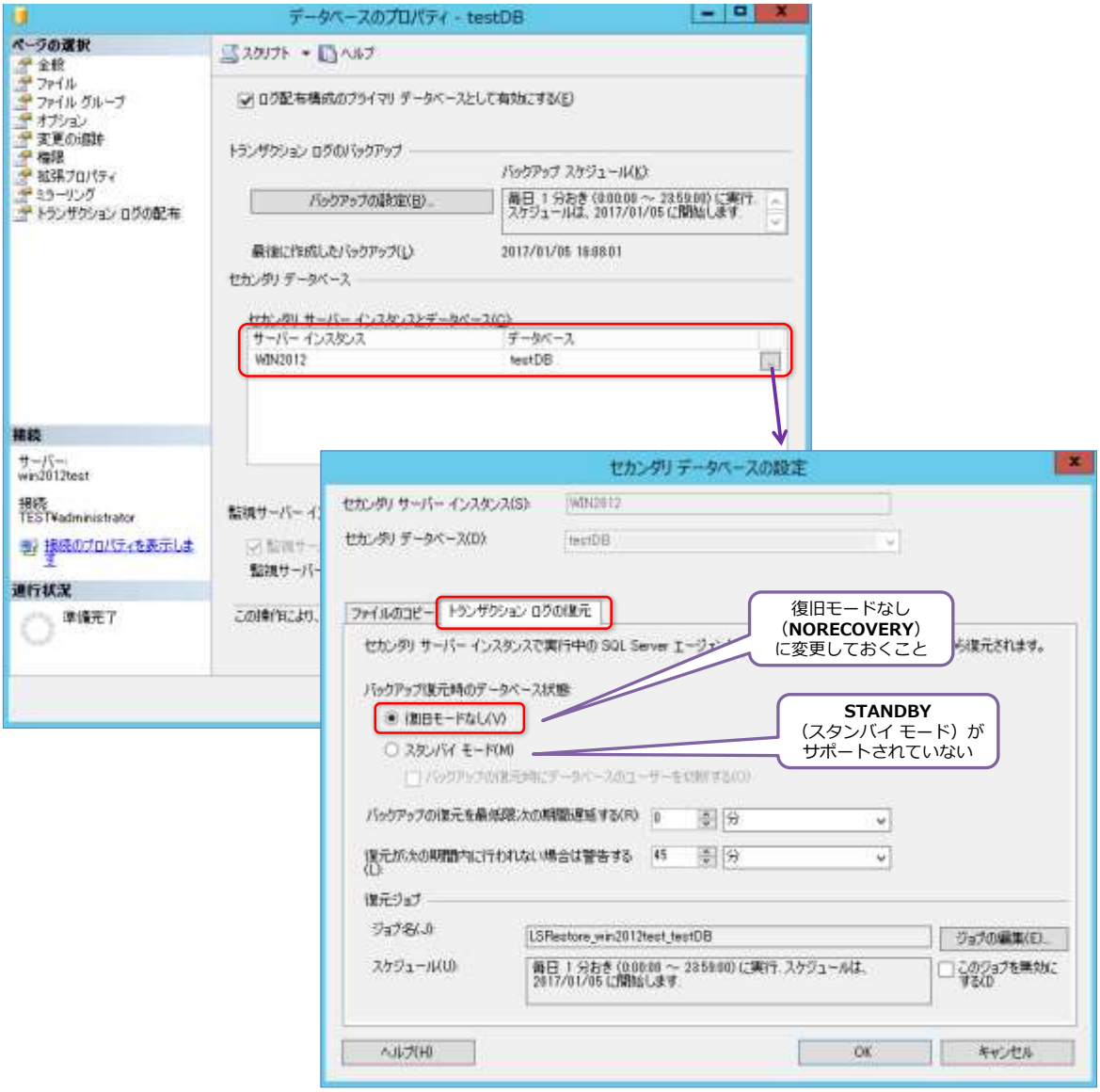
SQL Server レプリケーションにおける重大な変更
http://msdn.microsoft.com/ja-jp/library/ms143470.aspx


レプリケートされたデータベースのアップグレード
http://msdn.microsoft.com/ja-jp/library/ms143699.aspx



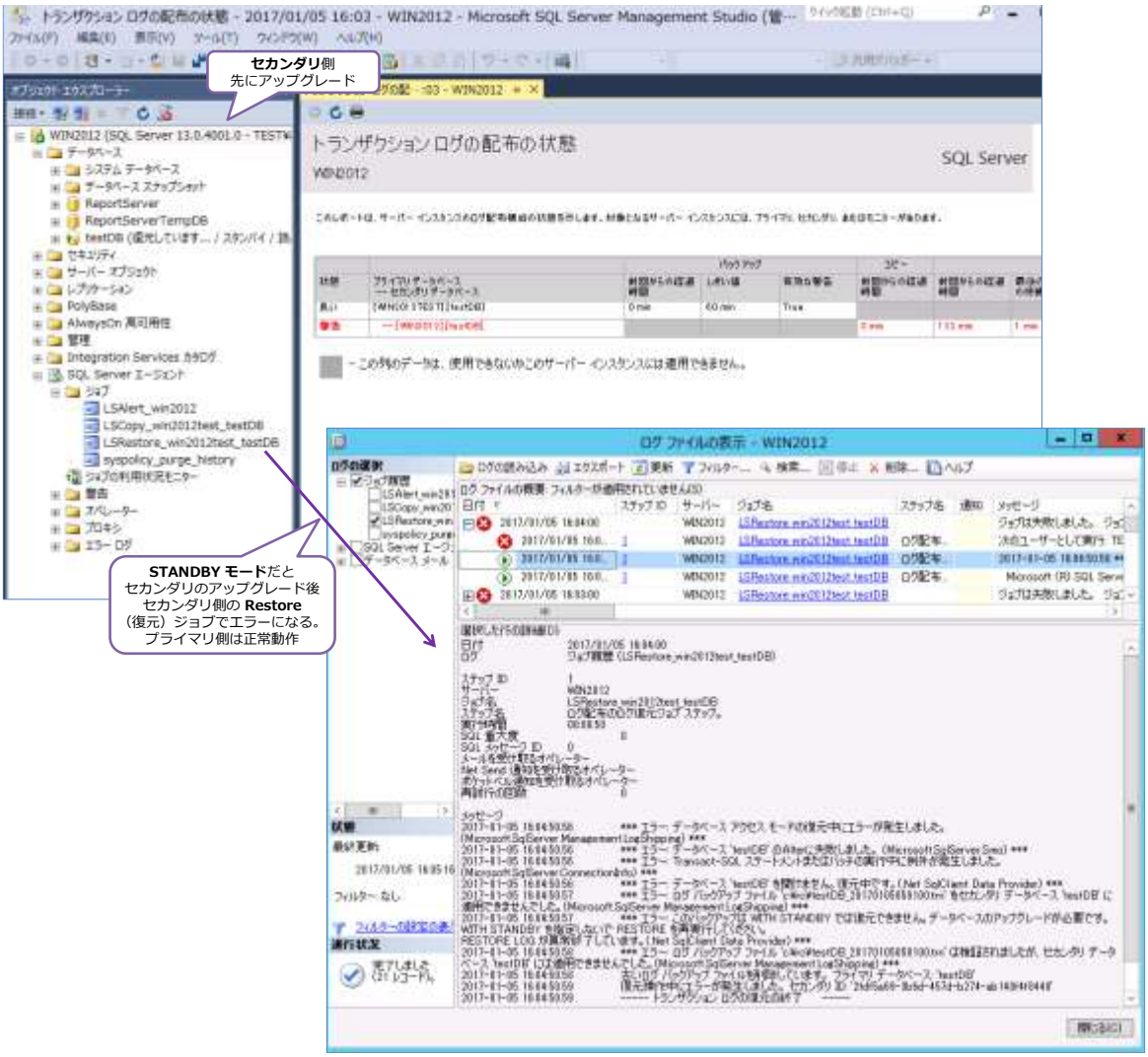
## 3.19 ログ配布のアップグレード

**ログ配布**は、**セカンダリ**（**スタンバイ サーバー**）側から先にアップグレードすることで、以前のバージョンで利用していたログ配布の設定を引き続き利用することができます。セカンダリをアップグレード中でも、（旧バージョンの）プライマリ サーバーは、引き続き利用することができます（ダウンタイムを最小限に抑えられます）。注意点としては、アップグレード中は、復旧状態オプションの **STANDBY**（スタンバイ）がサポートされていないことです。したがって、アップグレードを行う前に、セカンダリの［**トランザクション ログの復元**］タブで、次のように［**復旧モードなし**］（**NORCOVERY**）へ変更しておく必要があります（以下の画面は SQL Server 2008 ですが、他のバージョンでも同じように操作できます）。




この設定は、両方のサーバーのアップグレードが完了した後に、元に戻す（**STANDBY** に戻す）ことができます。

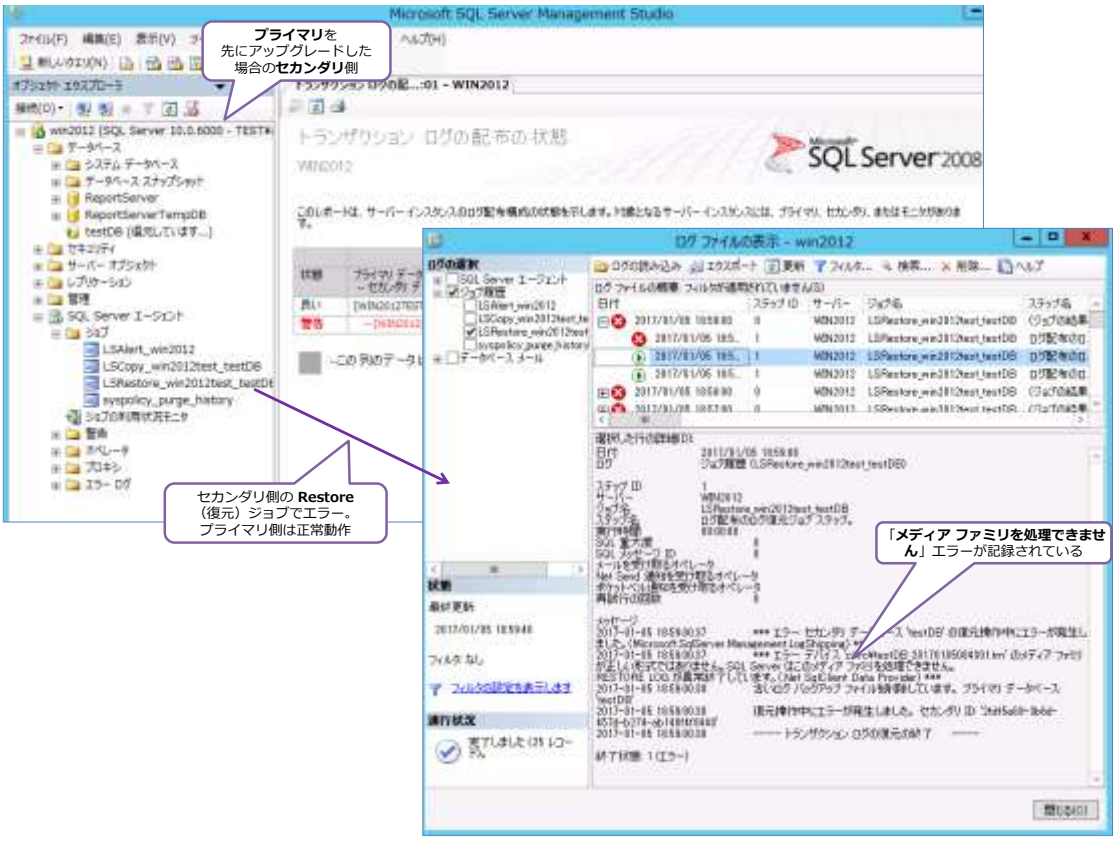
もし、**STANDBY** のまま、セカンダリのアップグレードを行った場合は、次のように **Restore**（復元）ジョブでエラーになってしまいます。




このエラーを解消するには、プライマリ側のログ配布の設定で、**STANDBY** から **NORECOVERY**（復旧モードなし）へ変更するようにします（変更後、次にジョブが定期実行されるタイミングでエラーが解消されます）。このように、ログ配布をアップグレードするにあたっては、STANDBY を利用している場合は、一度、NORECOVERY へ変更しておく必要があります（このエラーのようにアップグレードの実行した後ではなく、実行する前に変更しておくことをお勧めします）。

## プライマリからアップグレードしてしまった場合

もし、ログ配布をプライマリからアップグレードしてしまった場合は、次のようにセカンダリの Restore ジョブでエラーが記録されてしまいます。




メッセージには「**メディア ファミリを処理できません**」と記録されていて、アップグレードしたプライマリ（SQL Server 2016）側のバックアップ ファイルの形式が、セカンダリ側で認識できないために発生しています。これを解消するには、セカンダリも SQL Server 2016 へアップグレードします。アップグレードが完了した後は、（バックアップ ファイルを認識できるようになるので）Restore ジョブが正常に動作するようになり、ログ配布を以前と同じように利用していくことができます。

## ローリング アップグレードを行う場合

セカンダリを SQL Server 2016 へアップグレードした後に、このセカンダリをプライマリへロール変更することで、さらにダウンタイムを最小限にすることもできます。新プライマリがユーザーからの処理を引き続き処理できるようになり、その間に旧プライマリ（現在のセカンダリ）を SQL Server 2016 へアップグレードするという形です。この方法については、オンライン ブックの以下のトピックに記載されているので、参考になると思います。

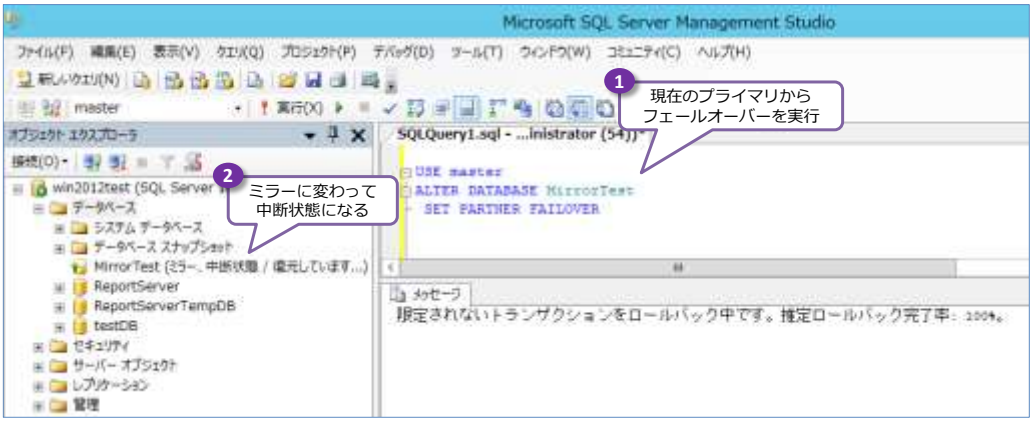
SQL Server 2016 へのログ配布のアップグレード
http://msdn.microsoft.com/ja-jp/library/cc645954.aspx

## 3.20 DBM のアップグレード（ローリング アップグレード）

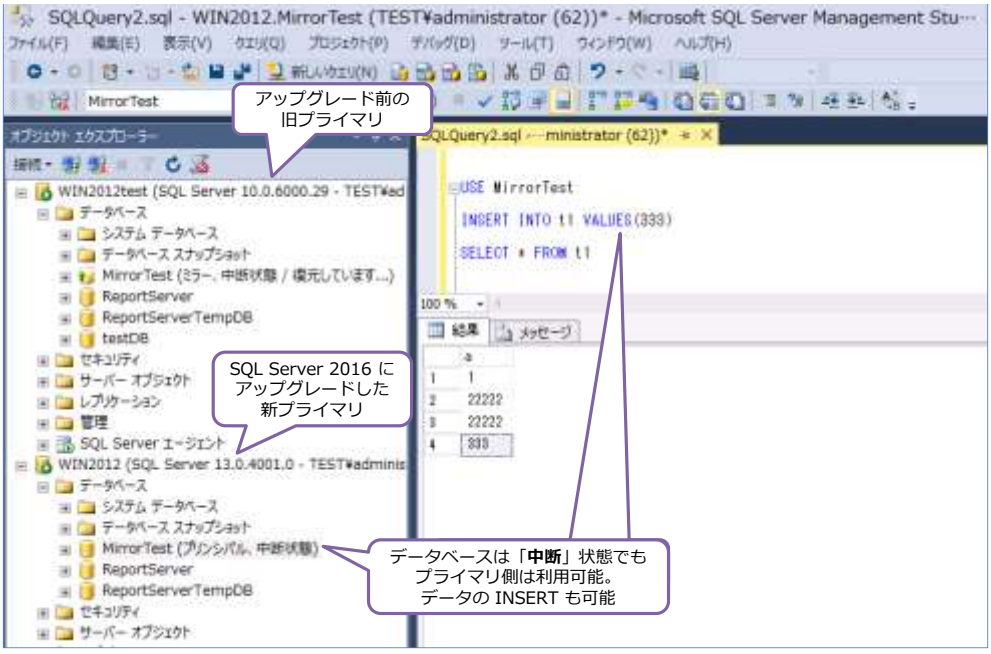
**データベース ミラーリング**（**DBM**）は、**セカンダリ**（**ミラー**）側から先にアップグレードすることで、**ローリング アップグレード**をすることができます。セカンダリをアップグレード中でも、（旧バージョンの）プライマリ（プリンシパル）は、引き続き利用することができます。

**セカンダリ**を **SQL Server 2016** へアップグレードした後は、セカンダリをプライマリへ**フェールオーバー**（ロール変更）します。これは、次のように **ALTER DATABASE** ステートメントを実行します（画面は SQL Server 2008 ですが、他のバージョンでも同じように操作できます）。

```
USE master
ALTER DATABASE データベース名
 SET PARTNER FAILOVER
```



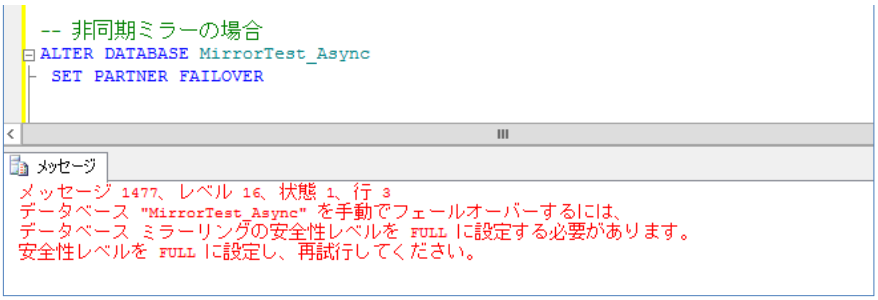

これは、**現在のプライマリ**（アップグレードする前のサーバー）側から実行します。これを実行すると、プライマリがミラーへ変わって、データベースが中断状態になりますが、代わりに SQL Server 2016 へのアップグレードを行ったセカンダリ（**新しいプライマリ**）側のデータベースが利用できる状態になります。



したがって、ユーザーから見れば、ダウンタイムは、この**フェールオーバー**を実行している間（役割変更が完了するまでの間）だけです。フェールオーバーにかかる時間は、その間に実行されていたトランザクション量にもよりますが、トランザクション量が少なければ数秒で完了します。

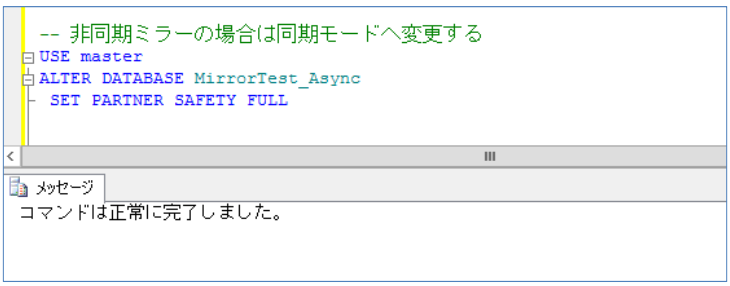
### 非同期モードの場合 ～いったん同期モードへ変更～

データベース ミラーリングを**非同期モード**（高パフォーマンス）で構成している場合には、上記のフェールオーバー（ロール変更）を実行しようとするときに、次のようにエラーとなります。




非同期モードでは、ロール変更を行うには、いったん**同期モード**へ変更する必要があります。これは、次のように「**SET PARTNER SAFETY FULL**」を実行することで行えます。

```
USE master
ALTER DATABASE データベース名
 SET PARTNER SAFETY FULL
```



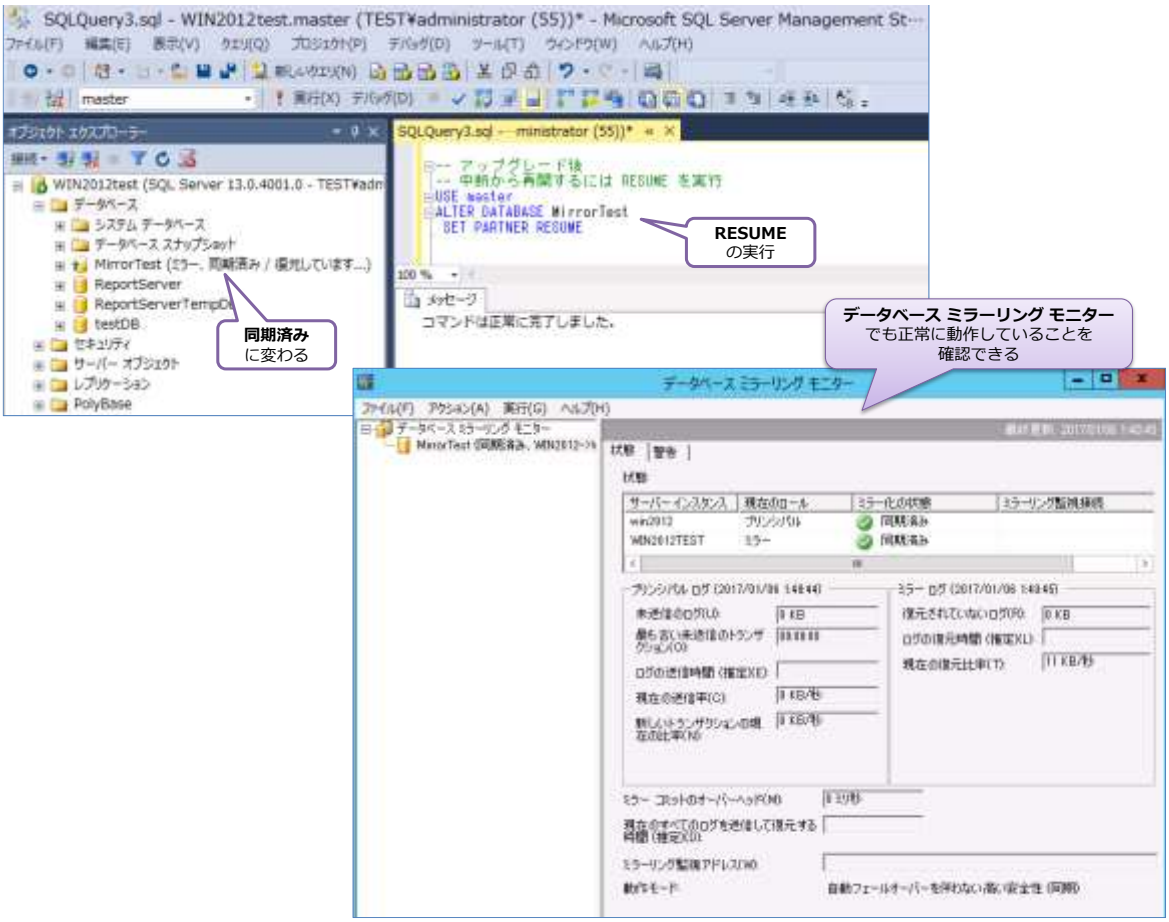

この後、もう一度、「**SET PARTNER FAILOVER**」を実行すれば、ロール変更することができます。

### 旧プライマリ（現在のセカンダリ）の SQL Server 2016 へのアップグレード、RESUME

ロール変更が完了したら、次は旧プライマリ（現在のセカンダリ）を **SQL Server 2016** へアップグレードします。

アップグレードが完了した後は、データベースの状態が「**中断**」になったままなので、これを「**RESUME**」（**再開**）します。RESUME を行うには、次のように「**SET PARTNER RESUME**」を実行します。

```
USE master
ALTER DATABASE データベース名
 SET PARTNER RESUME
```




以上で、データベース ミラーリングを以前のバージョンで利用していたのと同じように利用できるようになります（この後、役割を元に戻したい場合にはフェールオーバーを実行し、同期モードを非同期へ変更したい場合には SET SAFETY OFF を実行します）。このように、**ローリング アップグレード**を利用することで、ダウンタイムを非常に小さくすることができます。

データベース ミラーリングのアップグレードに関しては、オンライン ブックの以下のトピックも一読しておくことをお勧めします。

ミラー化されたインスタンスのアップグレード
http://msdn.microsoft.com/ja-jp/library/bb677181.aspx

## 3.21 AlwaysOn 可用性グループのローリング アップグレード

**AlwaysOn 可用性グループ**（SQL Server 2012 からの新機能）に関しても、データベース ミラーリングの場合とほとんど同じ操作で**ローリング アップグレード**を行うことができます。

手順の概要は、次のとおりです。

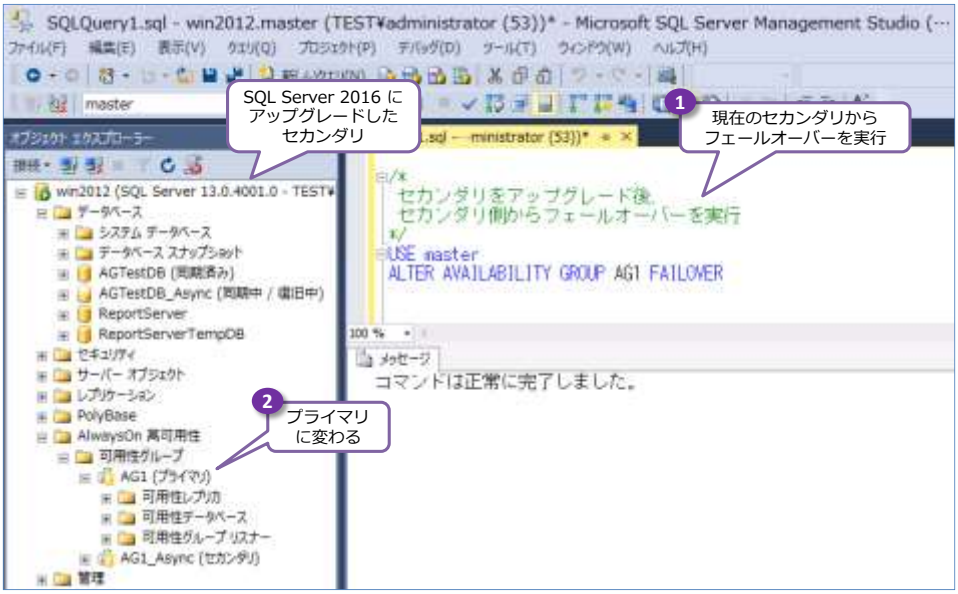
1. セカンダリを SQL Server 2016 へアップグレード
2. アップグレード完了後にロール変更（ダウンタイムはこの間のみ）
3. 旧プライマリを SQL Server 2016 へアップグレード
4. RESUME を実行

### ローリング アップグレードの手順

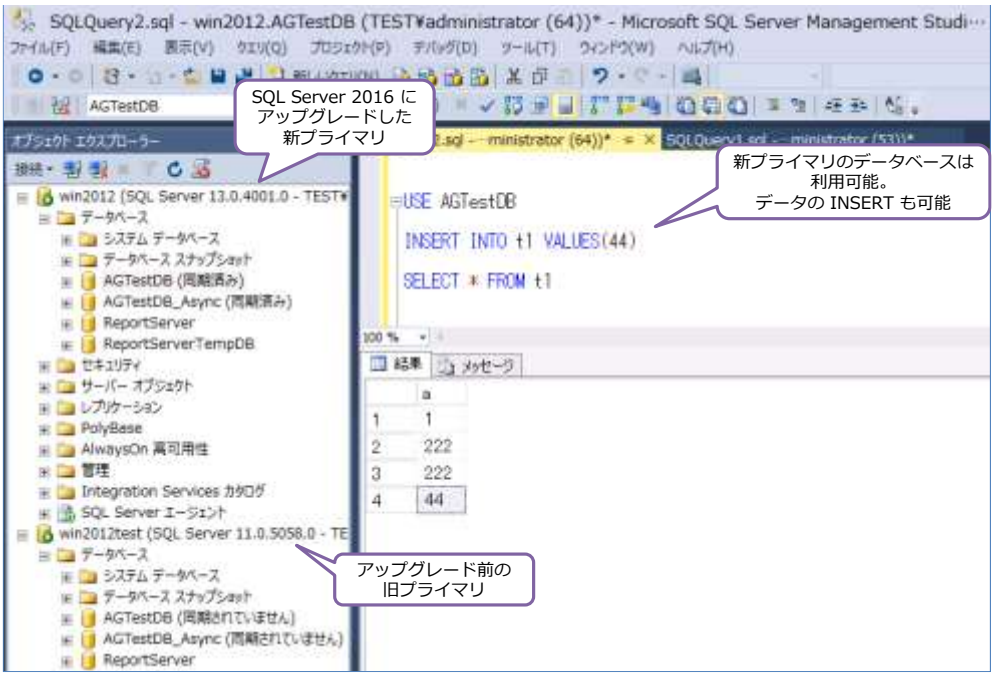
ローリング アップグレードを行うには、まず、**セカンダリ**を **SQL Server 2016** へアップグレードします。

アップグレードが完了したら、**フェールオーバー**（ロール変更）を実行して、セカンダリをプライマリへ変更します。これは、次のように **ALTER AVAILABILITY GROUP** ステートメントを実行します。

```
USE master
ALTER AVAILABILITY GROUP 可用性グループ名 FAILOVER
```




これは、**現在のセカンダリ**（アップグレード済みのサーバー）側から実行します。これを実行すると、SQL Server 2016 へのアップグレードを行ったセカンダリを**新しいプライマリ**へ変更して、データベースを利用できる状態になります。




したがって、ユーザーから見れば、ダウンタイムは、この**フェールオーバー**を実行している間（ロール変更が完了するまでの間）だけです。フェールオーバーにかかる時間は、その間に実行されていたトランザクション量にもよりますが、トランザクション量が少なければ数秒で完了します。

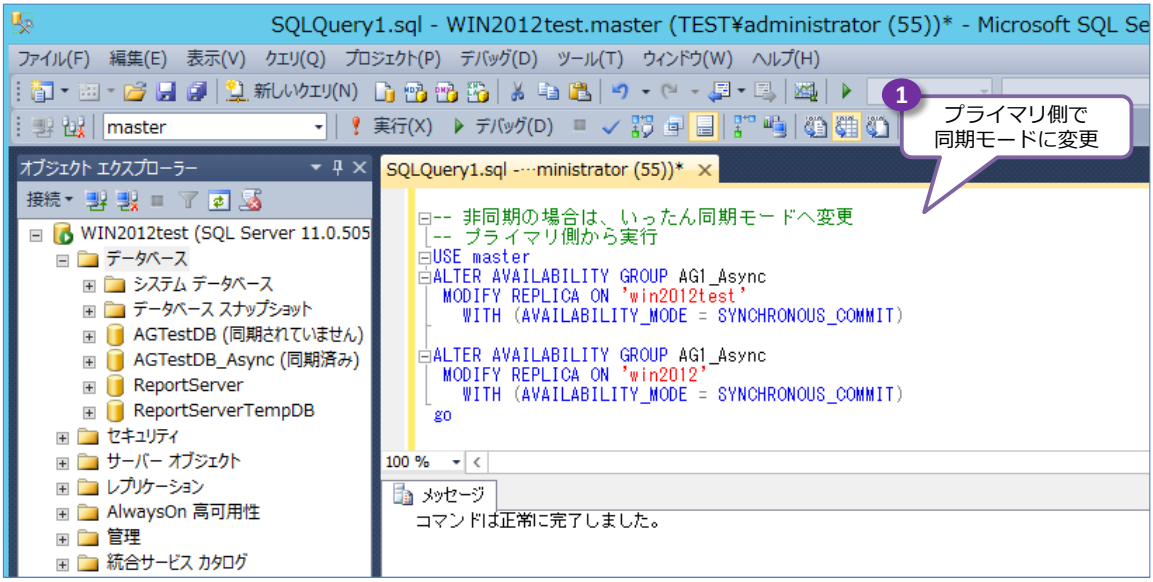
**非同期モードの場合 ～いったん同期モードへ変更～**

可用性グループを**非同期モード**で構成している場合には、上記のフェールオーバー（ロール変更）を実行しようとするときに、次のようにエラーとなります。


非同期モードでは、ロール変更を行うには、いったん**同期モード**へ変更する必要があります。これは、次のように **ALTER AVAILABILITY GROUP** ステートメントを実行することで行えます。

```
USE master
ALTER AVAILABILITY GROUP 可用性グループ名
 MODIFY REPLICA ON 'プライマリ名'
   WITH (AVAILABILITY_MODE = SYNCHRONOUS_COMMIT)

ALTER AVAILABILITY GROUP 可用性グループ名
 MODIFY REPLICA ON 'セカンダリ名'
   WITH (AVAILABILITY_MODE = SYNCHRONOUS_COMMIT)
```



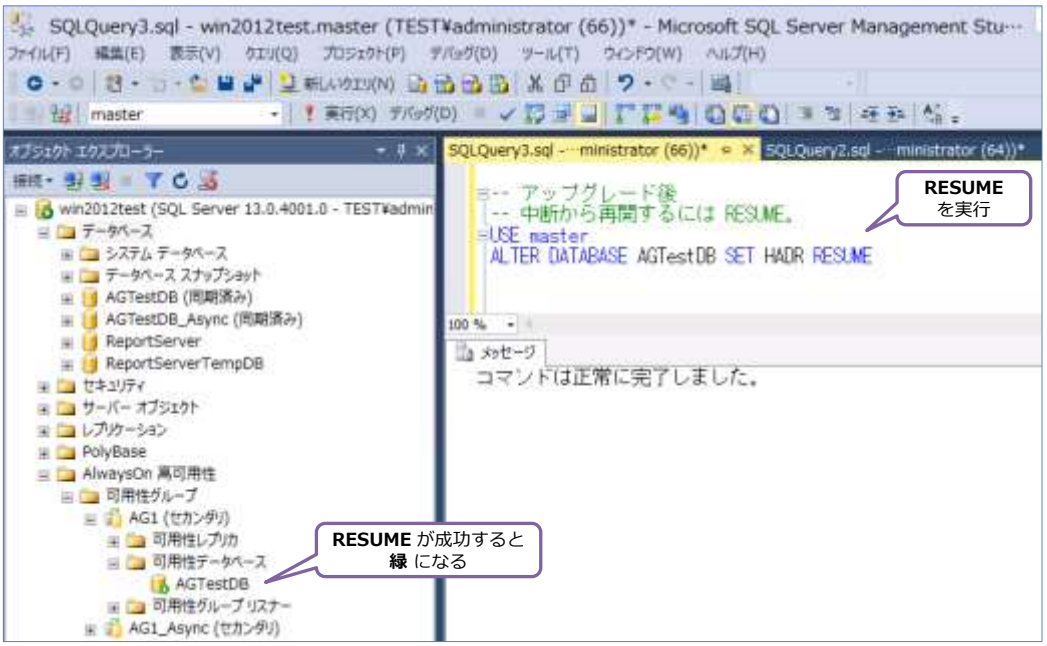

この後、もう一度、**ALTER AVAILABILITY GROUP** ステートメントを実行すれば、ロール変更することができます。

## 旧プライマリ（現在のセカンダリ）の SQL Server 2016 へのアップグレード、RESUME

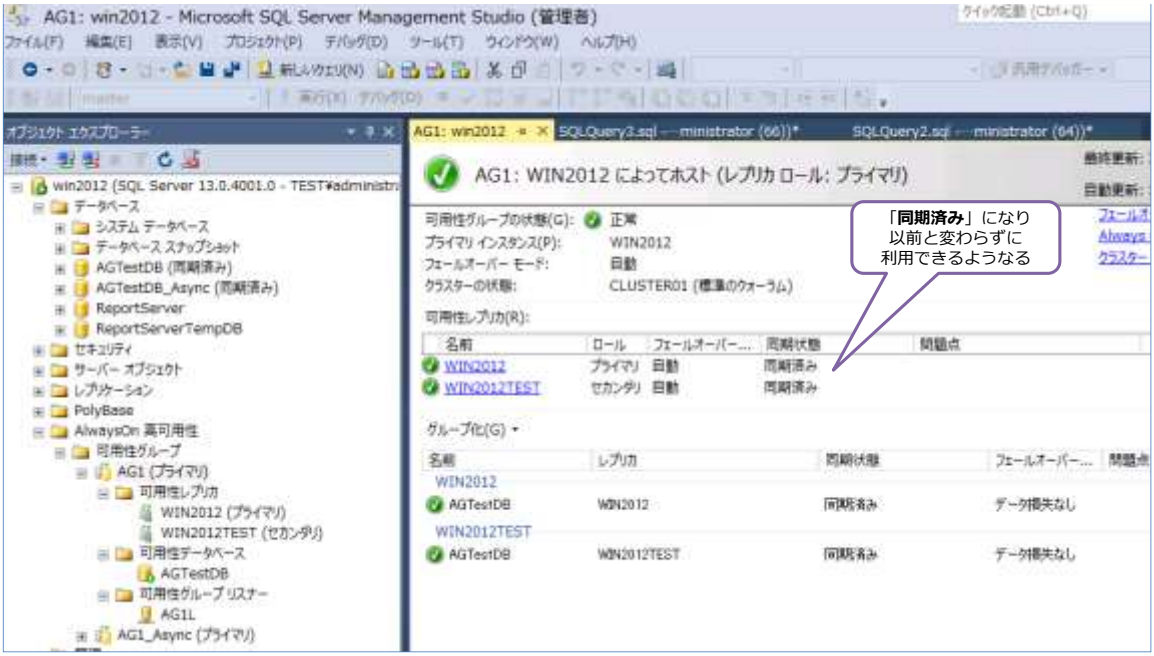
ロール変更が完了したら、次は旧プライマリ（現在のセカンダリ）を **SQL Server 2016** へアップグレードします。

アップグレードが完了した後は、データベースの状態が「**中断**」になっているので、これを「**RESUME**」（**再開**）します。RESUME を行うには、次のように「**SET PARTNER RESUME**」を実行します。

```
USE master
ALTER DATABASE データベース名 SET HADR RESUME
```



以上で、可用性グループを以前のバージョンで利用していたのと同じように利用できるようになります。




この後、役割を元に戻したい場合にはフェールオーバーを実行し、同期モードを非同期へ変更したい場合には ALTER AVAILABILITY GROUP ステートメントを実行します。

このように、**ローリング アップグレード**を利用することで、ダウンタイムを非常に小さくすることができます。

可用性グループのアップグレードに関しては、オンライン ブックの以下のトピックも一読しておくことをお勧めします。

AlwaysOn 可用性グループのレプリカ インスタンスのアップグレード
http://msdn.microsoft.com/ja-jp/library/dn178483.aspx

## 3.22 WSFC の SQL Server インスタンスのアップグレード

**WSFC**（Windows Server フェールオーバー クラスタリング）上の SQL Server インスタンス（通称 **FCI**：フェールオーバー クラスター インスタンス）も SQL Server 2016 へのアップグレード インストールが可能です。

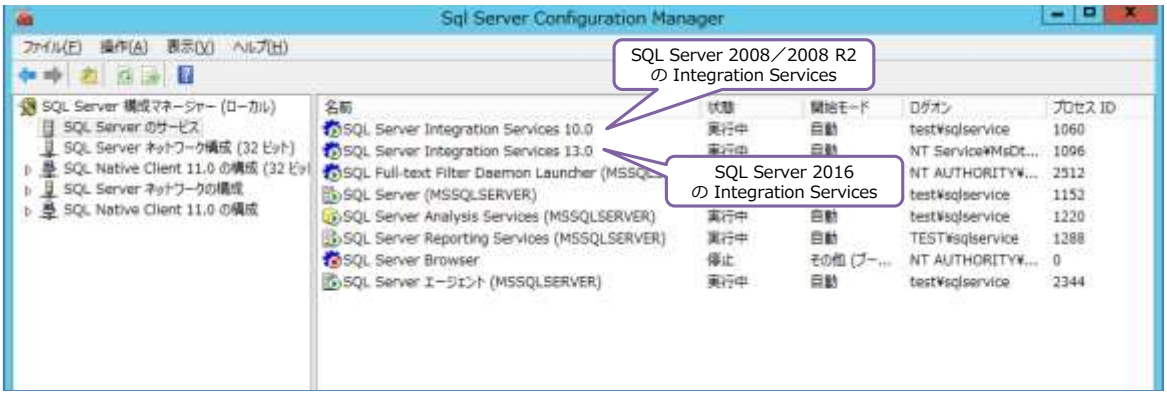
**FCI のアップグレード**に関しては、オンライン ブックの以下のトピックが参考になると思います。

SQL Server フェールオーバー クラスターのアップグレード
http://msdn.microsoft.com/ja-jp/library/ms191009.aspx


SQL Server フェールオーバー クラスター インスタンスのアップグレード (セットアップ)
http://msdn.microsoft.com/ja-jp/library/ms191295.aspx

## 3.23 Integration Services のアップグレード

**Integration Services** を利用している場合は、SQL Server 2016 へのアップグレード インストールによって、Integration Services もアップグレードされます。Integration Services の場合は、アップグレード後に、古いバージョンの Integration Services が残り、サイド バイ サイドで実行されます。これは、**構成マネージャー**で次のように確認することができます。

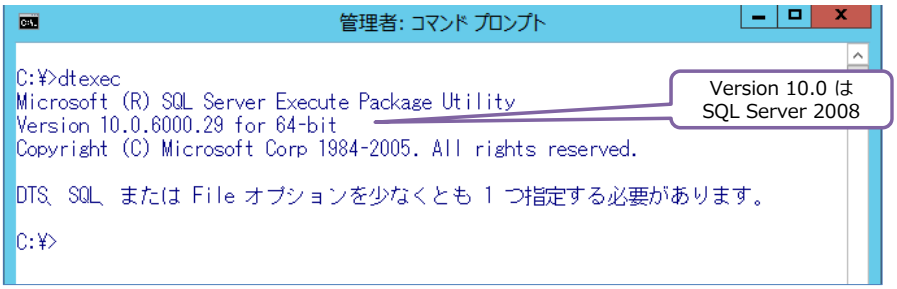



過去の Integration Services のバージョンは、それぞれ次のとおりです。

- SQL Server 2005 は **Integration Services**（バージョン番号なし）
- SQL Server 2008／2008 R2 は **Integration Services 10.0**
- SQL Server 2012 は **Integration Services 11.0**
- SQL Server 2014 は **Integration Services 12.0**
- SQL Server 2016 は **Integration Services 13.0**

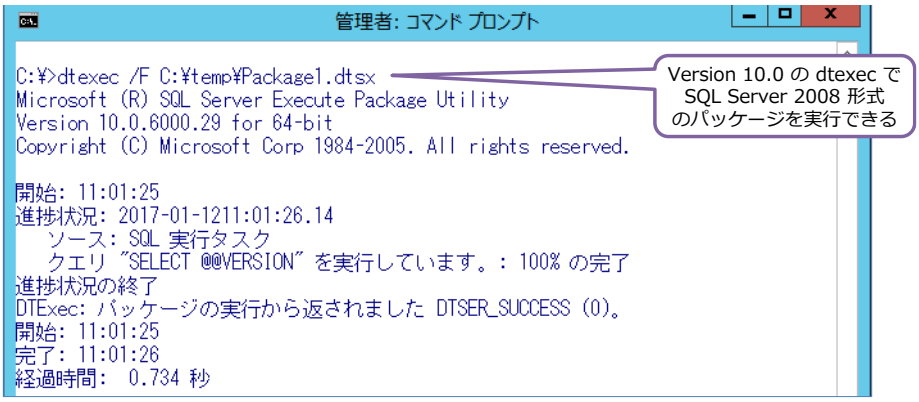
### dtexec を利用している場合

**dtexec** を利用して、SSIS パッケージ（Integration Services のパッケージ）を実行している場合は、**dtexec** の**既定のパス**（パスを指定しないで実行した場合）が以前のバージョンになることに注意する必要があります。これは次のような状況です。




これは、SQL Server 2008 から SQL Server 2016 へアップグレードを行った後に、**dtexec** を実行したときのものですが、**Version 10.0** と表示されて、**SQL Server 2008 版**の **dtexec** が実行されていることを確認できます。後述しますが、**SSIS パッケージ**（**.dtsx** ファイル）は、SQL Server 2016 へのアップグレードによってアップグレードされるわけではないので（SQL Server

2008 からのアップグレードの場合は、SQL Server 2008 形式の SSIS パッケージのままになるので)、それらを（以前のバージョン同士で）実行するための処置になっています。




**SQL Server 2016 版**の **dtexec** を利用するには、次のパスの exe ファイルを利用するようにします。

**C:¥Program Files¥Microsoft SQL Server¥130¥DTS¥Binn**


毎回のパス移動が面倒な場合は、前述の sqlcmd や bcp コマンドと同様、PATH 環境変数を変更することで、既定のパスを SQL Server 2016 に変更することができます。

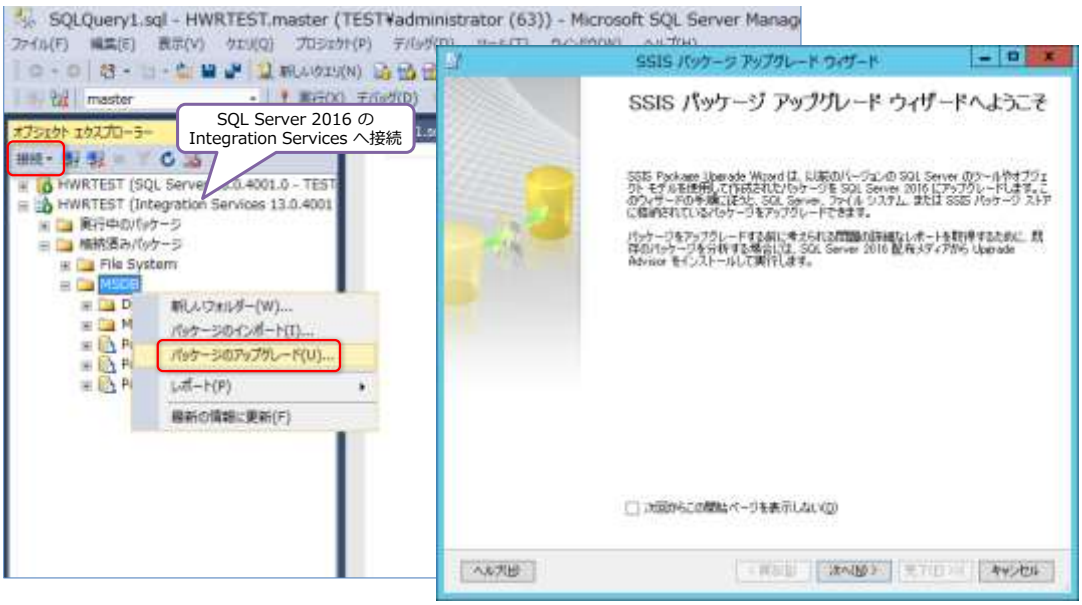
## SSIS パッケージのアップグレード ～パッケージのアップグレード ウィザード～

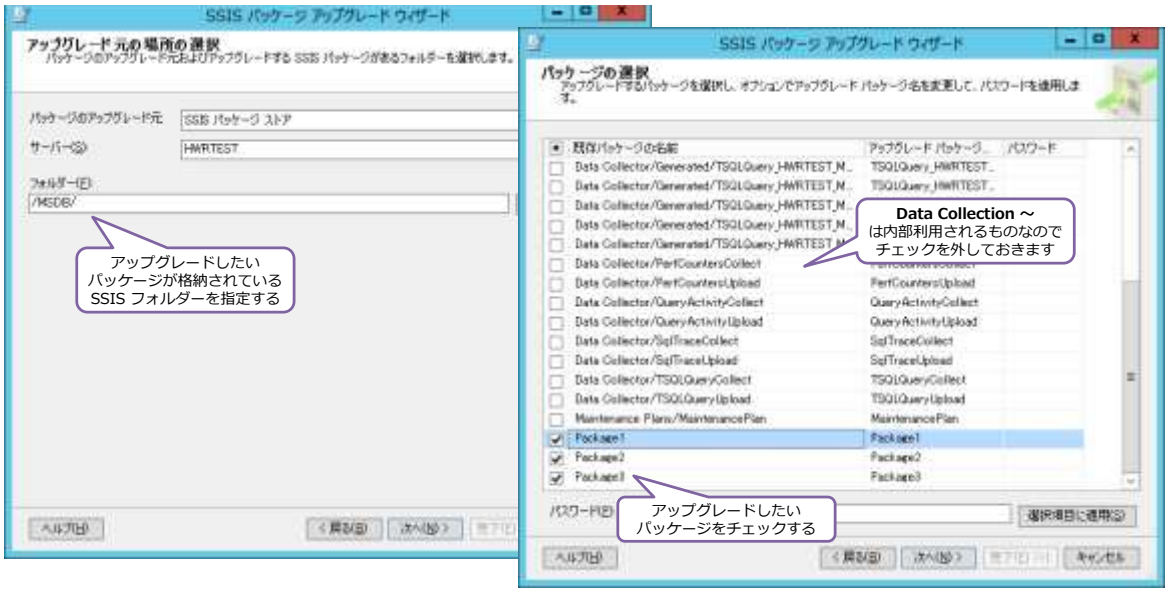
SQL Server 2016 へのアップグレードを行っても、**SSIS パッケージ**はアップグレードされません。これについては、オンライン ブックの次のトピックを一読することをお勧めします。

Integration Services パッケージのアップグレード
http://msdn.microsoft.com/ja-jp/library/cc280546.aspx


古い形式のまま実行できる SSIS パッケージの場合は、このトピックに記載されているように、パフォーマンスのオーバーヘッドがある（∵内部的にパッケージの変換を行った上で実行しているため）ことに注意する必要があります。また、古い形式のままでは実行できないもの（例えば、SQL Server 2012 以降に廃止された ”DTS 2000 パッケージ実行タスク” を含んでいるパッケージなど）がある場合は、それらを修正していく必要があります。

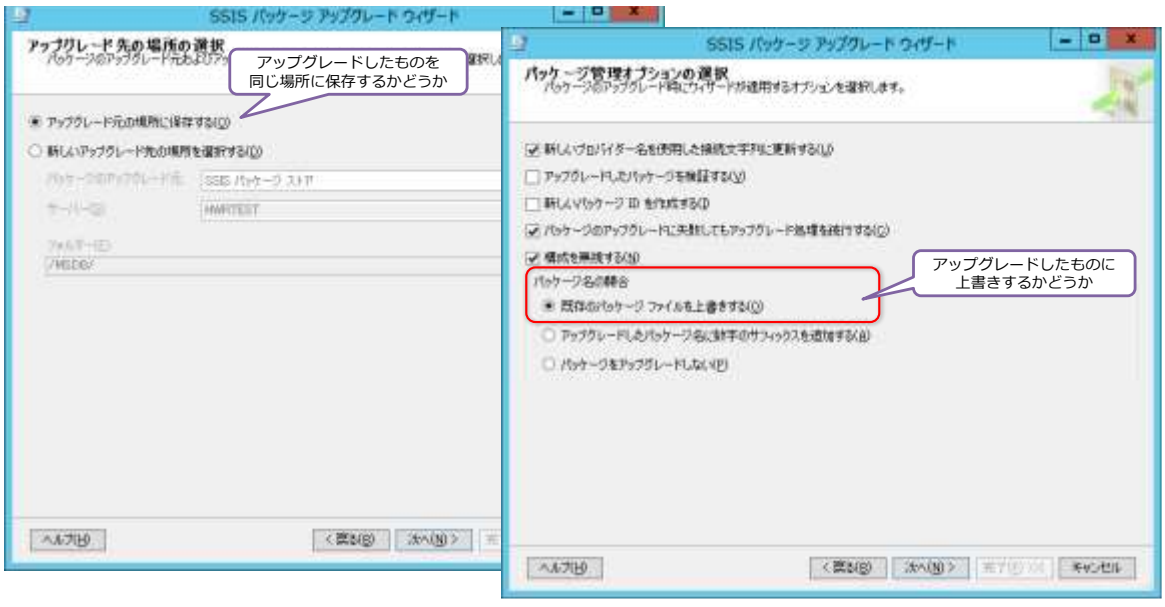
SSIS パッケージのアップグレードは、「**アップグレード ウィザード**」を利用することで簡単に SQL Server 2016 形式に変換することができます。アップグレード ウィザードは、SSDT や、Management Studio から起動することができますが、Management Studio を利用する場合は、次のようにオブジェクト エクスプローラーの [**接続**] メニューで Integration Services へ接続して、[**MSDB**] フォルダーを右クリックして、[**パッケージのアップグレード**] をクリックします（これで SQL Server 内に格納されたすべての Integration Services をまとめてアップグレードすることができます）。





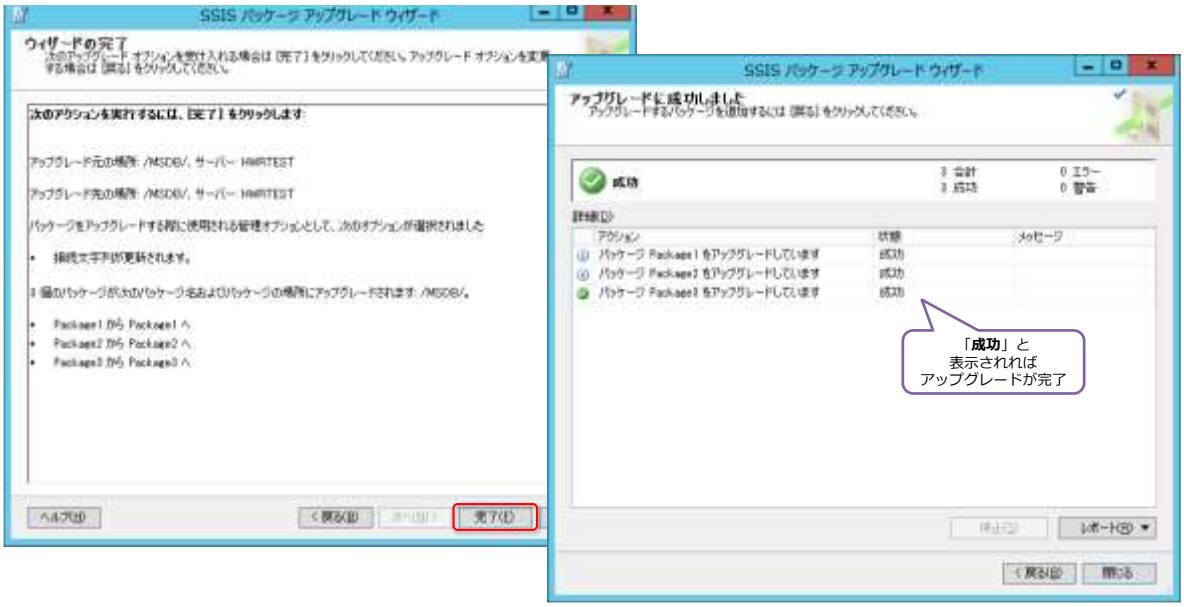

［**パッケージの選択**］ページでは、すべての SSIS パッケージが表示されますが、**Data Collection ～** と名前の付くパッケージは後述のパフォーマンス データ コレクションで内部利用されるものなので、これらのチェックは外しておきます。ここでは、アップグレードしたいパッケージのみをチェックするようにします。

次の［**アップグレード先の場所の選択**］ページでは、アップグレードしたものを同じ場所に保存するかどうかを指定します。




［**パッケージ管理オプションの選択**］ページでは、アップグレードしたパッケージを上書きするかどうか（以前のバージョンを残すかどうか）を指定します。既定では「**アップグレードしたパッケージ名に数字のサフィックスを追加する**」が選択されていて、既存のパッケージが「**A**」なら、アップグレード後のパッケージが「**A(1)**」という名前になり、既存のパッケージを上書きせずに残すことができます。

次の［**ウィザードの完了**］ページでは、［**完了**］ボタンをクリックすることで、アップグレードが開始されます。




## その他の Integration Services のアップグレードに関する情報

その他の Integration Services のアップグレードに関する情報は、オンライン ブックの以下のトピックが参考になると思います。

Integration Services のアップグレード
http://msdn.microsoft.com/ja-jp/library/cc879336.aspx


**Note： DTS 2000 パッケージ実行タスクを利用している場合**

SQL Server 2012 からは、SQL Server 2000 のときの DTS や DTS 2000 パッケージ実行タスクが廃止されているので、これを利用している場合には、SSIS パッケージを修正する必要があります。これについては、SQL Server 2012 自習書シリーズの「**SQL Server 2000 ユーザーのための Integration Services (SSIS) 入門**」で詳しく説明しているので、こちらもぜひご覧いただければと思います。
http://www.microsoft.com/ja-jp/sqlserver/2012/technology/self-learning.aspx

## 3.24 SQL Server 2008 からの新機能のアップグレード

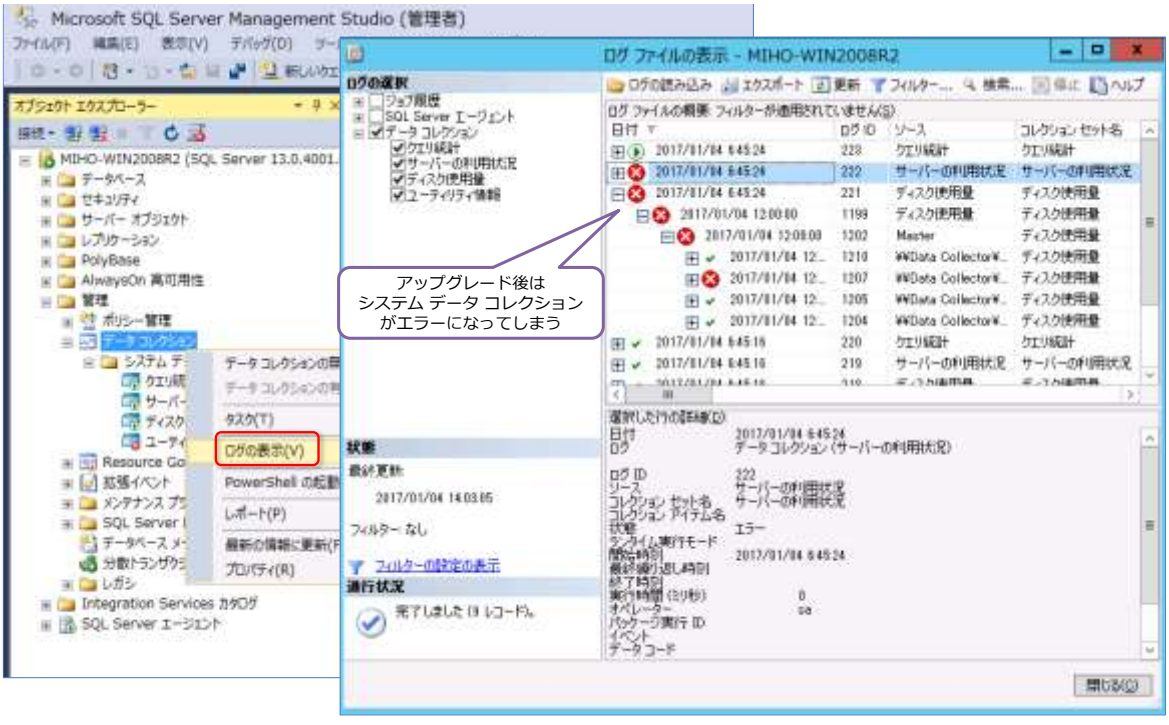
SQL Server 2008 から提供された主な新機能には、次のものがあります。

- **SQL Server Audit（SQL Server 監査）**
- **透過的なデータ暗号化（TDE : Transparent Data Encryption）**
- **ポリシー ベースの管理**
- **CDC（変更データ キャプチャ）**
- **リソース ガバナー**
- **パフォーマンス データ コレクション**

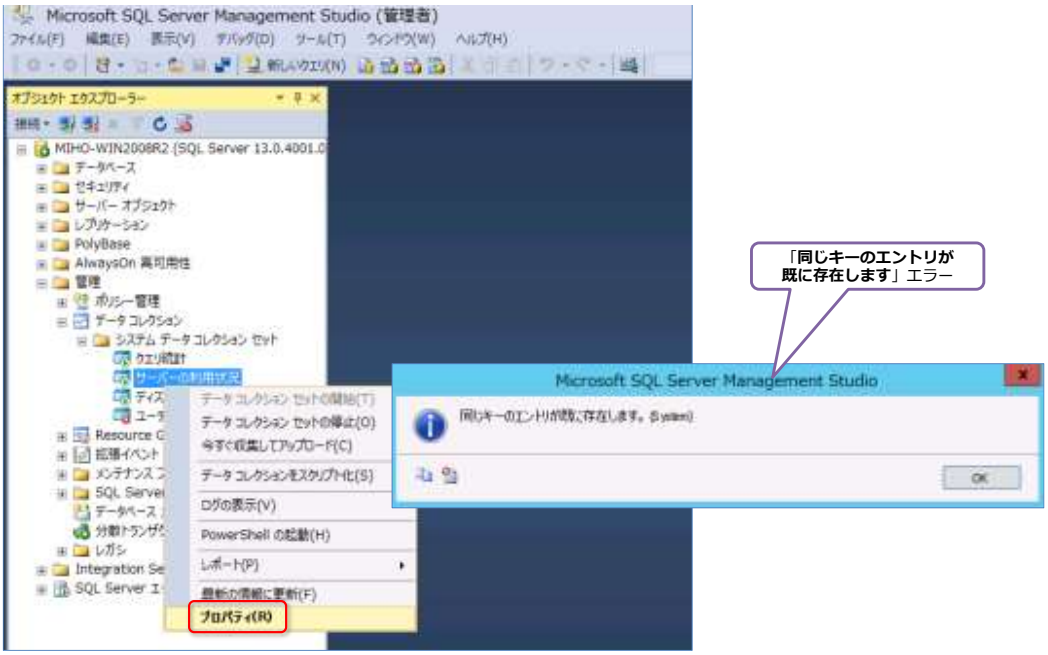
これらは、パフォーマンス データ コレクションを除いて、SQL Server 2016 へアップグレード インストールを行っても、そのまま動作させることができます。SQL Server Audit の監査設定も、透過的なデータ暗号化も、ポリシー ベースの管理で設定したポリシーも、CDC も、リソース ガバナーで作成したリソース プールも、そのまま動作させることができます。

### パフォーマンス データ コレクションの修正

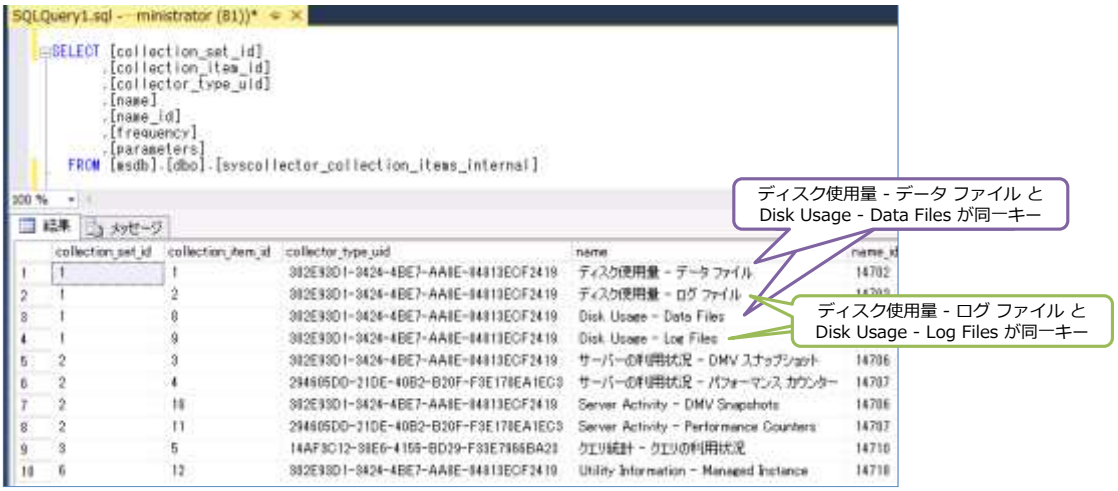
SQL Server 2008 および 2008 R2 で**パフォーマンス データ コレクション**を設定済みで、SQL Server 2016 へアップグレードした場合には、パフォーマンス データ コレクション（サーバーの利用状況やクエリ統計、ディスク使用量などのシステム データ コレクション）でエラーとなってしまいます。これは次のような状況です（SQL Server 2012／2014 の場合は発生しません）。



このエラーの原因は、次のようにプロパティを開こうとすることで確認できます。




「**同じキーのエントリが既に存在します**」というエラーが表示されて、これはデータ コレクションが格納されている **msdb** データベース内の「**syscollector_collection_items_internal**」テーブルに重複値があるというエラーになります。




このように、同じキー（**collector_type_uid**）の日本語のデータ コレクションが重複して格納されてしまっていることが原因なので、次のように **DELETE** ステートメントを実行して、重複しているデータを削除します。

```
USE msdb
DELETE FROM [syscollector_collection_items_internal]
 WHERE name IN('ディスク使用量 - データ ファイル'
              ,'ディスク使用量 - ログ ファイル'
              ,'サーバーの利用状況 - DMV スナップショット'
              ,'サーバーの利用状況 - パフォーマンス カウンター' )
```

```
USE msdb
DELETE FROM [syscollector_collection_items_internal]
 WHERE name IN('ディスク使用量 - データ ファイル'
              ,'ディスク使用量 - ログ ファイル'
              ,'サーバーの利用状況 - DMV スナップショット'
              ,'サーバーの利用状況 - パフォーマンス カウンター' )
```

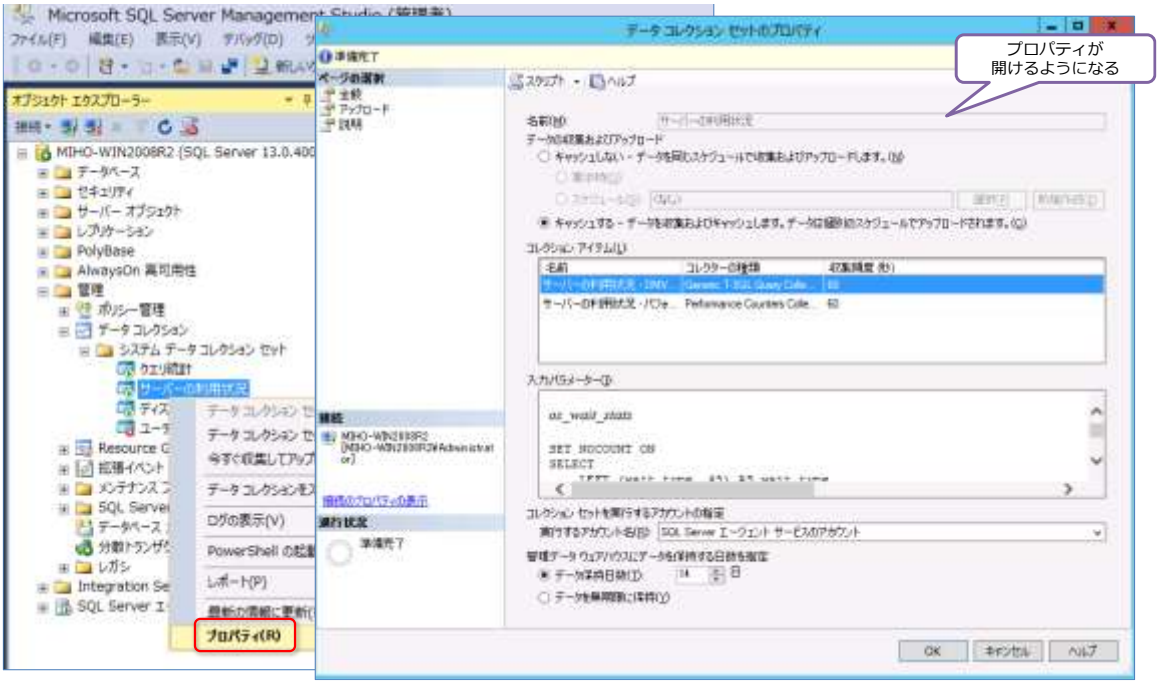
重複しているものを
すべて削除する

100 %

メッセージ

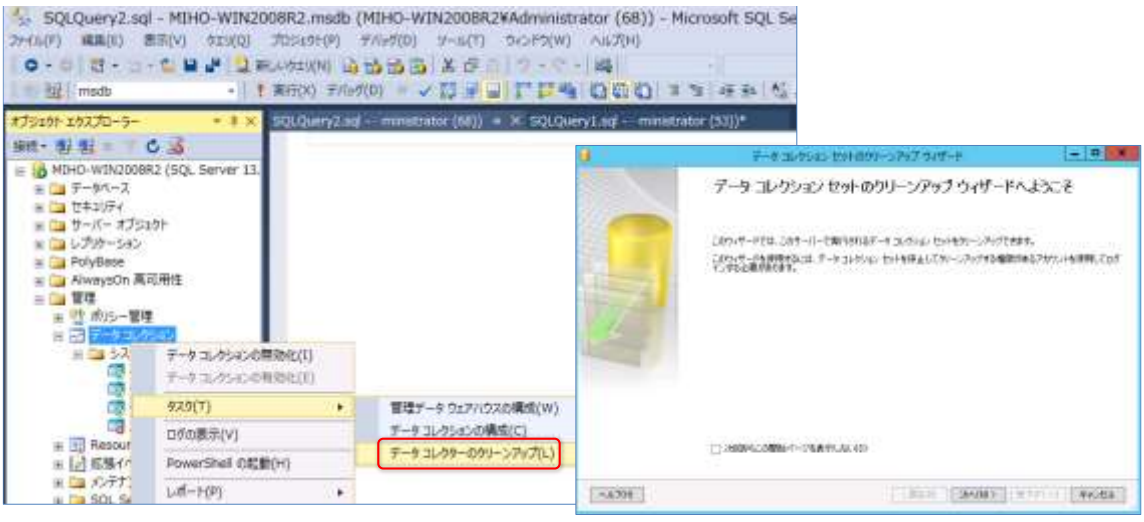
(4 行処理されました)

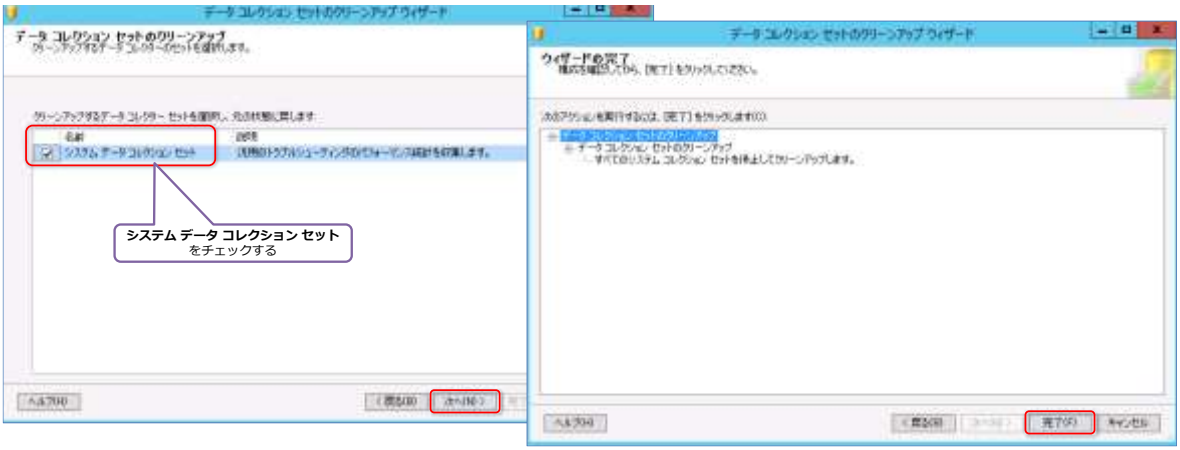
このように、重複データを削除することで、プロパティが開けるようになります。




削除後も、まだエラーは出るので、次に、パフォーマンス データ コレクションの**クリーンアップ**を行って、さらに**再構成**をします。

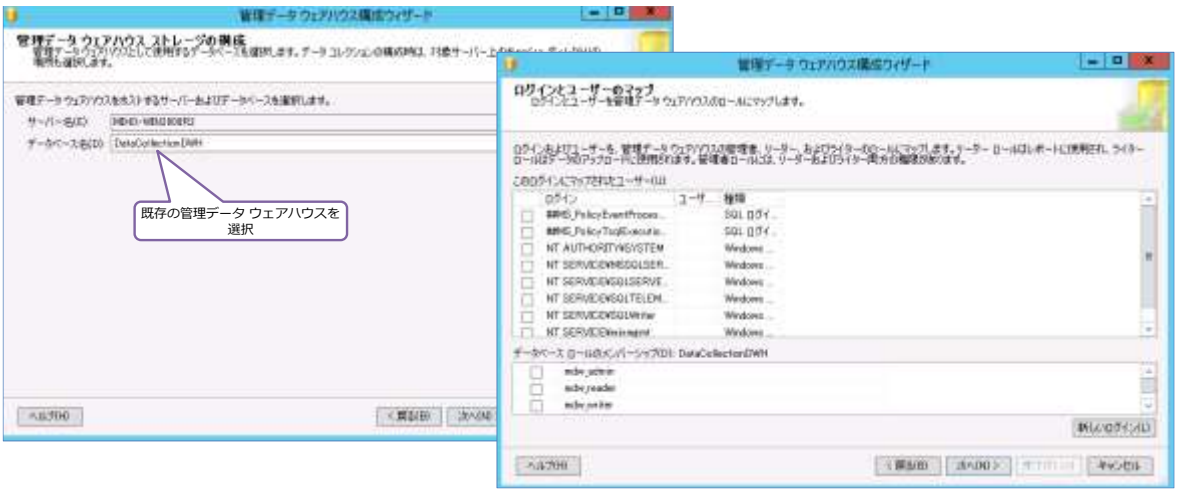
クリーンアップを行うには、次のように［**データ コレクション**］を右クリックして、［**タスク**］メニューの［**データ コレクターのクリーンアップ**］をクリックします。





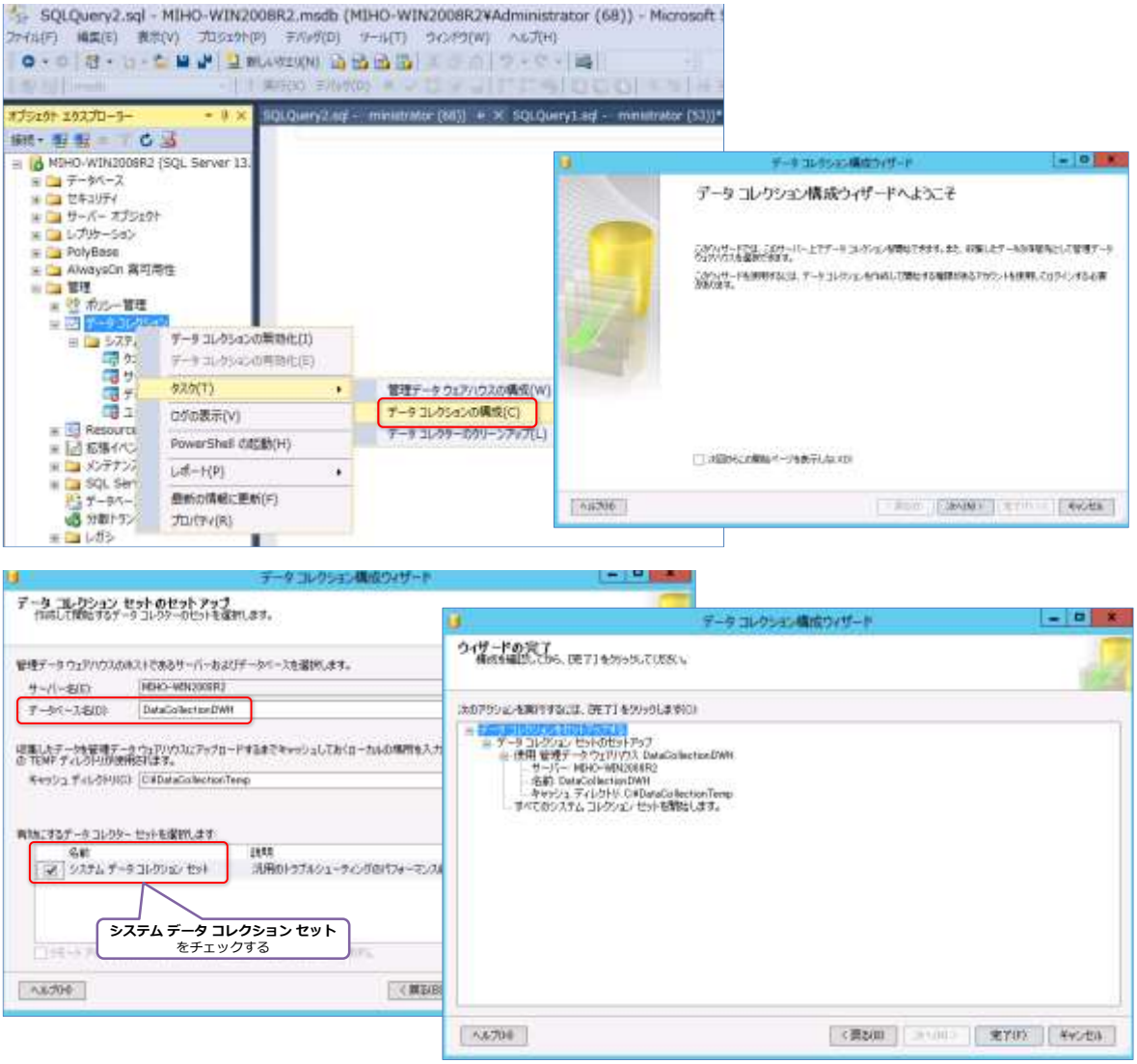

これで、クリーンアップが完了です。

クリーンアップが完了したら、次は、パフォーマンス データ コレクションを再構成します。再構成は、新規作成時と同様、[**データ コレクション**] を右クリックして、[**タスク**] メニューの [**管理データ ウェアハウスの構成**] をクリックします。

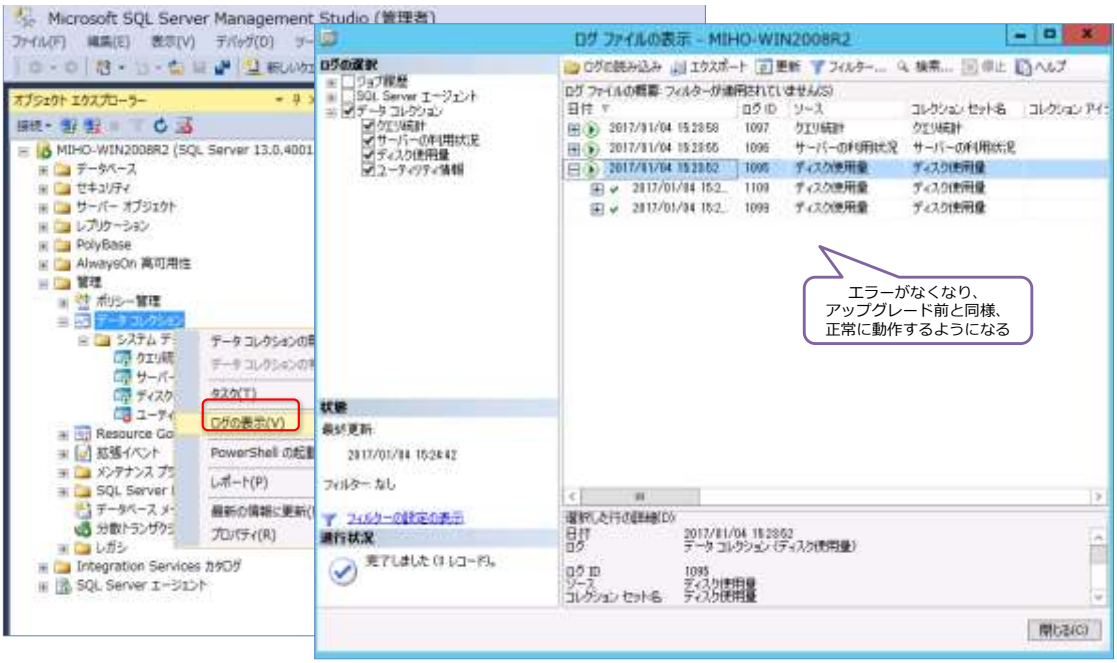

[**データベース**] では、アップグレード前に利用していた既存の管理データ ウェアハウスを選択する以外は、新規作成時と同様です。

管理データ ウェアハウスの構成が完了したら、次は［**データ コレクション**］を右クリックして、［**タスク**］メニューの［**データ コレクションの構成**］をクリックして、システム データ コレクションを構成します。



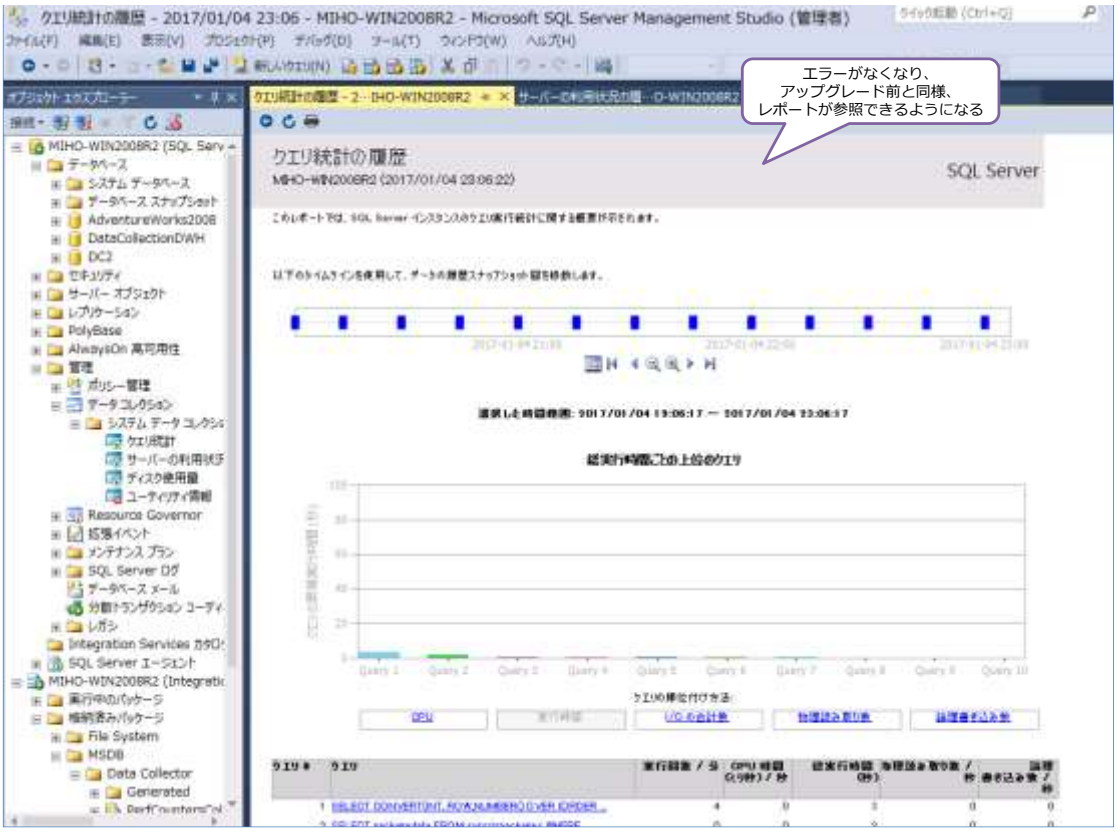

［**キャッシュ ディレクトリ**］でキャッシュ フォルダーを指定して（フォルダーがない場合は新規作成します）、［**名前**］で「**システム データ コレクション セット**」をチェックします。

以上で、パフォーマンス データ コレクションの再構成が完了です。これでエラーが消えて、アップグレード前と同じように、パフォーマンス データ コレクションを動作させることができます。

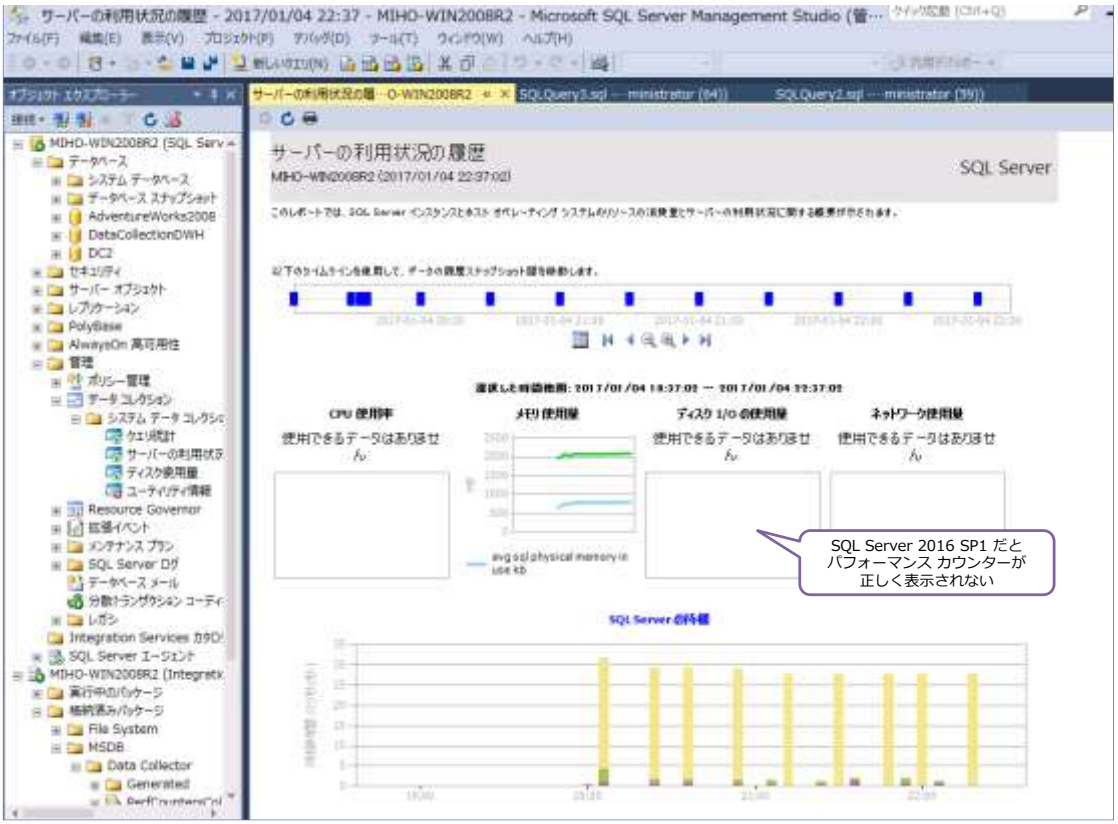



もし、これでもエラーが消えない場合は、**Integration Services** がインストールされているかどうかを確認してみてください（SQL Server 2008 では、パフォーマンス データ コレクションを動作させるために **Integration Services** は必須ではなかったのですが、SQL Server 2016 では必須になっています）。

エラーが消えた後は、レポートに関しても、以前と同様に参照できるようになります。



なお、SQL Server 2016 の **CU4** 以前のバージョンでは、データ コレクションのパフォーマンス カウンターが正しく表示されないという不具合があります（サーバーの利用状況の履歴レポートが以下のように表示されます）。



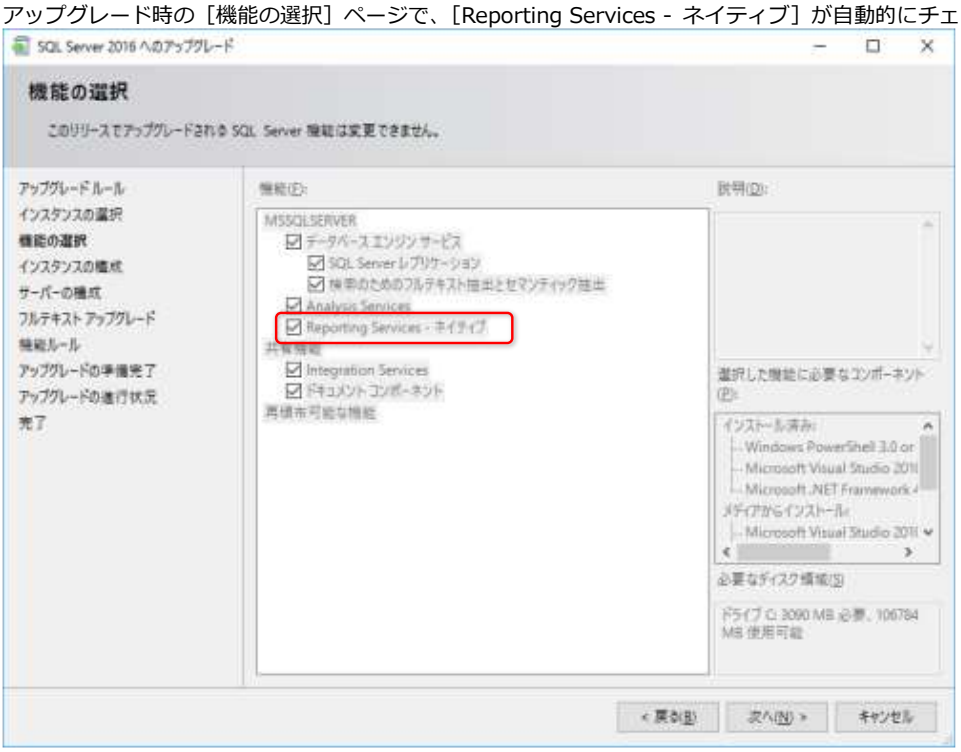

これについては、CU4 を適用することで修正されているので、パフォーマンス データ コレクションを利用する場合には、CU4 を適用することをお勧めします。これについては、以下の KB が参考になります。

FIX: "No Data Available" in the SQL Server Memory Usage page in the SQL Server 2014 or 2016 MDM report
https://support.microsoft.com/ja-jp/help/3209442/fix-no-data-available-in-the-sql-server-memory-usage-page-in-the-sql-s

## 3.25 Reporting Services のアップグレード

**Reporting Services** を利用している場合は、SQL Server 2016 へのアップグレード インストールによって、Reporting Services もアップグレードされます。

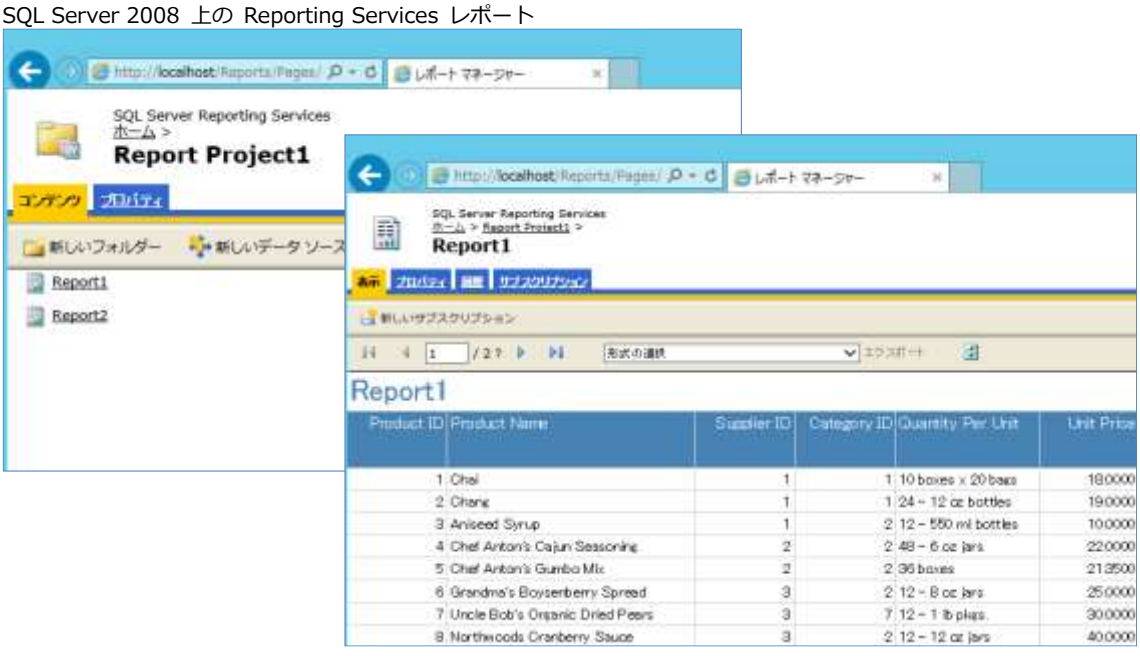
アップグレード時の［機能の選択］ページで、［Reporting Services - ネイティブ］が自動的にチェックされる



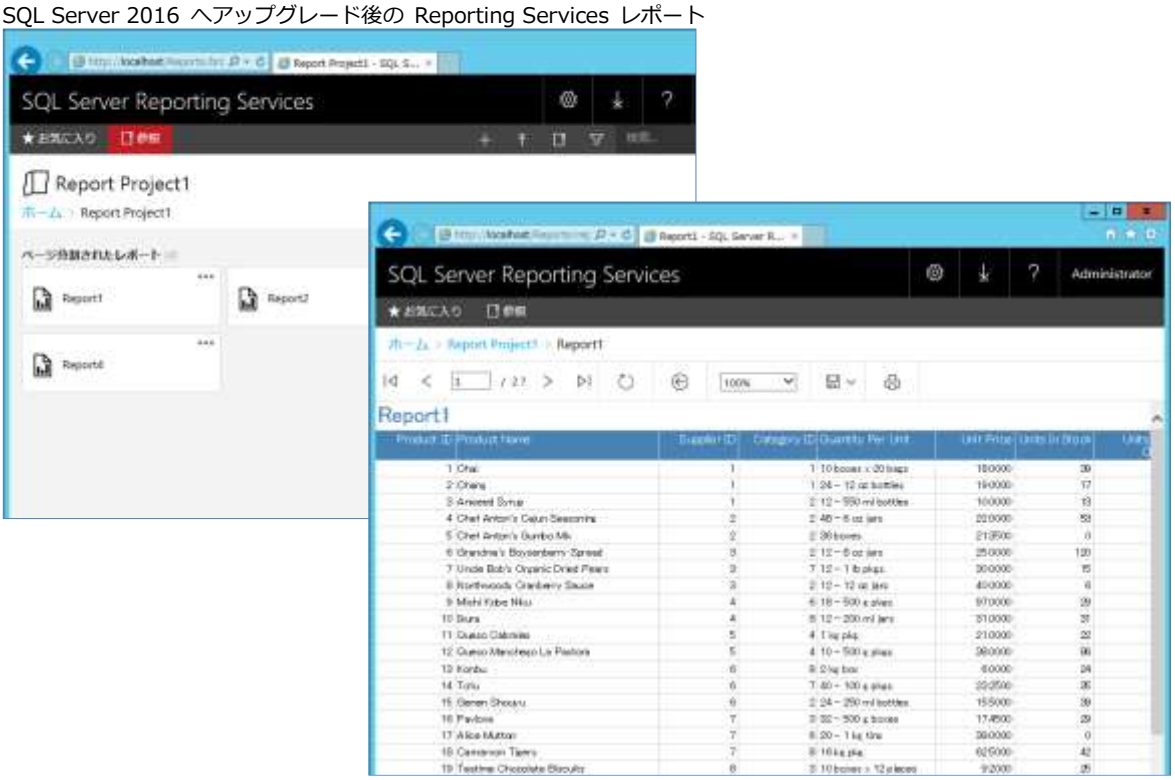

アップグレード時の［**機能の選択**］ページでは、以前のバージョンで利用していた機能が一覧されて、自動的にチェックが付き、アップグレード対象となります。

アップグレード完了後は、以前のバージョンで利用していたレポートは、SQL Server 2016 になっても、そのまま利用／参照することができます。

SQL Server 2008 上の Reporting Services レポート

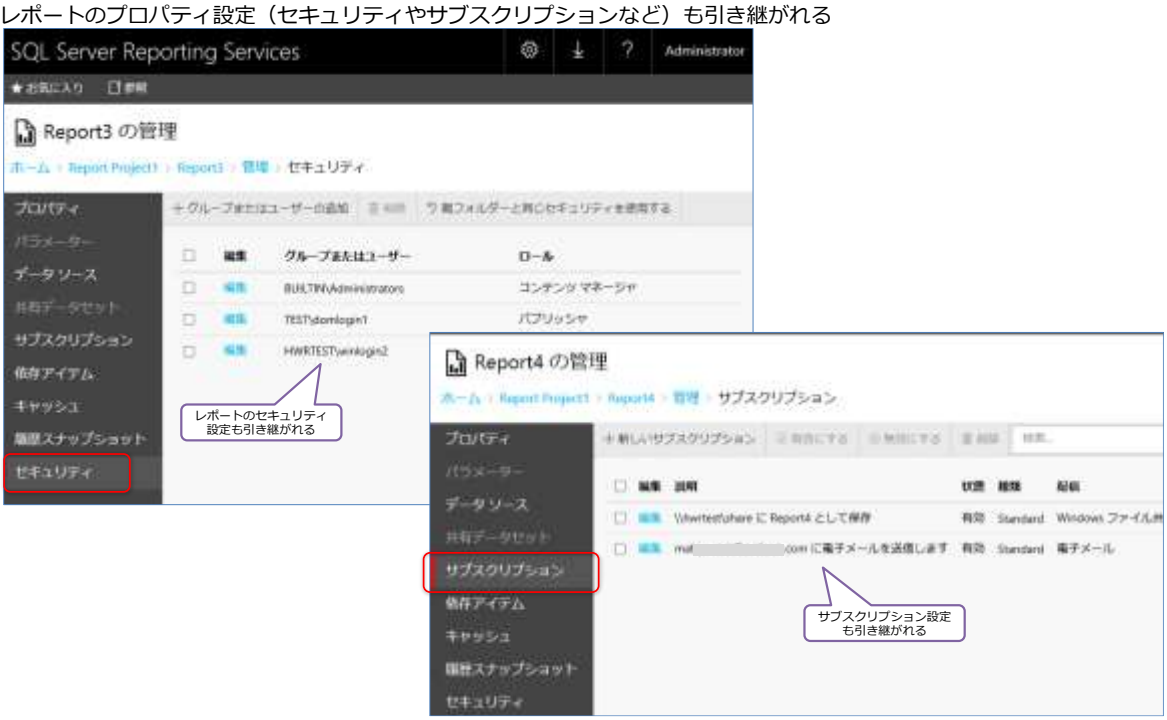


SQL Server 2016 へアップグレード後の Reporting Services レポート

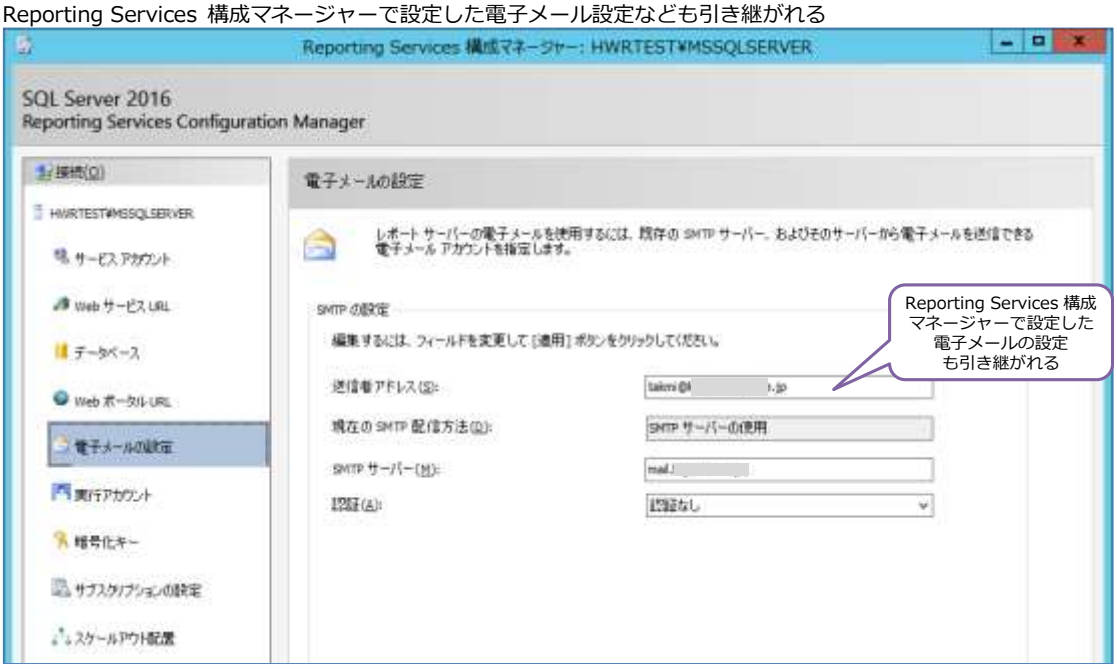



レポートのセキュリティ設定（ロール）や、サブスクリプション設定、キャッシュの設定、レポート サーバーの構成（電子メール設定）なども、アップグレードによって、すべて引き継がれるので、そのまま利用することができます。

レポートのプロパティ設定（セキュリティやサブスクリプションなど）も引き継がれる



Reporting Services 構成マネージャーで設定した電子メール設定なども引き継がれる




## Reporting Services のアップグレードに関するその他の情報

その他の Reporting Services のアップグレードに関する情報は、オンライン ブックの以下のトピックが参考になると思います。

Reporting Services のアップグレード
http://msdn.microsoft.com/ja-jp/library/ms143747.aspx


また、以下の互換性に関するトピックも参考になると思います。

Reporting Services の旧バージョンとの互換性
http://msdn.microsoft.com/ja-jp/library/ms143251.aspx


SQL Server 2016 で廃止された SQL Server Reporting Services の機能
http://msdn.microsoft.com/ja-jp/library/ms144231.aspx

SQL Server 2016 における SQL Server Reporting Services の非推奨機能
http://msdn.microsoft.com/ja-jp/library/ms143509.aspx

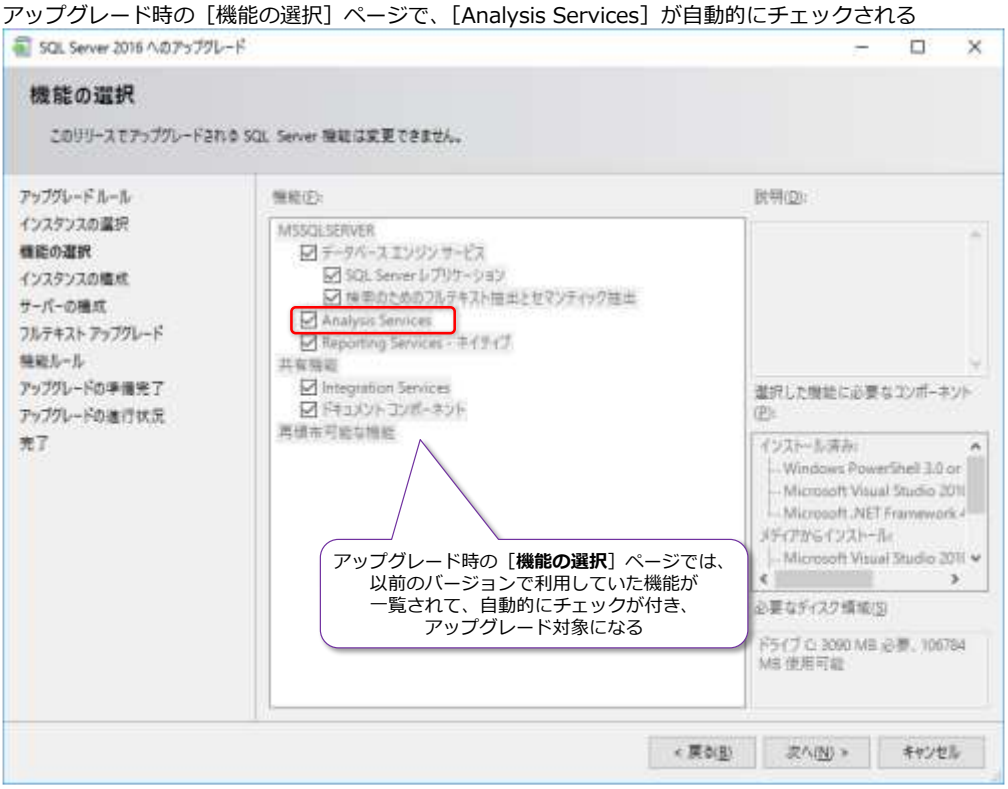

SQL Server 2016 における SQL Server Reporting Services の重大な変更
http://msdn.microsoft.com/ja-jp/library/ms143380.aspx

SQL Server 2016 における SQL Server Reporting Services の動作変更
http://msdn.microsoft.com/ja-jp/library/ms143200.aspx

## 3.26 Analysis Services のアップグレード

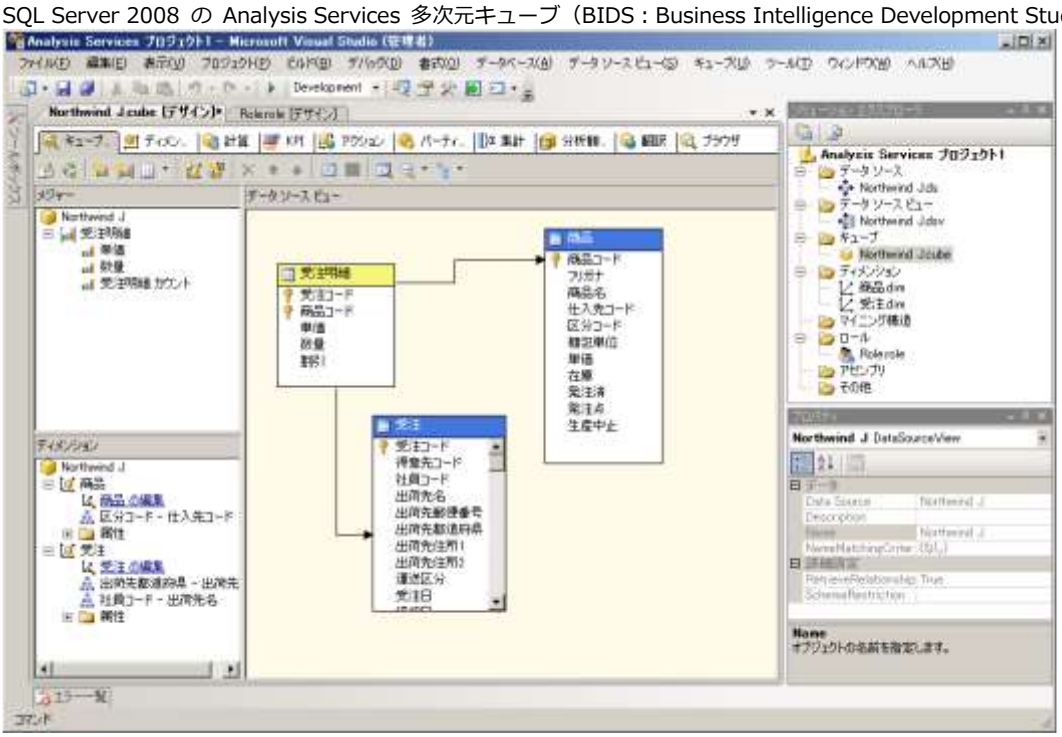
**Analysis Services** を利用している場合は、SQL Server 2016 へのアップグレード インストールによって、Analysis Services もアップグレードされます。

アップグレード時の［機能の選択］ページで、［Analysis Services］が自動的にチェックされる

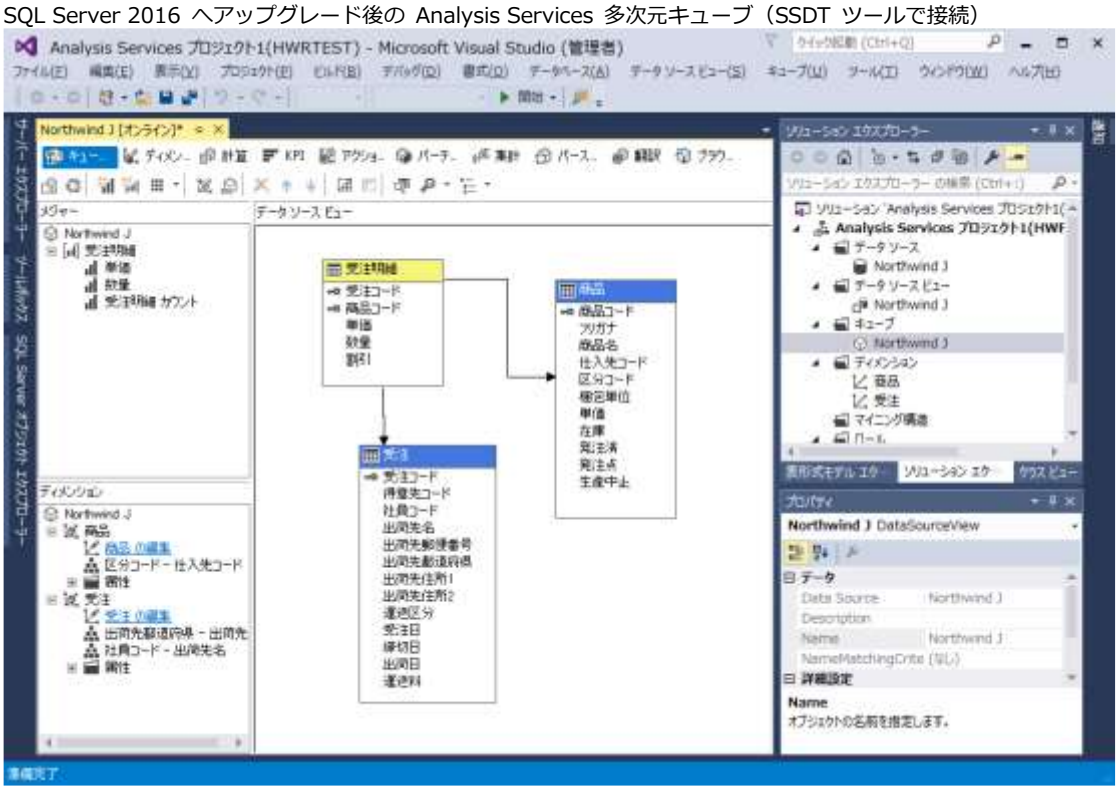



アップグレード完了後は、以前のバージョンで利用していた多次元キューブ（OLAP）は、SQL Server 2016 になっても、そのまま利用／参照することができます。

SQL Server 2008 の Analysis Services 多次元キューブ（BIDS : Business Intelligence Development Studio）

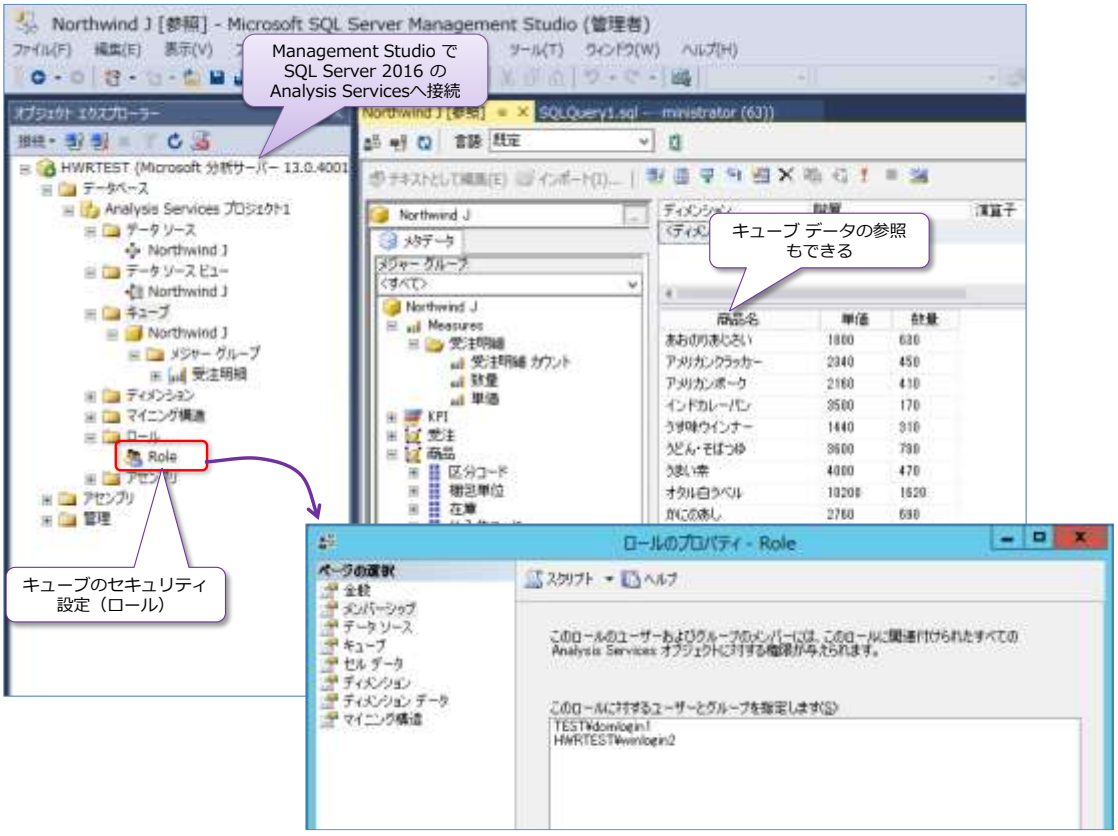


SQL Server 2016 へアップグレード後の Analysis Services 多次元キューブ（SSDT ツールで接続）

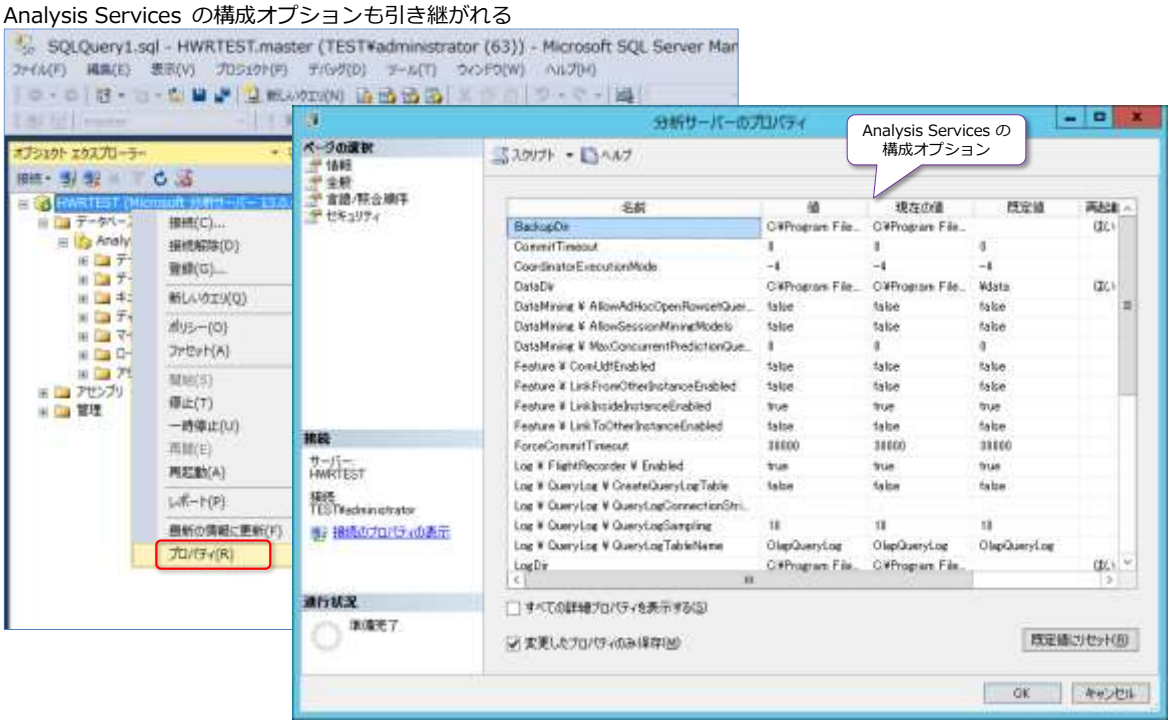



キューブのセキュリティ設定（ロール）や、Analysis Services の構成オプションなども、アップグレードによって、すべて引き継がれるので、そのまま利用することができます。

キューブのセキュリティ設定（ロールなど）も引き継がれる



Analysis Services の構成オプションも引き継がれる




## Analysis Services のアップグレードに関するその他の情報

その他の Analysis Services のアップグレードに関する情報は、オンライン ブックの以下のトピックが参考になると思います。

Analysis Services のアップグレード
http://msdn.microsoft.com/ja-jp/library/ms143686.aspx

また、以下の互換性に関するトピックも参考になると思います。

Analysis Services の旧バージョンとの互換性
http://msdn.microsoft.com/ja-jp/library/ms143479.aspx


SQL Server 2016 で提供が中止された Analysis Services の機能
http://msdn.microsoft.com/ja-jp/library/ms143229.aspx

SQL Server 2016 における Analysis Services 非推奨機能
http://msdn.microsoft.com/ja-jp/library/ms143346.aspx

SQL Server 2016 における Analysis Services 機能の動作の変更
http://msdn.microsoft.com/ja-jp/library/ms143682.aspx

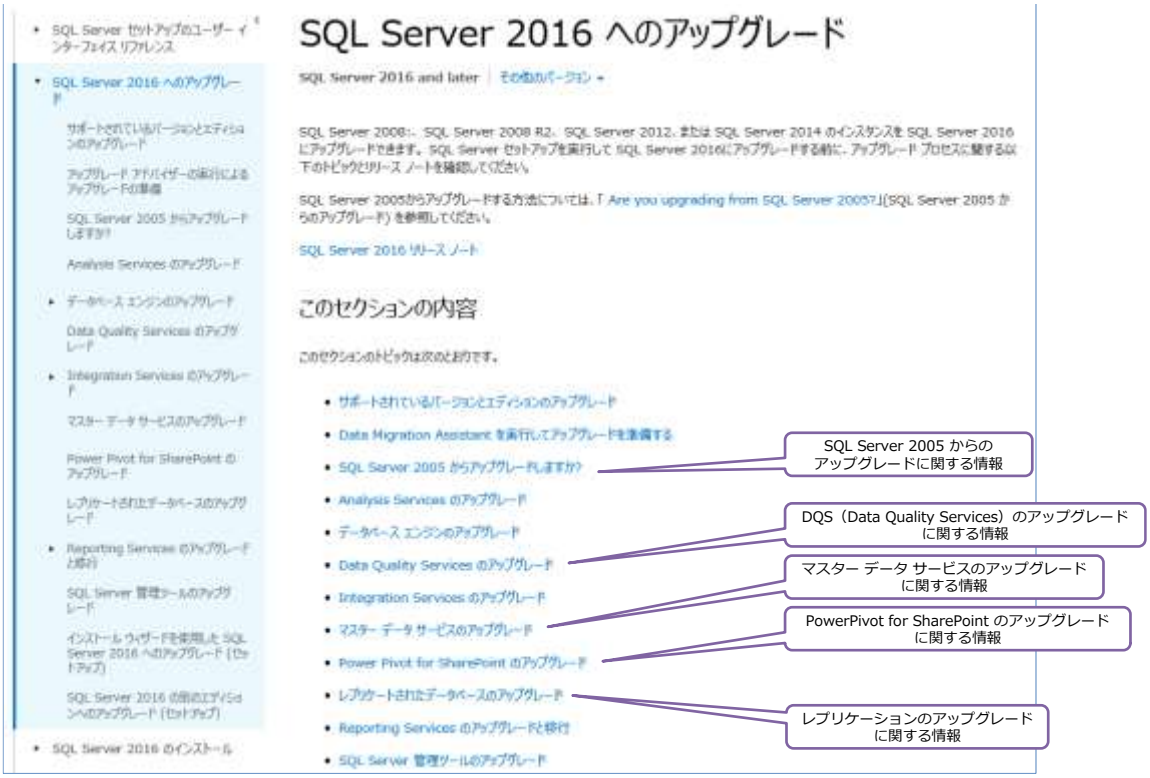
SQL Server 2016 の Analysis Services 機能における重大な変更
http://msdn.microsoft.com/ja-jp/library/ms143742.aspx

多次元データベースの互換性レベルの設定
http://msdn.microsoft.com/ja-jp/library/gg471593.aspx

## 3.27 その他の機能のアップグレード

**SQL Server 2008 R2** から提供された **PowerPivot for SharePoint** や、**SQL Server 2012** から提供された **Data Quality Services** や**マスター データ サービス**など、SQL Server のその他の機能のアップグレードに関しては、オンライン ブックの以下のトピックが参考になると思います。

SQL Server 2016 へのアップグレード（ポータル ページ）
http://msdn.microsoft.com/ja-jp/library/bb677622(v=sql.130).aspx




PowerPivot for SharePoint のアップグレード
http://msdn.microsoft.com/ja-jp/library/ee210646.aspx

Data Quality Services のアップグレード
http://msdn.microsoft.com/ja-jp/library/dn439769.aspx

マスター データ サービスのアップグレード
http://msdn.microsoft.com/ja-jp/library/gg488708.aspx

SQL Server 2005 からアップグレードしますか？
http://msdn.microsoft.com/ja-jp/library/mt168847.aspx

# STEP 4. ケース２ 新規サーバーへのアップグレード

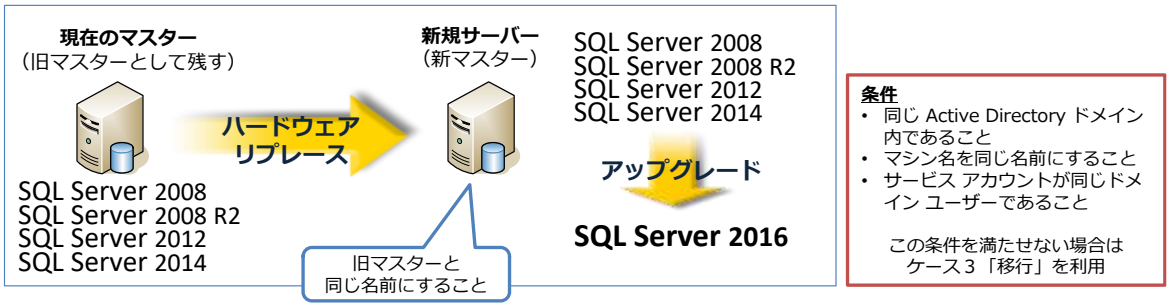
この章では、このドキュメントで扱う 3 つのケースのうちの「**ケース２　新規サーバー（別マシン）へのアップグレード**」について説明します。

このケースは、**同じ Active Directory ドメイン内**で、**同じ名前**の新規サーバーを構築する場合のアップグレード方法になるので、**異なる Active Directory ドメイン**や**ワークグループ環境**、**違う名前の新規サーバー**を構築する場合には、次の章で説明する「**ケース３ 別マシンへの移行**」（マイグレーション）を利用する必要があります。

- ✓ 新規サーバーへのアップグレード手順の概要
- ✓ 別マシンへの完全複製の手順（ハードウェア リプレース時の複製手順）
- ✓ Windows ローカル ユーザーを利用している場合の注意点
- ✓ アップグレード時間の見積もり

## 4.1 ケース２「新規サーバー（別マシン）へのアップグレード」

この章では、このドキュメントで扱う 3 つのケースのうちの「**ケース２ 新規サーバー（別マシン）へのアップグレード**」について説明します。




前の章のケース１は、同一マシンでの単純アップグレードでしたが、このケースは**別マシン（ハードウェア リプレース**による**新規サーバーの導入**）を利用したアップグレードです。ケース１は、単純なアップグレード（以前の SQL Server を上書き）なので、元の環境を残せないのがデメリットでしたが、このケースでは、旧システム環境を残すことができます（万が一のアップグレード失敗時に元の環境へ簡単に戻すことができます）。

「**現在のマスターと同じ名前**」の**新規サーバー**を、**同じドメイン内**に構築したい場合には、この方法が一番簡単で確実です。この方法は、あくまでも同じ名前の新規サーバーを同じドメイン内に作成できること、**サービス アカウント**が同じドメイン ユーザーであることが条件になるので、異なるドメインやワークグループ環境、違う名前の新規サーバーを構築する場合には、次の章で説明する「**ケース３ 別マシンへの移行**」（マイグレーション）を利用する必要があります。

また、**クロス プラットフォーム**（32 ビットから 64 ビットへの変更）にも対応していないので、この場合も「**ケース３ 別マシンへの移行**」を利用する必要があります。

### アップグレード手順の概要

このケースでのアップグレード手順の概要は、次のとおりです。

1. **現在のマスター環境**を**丸ごと新規サーバーへ複製**する

   - 現在のマスターで**オフライン バックアップ**（全データベース）を取得する
     （ユーザー データベースに関しては、オンライン バックアップでも可）
   - 現在のマスターを停止して、**ネットワークから切り離す**（旧マスターとなる）
   - **新規サーバー**に **OS** をインストールし、マシン名を**旧マスターと同じ名前**へ変更する
     （インストールする **OS** は、旧マスターと異なるものでも可。
     異なる OS を利用する場合は、SQL Server を動作させるための SP 要件も確認必須）
   - **新規サーバー**を **Active Directory ドメインへ参加**させる
   - 新規サーバーへ旧マスターと**同じバージョンの SQL Server** をインストールする
   - 旧マスターの SQL Server にインストール済みの**修正プログラム**を、新規サーバーにもインストールする

- 新規サーバーの **SQL Server を停止**する
- 旧マスターで取得した**オフライン バックアップを上書きコピー**する（復元する）
  ユーザー データベースをオンライン バックアップで取得している場合は、SQL Server の起動後に、該当データベースを復元する
- 新規サーバーの **SQL Server を起動**する
- **レジストリ**に格納されている情報を再設定する（TCP ポート番号や起動時パラメーターでのトレースフラグ設定などのうち、旧マスターで設定を変更しているものがある場合はそれらを再設定する）
- **OS の設定**で、旧マスターで変更しているものがある場合は、それらを再設定する
  （フォルダー構成や、NTFS アクセス許可、ユーザーの権利、共有フォルダーなど）

以上で旧マスターとまったく同じ環境を丸ごと新規サーバー上で動作させることができます。

2. **Data Migration Assistant** による事前チェックを行う
3. **SQL Server 2016** への**アップグレード要件**を確認する（アップグレード可能な Service Pack を確認／インストールする）

   SQL Server 2008 なら SP4、SQL Server 2008 R2 なら SP3、SQL Server 2012 なら SP2 が必要。OS に Windows Server 2003 や 2003 R2、2008、2008 R2 を利用している場合は、Windows Server 2012 以上にアップグレードする。
4. **新規サーバー**を **SQL Server 2016 へアップグレード**する
5. **SQL Server 2016** の**最新の修正プログラム**（CU や Service Pack など）をインストールする

   CU2 には、性能向上に関する修正が入っているので、できる限り最新の修正プログラムを適用しておくことをお勧めします。

   Reporting Services を利用している場合は、SP1 に対する重要な更新プログラムがあるので、CU4（SP1＋CU1）以上を適用しておくことをお勧めします。
6. **Management Studio**（管理ツール）の最新版をダウンロードして、インストールする（オプション）
7. **統計**（Statistics）を更新する
8. **データベースの互換性レベルを 130** へ上げる（オプション）

   互換性レベルの影響については**クエリ ストア**機能を利用することで簡単にチェック可能（実行プランの比較や、プラン強制もできる）。
9. **BIDS**（Business Intelligence Development Studio）や **SSDT-BI**（SQL Server Data Tools - Business Intelligence）を利用している場合は、**SSDT** の最新版をダウンロードして、インストールする（オプション）

このケースは、手順 1 の「**丸ごと複製**」がポイントになり、同じドメイン内で、同じ名前の新規サーバー、同じサービス アカウントを利用している場合であれば、この丸ごと複製を簡単に行うことができます（SQL Server のオフライン バックアップを復元するだけで複製できます）。

丸ごと複製が完了した後は、前の章と同様、SQL Server 2016 へのアップグレード（インプレース アップグレード）を行う手順になります（この手順は、前の章と全く同じです）。

以降では、丸ごと複製を行う際の、具体的な操作方法を説明します。丸ごと複製時のポイントとしては、**Active Directory ドメインのユーザー**ではなく、**Windows のローカル ユーザー**を利用している設定があるかどうかです。Windows のローカル ユーザーに関しては、マシンが別のものになると、たとえ同じ名前のマシンでも、同じユーザーを作成したとしても、**内部的な SID**（Security ID）が異なるものが割り当てられるので、設定が複製できないということが発生します（Active Directory ドメイン ユーザーであれば、この問題は発生しません）。どういった場面で Windows のローカル ユーザーが利用される可能性があるかについても、次項以降で説明します。

## 4.2 現在のマスター環境を新規サーバーへ複製（HW リプレース）

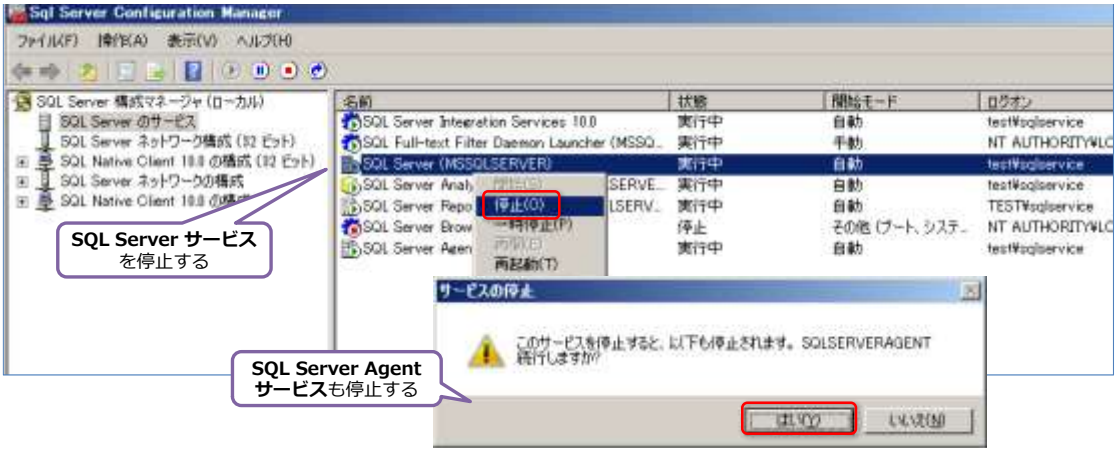
**現在のマスター環境を丸ごと新規サーバーへ複製**する（ハードウェア リプレースを行う）具体的な手順は、次のとおりです。

1. 現在のマスターの **SQL Server サービスを停止**する（∵オフライン バックアップのため）
2. 現在のマスターで**オフライン バックアップ**（全データベース）を取得する
   （ユーザー データベースに関しては、オンライン バックアップでも可）
3. 現在のマスターを停止して、**ネットワークから切り離す**（旧マスターとなる）
4. **新規サーバー**に **OS** をインストールし、マシン名を**旧マスターと同じ名前**へ変更する
   （インストールする **OS** は、旧マスターと異なるものでも可）
5. **新規サーバー**を **Active Directory ドメインへ参加**させる
6. 新規サーバーへ旧マスターと**同じバージョンの SQL Server** をインストールする
7. 旧マスターの SQL Server にインストール済みの**修正プログラム**を、新規サーバーにもインストールする
8. 新規サーバーの **SQL Server を停止**する
9. 旧マスターで取得した**オフライン バックアップを上書きコピー**する（復元する）
   ユーザー データベースをオンライン バックアップで取得している場合は、SQL Server の起動後に、該当データベースを復元する
10. 新規サーバーの **SQL Server を起動**する
11. **レジストリ**に格納されている情報を再設定する
12. **OS の設定**で、旧マスターで変更しているものがある場合は、それらを再設定する

以降では、これらの手順を詳しく説明します。

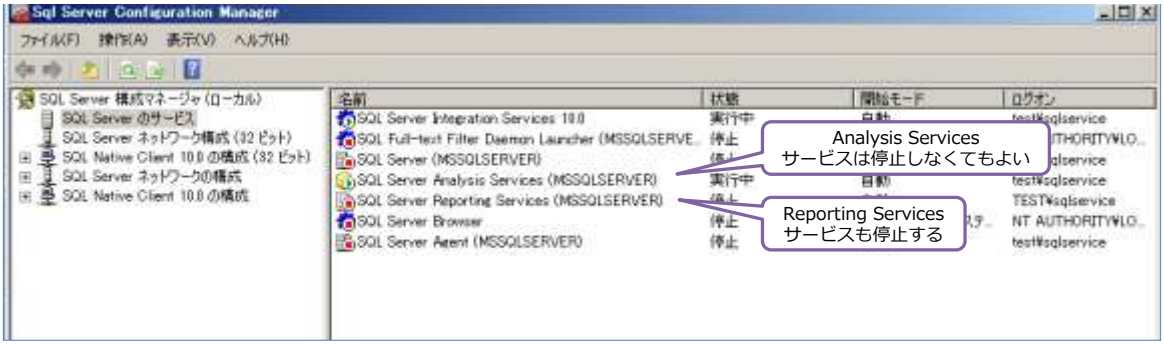
## 1. 現在のマスターの SQL Server サービスを停止

まずは、現在のマスターでオフライン バックアップを取得するために **SQL Server サービス**を停止します。SQL Server サービスは、**構成マネージャー** ツールを利用して、次のように停止します（**SQL Server Agent サービス**も停止します）。




**フルテキスト検索**機能を利用している場合は、**SQL Full-text Filter Daemon Launcher** サービスも停止します。

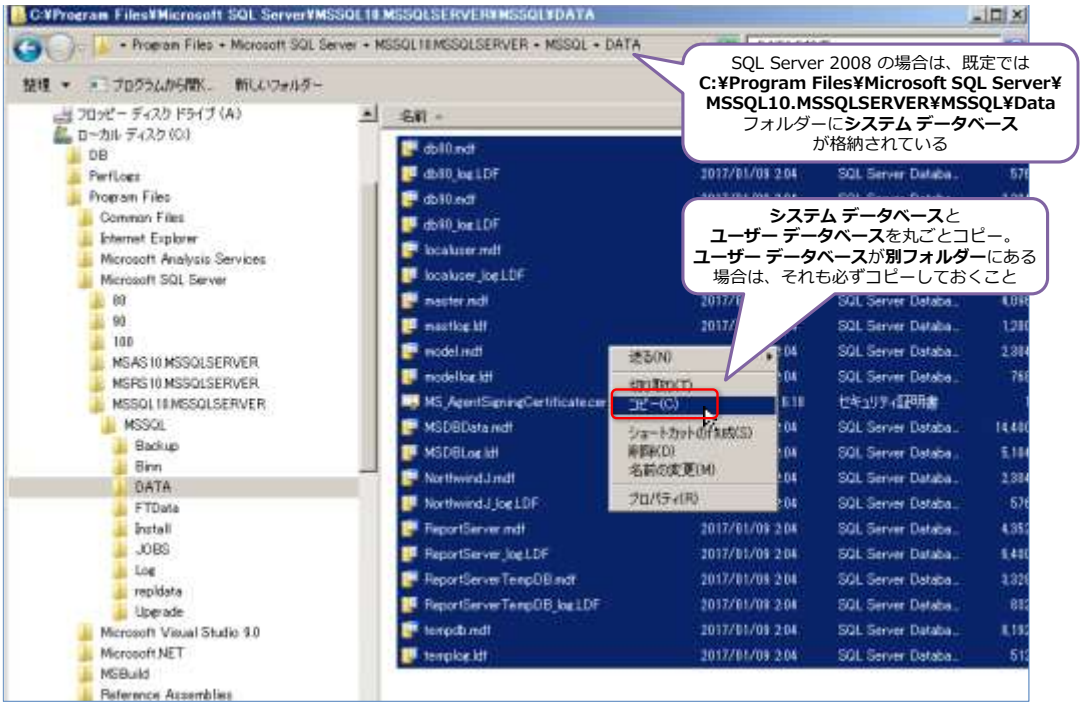
**Reporting Services** を利用している場合は、次のように Reporting Services サービスも停止します（**Analysis Services** に関しては停止しなくても大丈夫です）。




Reporting Services や Analysis Services の場合の丸ごと複製を行う手順については、この章の後半でまとめて説明します。ここでは、まず SQL Server のデータベース エンジンを丸ごと複製する方法について説明します。

## 2. 現在のマスターでオフライン バックアップを取得する

SQL Server サービスの停止が完了したら、**すべてのデータベース（システム データベースとユーザー データベースをすべて）**の**データ ファイル**（**.mdf**）と**トランザクション ログ ファイル**（**.ldf**）を、**Windows エクスプローラー**からファイル コピーで取得します。




**SQL Server 2008** の場合は、既定では**システム データベース**は、次のフォルダーに格納されています。

**C:¥Program Files¥Microsoft SQL Server¥MSSQL10.MSSQLSERVER¥MSSQL¥DATA**

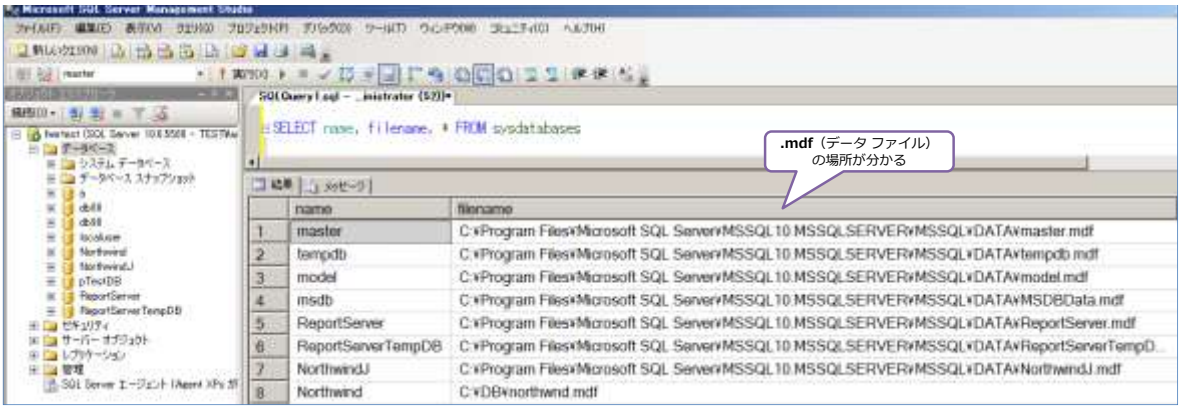
**SQL Server 2008 R2／2012／2014** の場合は、既定では次のフォルダーになります。

SQL Server 2008 R2 の場合
**C:¥Program Files¥Microsoft SQL Server¥MSSQL10_50.MSSQLSERVER¥MSSQL¥DATA**

SQL Server 2012 の場合
**C:¥Program Files¥Microsoft SQL Server¥MSSQL11.MSSQLSERVER¥MSSQL¥DATA**

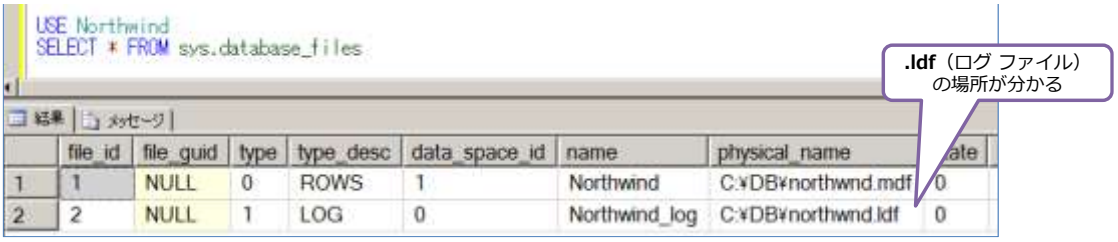
SQL Server 2014 の場合
**C:¥Program Files¥Microsoft SQL Server¥MSSQL12.MSSQLSERVER¥MSSQL¥DATA**

システム データベースを含めた、ユーザー データベースが格納されているフォルダーは、次のように **sysdatabases** 互換ビューの **filename** 列を参照することで確認することができます。




ここにリストされるデータベースのファイル（**.mdf**）とそれに対応したログ ファイル（**.ldf**）をす

べてコピーするようにします。このビューでは、**.ldf** ファイルの場所が分からないので、これを確認したい場合は、次のように **sys.database_files** システム ビューの **physical_name** 列を参照するようにします。




| | file_id | file_guid | type | type_desc | data_space_id | name | physical_name | ate |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | NULL | 0 | ROWS | 1 | Northwind | C:¥DB¥northwnd.mdf | 0 |
| 2 | 2 | NULL | 1 | LOG | 0 | Northwind_log | C:¥DB¥northwnd.ldf | 0 |

**ファイルのコピー先**には、**ファイル サーバー**（新規サーバーからもアクセス可能なマシン）や、**クラウド上のストレージ**（新規サーバーからもアクセス可能なストレージ）、**USB 接続の HDD** など持ち運びが可能なメディアを利用します（∵現在のマスターはこの後ネットワークから切り離すため）。ただし、USB 接続の場合は、データベース サイズが大きい場合には、転送スピードが遅いので（特に USB 2.0 の場合）、アップグレード時のダウンタイムの増加に繋がることになります（これについては非常に重要なので後述します）。

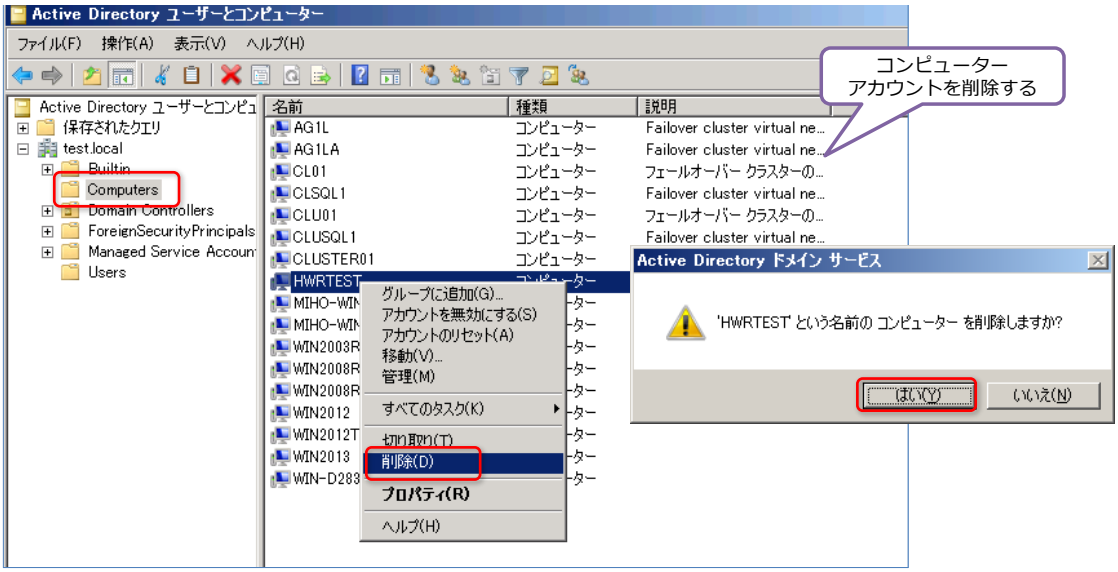
なお、ユーザー データベースに関しては、**オンライン バックアップ**（**BACKUP** ステートメント）で取得することも可能ですが、これもダウンタイムとの兼ね合いがあるので、詳しくは後述します。

## 3. 現在のマスターを停止して、ネットワークから切り離す

次に、現在のマスター（OS）をシャットダウンして、ネットワークから切り離します。これにより、現在のマスターを完全停止状態にし、万が一のアップグレード失敗時まで寝かせておきます（以降の手順では、現在のマスターを**旧マスター**と呼びます）。なお、アップグレードの成功後は、OS を再インストールすることで、新しいマシン（完全な別マシン）として利用することもできます。

## 4. Active Directory ドメインから旧マスターを削除する

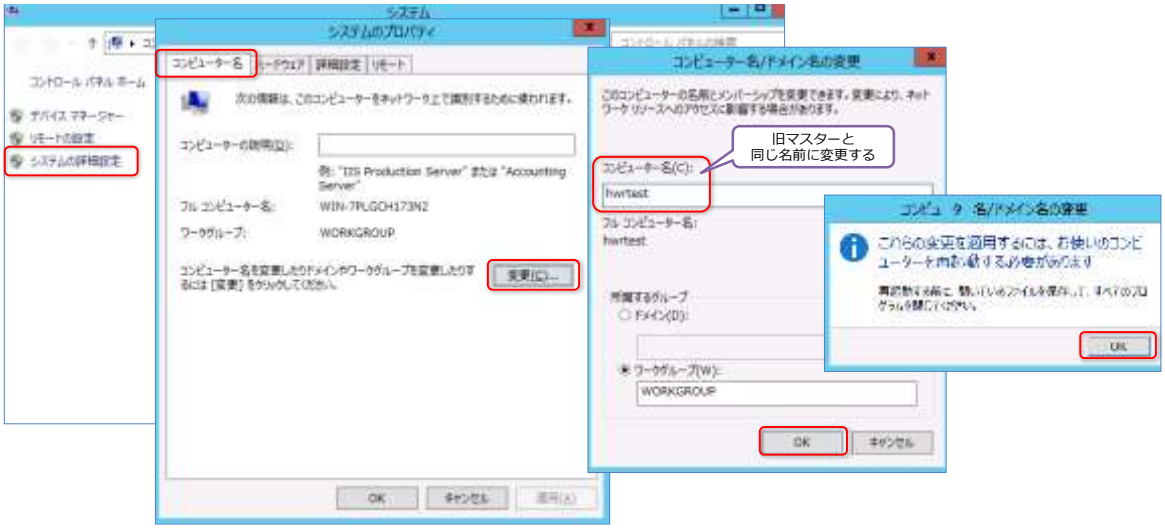
次に、旧マスターを **Active Directory ドメイン**の**コンピュータ アカウント**から削除します。これは、ドメイン コントローラーで「**Active Directory ユーザーとコンピューター**」ツールを利用して行います。




## 5. 新規サーバーへ OS をインストールして、マシン名を同じにする（最も重要）

次に、新規サーバーへ **OS** をインストールして、マシン名を旧マスターと**同じ名前**にします。OS をインストールするときは、**同じドライブ構成**（旧マスターが C: ドライブへインストールしているなら、新規サーバーも C: ドライブへインストール）になるようにします。

マシン名の変更は、コンピューター（PC）を右クリックして、[**プロパティ**] から [**システムの詳細設定**] をクリックし、[**システムのプロパティ**] ダイアログから、次のように行います。

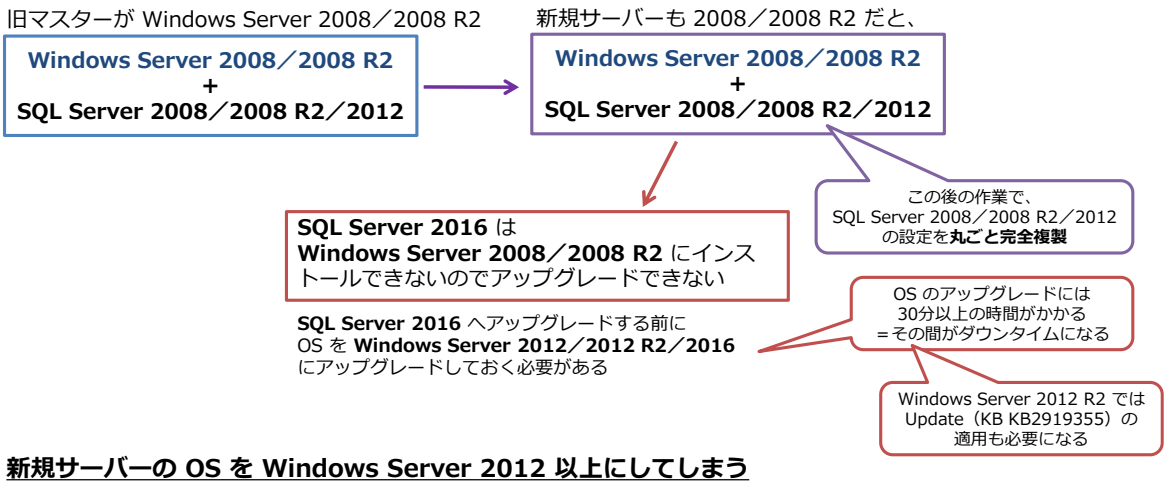



旧マスターと同じ名前に変更することは、一連の作業（丸ごと複製作業）の中で最も重要になります。また、名前の変更は、SQL Server をインストールする前に行っておくことも非常に重要になります。もし、SQL Server をインストールした後に、名前を変更した場合には、いくつかの機能で正しく動作しない場合があるので、必ず SQL Server をインストールをする前にマシン名を同じにしておくようにします。

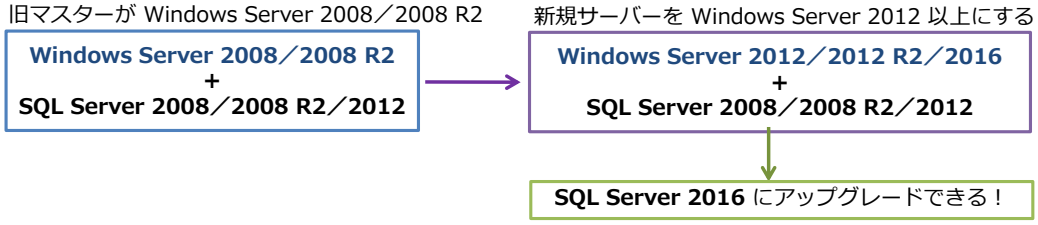
インストールする **OS** は、旧マスターと同じ OS でも、異なる OS でもかまいません。**SQL Server 2016** は **Windows Server 2012**（X64）以降の OS でのみサポートされるので、この

タイミングで、OS を Windows Server 2012 以上に入れ替えてしまうというのも 1 つの方法です（∵OS のアップグレードには **30 分以上**の時間がかかるため）。

**新規サーバーの OS が Windows Server 2008／2008 R2 の場合**




**新規サーバーの OS を Windows Server 2012 以上にしてしまう**




**SQL Server 2008** や **2008 R2／2012／2014** を **Windows Server 2008／2008 R2** 上で動作させている場合は、新規サーバーの OS を旧マスターと同じにすると、そのままでは **SQL Server 2016** にアップグレードすることができません。SQL Server 2016 へアップグレードする前に、**OS のアップグレード**（**Windows Server 2012** 以上へのアップグレード）をしなければなりません。OS のアップグレードには **30 分**以上かかってしまうので（Windows Server 2012 R2 の場合は、Update パッケージも適用する必要があります）、最初から、新規サーバーに **Windows Server 2012** 以上をインストールしてしまうも 1 つの方法になります。

OS が変わることに不安を持つ方もいらっしゃると思いますが、SQL Server は、「**SQL Server だけで完結した製品**」なので、OS の影響をほとんど受けません（OS が変わったとしても、SQL Server の動作にはほとんど影響ありません）。

ただし、OS を変更する場合は、SQL Server を動作させるための Service Pack 要件に注意する必要があります（以下）。

**SQL Server を動作させるために必要となる Service Pack**

| | Windows Server 2012 | Windows Server 2012 R2 | Windows Server 2016 |
|---|---|---|---|
| SQL Server 2008 | SP3 | SP3 | 未サポート |
| SQL Server 2008 R2 | SP1 | SP2 | 未サポート |
| SQL Server 2012 | RTM | SP1 | SP2 |
| SQL Server 2014 | RTM | RTM | SP1 |

**Windows Server 2012** を利用する場合には、**SQL Server 2008** なら **SP3** 以上（SQL Server

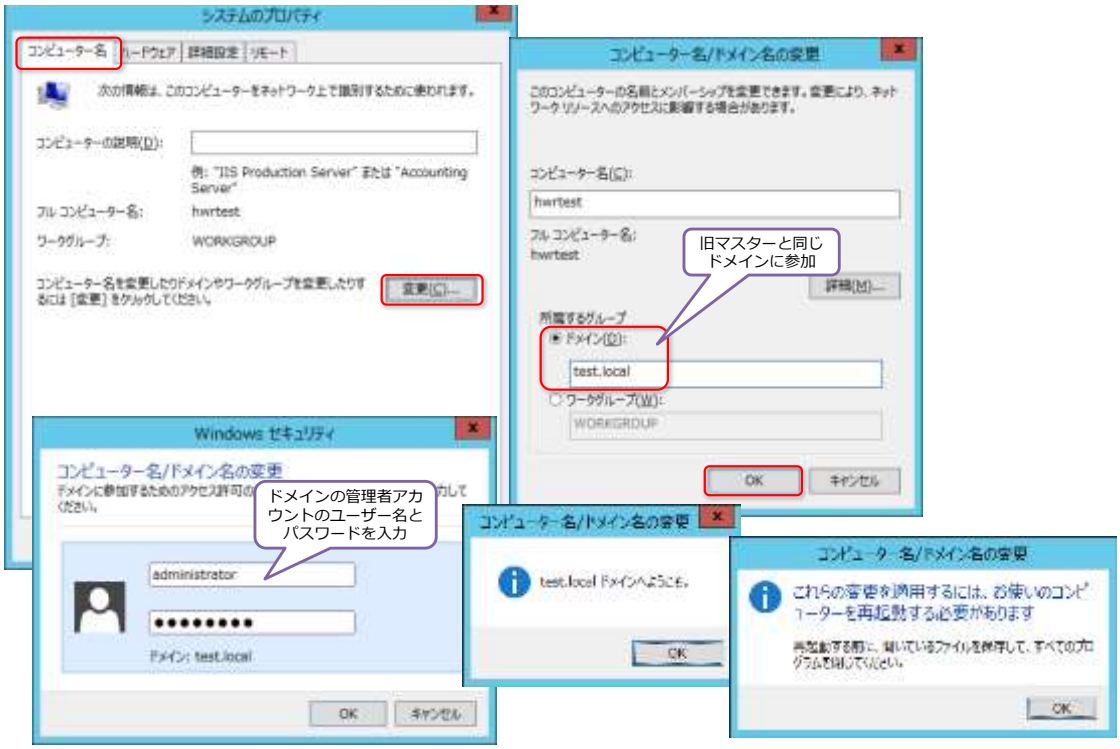
2016 にアップグレードするためには **SP4** 以上)、**SQL Server 2008 R2** なら **SP1** 以上 (SQL Server 2016 にアップグレードするためには **SP3** 以上) を適用しておく必要があります。

したがって、この Service Pack は、旧マスター上で適用しておく必要があります (∵丸ごと複製作業には、旧マスターと新規サーバーで同じ Service Pack を適用しておくことが重要になるため)。

なお、新規サーバーへの OS のインストールは、一番最初の作業 (手順 1 の旧マスターでのオフライン バックアップよりも前) として行っておくこともできます。この場合は、**任意のマシン名**を設定してインストールしておき、この手順のタイミング (現在のマスターをネットワークから切り離したタイミング) で**マシン名を旧マスターと同じ名前**へ変更します。あるいは、新規サーバーをネットワークから切り離しておいて、OS をインストールし、マシン名を旧マスターと同じ名前にしておきます (この場合は、旧マスターが起動している間は、新規サーバーをネットワークへ接続しないように注意します)。

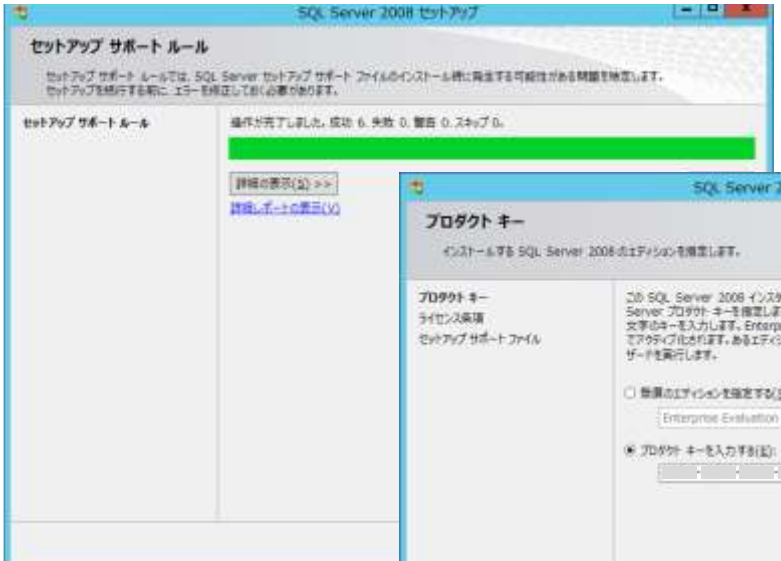
## 6. 新規サーバーを Active Directory ドメインへ参加させる

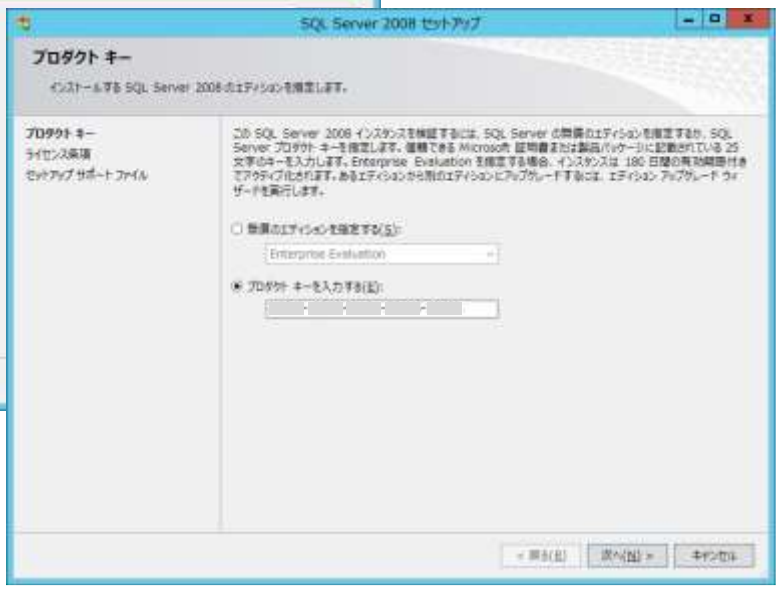
次に、新規サーバーを Active Directory ドメインへ参加させます。

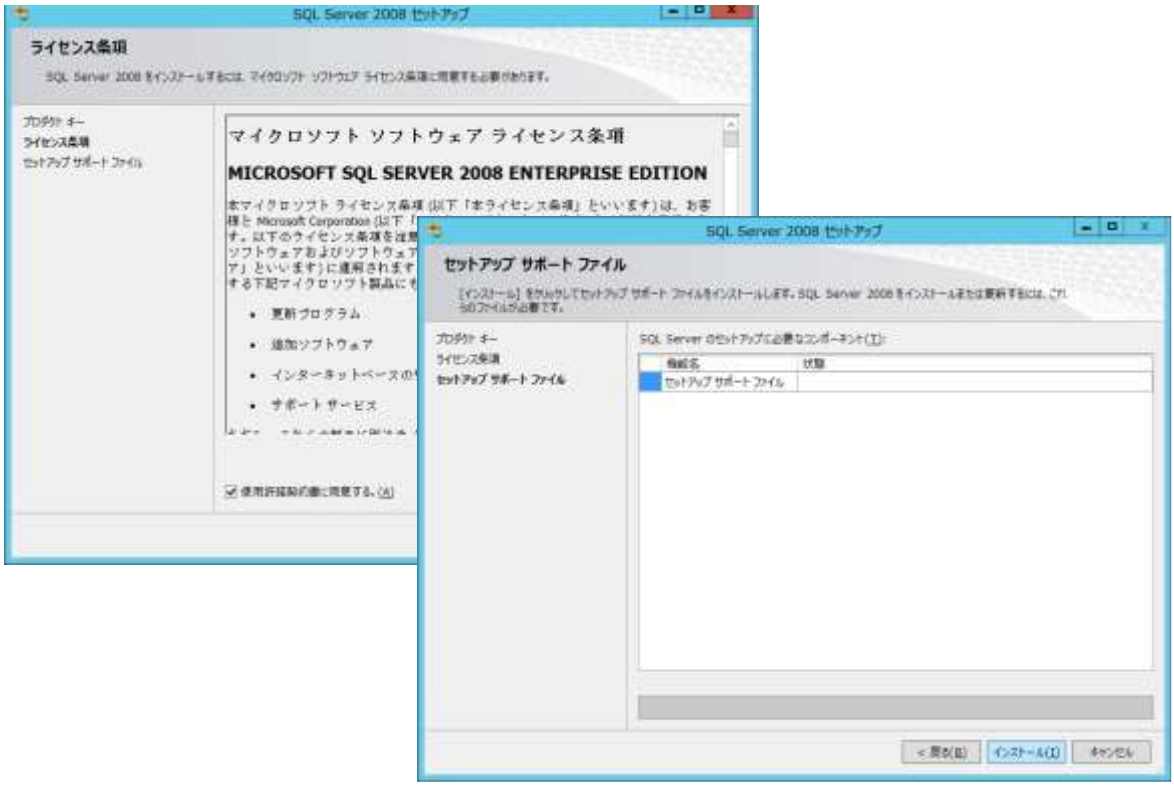


## 7. 新規サーバーへ旧マスターと同じ SQL Server をインストールする

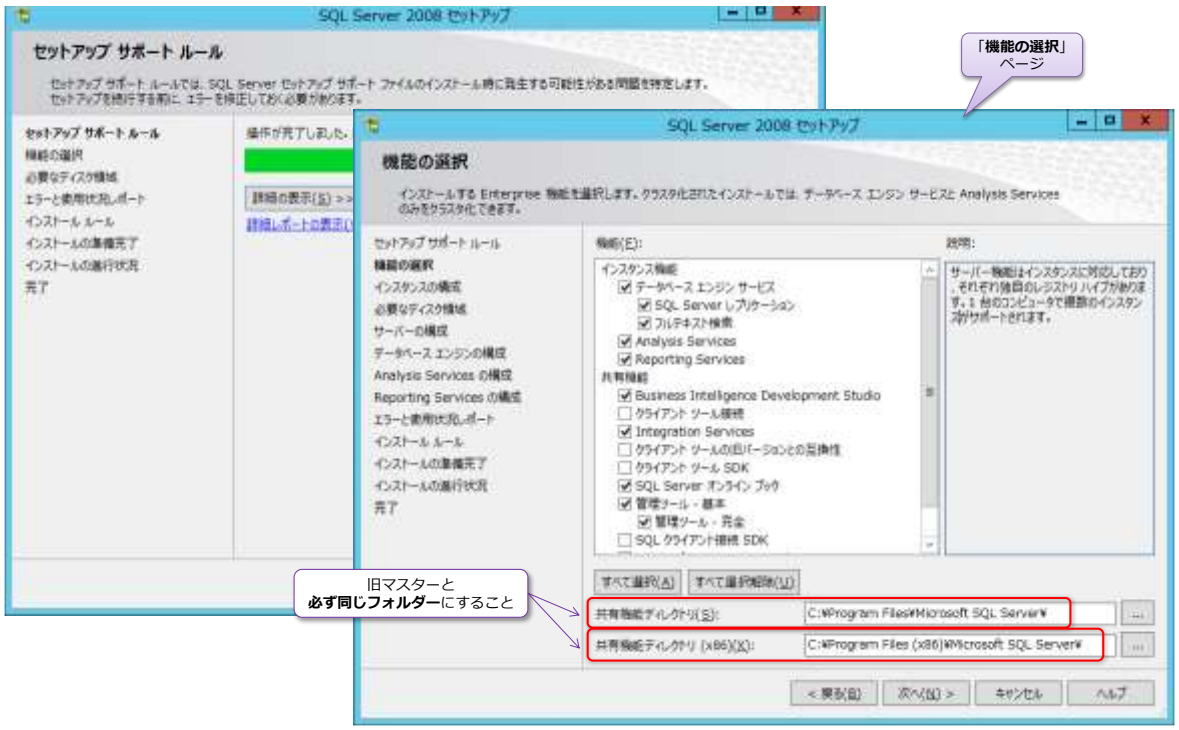
次に、新規サーバーへ**旧マスターと同じバージョンの SQL Server** をインストールします（**SQL Server 2016** をインストールするわけではないことに注意してください）。旧マスターが **SQL Server 2008** の場合には、次のようにインストールします。

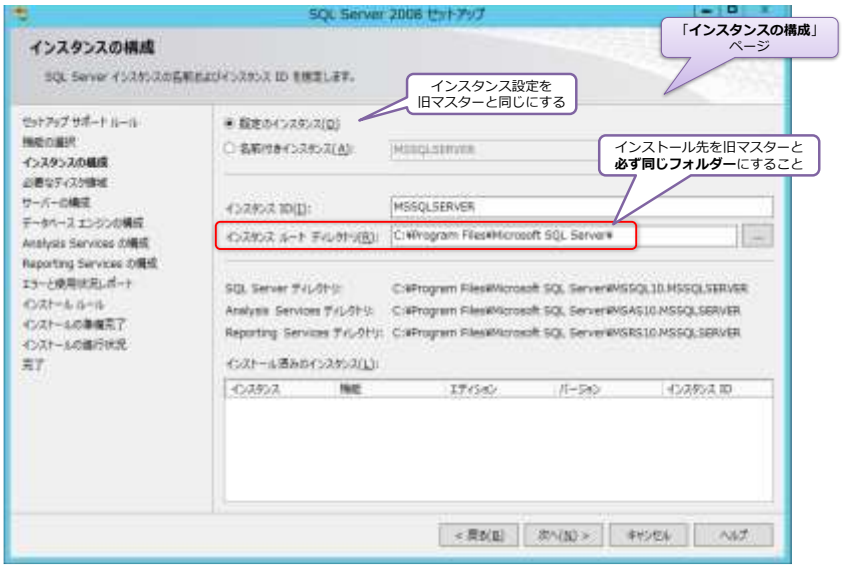



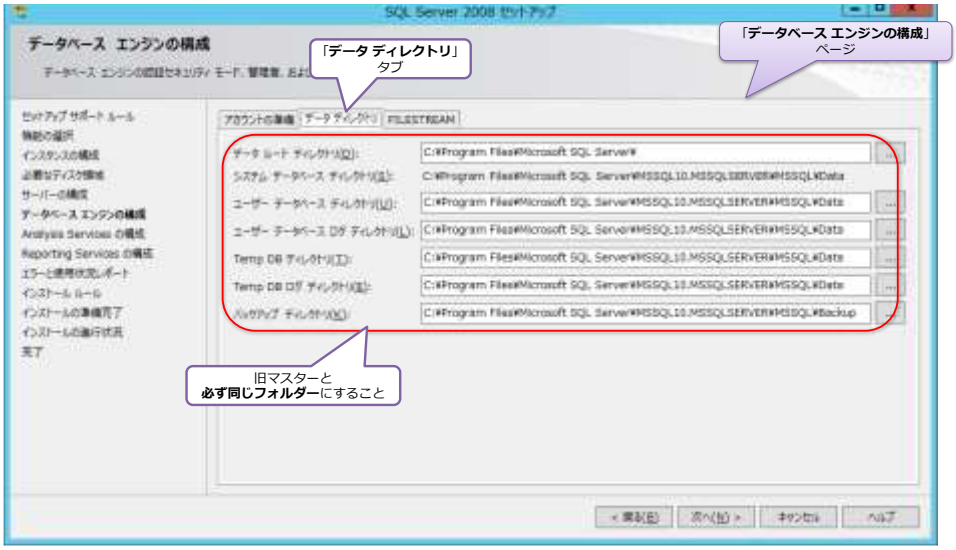






SQL Server をインストールする際の注意点は、インストール先のフォルダーを、旧マスターをインストールしたときのパスとまったく同じにするという点です。SQL Server 2008 であれば、インストール先のフォルダーは、以下の［**機能の選択**］と［**インスタンスの構成**］、［**データベース エンジンの構成**］ページで設定しています。

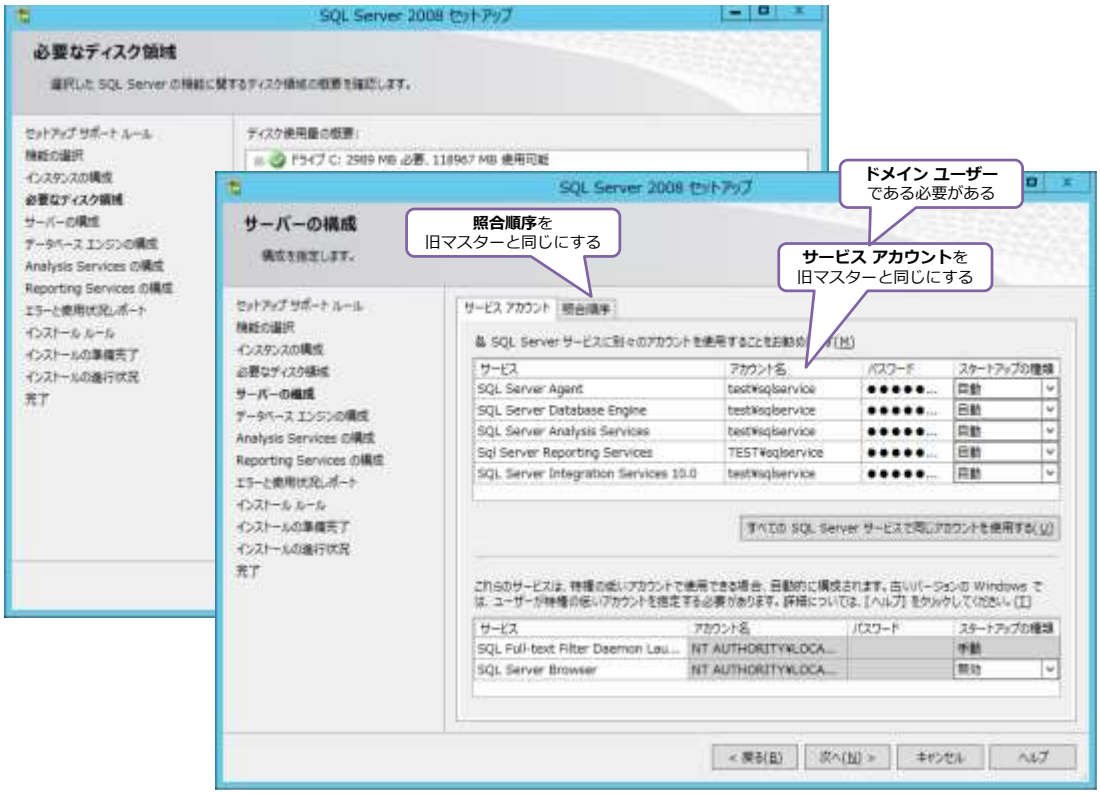


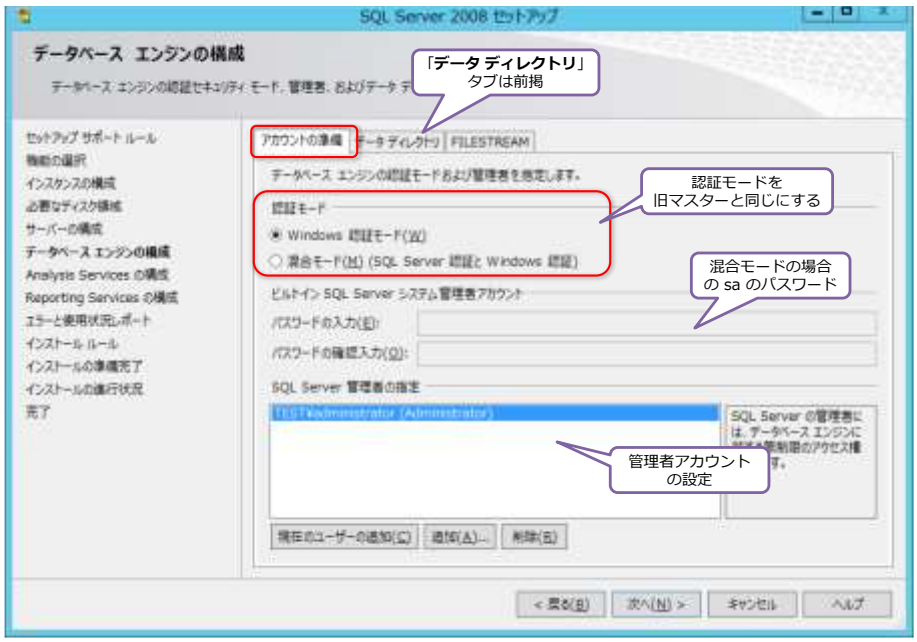


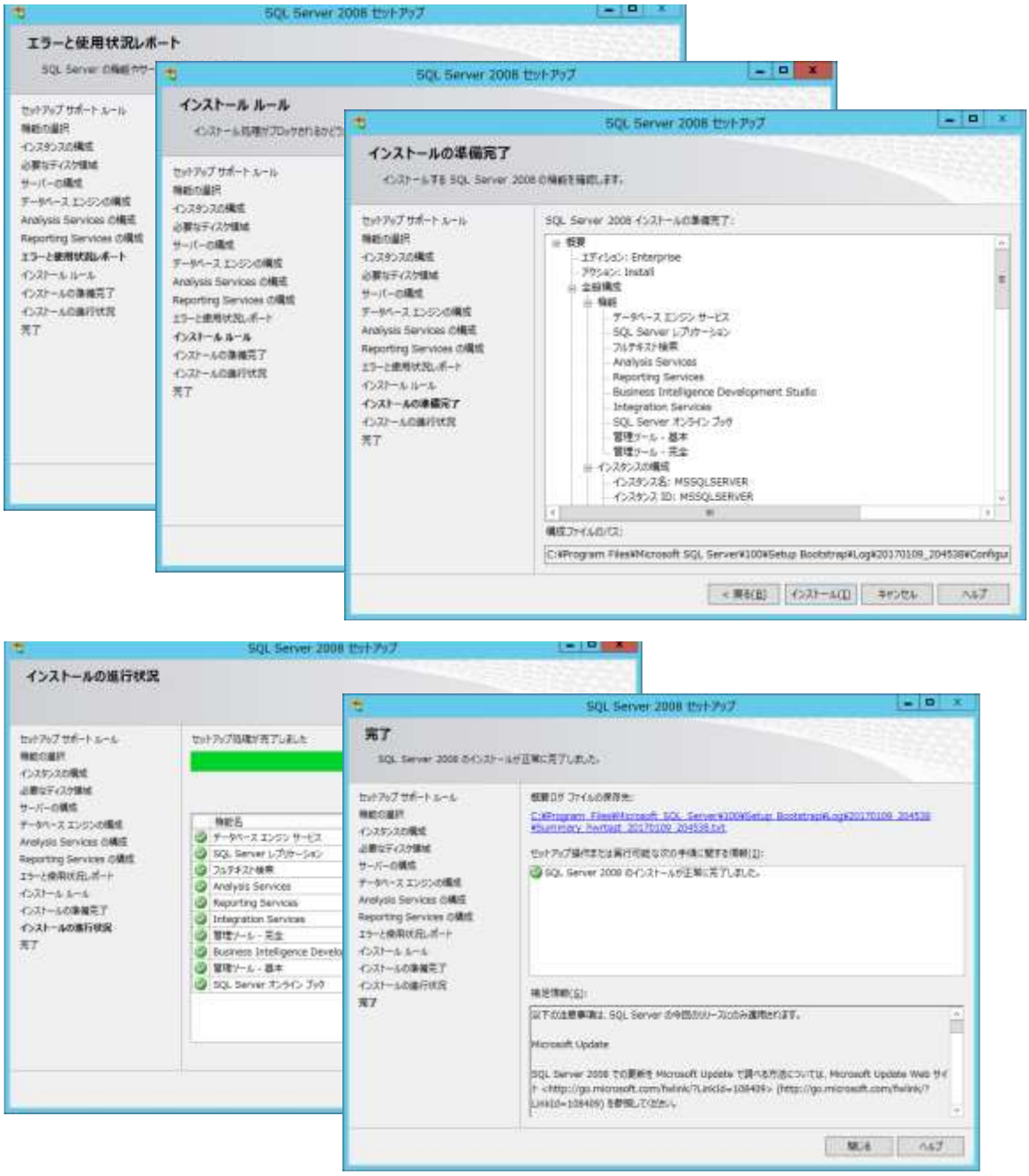




もし、インストール先のフォルダーを旧マスターと違うものにしてしまうと、後述の手順でオフライン バックアップを復元した後に、SQL Server サービスが正しく起動しなくなるので、注意してください（∵master データベース内には、msdb や tempdb、ユーザー データベースへのファイル パスが格納されているので、インストール先のフォルダーが異なるとデータベースを認識することができなくなります）。

また、インストール時は、インスタンス名やサービス アカウント、認証モード、照合順序なども旧マスターと同じ設定にする必要があります。**サービス アカウント**は、**ドメイン ユーザー**である必要があり、旧マスターで設定しているドメイン ユーザーと同じにする必要もあります。


SQL Server 2008 セットアップ
必要なディスク領域
SQL Server 2008 セットアップ
サーバーの構成
照合順序を
旧マスターと同じにする
ドメイン ユーザー
である必要がある
サービス アカウントを
旧マスターと同じにする

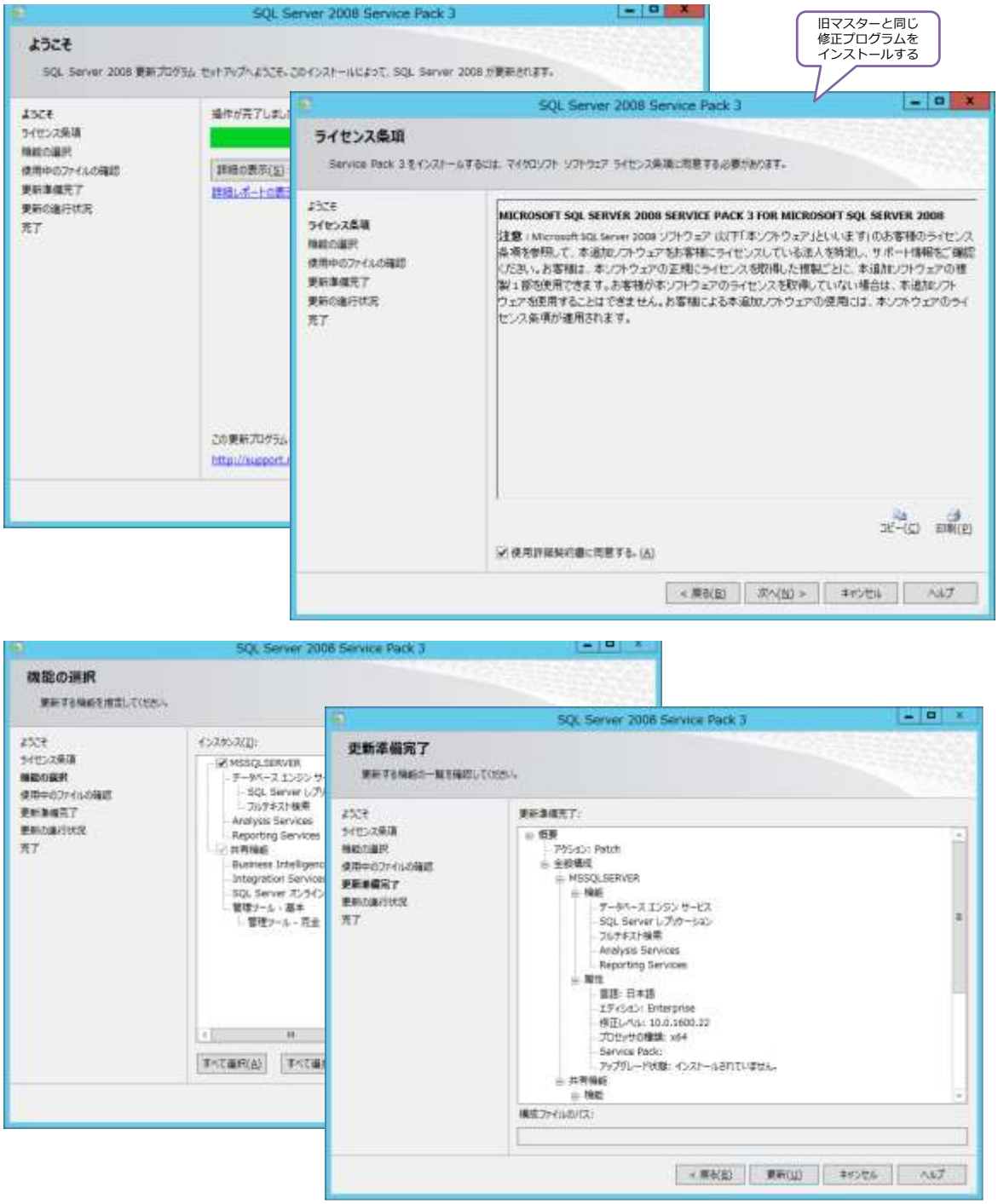

SQL Server 2008 セットアップ
データベース エンジンの構成
「データ ディレクトリ」
タブは前掲
認証モードを
旧マスターと同じにする
混合モードの場合
の sa のパスワード
管理者アカウント
の設定




完全複製（後述のオフライン バックアップからの復元）をするには、**旧マスターと全く同じ**ように SQL Server をインストールすることが非常に重要になります。

## 8. 旧マスターへインストール済みの修正プログラムをインストールする

SQL Server のインストールが完了した後は、旧マスターへインストール済みの **SQL Server の修正プログラム**（**Service Pack** や **CU** など）を、新規サーバーにもインストールします。

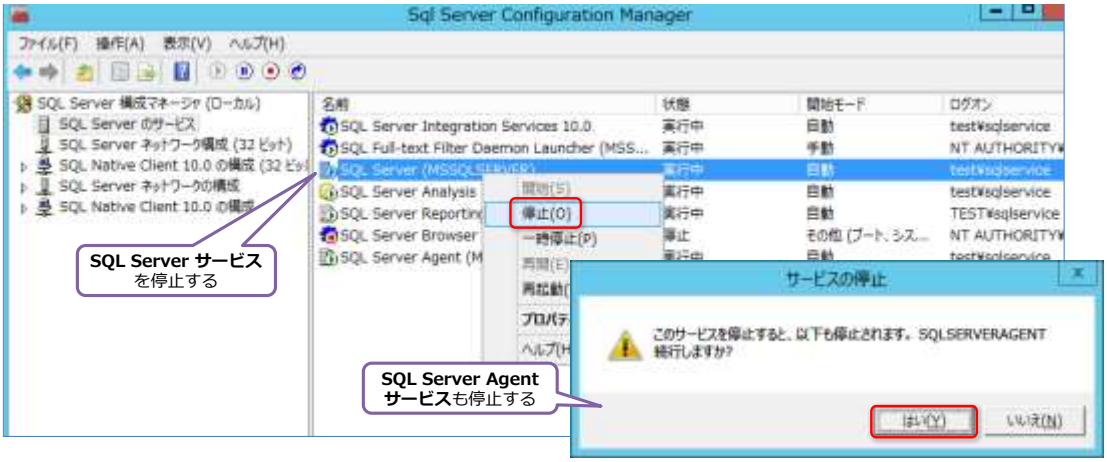



このときにインストールする修正プログラムは、旧マスターへインストールしているものとまったく同じもののみをインストールすることに注意してください。例えば、旧マスターが SQL Server 2008 の SP3 である場合は、SP3 をインストールします。

もし、旧マスターよりも最新版の修正プログラムをインストールした場合には、オフライン バックアップからの復元後に、SQL Server サービスが正しく起動しない場合があるので、必ず同じものをインストールするようにします（∵システム データベースは、修正プログラムの影響を受けるため、修正プログラムのバージョンが異なると SQL Server サービスが起動しなかったり、バックアップからの復元で失敗する可能性があります）。

## 9. 新規サーバーの SQL Server を停止する

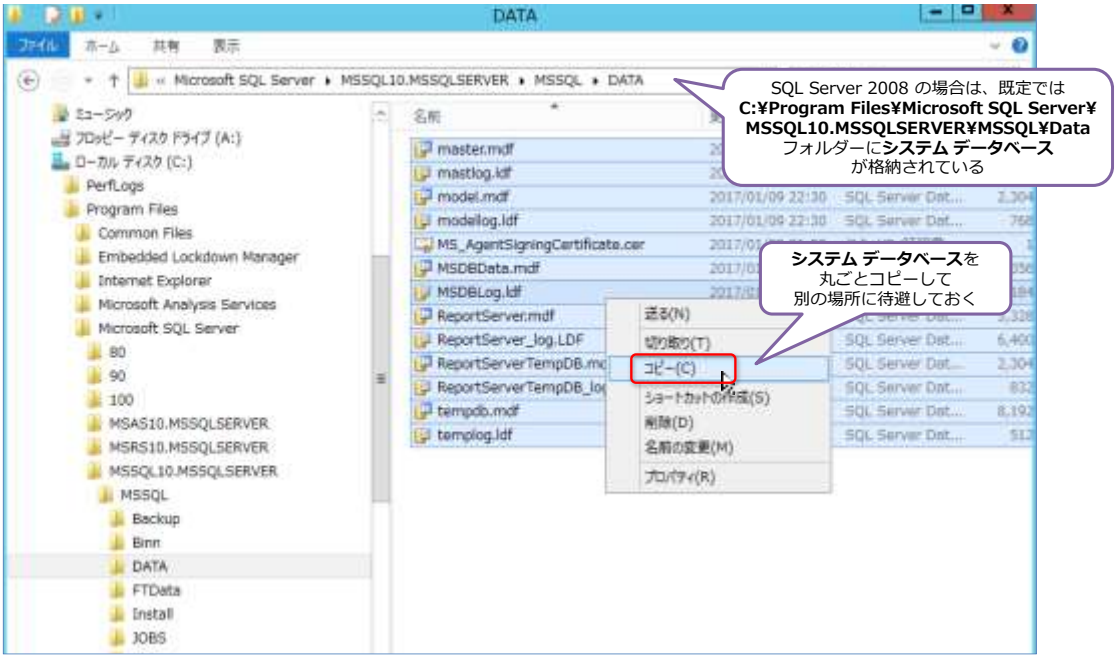
次に、旧マスターで取得したオフライン バックアップを新規サーバーへ適用するために、**SQL Server サービスを停止**します。SQL Server サービスは、**構成マネージャー** ツールを利用して、次のように停止します（**SQL Server Agent サービス**も停止します）。




フルテキスト検索機能を利用している場合は、**SQL Full-text Filter Daemon Launcher** サービスも停止します。

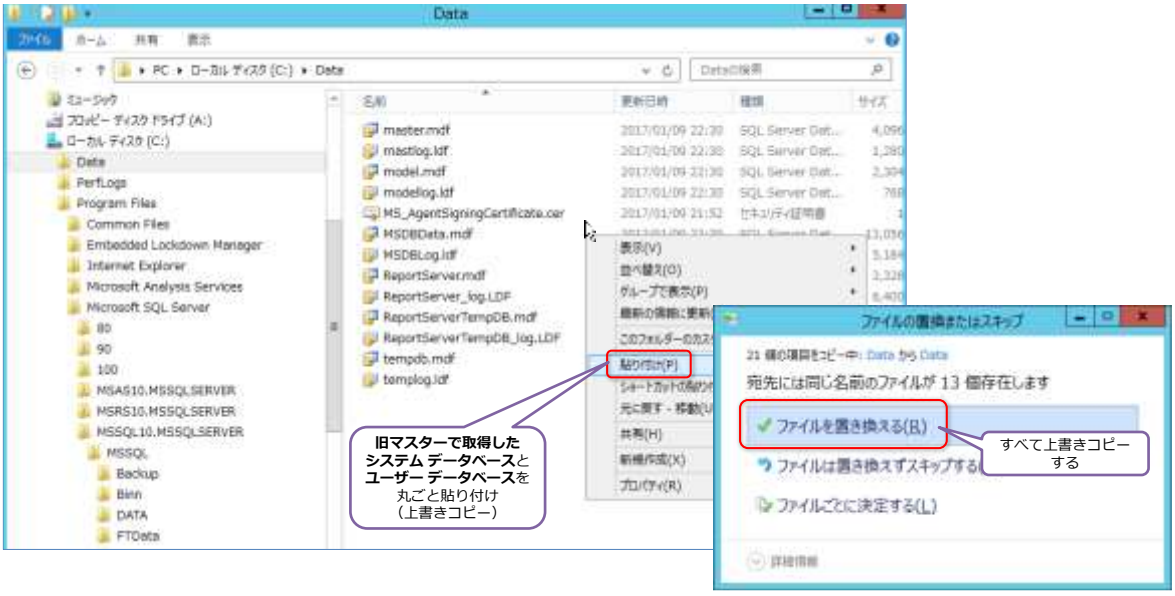
## 10. 新規サーバーのシステム データベースを丸ごとオフライン バックアップする

この作業は、万が一、次の手順が失敗したときのために、念のため実施します。SQL Server のインストールおよび修正プログラムを適用した後の、現在の**システム データベース**をオフライン バックアップしておき、いつでもこの状態へ戻せるようにしておきます。



## 11. 旧マスター上で取得したオフライン バックアップを新規サーバーへ復元する

次に、旧マスターで取得した**すべてのデータベース（システム データベースとユーザー データベース）**の**オフライン バックアップ（.mdf と .ldf）**を、Windows エクスプローラーから新規サーバー上にすべてコピー（上書きコピー）します。

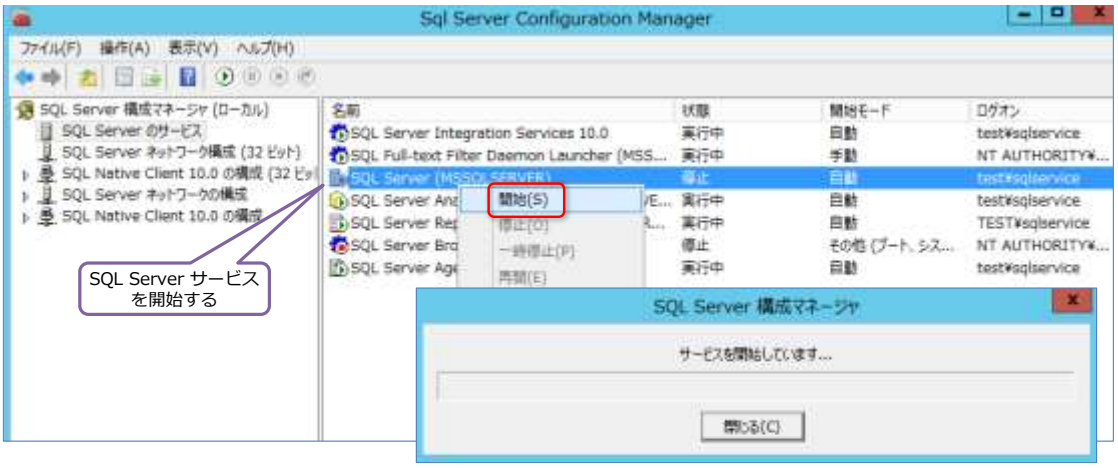



システム データベースは同じファイルが存在しますが、前の手順で念のためバックアップをしているので、安心して上書きコピーしてください。

なお、ユーザー データベースをオンライン バックアップで取得している場合は、次の手順で SQL Server を起動した後に、RESTORE ステートメントでバックアップから復元します。

## 12. 新規サーバーの SQL Server を起動する

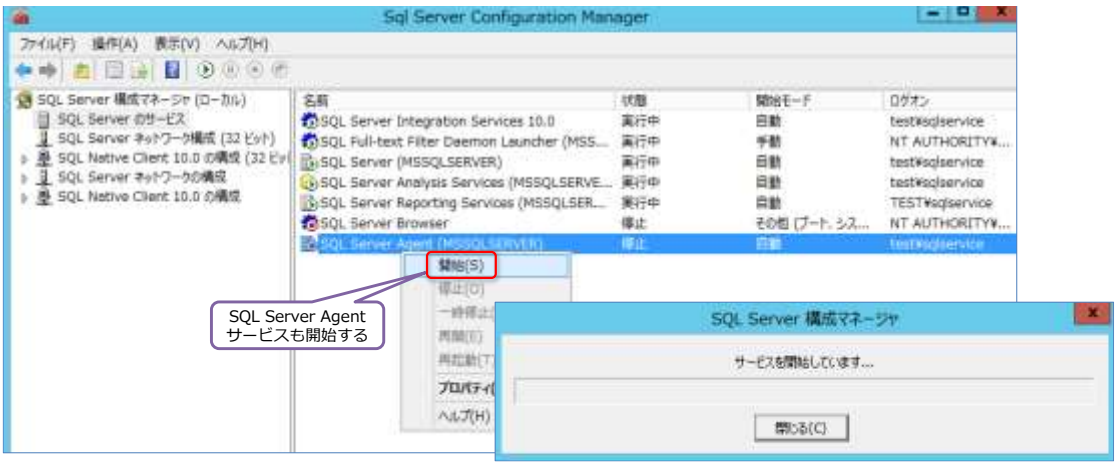
上書きコピーが完了したら、新規サーバーの **SQL Server サービスを起動**します。



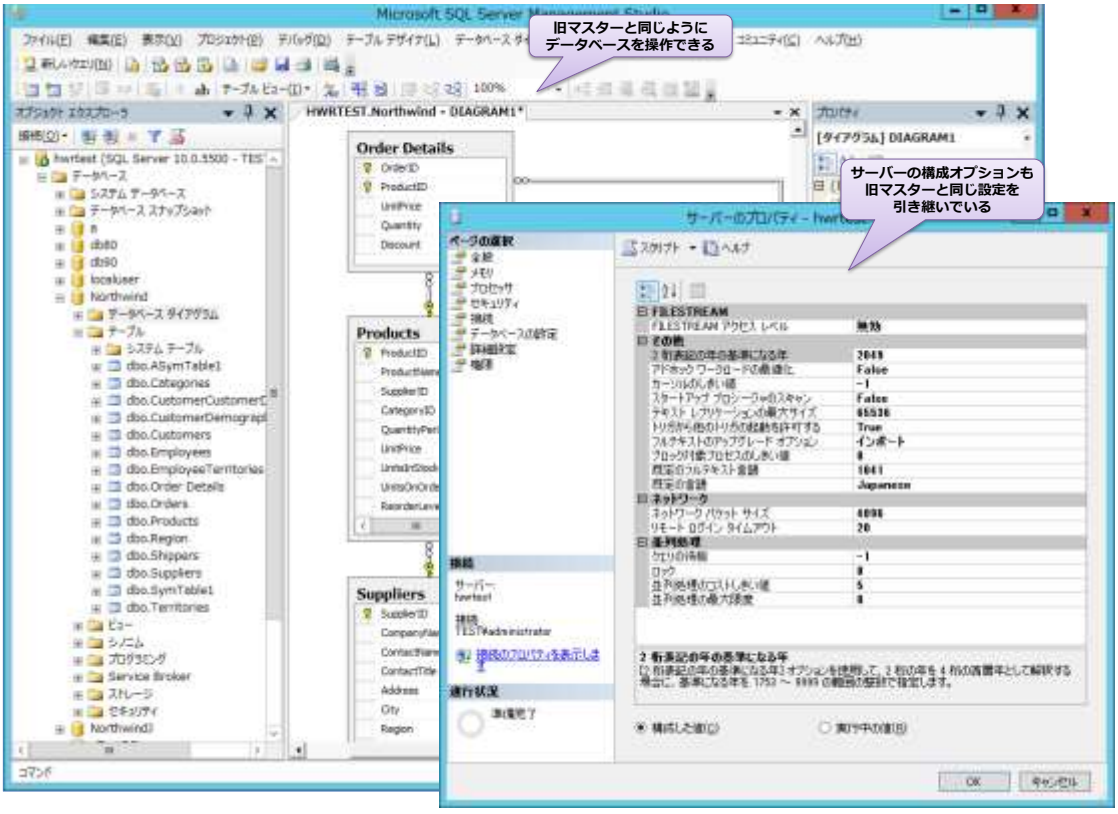

もし、SQL Server サービスが起動しない場合は、OS のフォルダー構成やドライブ構成が、旧マ

スターと同じになっていない場合（tempdb データベースを別ドライブへ移動しているなど）に起こり得ます（OS のフォルダー構成やドライブ構成は、旧マスターと全く同じにしておく必要があります）。

SQL Server サービスが起動した後は、**SQL Server Agent サービス**も起動します。




以上で、（これだけで）**旧マスターとほぼ同じ状態の SQL Server** として動作させることができます。**データベース**を以前と変わらずに利用できることはもちろん、**ログイン アカウント**や**オブジェクト権限**、**データ パーティション**、**ジョブ**、**警告**、**暗号化**、**リンク サーバー**、**メンテナンス プラン**（保守計画）、**リソース ガバナー**、**監査**（SQL Server Audit）、**データベース メール**、**サーバーの構成オプション**、**パフォーマンス データ コレクション**、**TDE**（透過的なデータ暗号化）など、旧マスターで設定／利用していた機能を、そのまま新マスター上でも利用することができます。



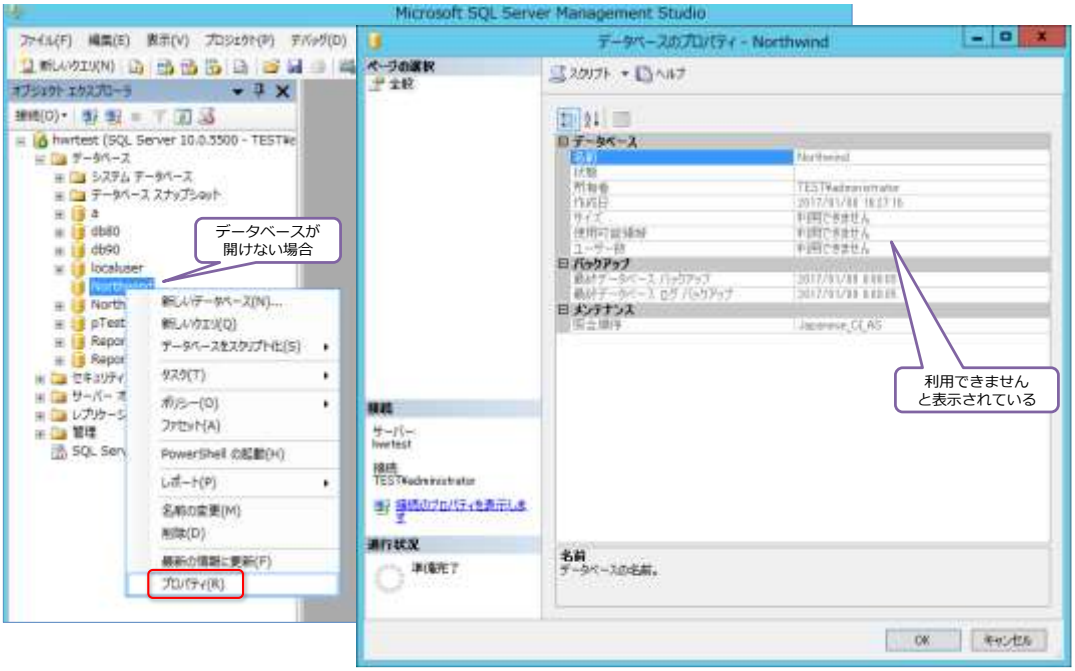

SQL Server は、起動時に **master** データベースから各種の動作情報を取得するので、**システム**

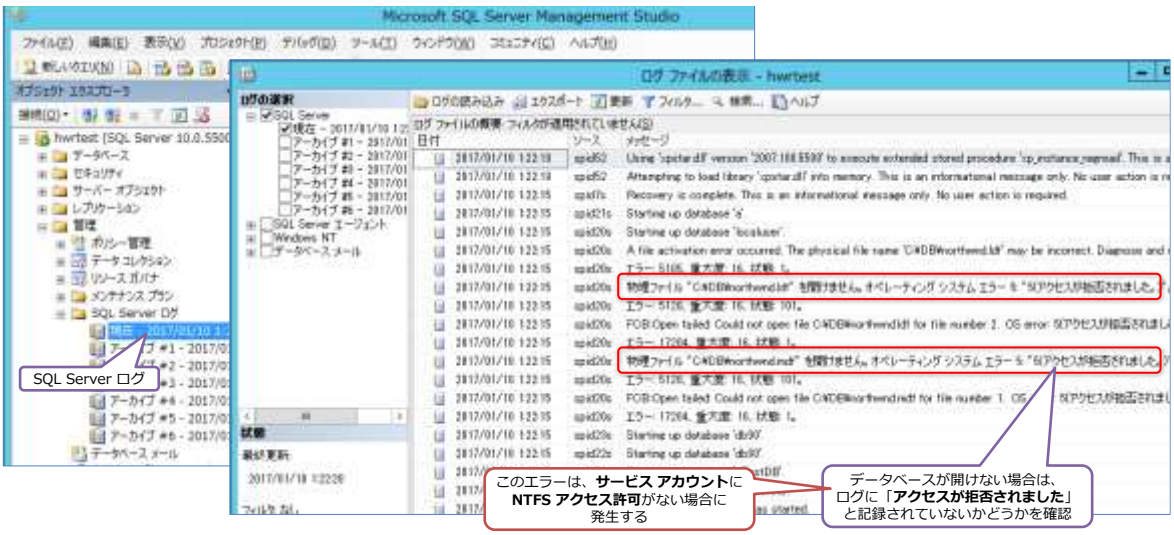
**データベース**を上書きするだけで、環境をガラリと変更することができます（システム データベースに含まれているものに関しては、第５章でも詳しく説明しています）。

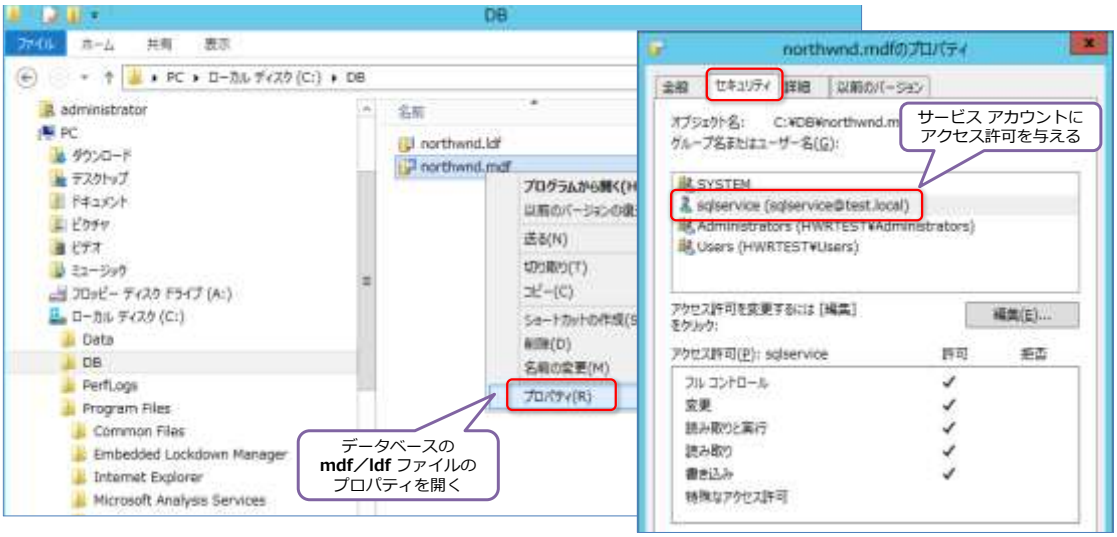
もし、SQL Server サービスや SQL Server Agent サービスが起動しない場合には、手順７ でのインストール パスが正しいこと（旧マスターと同じパスであること）などを確認してみてください。また、旧マスターで **tempdb** データベースを別ドライブへ移動している場合には、そのドライブと同じドライブ文字のドライブを新規サーバー上でも作成しておく必要があります。

## サービス アカウントに NTFS アクセス許可があることの確認

SQL Server サービスおよび SQL Server Agent サービスを開始した後に、次のようにデータベースが開けない場合は、**SQL Server サービス**の**サービス アカウント**に対する **NTFS アクセス許可**があるかどうかを確認してみてください。

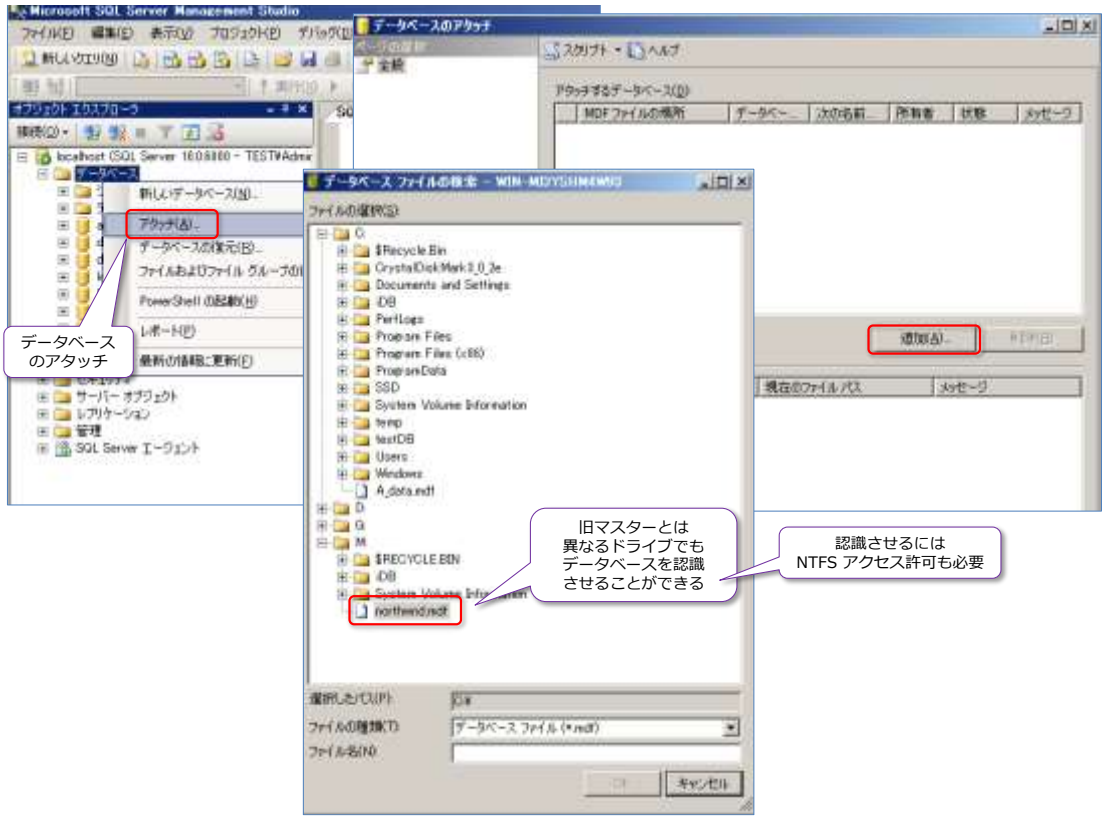





## ユーザー データベースを同じドライブへ配置できない場合（アタッチで認識することも可能）

もし、新規サーバー上で同じドライブを作成できない場合には、前述と同様の「**ファイルにアクセスできない**」エラーが表示されて、ユーザー データベースを開くことができません。この場合は、データベース（**.mdf／.ldf ファイル**）を認識させられる代替案があります。例えば、旧マスターで **F:** ドライブに作成していたユーザー データベースが、新規サーバーでは **M:** ドライブになっているような場合には、次のように「**アタッチ**」操作をすることで、新しい場所にある **.mdf／.ldf ファイル**をデータベースとして認識させることができます（アタッチ前に、開けなかったデータベースを削除してから、アタッチ操作を実行します）。



## 追加の作業（レジストリ、OS の設定）

以上で、ほとんどの機能をそのまま動作させることができますが、次の 2 つの作業を追加で行っておく必要があります。

- **レジストリに格納されている情報を再設定する**
  TCP ポート番号や起動時パラメーターでのトレースフラグの設定などのうち、旧マスターで設定を変更しているものがある場合は、それらを再設定する
- **OS の設定で、旧マスターで変更しているものがある場合は、それらを再設定する**
  フォルダー構成や NTFS アクセス許可、ユーザーの権利、共有フォルダーなど

OS を変更したことによって、レジストリに格納される設定や、OS の設定に関しては、再設定が必要になります（どういったものに再設定が必要になるかは、後述します）。

## Windows のローカル ユーザーを利用している場合の注意点

旧マスター環境で、次のように **Windows のローカル ユーザー**を利用している場合には、注意点があります。

- **データベースの所有者**が **Windows のローカル ユーザー**の場合には、SQL CLR オブジェクトで権限セットを「**UNSAFE**」または「**外部**」に設定しているものが動作しない。また、データベース ダイアグラムも表示できない
- **ジョブの所有者**が **Windows のローカル ユーザー**の場合には、ジョブが実行エラーになる
- **ログイン アカウント**が **Windows のローカル ユーザー**の場合は、**ログイン アカウント**およびそれに紐付いた**データベース ユーザー／オブジェクト権限／データベース ロール**が動作しなくなる
- **管理者アカウント**（sysadmin ロール）が **Windows のローカル ユーザー**のみの場合で、認証モードが「**Windows 認証**」の場合には、SQL Server へログインできるユーザーがいなくなってしまう（なお、SQL Server 2005 のときには、既定で Administrators グループが sysadmin ロールとして登録されていたが、SQL Server 2008 以降は、インストール時に管理者に設定したアカウントのみが sysadmin になる）
- **サービス アカウント**が **Windows のローカル ユーザー**の場合には、サービス マスター キーが関連するものが動作しなくなる（リンク サーバーやデータベース メール、透過的なデータ暗号化など）
- その他、**Windows のローカル ユーザー**を利用している設定があるものは動作しない

Windows のローカル ユーザーは、OS が変わると別のユーザーになるため（内部的な Security ID が異なるため）、こういったことが起こりますが、これらの対処方法については、後述します。

## 4.3 レジストリに格納されている情報の再設定

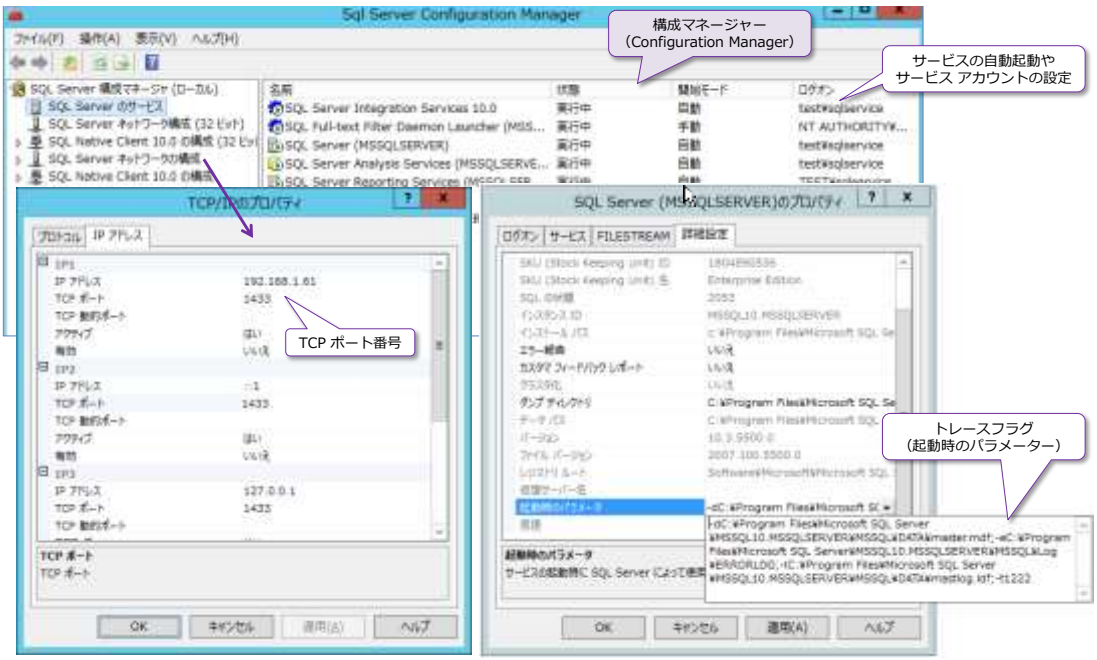
新規サーバー（新マスター）では、**レジストリ**に格納されている情報も再設定する必要がありますが、レジストリに格納されている情報はほとんどありません（∵SQL Server の動作に関する設定は、ほとんどがシステム データベース内に格納されているため）。

レジストリに格納されている主な情報は、次のとおりです。

| | レジストリに格納される主な情報 |
|---|---|
| **SQL Server サービス関連**<br>構成マネージャー（Configuration Manager）ツールで設定したものなど | ・サービスの自動起動の設定<br>・サービス起動時のトレース フラグ（起動時のパラメーター）<br>・サービス アカウントの設定<br>・セキュリティ モード（認証モード）<br>・TCP ポート番号<br>・ログイン監査設定 (成功／失敗のログインをログへ記録するかどうか) |
| **SQL Server Agent サービス関連**<br>SQL Server エージェントのプロパティで設定したものなど | ・サービスの自動起動の設定<br>・サービス アカウントの設定<br>・［警告システム］ページの設定（メール セッションでのデータベース メールの設定、緊急時のオペレーターなど）<br>・ジョブ履歴の最大行数<br>・予期しない停止時の自動再起動の設定<br>・パフォーマンス条件警告でのパフォーマンス カウンターの更新頻度<br>・CPU をアイドルとみなす秒数／使用率（ジョブのスケジュールで利用している場合） |

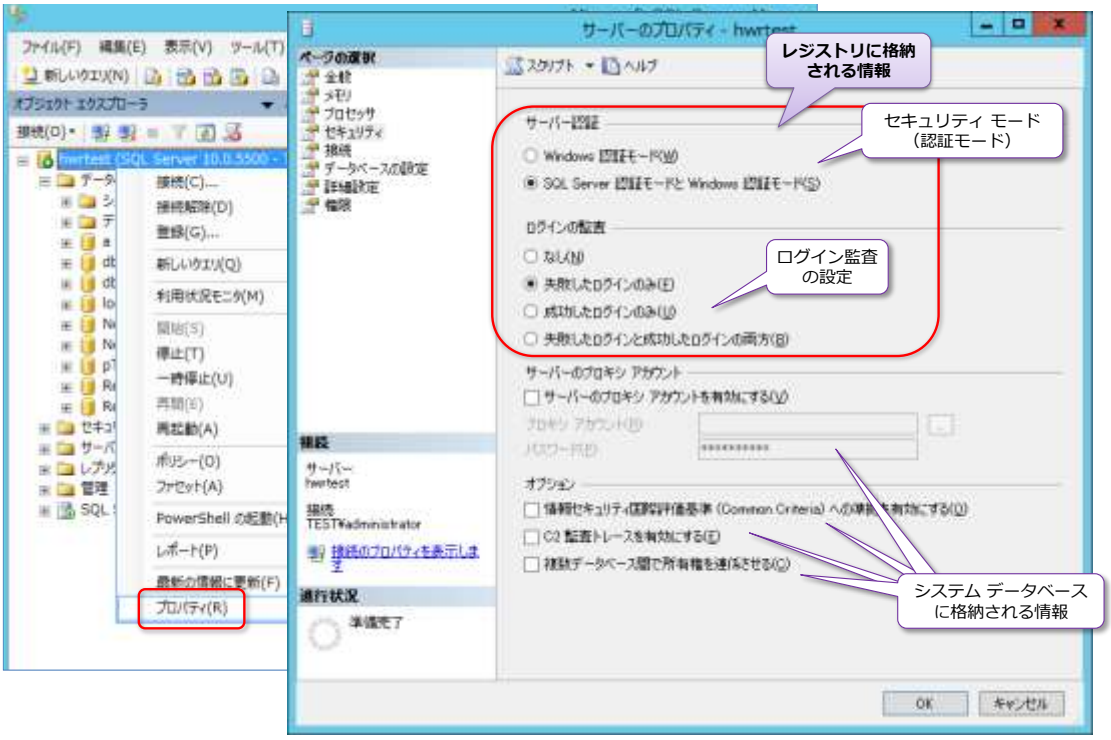
### SQL Server サービス関連の設定

SQL Server サービス関連の設定は、次のように「**SQL Server 構成マネージャー**」ツールで設定できる情報がほとんどで、**サービスの自動起動**や**サービス アカウント**の設定（SQL Server のインストール時に設定したもの）、**サービス起動時のトレース フラグ**（起動時のパラメーター）、**TCP ポート番号**などです（以下の画面は SQL Server 2008 の場合）。



弊社のお客様の場合は、デッドロックをログへ記録するためのトレース フラグ「**1222**」を設定している場合が多くあります。このように設定を変更している場合は、新規サーバー側でも再設定をする必要があります。

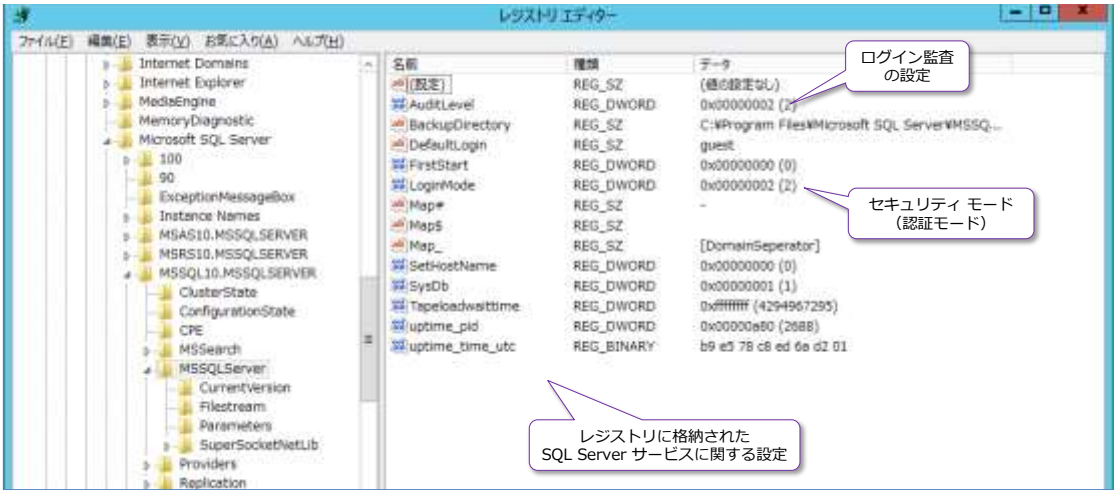
また、次のようにサーバーのプロパティで設定できる**認証モード**（セキュリティ モード）や、**ログイン監査の設定**などもレジストリに格納されています。




これらの設定を旧マスター上で変更している場合には、新規サーバーでも再設定をする必要があります。

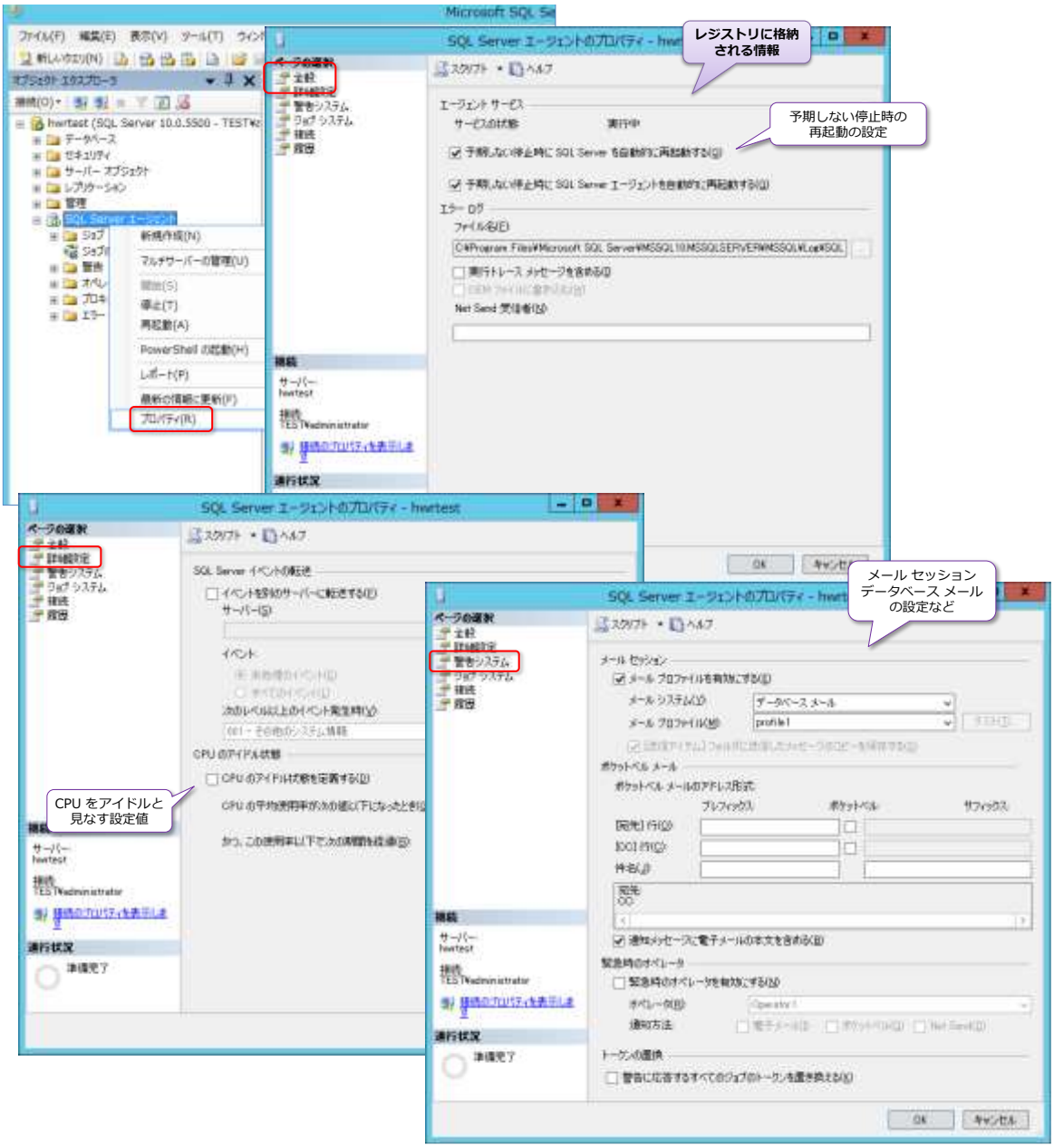
なお、これらのレジストリに格納される情報は、次のキーに格納されています（SQL Server 2008 の既定のインスタンスの場合）。

**HKEY_LOCAL_MACHINE¥SOFTWARE**
**¥Microsoft¥Microsoft SQL Server¥MSSQL.10.MSSQLSERVER**

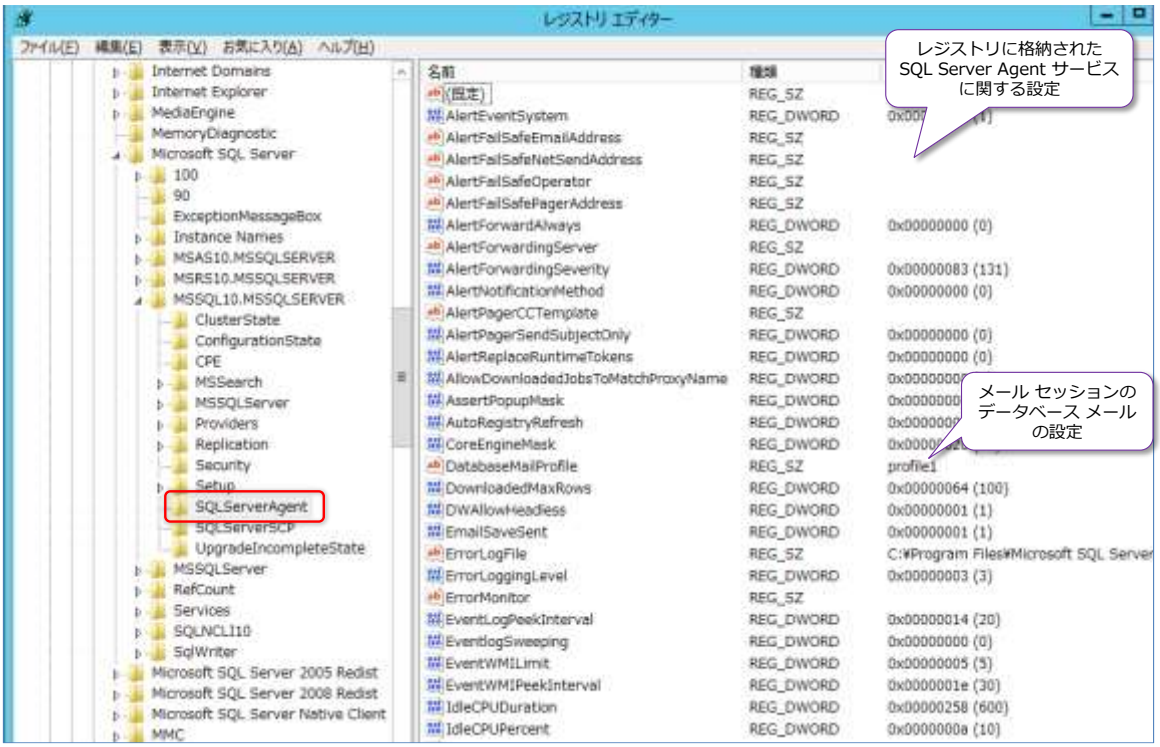


## SQL Server Agnet サービス関連の設定

SQL Server Agent サービス関連の設定は、SQL Server のインストール時に設定した**サービスの自動起動**や**サービス アカウント**、次のように［**SQL Server エージェント**］のプロパティで設定するものなどです。




［**全般**］ページでの予期しない停止時の自動再起動の設定や、［**詳細設定**］ページでの CPU をアイドルとみなす秒数／使用率、［**警告システム**］ページでの設定（メール セッションでのデータベース メールの設定、緊急時のオペレーターなど）、［**履歴**］ページでのジョブ履歴の最大行数などがレジストリに格納されています。




したがって、これらの設定を旧マスター上で変更している場合には、新規サーバー（新マスター）でも再設定をする必要があります。データベース メールの設定などは、利用している方がいらっしゃるのではないでしょうか。

以上のように、レジストリに格納される情報は少なく、あまり変更することのない設定が多いと思います。SQL Server の動作に関する設定は、ほとんどがシステム データベース内に格納されているので、（オフライン バックアップによって）システム データベースをきちんと複製することができれば、ほとんどの情報を複製することができます。

## 4.4 新規サーバーの OS の設定を再設定する

新規サーバー（新マスター）では、**OS の設定**についても、旧マスターで設定を変更しているものがある場合は、それらを再設定する必要があります。

**OS の設定**には、次のようなものがあります。

**1. 旧マスターで利用しているドライブを、新規サーバーにも作成する**

例えば、**tempdb** データベースに関しては、旧マスター側で移動している場合には、新規サーバー側にも同じドライブおよびフォルダー構成がないと、SQL Server サービスの起動に失敗してしまいます。tempdb は SQL Server サービスの起動時に再作成されるものなので、作成先が見つからない場合には、SQL Server サービスの起動に失敗してしまいます。

なお、**ユーザー データベース**に関しては、前述したように、データベースのアタッチ操作をすることで、異なるドライブであったとしても、データベースを認識させることができます。

**2. 旧マスターで作成しているフォルダーを、新規サーバーにも作成する**

例えば、**パフォーマンス データ コレクション**では、キャッシュ フォルダーを設定するので、キャッシュ フォルダーを作成しておかないとエラーになります。

また、**SQL Server Audit** で監査ログをフォルダー／ファイルへ記録している場合は、そのフォルダーが必要になります。

**3. 旧マスターで、OS のファイルとして実装していたものを、新規サーバー上にも作成する**

例えば、**SSIS パッケージ**（**.dtsx**）を、SQL Server（msdb）内に保存する形ではなく、ファイルとして OS 上に保存している場合は、同じファイルが新規サーバー上にも必要になります（SQL Server 内に保存したものは、msdb データベースをオフライン バックアップから復元することで、復元することができます）。

また、**SSIS パッケージ**や **bcp**、**BULK INSERT** などで取り込むファイルがある場合には、それらが新規サーバー上でも同じフォルダー内に存在している必要があります。

**4. 旧マスターで NTFS アクセス許可を設定している場合は、新規サーバーでも設定する**

前述したように、**サービス アカウント**に対する NTFS アクセス許可がない場合には、データベースを認識することができないので、NTFS アクセス許可を設定しておく必要があります。

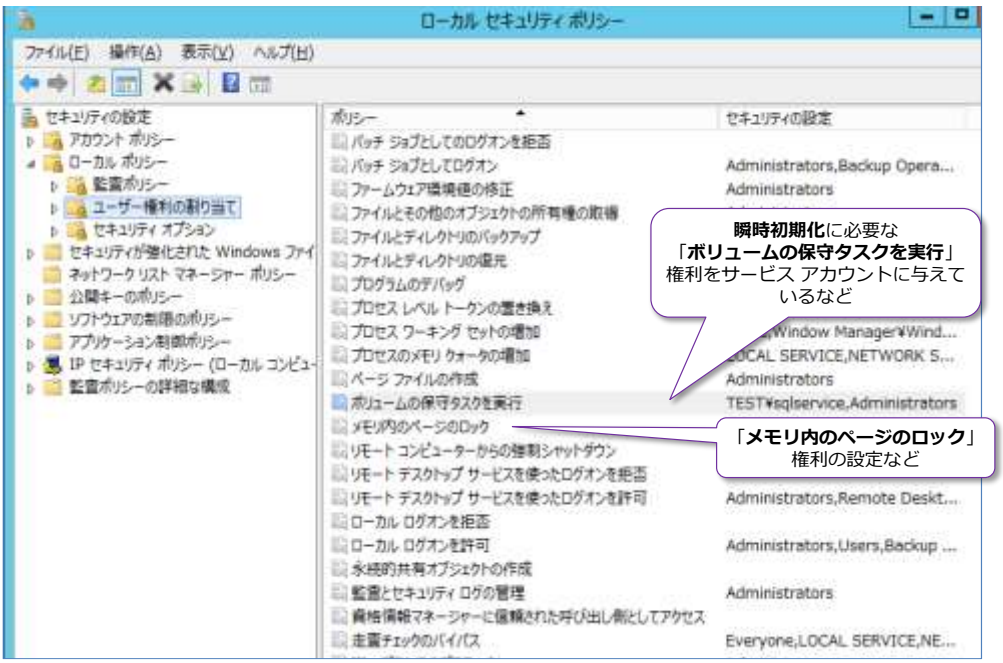
その他、**バックアップ先となるフォルダー**や、**SSIS パッケージ／bcp／BULK INSERT** などでファイルをインポートしている場合には、それらに対する NTFS アクセス許可が必要になります。

**5. 旧マスターで共有フォルダーを作成している場合は、新規サーバーにも作成する**

例えば、バックアップやログ配布などで、共有フォルダーを利用している場合には、新規サーバーにも同じように作成するようにします。

### 6. 旧マスターでユーザーの権利を変更している場合には、新規サーバーでも変更する

例えば、次のように、**瞬時初期化**に必要となる「**ボリュームの保守タスクを実行**」権利をサービス アカウントに設定していたり、「**メモリ内のページのロック**」権利を設定している場合には、新規サーバーでも同じように設定するようにします。




ユーザーの権利は、[**管理ツール**] メニューの [**ローカル セキュリティ ポリシー**] ツールから変更することができます。

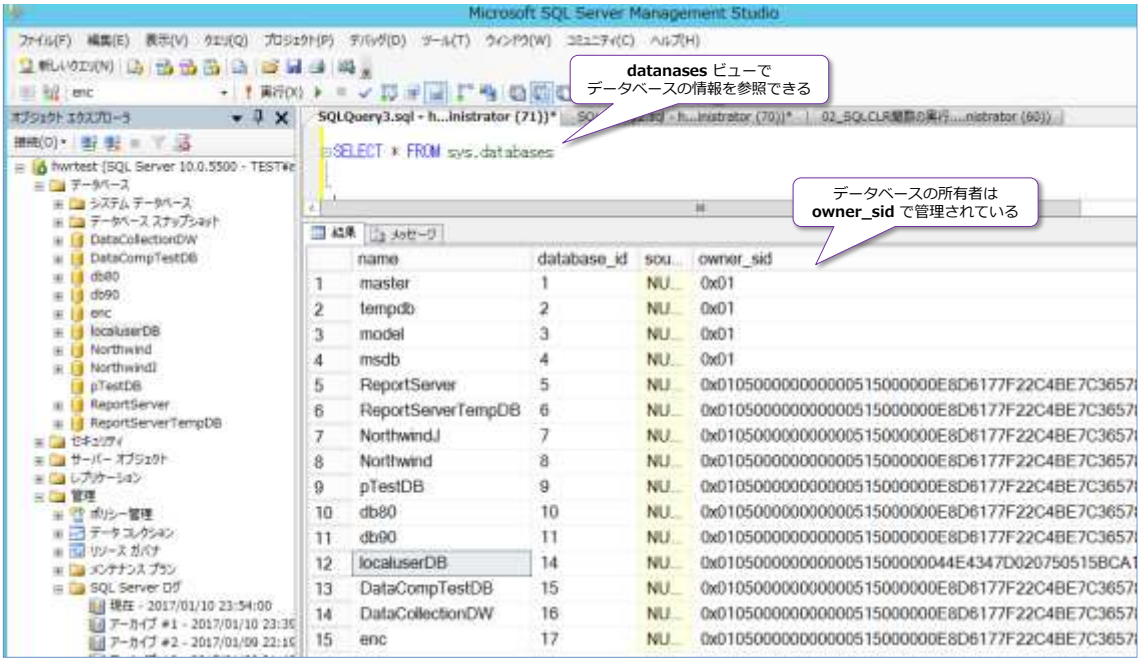
以上で、**ハードウェア リプレース時の完全複製作業（旧マスター環境を新規サーバーへ丸ごと複製**）が完了です。このように、**システム データベースのオフライン バックアップ**を利用すれば、後はレジストリと OS の設定を再設定するだけで、非常に簡単にハードウェア リプレース／同じ名前の新規サーバーの構築を行うことができます。

## 4.5 データベースの所有者が Windows のローカル ユーザーの場合

旧マスターでの**データベースの所有者**が **Windows のローカル ユーザー**（**マシン名¥ユーザー名**形式のユーザー）の場合には、次の 2 つの機能が動作しません。

- **SQL CLR オブジェクト**で、権限セットを「**UNSAFE**」（アンセーフ）または「**外部**」（EXTERNAL_ACCESS）に設定しているものが動作しない（**SAFE** に設定しているものは動作する）
- **データベース ダイアグラム**を表示できない

この問題は、データベースの所有者が、次のように **owner_sid**（所有者の Security ID）で管理されていることによって発生しています。



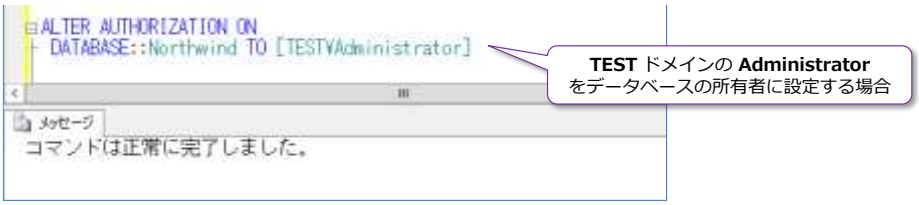

Windows のローカル ユーザーは、マシン（OS）が変わったとすると、同じ名前のユーザーを作成したとしても、**内部的な SID（Security ID）**が異なってしまうので、該当 SID が存在しないという状態になります。

これによって、冒頭の 2 つの機能（SQLCLR とデータベース ダイアグラム）がうまく動作しないという事態が発生します。これを解決するには、データベースの所有者を設定する必要があります。

### データベースの所有者を設定する

データベースの所有者を設定するには、次のように **ALTER AUTHORIZATION** ステートメントを実行します。

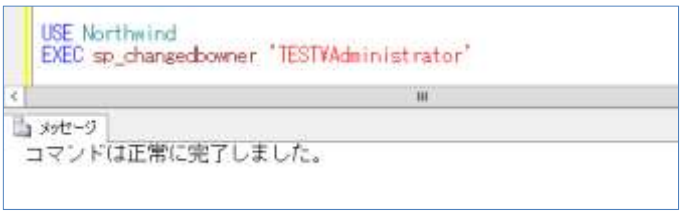
```
ALTER AUTHORIZATION ON
 DATABASE::データベース名 TO [所有者名]
```




ドメイン ユーザーを指定する場合は、[ ] で囲んで、**ドメイン名¥ユーザー名** 形式で指定します。

以前のバージョンで所有者名を変更するために利用できた「**sp_changedbowner**」を利用しても、次のように所有者名を変更することができますが、このストアド プロシージャは、将来のバージョンでは削除される予定なので、**ALTER AUTHORIZATION** ステートメントを利用することをお勧めします。

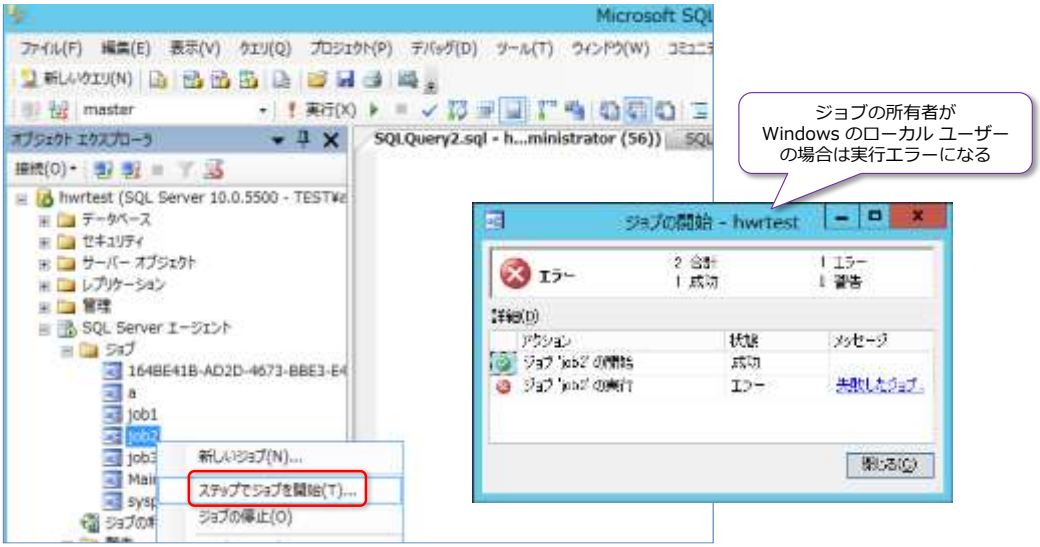
```
USE データベース名
EXEC sp_changedbowner '所有者名'
```

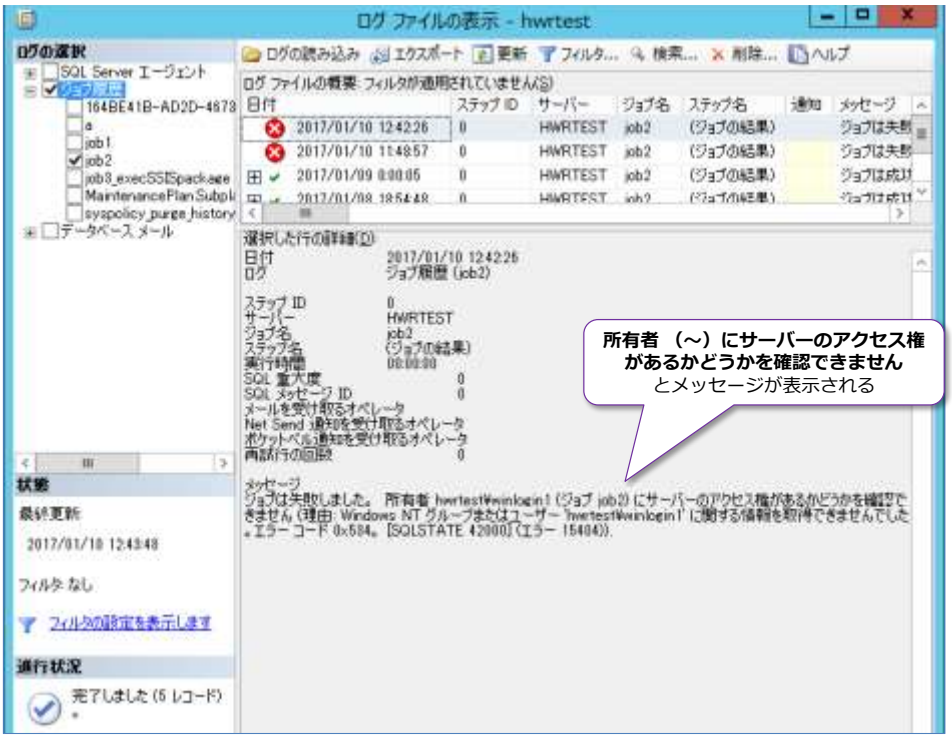



なお、データベースの所有者が Windows のローカル ユーザーになってしまうのは、Management Studio を利用するときに、Active Directory ドメインのユーザーではなく、Windows のローカル ユーザー（ローカルの Administrator など）でログインして操作している場合です。Windows ローカル ユーザーでログインした状態でデータベースを作成すると、データベースの所有者がそのユーザーになります。

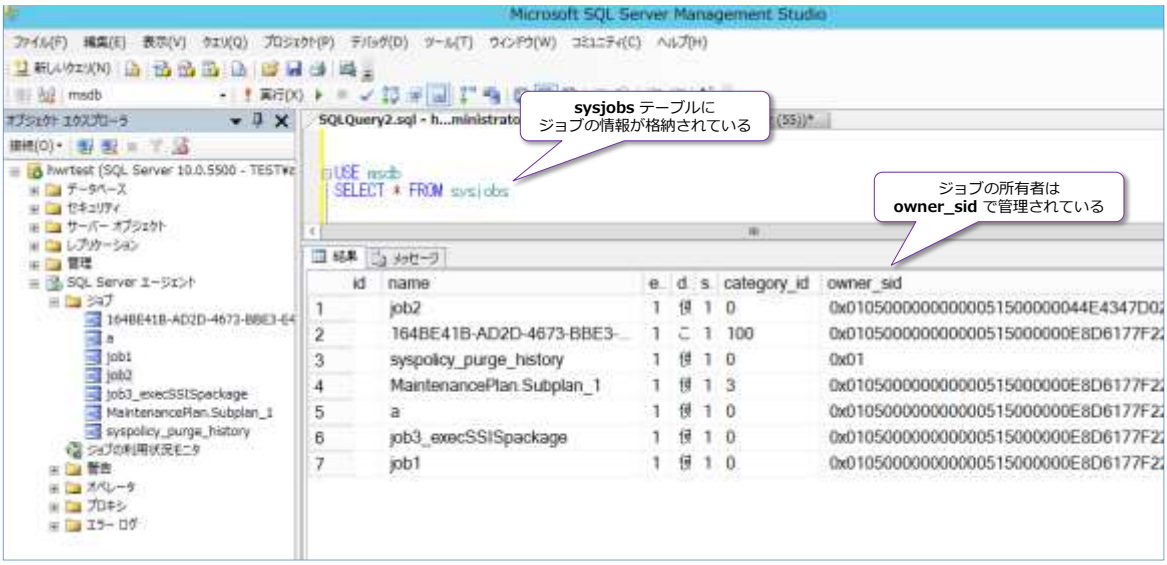
## 4.6 ジョブの所有者が Windows のローカル ユーザーの場合

旧マスターでの**ジョブの所有者**が **Windows のローカル ユーザー**（**マシン名¥ユーザー名** 形式のユーザー）の場合には、ジョブの実行エラーとなります（スケジュール設定しているジョブは、そのタイミングでエラーになります）。これは、次のような状況です（画面は SQL Server 2008）。



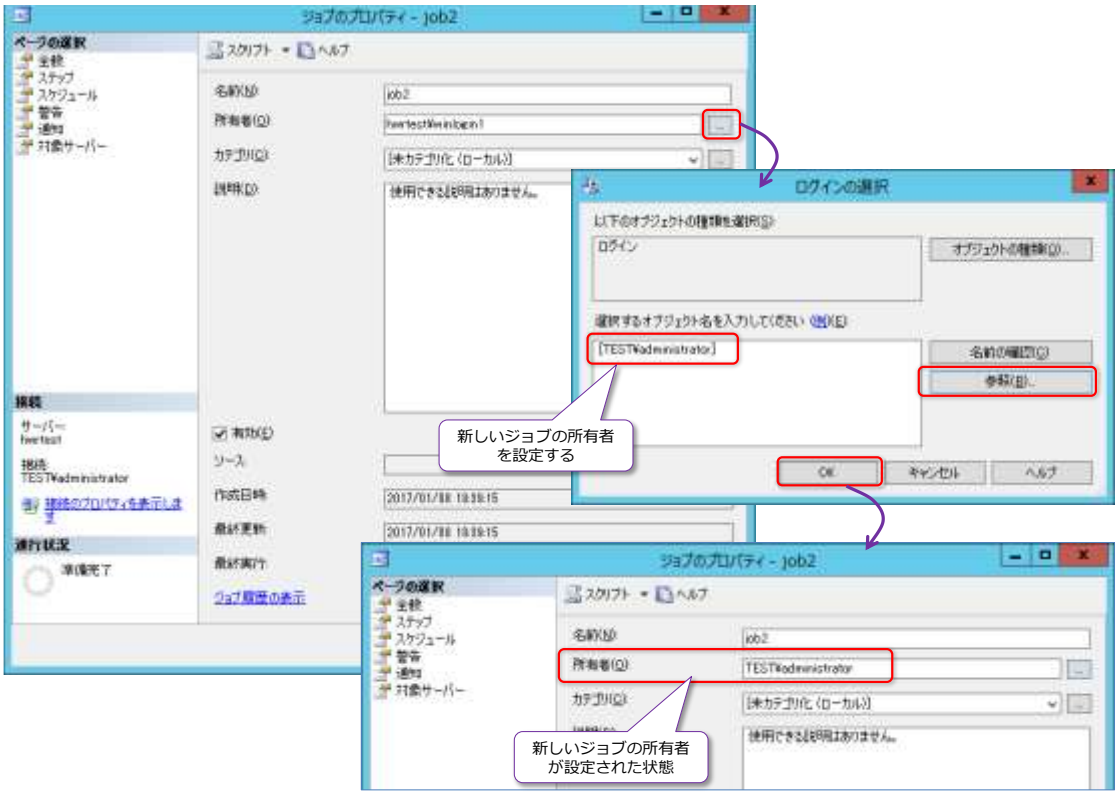

ジョブの実行履歴に記録されるエラーには、「**所有者（～）にサーバーのアクセス権があるかどうかを確認できません**」となります。これは、ジョブの所有者が、次のように **owner_sid**（所有者の Security ID）で管理されているために発生しています。



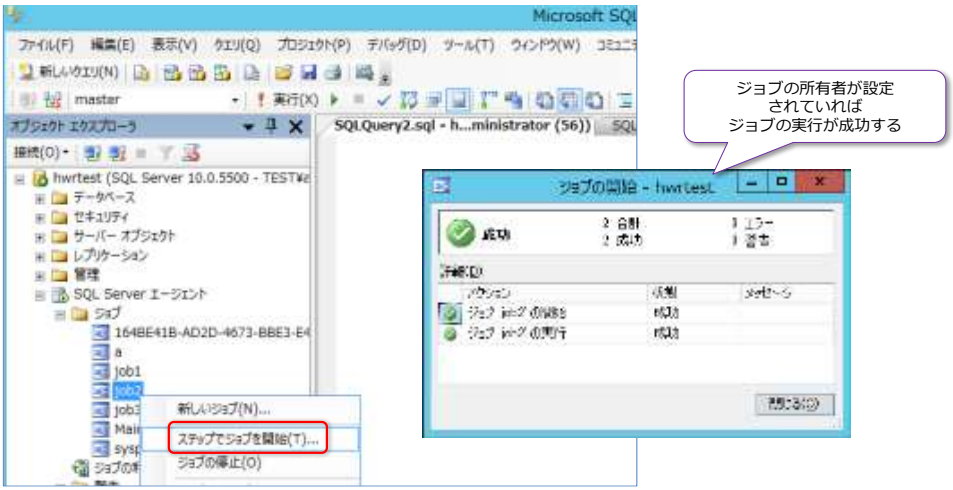

前述したように、Windows のローカル ユーザーは、マシン（OS）が変わったとすると、同じ名前のユーザーを作成したとしても、**内部的な SID（Security ID）**が異なってしまうので、こういったことが起こります。

## ジョブの所有者を設定すれば解決

ジョブを正しく動作させるようにするには、次のように新しい所有者を設定します。




このように、新しい所有者が設定されていれば、ジョブが正しく実行できるようになります。



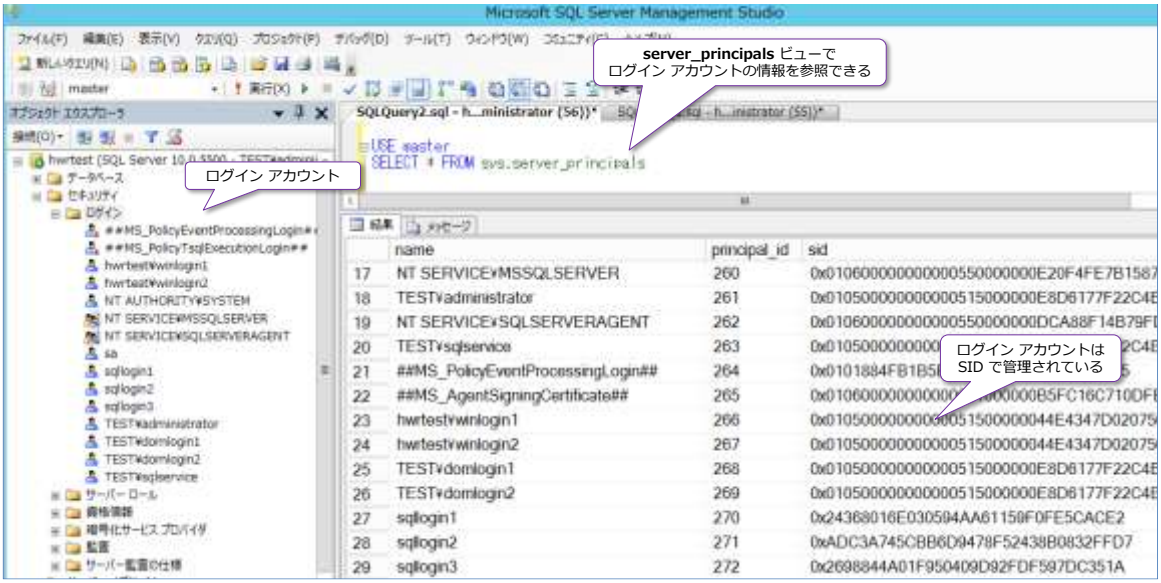

ジョブは、さまざまな機能でスケジューリング（定期実行）するために利用されているので、うまく動作しないジョブは、所有者を確認してみてください。

データベースの所有者のところで説明したのと同様、ジョブの所有者が Windows のローカル ユーザーとなってしまうのは、Management Studio を利用するときに、Windows のローカル ユーザーでログインして操作している場合です。Windows のローカル ユーザーでログインした状態でジョブを作成すると、ジョブの所有者がそのユーザーになってしまいます。

したがって、こうしたハードウェア リプレースなどでのデータベース移動を考慮すると、SQL Server を操作するときには（Management Studio を利用するときには）、**Active Directory のドメイン ユーザー**を利用するようにすることをお勧めします。

## 4.7 ログイン アカウントが Windows のローカル ユーザーの場合

旧マスターで、**ログイン アカウント**に、**Windows のローカル ユーザー**（マシン名¥ユーザー名）を利用している場合は、そのログイン アカウントが正しく動作しません。理由は、ジョブの場合と同様で、ログイン アカウントは、Windows ローカル ユーザーの**内部的な SID**（**Security ID**）で管理されているためです。



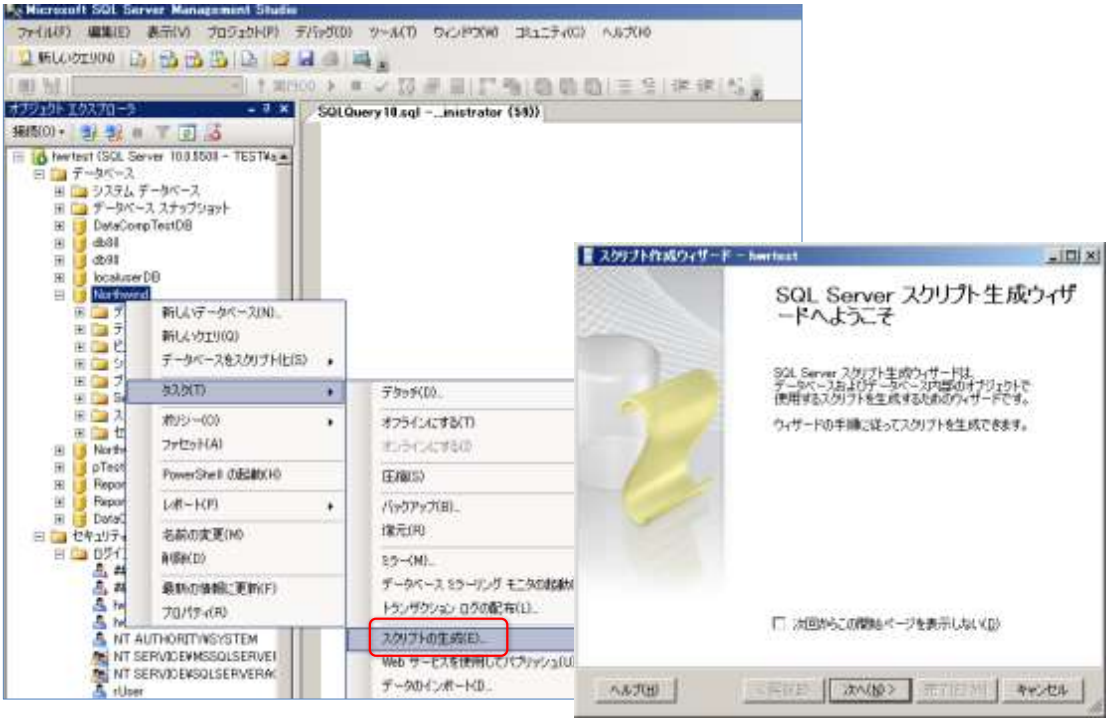

ただし、動作しないのは、あくまでも **Windows のローカル ユーザー**に対応したログイン アカウントだけであって、**Active Directory ドメインのユーザー**に対応したログイン アカウントや **SQL Server 認証用のログイン アカウント**（sa など）に関しては、問題なく動作します。Active Directory ドメインのユーザーであれば、マシン（OS）が変わっても、同じ SID（ドメインのユーザーに割り当てられた SID）を利用できるからです。また、SQL Server 認証用のログイン アカウントに関しては、master データベース内に、そのアカウントの SID を格納しているので、master を復元することで、こちらも問題なく利用できます。

**Windows のローカル ユーザー**に対応したログイン アカウントが利用できない場合は、そのログイン アカウントに紐付いた**データベース ユーザー**も利用できなくなり、データベース ユーザーに紐付いた**オブジェクト権限**（テーブルやビューに対する SELECT 権限や INSERT 権限、ストアド プロシージャに対する EXECUTE 権限など）も無力になってしまいます。これを修復する方法は用意されていないので、これを解決するには、ログイン アカウントとデータベース ユーザー、オブジェクト権限をすべて再作成／再設定しなければなりません（後述のスクリプト生成ウィザードを利用すれば、簡単に再作成／再設定できます）。

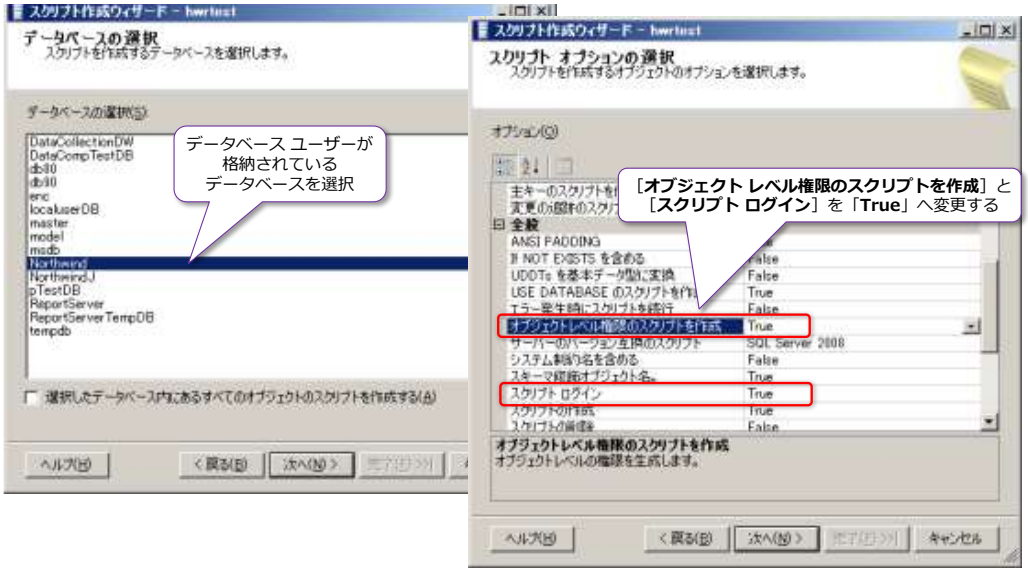
繰り返しになりますが、これは Active Directory ドメインのユーザーを利用している場合には、問題にはならず、あくまでも Windows のローカル ユーザーを利用している場合のみの話しになります（Active Directory ドメインのユーザーであれば再作成は必要ありません）。

## Windows のローカル ユーザーに関する設定を再作成／再設定する便利な方法

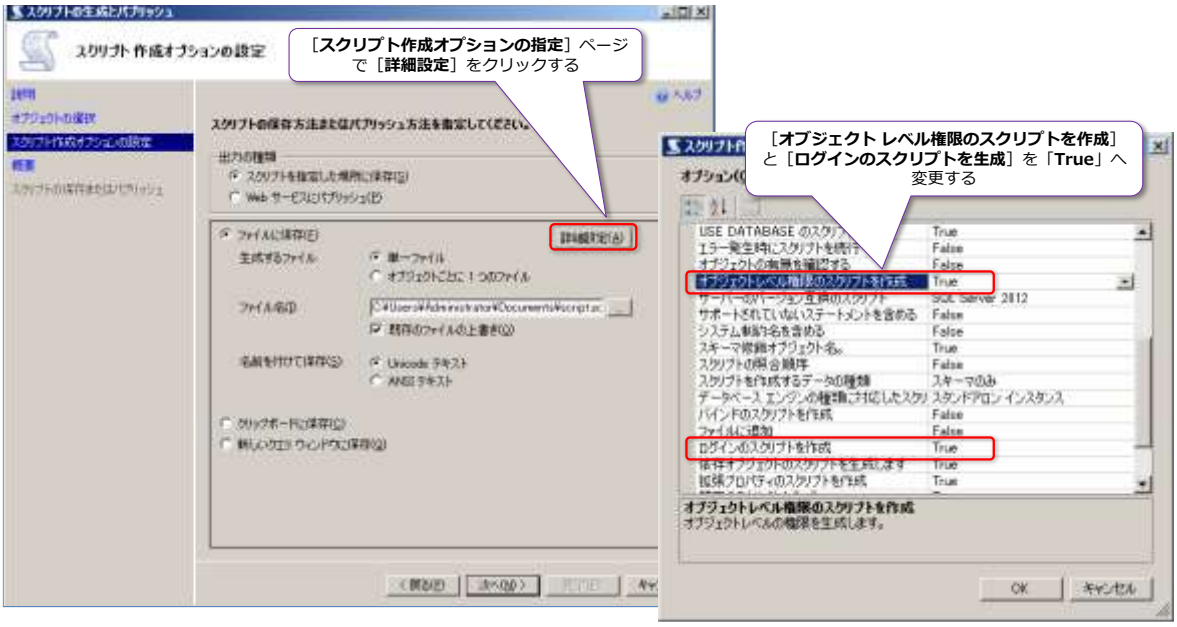
Windows のローカル ユーザーに関する設定（ログイン アカウントやデータベース ユーザー、オブジェクト権限）を再作成／再設定するには、事前に旧マスター側で、**スクリプト生成ウィザード**を利用して、スクリプト化をしておくと便利です。これを利用するに次のようにデータベースを右クリックして、[**タスク**] メニューの [**スクリプトの生成**] をクリックします（以降の画面は、SQL Server 2008 の場合ですが、他のバージョンでも同じように操作できます）。



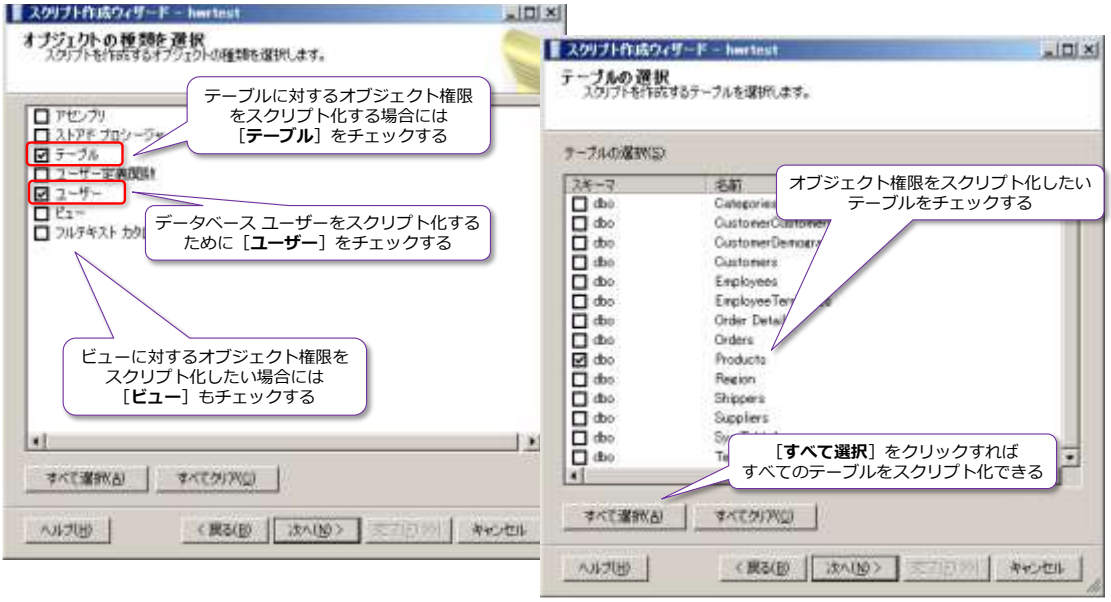

スクリプト生成ウィザードでは、次のように [**スクリプト オプションの選択**] ページで、[**オブジェクト レベル権限のスクリプトを作成**] と [**スクリプト ログイン**] を「**True**」へ変更することで、データベース ユーザーとオブジェクト権限、データベース ユーザーに対応したログイン アカウントをスクリプト化することができます（SQL Server 2008 の場合）。



SQL Server 2008 R2／2012／2014 の場合は、次のように操作します。



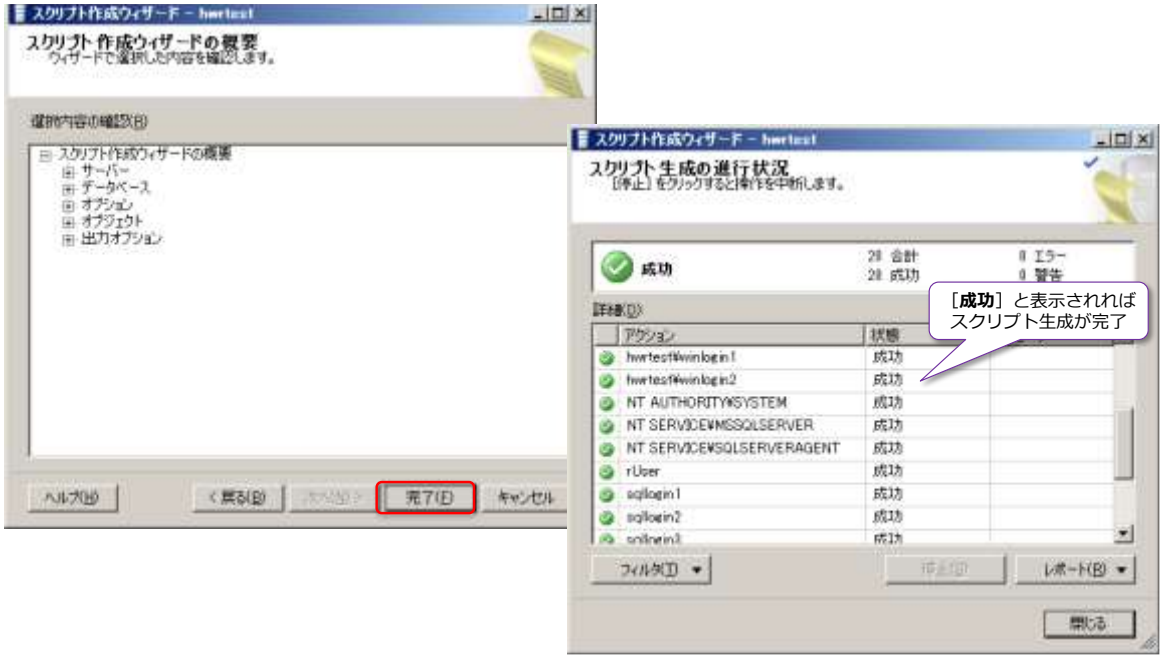

次の［**オブジェクトの種類を選択**］ページでは、データベース ユーザーをスクリプト化するために［**ユーザー**］をチェックし、テーブルに対するオブジェクト権限をスクリプト化する場合には［**テーブル**］をチェックします。ビューやストアド プロシージャに対するオブジェクト権限をスクリプト化するには、それらもチェックしておきます。

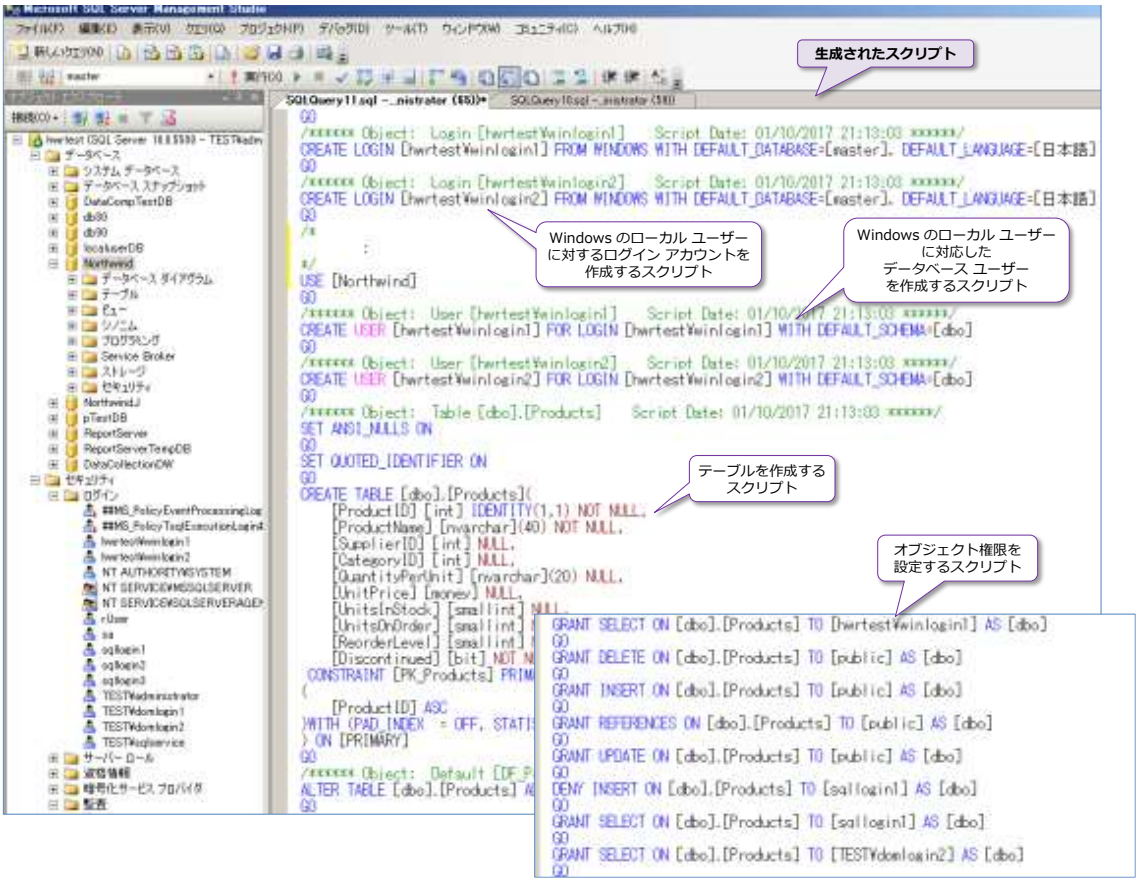



［**テーブルの選択**］ページでは、オブジェクト権限をスクリプト化したいテーブルをチェックします（画面は **Products** テーブルをチェック。［**すべて選択**］ボタンをクリックした場合は、すべてのテーブルを選択することができます）。これらの画面は、SQL Server 2008 の場合ですが、SQL Server 2008 R2／2012／2014 の場合は、ウィザードの最初のページで、ユーザーやテーブルを選択することができます。

SQL Server 2008 の場合は、次の［**ユーザーの選択**］ページで、スクリプト化したいデータベース ユーザーを選択します。

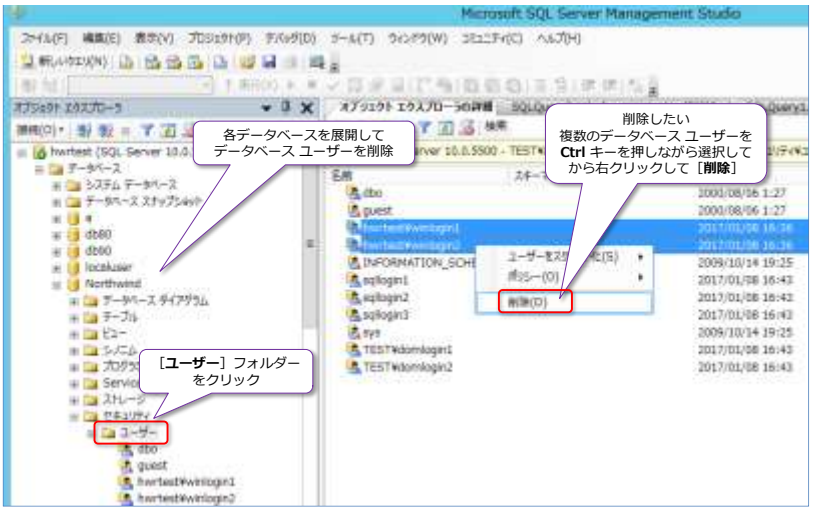

［**出力オプション**］ページで、［**スクリプトを新しいクエリ ウィンドウに保存**］を選択すれば、クエリ エディターにスクリプトを生成することができます。あとは、次のように［**完了**］ボタンをクリックすれば、スクリプト生成が開始されます。

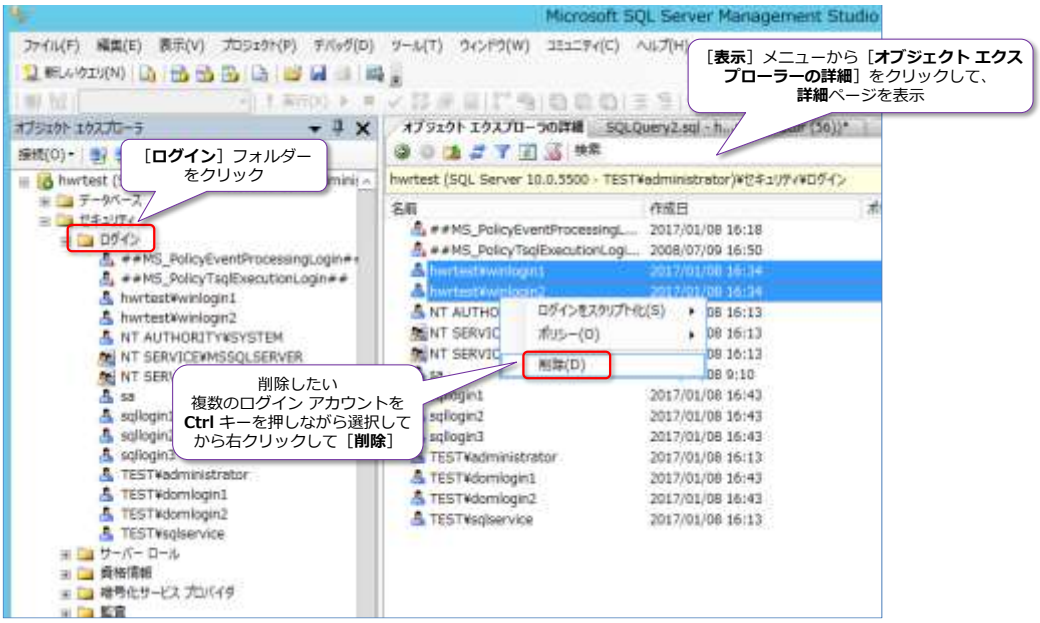





テーブルを作成するスクリプト（**CREATE TABLE**）や制約を作成するスクリプト（**ALTER TABLE .. ADD CONSTRAINT**）などが余分に出力されてしまいますが、ログイン アカウントを作成するスクリプト（**CREATE LOGIN**）や、データベース ユーザーを作成するスクリプト（**CREATE USER**）、オブジェクト権限を設定するスクリプト（**GRANT**、**DENY** など）が生成されていることを確認できると思います。

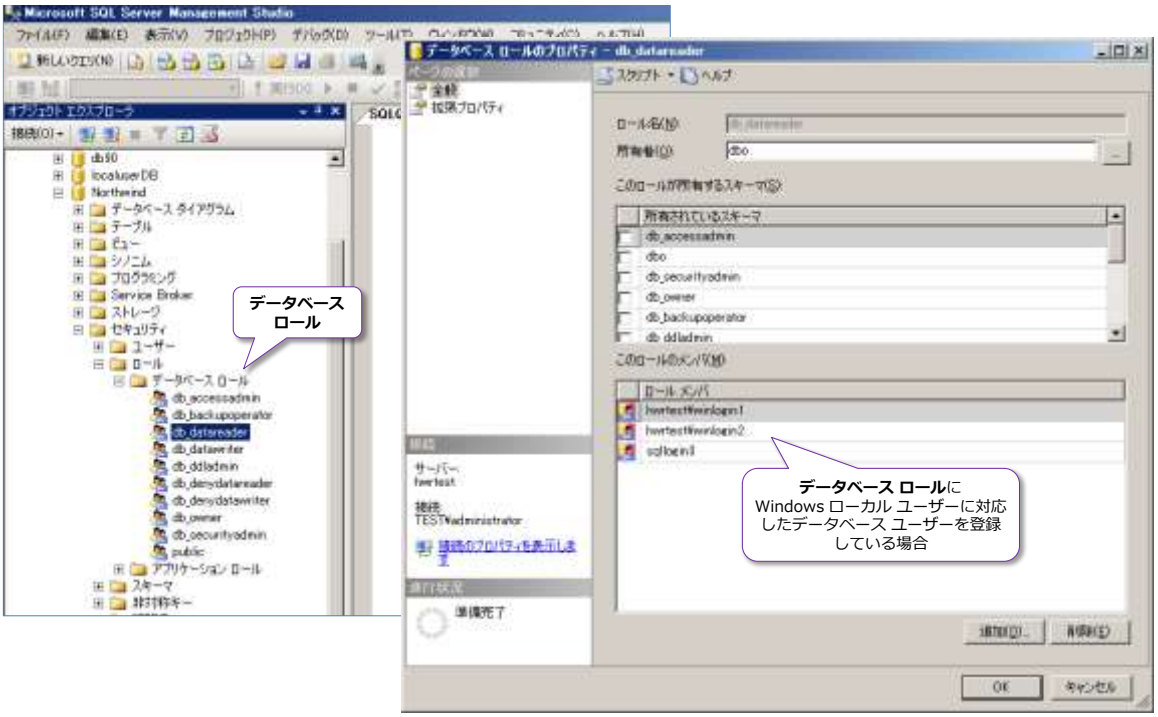
あとは、このスクリプトの **CREATE TABLE**／**ALTER TABLE** の部分を削除して、新マスターで実行するようにしますが、新マスター側では、オフライン バックアップによって復元されてしまった、利用不能な Windows のローカル ユーザーに対応したデータベース ユーザーおよびログイン アカウントを次のように削除しておきます。





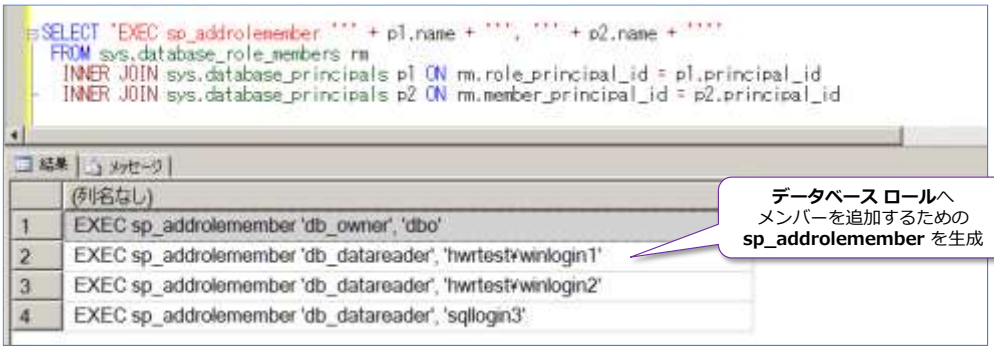

## データベース ロールのスクリプト化

データベース ロールを利用して、データベース ユーザーをグループ化している場合には、データベース ロールの設定もスクリプト化しておく必要があります。



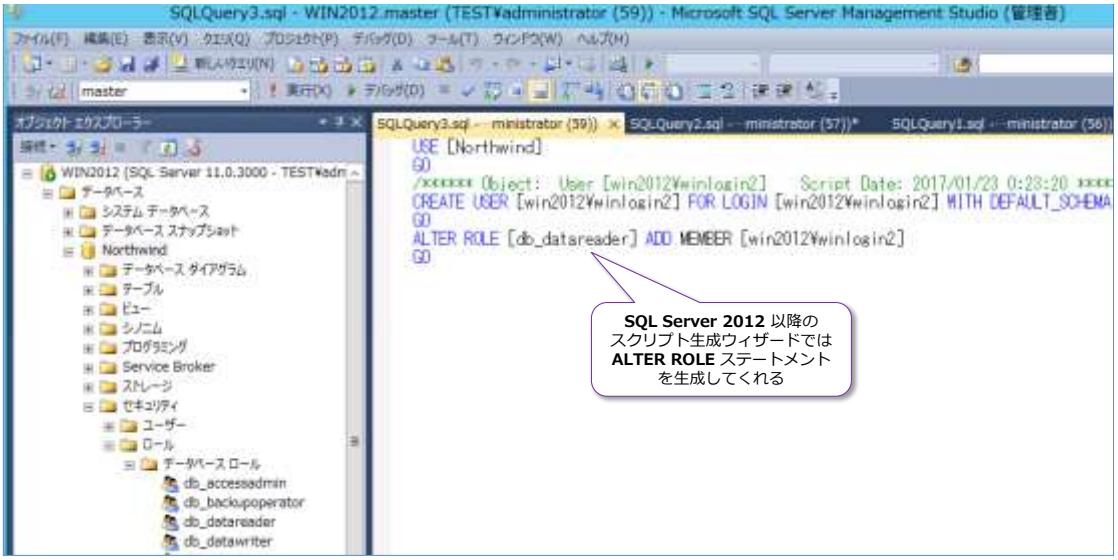

前述の**スクリプト生成ウィザード**では、SQL Server 2008／2008 R2 の場合は、データベース ロールの設定をスクリプト化することができないので、データベース ロールのメンバー情報を参照できる **database_role_members** システム ビューとデータベース ユーザーの情報を参照できる **database_principals** システム ビューを、次のように利用します。

```
SELECT 'EXEC sp_addrolemember ''' + p1.name + ''', ''' + p2.name + ''''
 FROM sys.database_role_members rm
  INNER JOIN sys.database_principals p1 ON rm.role_principal_id = p1.principal_id
  INNER JOIN sys.database_principals p2 ON rm.member_principal_id = p2.principal_id
```




これで、データベース ロールへメンバーを追加することできる **sp_addrolemember** を生成することができるので、これを新規マスター側で実行すれば、同じ設定にすることができます。

SQL Server 2012 以降を利用している場合は、スクリプト生成ウィザードが **ALTER ROLE** ステートメントを生成してくれるので、これを新規マスター側で実行すれば、データベース ロールの設定を同じように利用することができます。

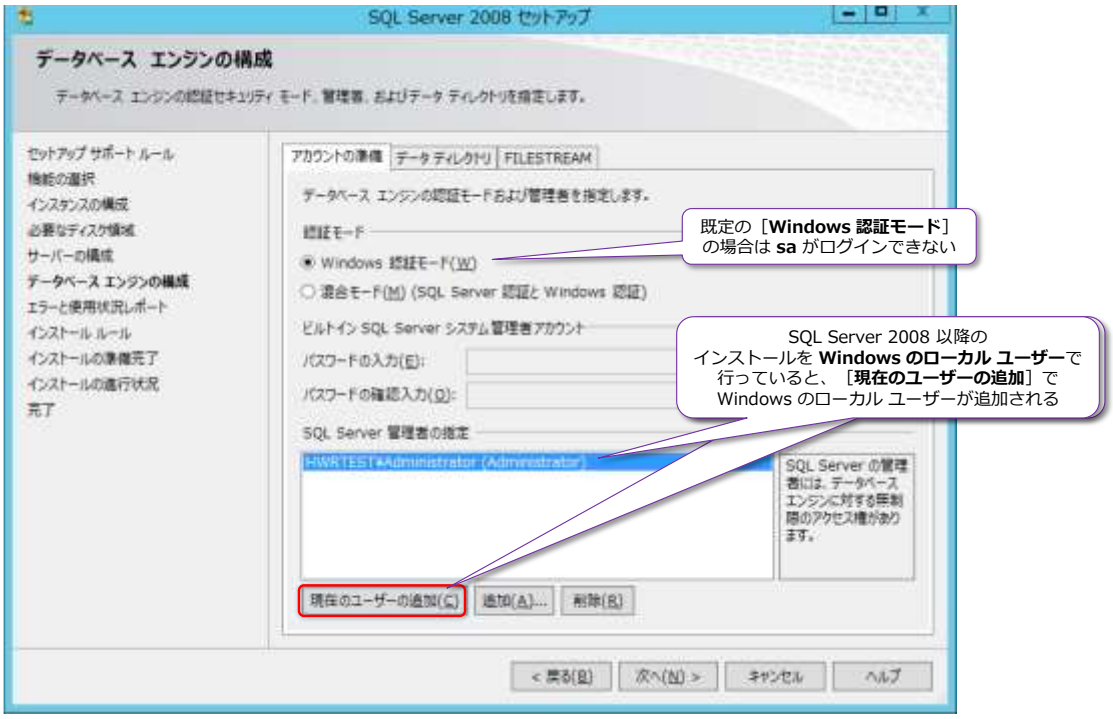


## 4.8 管理者アカウントが Windows ローカル ユーザーの場合

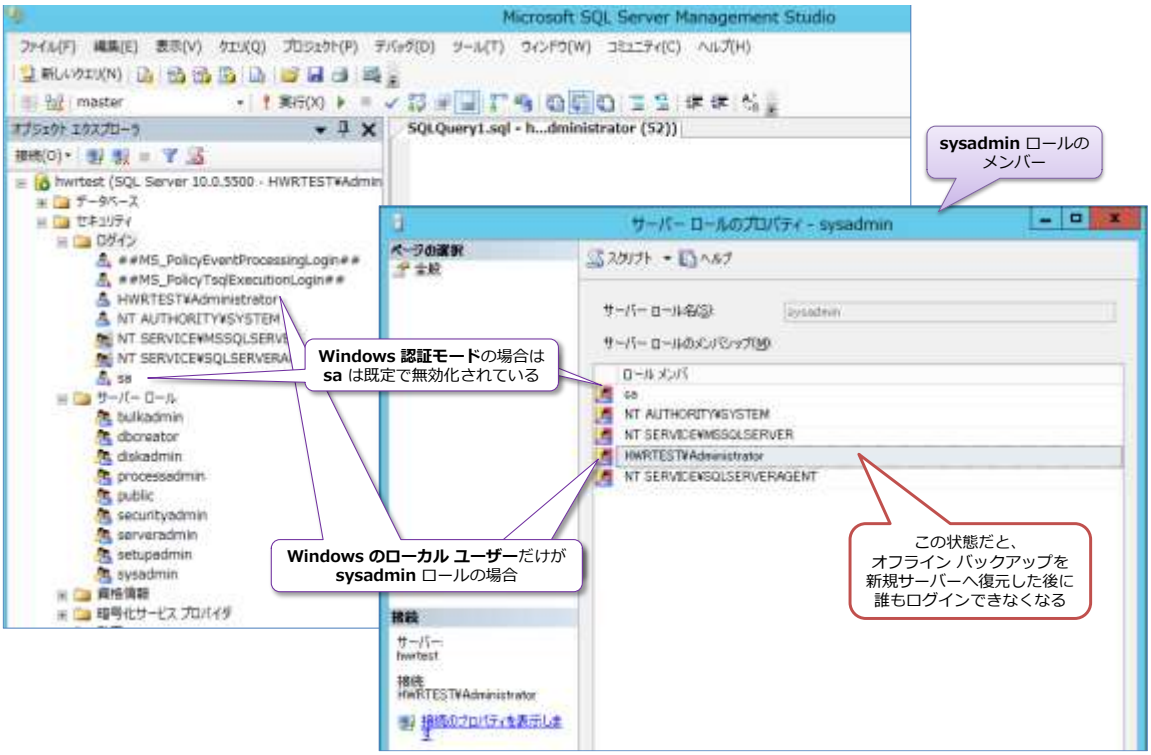
旧マスターでの**管理者アカウント**（**sysadmin** ロール）が、**Windows のローカル ユーザー**（**マシン名¥ユーザー名** 形式のユーザー）のみで、認証モードが「**Windows 認証**」の場合には、SQL Server へログインできるユーザーがいなくなってしまうことに注意する必要があります。理由は前項とまったく同様です。

これは、**Active Directory ドメイン ユーザー**が管理者アカウントであれば、全く問題ではなく、あくまでも Windows のローカル ユーザーの場合のみの話しになります。また、認証モードが「**混合モード**」で、「**sa**」アカウントが有効になっていれば、**sa** でログインすることができるので、この場合も大丈夫です。なお、**SQL Server 2005** のときには、既定で **Administrators** グループが **sysadmin** ロールとして登録されていたので、この場合は、**Administrators** グループへ Active Directory ドメインのユーザーを追加すれば、ログインすることができました。

SQL Server 2008 以降を利用している場合は、次のように **SQL Server のインストール**を **Windows のローカル ユーザー**（**マシン名¥Administrator** など）で行っている場合に注意が必要です。




Windows のローカル ユーザーでインストールを行っていると、［**現在のユーザーの追加**］ボタンをクリックした場合に、Windows のローカル ユーザーが管理者アカウントとして登録されて、ここで認証モードを「**Windows 認証モード**」へ設定していると、旧マスターを新マスターへ入れ替えた場合（オフライン バックアップを復元した後に）、誰もログインできない状態となってしまいます。




これを避けるには、事前に、旧マスター環境で、**Active Directory のドメイン ユーザー**を管理者アカウント（**sysadmin**）として追加しておくようにします。

## 4.9 ケース２ の残りの作業（ケース１ と同じ）

「**ケース２ 新規サーバー（別マシン）へのアップグレード**」での完全複製（旧マスターを新マスターに丸ごと複製）が完了した後は、**SQL Server 2016 へのアップグレード**を行うことになりますが、作業手順は次のとおりです（以降の手順は、ケース１ と同じです）。

このケースでのアップグレード手順は、次のとおりです。

1. **Data Migration Assistant** による事前チェックを行う
2. **SQL Server 2016** への**アップグレード要件**を確認する（アップグレード可能な Service Pack を確認／インストールする）

   SQL Server 2008 なら SP4、SQL Server 2008 R2 なら SP3、SQL Server 2012 なら SP2 が必要。OS に Windows Server 2003 や 2003 R2、2008、2008 R2 を利用している場合は、Windows Server 2012 以上にアップグレードする。
3. **新規サーバー**を **SQL Server 2016 へアップグレード**する
4. **SQL Server 2016** の**最新の修正プログラム**（CU や Service Pack など）をインストールする

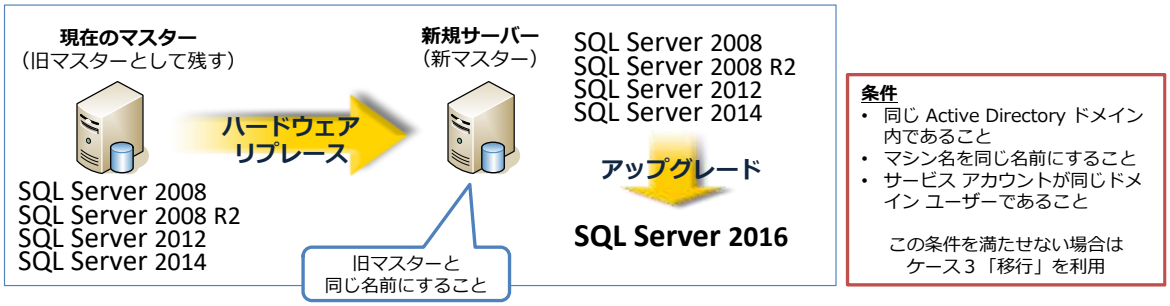
   Reporting Services を利用している場合は、SP1 に対する重要な更新プログラムがあるので、SP1＋CU1 または CU4 以降を適用しておく。
5. **Management Studio**（管理ツール）の最新版をダウンロードして、インストールする（オプション）
6. **統計**（Statistics）を更新する
7. **データベースの互換性レベルを 130** へ上げる（オプション）

   互換性レベルの影響については**クエリ ストア**機能を利用することで簡単にチェック可能（実行プランの比較や、プラン強制もできる）。
8. **BIDS**（Business Intelligence Development Studio）や **SSDT-BI**（SQL Server Data Tools - Business Intelligence）を利用している場合は、**SSDT** の最新版をダウンロードして、インストールする（オプション）

以上で、ハードウェア リプレース時を伴った場合の SQL Server 2016 へのアップグレード作業が完了です。システム データベースのオフライン バックアップを利用すれば、非常に簡単にハードウェア リプレース／同じ名前の新規サーバーの構築および SQL Server 2016 へのアップグレードを行うことができます。

## 4.10 ケース 2「新規サーバーへのアップグレード」のまとめ

ケース 2 での操作手順をまとめると、次のようになります。




1. **現在のマスター環境**を**丸ごと新規サーバーへ複製**する

   - 現在のマスターで**オフライン バックアップ**（全データベース）を取得する
     （ユーザー データベースに関しては、オンライン バックアップでも可）
   - 現在のマスターを停止して、**ネットワークから切り離す**（旧マスターとなる）
   - **新規サーバー**に **OS** をインストールし、マシン名を**旧マスターと同じ名前**へ変更する
     （インストールする **OS** は、旧マスターと異なるものでも可
     異なる OS を利用する場合は、SQL Server を動作させるための SP 要件も確認必須）
   - **新規サーバー**を **Active Directory ドメインへ参加**させる
   - 新規サーバーへ旧マスターと**同じバージョンの SQL Server** をインストールする
   - 旧マスターの SQL Server にインストール済みの**修正プログラム**を、新規サーバーにもインストールする
   - 新規サーバーの **SQL Server を停止**する
   - 新規サーバーで、念のため、システム データベースの**オフライン バックアップ**を取得しておく（万が一の上書き失敗時に、元に戻せるようにするため）
   - 旧マスターで取得した**オフライン バックアップを上書きコピー**する（復元する）
     ユーザー データベースをオンライン バックアップで取得している場合は、SQL Server の起動後に、該当データベースを復元する
   - 新規サーバーの **SQL Server を起動**する
   - **レジストリ**に格納されている情報を再設定する（TCP ポート番号や起動時パラメーターでのトレースフラグ設定などのうち、旧マスターで設定を変更しているものがある場合はそれらを再設定する）
   - **OS の設定**で、旧マスターで変更しているものがある場合は、それらを再設定する
     （フォルダー構成や、NTFS アクセス許可、ユーザーの権利、共有フォルダーなど）

   以上で旧マスターとまったく同じ環境を丸ごと新規サーバー上で動作させることができます。

2. **Data Migration Assistant** による事前チェックを行う

3. **SQL Server 2016** への**アップグレード要件**を確認する（アップグレード可能な Service Pack を確認／インストールする）

   SQL Server 2008 なら SP4、SQL Server 2008 R2 なら SP3、SQL Server 2012 なら

SP2 が必要。OS に Windows Server 2003 や 2003 R2、2008、2008 R2 を利用している場合は、Windows Server 2012 以上にアップグレードする。

4. **新規サーバー**を **SQL Server 2016 へアップグレード**する
5. **SQL Server 2016** の**最新の修正プログラム**（CU や Service Pack など）をインストールする

   CU2 には、性能向上に関する修正が入っているので、できる限り最新の修正プログラムを適用しておくことをお勧めします。

   Reporting Services を利用している場合は、SP1 に対する重要な更新プログラムがあるので、CU4（SP1＋CU1）以上を適用しておくことをお勧めします。
6. **Management Studio**（管理ツール）の最新版をダウンロードして、インストールする（オプション）
7. **統計**（Statistics）を更新する
8. **データベースの互換性レベルを 130** へ上げる（オプション）

   互換性レベルの影響については**クエリ ストア**機能を利用することで簡単にチェック可能（実行プランの比較や、プラン強制もできる）。
9. **BIDS**（Business Intelligence Development Studio）や **SSDT-BI**（SQL Server Data Tools - Business Intelligence）を利用している場合は、**SSDT** の最新版をダウンロードして、インストールする（オプション）

作業のポイントは、ハードウェア リプレース時の新規サーバーへの完全複製です。旧マスターと**同じ名前のサーバー**を作成すること、**同じ Active Directory ドメイン**に参加していること、旧マスターの SQL Server と**同じパスにインストール**すること、**同じ名前のインスタンス名**にすること、**同じサービス アカウント**を利用すること（サービス アカウントは**ドメイン ユーザー**であること）、**同じ種類の修正プログラム**を適用すること、**同じフォルダー構成／NTFS アクセス許可**にすることなどを守っていれば、問題が起こることはありません（弊社のお客様では、問題なく動作しています）。

唯一の問題は、**Windows のローカル ユーザー**を利用している場合ですが、この場合はいくつかのものが動作しなくなりますが、データベースとジョブについては、所有者を設定することで正常に動作するようになります。ログイン アカウントについては、スクリプト生成ウィザードでスクリプト化を行っておくことで再設定を容易に行えます。

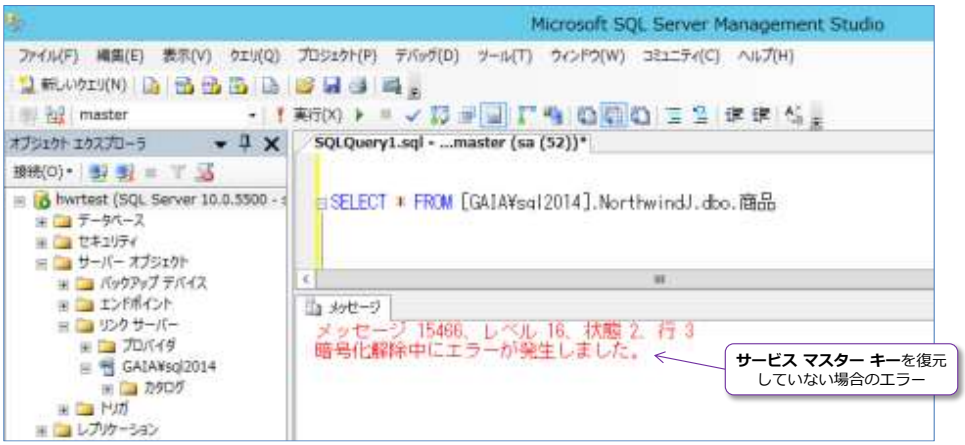
Windows のローカル ユーザーの問題は、（旧マスターで）**Management Studio** を Windows のローカル ユーザーで操作していることに起因するので、今後のことを考慮すると、Management Studio を利用するときは、**Active Directory のドメイン ユーザー**で操作することを強くお勧めします。

**Note： サービス アカウントが同じドメイン ユーザーではない場合**

このケースでは、SQL Server サービスの**サービス アカウント**が、旧マスターと新規サーバーで**同一のドメイン ユーザー**であることが条件でしたが、これは各種の暗号化に関わる**サービス マスター キー**が関係しています。サービス マスター キーは、SQL Server のセットアップ時にサービス アカウントに紐付いて作成されるので、旧マスターと新規サーバーで同じサービス アカウントを利用することで、共通のサービス マスター キーを利用できるようになります。

サービス マスター キーは、**リンク サーバー**のセキュリティ設定（リモート ログインのパスワードなど）や、**データベース メール**でのメール サーバーへのログイン パスワード、**TDE**（透過的なデータ暗号化）で利用するデータベース マスター キーなどで利用され、（内部的なデータを含めた）**各種のデータを暗号化**するための最も重要なキーになります。

このため、サービス アカウントが同じドメイン ユーザーではない場合は（Windows のローカル ユーザーなどをサービス アカウントにしている場合は）、リンク サーバーや、データベース メール、TDE などが、新規サーバー側で動作しないということになります。サービス マスター キーが原因で、うまく動作しない場合には、次のように「**暗号化解除中にエラーが発生しました**」エラーが発生します。




これを回避するには、サービス マスター キーを作成し直します（**REGENERATE** します）。作成し直しは、次のように **ALTER SERVICE MASTER KEY** ステートメントを実行します。

```
USE master
ALTER SERVICE MASTER KEY REGENERATE
```

このステートメントを実行しても、「**暗号化解除中にエラーが発生しました**」エラーが発生する場合には、次のように **FORCE**（強制）オプションを付けて、作成し直します。

```
USE master
ALTER SERVICE MASTER KEY FORCE REGENERATE
```

ただし、このように、**FORCE** を利用した場合は、暗号設定が失われる可能性があるので、暗号化が関連する機能（リンク サーバーのセキュリティ設定やデータベース メールでのメールサーバーへのログイン情報など）の再設定が必要になる場合があります。TDE を利用している場合には、データベース マスター キーや証明書のバックアップが必要になる、あるいは再設定が必要になる場合があります。

したがって、サービス アカウントは、同一のドメイン ユーザーを利用するようにすることが重要になります。サービス アカウントが同一のドメイン ユーザーであれば、こういった問題は発生しません。もし、サービス アカウントに Windows のローカル ユーザーを利用している場合は、第 5 章で説明するリンク サーバーやデータベース メールのスクリプト化の手順を利用して、旧マスター側で設定をスクリプト化しておくことをお勧めします（これで再設定が容易になります）。

## 4.11 アップグレード時間（ダウンタイム）の見積もり

アップグレードにかかる時間（ダウンタイム）は、次のように考えることができます。

| | | 移行作業 | 作業時間の見積もり | 説明 |
|---|---|---|---|---|
| 1 | 1 | 現在のマスターでオフライン バックアップ（全データベース）を取得 | 大きく変動 | データベース サイズやストレージの性能、ネットワーク速度によって大きく変化する。ユーザー データベースに関しては、オンライン バックアップを利用することも検討する |
| | 2 | 現在のマスターを停止<br>（これにより、旧マスターになる） | 数分 | ネットワークから切り離して、新マスターとはネットワーク的に繋がらないようにする |
| | 3 | 新規サーバーへ OS をインストール<br>（SQL Server 2016 のインストールには<br>Windows Server 2012 以上が必要） | 20分～2時間 | OS として Windows Server 2008 R2 を利用する場合は **with SP1 のパッケージ**を利用することで、OS と SP1 のアップグレードを同時に行うことが可能。<br>作業時間は、ハードウェア環境によって作業時間が前後する。OS の再起動に 10分程度かかるサーバー機もあるため、機種によっては 1時間以上になることも。<br>SQL Server 2016 へのアップグレードには **Windows Server 2012** 以上が必要になるので、このタイミングで Windows Server 2012 以上をインストールしておくことも考慮 |
| | 4 | 新規サーバーの OS のマシン名を<br>旧マスターと同じ名前に変更する | 数分～15分 | マシン名の変更には、OS の再起動が伴い、OS の再起動にかかる時間は、サーバー機によって大きく異なる |
| | 4 | 新規サーバーを Active Directory ドメインに参加させる（旧マスターはドメインから削除する） | 数分～15分 | ドメインの参加には、OS の再起動が伴い、OS の再起動にかかる時間は、サーバー機によって大きく異なる |
| | 5 | 新規サーバーへ旧マスターと同じバージョンの SQL Server をインストールする | 15分～<br>1時間30分 | ハードウェア環境や、インストールしている機能の種類によって作業時間が前後する。 |
| | 6 | 旧マスターへインストール済みの SQL Server の修正プログラムと同じものを、新規サーバーへもインストールする | 15分～<br>1時間30分 | ハードウェア環境や、インストールしている機能の種類によって作業時間が前後する。 |
| | 7 | 新規サーバーの SQL Server を停止して、旧マスターで取得したオフライン バックアップを上書きコピー | 大きく変動 | データベース サイズやストレージの性能、ネットワーク速度によって大きく変化する |
| | 8 | 新規サーバーの SQL Server を起動 | 数分～ | もし、上の手順で tempdb データベースを上書きしていない場合で、tempdb のサイズが大きい場合には、その分起動後のデータベースの復旧に時間がかかることに注意する（この場合に、瞬時初期化が有効でない場合はさらに時間がかかるので注意する） |
| | 9 | レジストリの設定を旧マスターと同様にする | 数分 | レジストリに格納されている情報はほとんどないため、作業はすぐに完了する |
| | 10 | OS の設定を旧マスターと同様にする | 数分～数十分 | 作成するフォルダー数や、共有フォルダー、NTFS アクセス許可、ユーザーの権利（瞬時初期化やメモリ内ページ ロックの権利）など、状況によって作業時間が左右する |
| 2 | | Data Migration Assistant による互換性のチェックを行う | - | 事前作業。<br>データベースの互換性を事前にチェックしておく |
| 3 | | SQL Server 2016 のアップグレードに必要な SP がインストールされていない場合は、SP をインストールする。OS の要件も確認／アップグレードする | 10分～<br>1時間 | SQL Server 2008 は **SP4**、SQL Server 2008 R2 は **SP3**<br>SQL Server 2012 は **SP2** が必須条件。2014 は SP なしで OK。<br>OS は Windows Server 2012 以上が必要。OS のアップグレードが必要な場合は 20分～1時間半ぐらいの追加時間が発生 |
| 4 | | SQL Server 2016 へのアップグレード インストールを行う | 15分～<br>1時間30分 | ハードウェア環境や、以前のバージョンでインストールされていた機能の種類によって作業時間が前後する。<br>フルテキスト検索機能を利用している場合は、「再構築」を選択した場合に、フルテキスト インデックス作成時と同等の時間がかかることに注意する |
| 5 | | SQL Server 2016 の最新の修正プログラム（CU や SP）をインストールする | 15分～<br>1時間30分 | ハードウェア環境や、インストールしている機能の種類によって作業時間が前後する。<br>SP1 以降を適用しておくことをお勧めします。<br>Reporting Services を利用する場合は、SP1 に対する重要な更新プログラム（KB3207512）もインストールしておく |
| 6 | | Management Studio の最新版をインストールする（オプション） | 10分～1時間 | 事前にインストーラーをダウンロードしておくことで<br>インストール時間を短縮できる |
| 7 | | 統計（Statistics）を更新する | 数分～ | テーブル サイズが大きい場合は実行に時間がかかることがある |
| 8 | | データベースの互換性レベルを 130 へ上げる（オプション） | 数分 | 必要に応じて、クエリストア機能を利用して、<br>互換性レベルの影響（特に性能面）をチェックする |
| 9 | | 動作チェックを行う | 1時間～ | アプリケーションやストアド プロシージャが問題なく動作するかをチェックする。当日のサービス インまでに確認しておくべき最重要クエリの動作チェックを行う |
| 10 | | SSDT のインストール（オプション） | 10分～1時間 | 事前にダウンロード版のインストーラー（iso）をダウンロードしておくことで、インストール時間を短縮できる |
| 99 | | OS を Windows Server 2016 へアップグレードする（オプション） | 20分～2時間 | : |
| 99 | | 万が一の失敗時のロールバック作業 | ～30分 | 旧マスター環境が残っているので、新マスターをネットワークから切り離して、旧マスターを Active Directory ドメインへ参加し直せば、元の状態へ戻すことができる |

この手順の場合は、マシン名の変更や Active Directory ドメインの参加、旧バージョンの SQL

Server＋修正プログラムのインストールなどで、OS の再起動が必要になるので、その分の時間を加味しておく必要があります（∵最近のサーバー機は、OS の再起動に 5～10 分程度かかるものが多いため）。

この手順の中で、作業時間（ダウンタイム）を大きく左右するのは、**全データベースのオフライン バックアップ**と、その**バックアップ ファイルのコピー**にかかる時間、これを**新規サーバーへ上書きコピー**するのにかかる時間です。これは、**データベース サイズ**が大きい場合には、非常に時間がかかることになります。また、**ストレージの性能**（読み取りおよび書き込み速度）と**ネットワーク速度**にも大きな影響を受けます。

## バックアップ時間の見積もり

**オフライン バックアップ**の場合は、SQL Server を停止することで、ファイル（.mdf／.ldf）を直接操作できるようにするものなので、バックアップ時間は、次のとおりです。

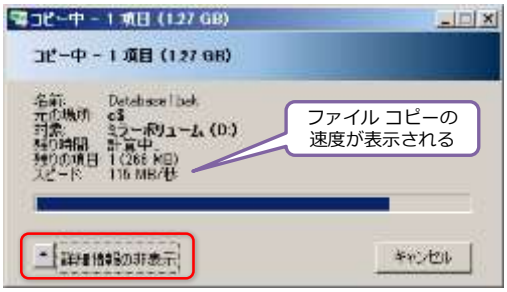
**オフライン バックアップにかかる時間**
**＝ ファイル（.mdf／.ldf）をコピーするのにかかる時間**

オンライン バックアップの場合は、オンライン バックアップ（BACKUP ステートメントの実行）にかかる時間とファイルのコピーにかかる時間が、バックアップにかかる時間になりますが、オフライン バックアップの場合は、ファイルをコピーする時間のみがバックアップにかかる時間になります。

## ファイルのコピーにかかる時間の見積もり

ファイルのコピーにかかる時間は、Windows Server 2008 以降の OS では、次のように確認することができます。

Windows Server 2008／2008 R2 の場合




Windows Server 2012／2012 R2 の場合


ファイルのコピー中に、[**詳細情報**] をクリックすると、このように **1 秒あたり何 MB** が転送されているか（**MB/sec**）を確認することができます。したがって、事前に数 GB～数十 GB レベルの実際のデータベースよりも小さいファイルをコピーして、転送時間を確認しておくことが重要です。この転送時間をもとに、オフライン バックアップにかかる時間を（かなり正確に）見積もることができます（コピーをネットワーク経由で行う場合は、移行当日のネットワーク利用状況を加味する

必要もあるので、本番を想定したネットワーク利用状況で、転送時間をチェックしておくことをお勧めします)。

見積もりの考え方としては、オフライン バックアップのファイル サイズ（コピー対象のファイルの総サイズ）が **100GB** で、転送速度が **100MB/sec** の場合には、**約 1,000 秒**（16 分 40 秒）の時間がかかり、転送速度が半分の **50MB/sec** だったとすると、転送時間は 2 倍の **2,000 秒**（33 分 20 秒）かかってしまう、転送速度が 2 倍の **200MB/sec** の場合は、転送時間は半分の **500 秒**（8 分 20 秒）で済む、といった具合です。概算は、次のように計算することができます。

| | | **サイズ** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **1GB** | **10GB** | **50GB** | **100GB** | **200GB** | **400GB** |
| **転送速度 MB/sec** | **1** | 1,000 (16:40) | 10,000 (2:46:40) | 50,000 (13:53:20) | 10,000 (27:46:40) | 10,000 (55:33:20) | 10,000 (111:6:40) |
| | **20** | 50 | 500 (8:20) | 2,500 (41:40) | 5,000 (1:23:20) | 10,000 (2:46:40) | 20,000 (5:33:20) |
| | **50** | 20 | 200 (3:20) | 1,000 (16:40) | 2,000 (33:20) | 4,000 (1:06:40) | 8,000 (2:13:20) |
| | **100** | 10 | 100 (1:40) | **500 (8:20)** | **1,000 (16:40)** | **2,000 (33:20)** | 4,000 (1:06:40) |
| | **200** | 5 | 50 | 250 (4:10) | 500 (8:20) | 1,000 (16:40) | 2,000 (33:20) |
| | **500** | 2 | 20 | 100 (1:40) | 200 (3:20) | 400 (6:40) | 800 (13:20) |

**1Gbps**（ギガビット）の **LAN** 環境で、標準以上のストレージを利用している場合は、80～110MB/sec ぐらいは転送速度が出るので、仮に **100MB/sec** と想定すると、**100GB** の転送には **1,000 秒**（**16 分 40 秒**）ぐらいだろうと推測できます。事前に、**10GB** 程度のサイズのファイルをコピーしてみて、これが 10 分の 1 の **100 秒**(1 分 40 秒) ぐらいで完了するのであれば、その見積もりはかなり正しいということになります。

このように、LAN 環境であれば、400GB のファイル サイズであっても、1 時間程度で完了することを考慮すると、オフライン バックアップでも (ネットワークが安定していれば) 問題ありません。ただし、ファイル コピーの場合は、もしコピーの途中でネットワークが切れた場合には、ファイルのコピーをイチからやり直さなければいけない可能性があるので、そのリスクも考慮しておく必要があります。

例えば、ネットワーク経由だけでなく、**USB HDD** にも二重でコピーしておいて、万が一ネットワークが切れた場合には、USB HDD からも復旧できるようにしておくなどです。ただ、USB HDD の場合には、まだまだ多くのサーバー機では、最近のデスクトップ／ノート PC での主流である **USB 3.0／3.1** ではなく、古い **USB 2.0** のみを搭載しているということがあります。**USB 3.0** であれば、接続する USB HDD のスピードにもよりますが **70MB/sec** 程度は出るところを、**USB 2.0** だと **20～25MB/sec** ぐらいしか出ません（USB 3.0 より 3 倍ぐらい遅いことが多い)。また、USB 2.0 の古い USB メモリを利用する場合には、**10MB/sec** ぐらいしか出ないものもあります。

したがって、USB HDD などに保存する場合には、ファイルを **gzip** や **7zip** などで**圧縮**してからコピーすることを検討するのをお勧めします。もちろん、圧縮には別途時間がかかりますが、コピー先の媒体が遅い場合には、圧縮をしてファイル サイズを小さくした方が全体として速く実行で

きる場合があります（事前に、圧縮にどれぐらいに時間がかかるのかを調査して、どちらが速く実行できるかを比較しておくことをお勧めします）。

上の転送速度の表には「**1MB/sec**」の場合も入れていますが、これは**データセンター間**をまたがったデータ転送（WAN 環境）などを想定しています。WAN 環境だと「**1MB/sec**」程度しか出ないという場合もまだまだ多いと思います。この場合は、**10GB** のサイズでも **2 時間 46 分**ぐらいかかてしまうことになります。これであれば、**USB 2.0** が **20MB/sec**（20 倍のスピード）ぐらい出るので、10GB でも **8 分 20 秒**という見積もりになります。

以前の弊社のデータセンターを変更した案件では、データセンター間の転送速度が **1MB/sec** だったので、ユーザー データベースに関しては、オフライン バックアップではなく、後述の**オンライン バックアップ**（BACKUP ステートメント）で実行して、それを **gzip** に圧縮（∵SQL Server 2005 だったのでバックアップ圧縮が利用できなかったため）、圧縮したファイルを **1MB/sec** の転送速度でコピーを行いました。このとき、万が一ネットワークが切れた場合を考慮して、並行して USB HDD へもファイルをコピーして（変更前のデータセンター内での作業）、コピー完了後に、それをタクシーで新しいデータセンターまで運ぶということを行いました。幸い、深夜だったので、ネットワークが切れることもなく、タクシーが渋滞にハマることもなく、どちらもスムーズだったのですが、ネットワークに遅れること 10 分で、USB HDD が到着した、ということがありました。いずれにしても、二重、三重で万が一の事態に備えることは重要なので、実際の移行やアップグレードにあたっては、事前にいろいろなパターンを検証しておくことが重要です。

## tempdb をコピーしないという選択も有り

オフライン バックアップでは、システム データベースを丸ごと複製する、と説明しましたが、実は **tempdb** データベースに関しては、旧マスター側でバックアップを取得していなくても、新規サーバー側で復元することができます。弊社のお客様では、tempdb は、（性能のために）あらかじめ大きいサイズへ設定していることがほとんどなのですが、これをオフライン バックアップする（ファイル コピーする）となると、それだけで大変な時間がかかってしまいます。例えば、1MB/sec の WAN 環境で 50GB の tempdb をコピーしようとすると、それだけで 13 時間 53 分もかかってしまうわけです。このような場合には、tempdb はコピーしなくても大丈夫です。

tempdb データベースは、SQL Server の起動時に再作成されるので、tempdb データベースのファイル（tempdb.mdf、templog.ldf）が存在しなくても、作成先となるフォルダー パスへの NTFS アクセス許可（サービス アカウントに対する NTFS アクセス許可）があれば、tempdb データベースを復活させることができます。このとき、SQL Server サービスの起動が完了した後に、tempdb データベースの復旧作業が行われるので、この復旧が完了するまでは、tempdb データベースへアクセスすることができません。また、.mdf ファイルの復旧には、**瞬時初期化**機能が有効であれば、あっという間に完了しますが、これが有効になっていない場合は復旧にも時間がかかります（復旧時間は、ストレージの書き込み性能に左右されます）。瞬時初期化機能を有効化するには、サービス アカウントに対して「**ボリュームの保守タスクを実行**」権利を付与しておきます。

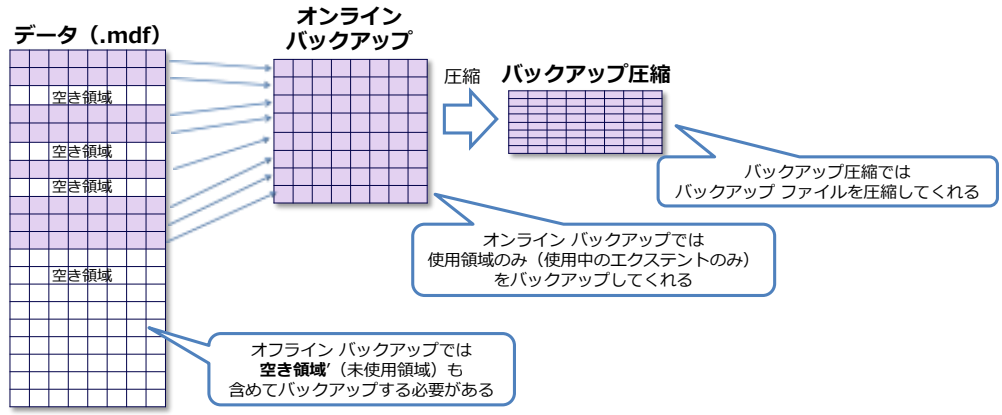
## オンライン バックアップ時間の見積もり（ユーザー データベースの場合）

1MB/sec など低速な WAN 環境の場合には、**ユーザー データベース**に関しては、オフライン バックアップで取得するよりも、**オンライン バックアップ**（完全バックアップ）で取得した方が速く完了する場合があります。オンライン バックアップにおけるバックアップ時間は、次のように考えることができます。

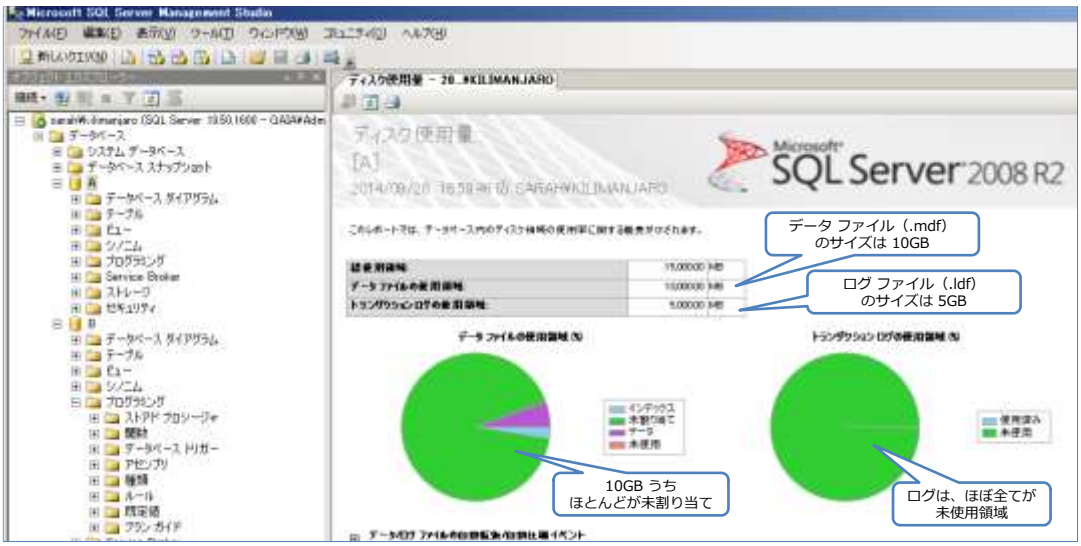
**オンライン バックアップにかかる時間**
**= オンライン バックアップ（BACKUP ステートメントの実行）にかかる時間**
**+ バックアップ ファイルをコピーするのにかかる時間**

オンライン バックアップでは、**バックアップ圧縮**機能を利用して、（バックアップ時に）バックアップ ファイルを圧縮しておくのがお勧めです（後述）。

オフライン バックアップよりも、オンライン バックアップのほうが手順が増えますが、オンライン バックアップを利用するメリットは、次のようになります。



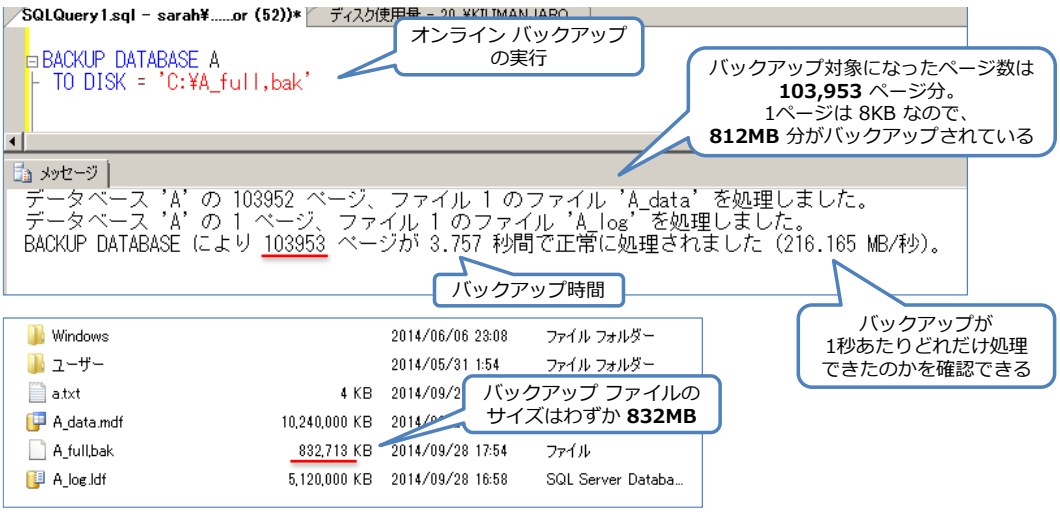

**オフライン バックアップ**では、**.mdf／.ldf** ファイルをコピーで取得するので、ファイル内の使用領域と未使用領域に関係なく、**ファイル サイズの分だけコピー**をする必要があるのに対して、**オンライン バックアップ**であれば、**使用領域のみ**（使用中のエクステントのみ）のバックアップで済みます。データベース内の使用領域は、Management Studio で次のように確認できます。



該当データベースを右クリックして、[**レポート**] メニューの [**ディスク使用量**] からこのレポートを参照することができます（画面は SQL Server 2008 R2 ですが、他のバージョンでも同じように操作できます）。このレポートでは、[**データ ファイルの使用領域**] が .mdf ファイルのサイズで、下の円グラフが実際の使用領域（データやインデックスで利用している領域）なのか、未使用領域（未割り当て／未使用）なのかを確認できます。円グラフのほとんどが「**緑**」の場合は、ほとんどが未使用領域という意味になります。このデータベースは、**.mdf ファイル**が **10GB** で、そのうちの使用領域（使用中のエクステント）が **1GB** 程度、**.ldf ファイル**（ログ）が **5GB** で、そのほぼすべてが未使用領域の状態です。このとき、**オフライン バックアップ**を実行する場合は、10GB と 5GB の**合計 15GB** 分のファイルをコピーしなければなりません。


仮に、ファイルの転送測度が **20MB/sec**（USB 2.0 を想定）だとすると、**12 分 30 秒**かかることになります。

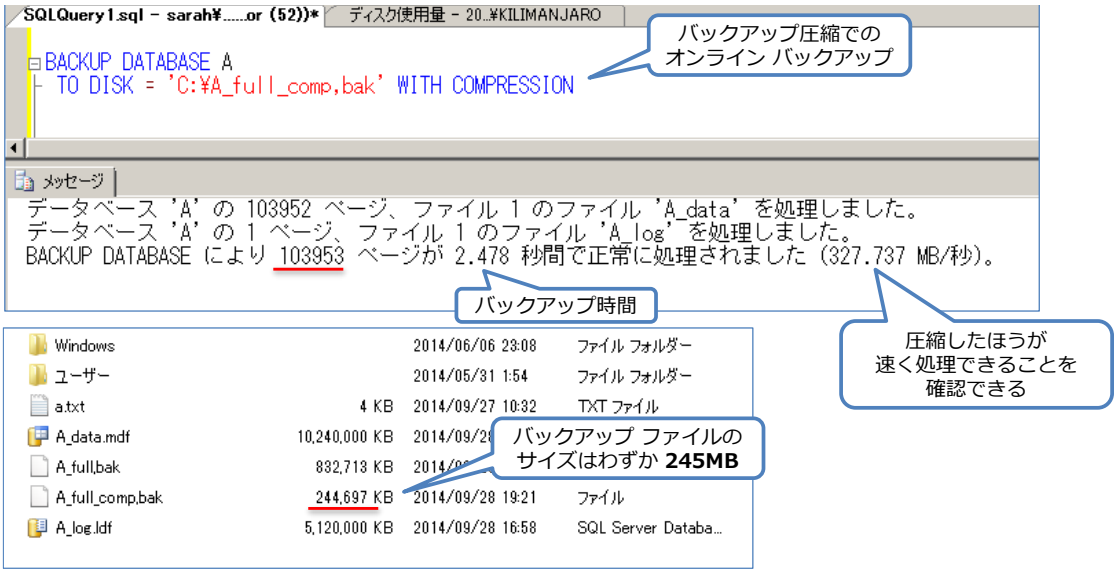
これに対して、オンライン バックアップであれば、使用領域（使用中のエクステント）のみのバックアップで済むので、次のようにバックアップ ファイルのサイズは、わずか **832MB** で済みます。




この例では、バックアップ時間は **約 4 秒**（3.757 秒）で、ファイル サイズが **832MB** だったので、ファイルの転送速度が **20MB/sec** だとすると、**42 秒**でファイルのコピーが完了することに

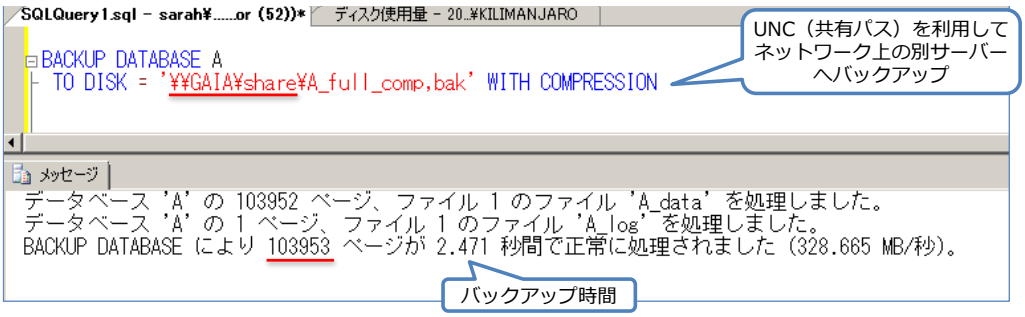
なります（合計で **46 秒**）。これは極端な例になりますが、データ ファイル（.mdf／.ldf）と使用領域に差があるような場合には、このような差が生まれることになるので、ユーザー データベースに関しては、低速回線の場合や、USB 2.0 などの低速ストレージにコピーする場合には、オンライン バックアップを利用することを検討してみてください。

オンライン バックアップでは、SQL Server 2008 Enterprise または SQL Server 2008 R2 Standard エディション以上であれば、**バックアップ圧縮**機能を利用することもできます（SQL Server 2008 のときは Enterprise エディションが必要でしたが、SQL Server 2008 R2 以降では Standard エディションでも利用できるようになりました）。バックアップ圧縮は、次のように **WITH COMPRESSION** キーワードを付けるだけで実行できます。




バックアップ時間は **約 2.5 秒**（2.478 秒）に短縮でき、ファイル サイズが **245MB**（1/3 以下）にまで圧縮できているので、ファイルの転送速度が **20MB/sec** だとすると、**12 秒**でファイルのコピーが完了することになります（合計で **13.5 秒**）。圧縮なしの場合は **45 秒**、オフライン バックアップの場合は **12 分 30 秒**かかることを考えると、バックアップ圧縮の効果が分かると思います。

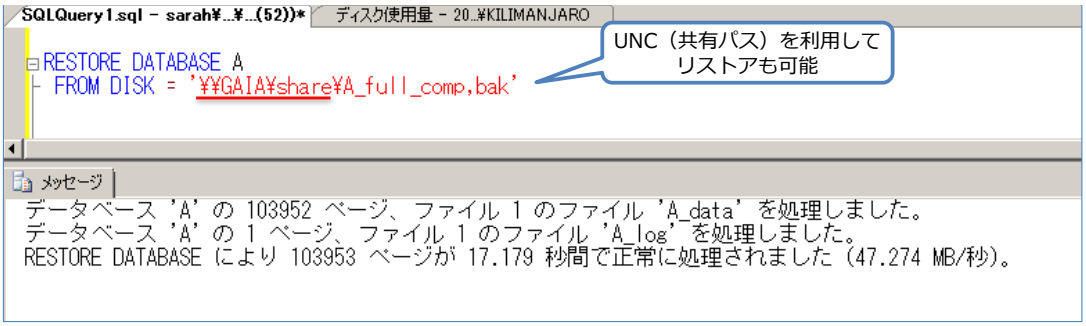
オンライン バックアップでは、次のように **UNC**（共有フォルダー）を指定して、ネットワーク上のサーバーへ直接バックアップすることも可能です。




これを利用すれば、バックアップをしつつ、ネットワーク コピーも行ってしまうということができます。LAN 環境などのネットワークが安定している場合には、お勧めのバックアップ方法です。一方、WAN 環境などのネットワークが安定していない場合には、万が一ネットワークが切れてしま

った場合にバックアップをイチからやり直さなければならなくなるので、お勧めではありません。

オンライン バックアップからの復元時（**RESTORE DATABASE** ステートメント）も、次のように **UNC** を指定して実行することができます。




このように UNC を直接利用することでも、実行時間を短くできる場合があるので、検討してみることをお勧めします（ネットワークを介す場合は**圧縮**も重要になります）。

別マシンへの環境の丸ごと複製では、ユーザー データベースのバックアップおよび復元にかかる時間が大きな時間を占めることになるので、事前に入念なテストを行って、しっかりと実行時間を見積もっておくことが重要になります。

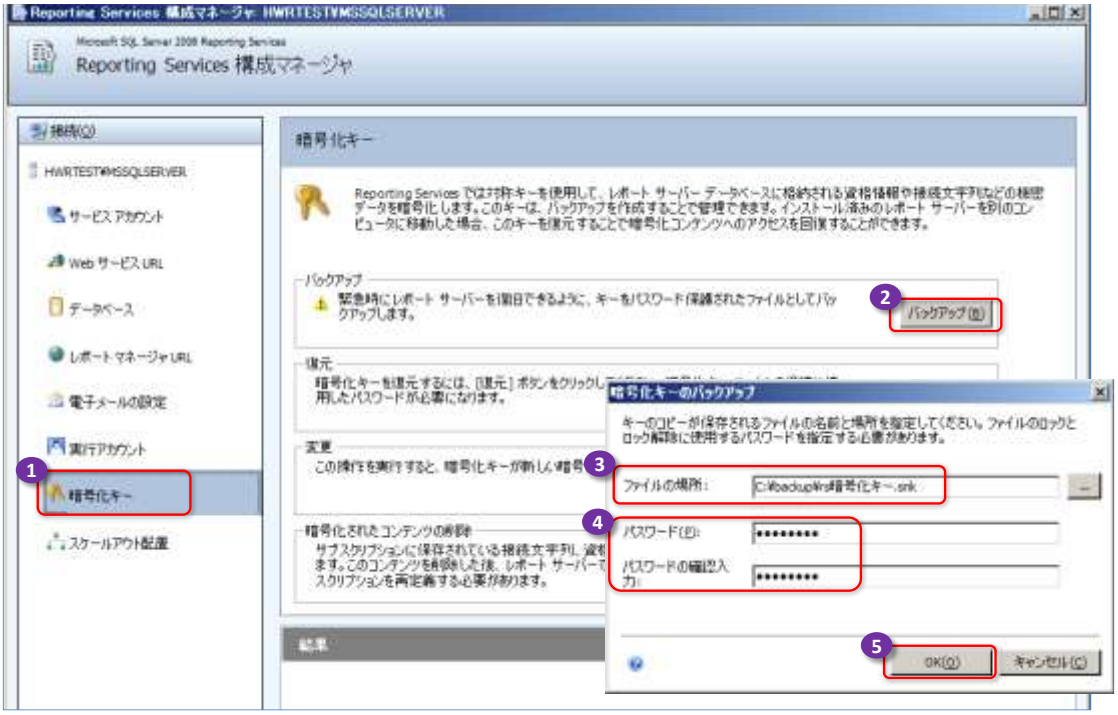
なお、オンライン バックアップに関しては、日々の定期バックアップにかかる時間を測定している場合は、そこからバックアップにかかる時間を見積もることができます。また、完全バックアップとログ バックアップを定期的に取得している環境の場合は、データベースの複製時には、最後のログ バックアップ以降の（未取得の）ログ バックアップのみを取得するだけで、最新の状態へ複製していくことも可能です。

## 4.12 Reporting Services の新規サーバーへのアップグレード

**Reporting Services** を新規サーバーへ完全複製して、その後 SQL Server 2016 へアップグレードする場合は、追加の手順が必要になります。ここではその作業を説明します。

### 旧マスターで暗号化キーをバックアップする

Reporting Services を新規サーバーへ完全複製するには、旧マスターで**暗号化キー**をバックアップしておく必要があります。暗号化キーのバックアップは、「**Reporting Services 構成マネージャー**」ツールを利用して、次のように［**暗号化キー**］ページから行います（画面は、SQL Server 2008 ですが、他のバージョンでもほとんど同じ操作で行えます）。



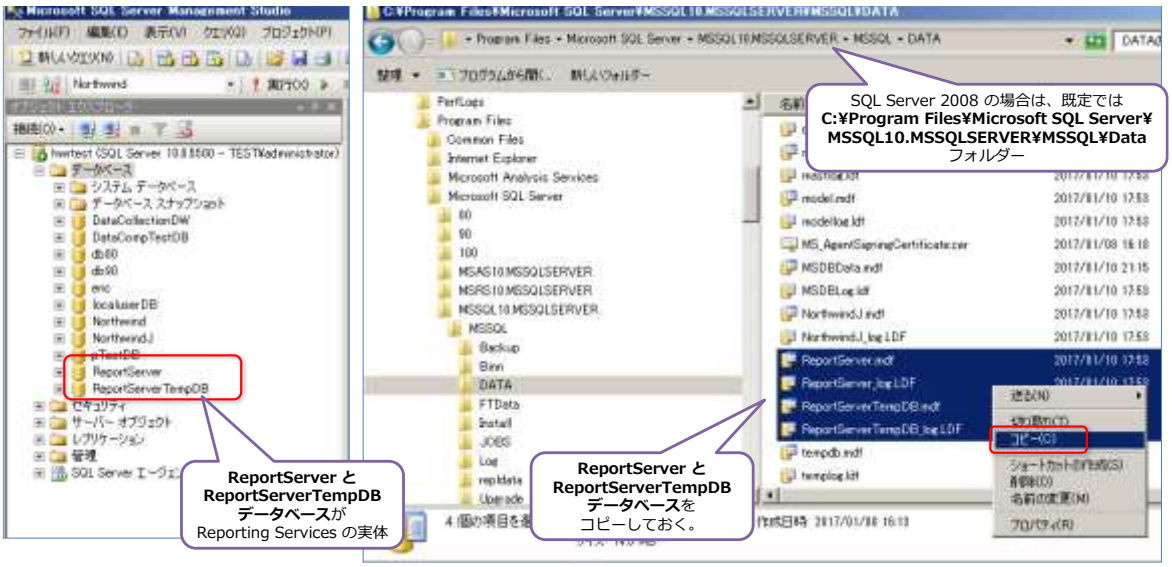

このページでは、［**バックアップ**］をクリックして、［**暗号化キーのバックアップ**］ダイアログが表示されたら、［**パスワード**］と［**パスワードの確認**］に任意のパスワードを設定し、［**ファイルの場所**］にバックアップ先のファイル名を指定します（画面は **C:¥backup** フォルダーの下に **rs 暗号化キー.snk** というファイル名を指定）。ここで設定するパスワードは、**複雑なパスワード**（大文字・小文字と @などの特殊文字を入れたりするなど）を指定する必要があり、これを行っていない場合は、次のように失敗します。

バックアップが成功した場合は、次のように表示されます。


また、設定したパスワードは、新規サーバーで復元する際に必要になるので覚えておいてください。バックアップしたファイル（**～.snk**）は、新規サーバーからアクセス可能なファイル サーバーや USB HDD など持ち運び可能なメディアに保存します。

## ReportServer、ReportServerTempDB データベースのバックアップ

**Reporting Services の実体**（レポート本体や、レポートに関する各種の設定）は、SQL Server 上の **ReportServer** および **ReportServerTempDB** データベースです。




このデータベース（**.mdf／.ldf**）は、既定では SQL Server のシステム データベースと同じ場所

に作成されます（SQL Server 2008 の場合は以下のフォルダー）。

**C:¥Program Files¥Microsoft SQL Server¥MSSQL10.MSSQLSERVER¥MSSQL¥DATA**

このフォルダーにある **ReportServer** の実体（**ReportServer.mdf**、**ReportServer_log.ldf**）と **ReportServerTempDB** データベースの実体（**ReportServerTempDB.mdf**、**ReportServerTempDB_log.ldf**）を**オフライン バックアップ**します（SQL Server サービスと Reporting Services サービスを停止した上で、**ファイルをコピー**します）。

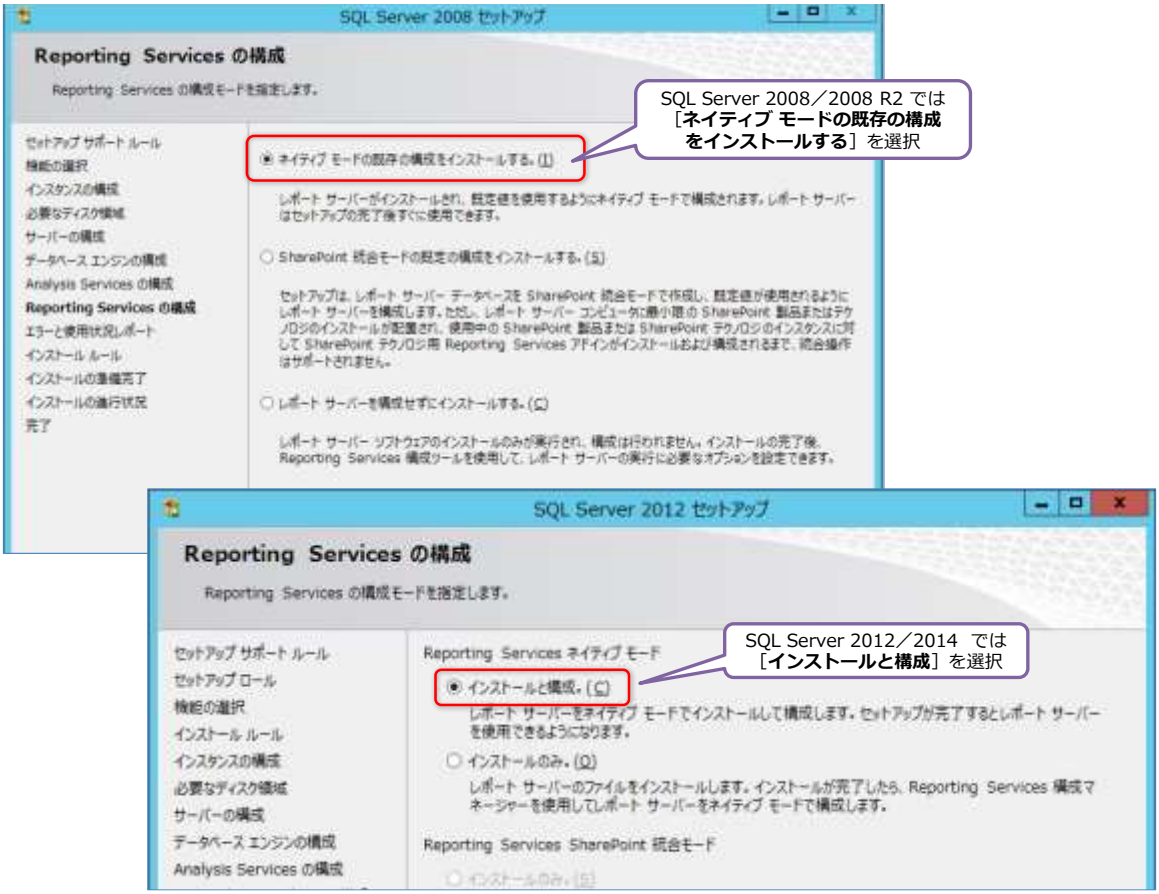
この 2 つのデータベースは、次のように**オンライン バックアップ**（**BACKUP** ステートメント）でも取得しておくことを強くお勧めします（これは SQL Server を起動した状態で実行します）。

```
BACKUP DATABASE ReportServer
 TO DISK = 'C:¥backup¥ReportServer_full.bak'
BACKUP DATABASE ReportServerTempDB
 TO DISK = 'C:¥backup¥ReportServerTempDB_full.bak'
```


もし、オフライン バックアップからの復元がうまくいかなかった場合には、このオンライン バックアップを利用して、復元を行うことができるので、取得しておくことをお勧めします（∵ Reporting Services は、万が一のときでも、この **ReportServer** と **ReportServerTempDB** のオンライン バックアップと**暗号化キー**のバックアップさえあれば、元の状態に戻すことができるからです。オンライン バックアップから復元手順については、次の章で説明しているので、以降の手順でどうしてもうまく復元できない場合には、こちらもご覧いただければと思います。次の章は、SQL Server 2016 での操作ですが、SQL Server 2008 や 2008 R2、2012、2014 でも同じように復元できます）。

## Reporting Services インストール時の構成の有無

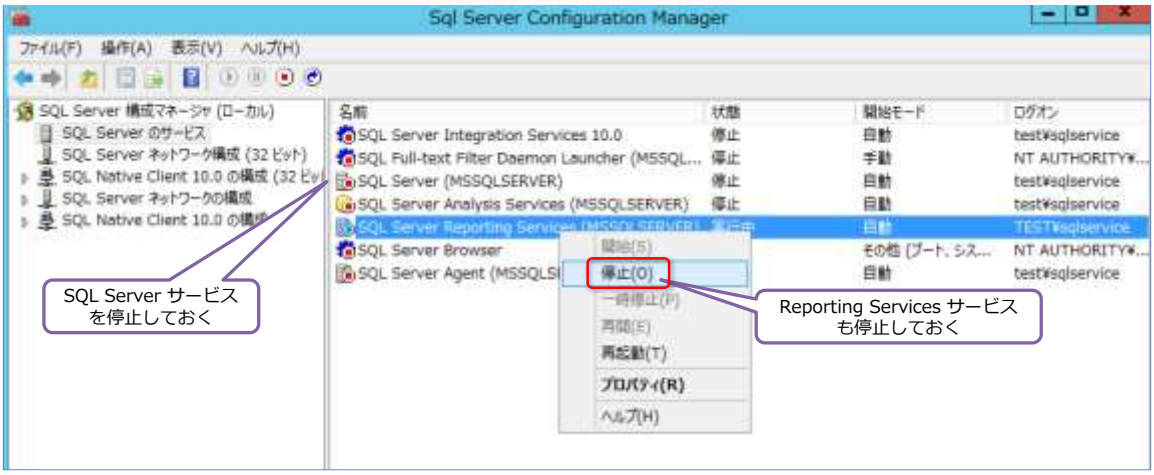
以降の操作は、SQL Server の Reporting Services をインストールするときに、次のように［**既定の構成をインストールする**］または「**インストールと構成**」を選択した場合の操作手順になります。




もし、［**レポート サーバーを構成せずにインストールする**］または「**インストールのみ**」を選択している場合は、Reporting Services 構成マネージャーでサーバーを構成をした後に、以降の手順を実施してください。

## 新規サーバーでのオフライン バックアップからの復元

旧マスター側で取得したオフライン バックアップは、SQL Server の他のシステム データベースと同様、**ファイル コピー**（上書きコピー）で復元します。この際、**SQL Server サービス**と**Reporting Services サービス**を停止しておくようにします。



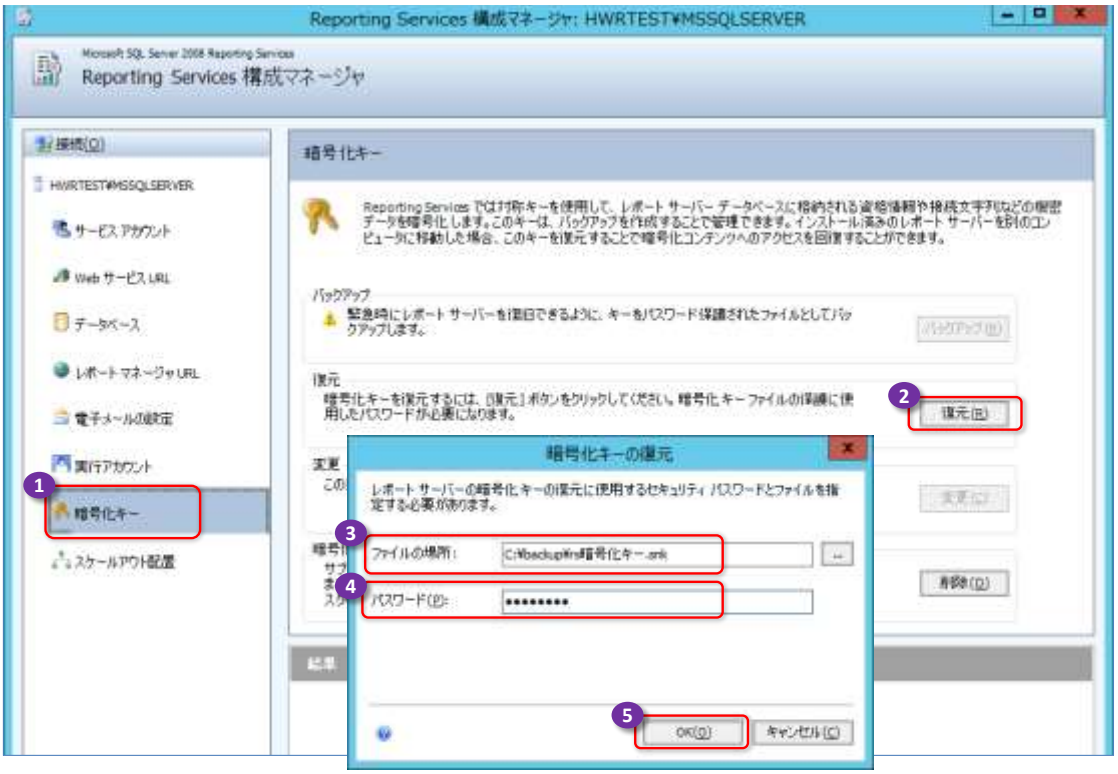

ファイルのコピーが完了したら、SQL Server サービスを起動した後に、Reporting Services サービスを起動します（ReportServer と ReportServerTempDB データベースを SQL Server に認識させた後に、Reporting Services サービスを起動します）。

サービス起動後は、Reporting Services がこれだけで完全復元されているわけではなく、**レポート マネージャー**（**http://localhost/Reports**）や**レポート サーバー**（**http://localhost/ReportServer**）にアクセスしても、次のようにエラーになります。


どちらも「**初期化されていません**」エラーが表示されますが、これを解決するには、暗号化キーを復元するようにします。

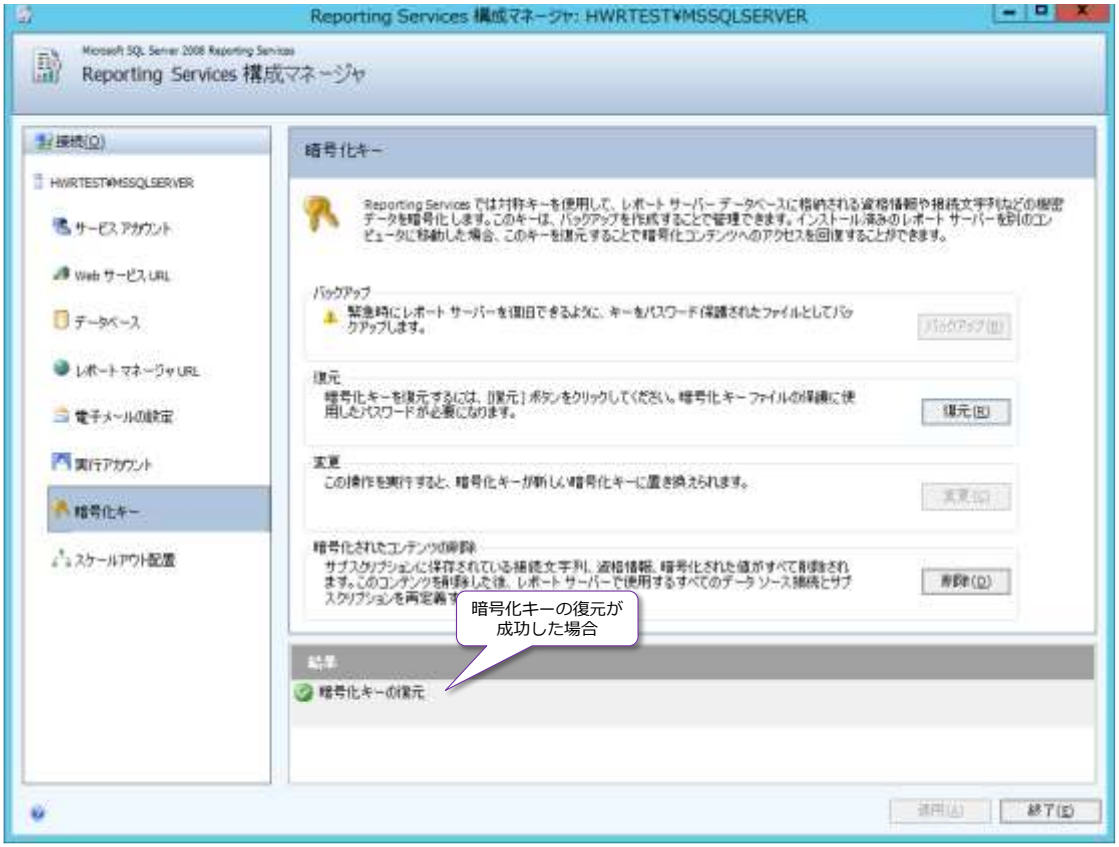
## 暗号化キーの復元

**暗号化キー**の復元は、次のように［**暗号化キー**］ページで、［**復元**］ボタンをクリックします。



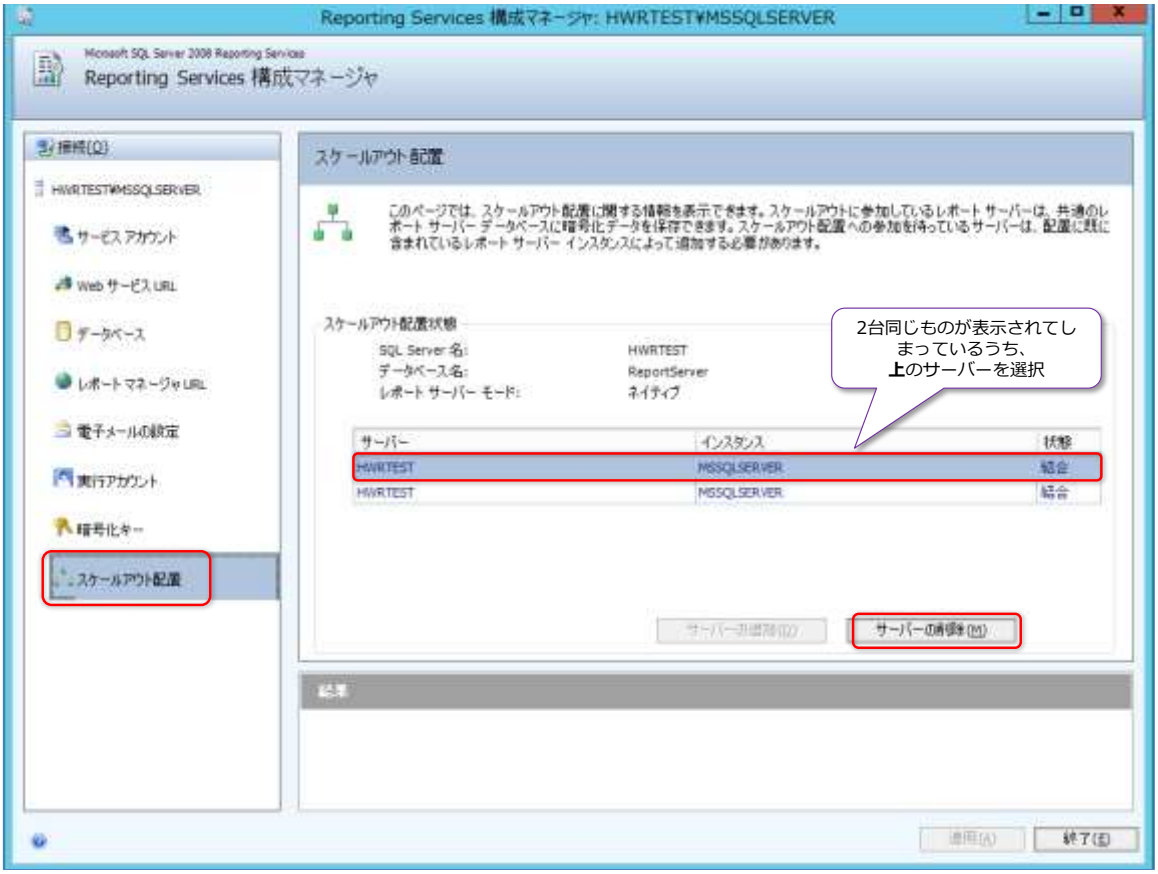

［**暗号化キーの復元**］ダイアログが表示されたら、［**ファイルの場所**］でバックアップした暗号化キー（**.snk**）を指定して、［**パスワード**］にバックアップ時に指定したパスワードを入力します。

暗号化キーの復元が成功すると、次のように表示されます。



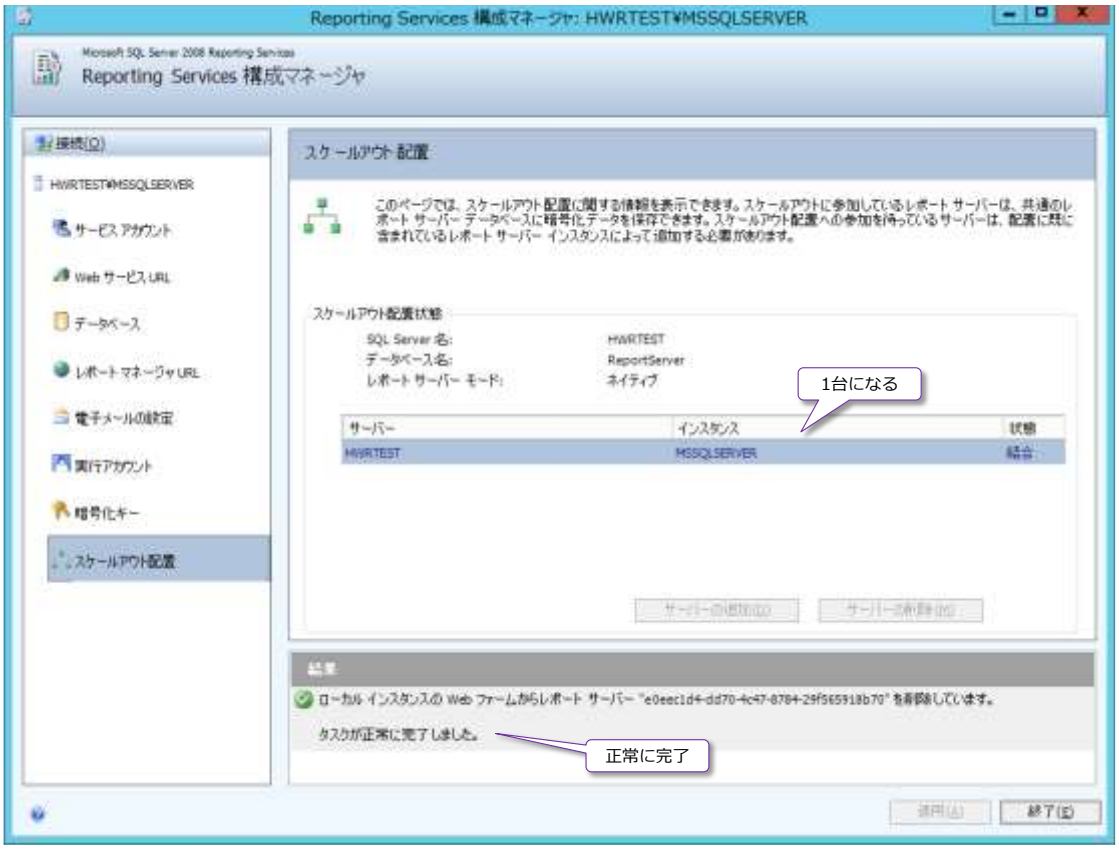
## スケールアウト配置の設定を変更

暗号化キーの復元が完了すると、次のように［**スケールアウト配置**］ページが表示されます。

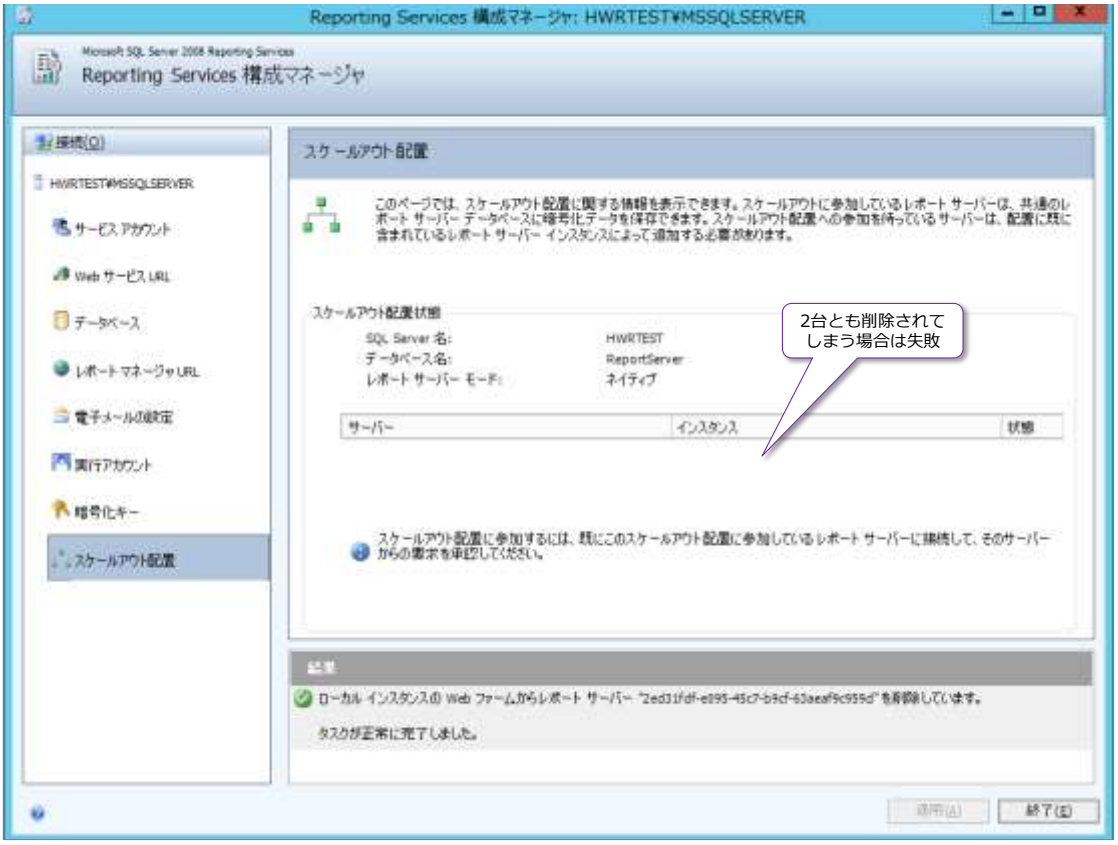



このページでは、2 台のサーバーが表示されてしまいます。2 台のうち、上に表示されるものが、古いサーバー（旧マスター）になるので、これを選択して、［**サーバーの削除**］ボタンをクリックします。次のように確認ダイアログが表示されるので、［**OK**］ボタンをクリックして実行します。


これで、数秒すると次のように 1 台のサーバーのみになります。




もし、1 台のサーバーが残らずに、次のように 2 台とも削除されてしまう場合は、削除に失敗しています。

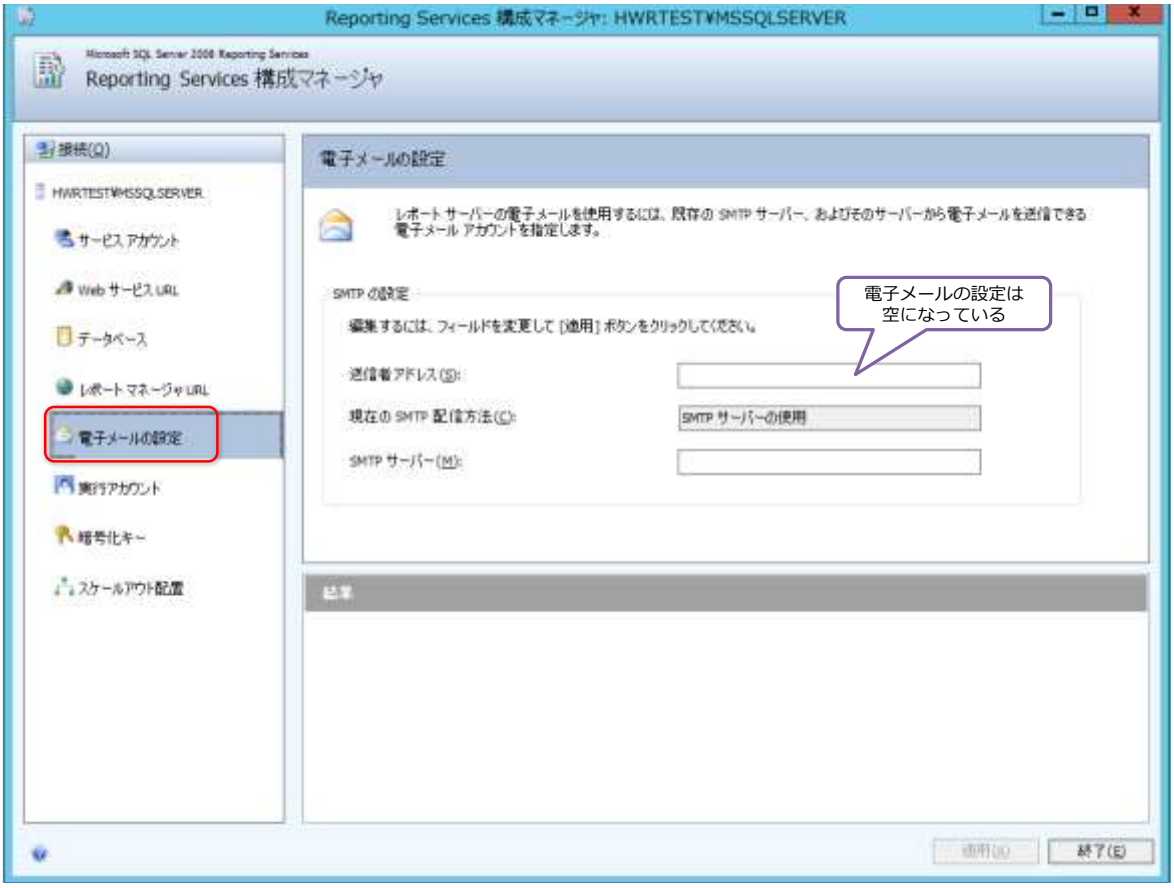



この場合は、［**サーバーの削除**］ボタンをクリックしたときに選択したサーバーが間違っているの

で、もう一度、**暗号化キー**をバックアップから復元し、再度、[**サーバーの削除**] ボタンで、今度は違うサーバーを選択して、実行してみてください（[**スケールアウト配置**] に 1 台のみ表示されるようになるまで実行してみてください）。

## 電子メールの設定

Reporting Services の電子メールの設定に関しては、手動で再設定をする必要があります。




電子メールの設定は、「**rsreportserver.config**」ファイルに格納されているので、これは手動で再設定する必要があります。なお、このファイルは、SQL Server 2008 の場合は、既定で次のフォルダーに作成されています。

**C:¥Program Files¥Microsoft SQL Server¥MSRS10.MSSQLSERVER¥Reporting Services¥ReportServer**

SQL Server 2008 R2、2012、2014 の場合は、既定は次のフォルダーになります。

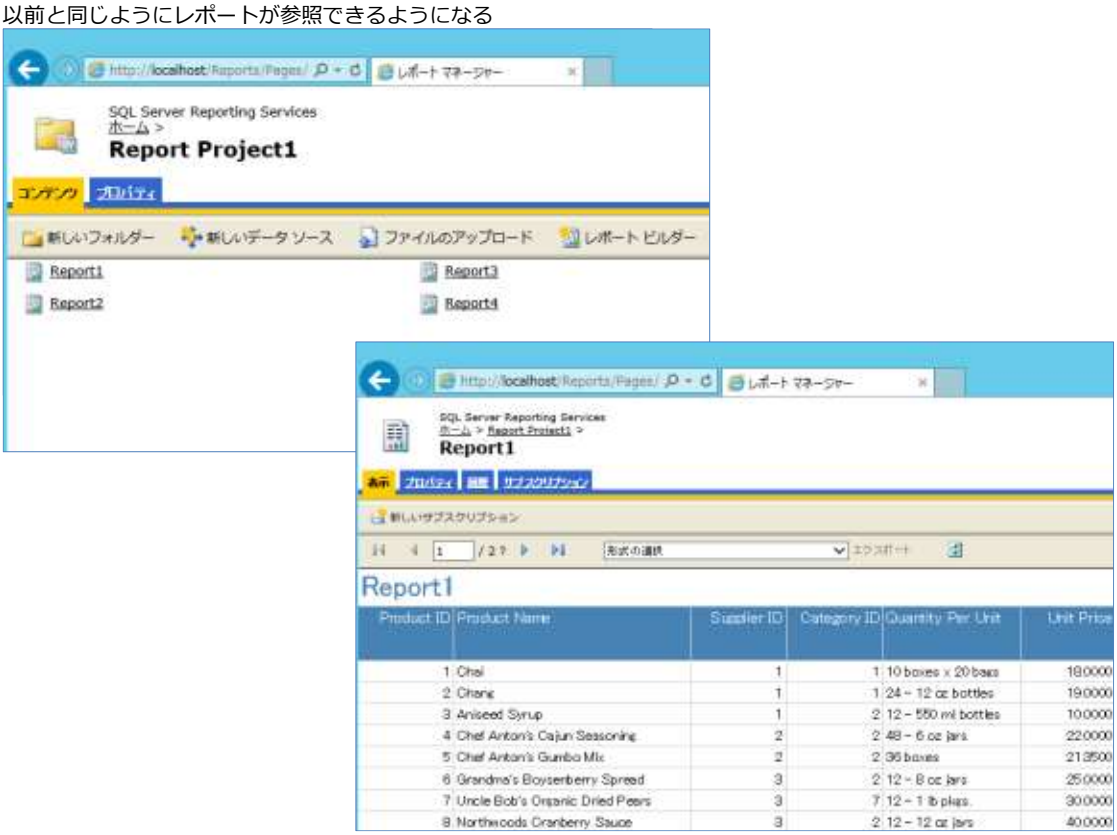
SQL Server 2008 R2
**C:¥Program Files¥Microsoft SQL Server¥MSRS10_50.MSSQLSERVER¥Reporting Services¥ReportServer**

SQL Server 2012
**C:¥Program Files¥Microsoft SQL Server¥MSRS11.MSSQLSERVER¥Reporting Services¥ReportServer**

SQL Server 2014
**C:¥Program Files¥Microsoft SQL Server¥MSRS12.MSSQLSERVER¥Reporting Services¥ReportServer**

以上で、作業が完了です。これで旧マスターと同じようにレポートを参照できるようになります。もしうまく復元できない場合は、次の章で説明するオンライン バックアップからの復元手順を試してみてください。

以前と同じようにレポートが参照できるようになる




これで、SQL Server 2016 へアップグレードする準備ができました。SQL Server 2016 へのアップグレード方法については、前の章を参照してください。

## 追加手順のまとめ

行った作業をまとめると、次のようになります。

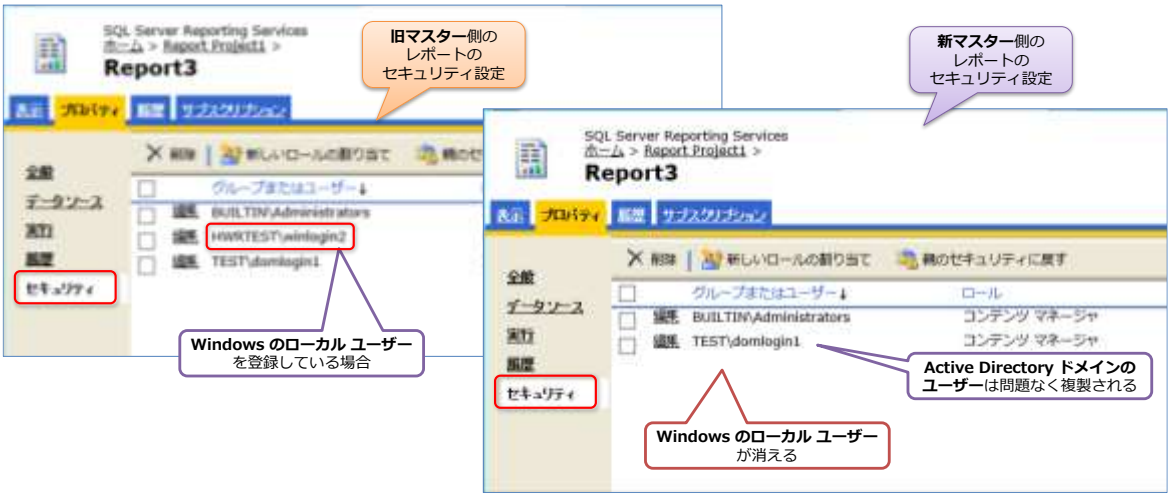
- 旧マスターで、Reporting Services 構成マネージャーで**暗号化キー**をバックアップする
- 旧マスターで **ReportServer** と **ReportServerTempDB** データベースの**オフライン バックアップ**を取得する（SQL Server サービスと Reporting Services サービスを停止した上で、ファイルをコピーする）
- 万が一のときのために、**ReportServer** と **ReportServerTempDB** データベースの**オンライン バックアップ**を取得しておく（SQL Server サービスを起動している状態で、**BACKUP** ステートメントを実行する）
- 新マスターで、SQL Server サービスと Reporting Services サービスを停止した上で、**オフライン バックアップを復元**する（ファイルの上書きコピー）
- 復元後、SQL Server サービスを起動した後に、Reporting Services サービスを起動す

る

- サービスが起動したら、Reporting Services 構成マネージャーで**暗号化キー**を復元する
- **スケールアウト配置**で余計なサーバーを削除する
- **電子メールの設定**を再設定する
- **SQL Server 2016 へのアップグレード**を実施する（**第 3 章の 3.25** を参照）

## Windows のローカル ユーザーを利用している場合の注意点

新規サーバーへの複製後の注意点としては、**レポートのセキュリティ設定**で **Windows のローカル ユーザー**を利用している場合です。これは、次のような状況です。

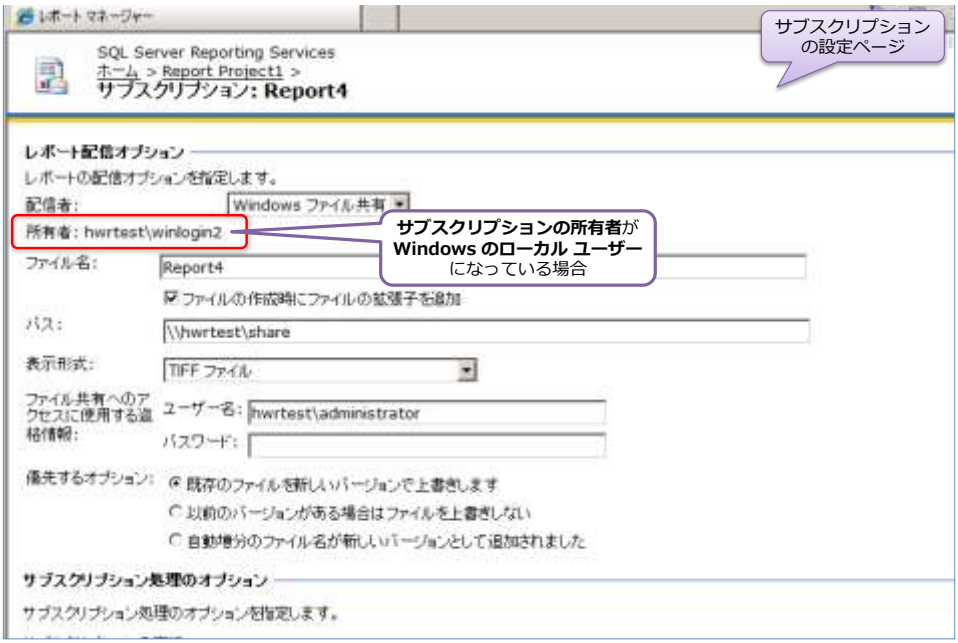



Windows のローカル ユーザーの場合は、マシン（OS）が変わったとすると、同じ名前のユーザーを作成したとしても、**内部的な SID**（**Security ID**）が異なってしまうので、Reporting Services にとっては、認識できないユーザーとなってしまいます。このため、新規サーバーでは、Windows のローカル ユーザーが全く表示されない形になります。

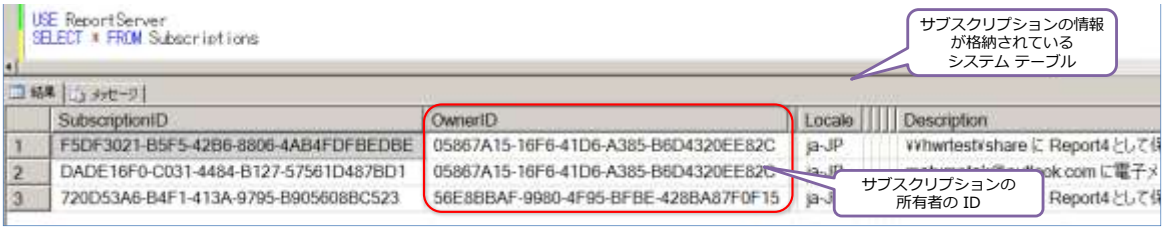
これを回避する方法はないので、Windows のローカル ユーザーを利用している場合は、再設定をする必要があります。これは、あくまでも Windows のローカル ユーザーを利用している場合のみの話しで、Active Directory ドメインのユーザーであれば問題なく複製されている（復元できている）ので、再設定をする必要はありません。

## サブスクリプションの所有者が Windows のローカル ユーザーの場合

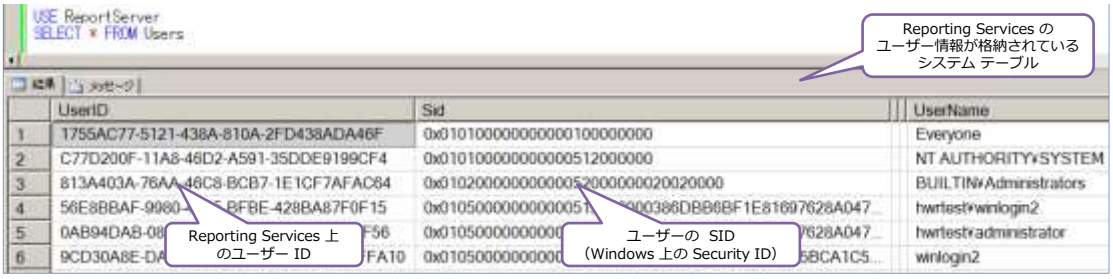
サブスクリプションの所有者が **Windows のローカル ユーザー**の場合は、サブスクリプションがうまく動作しません。これは次のような状況です。




これを回避するには、サブスクリプションを削除して、再作成をするか、**Subscriptions** テーブル（Reporting Services が利用しているシステム テーブル）を修正するようにします。**Subscriptions** テーブルは、**ReportServer** データベース内に存在します。




このテーブルには、**OwnerID** 列があり、ここにサブスクリプションの所有者の ID が格納されています。この ID は、Reporting Services 上の ID で、**Users** テーブルで管理されています。




**Users** テーブルでは、**UserID** 列が Reporting Services 上の ID（Reporting Services で管理するユーザーを識別するための ID）、**Sid** 列に Windows 上の **SID**（Security ID）が格納されています。したがって、次のように **Subscriptions** テーブルの **OwnerID** 列の値を、**Users** テーブルで確認した**ドメインのユーザー**の値に変更（**UPDATE**）すれば、この問題が解決します。

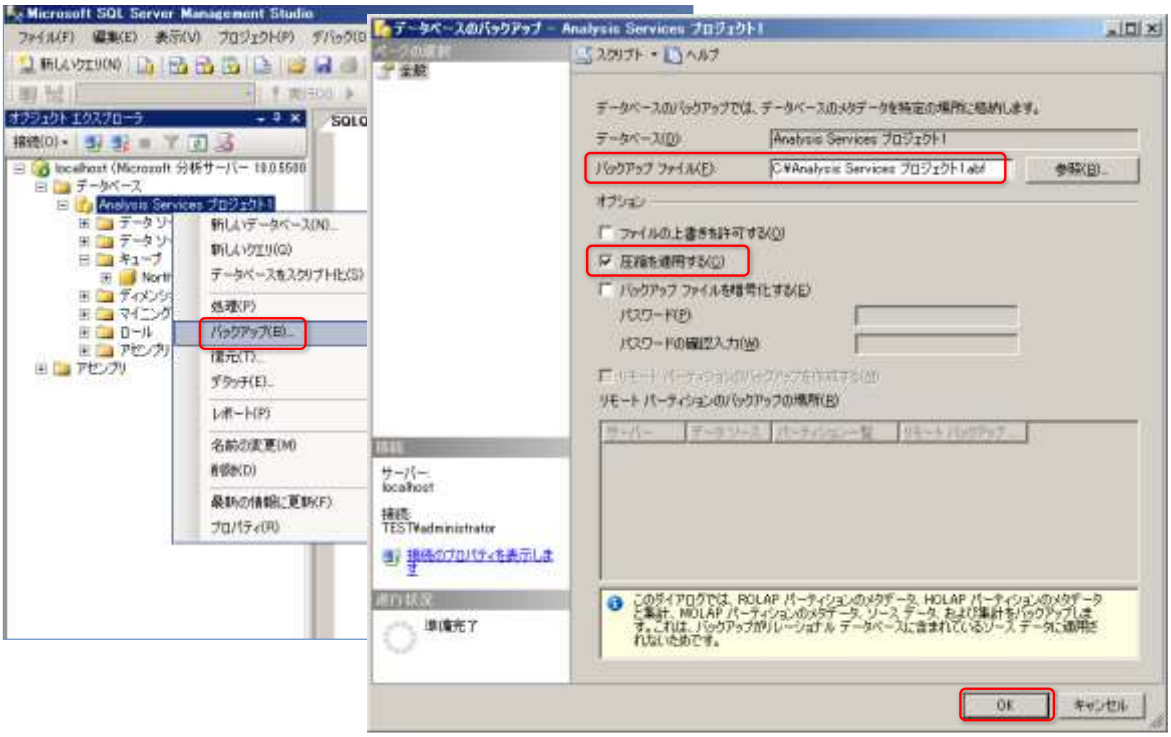
## 4.13 Analysis Services の新規サーバーへのアップグレード

**Analysis Services**（OLAP キューブ）を新規サーバーへ完全複製して、その後 SQL Server 2016 へアップグレードする場合は、追加の手順が必要になりますが、Reporting Services ほど手順が多いわけではありません。ここではその作業を説明します。

なお、SQL Server 2012 からは、インメモリ BI を実現できる「**表形式モード**」（**Tabular Mode**）も登場していますが、ここでは従来ながらの「**多次元モード**」（Multi Dimensional Mode、OLAP キューブ）での移行方法を説明します。

### 旧マスターでデータベースのオンライン バックアップ

**Analysis Services** では、**オフライン バックアップ**ではなくて、**オンライン バックアップ**でデータベースをバックアップするのがお勧めです。オンライン バックアップは、次のように Management Studio で Analysis Services に接続して行います（Analysis Services サービスは停止せずに、起動したまま実行します。画面は SQL Server 2008 ですが、他のバージョンでも同じように操作できます）。




データベースを右クリックして、[**バックアップ**] をクリックし、[**バックアップ ファイル**] へバックアップ先のファイル名をフル パスで入力します。[**圧縮を利用する**] はチェックを付けておくことをお勧めします（バックアップ ファイルを圧縮することができます）。

このように、バックアップ ファイルを圧縮できることが、オンライン バックアップをお勧めする理由で、特に Analysis Services のデータベースは巨大になりがちなので、できる限り、データベースをコンパクトにして、移動したほうが移動時間（ダウンタイム）を少なくできるので、お勧めです。

**Note： オフライン バックアップはお勧めではない**

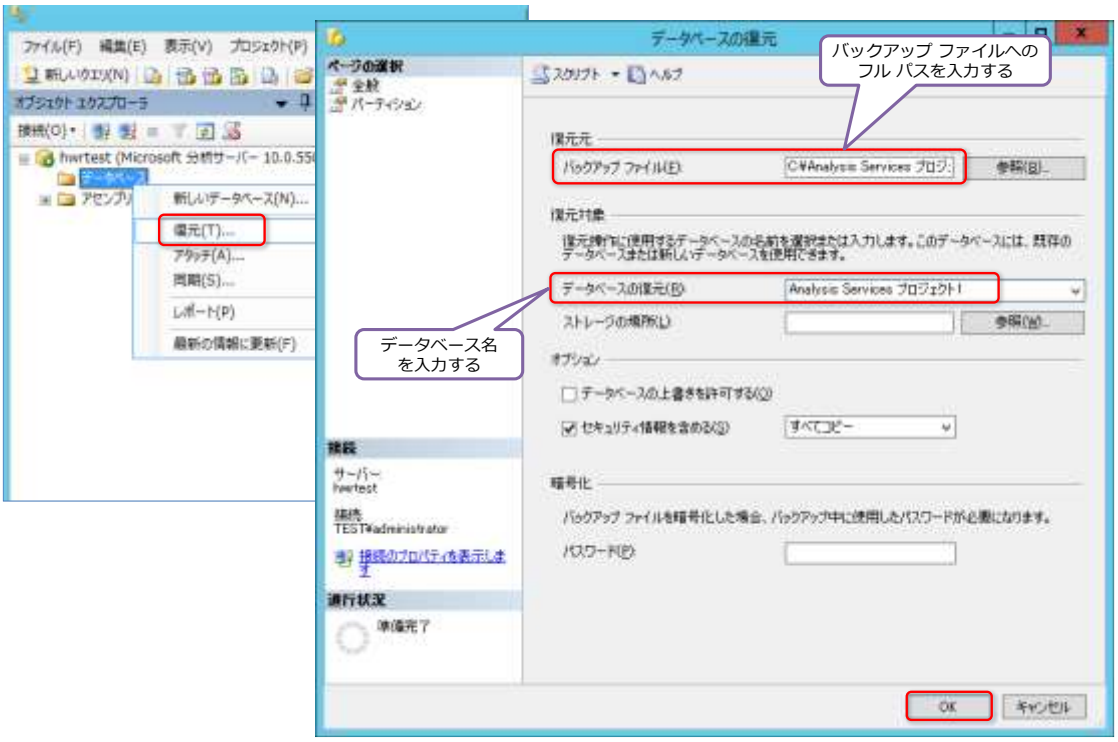
Analysis Services の多次元データベース（キューブ）の実体は、OS 上のファイル群です。SQL Server 2008 の場合は、既定では、以下のフォルダーに格納されています。

**C:¥Program Files¥Microsoft SQL Server¥MSAS10.MSSQLSERVER¥OLAP¥Data**

オフライン バックアップの場合は、Analysis Services サービスを停止した上で、この **Data** フォルダーを丸ごとコピーして、新規サーバー上へ丸ごと復元（上書きコピー）すれば、旧マスターと同じようにデータベースを利用できるようになります。ただし、**Data** フォルダーをコピーするときには、直下の **TMP** ファイルのコピーに失敗するので、これを除外してコピーする必要があります。Analysis Services の場合は、オンライン バックアップのほうが簡単で分かりやすいので、オンライン バックアップを利用することがお勧めになります。

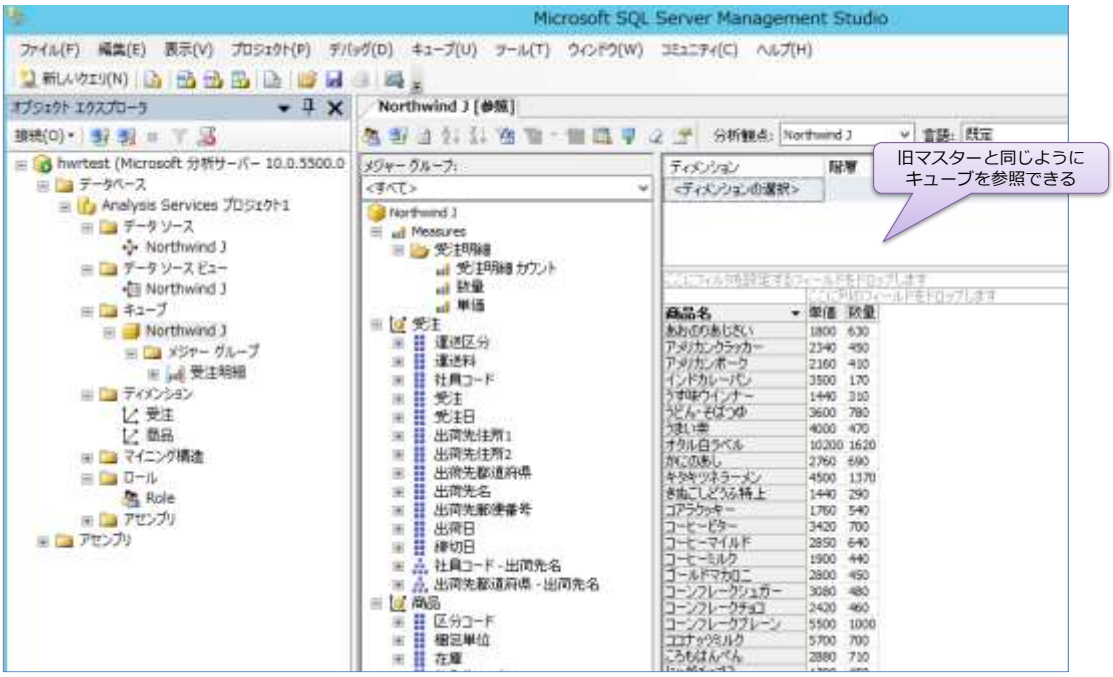
## 新規サーバーでのオンライン バックアップからの復元

旧マスター側で取得したオンライン バックアップは、次のように Management Studio で Analysis Services に接続して、復元できます（Analysis Services サービスを起動した状態で実行します。画面は SQL Server 2008 の場合）。



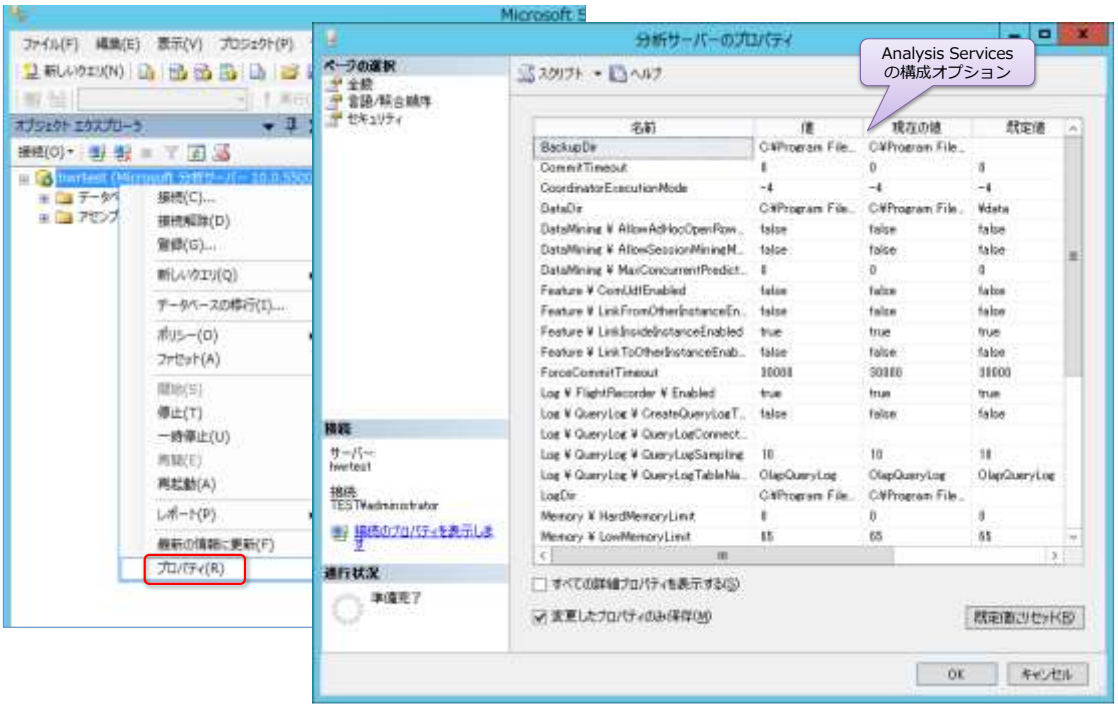

［**データベース**］フォルダーを右クリックして、［**復元**］をクリックし、［**バックアップ ファイル**］へバックアップ ファイルをフル パスで入力します。［**データベースの復元**］には復元後のデータベース名を入力して、［**OK**］ボタンをクリックします。

復元が完了すると、旧マスターと同じようにデータベースを参照できるようになります。




## Analysis Services の構成オプション

旧マスターで、**Analysis Services の構成オプション**を変更している場合は、新規サーバー側でも再設定をする必要があります。




なお、これらの設定は、「**msmdsrv.ini**」という設定ファイルに格納されていて、SQL Server 2008 の場合は、既定で次のフォルダーに作成されています。

**C:¥Program Files¥Microsoft SQL Server¥MSAS10.MSSQLSERVER¥OLAP¥Config**

SQL Server 2008 R2 の場合は「**MSAS10.MSSQLSERVER**」の部分が「**MSAS10_50.～**」、SQL Server 2012 では「**MSAS11.～**」、SQL Server 2014 では「**MSAS12.～**」に代わります。

多くの構成オプションを変更している場合は、このファイルをもとに修正すると、作業が楽になると思います。

以上で、作業が完了です。これで旧マスターと同じように Analysis Services（OLAP キューブ）を利用できるようになります。

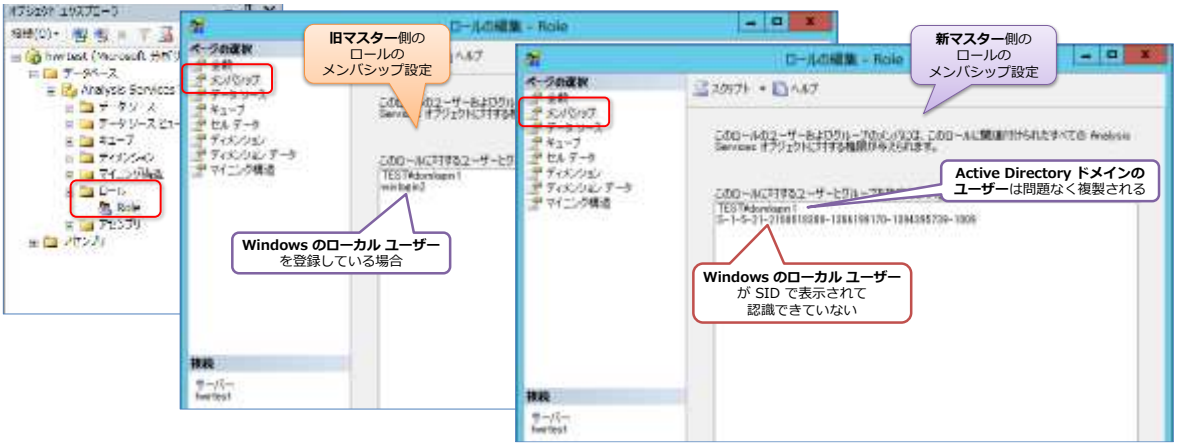
## 手順のまとめ

作業手順をまとめると、次のようになります。

- 旧マスターで、**Analysis Services のデータベース**を**オンライン バックアップ**する
- 新マスターで、**オンライン バックアップ**から復元する
- **Analysis Services の構成オプション**を変更している場合は、再設定する
- **SQL Server 2016 へのアップグレード**を実施する（**第 3 章の 3.26** を参照）

## Windows のローカル ユーザーを利用している場合の注意点

新規サーバーへの複製後の注意点としては、**Analysis Services のセキュリティ設定**（**ロール**）で**Windows のローカル ユーザー**を利用している場合です。これは、次のような状況です。




Windows のローカル ユーザーの場合は、マシン（OS）が変わったとすると、同じ名前のユーザーを作成したとしても、**内部的な SID**（**Security ID**）が異なってしまうので、Analysis Services にとっては、認識できないユーザーとなってしまいます。このため、新規サーバーでは、Windows のローカル ユーザーの名前が不明になるので、SID で表示される形になります。

これを回避する方法はないので、Windows のローカル ユーザーを利用している場合は、再設定をする必要があります。これは、あくまでも Windows のローカル ユーザーを利用している場合のみの話しで、Active Directory ドメインのユーザーであれば問題なく複製されている（復元できている）ので、再設定をする必要はありません。

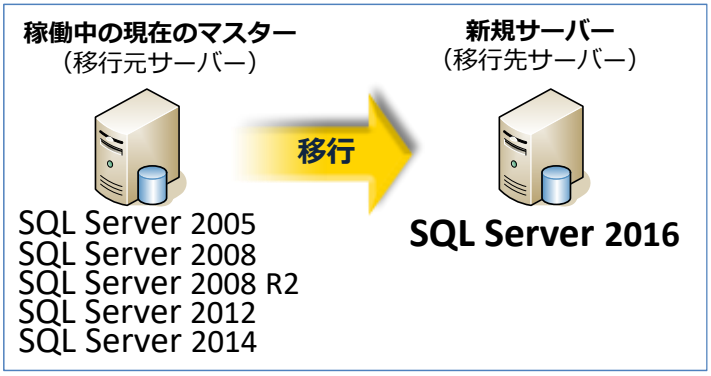
# STEP 5. ケース 3　新規サーバーへの移行

この章では、このドキュメントで扱う 3 つのケースのうちの「**ケース 3　新規サーバー（別マシン）への移行**」（マイグレーション）について説明します。

- ✓ SQL Server 2016 インストール
- ✓ データベースの移行方法
- ✓ 統計の更新
- ✓ フルテキスト インデックスの再構築
- ✓ データベースの互換性レベルを 130 へ上げる
- ✓ システム データベース関連のオブジェクト（ログイン アカウント、リンク サーバー、tempdb 設定、構成オプション、ジョブ、警告、データベース メール、オペレーターなど）の再作成
- ✓ レジストリに格納されている情報の再設定
- ✓ 移行にかかる時間（ダウンタイム）の見積もり

## 5.1 ケース３「新規サーバー（別マシン）への移行」

この章では、このドキュメントで扱う 3 つのケースのうちの「**ケース３ 新規サーバー（別マシン）への移行**」（マイグレーション）について説明します。




このケースは、ハードウェアの保守切れや老朽化などによって、**新規サーバーの購入**（**ハードウェア リプレース**）を行って、そのマシンへ **SQL Server 2016** をインストールし、データベースや各種の設定を**移行**（マイグレーション）する方法です。弊社のお客様で一番多いのが、このケースです。ハードウェア リプレースを機会に、32 ビットから 64 ビット環境へ移行したり、SSD やフラッシュ ストレージなどの高速ストレージ環境へ移行したりするというお客様も多くいらっしゃいます。

この方法を利用すれば、**SQL Server 2005** のデータベースを SQL Server 2016 に移行することもできます（3 章と 4 章で紹介したインプレース アップグレードでは、SQL Server 2005 は未対応でした）。

また、**32 ビット環境**のオンプレミス上の **SQL Server 2005** や **2008**、**2008 R2**、**2012**、**2014** などを、Microsoft Azure（クラウド）上の仮想マシン（**Windows Server 2012** や **2012 R2**、**2016**）で動作させている **SQL Server 2016**（64 ビット）へ移行するといったことも簡単に行うことができます。

こうしたマシンをまたがった場合でも、SQL Server では簡単に移行を行うことができます。**データベースの移行**は、**標準のバックアップと復元**機能で行うことができ、バックアップと復元機能を利用すれば、別マシンであっても簡単にデータベースを複製することができ、32 ビットから 64 ビットへ変更したとしても、移行先の OS が変わったとしても、オンプレミスからクラウドに変わったとしても、簡単に移行することができます。

各種の設定（ログイン アカウントやジョブ、警告、リンク サーバーなどの設定）に関しても、**スクリプト生成**機能を利用することで、簡単に移行することができます。

## 移行手順の概要

このケースでの移行手順の概要は、次のとおりです

1. **Data Migration Assistant** による事前チェックを行う
2. **新規サーバー**へ **Windows Server 2012**（X64 版）以上の OS をインストールする（移行元と同じ OS である必要はありません）
3. **Active Directory ドメイン**環境の場合は、**新規サーバー**をドメインに参加させる
4. **SQL Server 2016** を**インストールするためのソフトウェア要件**を確認する

   OS は、Windows Server 2012 以降の X64 版をサポート、Windows Server 2012 R2 の場合は KB 2919355 が必要など。データベース メール機能を利用している場合は、.NET Framework 3.5 SP1 をインストールしておく必要がある
5. **新規サーバー**へ **SQL Server 2016** をインストールする
6. **新規サーバー**へ **SQL Server 2016** の**最新の修正プログラム**（CU や Service Pack など）をインストールする

   CU2 には、性能向上に関する修正が入っているので、できる限り最新の修正プログラムを適用しておくことをお勧めします。

   Reporting Services を利用している場合は、SP1 に対する重要な更新プログラムがあるので、CU4（SP1＋CU1）以上を適用しておくことをお勧めします。
7. **Management Studio**（管理ツール）の最新版をダウンロードして、インストールする（オプション）
8. **移行元サーバー**（SQL Server 2005／2008／2008 R2／2012／2014）の**データベース**を、**新規サーバー**（SQL Server 2016）へ移行する（バックアップと復元機能を利用）
9. **統計**（Statistics）を更新する
10. **フルテキスト インデックス**を再構築する（フルテキスト検索機能を利用している場合）
11. **データベースの互換性レベルを 130** へ上げる（オプション）

    互換性レベルの影響については**クエリ ストア**機能を利用することで簡単にチェック可能（実行プランの比較や、プラン強制もできる）。
12. **データベースの所有者**を確認／設定する（∵所有者が空の場合には、**データベース ダイアグラム**と後述の **SQL CLR オブジェクト**が動作しないため）
13. **SQL CLR オブジェクト**の権限セットで「**UNSAFE**」または「**外部**」を利用している場合は、**TRUSTWORTHY** オプションを有効化する
14. **システム データベース**関連のオブジェクトを移行する（ログイン アカウントやサーバー ロ

ール、tempdb の設定、リンク サーバー、ユーザー定義エラー、構成オプション、データベース メール、ジョブ、警告、オペレーターなどのうち、移行元で設定を変更／利用しているものがある場合は、それらを移行する）

15. **レジストリ**に格納されている情報を再設定する（サービスの自動起動やサービス アカウント、認証モード、TCP ポート番号、起動時パラメーターでのトレースフラグの設定などのうち、移行元で設定を変更しているものがある場合は、それらを再設定する）
16. **OS の設定**で、移行元で変更しているものがある場合は、それらを再設定する（フォルダー構成や、NTFS アクセス許可、ユーザーの権利、共有フォルダーなど）
17. **メンテナンス プラン**（保守計画）を利用している場合は、メンテナンス プランを再作成する
18. **BIDS**（Business Intelligence Development Studio）または **SSDT-BI**（SQL Server Data Tools - Business Intelligence）を利用している場合は、**SSDT** の最新版をダウンロードして、インストールする（オプション）
19. **Integration Services** を利用している場合は、SSIS パッケージを移行する
20. **SQL Server Audit** や**リソース ガバナー**、**パフォーマンス データ コレクション**などのサーバー管理機能を利用している場合は、これらを再設定する
21. **レプリケーション**や**ログ配布**、**可用性グループ**、**データベース ミラーリング**などのサーバー間の連携機能を利用している場合は、これらを再設定する

前述したように、新規サーバーへの**データベースの移行**は、**標準のバックアップと復元**機能を利用して、簡単に行うことができます。データベース ダイアグラムや SQL CLR オブジェクトを利用している場合は、追加の作業が必要になりますが、これらはコマンドを 1 つ 2 つ実行するだけなので簡単な作業です。

**各種の設定**（**システム データベース**や**レジストリ**に格納されている設定、**メンテナンス プラン**など）は、移行元で設定を変更していたり、該当オブジェクトを利用したりしている場合には、再設定／再作成が必要になります。**システム データベース**に格納されているものに関しては、ほとんどのものが GUI 操作で**スクリプト生成**することができるので、簡単に移行することができます。**レジストリ**に格納されているものに関しては、SQL Server はレジストリをほとんど利用していないので、こちらも再設定は簡単です。

この移行方法を利用するメリットは、**OS** を簡単に変更できること（**Windows Server 2003／2003 R2** や **Windows Server 2008／2008 R2** から **Windows Server 2012** や **2012 R2**、**Windows Server 2016** へ変更するなど）、クロス プラットフォーム（移行元が **32 ビット**で、移行先が **64 ビット**など）でも関係がないこと、**Microsoft Azure** などの**クラウド環境**であったとしてもデータベースの移行が行えること、**SQL Server 2005 からの移行**にも対応していることです。

以降では、1 つ 1 つの手順を詳しく説明します。

## 5.2 新規サーバーへの SQL Server 2016 のインストール要件

まずは、**新規サーバーへ SQL Server 2016 をインストール**する方法について説明します。

### インストール要件

**SQL Server 2016** をインストールするための**ソフトウェア要件**は、アップグレードのときと同様、次のとおりです。

| | 要件 |
|---|---|
| OS | Windows Server 2012 以降 または<br>Windows Server 2012 R2 + KB 2919355 以降 または<br>Windows Server 2016 以降<br>ただし、X64（64ビット版）の OS のみがサポートされます。 |
| ソフトウェア | ・.NET Framework 4.6（事前インストールは不要で、SQL Server 2016 のインストール時に自動的にインストールされます） |

* Developer エディションの場合は、Windows 8、Windows 8.1、Windows 10 にインストールすることも可能です。
* SQL Server 2016 からは、Management Studio と SSDT（SQL Server Data Tools）は、別途ダウンロード／インストールする形に変更されました。
  Management Studio は、32ビット版の Windows 7（SP1 以降）や Windows 8 にインストールすることもできます。
  Management Studio を Windows Server 2012 または Windows 7／8／8.1 にインストールする場合には KB 2862966 の Security Update が必要になります。
* データベース メール機能を利用する場合は、.NET Framework 3.5 SP1 をインストールする必要があります（サーバー マネージャーから追加できます）。
* Polybase（Hadoop 連携）機能を利用する場合は、64ビット版の JRE 7.51（JRE 7 Update 51）以上が必要になります。
* なお、SQL Server 2005 では、Reporting Services をインストールするために IIS や ASP.NET が必須コンポーネントになっていましたが、SQL Server 2008 からは IIS や ASP.NET が不要になりました。

以前の SQL Server との大きな違いは、**SQL Server 2016** が **X64**（64 ビット版）の OS のみでサポートされるようになったことと、**Windows Server 2012** 以降の OS が必要になることです。SQL Server 2014 のときは Windows Server 2008 SP2 以降の OS がサポートされていましたが、SQL Server 2016 から変更になりました。

Windows Server 2012 R2 を利用する場合は、**KB 2919355** が必要になります。これは **with Update** パッケージを利用して OS をインストールすることで、OS のインストール時に KB 2919355 を一緒にインストールしておくこともできます。

## .NET Framework 3.5 SP1 のインストール（データベース メール利用の場合）

**SQL Server 2016** では、データベース メール機能を利用する場合のみ、**.NET Framework 3.5 SP1** が必要になります。これは、次のように**サーバー マネージャー**の［**役割と機能の追加**］ウィザードから簡単にインストールすることができます。

Windows Server 2012／2012 R2 での .NET Framework 3.5 SP1 のインストール


Windows Server 2016 での .NET Framework 3.5 SP1 のインストール


［**.NET Framework 3.5 Features**］の［**.NET Framework 3.5**］をチェックすることで、**.NET Framework 3.5 SP1** をインストールすることができます。これでデータベース メール機能を利用できるようになります。

## 5.3 SQL Server 2016 のインストール手順

SQL Server 2016 のインストール手順は、次のとおりです。

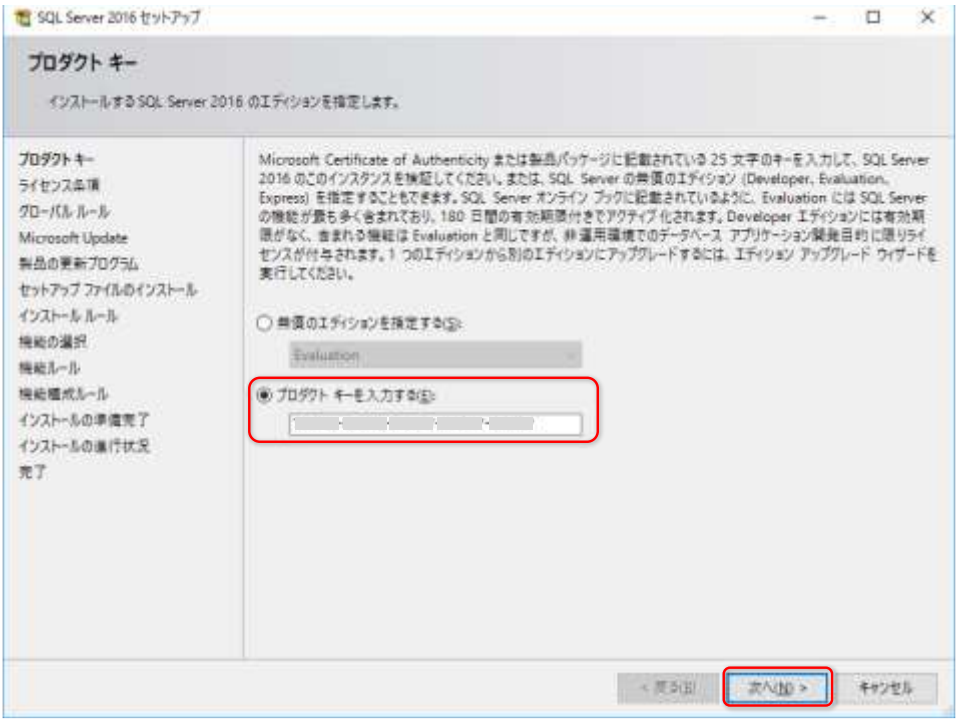
まずは、SQL Server のインストール メディアから **setup.exe** を実行して、次のように［**SQL Server インストール センター**］ページを表示します。


次に、［**インストール**］ページを開いて、次のように「**SQL Server の新規スタンドアロン インストールを実行するか、既存のインストールに機能を追加します**」をクリックします。


これで、インストール ウィザードが起動して SQL Server 2016 を新規インストールできるよう

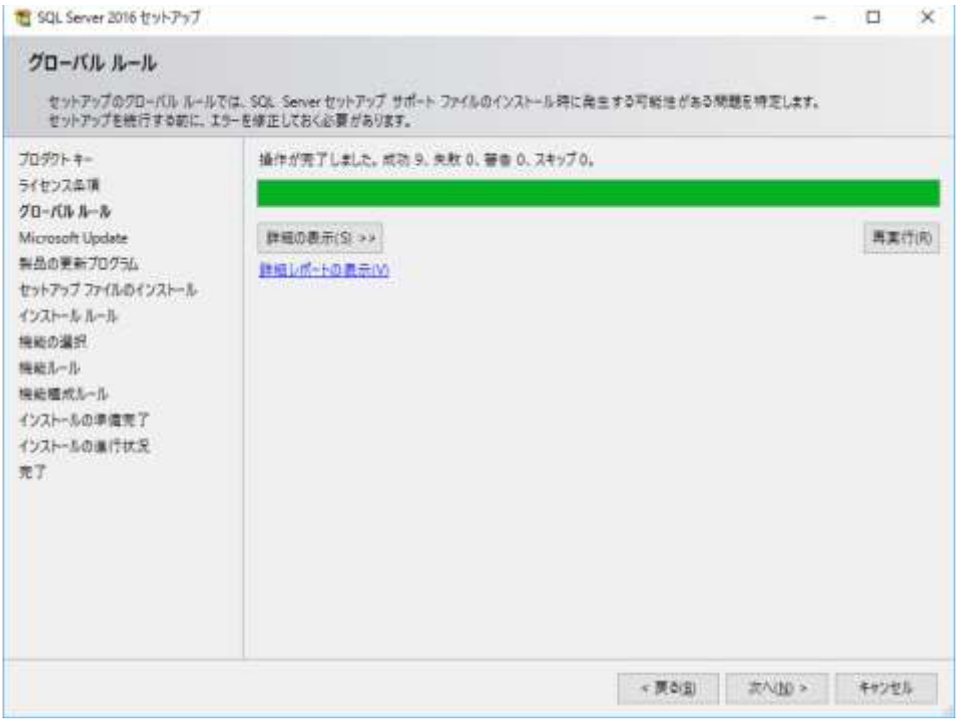
になります。

［**プロダクト キー**］ページが表示されたら、プロダクト キーを入力して、［**次へ**］ボタンをクリックします。



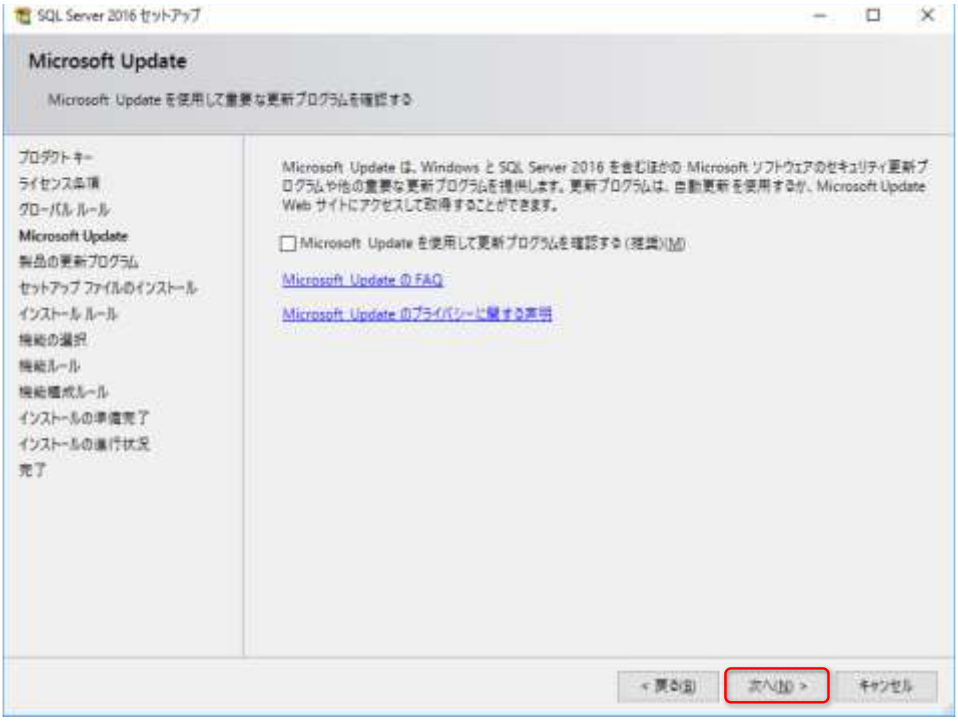

次の［**ライセンス条項**］ページでは、ライセンス条項の内容を確認した上で、［**ライセンス条項に同意します。**］をチェックして、［**次へ**］ボタンをクリックします。


次の［**グローバル ルール**］ページでは、インストール要件を満たしているかどうかがチェックされます。

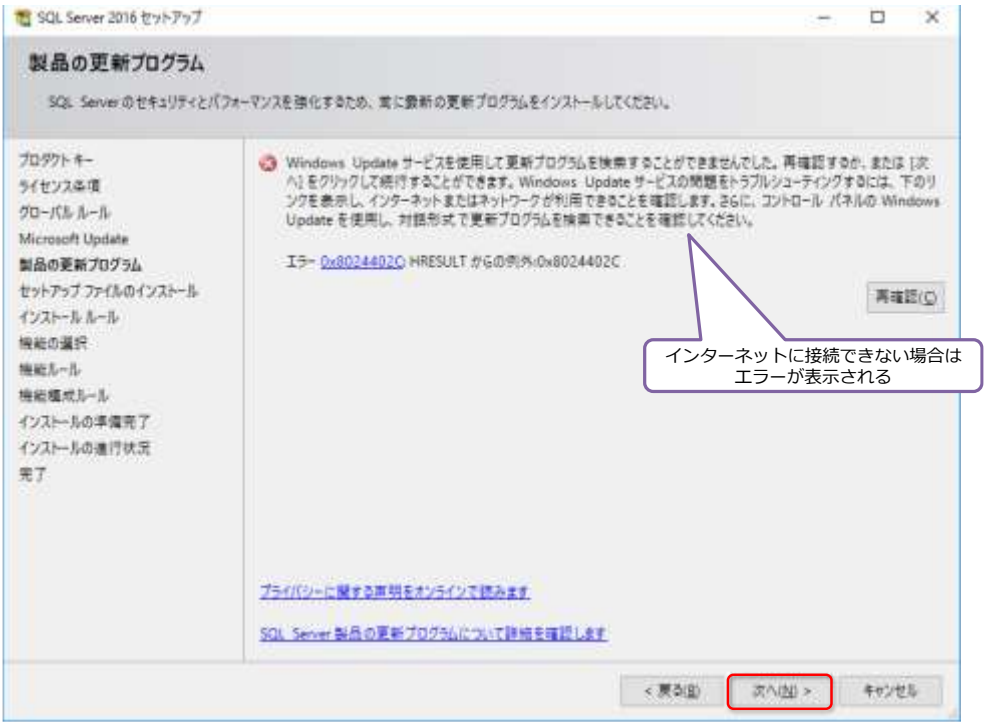



問題がない場合は、自動的に次のページへ進み、問題がある場合（警告や失敗がある場合）にはそれらが提示されます。

次の［**Microsoft Update**］ページでは、Microsoft Update を利用して、更新プログラムを確認するかどうかを設定して、［**次へ**］ボタンをクリックします。

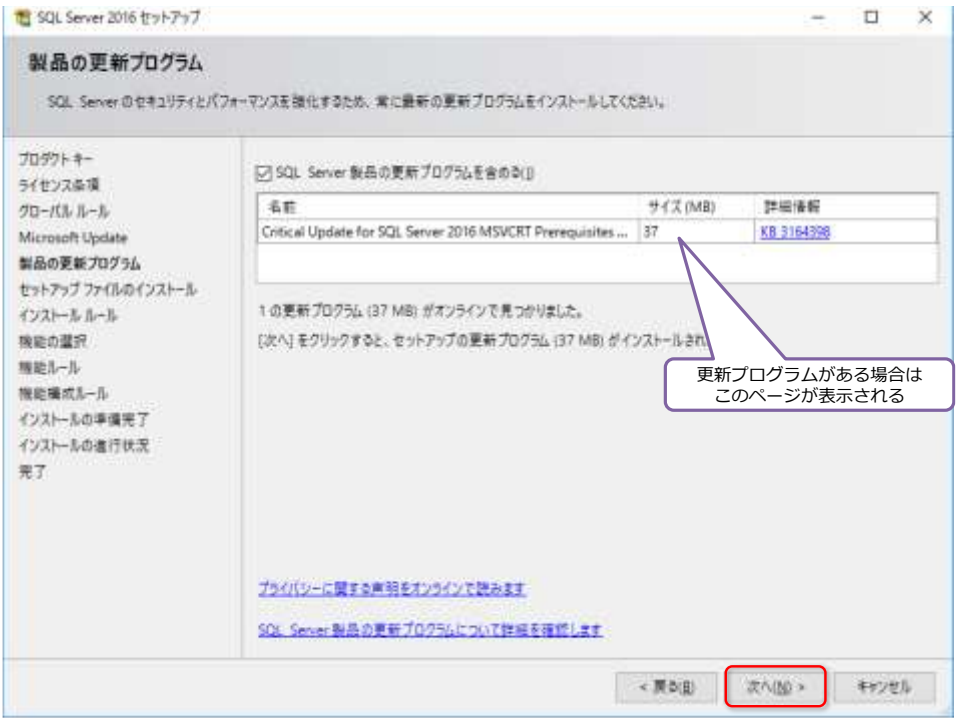



次の［**製品の更新プログラム**］ページでは、インターネットに接続できない環境の場合には、次のようにエラーが表示されます。



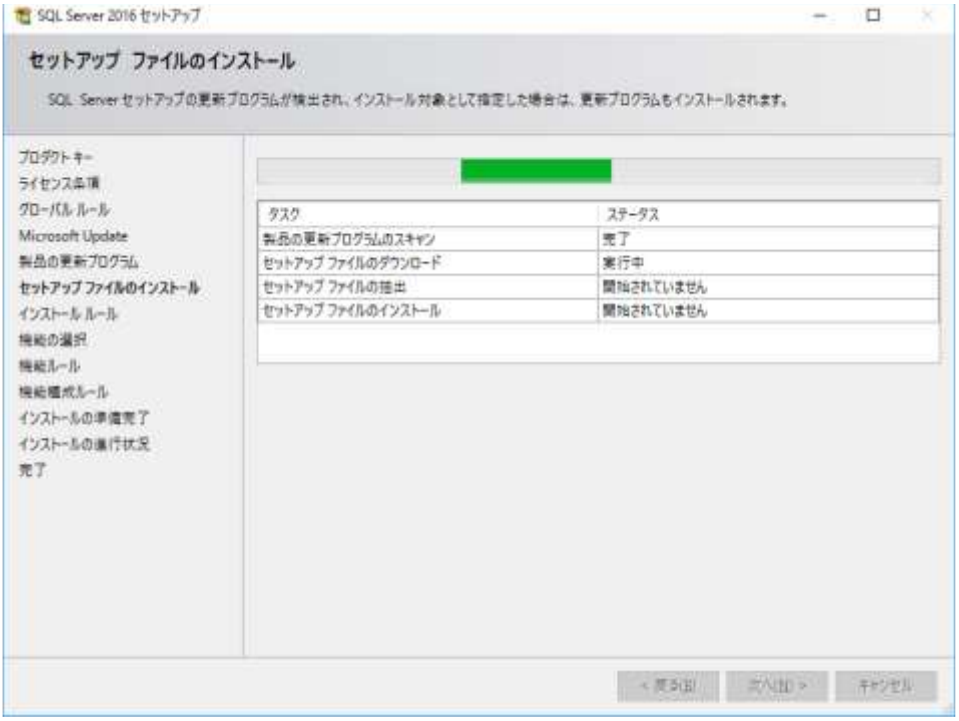

このエラーは、更新プログラムが取得できないという主旨のメッセージなので、[**次へ**] ボタンをクリックして、次のページへ進んで問題ありません。

インターネットに接続できる環境で、SQL Server に関する更新プログラムがある場合は、次のように［**製品の更新プログラム**］ページが表示されます。

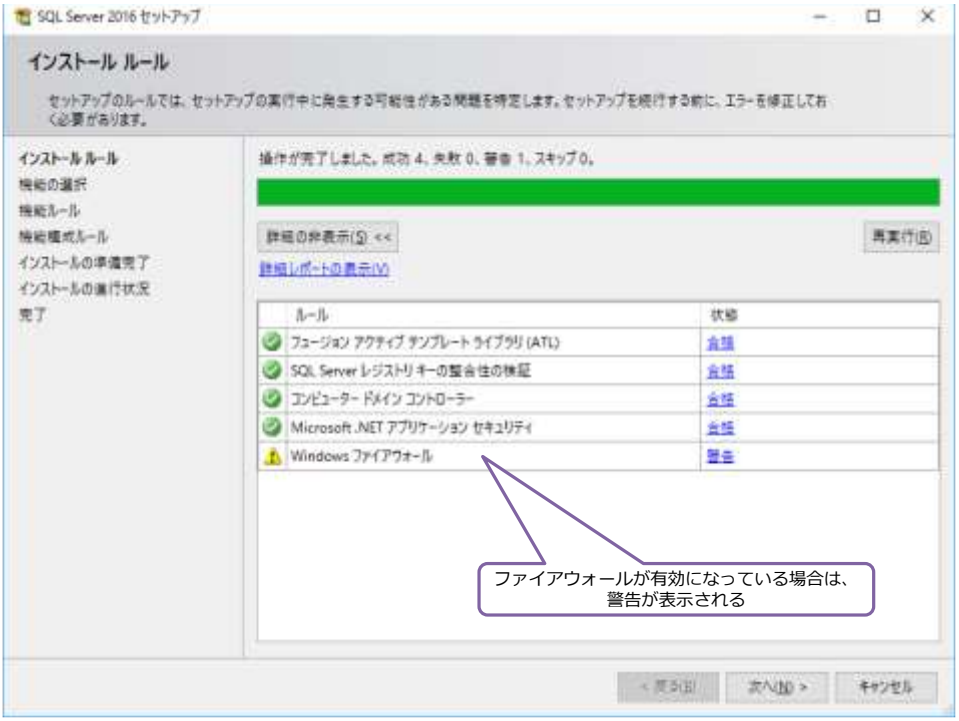



このページが表示される場合は、［**SQL Server 製品の更新プログラムを含める**］をチェックしておくことで、更新プログラムをダウンロードしてインストールすることができます。この更新プログラムは、インターネットからダウンロードすることになるので、インストールにかかる時間はその分長くなってしまいますが、インストールしておくことをお勧めします。

次の [**セットアップ ファイルのインストール**] ページでは、セットアップに必要なファイルがインストールされます。

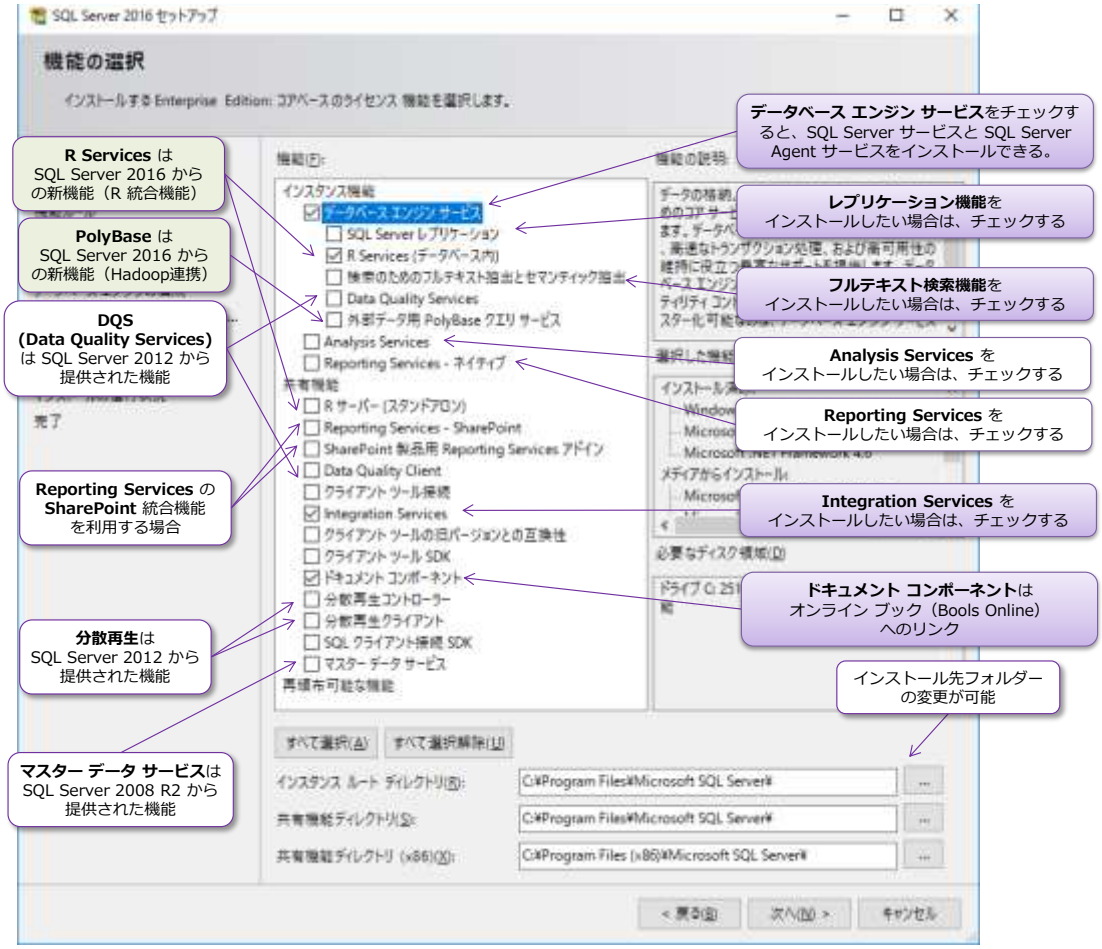



次の [**インストール ルール**] ページでは、SQL Server をインストールするにあたって、インストール要件を満たしているかどうかがチェックされます（問題がない場合は、自動的に次のページへ進み、問題がある場合（警告や失敗がある場合）にはそれらが提示されます）。



## インストールする機能の選択

次の［**機能の選択**］ページでは、次のように SQL Server 2016 のインストールしたい機能を選択します。




ここでは、「**データベース エンジン サービス**」をチェックすることで SQL Server サービスと SQL Server Agent サービスをインストールすることができ、「**SQL Server レプリケーション**」をチェックすればレプリケーション、「**検索のためのフルテキスト抽出とセマンティック抽出**」をチェックすればフルテキスト検索機能をインストールすることができます。SQL Server 2005 のときは、「**SQL Server データベース サービス**」を選択しておくと、自動的にレプリケーションとフルテキスト検索機能がインストールされていましたが、SQL Server 2008 以降では選択形式に変更されています。

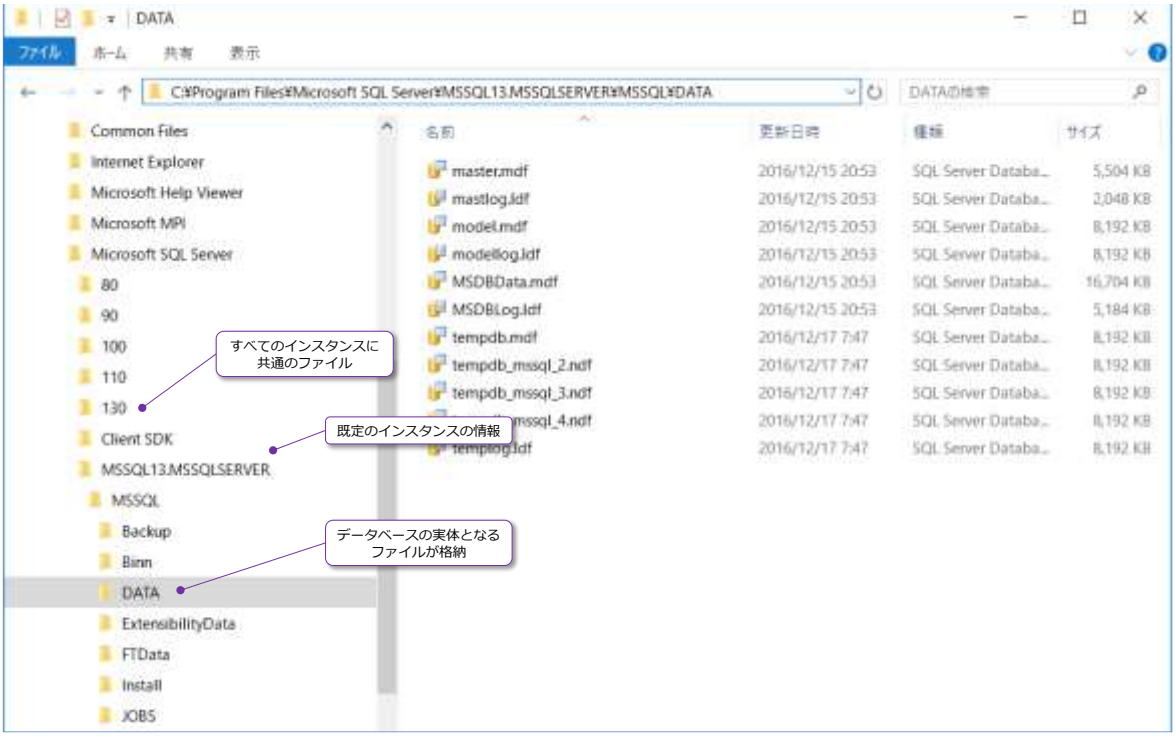
そのほかの機能の「**Integration Services**」や「**Analysis Services**」、「**Reporting Services**」などは、必要に応じてチェックして追加します。

「**ドキュメント コンポーネント**」は、SQL Server のヘルプである**オンライン ブック**（SQL Server 2005 までは **Books Online**：BOL と呼ばれていました）になりますが、SQL Server 2016 からは、これをチェックしてもローカル マシンにドキュメントがインストールされることはなくなりました。SQL Server 2016 では、オンライン ブックはインターネット上のコンテンツを参照する形になるので、そのコンテンツへのリンクをインストールする形になります。

SQL Server 2014 までは、この［**機能の選択**］ページで「**管理ツール**」を選択することで **Management Studio** や **SQL Server Profiler** などの管理ツールをインストールしていましたが、SQL Server 2016 からは管理ツールが別インストールに変わりました。管理ツールは、別途インターネット上からダウンロードして、インストールする必要があります。

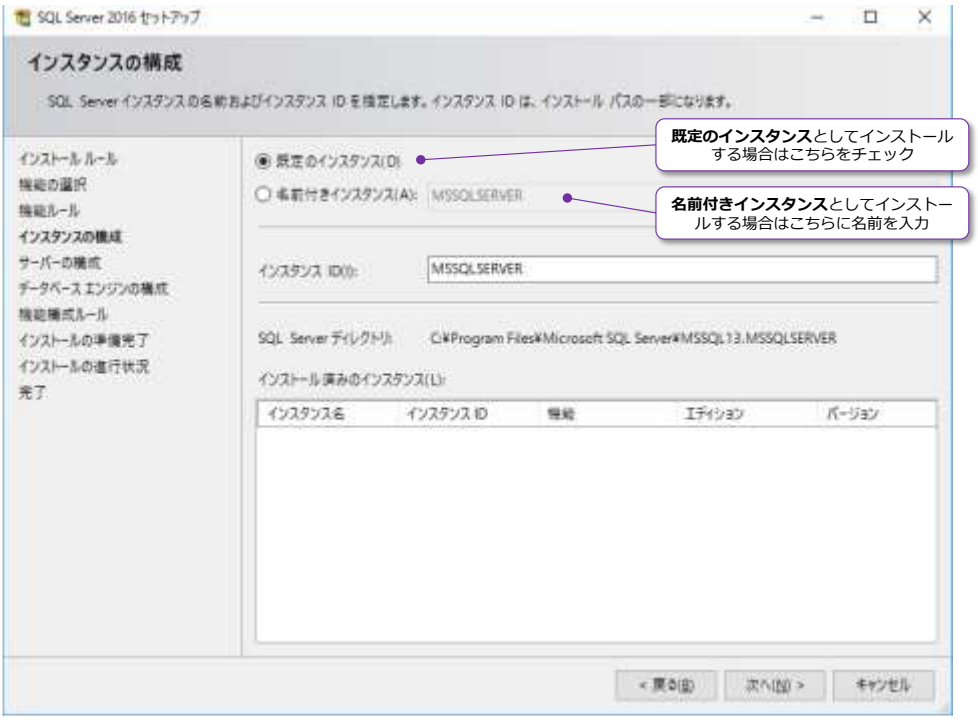
また、SQL Server 2005／2008／2008 R2 では、このページで BI 関連の開発ツールである「**Business Intelligence Development Studio**」（BIDS）、SQL Server 2012 では「**SQL Server Data Tools**」（SSDT）を選択してインストールすることができましたが、SSDT についても、SQL Server 2014 のときと同様、SQL Server 2016 では別インストールに変わりました。

［**機能の選択**］ページの［**インスタンス ルート ディレクトリ**］や［**共有機能ディレクトリ**］では、インストール先のフォルダーを変更することもできます。既定値は「**C:¥Program Files¥Microsoft SQL Server**」フォルダーに設定され、このフォルダーの下に次のようにインストールされます。



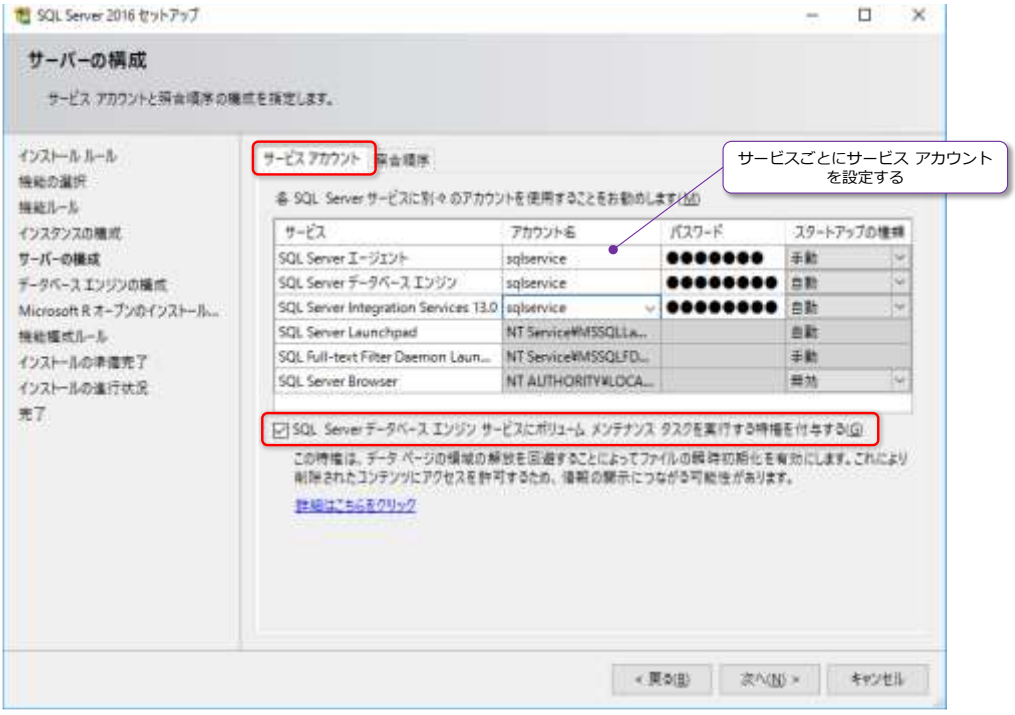

「**130**」フォルダーの下へ、各種ツールなど、すべてのインスタンスに共通のファイルが格納され、「**MSSQL13.MSSQLSERVER**」フォルダーの下へ、**既定のインスタンス**の情報（**DATA** や **LOG** フォルダー）、**名前付きインスタンス**としてインストールした場合は、「**MSSQL13.インスタンス名**」のフォルダーの下に格納されます。なお、**システム データベース**（master や msdb）が格納される **DATA** フォルダーに関しては、後述の「**データベース エンジンの構成**」ページから変更することができます。

次の［**インスタンスの構成**］ページでは、既定のインスタンスとしてインストールするか、名前付きインスタンスとしてインストールするかを設定します。




## サービス アカウントの設定

インストール ウィザードの［**サーバーの構成**］ページの［**サービス アカウント**］タブでは、サービス アカウントを設定します。

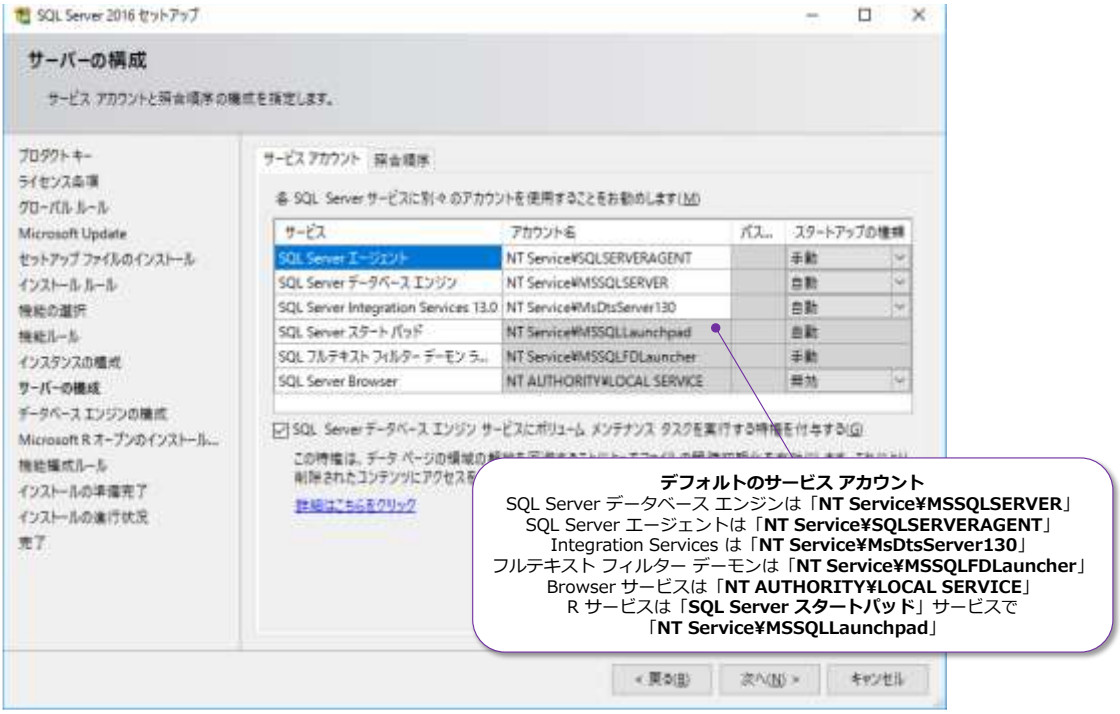



以前の SQL Server では、すべてのサービスに対して、まとめて同じサービス アカウントを設定することができましたが、SQL Server 2016 では、サービスごとにサービス アカウントを指定する必要があります。フルテキスト フィルター デーモン サービスに関しては **NT Service¥MSSQLFDLauncher**、Browser サービスに関しては、**NT AUTHORITY¥LOCAL SERVICE** とい

う内部アカウントが自動設定されるようになったので、サービス アカウントを設定する必要はなくなりました。なお、R サービスをインストールする場合は、**NT Service¥MSSQLLaunchpad** という名前の内部アカウントが自動設定されます。

このページでは、[**SQL Server データベース エンジン サービスにボリューム メンテナンス タスクを実行する権利を付与する**] をチェックすることで、インストール時に、サービス アカウントに対して、ユーザーの権利である「**ボリュームの保守タスクを実行**」を付与できるようになりました（SQL Server 2016 からの新機能）。この権利は、**瞬時初期化**（.mdf ファイルのサイズが拡張するときに、瞬間的にサイズを拡張できる機能）を有効化するために、以前のバージョンでは、手動で行っている作業でした（サービス アカウントが Administrators グループのメンバーである場合には、この権利は自動的に付与されています）。

なお、サービス アカウントを指定しない場合は、次のように内部アカウントが自動設定されます。



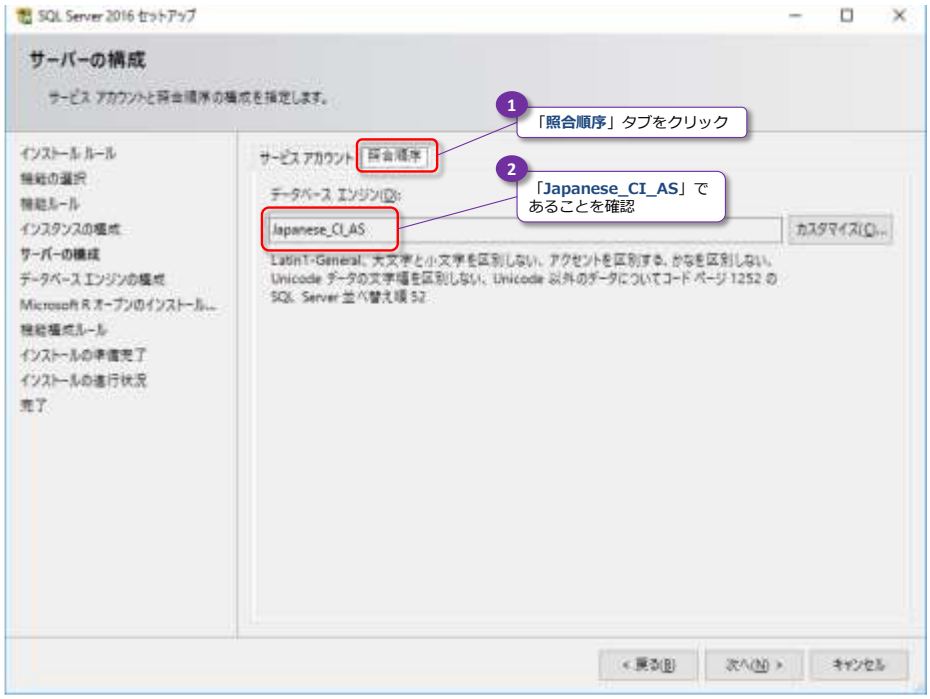

SQL Server データベース エンジンに対しては「**NT Service¥MSSQLSERVER**」、SQL Server エージェントには「**NT Service¥SQLSERVERAGENT**」、Integration Services には「**NT Service¥MsDtsServer130**」アカウント（ローカルの内部アカウント）が設定されます。

サーバーをまたがった操作（バックアップを UNC 経由で行ったり、データベース ミラーリングやレプリケーション、ログ配布、可用性グループなどのように複数の SQL Server と連携する機能を利用したりするなど）を行う場合には、これらの内部アカウントをドメイン ユーザーなどの Windows アカウントへ変更しておかないと利用することができないので、移行元と同じように設定するようにします。

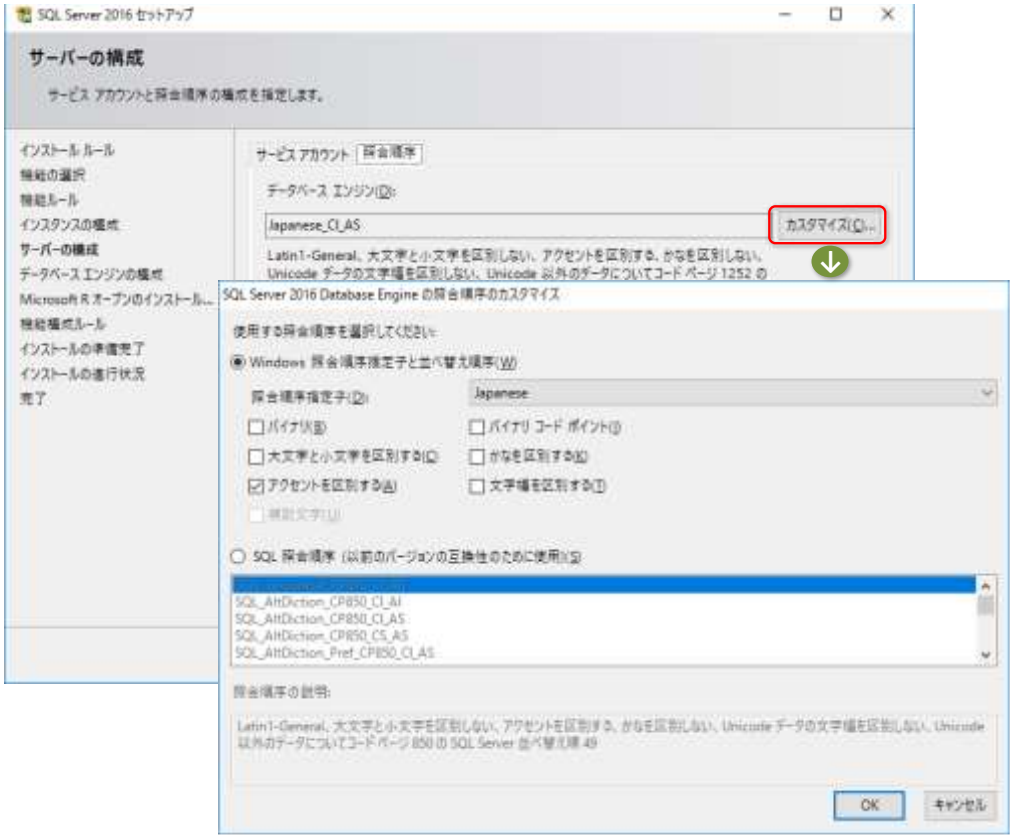
## 照合順序の設定（以前の SQL Server と同じ Japanese_CI_AS が既定値）

［**サーバーの構成**］ページでは、次のように［**照合順序**］タブで照合順序を設定することができま

す（既定値は、過去の SQL Server と同じ **Japanese_CI_AS** になっています）。



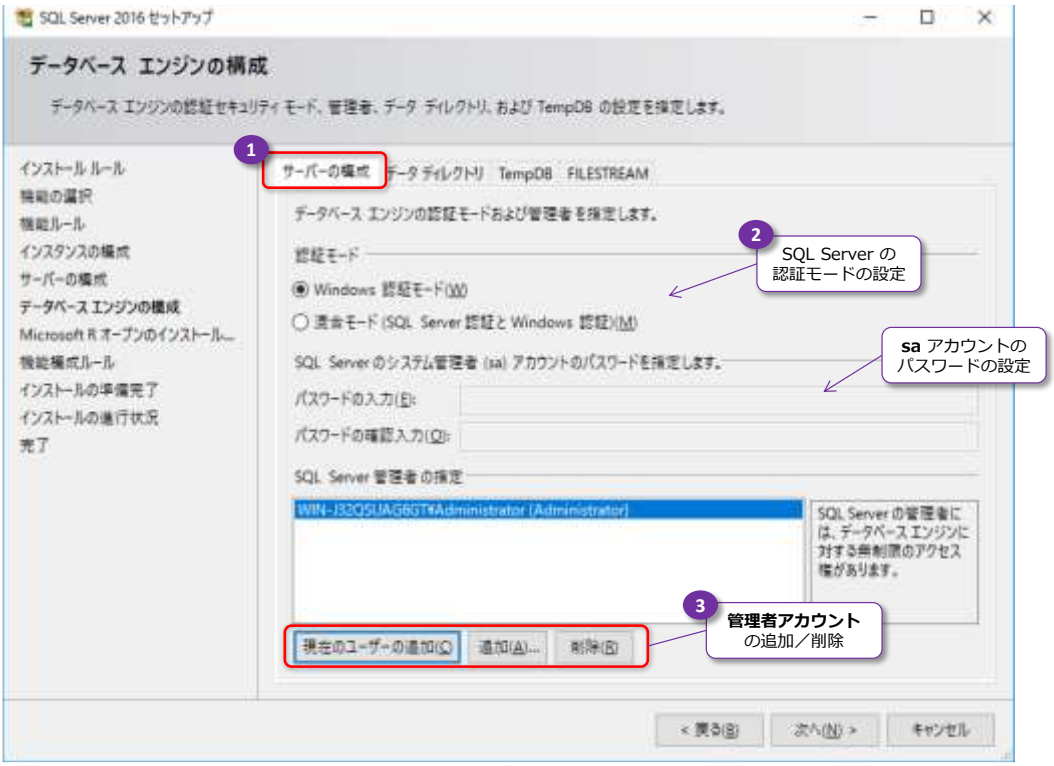

照合順序を変更したい場合には、次のように［**カスタマイズ**］ボタンをクリックします。




なお、ここで設定するサーバーの照合順序は、旧システム環境（移行元の SQL Server）と同じ照合順序を利用するようにします。サーバーの照合順序は、tempdb データベース内に作成されるオブジェクトや、レプリケーション、データベース ミラーリングなどのサーバー間の連携機能への影響があるので、旧システム環境と同じものへ設定することが重要になります。

## 認証モードの設定

次の［**データベース エンジンの構成**］ページでは、［**サーバーの構成**］タブで SQL Server の認証モードと管理者アカウントを設定します。



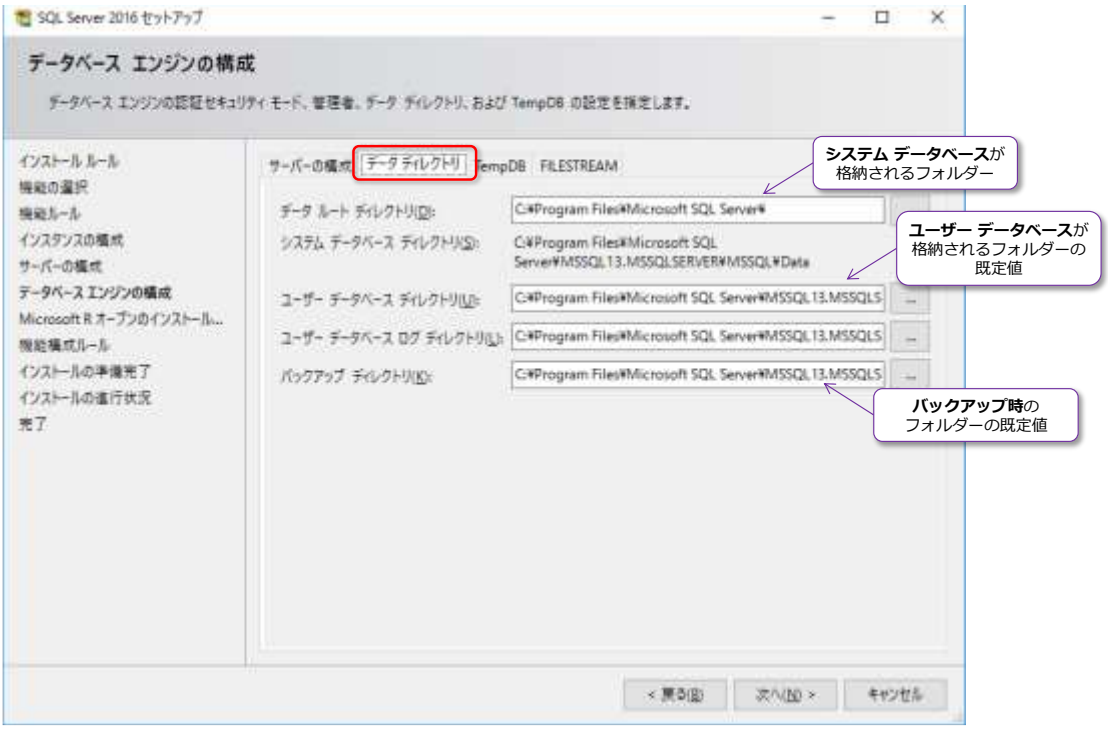

ここでは、旧システム環境（移行元の SQL Server）と同じ認証モードを選択し、混合モードを利用する場合には、ビルトインの SQL Server のシステム管理者アカウントである「**sa**」のパスワードを設定します。

SQL Server 2005 のときには、インストールの完了後に、**BUILTIN¥Administrators**（ローカルの管理者グループ）が管理者アカウント（**sysadmin** ロールのメンバー）として自動設定されていましたが、SQL Server 2008 以降では自動設定されなくなりました。このページで追加したユーザーのみが管理者アカウント（sysadmin ロールのメンバー）として設定されることになります（認証モードで**混合モード**を選択している場合には、**sa** も管理者アカウントとして設定されます）。

ここでは、［**現在のユーザーの追加**］ボタンをクリックすることで、インストールを行っているユーザーを管理者アカウントに設定することができます。

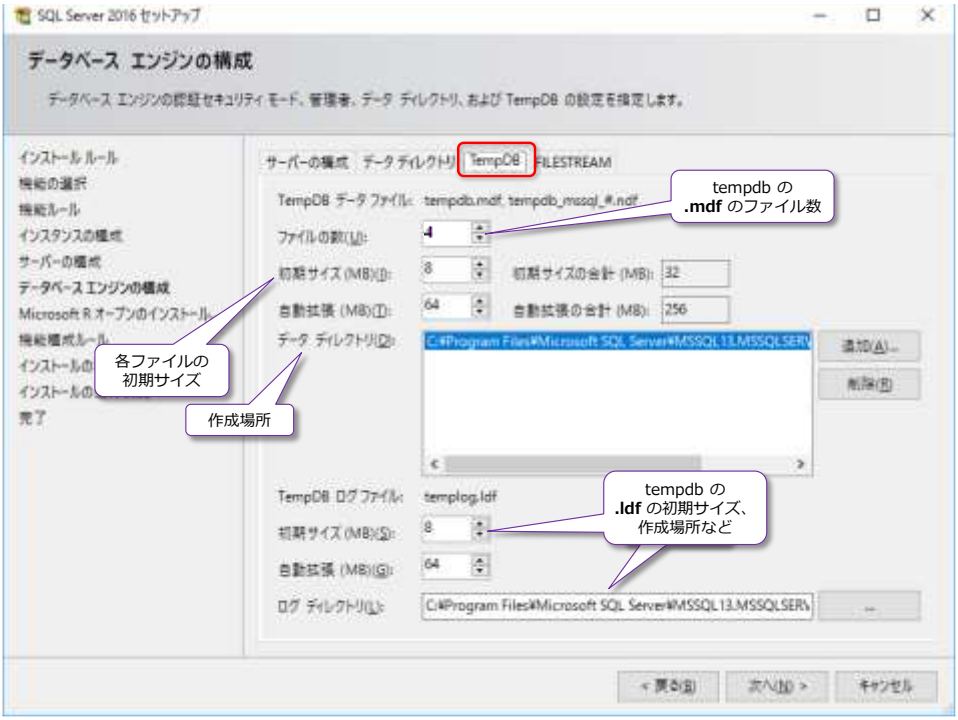
## データ ディレクトリ（DATA フォルダー）の設定

［**データベース エンジンの構成**］ページでは、［**データ ディレクトリ**］タブで、**システム データベース**が格納される **DATA** フォルダーの場所を設定することができます。




このタブでは、ユーザー データベースの既定のディレクトリ（データベースの新規作成時の既定のフォルダー）、バックアップ時の既定のディレクトリも変更することができます。

SQL Server 2016 からは、次のように［**TempDB**］ページを開くことで、**tempdb データベースの設定**も変更できるようになりました（今までの SQL Server では、tempdb の設定変更は、インストールが完了した後に、別途行う必要がありましたが、SQL Server 2016 からはインストール時に設定変更できるようになりました）。

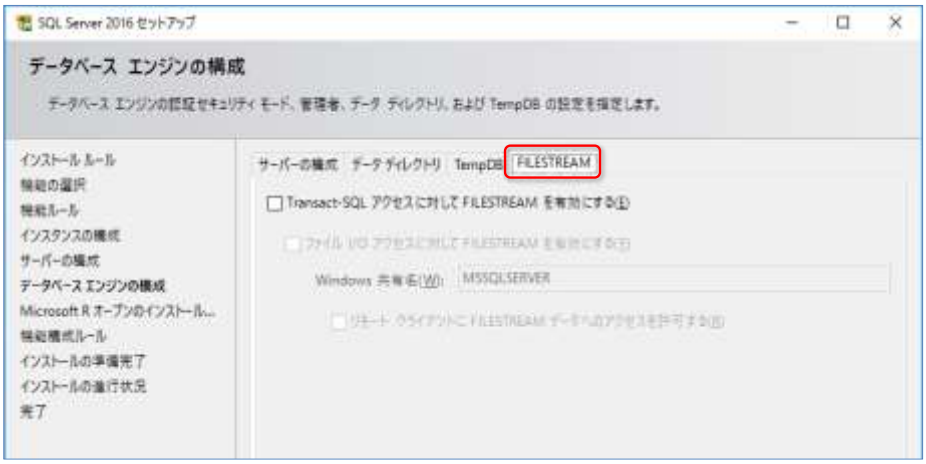



**.mdf**（データ ファイル）に関しては、ファイル数や初期サイズ、作成場所、**.ldf**（ログ ファイル）に関しては、初期サイズ、作成場所などを細かく設定できるようになりました。なお、.mdf のフ

ァイル数は、既定では搭載されている CPU のスレッド数または 8 スレッド以上の場合は 8 に設定されます（画面は 4 スレッドの場合の例）。また、既定の作成場所は、前のページで設定した SQL Server のインストール先の **DATA** フォルダーと同じになります。

**Note： FILESTREAM の有効化**

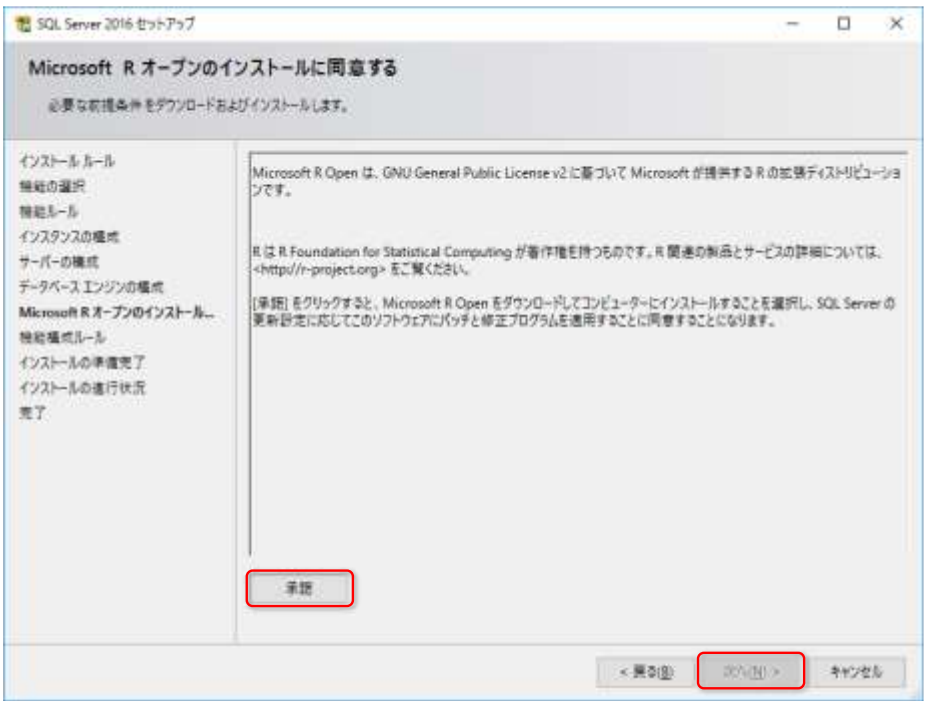
［**データベース エンジンの構成**］ページでは、［**FILESTREAM**］タブで FILESTREAM 機能を有効にするかどうかを設定することができます。FILESTREAM は、SQL Server 2008 から提供された新機能で、ファイルをデータベース内へ格納できる機能です。




FILESTREAM 機能の有効化は、インストール後にも簡単に設定できるので、ここでは無効のままで大丈夫です。

**Note： R サービス（SQL Server R Services）をインストールする場合**

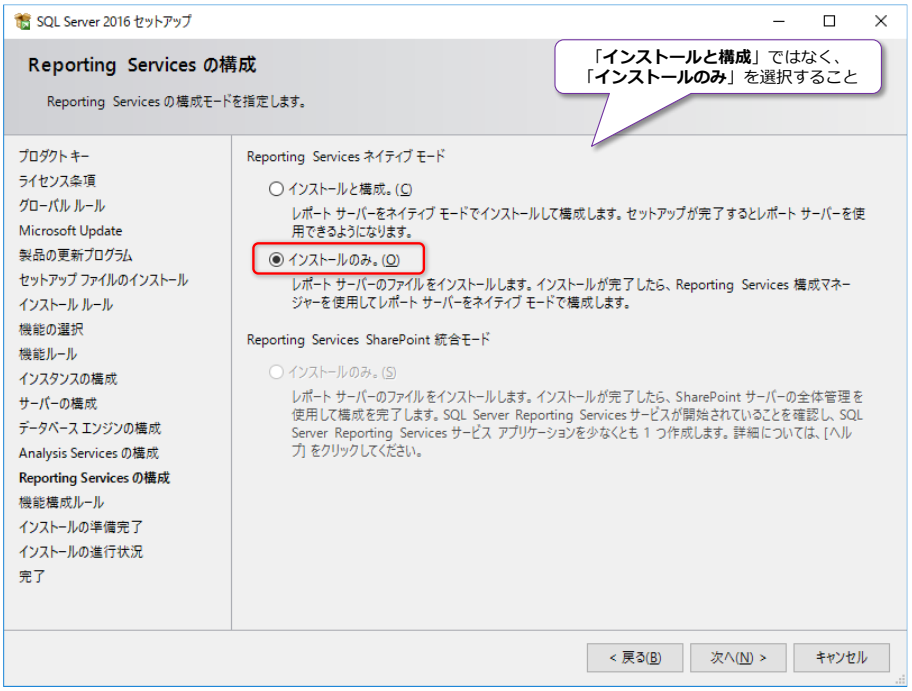
［**機能の選択**］ページで［**R サービス（データベース内）**］を選択して、**SQL Server R Services** をインストールする場合には、次のように［**Microsoft R オープンのインストールに同意する**］ページが表示されます。




R サービスは、上記の内容に［**承諾**］をすることで利用できるようになります。

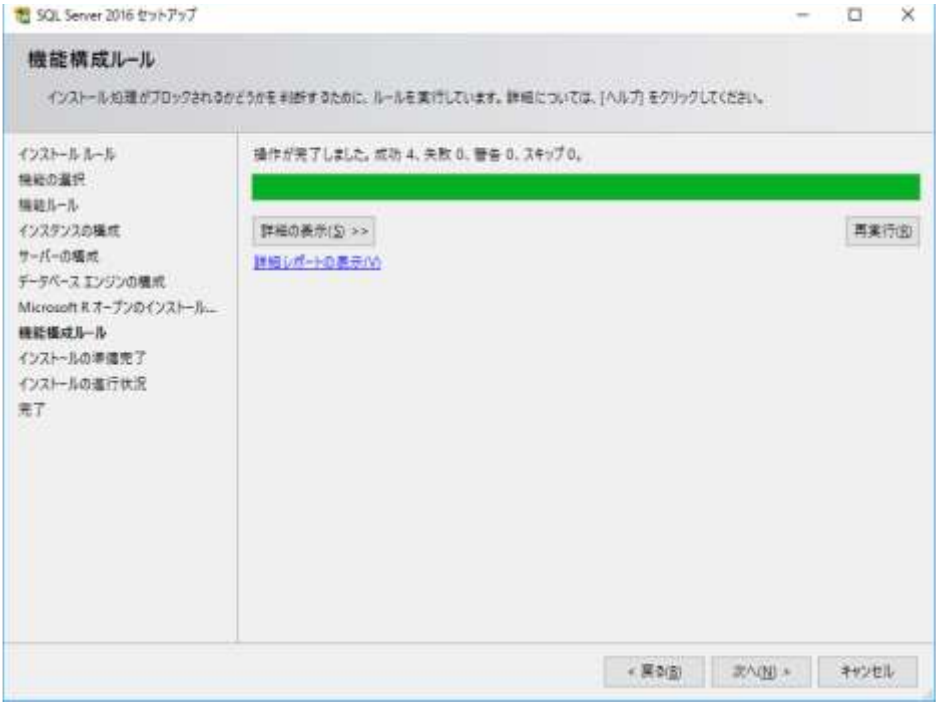
**Note： Reporting Services をインストールする場合**

［**機能の選択**］ページで［**Reporting Services - ネイティブ**］を選択して、**Reporting Services** をインストールする場合には、次のように［**Reporting Services の構成**］ページが表示されます。



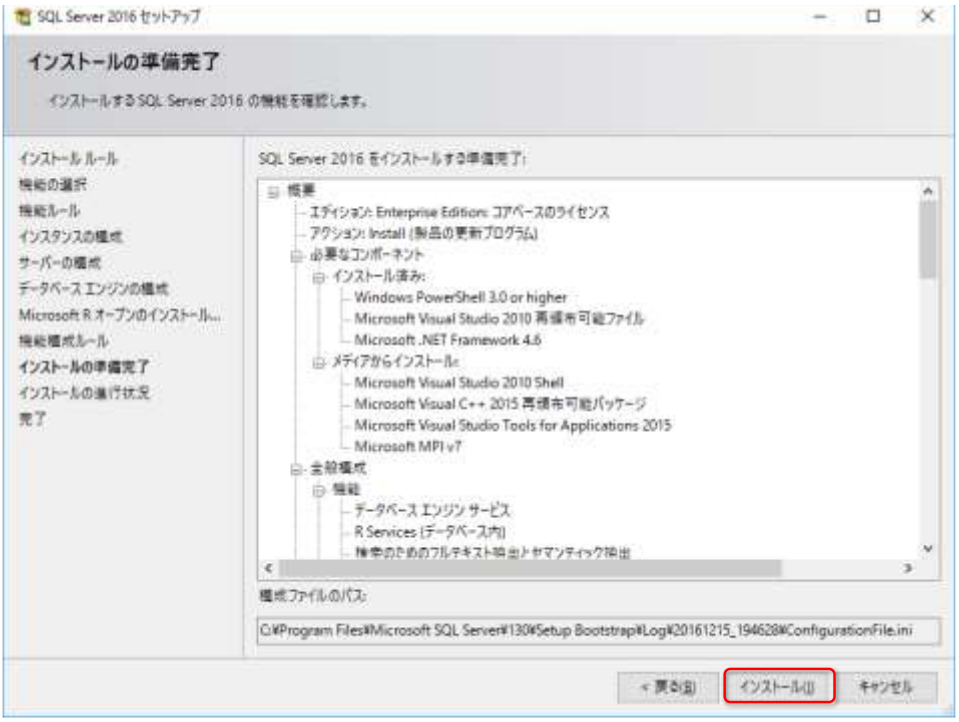

詳しくは、章の後半で説明しますが、Reporting Services を移行するには、このページで「**インストールのみ**」を選択しておく必要があります。既定は「**インストールと構成**」なので、間違えないようにしてください。

次の［**機能構成ルール**］ページでは、インストールにあたっての条件を満たしているかどうかの最終チェックが行われますが、問題がない場合は、自動的に次のページへ進みます。

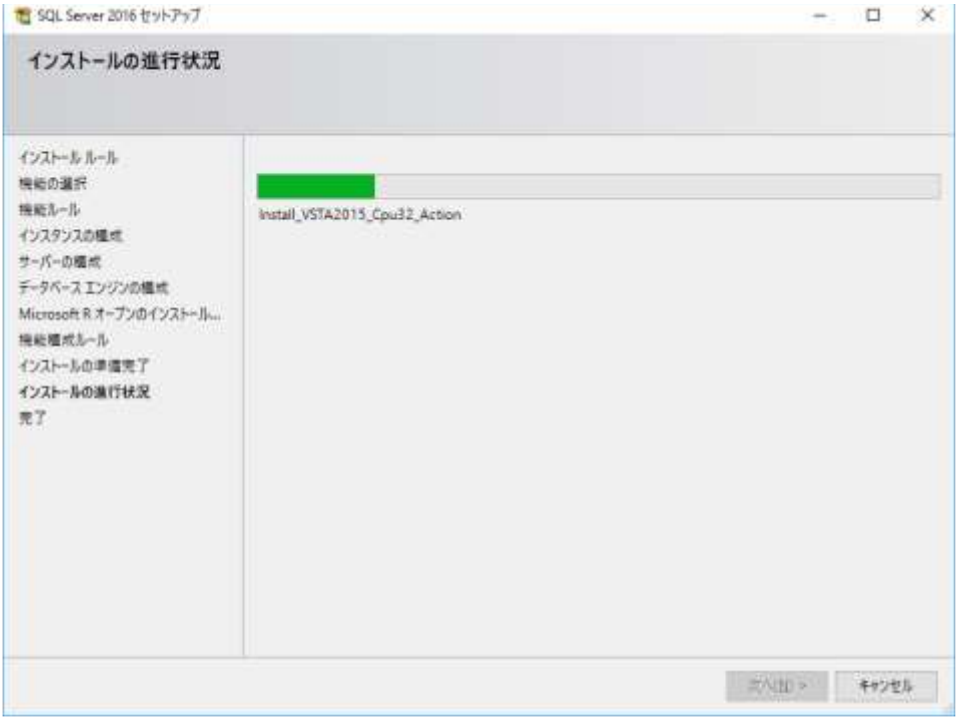



次の［**インストールの準備完了**］ページでは、設定した内容を確認して、［**インストール**］ボタンを

クリックすれば、インストールが開始されます。

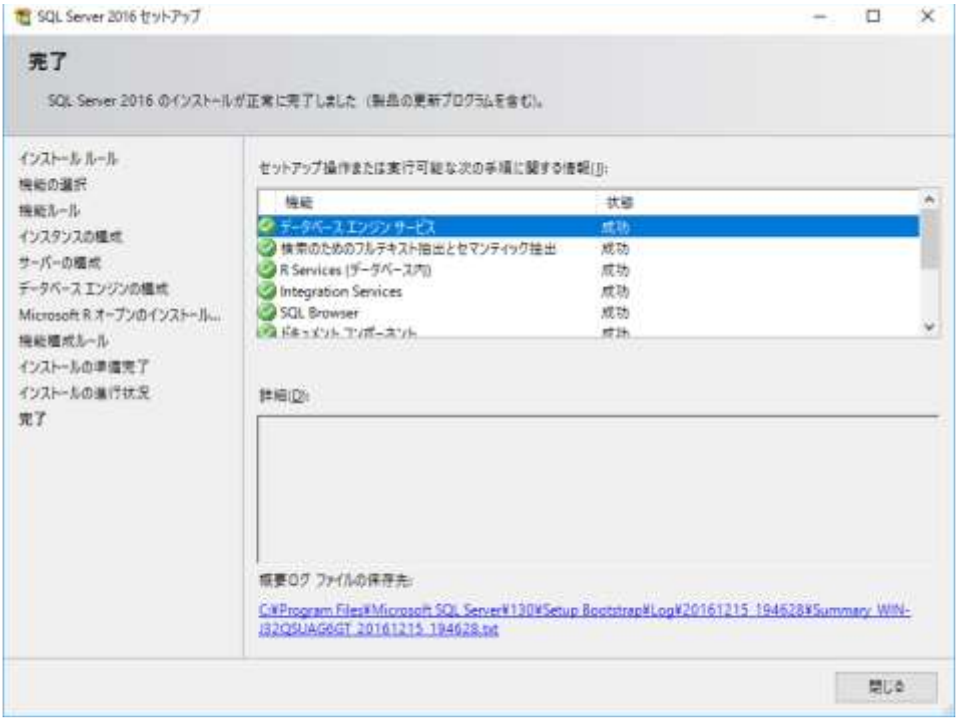



インストール中は、次のように進行状況が表示されます。



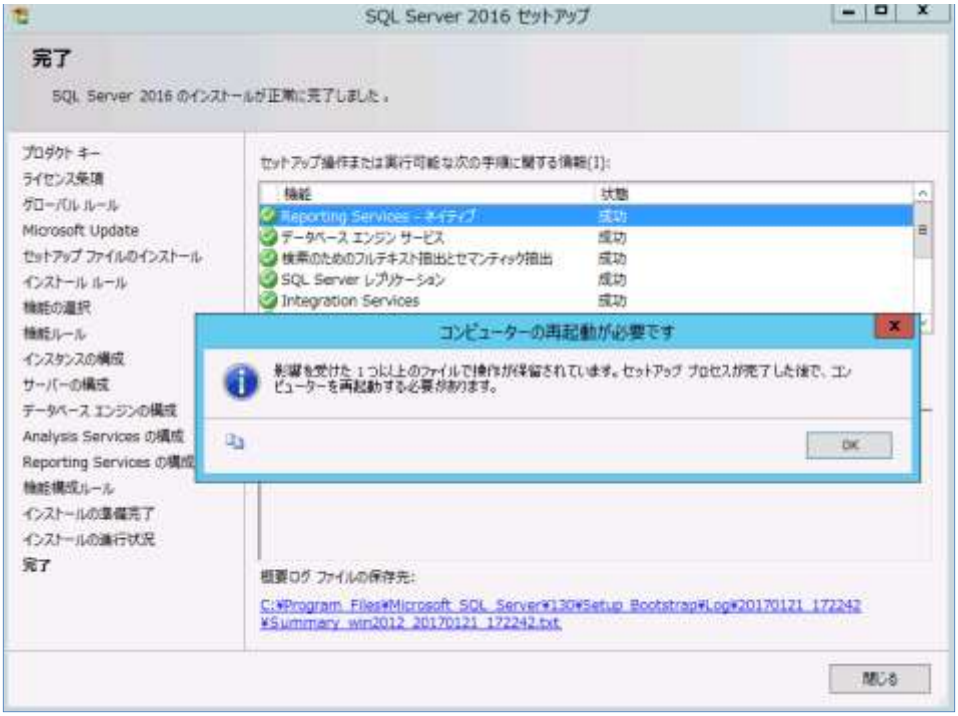

インストールにかかる時間は、選択した機能の種類や、ハードウェア環境（SSD などの高速ストレージかどうか）などによって、大きく前後しますが、15 分～1 時間 30 分程度で完了します。

インストールが完了すると、次のように［**完了**］ページが表示されます。




以上が SQL Server 2016 をインストールする手順です。

OS に Windows Server 2012 や 2012 R2 を利用している場合は、インストールの完了に **OS の再起動**が必要になる場合があり、その場合は、次のように表示されます。



**Note： SQL Server 2005 からの変更点**

SQL Server 2016 のインストールに関して、SQL Server 2005 のときのインストールと比較すると、次の変更点があります。

・管理者アカウントを指定する必要があること
（**Administrators** グループのメンバーが自動的に管理者にならなくなった）
・**サービス アカウント**の既定値が内部アカウントに変更された
・管理ツール（Management Studio や SQL Server Profiler）は、
別途インストールする形に変更された
・BI 関連の開発ツールである **BIDS**（Business Intelligence Development Studio）は、
別途インストールする形に変更された
（SQL Server 2012 から、名称が SSDT : SQL Server Data Tools に変更された）
・**Reporting Services** を利用する場合に IIS と ASP.NET が必須ではなくなった
（SQL Server 2008 からは Reporting Services 自体が **http.sys** を実装するように
なったので、IIS が必須コンポーネントではなくなった）
・**オンライン ブック**（Books Online）がローカル マシンにインストールされなくなった

## 5.4 SQL Server 2016 の修正プログラムのインストール

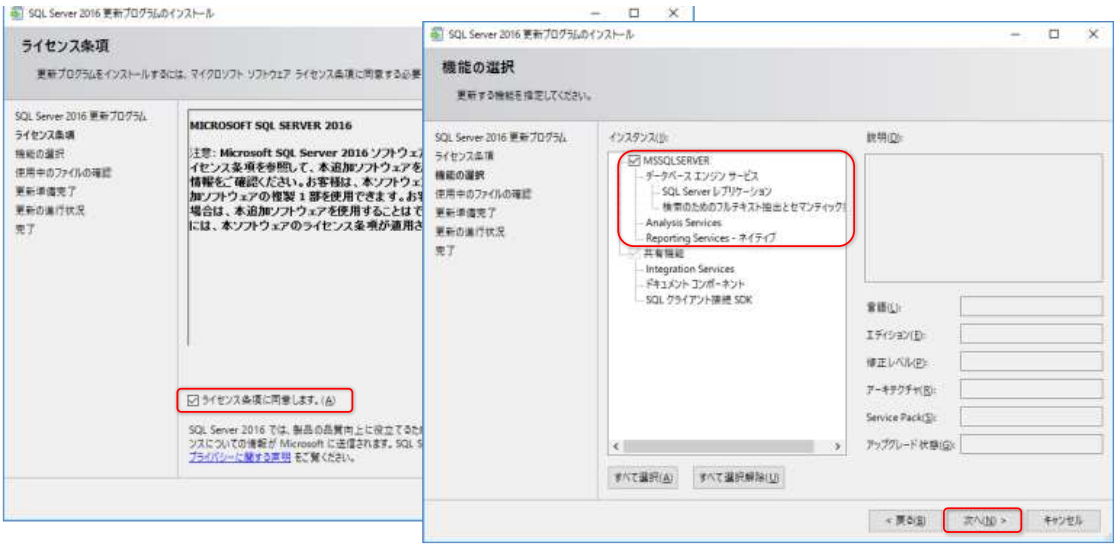
**SQL Server 2016** の製品出荷後には、**修正プログラム**がリリースされているので、これもインストールしておくことをお勧めします。**CU2**（累積的な更新プログラム２）には、性能向上に関する修正が入っているので、できる限り最新の修正プログラムを適用しておくことをお勧めします。また、**Reporting Services** を利用している場合は、SP1 に対する重要な更新プログラム（KB 3207512）が提供されているので、これを含んだ **CU4** または **SP1＋CU1** 以上を適用しておくことをお勧めします。

なお、本ドキュメントの執筆時点（2017 年 1 月）では、**CU4** と **SP1**（Service Pack 1）＋**SP1 に対する CU1** が最新の修正プログラムですが、今後 **SP2** や **CU5** など、新しい修正プログラムが提供された場合には、それらをインストールしておくことをお勧めします。

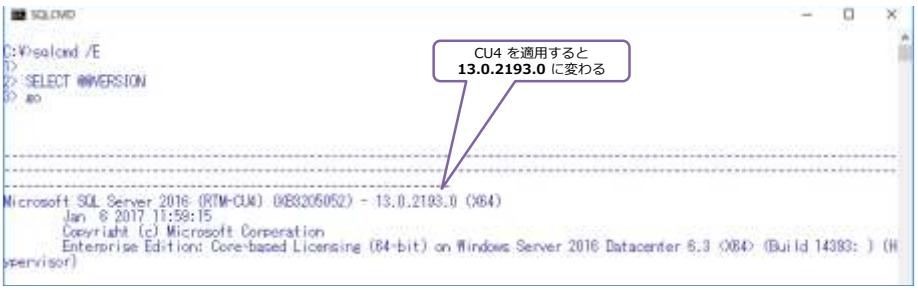
### SQL Server 2016 CU4 のダウンロード／インストール

**SQL Server 2016** の **CU4**（KB 3205052）は、次の URL からダウンロードできます。

Microsoft SQL Server 2016 RTM Latest Cumulative Update
https://www.microsoft.com/en-us/download/details.aspx?id=53338




CU4 のインストールに関しては、**第３章の 3.6** で詳しく説明しているので、こちらもご覧いただければと思います。CU4 のインストールが完了すると、SQL Server のビルド番号は「**13.0.2193.0**」に変わります。

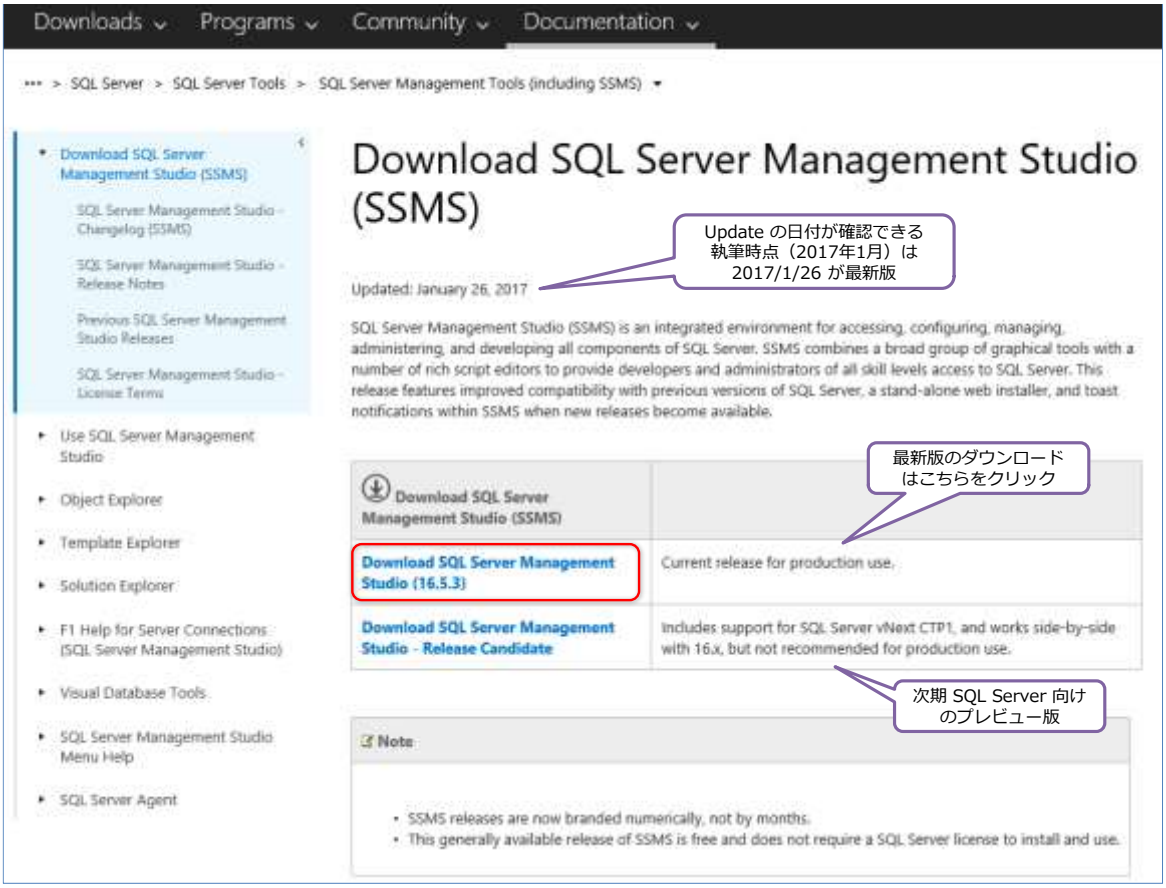


## 5.5 最新版の Management Studio のダウンロード／インストール

SQL Server 2016 からは、Management Studio（SQL Server の管理ツール）がダウンロード版の提供のみに変更されました。

最新版の Management Studio は、次の URL からダウンロードすることができます。

Management Studio の最新版のダウンロード
http://msdn.microsoft.com/en-us/library/mt238290.aspx




インストール方法に関しては、**第３章の 3.7** で詳しく説明しているので、こちらもご覧いただければと思います。

## 5.6 データベースの移行 ～バックアップと復元機能を利用～

新規サーバーへの SQL Server 2016 のインストールと修正プログラムのインストールが完了した後は、旧システム環境（移行元の SQL Server）のデータベースを SQL Server 2016 へ移行します。データベースの移行は、**SQL Server 標準のバックアップと復元機能**を利用することで簡単に行うことができます。

### 移行可能なデータベース

**SQL Server 2016** への**移行**が可能なデータベース（バックアップと復元機能で移行できるデータベース）は、次のとおりです。

- **SQL Server 2005** 上のデータベース
- **SQL Server 2008** 上のデータベース
- **SQL Server 2008 R2** 上のデータベース
- **SQL Server 2012** 上のデータベース
- **SQL Server 2014** 上のデータベース

3 章と 4 章で紹介したインプレース アップグレードは SQL Server 2008 以上が対象でしたが、ここで紹介するバックアップと復元機能では、**SQL Server 2005** のデータベースを SQL Server 2016 に移行することもできます。ただし、**SQL Server 2005** のデータベースを復元した場合には、**データベースの互換性レベル**は **100**（SQL Server 2008 レベル）に自動的に上がります。

**SQL Server 2008／2008 R2** のデータベースの場合は、互換性レベルは **100** がキープされて、**SQL Server 2012** のデータベースの場合は **110**、**SQL Server 2014** のデータベースの場合は **120** がキープされます（互換性レベルについては後述します）。

### データベースの移行の概要

データベースの移行は、次のように行います。

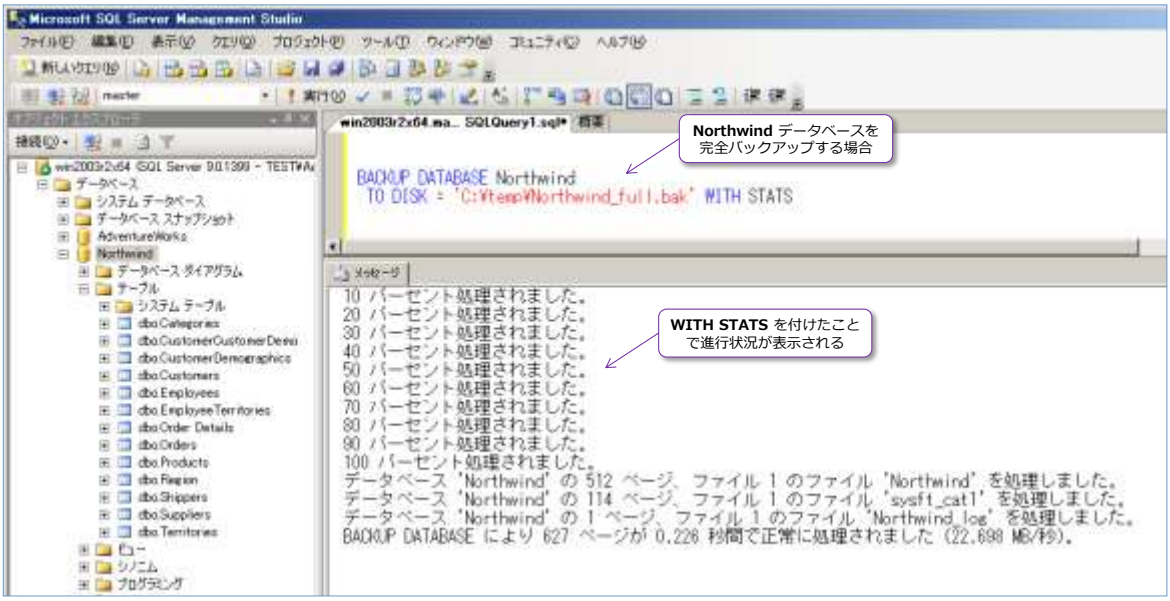
1. 移行元の SQL Server でデータベースの完全バックアップを取得する
2. バックアップしたファイルを移行先サーバーへコピーする
3. 移行先サーバー（SQL Server 2016）でバックアップから復元する

### 1. 移行元の SQL Server でデータベースの完全バックアップを取得する

まずは、旧システム環境（移行元の SQL Server）でデータベースの完全バックアップを取得しま

す。完全バックアップを取得するには、次のように **BACKUP DATABASE** ステートメントを実行します。

```
BACKUP DATABASE データベース名
 TO DISK = 'ファイルパス' WITH STATS
```




上の画面は、**SQL Server 2005** の場合ですが、**SQL Server 2008** や **2008 R2**、**2012**、**2014** でも同じように実行できます。画面では、移行対象のデータベースに「**Northwind**」、バックアップ先となるファイル パスには「**C:¥temp¥Northwind_full.bak**」を指定しています。

バックアップ時の **WITH STATS** オプションは、必須ではないのですが、バックアップの進行状況（**～パーセント処理されました**）を表示できるようになるので、付けておくことをお勧めします。

バックアップにかかる時間は、データベースのサイズ（使用領域）や、バックアップ先となるストレージの性能によって大きく変わってきますが、実行時間の見積もり方法については、**第４章の 4.11** で詳しく説明しているので、こちらもご覧いただければと思います。

**Note： SQL Server 2008 以降ならバックアップ圧縮を利用するのがお勧め**

SQL Server 2008 Enterprise または SQL Server 2008 R2 Standard エディション以上を利用している場合には、**バックアップ圧縮**（バックアップ時のファイル圧縮）機能を利用することをお勧めします。

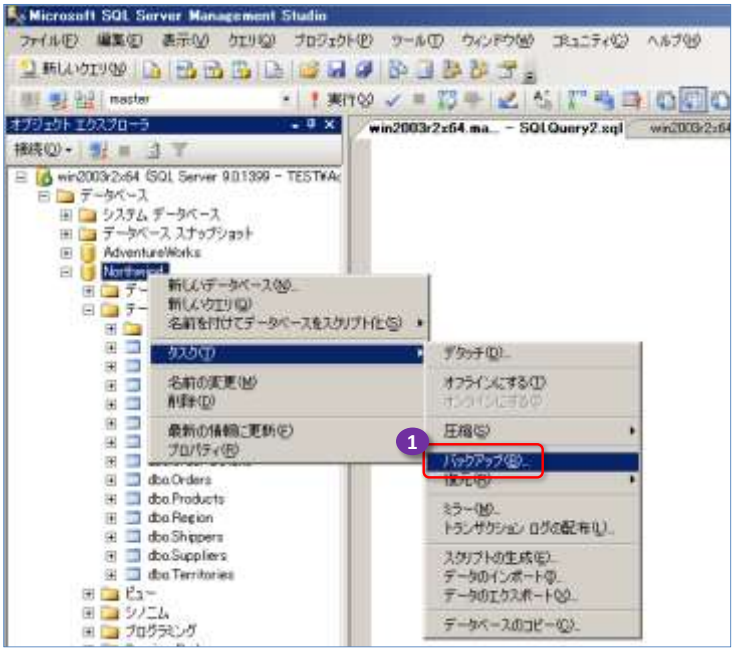
バックアップ圧縮は、次のように **WITH COMPRESSION** を付けるだけで実行できます。

```
BACKUP DATABASE Northwind
 TO DISK = 'C:¥temp¥Northwind_full.bak' WITH COMPRESSION, STATS
```

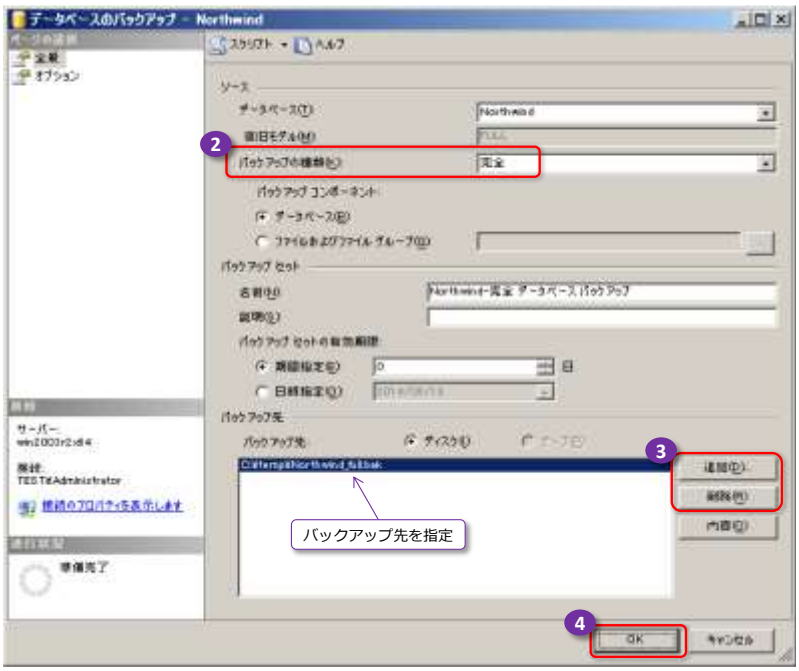
このように、バックアップ圧縮を利用することで、バックアップ ファイルのサイズを小さくすることができるので、バックアップ時間の短縮や、移行先へのファイル コピー時間の短縮、リストア時間の短縮など、実際の移行時のダウンタイムを小さくできるようになるのでお勧めです。

**Note： GUI 操作で完全バックアップを取得したい場合**

Management Studio を利用して、GUI 操作で完全バックアップを取得したい場合には、次のようにオブジェクト エクスプローラーで、該当データベースを右クリックして、[**タスク**] メニューの [**バックアップ**] をクリックします（画面は、SQL Server 2005 の場合ですが、SQL Server 2008／2008 R2／2012／2014 でも同じように操作できます）。




次のように [**データベースのバックアップ**] ダイアログが表示されたら、[**バックアップの種類**] で「**完全**」を選択します。



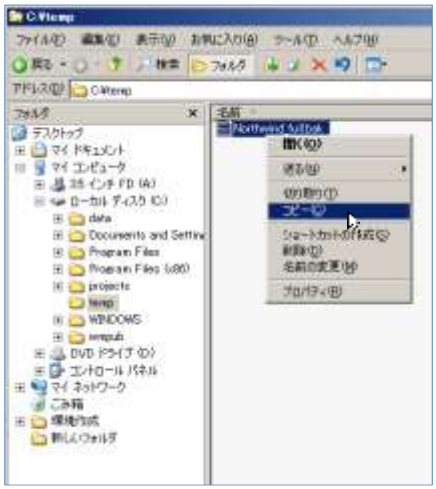

[**追加／削除**] ボタンを利用して、バックアップ先となるファイルを設定し、[**OK**] ボタンをクリックすれば、バックアップが開始されます。

バックアップが正常に完了すると、次のようにダイアログが表示されます。

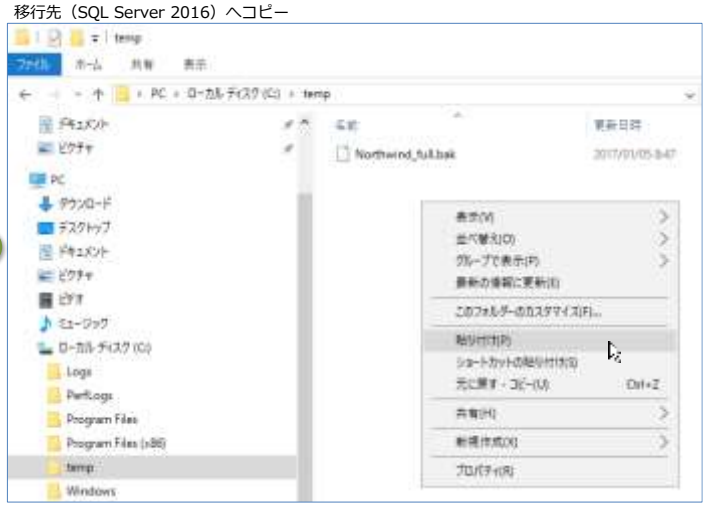

## 2. 移行元で取得した完全バックアップを移行先サーバーへコピー

次に、移行元で取得した完全バックアップのファイル（**.bak**）を移行先のサーバー（**SQL Server 2016**）へコピーします。この操作は、Windows エクスプローラーから行います。

バックアップしたファイルをコピー（移行元）

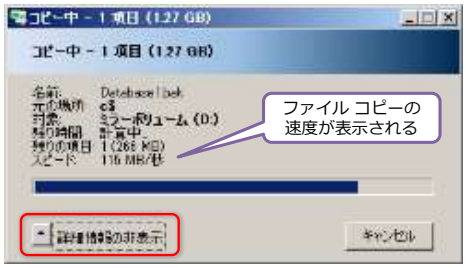


移行先（SQL Server 2016）へコピー



ファイルのコピーの中は、次のように［**詳細情報**］をクリックすることで、**ファイルのコピー速度**（**～MB/s**：1 秒あたりに何 MB を転送できたのか）を確認することができるので、確認しておくことをお勧めします（ファイルのコピーにかかる時間は、移行時のダウンタイム時間に関わってくる重要なものなので、事前に転送速度をチェックしておくことをお勧めします。**第 4 章 4.11** 参照）。

Windows Server 2008／2008 R2 の場合

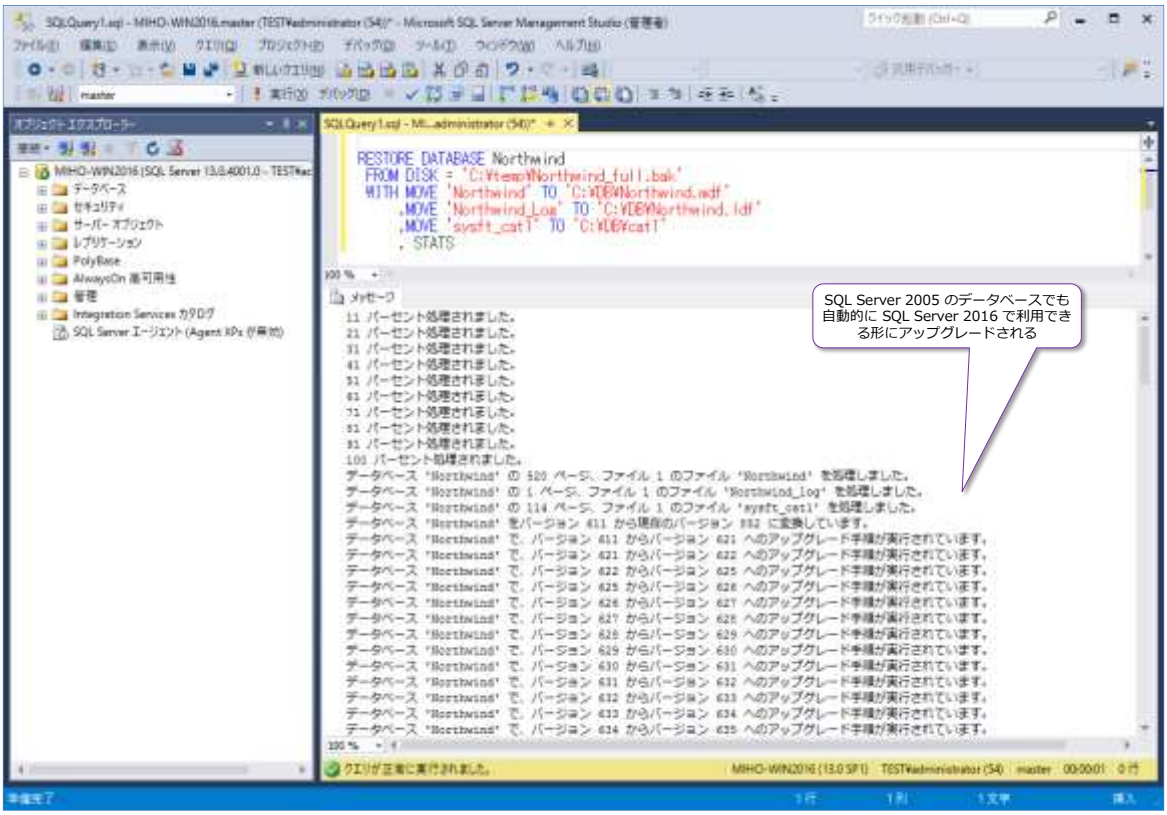



Windows Server 2012／2012 R2 の場合


## 3. 移行先サーバー（SQL Server 2016）でバックアップを復元する

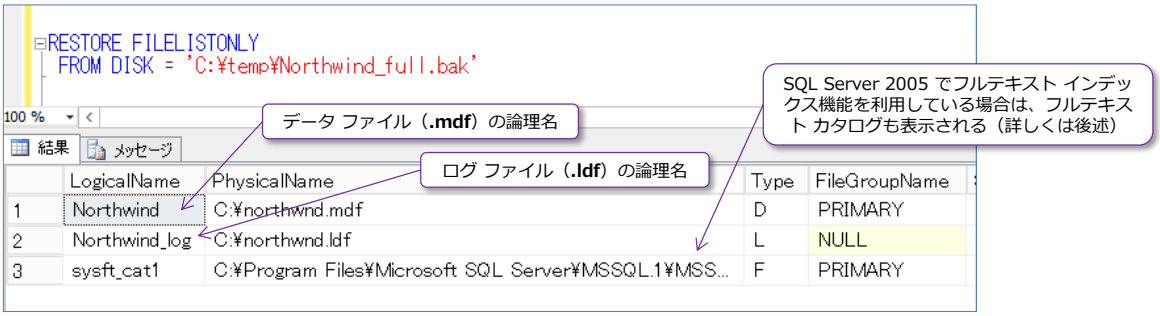
次に、手順 2 でコピーしたバックアップ ファイルを**移行先サーバー**（**SQL Server 2016**）上で復元（リストア）します。復元を行うには、次のように **RESTORE DATABASE** ステートメントを実行します（Northwind データベースを復元する場合）。

```
RESTORE DATABASE Northwind
 FROM DISK = 'C:¥temp¥Northwind_full.bak'
 WITH MOVE 'Northwind' TO 'C:¥DB¥Northwind.mdf'
     ,MOVE 'Northwind_Log' TO 'C:¥DB¥Northwind.ldf', STATS
```



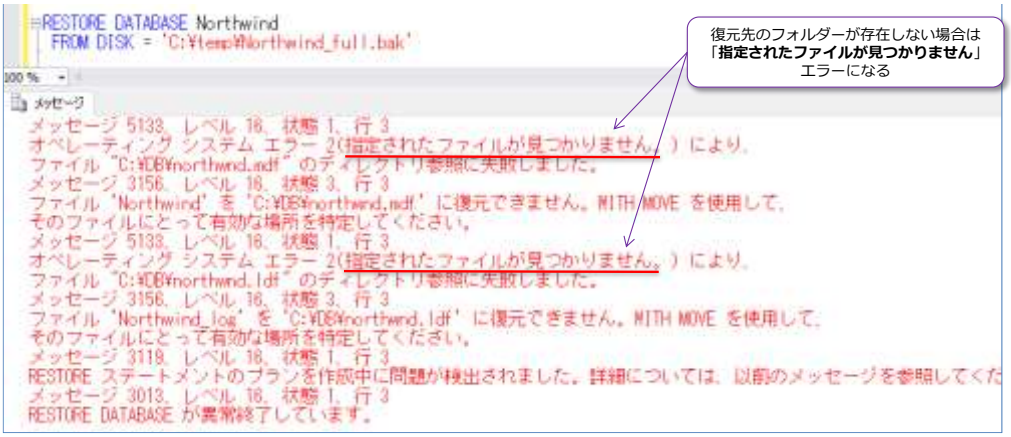

**Northwind** の部分は移行対象のデータベース名、**C:¥temp¥Northwind_full.bak** の部分は任意のファイル パスへ変更して実行するようにします。**MOVE .. TO** では、「**MOVE '論理名' TO '復元先のパス'**」のように復元先のパスをデータ ファイル（**.mdf**）とトランザクション ログ ファイル（**.ldf**）ごとに指定します。各ファイルの**論理名**（LogicalName）は、次のように **RESTORE FILELISTONLY** ステートメントを実行して確認することができます。

```
RESTORE FILELISTONLY
 FROM DISK = 'C:¥temp¥Northwind_full.bak'
```




**LogicalName** で表示された名前（画面は **Northwind** や **Northwind_log**）を **MOVE .. TO** の **..** へ指定します。なお、**RESTORE DATABASE** ステートメントを実行するときに **MOVE .. TO** を指定しなかった場合は、上記の **PhysicalName**（旧環境で元々使っていた場所）へデータ

ベースが復元されるようになります。新しい環境で該当パスが存在しない場合は、復元時に次のように「**指定されたファイルが見つかりません**」エラーになります。

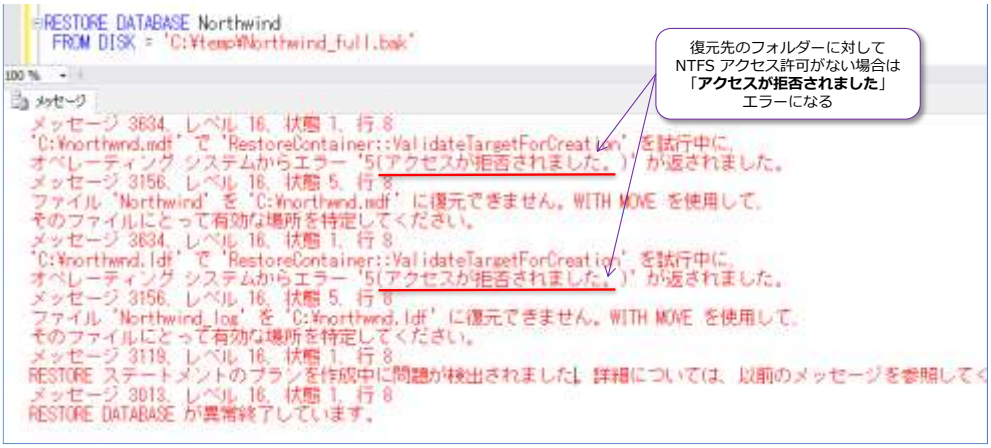



このエラーが出る場合は、**MOVE .. TO** オプションを利用して、復元先を変更するようにします。

| **Note： SQL Server 2005 のフルテキスト インデックス機能を利用している場合** |
| --- |
| **SQL Server 2005** の**フルテキスト インデックス**機能を利用している場合は、データベースの復元時に、フルテキスト カタログも復元されます。このときの復元先は、旧環境で元々使っていた場所になります。この場所も **MOVE .. TO** で変更することができますが、これについては後述します（**SQL Server 2008** 以降の場合は、フルテキスト カタログがデータベース エンジンに統合されているので、この作業は不要です）。 |

## NTFS アクセス許可がない場合のエラー

**RESTORE DATABASE** ステートメントを実行するときに、復元先のフォルダーに対して、SQL Server サービスの**サービス アカウント**に **NTFS アクセス許可**がない場合には、次のように「**アクセスが拒否されました**」エラーになります



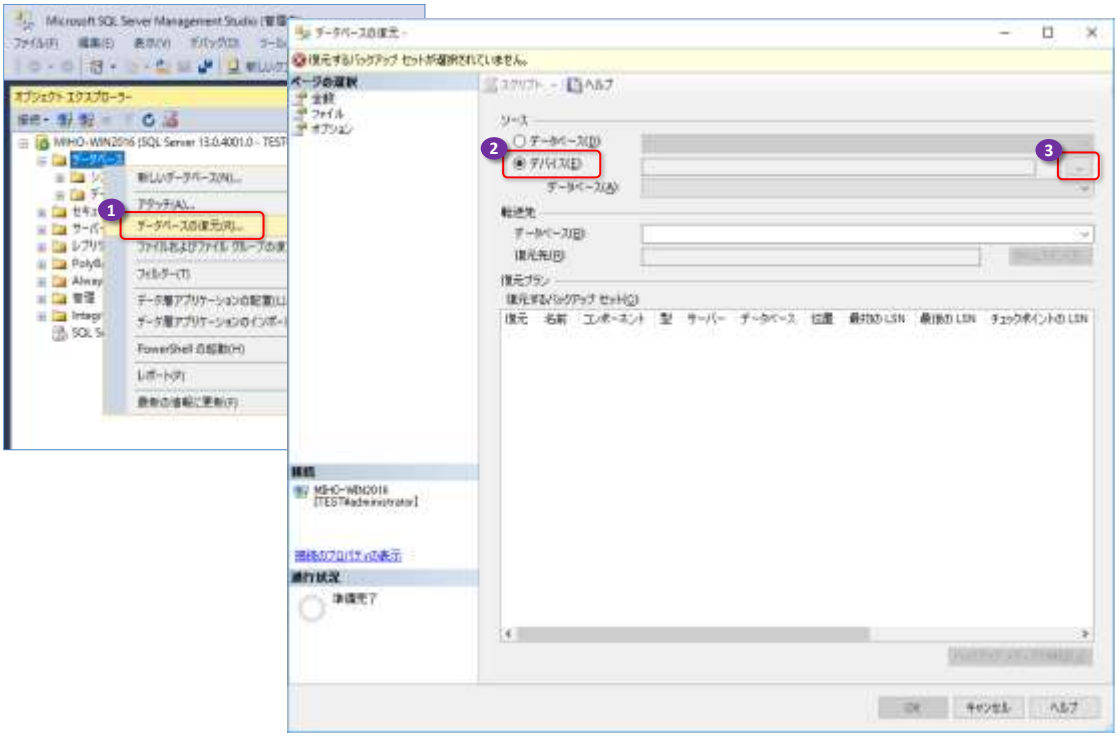

この場合は、NTFS アクセス許可を設定するようにします。

以上のように、以前のバージョンの SQL Server で取得したバックアップは、簡単に SQL Server 2016 上へ復元することができます。復元（リストア）にかかる時間は、データベース サイズやス

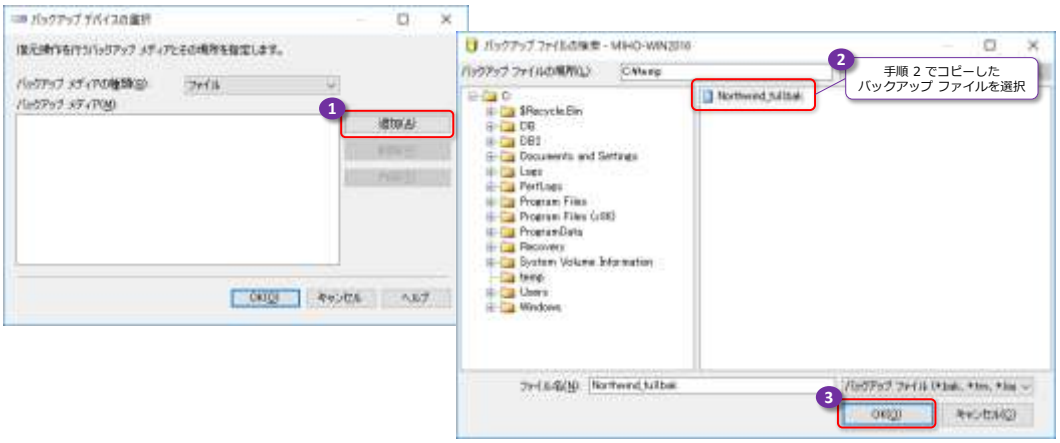
トレージの性能によって大きく変わってくるので、事前にどれぐらいの時間で復元ができるかをチェックしておくことも重要です（**第４章の 4.11** もご覧いただければと思います）。

**Note： GUI 操作でデータベースを復元したい場合**

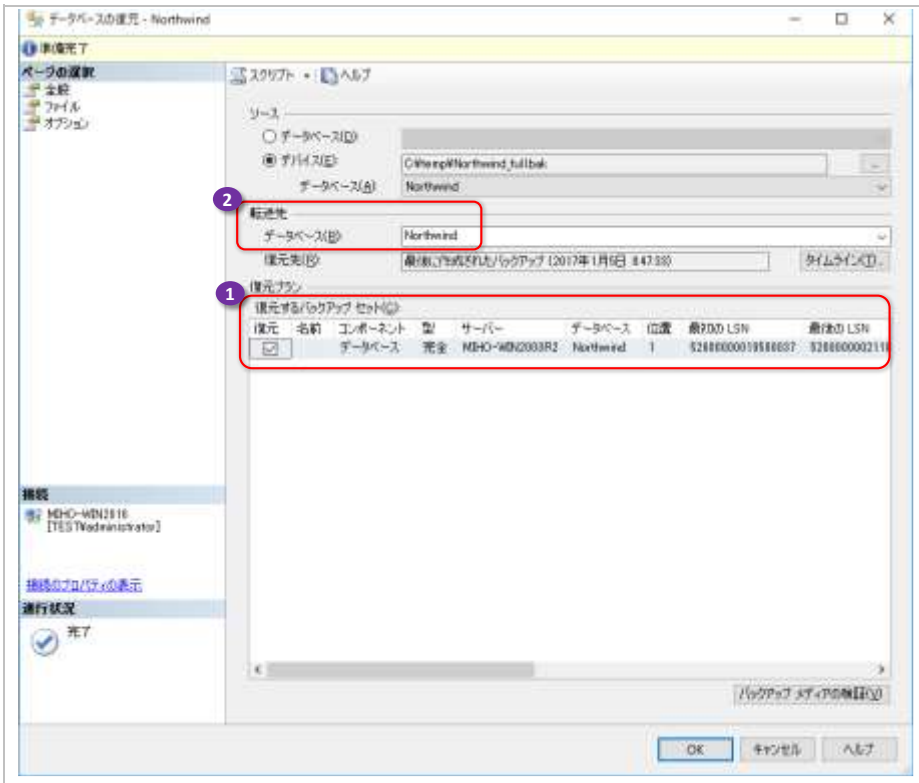
Management Studio を利用して、GUI 操作でデータベースを復元したい場合には、オブジェクト エクスプローラーで、次のように、[**データベース**] フォルダーを右クリックして、[**データベースの復元**] をクリックします。



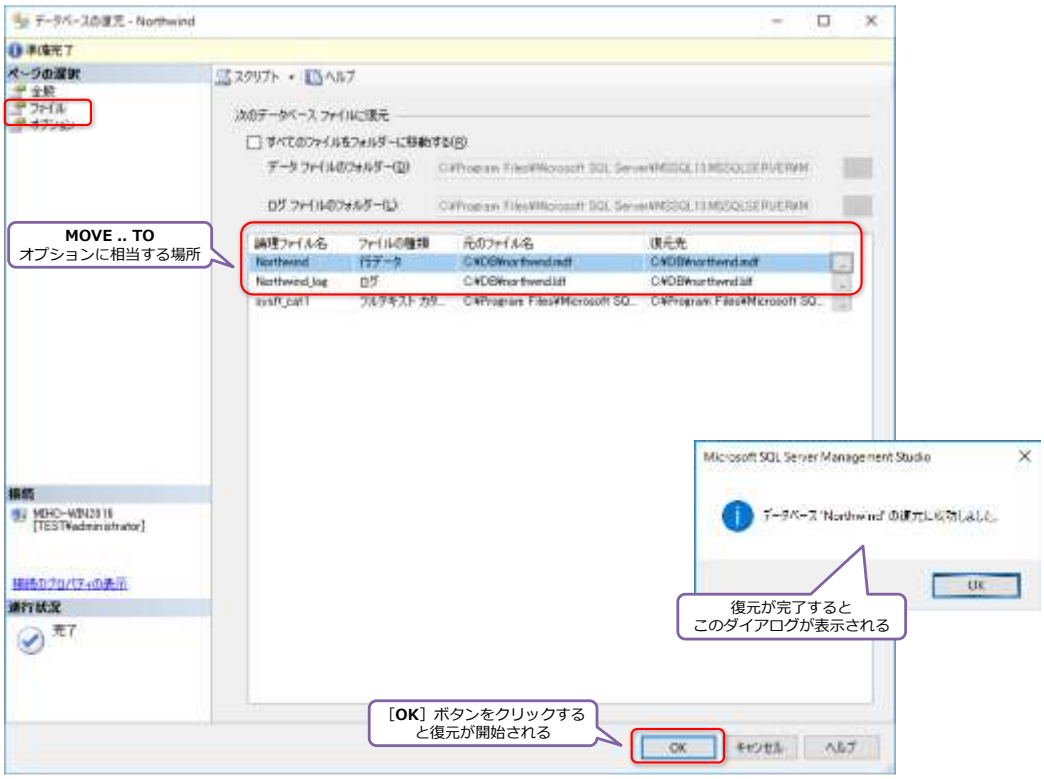

[**データベースの復元**] ダイアログでは、「**デバイス**」を選択して「**...**」ボタンをクリックします。次のように [**バックアップ デバイスの選択**] ダイアログが表示されたら、[**追加**] ボタンをクリックして、手順２ でコピーしたバックアップ ファイルを選択します。




[**OK**] ボタンをクリックして、[**データベースの復元**] ダイアログへ戻ったら、次のように [**復元するバックアップ セット**] で「**完全**」となっているバックアップが選択されていることを確認して、[**転送先**] の [**データベース**] に移行元のデータベース名が入力されていることを確認します（画面は **Northwind**）。



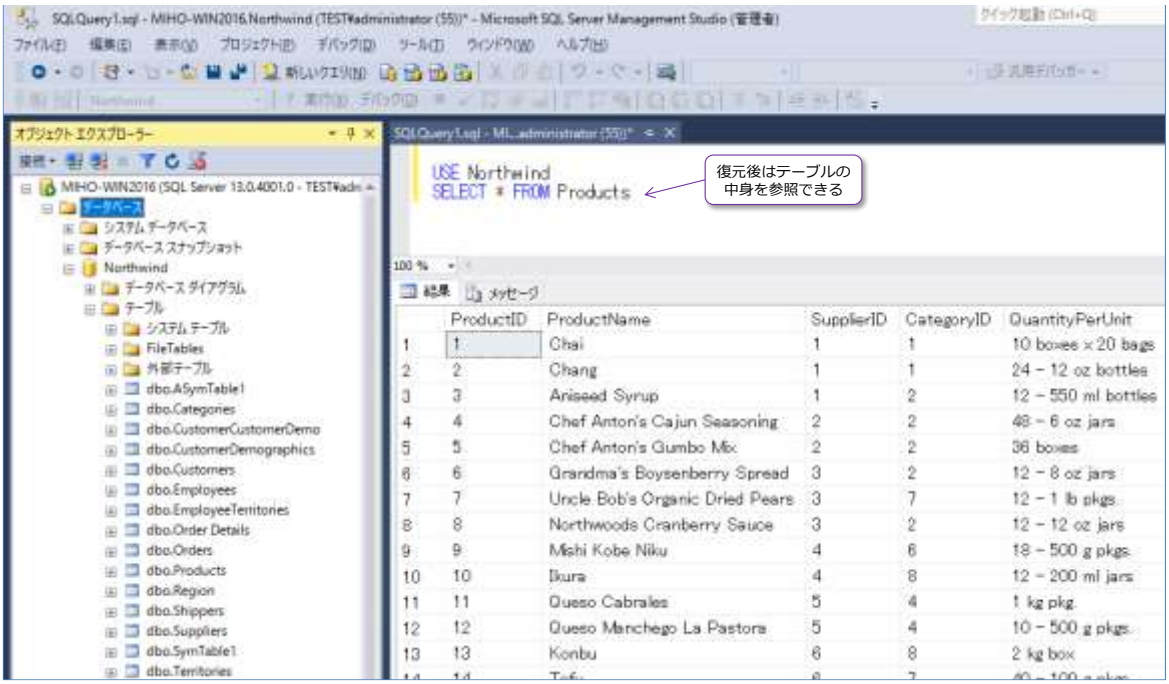
このダイアログでは、次のように［**ファイル**］ページから復元先のパスを変更することもできます（これは、前述の **MOVE .. TO** に相当し、移行元と移行先でドライブ構成やフォルダー構成が異なる場合に利用します）。




［**データベースの復元**］ダイアログでは、［**OK**］ボタンをクリックすると復元が開始され、復元が正常に完了すると、「**成功しました**」ダイアログが表示されます。

## 復元後のデータベースの状態

データベースの復元が完了すると、ほぼすべてのデータベース内のオブジェクトをそのまま利用することができ、テーブルの中身も参照することができます。




詳しくは後述しますが、テーブルやデータだけでなく、ビューやストアド プロシージャ、トリガー、インデックス、制約、ユーザー定義関数なども利用することができます。

## 復元後に行った方が良い作業

データベースを復元した後は、行っておいた方が良い作業がいくつかあります。その主なものは、次のとおりです。

- 統計の更新（sp_updatestats）
- フルテキスト検索機能を利用している場合は、フルテキスト インデックスの再構築
- 互換性レベルを 130 へ上げる（オプション）
- データベースの所有者を確認／設定する（所有者が空の場合は設定する）

詳しくは、次項以降で説明します。

## 復元後に動作しないもの

データベースを復元することによって、ほぼすべてのオブジェクトを以前と変わらずに利用することができますが、もちろん復元しただけでは操作しないものもあります（復元後に、追加の操作が必要になります）。その主なものは、次のとおりです。

- UNSAFE および外部にアクセス設定された SQL CLR オブジェクト（CLR 統合機能）
- データベース ダイアグラム（データベースの所有者が設定されていない場合のみ）
- データベース ユーザーとオブジェクト権限
- システム データベース関連のオブジェクト（ログイン アカウント、サーバー ロール、tempdb の設定、リンク サーバー、ユーザー定義エラー、構成オプション、ジョブ、警告、オペレーター、データベース メールなど）
- レジストリに格納された設定情報（サービスの自動起動やサービス アカウント、認証モード、TCP ポート番号、起動時パラメーターでのトレースフラグの設定など）
- メンテナンス プラン（保守計画）
- Integration Services を利用している場合は、SSIS パッケージ
- SQL Server Audit やリソース ガバナー、ポリシー ベースの管理、パフォーマンス データ コレクションなどのサーバー管理に関する機能
- レプリケーションやログ配布、可用性グループ、データベース ミラーリングなどサーバー間の連携機能

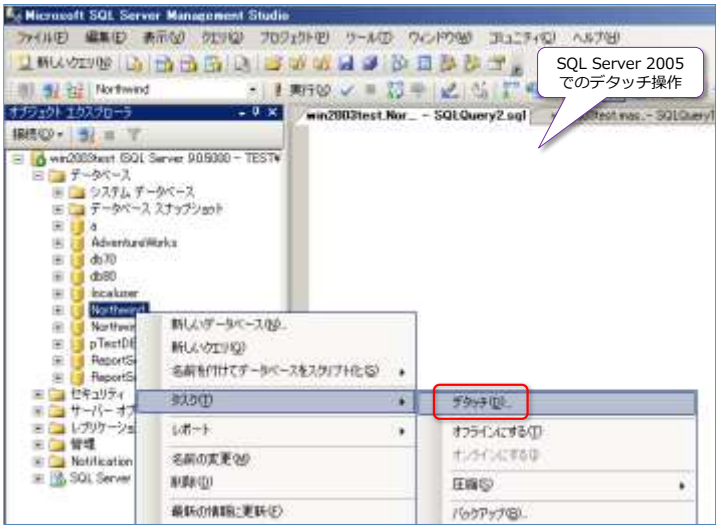
これらについても、後述します。

**Note： データベースを移行するその他の方法**

データベースを移行する方法には、そのほかにもいくつかあります。その主なものは、次の 3 つです。

**・デタッチとアタッチ**
**・データベース コピー ウィザード**
**・スクリプト生成ウィザード ＋ Integration Services によるデータの移動**

**デタッチとアタッチ**による移行は、次のように該当データベースを右クリックして［**タスク**］メニューから［**デタッチ**］をクリックすることで、いったん SQL Server 配下からデータベースを切り離す（.mdf／.ldf ファイルを操作できるようにする）ことができる機能です。

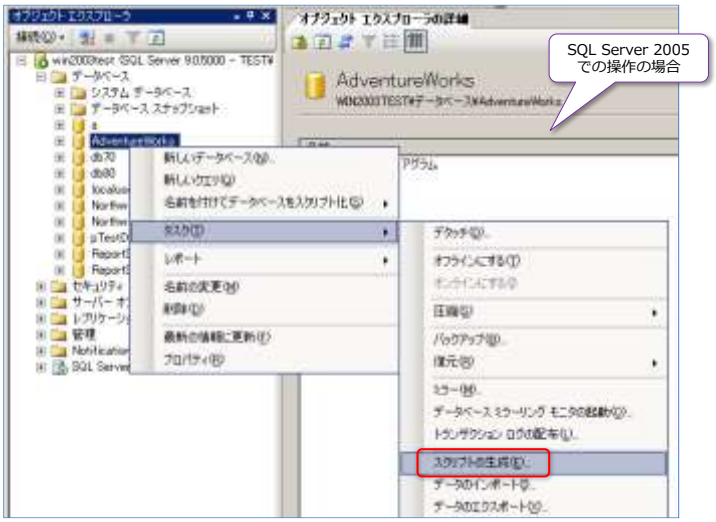


デタッチしたファイルを、移行先のサーバーにコピーして、移行先でこのファイルをアタッチすれば、データベースを移行することができます。バックアップ／復元機能での移行と同様、簡単にデータベースを移行することができるのですが、バックアップ／復元機能よりも、移行時のファイル サイズが大きくなってしまうことがデメリットです（∴お勧めではありません）。ファイル サイズが大きくなってしまうと、移行元から移行先にファイルをコピーするときの時間が長くなるので、移行時のダウンタイム時間が伸びてしまいます（第４章 4.11 参照）。

**データベース コピー ウィザード**は、移行元と移行先がネットワーク的に繋がっている場合に利用できる機能ですが、SQL Server 2016 の Management Studio 上から実行して、[**管理**] フォルダーを右クリックして、[**データベースのコピー**] で起動することができます。


内部的には、デタッチ／アタッチ操作または SMO（SQL Server 管理オブジェクト）が利用されていますが、ウィザードという形で何が行われているのかが分かりにくい部分があるので、お勧めではありません（∴ウィザードに頼らずに、機能を理解した上でデータベースの移行を行ったほうが、万が一の失敗時にリカバリが効くため）。

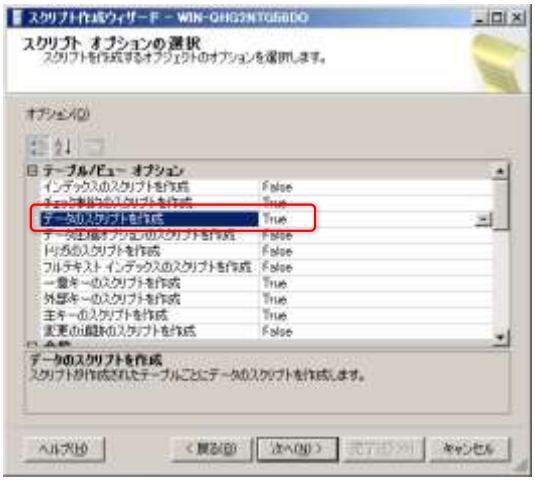
**スクリプト生成ウィザード**は、データベース内のオブジェクト（テーブルやビュー、ストアドプロシージャなど）の定義をすべてスクリプト化できる機能で、Management Studio で該当データベースを右クリックして、[**タスク**] の [**スクリプトの生成**] から実行できます。



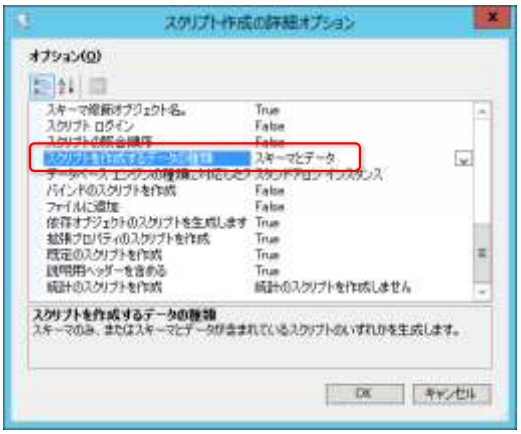
生成したスクリプトを移行先で実行して、同じオブジェクト（データは空）を作成し、データの移動（エクスポートとインポート）には、**Integration Services** や **bcp** コマンドを利用します。この方法は、データの移動を別途行わなければいけないので面倒です。

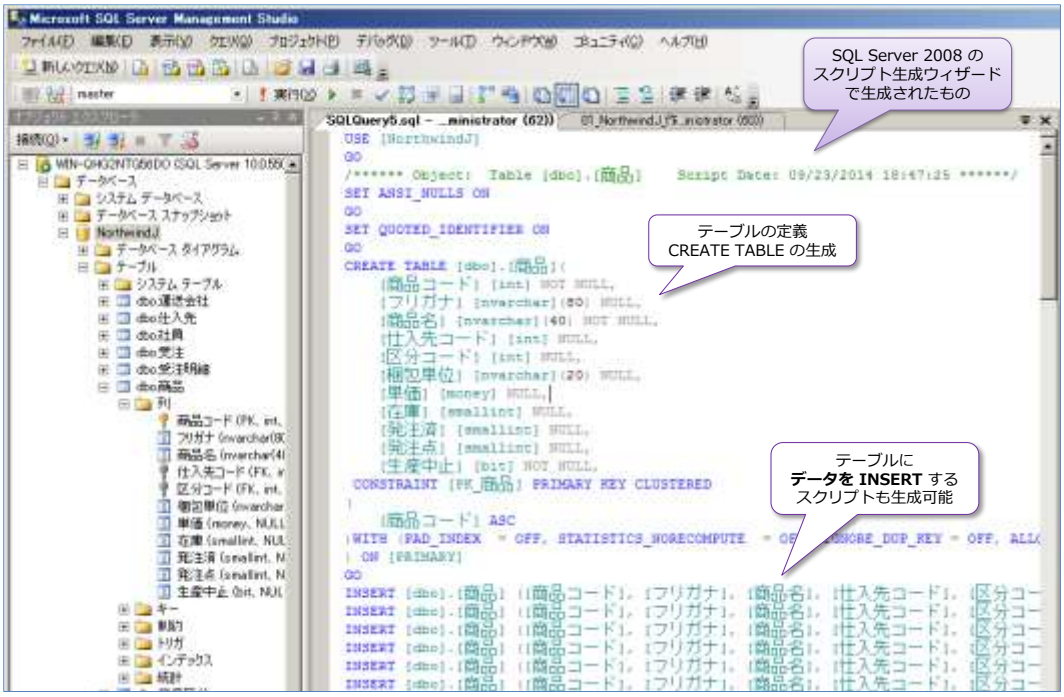
なお、SQL Server 2008 以降のスクリプト生成ウィザードでは、データを INSERT するスクリプトを生成する機能も追加されています。

SQL Server 2008 での設定画面




SQL Server 2008 R2 での設定画面（詳細設定ボタン）




これは、少量のデータを移行する場合には大変便利な機能なのですが、データ量が多い場合には、巨大なテキスト ファイルができあがってしまうので（INSERT ステートメントにすべての列名が埋め込まれているので、実際のデータ サイズよりも巨大になります）、データ量が多い場合にはお勧めはありません。

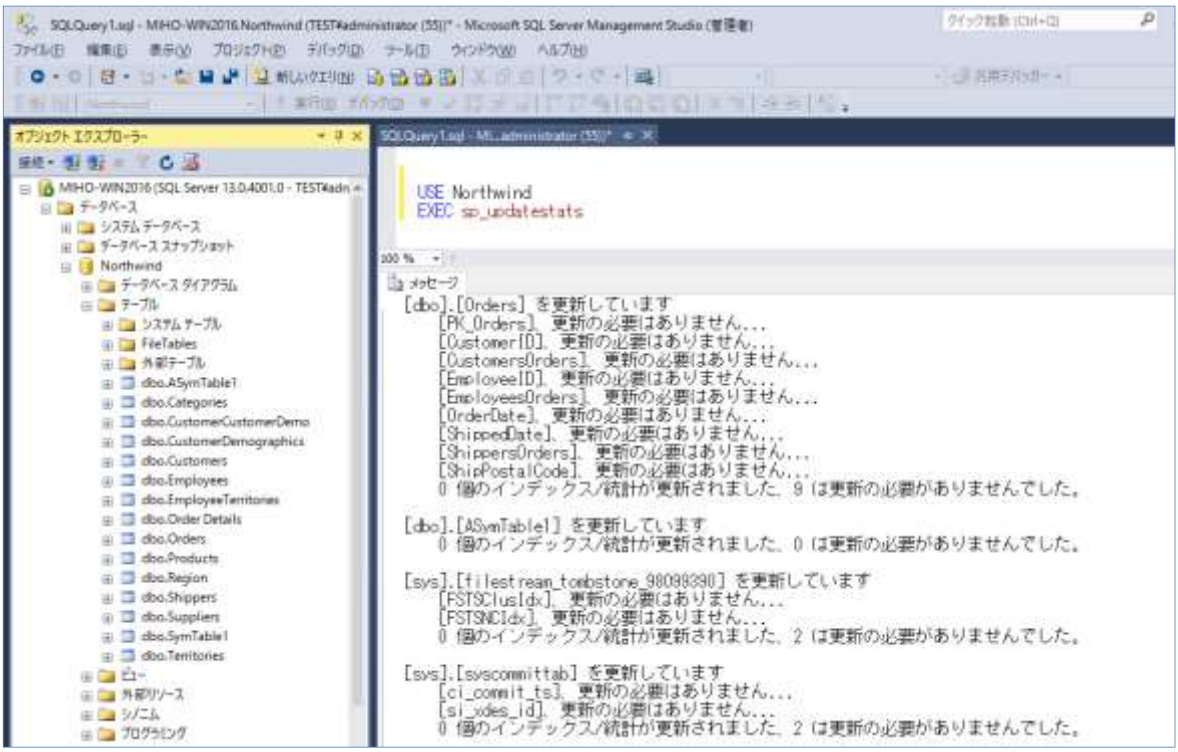
このように、データベースの移行を行えるツールは、たくさんあるのですが、一番のお勧めが本文中で説明したバックアップ／復元機能になります。

## 5.7 統計の更新

移行先のサーバーでデータベースを復元（リストア）した後は、**統計**（Statistics）を更新しておくことをお勧めします。統計は、クエリ オプティマイザーが最適な実行プランを選択するために、内部利用しているデータの分布状況です。移行後にも、最適な実行プランが選択されるようにするためには、統計を最新の状態へ更新しておくようにします。

統計を更新するには、次のように **sp_updatestats** システム ストアド プロシージャを実行します（第 3 章の 3.11 で説明したものと同じ操作です）。

```
USE データベース名
EXEC sp_updatestats
```




統計の更新は、テーブル サイズが大きい場合には、実行に時間がかかる場合があります。

統計の更新後は、実際に利用しているクエリの「**推定実行プラン**」や、可能であればクエリ実行時間などを確認して、アップグレード後にも問題なく、クエリが処理できているかどうかを確認しておくことをお勧めします。

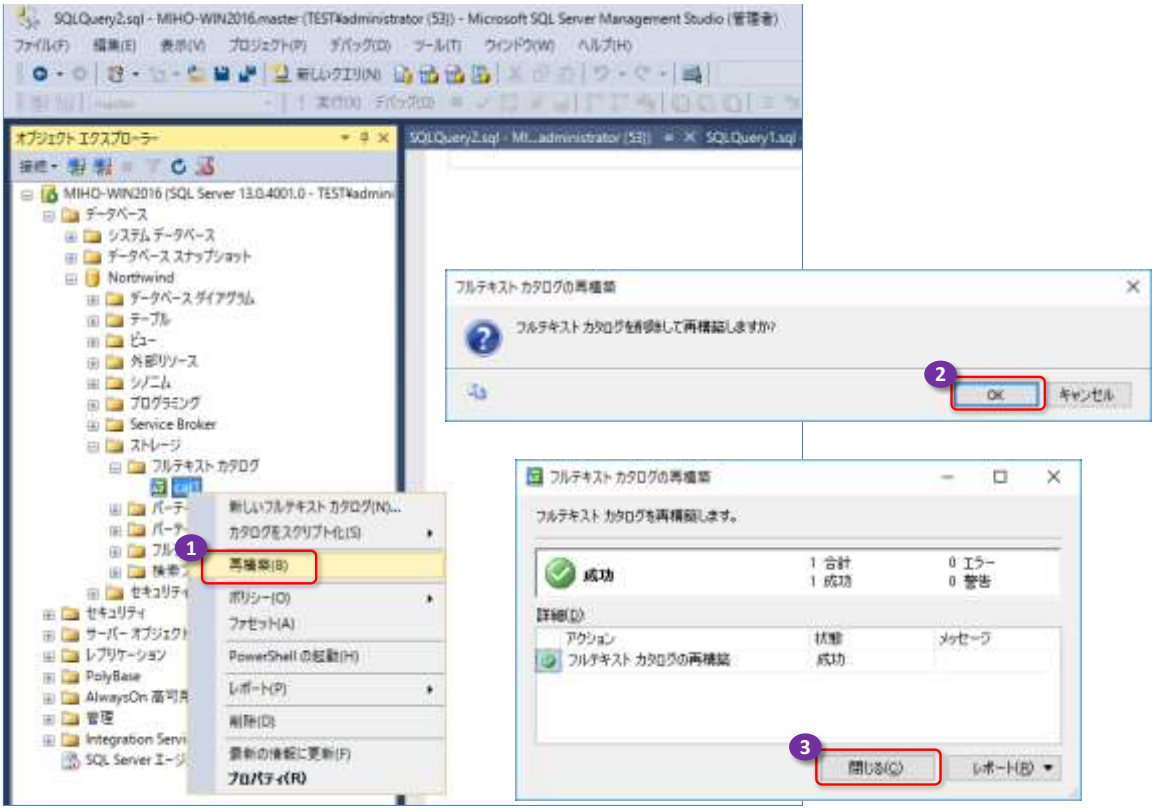
また、第 3 章の 3.13 で紹介した**クエリ ストア**（SQL Server 2016 からの新機能）を利用すれば、実行プランを確認したり、プラン強制（プランガイド）を行ったりすることもできるので、こちらもぜひ活用してみてください。

## 5.8 フルテキスト インデックスの再構築（フルテキスト検索利用の場合）

**フルテキスト検索**機能を利用している場合には、データベースを復元した後に、**フルテキスト インデックスの再構築**（カタログの再構築とインデックス作成の開始）をしておくことをお勧めします。SQL Server 2016 では、フルテキスト インデックスの内部アーキテクチャが変更されているためです。

### フルテキスト インデックスの再構築 ～カタログの再構築～

フルテキスト インデックスを再構築するには、まずフルテキスト カタログを再構築します。これを行うには、次のように［**ストレージ**］フォルダーの［**フルテキスト カタログ**］を展開して、該当カタログを右クリックして、「**再構築**」をクリックします。



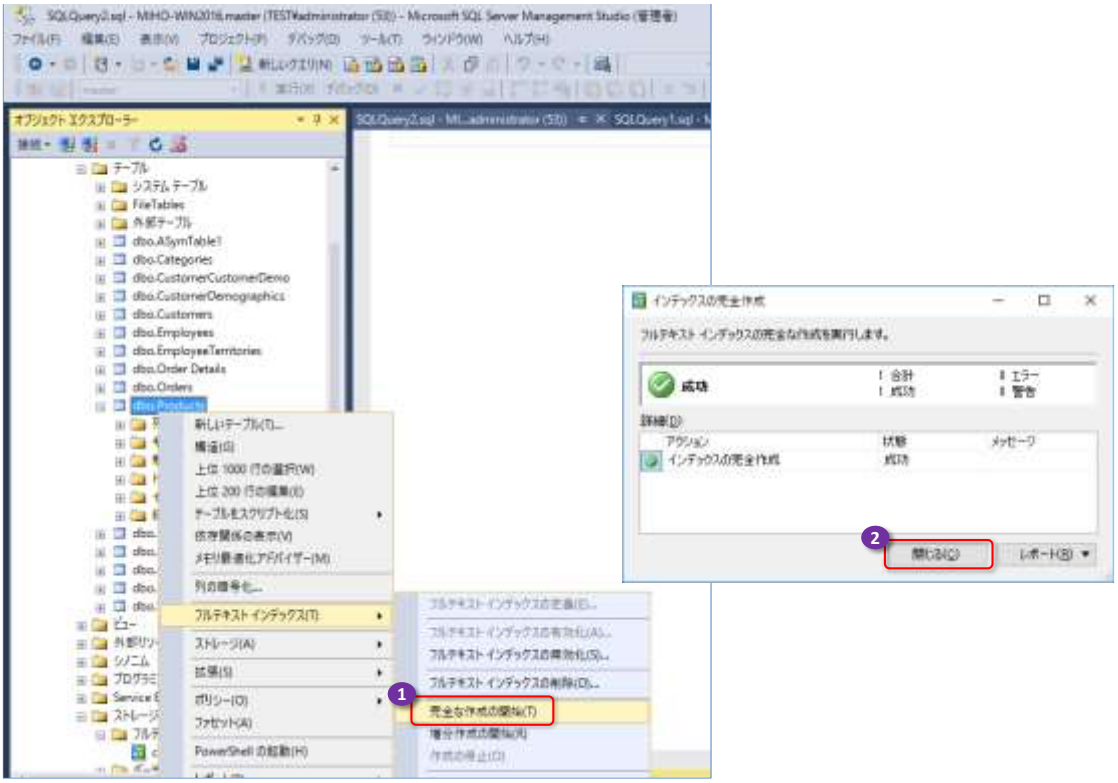

Transact-SQL ステートメントで実行する場合は、次のように **ALTER FULLTEXT CATALOG** ステートメントを実行します。

```
USE データベース名
ALTER FULLTEXT CATALOG カタログ名 REBUILD
```

### インデックス作成の開始

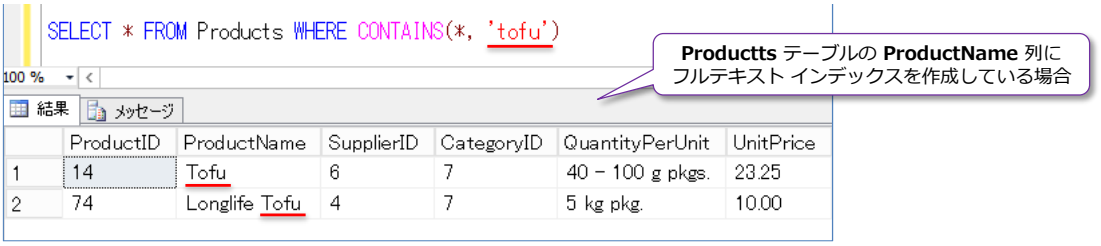
フルテキスト カタログを再構築すると、インデックスがリセットされるので、「**インデックス作成**

**の開始**」を行っておく必要があります。これを行うには、次のようにフルテキスト インデックスを作成しているテーブルを右クリックして（画面は **Products** テーブル）、[**フルテキスト インデックス**] の [**完全な作成の開始**] をクリックします。




この作業は、フルテキスト インデックスの完全な再作成（イチからの作成したのと同じ形）になるので、テーブル サイズが大きい場合には、実行時間がかかることに注意する必要があります。

以上で、フルテキスト インデックスの再構築が完了です。これで、SQL Server 2016 の新しいアーキテクチャで作成されたフルテキスト インデックスを利用できるようなります。




なお、Transact-SQL ステートメントを利用してインデックスの完全作成を開始する場合には、次のように **ALTER FULLTEXT INDEX** ステートメントを実行します。

```
ALTER FULLTEXT INDEX ON テーブル名
 START FULL POPULATION
```

**Note： SQL Server 2005 のフルテキスト インデックスとの違い**

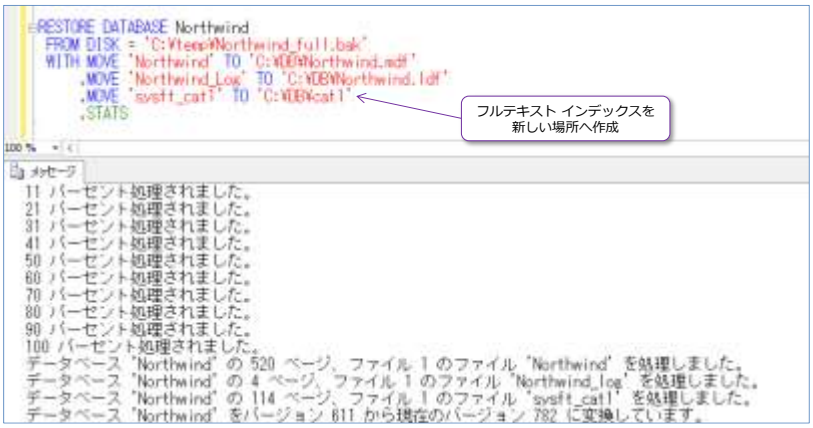
SQL Server 2005 で利用していたフルテキスト インデックスは、バックアップから復元（リストア）した後は、以前の環境で利用していた場所に作成されます。SQL Server 2005 の既定のインスタンスである場合には、既定では次の場所に作成されます。

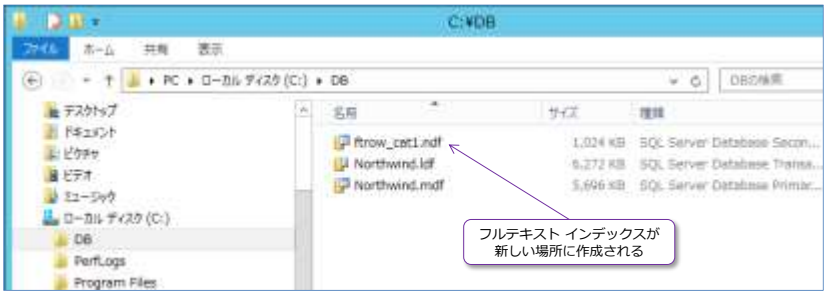
**C:¥Program Files¥Microsoft SQL Server¥MSSQL.1¥MSSQL¥FTData**

作成される場所は、次のように **RESTORE FILELISTONLY** ステートメントで確認することができます。


この場所を変更したい場合には、**RESTORE DATABASE** ステートメントでデータベースを復元するときに、次のように **MOVE .. TO** オプションを利用します。

```
RESTORE DATABASE Northwind
 FROM DISK = 'C:¥temp¥Northwind_full.bak'
 WITH MOVE 'Northwind' TO 'C:¥DB¥Northwind.mdf'
     ,MOVE 'Northwind_Log' TO 'C:¥DB¥Northwind.ldf'
     ,MOVE 'カタログの論理名' TO '新しいパス'
     ,STATS
```



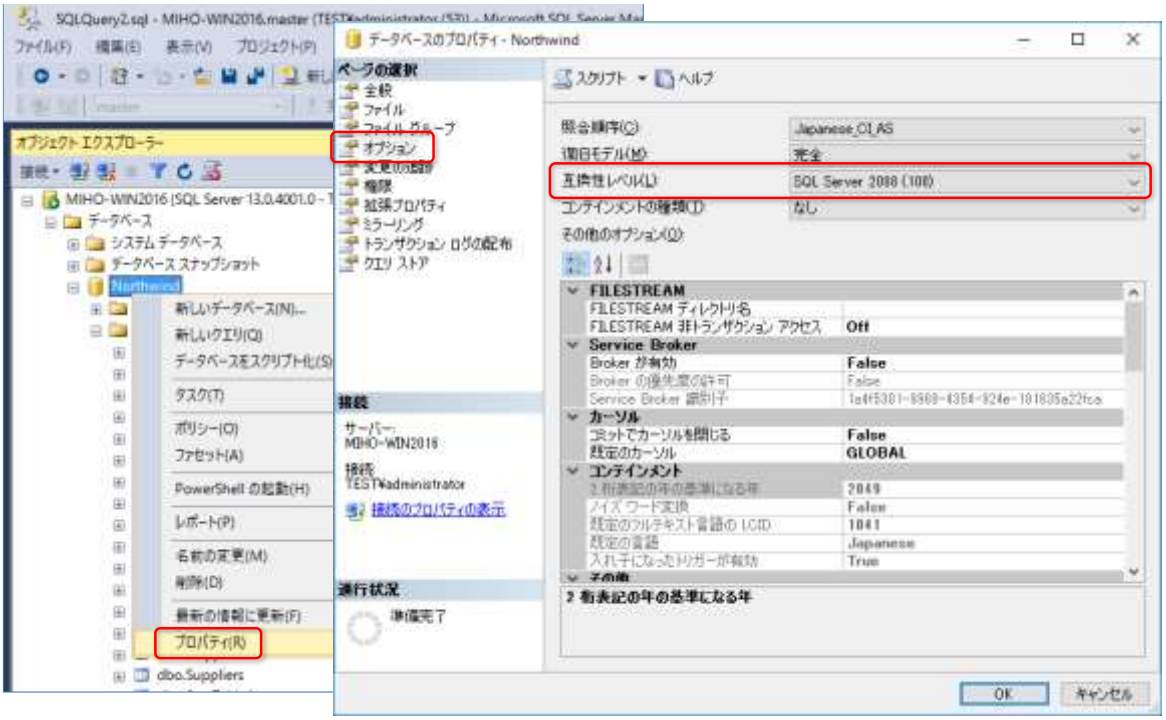

このように、フルテキスト インデックスの作成場所を変更したい場合には、MOVE .. TO オプションを利用します。なお、SQL Server 2005 では、フルテキスト インデックスは、複数のファイルで構成されていましたが、SQL Server 2008 からは、フルテキスト インデックスがデータベース エンジンに統合されたので、データベース ファイル（.ndf）として作成されるようになりました。

## 5.9 互換性レベルを 130 へ上げる（オプション）

移行元の **SQL Server** 上で取得した**データベースのバックアップ**の互換性レベルは、SQL Server 2016 に復元した後もその互換性レベルがキープされます。例えば、SQL Server 2008／2008 R2 で取得したバックアップを復元すると **100**、SQL Server 2012 の場合は **110**、SQL Server 2014 の場合は **120** といった具合です。

ただし、移行元のデータベースで**互換性レベル 90**（SQL Server 2005 レベル）以下を利用している場合は、**SQL Server 2016** に復元した後に、自動的に **100**（SQL Server 2008／2008 R2 レベル）に変更されます。

データベースの互換性レベルを確認／変更するには、次のようにユーザー データベースの［**プロパティ**］を開いて、［**オプション**］ページを表示します。




SQL Server 2016 では、互換性レベル **100** 以降がサポートされています。

### SQL Server 2016 で利用できるデータベースの互換性レベル

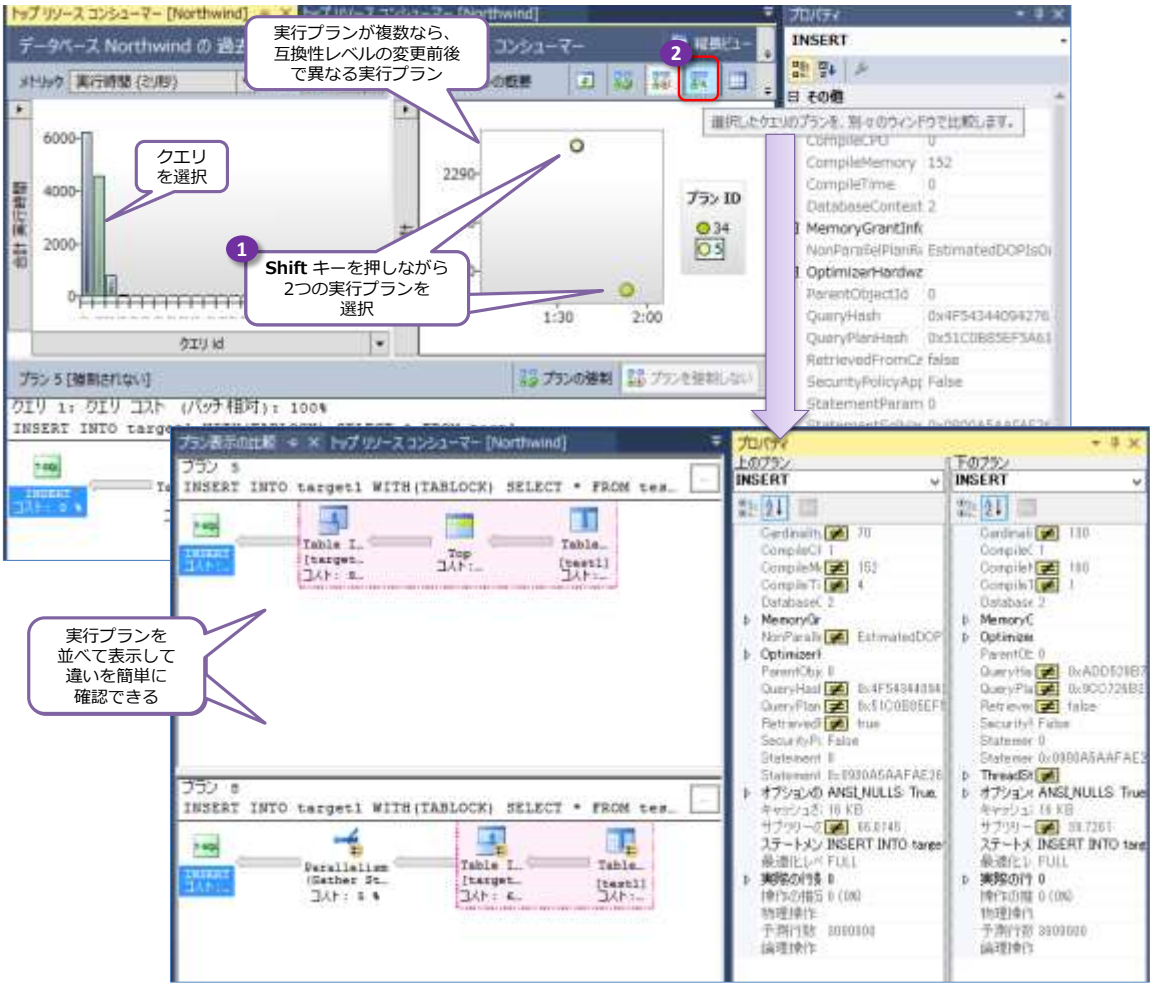
**SQL Server 2016** で利用できる**データベースの互換性レベル**は、次の 4 つです。

- **100**（SQL Server 2008／2008 R2 レベル）
- **110**（SQL Server 2012 レベル）
- **120**（SQL Server 2014 レベル）
- **130**（SQL Server 2016 レベル）

互換性レベルは、**100** や **110**、**120** のまま利用しても問題はありませんが、より良い性能を考慮

するのであれば、**130**（SQL Server 2016 レベル）に上げておくことがお勧めになります。**130** に変更すれば、SQL Server 2016 からの新機能である「**INSERT .. SELECT の並列処理**」や「**列ストア インデックスでのバッチ モードの性能向上**」、「**インメモリ OLTP での並列処理**」などを利用することができるからです。

互換性レベルを変更したことによる性能調査（実行プランの比較）には、SQL Server 2016 から提供された「**クエリ ストア**」（Query Store）機能を利用するのがお勧めです。**クエリ ストア**は、Store（蓄積）という名のとおり、**クエリの実行履歴／実行プラン**を保存できる機能で、次のように同じクエリに対して複数の実行プランが利用される場合を簡単に識別することができます。

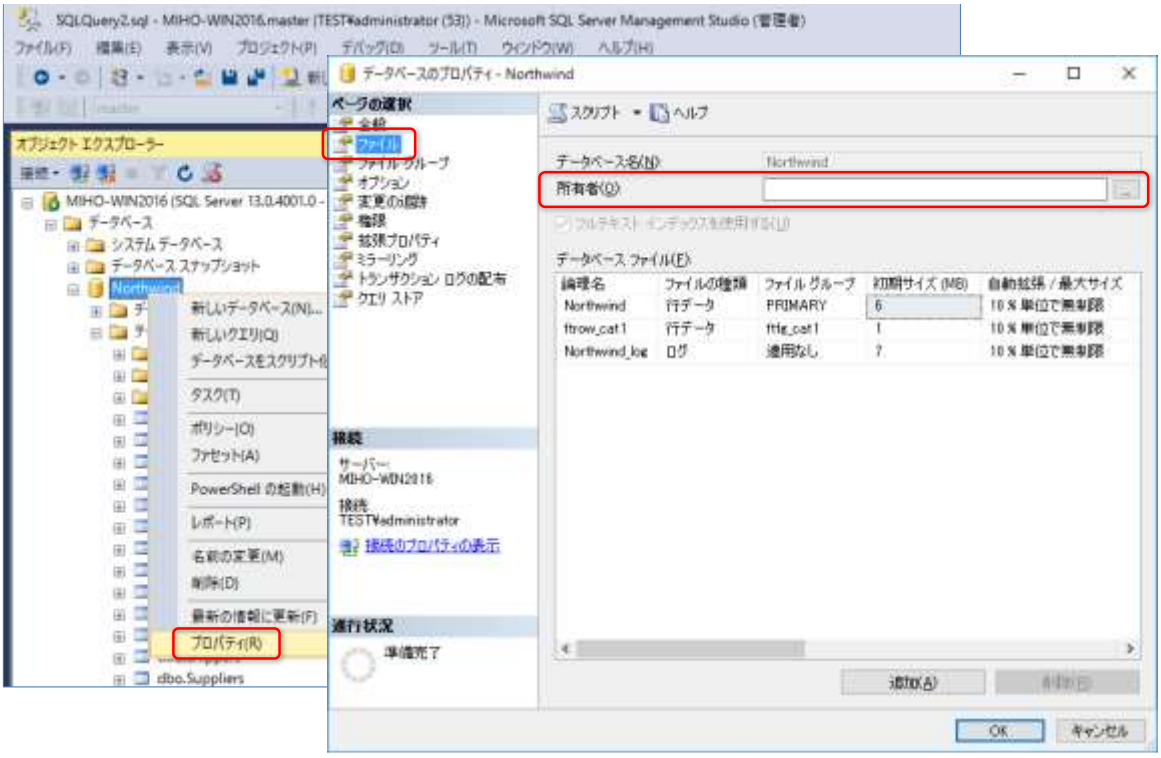



これを利用すれば、互換性レベルを変更する前後で、同じ実行プランが利用されたかどうかを簡単に確認することができるようになります。クエリ ストアの利用方法については、第 3 章の 3.13 で説明しているので、こちらもぜひご覧いただければと思います。

## 5.10 データベースの所有者を確認／空の場合は設定する

移行先でデータベースを復元（リストア）した後は、**データベースの所有者**が設定されていることを確認しておくようにします。**移行元**で、データベースの所有者が **Active Directory ドメインのユーザー**になっていて、**移行先**でも**同じドメインに参加**している場合であれば、そのユーザーが引き続きデータベースの所有者に設定されるのですが、異なるドメインや、ワークグループ環境へ復元した場合や、Windows のローカル ユーザーが所有者になっている場合は、復元後にデータベースの所有者が「**空**」に設定されてしまいます。

データベースの所有者が「**空**」になっているかどうかは、次のようにデータベースのプロパティの［**ファイル**］ページから確認することができます。




また、状況によっては、データベースのプロパティが開けない（エラーになる）こともあり、この場合もデータベースの所有者が「**空**」に設定されています。

### データベースの所有者が「空」の場合の影響

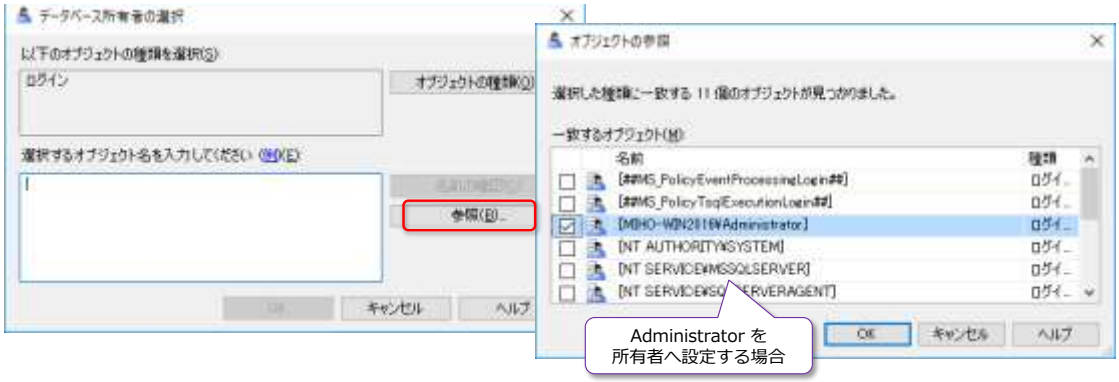
データベースの所有者が「**空**」の場合には、次の 2 つの機能が動作しません。

- **SQL CLR オブジェクト**で、権限セットを「**UNSAFE**」（アンセーフ）または「**外部**」（EXTERNAL_ACCESS）に設定しているものが動作しない。「**SAFE**」に設定しているものは動作する
- **データベース ダイアグラム**を表示できない

したがって、これらが動作するようにするには、**データベースの所有者を設定**します。

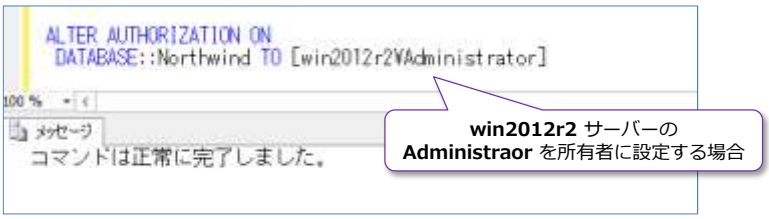
## データベースの所有者を設定する

データベースの所有者を設定するには、データベースのプロパティの［**ファイル**］ページで、［**...**］ボタンをクリックして、［**データベースの所有者の選択**］ダイアログを表示し、次のように任意のユーザー（**Administrator** や **sa** など）を所有者に設定します。




Transact-SQL ステートメントでデータベース所有者を設定する場合は、次のように **ALTER AUTHORIZATION** ステートメントを実行します。

```
ALTER AUTHORIZATION ON
 DATABASE::データベース名 TO［所有者名］
```



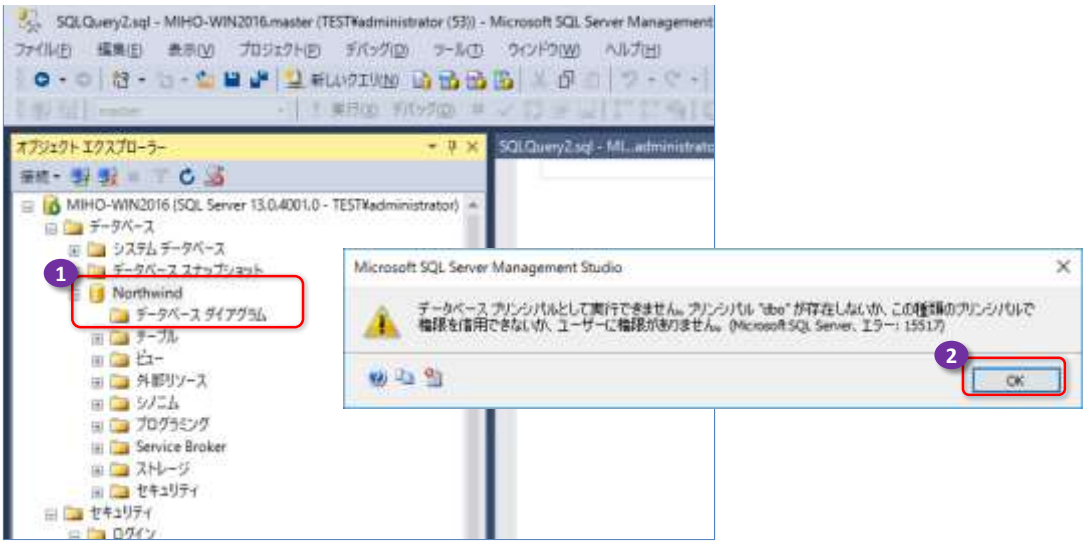

所有者名に Windows ユーザーを指定する場合は、[ ] で囲んで、**サーバー名¥ユーザー名** あるいは **ドメイン名¥ユーザー名** 形式で指定します。

なお、以前のバージョンで所有者名を変更するために利用できた「**sp_changedbowner**」を利用しても、次のように所有者名を変更することができますが、このストアド プロシージャは、将来のバージョンでは削除される予定なので、**ALTER AUTHORIZATION** ステートメントを利用することをお勧めします。

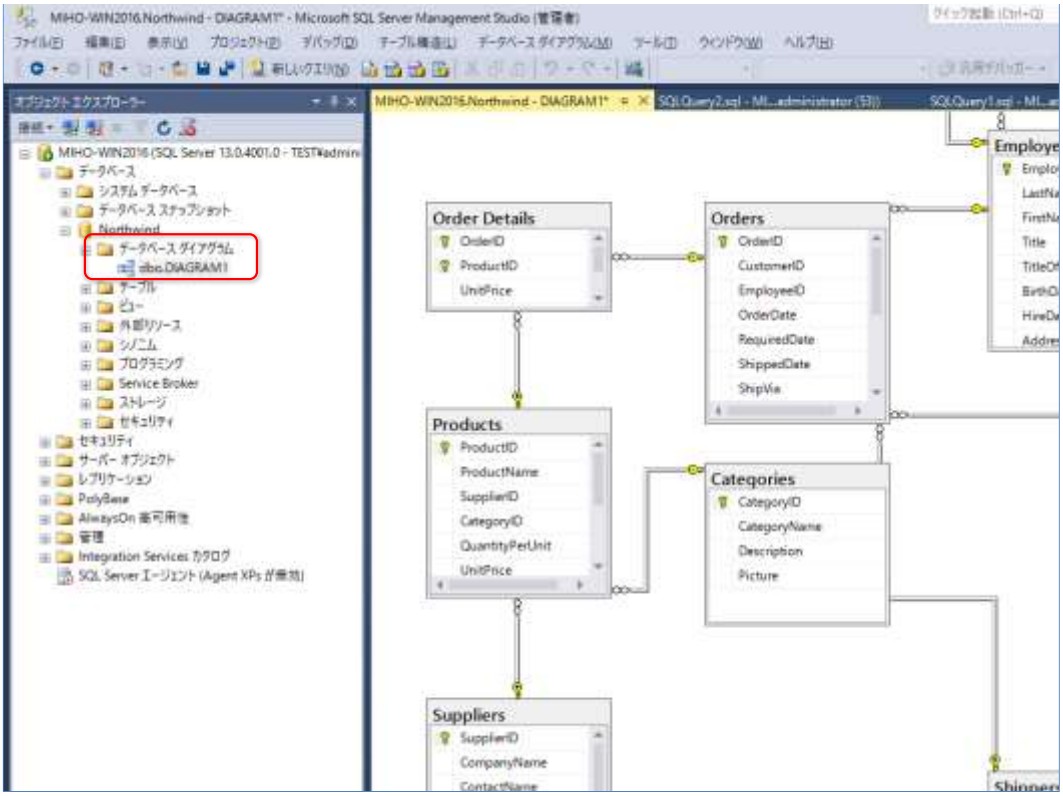
```
USE データベース名
EXEC sp_changedbowner '所有者名'
```

## データベース ダイアグラムの表示

**データベース ダイアグラム**は、データベースの所有者が設定されていない場合には、次のようにエラーが表示されます。

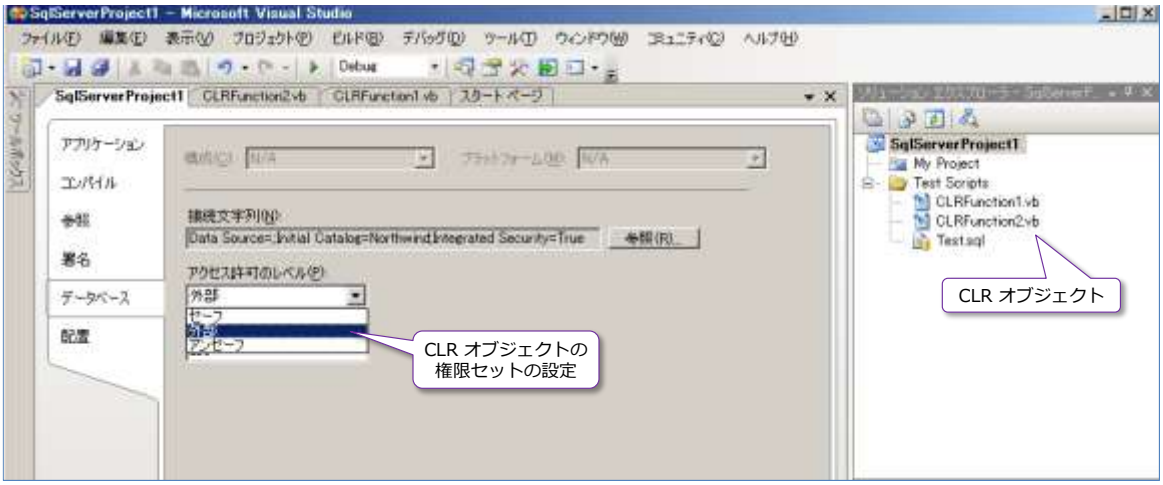



データベースの所有者が設定されていれば、次のようにデータベース ダイアグラムを開けるようになります。



## 5.11 SQL CLR オブジェクトの移行

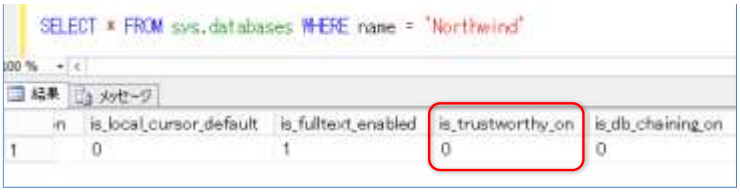
**SQL CLR オブジェクト**（.NET Framework の CLR 統合機能を利用した関数やストアド プロシージャなど）は、権限セットが「**SAFE**」（安全）に設定されたものである場合は、データベースの復元（リストア後）でも、そのまま動作するのですが、権限セットが、次のように「**UNSAFE**」（アンセーフ）および「**外部**」（EXTERNAL_ACCESS）に設定している場合は、そのままでは動作しません。




復元しただけでは、次のように「**サーバーのリソースが不足しているか、PERMISSION_SET が EXTERNAL_ACCESS または UNSAFE に設定されていて、アセンブリが信頼されていない可能性があります**」エラーが発生します。

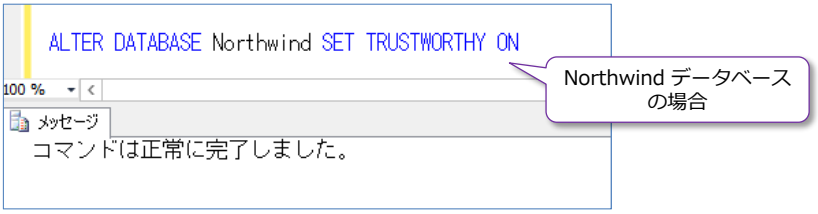

### TRUSTWORTHY を有効化する

権限セットを「**UNSAFE**」または「**外部**」に設定している CLR オブジェクトを実行するには、**TRUSTWORTHY** オプションを有効化しておく必要がありますが、データベースの復元後は、次のように **0**（無効）に設定されてしまいます。



データベースの復元では、ほとんどのデータベース設定は、そのまま復元することができるのですが、**TRUSTWORTHY** オプションに関しては復元することができません（**0** にリセットされてしまいます）。したがって、次のように **ALTER DATABASE** ステートメントを利用して、**TRUSTWORTHY** オプションを有効化しておく必要があります。

```
ALTER DATABASE データベース名
 SET TRUSTWORTHY ON
```




## データベースの所有者の設定（空の場合はエラーになる）

「**UNSAFE**」または「**外部**」に設定している CLR オブジェクトを実行するには、**データベースの所有者**が設定されている必要もあります。設定されていない場合は、前述のエラーと同じものが通達されるので、データベースの所有者は、必ず設定しておくようにします。

また、データベースの所有者が **sysadmin** ロールのメンバーでない場合は、次のように「**EXTERNAL ACCESS ASSEMBLY**」権限も付与しておくようにします。

```
USE master
GRANT EXTERNAL ACCESS ASSEMBLY TO［データベース所有者のログイン名］
```

## サービス アカウントにアクセス権限があることを確認

**外部**（EXTERNAL_ACCESS）に設定した CLR オブジェクトの場合は、SQL Server サービスの**サービス アカウント**が外部リソースへのアクセスを行うので、サービス アカウントがそのリソースに対してアクセス権限（**NTFS アクセス許可**など）を持っている必要があるので、うまく動作しない場合は、これについても確認してみてください。

## 5.12 バックアップ／復元で移行できるもの／できないもの

**RESTORE DATABASE** ステートメントによるデータベースの復元では、該当データベース内のオブジェクトをすべて復元することができるので、以下のオブジェクトを移行することができます。

| 移行可能なオブジェクト（RESTORE DATABASE で復元されるもの） | |
|---|---|
| ・ テーブル<br>（テーブル内のデータも含む） | ・ ユーザー定義関数 |
| ・ 制約 | ・ SQL CLR オブジェクト<br>（UNSAFE／外部の場合は追加作業が必要） |
| ・ インデックス | ・ データベースの設定<br>（TRUSTWORTHY オプションは 0 に設定される） |
| ・ 統計<br>（移行後、更新することを推奨） | ・ データベース ダイアグラム<br>（データベースの所有者を設定しておく必要がある） |
| ・ フルテキスト インデックス<br>（移行後、再構築をするのを推奨） | ・ オブジェクト権限<br>（追加作業としてログイン アカウントの再作成が必要） |
| ・ ビュー | ・ データベース ユーザー<br>（追加作業としてログイン アカウントの再作成が必要） |
| ・ ストアド プロシージャ | ・ データベース ロール<br>（データベース ユーザーを復旧できればそのまま利用可能） |
| ・ トリガー | ・ アプリケーション ロール |
| ・ ユーザー定義データ型 | ・ データベース マスター キー<br>・ 証明書／対称キー／非対称キー（EncryptByKey や DecryptByKey での暗号／復号化は復元後も利用可能） |
| ・ データ パーティション | |

このように、データベース内に含まれているものは、ほぼすべてを移行することができ、**テーブル**や**テーブル内のデータ**、**制約**、**インデックス**、**ビュー**、**ストアド プロシージャ**、**トリガー**、**データ パーティション**など、主要なオブジェクトは、何の問題もなく利用することができます。

前述したように、**統計**に関しては、**統計の更新**（**sp_updatestats**）、**フルテキスト インデックス**に関しては、**再構築**（カタログの再構築とインデックス作成の開始）を行っておくことがお勧めになります。

**データベース ダイアグラム**は、**データベースの所有者**を設定しておく必要があります。

**SQL CLR オブジェクト**は、権限セットが **UNSAFE**／**外部**の場合は、**TRUSTWORTHY** オプションを有効化して、**データベースの所有者**を設定しておく必要があります（権限セットが **SAFE** の場合には、データベースを復元するだけで **.dll** も自動復元されるので、SQL CLR オブジェクトをそのまま利用することができます）。

移行することはできても、追加の作業が必要になるのが、**オブジェクト権限**と**データベース ユーザー**です。データベース ユーザーは、ログイン アカウントとマッピングされるものですが、ログイン アカウントは、システム データベースである「**master**」内に含まれるものになるので、これを再作成しないと動作させることができません。

### バックアップ／復元では移行できないオブジェクト

データベースの復元は、あくまでも "**該当データベースのみ**" の移行になるので、**システム データベース**（**master** や **msdb**）に格納される情報や、**レジストリ**に格納される情報については、移行

が完了していません。したがって、データベースを復元しただけでは、主に次の機能を利用することができません。

- **オブジェクト権限**と**データベース ユーザー**
- **システム データベース**関連のオブジェクト（ログイン アカウント、サーバー ロール、tempdb の設定、リンク サーバー、ユーザー定義エラー、構成オプション、ジョブ、警告、オペレーター、データベース メール、TDE を設定したデータベースなど）
- **レジストリ**に格納された設定情報（サービスの自動起動やサービス アカウント、認証モード、TCP ポート番号、起動時パラメーターでのトレースフラグの設定など）
- **メンテナンス プラン**（保守計画）、Integration Services の SSIS パッケージ）
- SQL Server Audit やリソース ガバナー、ポリシー ベースの管理、パフォーマンス データ コレクションなどのサーバーの管理機能
- レプリケーションやログ配布、データベース ミラーリング、可用性グループなどサーバー間の連携機能

これらの移行方法については、次項以降で説明します。

## システム データベースに含まれるものは？

SQL Server の**システム データベース**には、**master**、**msdb**、**model**、**tempdb**、**distribution** の 5 つがありますが、**tempdb** は一時領域用のデータベースなので移行する必要はありません。また、**distribution** はレプリケーションで利用されるデータベースなので、レプリケーション環境での移行時にのみ関係します。**model** は、新しくデータベースを作成する際のテンプレート データベースなので、これも移行する必要はありません。なお、内部的に利用しているシステム データベースとして **Resource** もありますが、このデータベースも移行する必要はありません。

したがって、**システム データベース**の中に含まれるものを意識するのは、**master** と **msdb** の 2 つのデータベースだけで良いことになります。**master** と **msdb** データベースの中に含まれる情報は、次のとおりです。

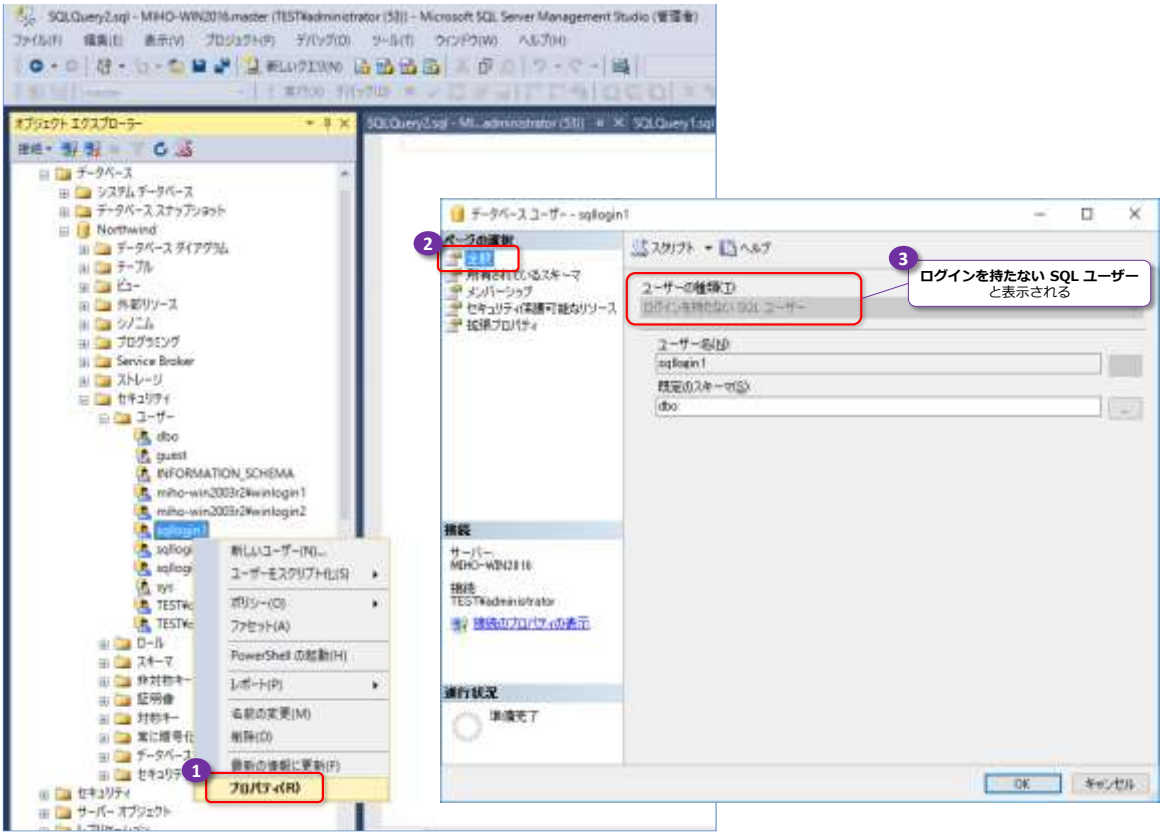
| master に含まれる主なもの | msdb に含まれる主なもの |
|---|---|
| ・ログイン アカウント | ・データベース メールの設定 |
| ・サーバー ロールの設定 | ・ジョブ |
| ・構成オプション（sp_configure） | ・警告 |
| ・tempdb の設定 | ・オペレーター |
| ・リンク サーバー設定 | ・Integration Services パッケージ |
| ・バックアップ デバイス | ・メンテナンス プラン（保守計画） |
| ・ユーザー定義エラー | ・レプリケーションの設定 |
| ・データベース ミラーリングの設定（msdb にも一部有り） | ・ログ配布の設定 |

**master** データベースには**ログイン アカウント**や**サーバー ロールの設定**、**構成オプション**、**tempdb の設定**、**リンク サーバーの設定**など、**msdb** データベースには**データベース メールの設定**や**ジョブ**、**警告**、**オペレーター**、**Integration Services パッケージ**、**メンテナンス プラン**（**保守計画**）などが格納されています。これらについて、次項以降で詳しく説明します。

## 5.13 ログイン アカウントの移行（DB ユーザー、オブジェクト権限）

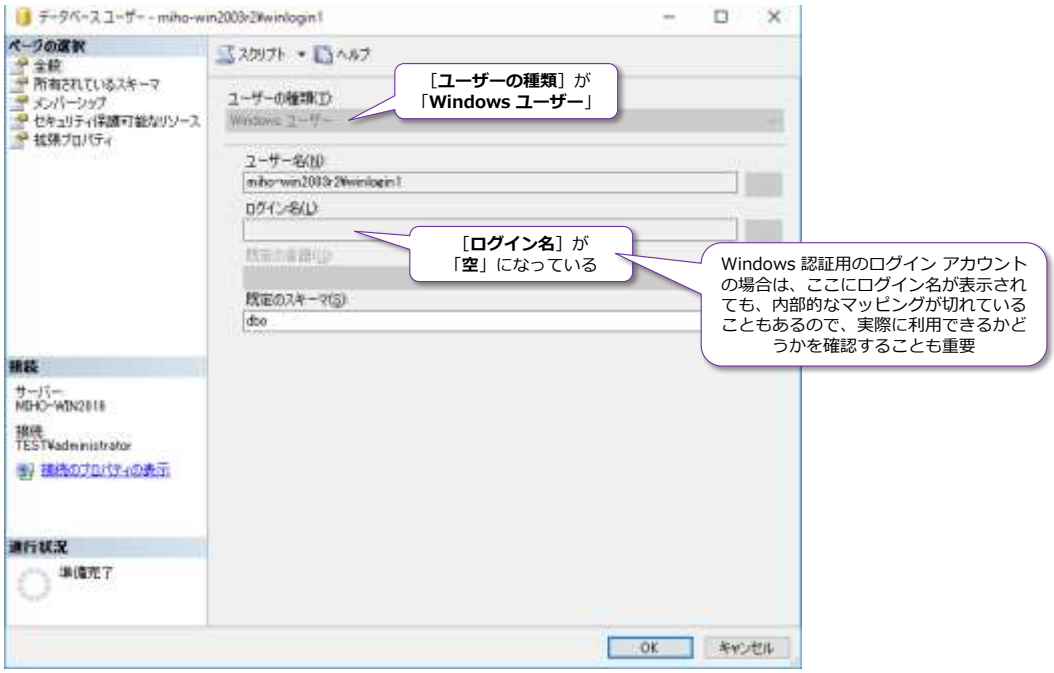
**データベース ユーザー**は、**ログイン アカウント**と紐付いている（マッピングされている）ので、ログイン アカウントの移行を行わないと、データベース ユーザーを利用することができません。また、データベース ユーザーに紐付いた**オブジェクト権限**（テーブルやビューに対する **SELECT 権限**や **INSERT 権限**、ストアド プロシージャに対する **EXECUTE 権限**など）も無力になってしまいます。

ログイン アカウントとのマッピングが切れたデータベース ユーザーは、「**不明なデータベース ユーザー**」や「**孤立ユーザー**」と呼ばれ、次のように表示されます。



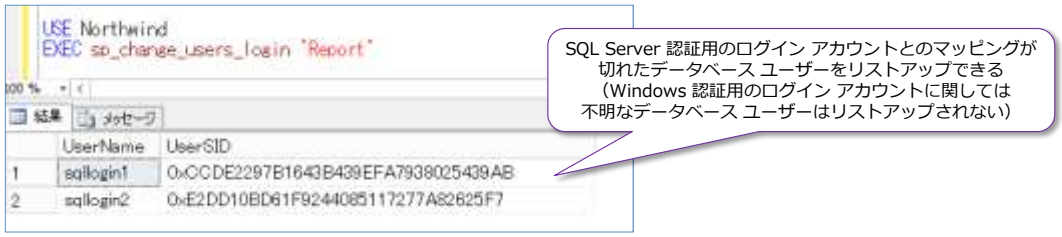

これは、**SQL Server 認証用のログイン アカウント**である「**sqllogin1**」に対応したデータベース ユーザーのプロパティを開いている画面で、[**ユーザーの種類**] が「**ログインを持たない SQL ユーザー**」になっています。データベース ユーザーに対応したログイン アカウントが存在しない場合（マッピングが切れている場合）には、このように表示されます。この状態では、**sqllogin1** ユーザーはデータベースにアクセスすることができません。

**Windows 認証用のログイン アカウント**（**Active Directory ドメイン ユーザー**や **Windows ローカル ユーザー**に対応したログイン アカウントの場合）は、次のように [**ユーザーの種類**] が「**Windows ユーザー**」と表示されて、[**ログイン名**] が "**空**" で表示されるものが、不明なデータベース ユーザーとなっていて、利用することができません。

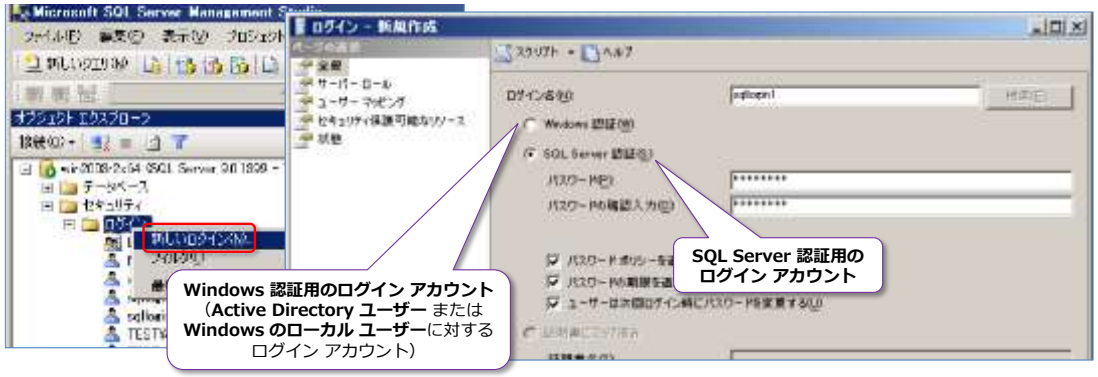



なお、SQL Server 認証用のログイン アカウントとのマッピングが切れたユーザー（不明なデータベース ユーザー）は、次のように **sp_change_users_login** というシステム ストアド プロシージャを利用して、リストアップすることもできます。

```
USE データベース名
EXEC sp_change_users_login 'Report'
```




このような不明なデータベース ユーザーを正しく利用できるようにするには、移行元と移行先で、同一のログイン アカウントを作成しておく必要があります。作成方法／注意点は、**Windows 認証用のログイン アカウント**である **Active Directory ユーザー**または **Windows のローカル ユーザー**の場合と、**SQL Server 認証用のログイン アカウント**の場合で異なるので、それぞれごとに説明します。

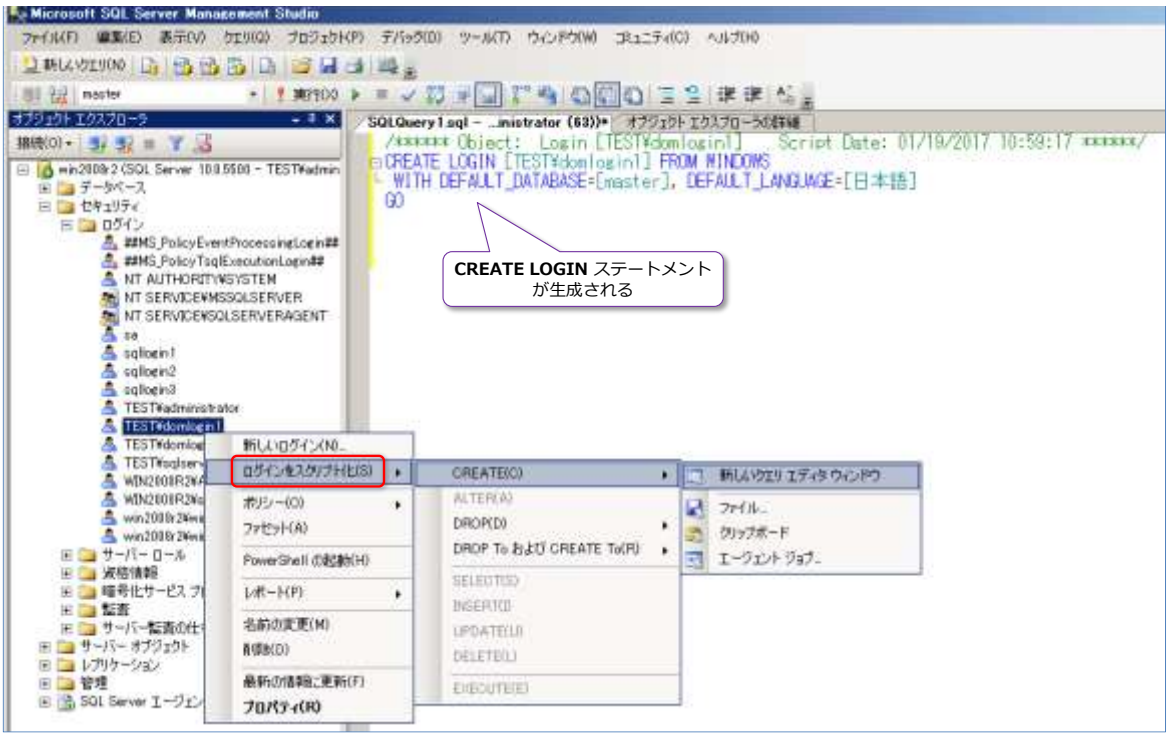


## Active Directory ユーザー（ドメイン ユーザー）の場合

ログイン アカウントが Windows 認証用で、かつ Active Directory ドメイン ユーザー、移行元と移行先が同じドメインに所属している場合には、同一のログイン アカウントを簡単に作成することができます。標準の「**スクリプト生成**」機能や、後述の「**sp_help_revlogin ストアド プロシージャ**」、「**データベース コピー ウィザード**」、「**Integration Services のログイン転送タスク**」などを利用して、ログイン アカウントを再作成または転送すれば、同一のログイン アカウントを作成できるので、不明なデータベース ユーザーを簡単に解消することができます（データベース ユーザーを移行元とまったく同じように利用できるようになります）。

### 標準の「スクリプト生成」機能を利用する場合

標準の「**スクリプト生成**」機能を利用する場合は、次のように、移行元の Management Studio で、スクリプト化をしたいログイン アカウントを右クリックして、[**名前を付けてログインをスクリプト化**] の [**CREATE**] から [**新しいクエリ エディタ ウィンドウ**] をクリックします（画面は SQL Server 2008 の場合ですが、他のバージョンでも同じように操作できます）。

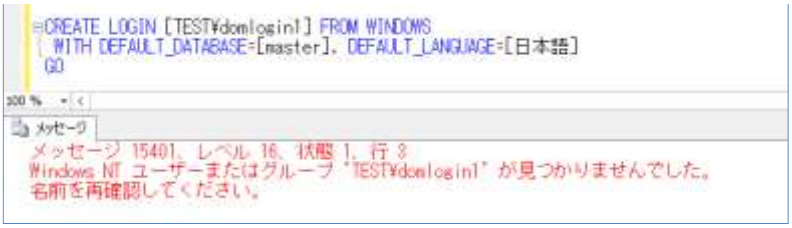



これで **CREATE LOGIN** ステートメントが生成されるので、このスクリプトを移行先（SQL Server 2016 上）で実行すれば、同じログイン アカウントを作成することができます。

移行先に同じログイン アカウントを作成しておけば、次のようにデータベース ユーザーとログイン アカウントのマッピング（紐付け）が復活して、データベース ユーザーおよびオブジェクト権限を移行元と同じように利用できるようになります。


このように、ログイン アカウントが Active Directory ユーザーで、同じドメイン内の別マシンへ移行する場合には、同一のログイン アカウントを作成するだけで、データベース ユーザーとオブジェクト権限を利用できるようになります。

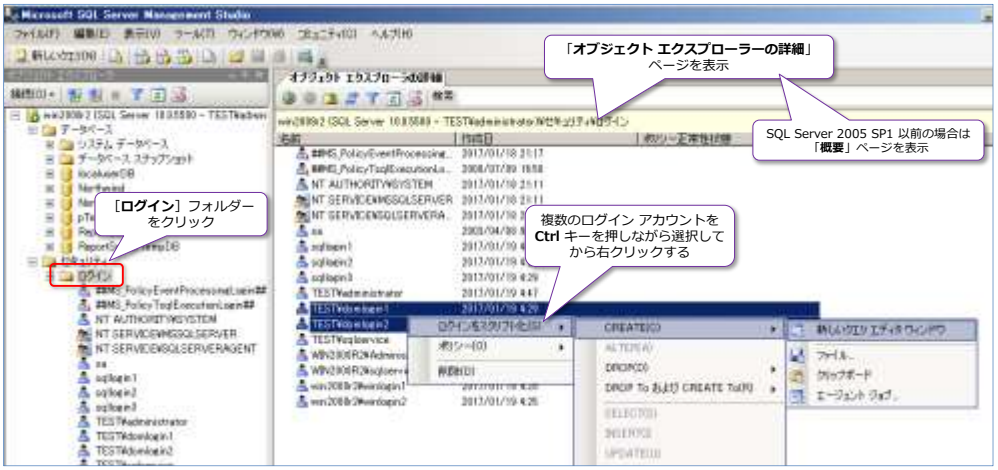
もし、移行元と移行先が異なるドメインに所属していたり、別のワークグループ環境の別マシンに復元したりする場合には、次のように **CREATE LOGIN** ステートメントの実行時にエラーになります。




ドメイン ユーザーに対応したログイン アカウントの作成は、あくまでも同じドメイン内に所属している場合にのみ行うことができます。

**Tips： 複数のログイン アカウントをまとめてスクリプト化したい場合**

移行元で、複数のログイン アカウントをまとめてスクリプト化したい場合は、Management Studio で［**表示**］メニューから［**オブジェクト エクスプローラーの詳細**］をクリックして、詳細ページを表示し、次のように複数のログイン アカウントを **Ctrl キー**を押しながら選択／右クリックします。



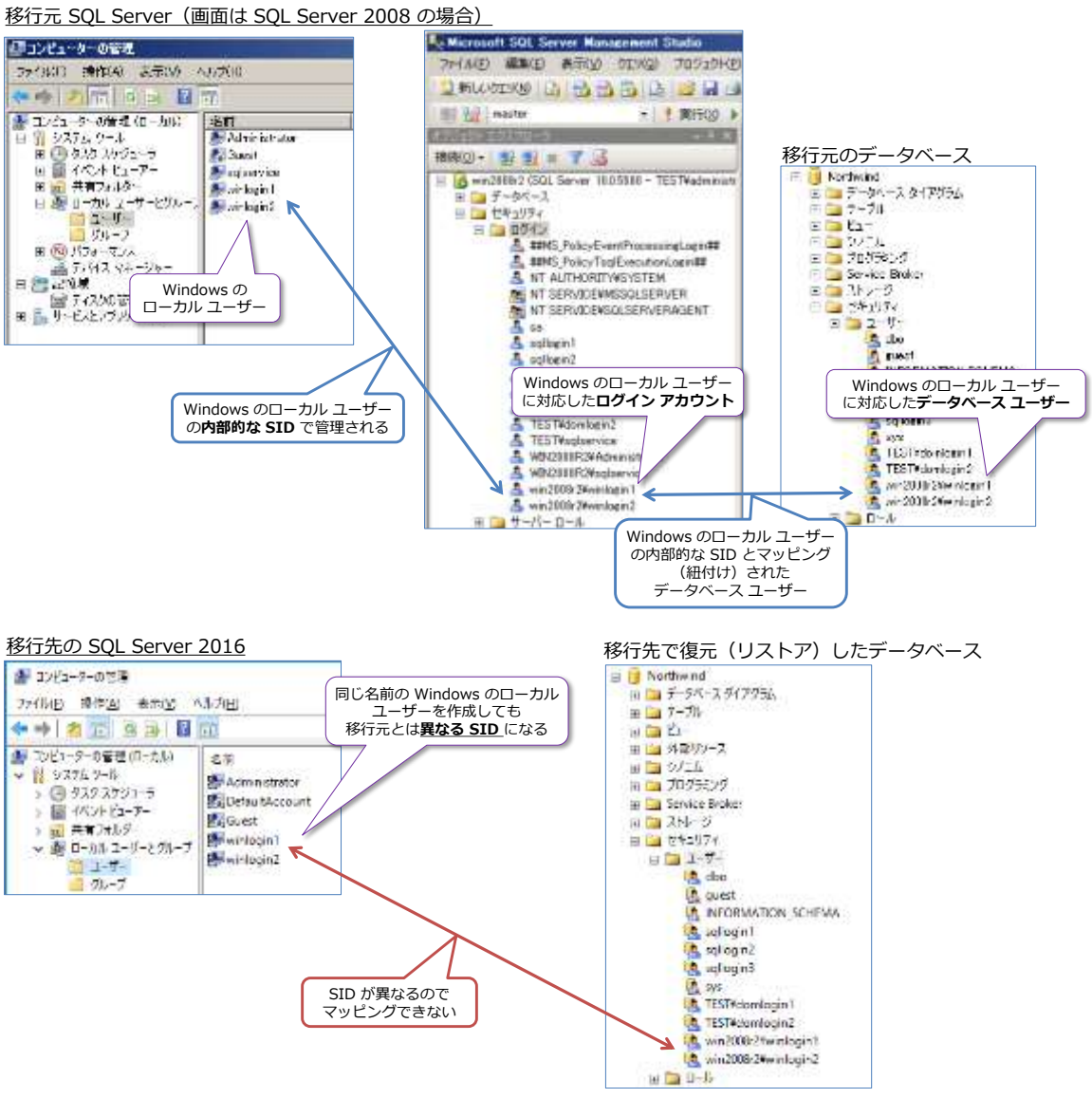

これでまとめてスクリプト化できるので、ログイン アカウントが多い場合に便利です。

なお、SQL Server 2005 の SP1 以前を利用している場合は、[**表示**] メニューから [**概要**] をクリックして、概要ページを表示することで、同様の操作ができます。

## Windows のローカル ユーザーの場合

ログイン アカウントが Windows のローカル ユーザーの場合は、移行先 (SQL Server 2016 上) でデータベース ユーザーを再作成する必要があります。これには、復元したデータベース ユーザーをいったん削除して、手動で同じデータベース ユーザーを作り直さなければなりません。したがって、データベース ユーザーへ設定したオブジェクト権限についても再設定しなければなりません。

この理由は、Windows のローカル ユーザーの場合は、同じ Windows ユーザー (OS 上のユーザー アカウント) を作り直しても、内部的に生成される **SID** (OS 上のユーザーを識別する Security ID) が異なるものが割り当てられるためです。



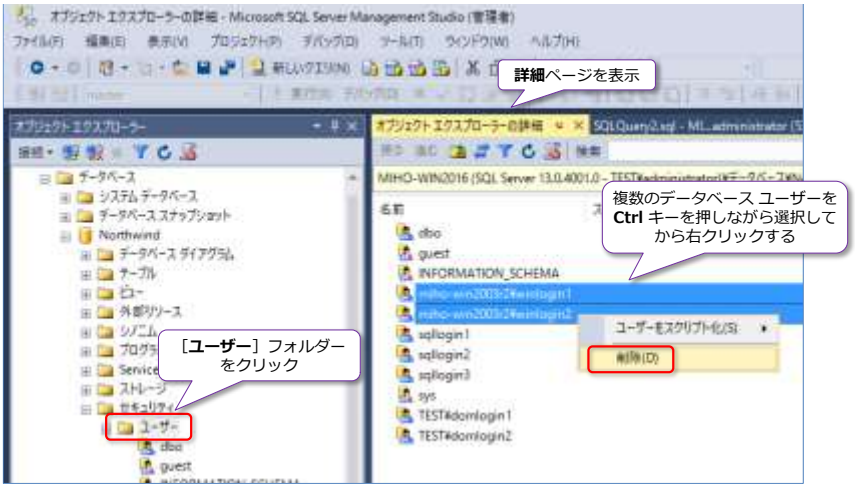
このように、データベース ユーザーは、内部的には、ログイン アカウントの名前とではなく、SID とマッピングされているので、Windows のローカル ユーザーの場合は、復元後のデータベース ユーザーとログイン アカウントをマッピングすることができません（SID が異なる場合には、不明なデータベース ユーザーを解消することができません）。

なお、前述の Active Directory ユーザー（ドメイン ユーザー）の場合に、同一のログイン アカウントを作成するだけで、簡単に再マッピングできたのは、SID が同じものを利用できるためです。同じドメインに所属していれば、共通（同一の SID）のドメイン ユーザーを利用することができます。

したがって、Windows ローカル ユーザーを利用していて、移行先でも同じ名前の Windows ローカル ユーザーを利用したい場合には、手動ですべてを再作成するか、データベース ユーザーやオブジェクト権限を移行元でスクリプト化しておくようにします（詳しくは後述します）。

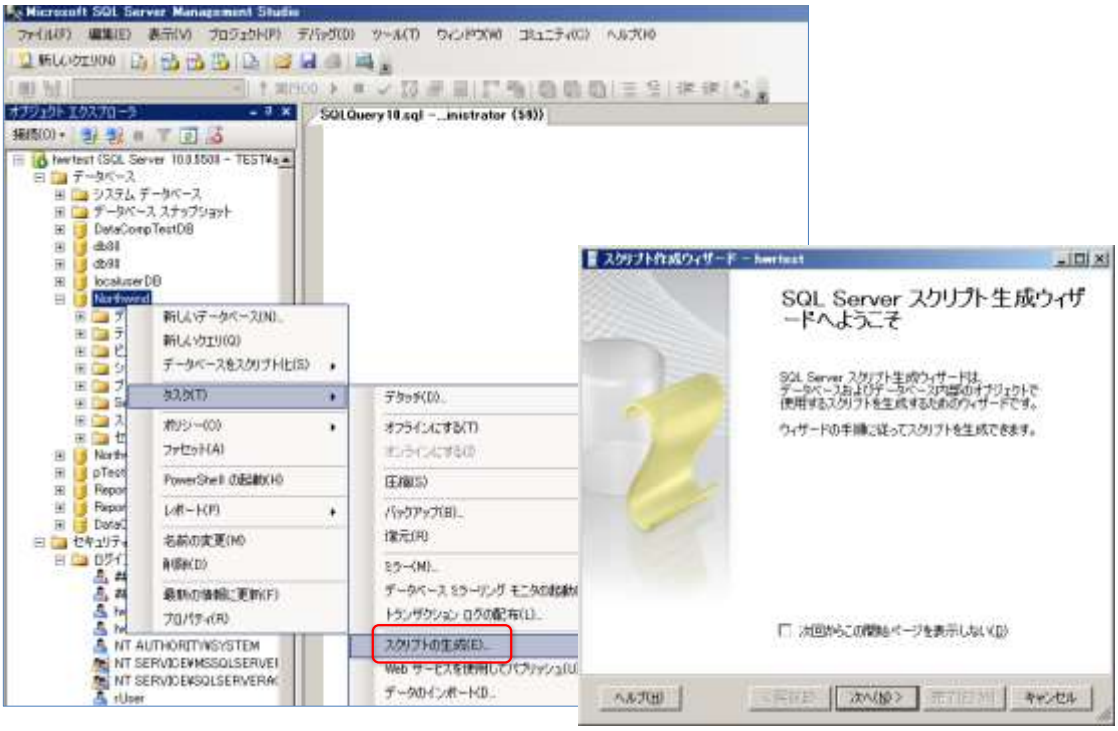
## 復元したデータベース ユーザーの削除

再作成にあたっては、移行先（SQL Server 2016 上）で、復元したデータベース ユーザーをいったん削除する必要がありますが、これは Management Studio で［**表示**］メニューから［**オブジェクト エクスプローラーの詳細**］をクリックして「**詳細**」ページを表示することで、次のように複数のデータベース ユーザーをまとめて削除することができるので便利です。

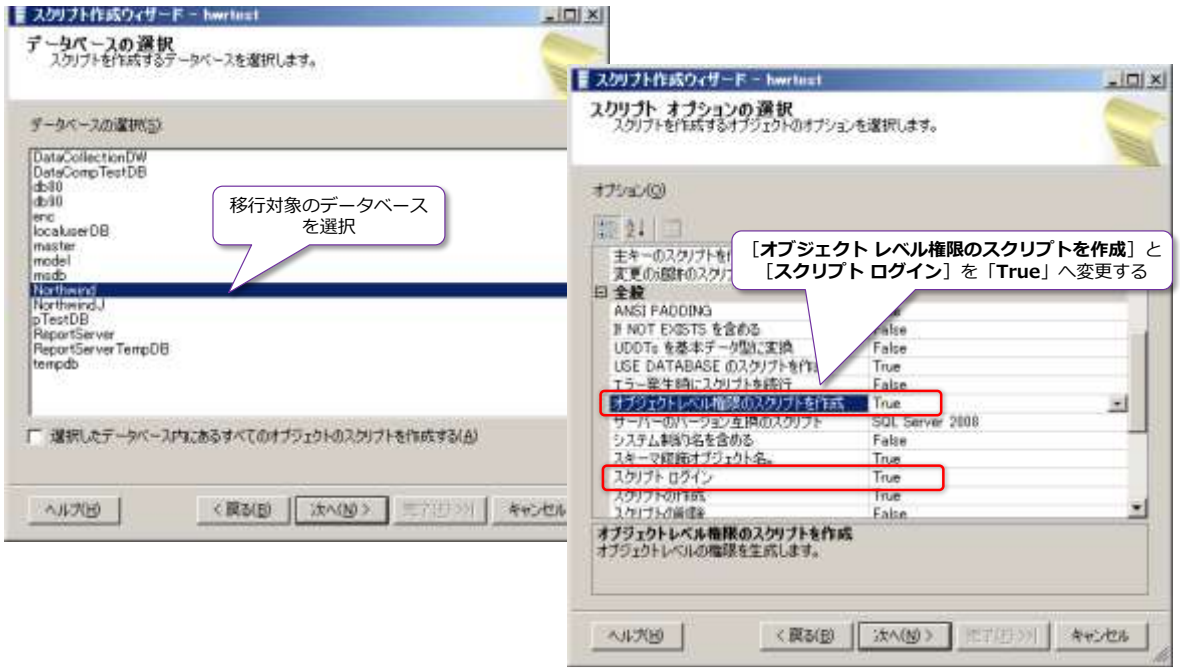



## データベース ユーザーとオブジェクト権限のスクリプト化（スクリプト生成ウィザード）

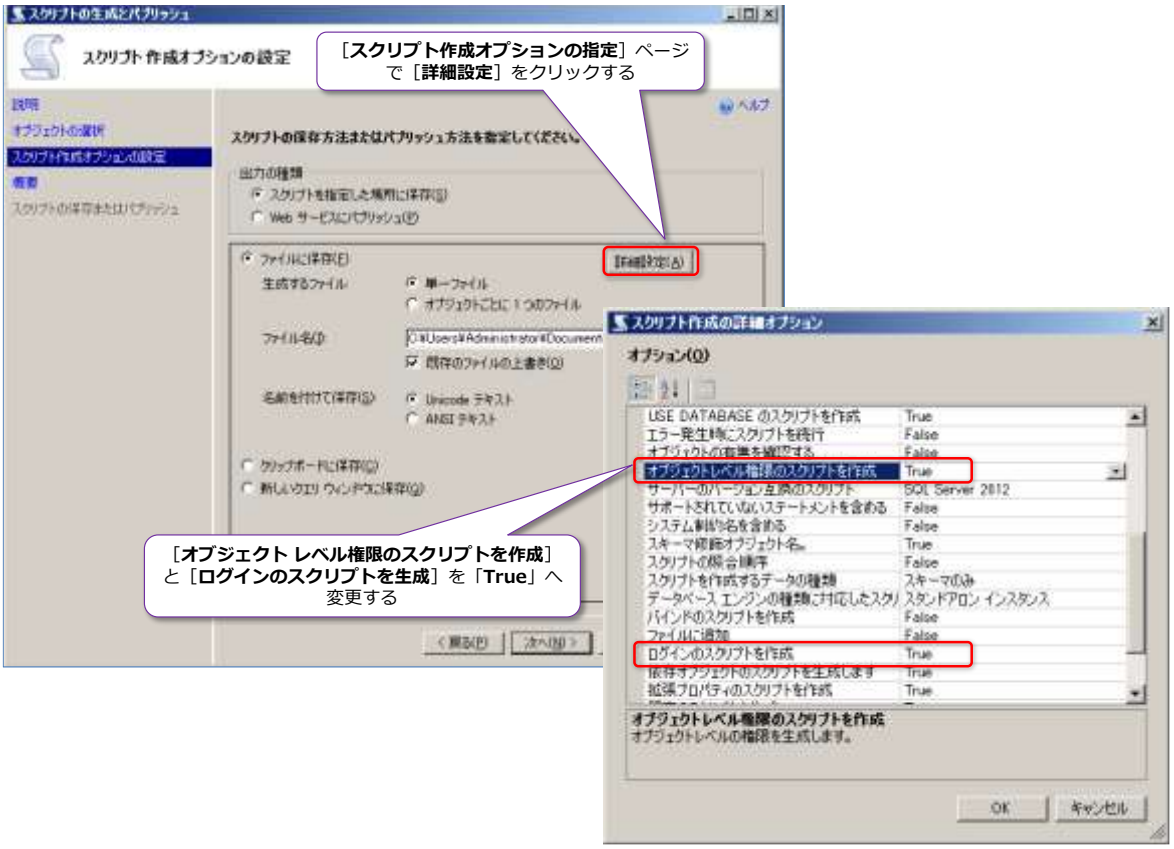
移行元で、ログイン アカウントやデータベース ユーザー、オブジェクト権限をスクリプト化するには、**スクリプト生成ウィザード**を利用すると便利です。このウィザードを利用するには、次のようにデータベースを右クリックして、［**タスク**］メニューの［**スクリプトの生成**］をクリックします（画面は SQL Server 2008 ですが、他のバージョンでも同じように操作できます）。



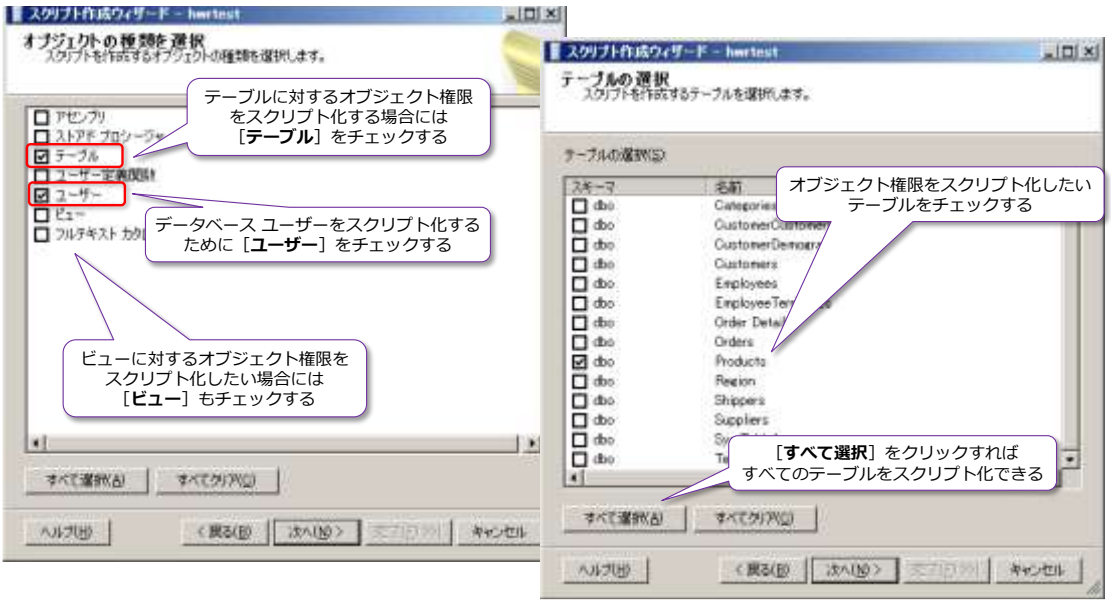

スクリプト生成ウィザードでは、次のように［**スクリプト オプションの選択**］ページで、［**オブジェクト レベル権限のスクリプトを作成**］と［**スクリプト ログイン**］を「**True**」へ変更することで、データベース ユーザーとオブジェクト権限、データベース ユーザーに対応したログイン アカウントをスクリプト化することができます（SQL Server 2005 と 2008 の場合）。




SQL Server 2008 R2 と 2012、2014 の場合は、次のように操作します。



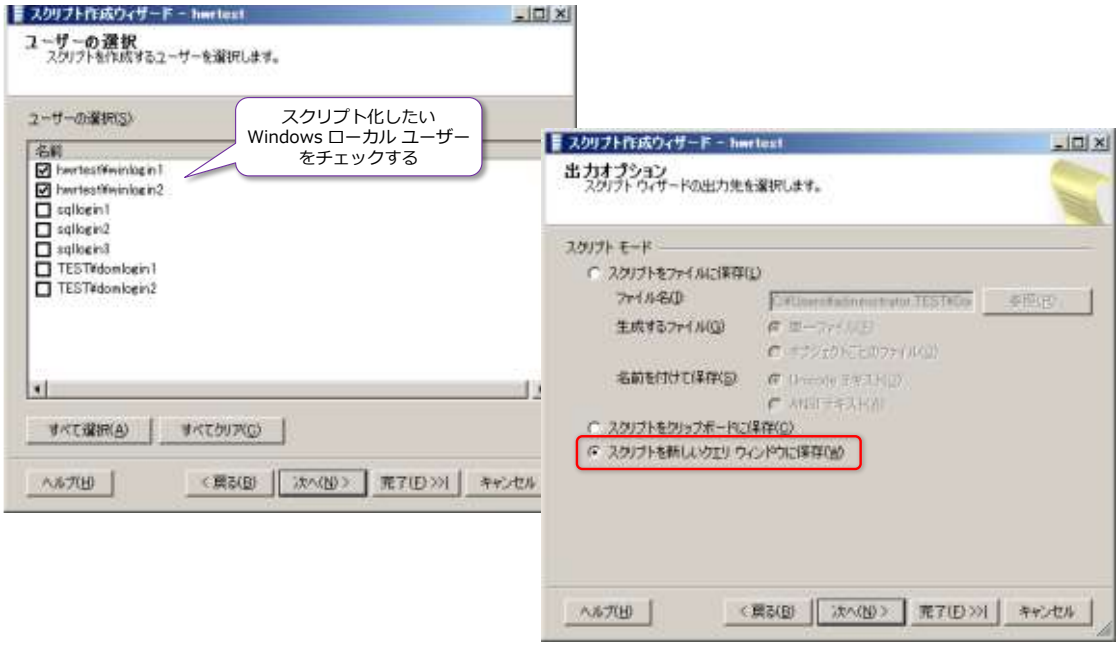

次の［**オブジェクトの種類を選択**］ページでは、データベース ユーザーをスクリプト化するために［**ユーザー**］をチェックし、テーブルに対するオブジェクト権限をスクリプト化する場合には［**テーブル**］をチェックします。ビューやストアド プロシージャに対するオブジェクト権限をスクリプト化するには、それらもチェックしておきます。



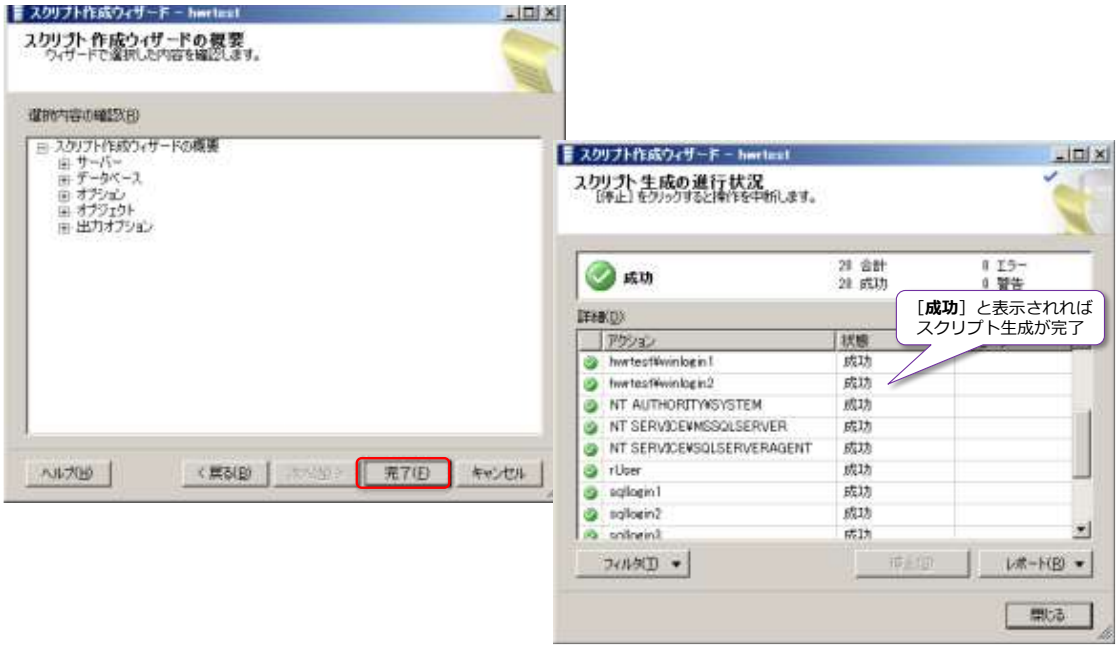

［**テーブルの選択**］ページでは、オブジェクト権限をスクリプト化したいテーブルをチェックします（画面は **Products** テーブルをチェック。［**すべて選択**］ボタンをクリックした場合は、すべてのテーブルを選択することができます）。これらの画面は、SQL Server 2005／2008 の場合ですが、SQL Server 2008 R2／2012／2014 の場合は、ウィザードの最初のページで、ユーザーやテ

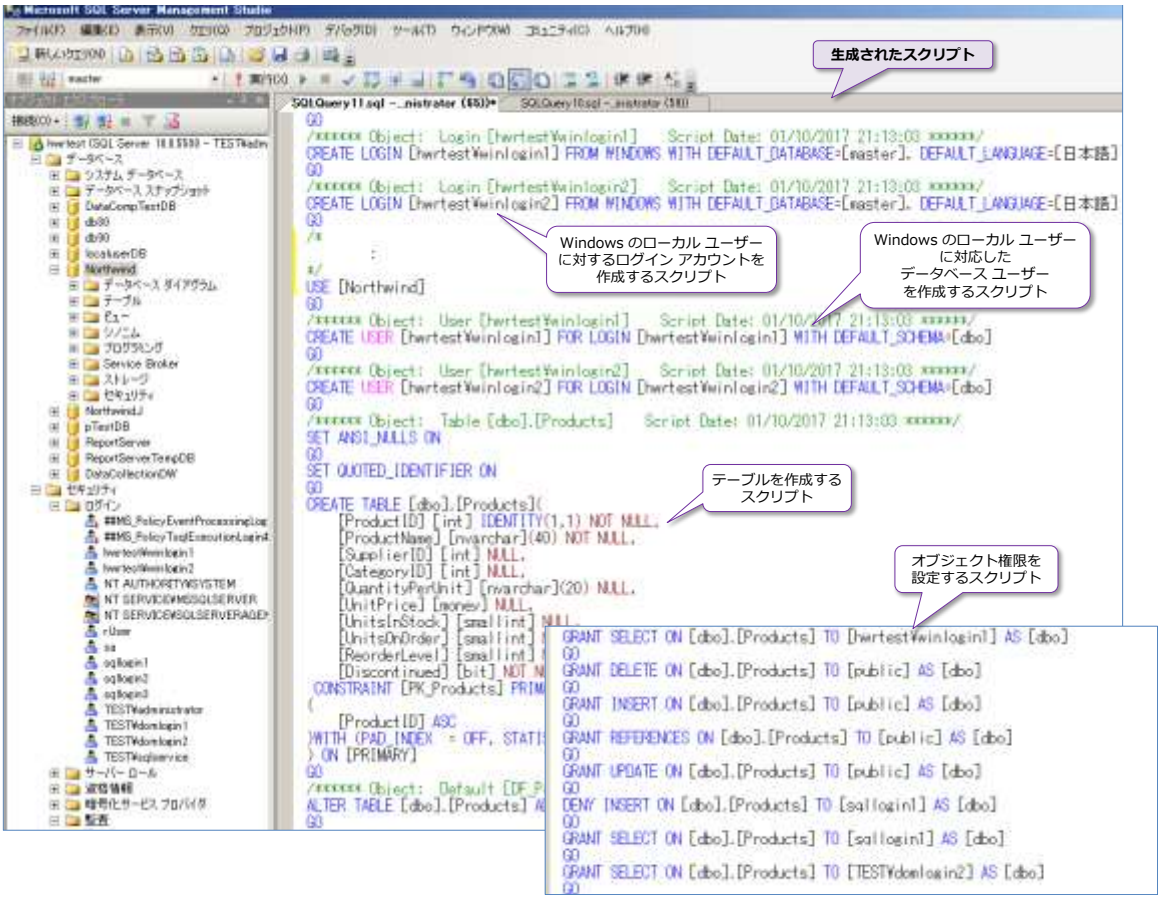
ーブルを選択することができます。

SQL Server 2005／2008 の場合は、次の［**ユーザーの選択**］ページで、スクリプト化したいデータベース ユーザーを選択します。




［**出力オプション**］ページで、［**スクリプトを新しいクエリ ウィンドウに保存**］を選択すれば、クエリ エディターにスクリプトを生成することができます。あとは、次のように［**完了**］ボタンをクリックすれば、スクリプト生成が開始されます。





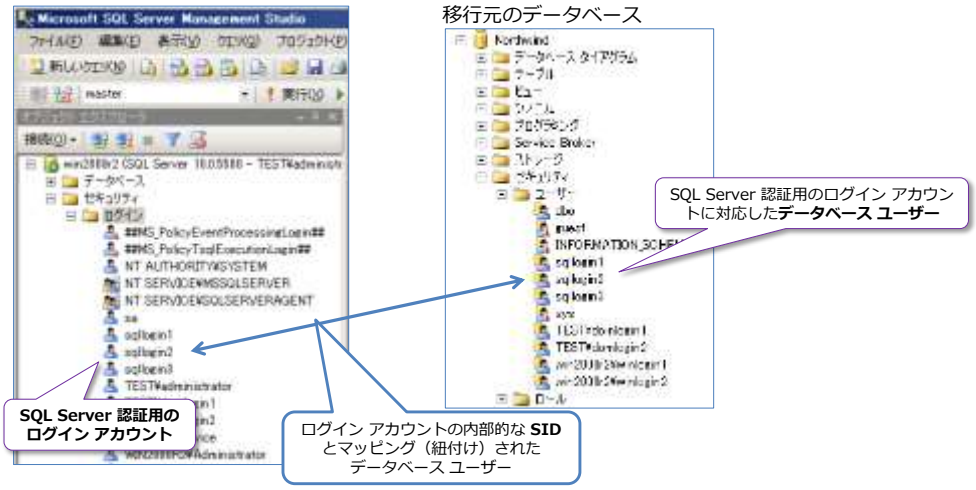

テーブルを作成するスクリプト（**CREATE TABLE**）などが余分に出力されてしまいますが、ログイン アカウントを作成するスクリプト（**CREATE LOGIN**）や、データベース ユーザーを作成するスクリプト（**CREATE USER**）、オブジェクト権限を設定するスクリプト（**GRANT**、**DENY** など）が生成されていることを確認できると思います。Windows のローカル ユーザーの場合は、「**サーバー名¥ユーザー名**」形式の名前になるので、サーバー名の部分を移行先のサーバー名へ変更すれば、移行先で同じユーザー、同じオブジェクト権限を再設定できるようになります。

## SQL Server 認証用のログイン アカウントの場合

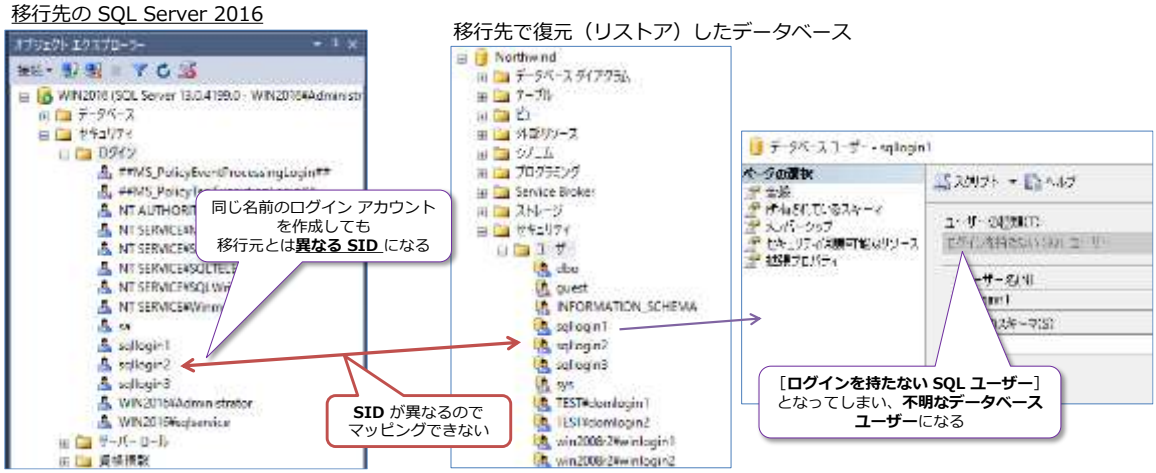
ログイン アカウントが SQL Server 認証用の場合は、SQL Server 標準のツール（スクリプト生成機能や、データベース コピー ウィザード、ログイン転送タスク ツールなど）では、同一のログイン アカウントを作成することができないことに注意する必要があります。標準ツールでは、パスワードと **SID**（SQL Server 認証用のログイン アカウントに内部的に割り当てられた Security ID）を生成／複製することができません（パスワードと SID は、移行元と移行先で異なるものが生成されてしまいます）。

SQL Server 認証用のログイン アカウントの場合にも、Windows 認証用と同様、データベース ユーザーは、SID とマッピングされているので、SID が異なる場合には、不明なデータベース ユーザーを解消することができません。

移行元 SQL Server（画面は SQL Server 2008 の場合）

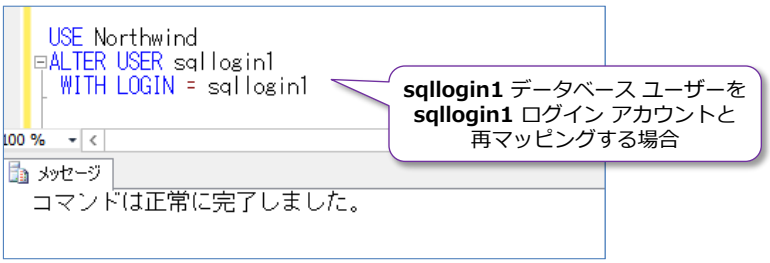



移行先の SQL Server 2016



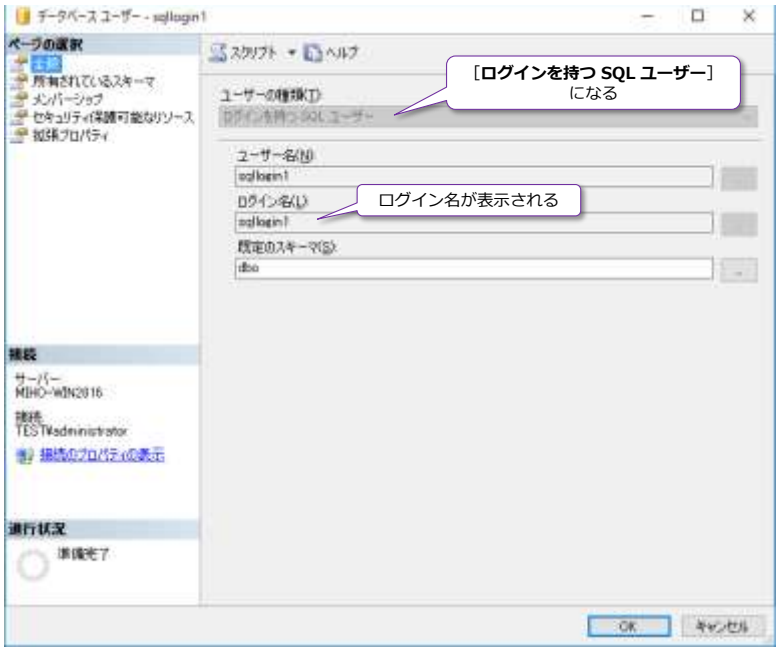

不明なデータベース ユーザーを解消するには（データベース ユーザーとログイン アカウントを再マッピングするには）、移行先（SQL Server 2016 上）で、同じ名前／パスワードのログイン アカウントを作成した後に、次のように **ALTER USER** ステートメントを実行します。

```
USE データベース名
ALTER USER 不明なデータベース ユーザーの名前
 WITH LOGIN = 移行先に作成した新しいログイン アカウントの名前
```




このように、**ALTER USER** ステートメントを実行すると、移行先で作成した新しいログイン アカウントと、復元したデータベース ユーザーを再マッピングすることができます（移行元の古い SID に紐付いていたものを、移行先の新しい SID へ紐付くように更新することができます）。

再マッピングが成功すると、データベース ユーザーのプロパティは、次のように表示されます。




なお、**ALTER USER** ステートメントの代わりに、次のように **sp_change_users_login** ストアド プロシージャを実行しても、再マッピングを行うことができますが、このストアド プロシージャは将来のバージョンでは削除される予定です。

```
USE データベース名
EXEC sp_change_users_login 'Update_One', '不明なユーザー名', '新しいログイン名'
```

**ALTER USER** ステートメントで不明なデータベース ユーザーの再マッピングを行う方法は、データベース ユーザーごとに実行する必要があるので、データベース ユーザーの数が多い場合には、大変になります（同じパスワードのログイン アカウントを作成する手間もかかります）。これらを解決してくれる方法が、次に説明する **sp_help_revlogin** ストアド プロシージャになります。これを利用すれば、同じ名前／パスワード かつ、同じ SID のログイン アカウントを一括で作成することができます。

### 同じ名前／パスワードで、同じ SID のログイン アカウントの作成： sp_help_revlogin

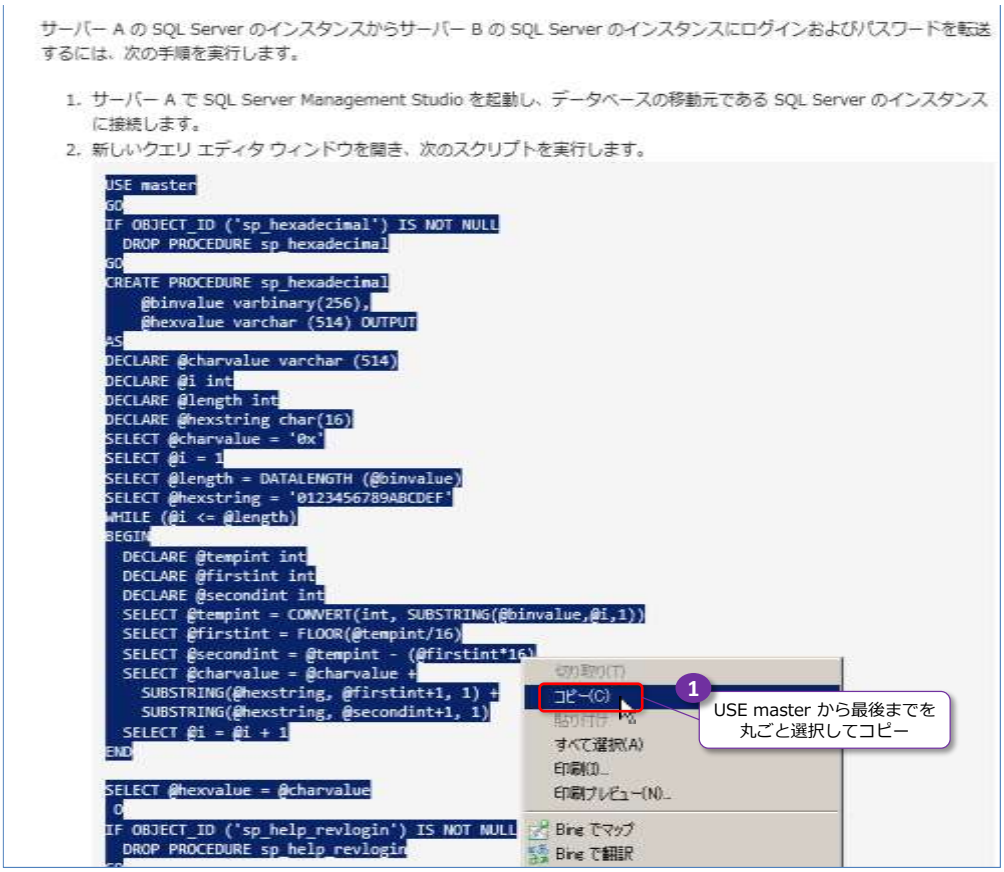
**sp_help_revlogin** は、同じ名前／パスワードで、同じ SID のログイン アカウントを一括で作成することができる大変便利なストアド プロシージャです。移行元と移行先で、同じ SID のログイン アカウントを移行先で作成すれば、前述の ALTER USER ステートメントを実行しなくても、不明なデータベース ユーザーを解消することができるからです。

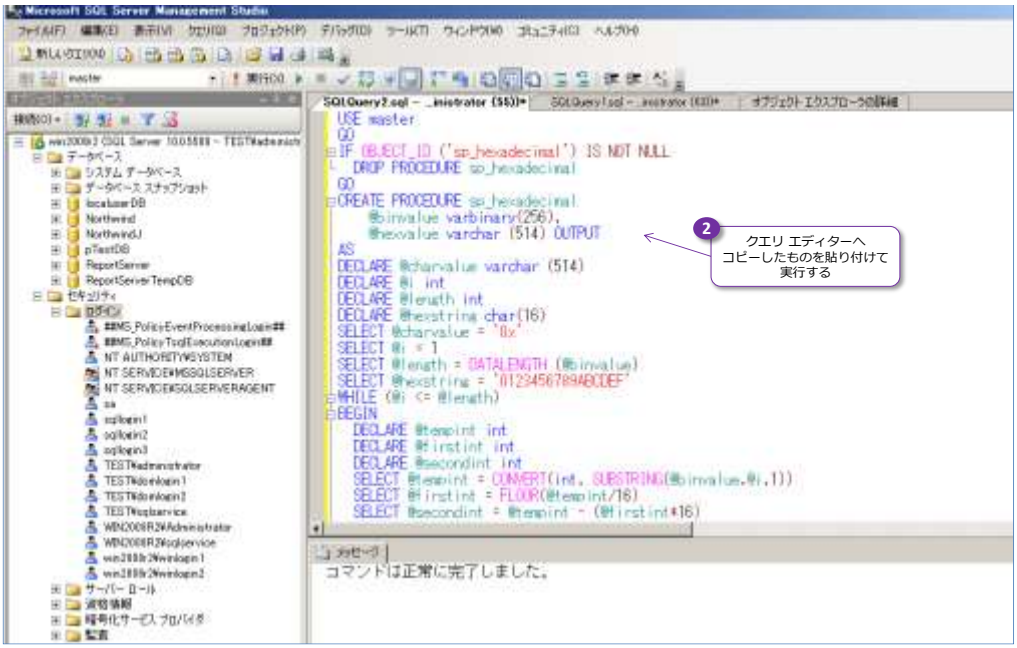
**sp_help_revlogin** ストアド プロシージャは、マイクロソフトのサポート技術情報（KB：Knowledge Base）の文書番号 **918992** で提供されています。

SQL Server 2005 のインスタンス間でログインおよびパスワードを転送する方法
http://support.microsoft.com/kb/918992

この文書で提供されるスクリプトを、次のように**丸ごとコピー**して（手順 2 の USE master からスクリプトの最後までを選択してコピー）、移行元の SQL Server のクエリ エディターから実行

すれば、**sp_help_revlogin** ストアド プロシージャを作成することができます。

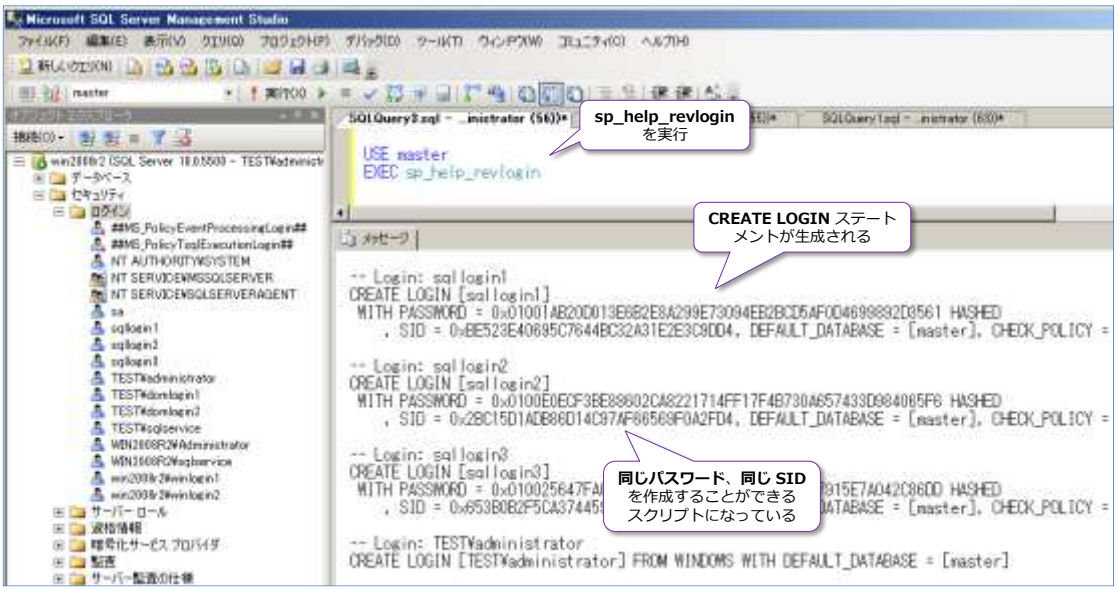



上の画面は SQL Server 2008 の場合ですが、このスクリプトは SQL Server 2005 や 2008 R2、2012、2014 でも同じように利用することができます。

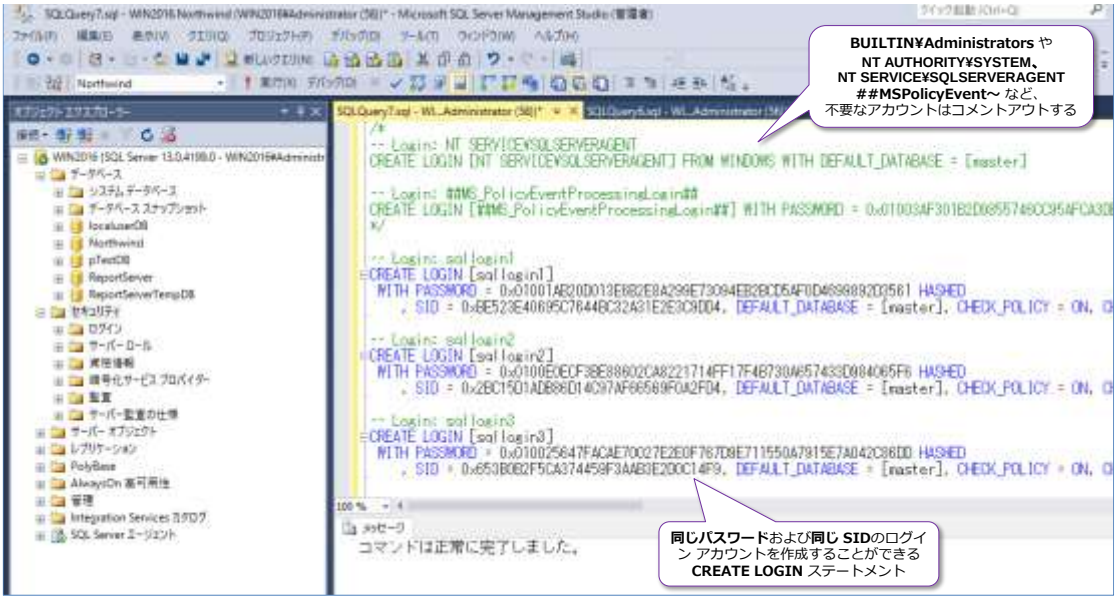
これにより、**master** データベース内に **sp_help_revlogin** ストアド プロシージャが作成されるので、次のように実行すれば、同一のパスワード／SID のログイン アカウントを作成するためのスクリプトを生成することができます。

```
USE master
EXEC sp_help_revlogin
```



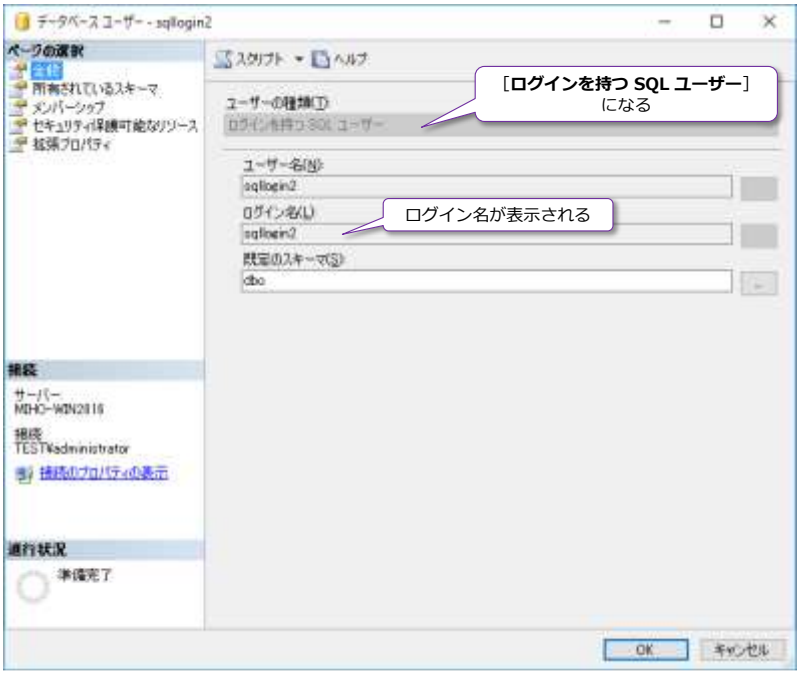

実行結果には、**CREATE LOGIN** ステートメントが生成されて、**WITH PASSWORD=** や **SID=** 付きのスクリプトが生成されて、同じパスワードおよび同じ SID のログイン アカウントを作成できるようになります。

なお、このスクリプトには、Windows 認証用のログイン アカウントを作成するものや、SQL Server の内部ユーザー（**NT AUTHORITY¥SYSTEM** や **NT SERVICE¥MSSQLSERVER**、**NT SERVICE¥SQLSERVERAGENT**、**##MSPolicyEvent～**、SQL Server 2005 の場合は **SQLServer2005MSSQLUser～** や、**SQLServer2005SQLAgetUser～** など）、グループなどを作成するための **CREATE LOGIN** ステートメント（SQL Server 2005 なら **BUILTIN¥Administrators** など）も生成されるので、それらの不要なログイン アカウントはコメントアウトして、必要なログイン アカウントのみを移行先（SQL Server 2016 上）で実行／再作成するようにします。



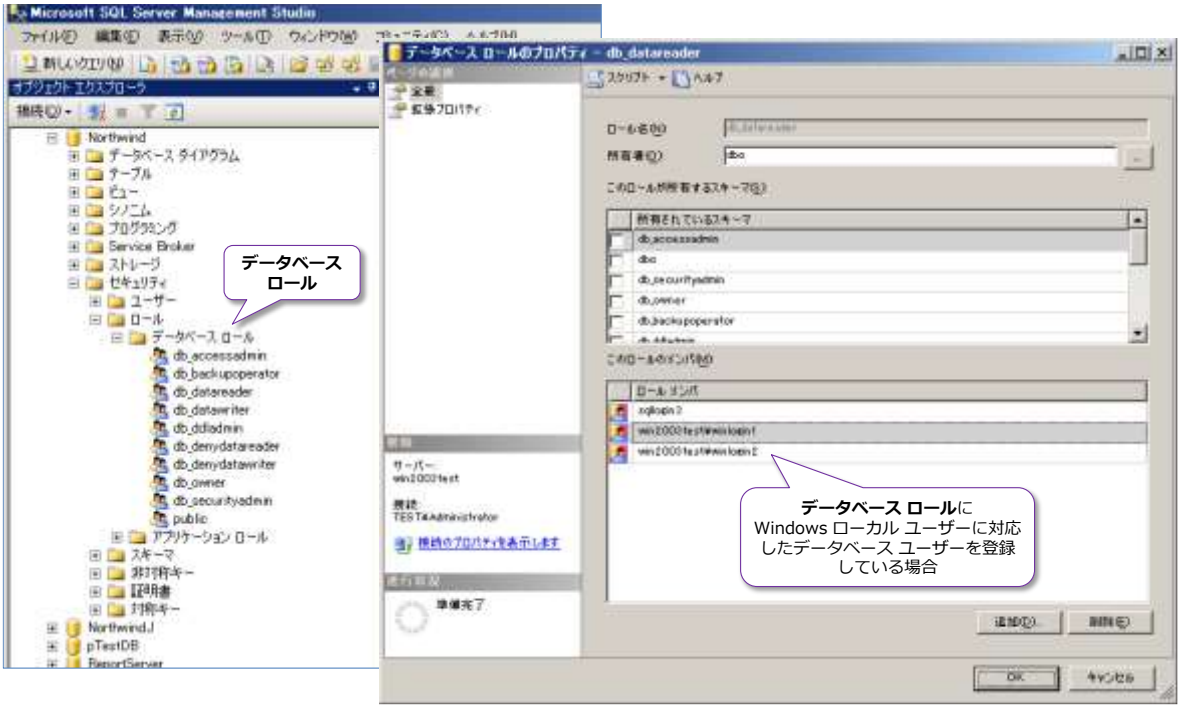
このように、移行元と移行先で、同じ SID のログイン アカウントを作成すれば、不明なデータベース ユーザーが解消されて、データベースの復元後にも、データベース ユーザーやオブジェクト権限をそのまま利用することができます。

移行元と移行先で同じ *SID* のログイン アカウントを作成すると、不明なデータベース ユーザーが解消される



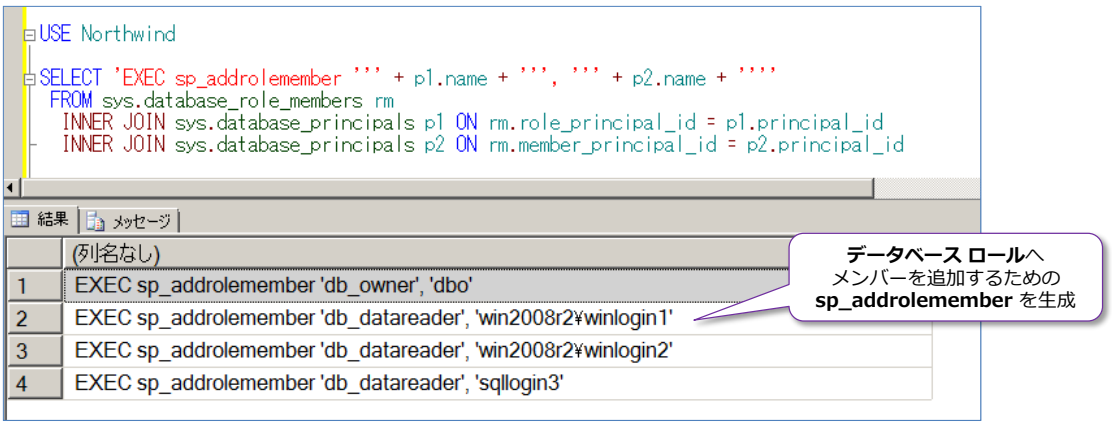

## データベース ロールのスクリプト化

データベース ロールを利用して、データベース ユーザーをグループ化している場合には、データベース ロールの設定もスクリプト化しておく必要があります。



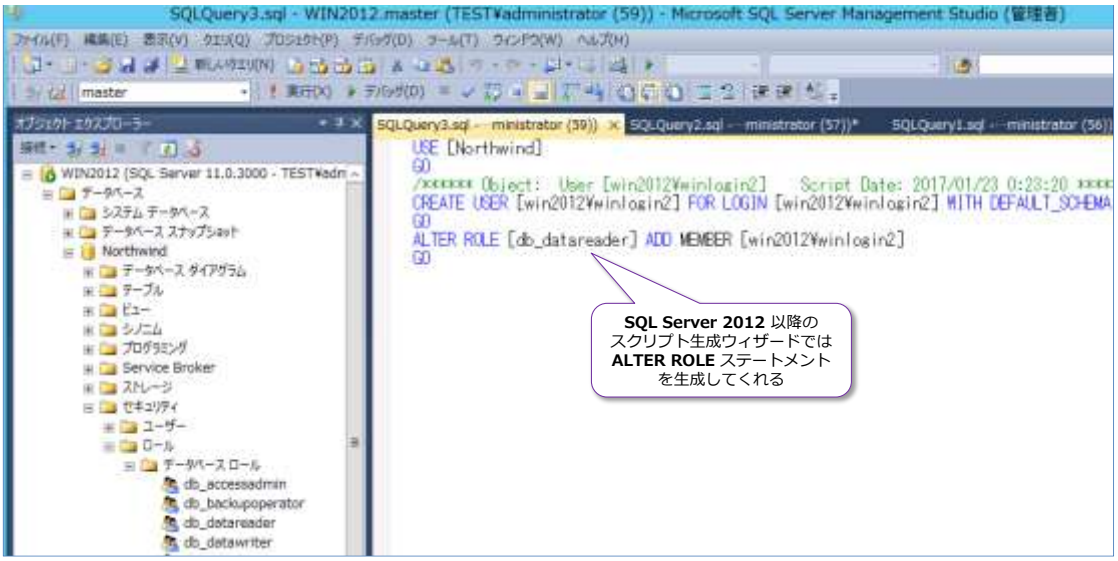
前述の**スクリプト生成ウィザード**では、SQL Server 2008／2008 R2 の場合は、データベース ロールの設定をスクリプト化することができないので、データベース ロールのメンバー情報を参照できる **database_role_members** システム ビューとデータベース ユーザーの情報を参照できる **database_principals** システム ビューを、次のように利用します。

```
SELECT 'EXEC sp_addrolemember ''' + p1.name + ''', ''' + p2.name + ''''
 FROM sys.database_role_members rm
  INNER JOIN sys.database_principals p1 ON rm.role_principal_id = p1.principal_id
  INNER JOIN sys.database_principals p2 ON rm.member_principal_id = p2.principal_id
```



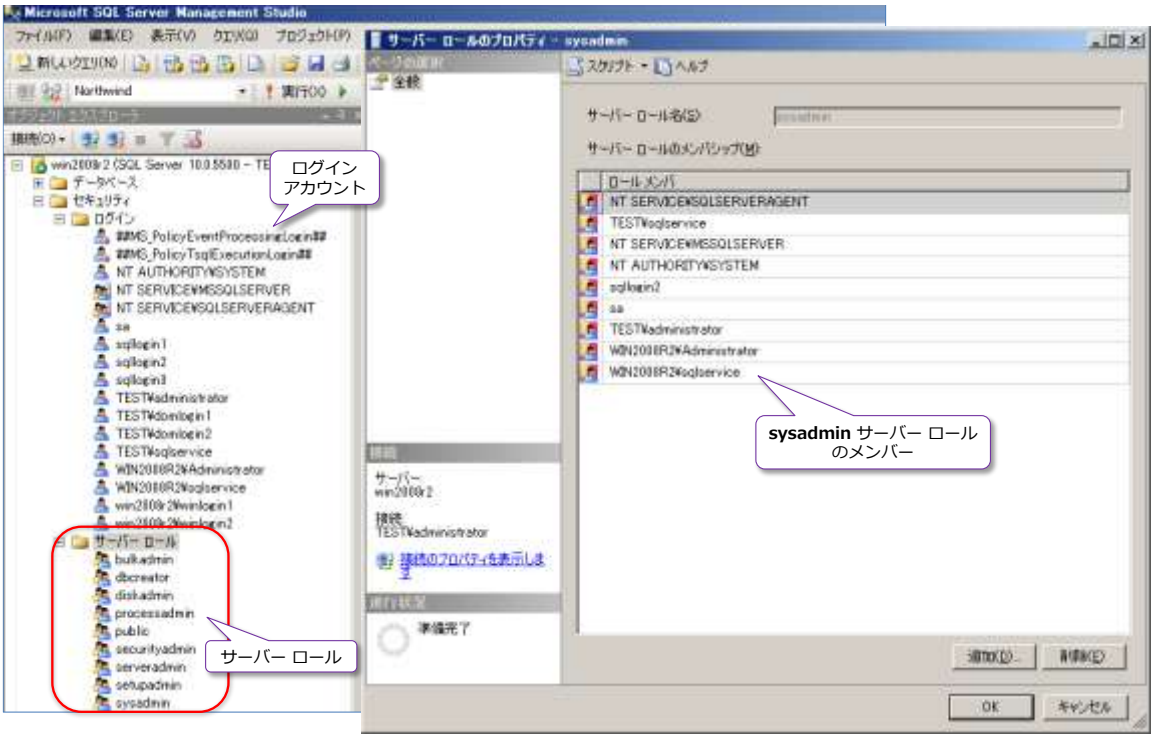

これで、データベース ロールへメンバーを追加することできる **sp_addrolemember** を生成することができるので、これを新規マスター側で実行すれば、同じ設定にすることができます。

SQL Server 2012 以降を利用している場合は、スクリプト生成ウィザードが **ALTER ROLE** ステートメントを生成してくれるので、これを移行先で実行すれば、データベース ロールの設定を同じように利用することができます。



## 5.14 サーバー ロール設定の移行

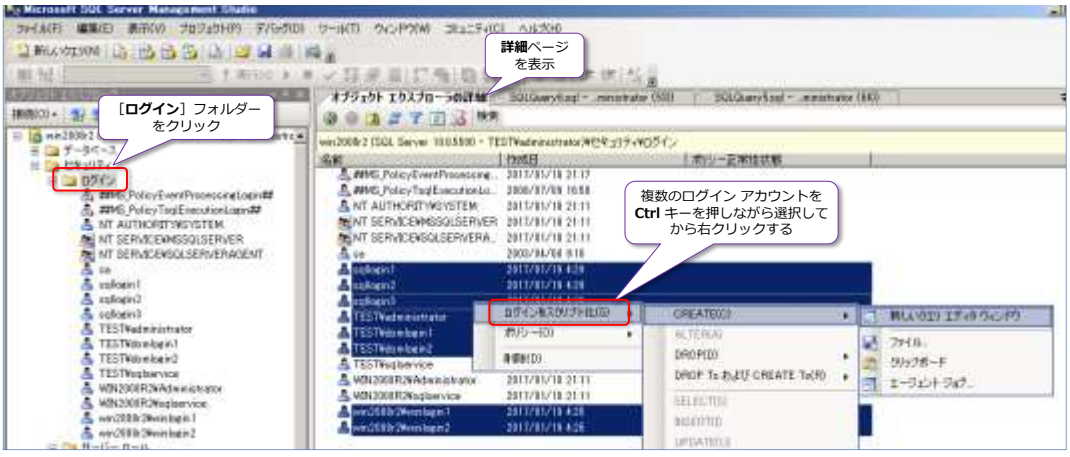
サーバー ロール（**sysadmin** や **securityadmin**、**serveradmin** など）は、ログイン アカウントをグループ化するための機能なので、この設定も **master** データベース（システム データベース）内に格納されています（以下の画面は SQL Server 2008 の場合）。




したがって、サーバー ロールの設定は、移行先で再設定をする必要があります。

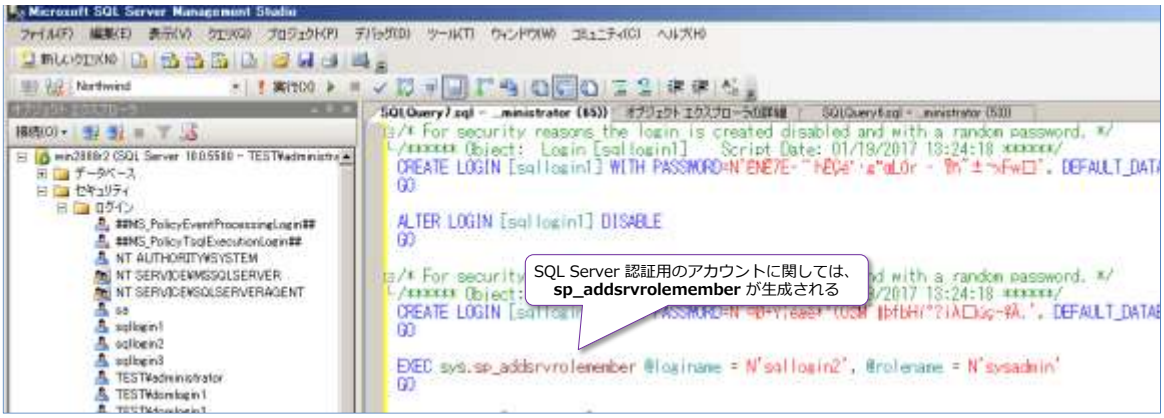
### サーバー ロール設定のスクリプト生成

サーバー ロールの設定は、**SQL Server 2005／2008／2008 R2** の標準のスクリプト生成機能だと、SQL Server 認証用のログイン アカウントしかスクリプトを生成することができません（Windows 認証用のログイン アカウントに関してはスクリプト生成ができません）。これは、次のような状況です（画面は SQL Server 2008 の場合）。



Management Studio の［**表示**］メニューから［**オブジェクト エクスプローラーの詳細**］をクリックして詳細ページを表示して（SQL Server 2005 の SP1 以前を利用している場合は［**概要**］をクリックして概要ページを表示して）、［**ログイン**］フォルダーで複数のログイン アカウントを Ctrl キーを押しながら選択し、右クリックして［**名前を付けてスクリプト化**］からスクリプト生成を行っています。

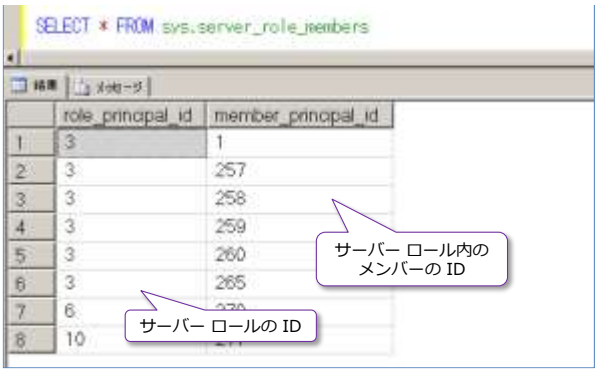
これで、ログイン アカウントを作成する **CREATE LOGIN** ステートメントとともに、サーバー ロールを設定するための **sp_addsrvrolemember**（サーバー ロールへメンバーを追加するためのシステム ストアド プロシージャ）が生成されるのですが、次のように SQL Server 認証用のログイン アカウントに対してはうまく生成されるのですが、Windows 認証用のログイン アカウントに関しては生成させることができません。




Windows 認証用のログイン アカウントに対しても、サーバー ロールの設定をスクリプト化するには、次の 2 つの方法があります。

- **server_role_members** システム ビュー（サーバー ロールとログイン アカウントの対応を確認しているビュー）から **sp_addsrvrolemember** を生成する
- **SQL Server 2016 の Management Studio** をインストールしたマシンから、移行元の SQL Server へリモート接続をして、スクリプトを生成する（SQL Server 2012 または 2014 の Management Studio からでも同様の操作が可能）

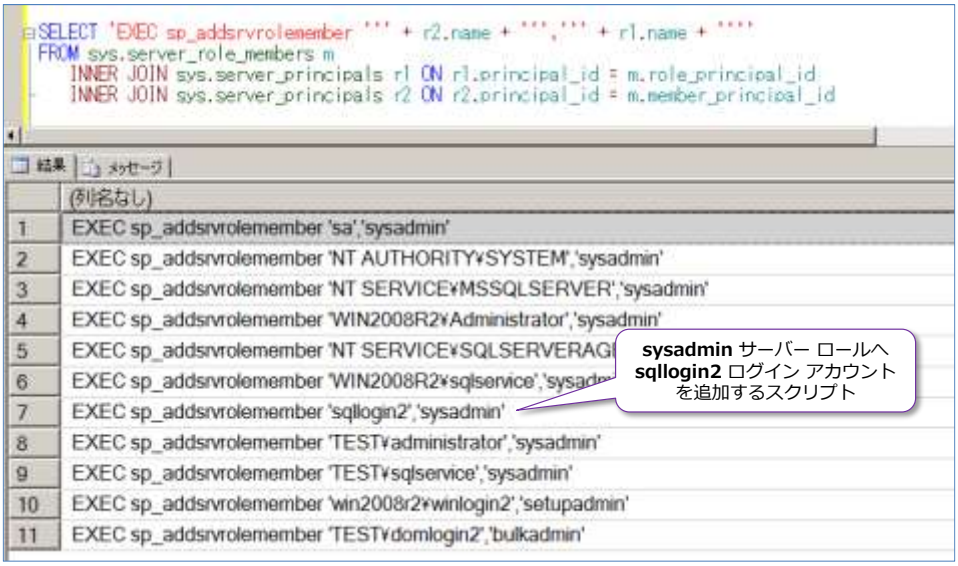
1 つ目の方法は、サーバー ロールとログイン アカウントの対応を確認することができる「**server_role_members**」システム ビューを利用する方法です。このビューでは、次のようにサーバー ロールとログイン アカウントの対応を確認できます。




この **ID**（**principal_id**）は、**server_principals** システム ビューと **JOIN** することで、サー

バー ロールの名前やログイン アカウントの名前を取得することができるので、次のようにサーバー ロールを設定するためのスクリプトを生成することができます。

```
SELECT 'EXEC sp_addsrvrolemember ''' + r2.name + ''',''' + r1.name + ''''
FROM sys.server_role_members m
   INNER JOIN sys.server_principals r1 ON r1.principal_id = m.role_principal_id
   INNER JOIN sys.server_principals r2 ON r2.principal_id = m.member_principal_id
```




| | (列名なし) |
|---|---|
| 1 | EXEC sp_addsrvrolemember 'sa','sysadmin' |
| 2 | EXEC sp_addsrvrolemember 'NT AUTHORITY¥SYSTEM','sysadmin' |
| 3 | EXEC sp_addsrvrolemember 'NT SERVICE¥MSSQLSERVER','sysadmin' |
| 4 | EXEC sp_addsrvrolemember 'WIN2008R2¥Administrator','sysadmin' |
| 5 | EXEC sp_addsrvrolemember 'NT SERVICE¥SQLSERVERAG |
| 6 | EXEC sp_addsrvrolemember 'WIN2008R2¥sqlservice','sysad |
| 7 | EXEC sp_addsrvrolemember 'sqllogin2','sysadmin' |
| 8 | EXEC sp_addsrvrolemember 'TEST¥administrator','sysadmin' |
| 9 | EXEC sp_addsrvrolemember 'TEST¥sqlservice','sysadmin' |
| 10 | EXEC sp_addsrvrolemember 'win2008r2¥winlogin2','setupadmin' |
| 11 | EXEC sp_addsrvrolemember 'TEST¥domlogin2','bulkadmin' |

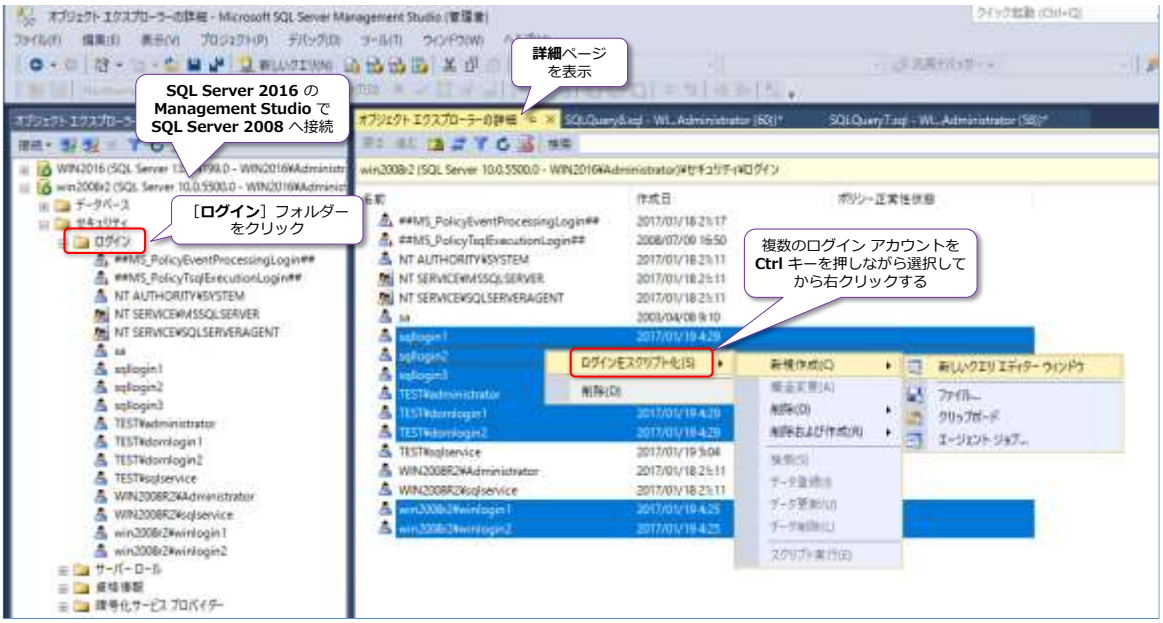
この生成された **sp_addsrvrolemember** システム ストアド プロシージャを、移行先（SQL Server 2016 上）で実行すれば、サーバー ロールの設定を移行することができます。

なお、上記の画面は、SQL Server 2008 の場合ですが、SQL Server が利用している内部ユーザー（**NT AUTHORITY¥SYSTEM** や **NT SERVICE¥MSSQLSERVER**、**NT SERVICE¥SQL SERVERAGENT**、SQL Server 2005 の場合は **SQLServer2005MSSQLUser～** や、**SQLServer2005SQLAgetUser～**、**BUILTIN¥Administrators** など）を **sysadmin** サーバー ロールに追加するものも含まれてしまっているので、不要なものは取り除いて、移行先（SQL Server 2016 上）で実行するようにします。

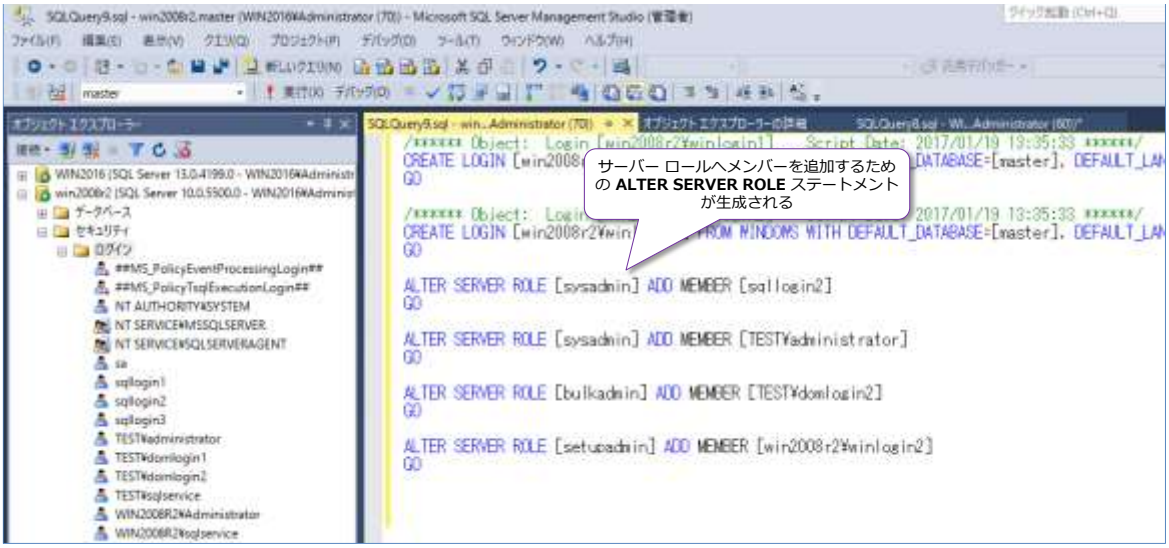
## SQL Server 2016 の Management Studio を利用する方法

サーバー ロールの設定をスクリプト化する 2 つ目の方法は、**SQL Server 2016 の Management Studio** をインストールしたマシンから、移行元の SQL Server へリモート接続をして、スクリプトを生成します。SQL Server 2012 以降の Management Studio であれば、Windows 認証用のログイン アカウントに関しても、サーバー ロールの設定をスクリプト化することができるので、その機能を利用します。

これを行うには、SQL Server 2016 の Management Studio で［**表示**］メニューから［**オブジェクト エクスプローラーの詳細**］をクリックして詳細ページを表示し、次のように［**ログイン**］フォルダーで複数のログイン アカウントを Ctrl キーを押しながら選択して、右クリックし、［**ログインをスクリプト化**］から、スクリプト生成を行います。



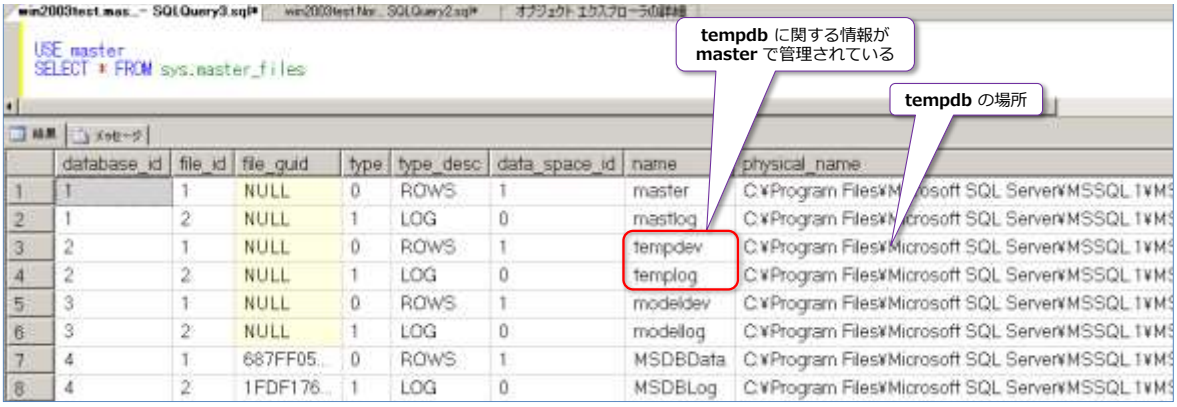

これで、ログイン アカウントを作成する **CREATE LOGIN** ステートメントとともに、サーバー ロールを設定するための **ALTER SERVER ROLE** ステートメントが生成されます（SQL Server 2012 以降では、**sp_addsrvrolemember** の代わりにこのステートメントでサーバー ロールへのメンバーの追加ができるようになりました）。




生成された **ALTER SERVER ROLE** ステートメントを、移行先（SQL Server 2016 上）で実行すれば、サーバー ロールの設定を移行することができます。

## 5.15 tempdb の設定の移行

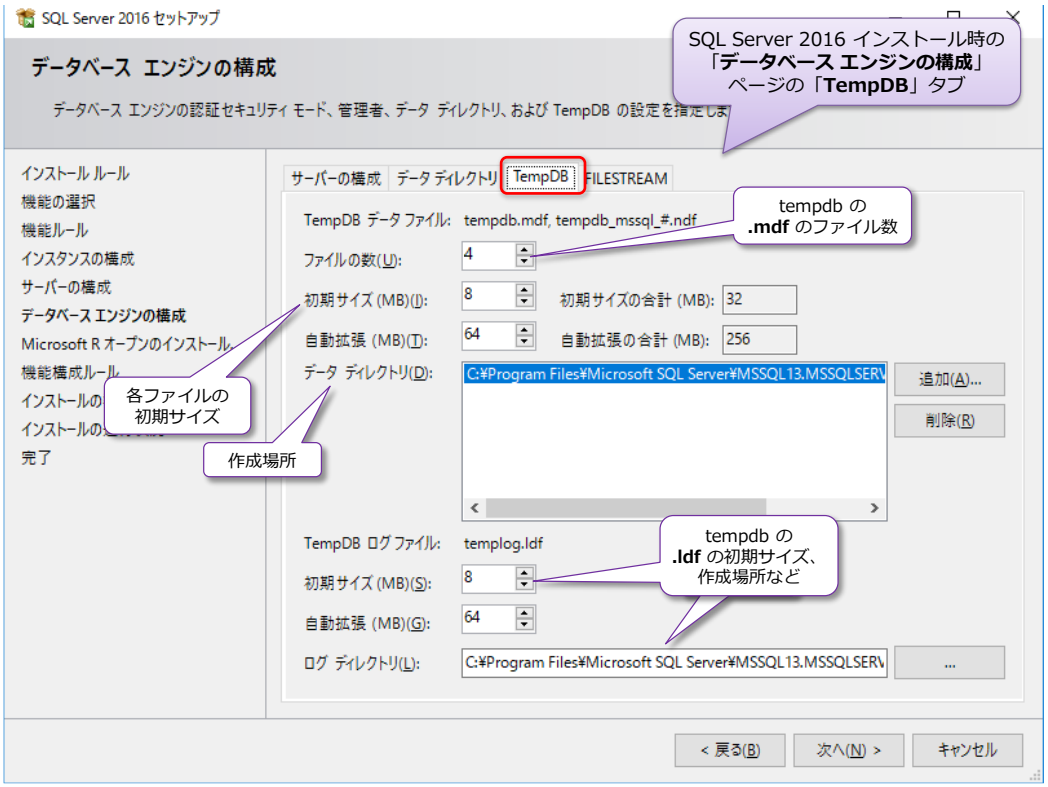
tempdb の設定も **master** データベース（システム データベース）内に格納されているので、移行元で設定を変更している場合には、別途移行を行う必要があります。




| | database_id | file_id | file_guid | type | type_desc | data_space_id | name | physical_name |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | NULL | 0 | ROWS | 1 | master | C:¥Program Files¥Microsoft SQL Server¥MSSQL |
| 2 | 1 | 2 | NULL | 1 | LOG | 0 | mastlog | C:¥Program Files¥Microsoft SQL Server¥MSSQL |
| 3 | 2 | 1 | NULL | 0 | ROWS | 1 | tempdev | C:¥Program Files¥Microsoft SQL Server¥MSSQL |
| 4 | 2 | 2 | NULL | 1 | LOG | 0 | templog | C:¥Program Files¥Microsoft SQL Server¥MSSQL |
| 5 | 3 | 1 | NULL | 0 | ROWS | 1 | modeldev | C:¥Program Files¥Microsoft SQL Server¥MSSQL |
| 6 | 3 | 2 | NULL | 1 | LOG | 0 | modellog | C:¥Program Files¥Microsoft SQL Server¥MSSQL |
| 7 | 4 | 1 | 687FF05... | 0 | ROWS | 1 | MSDBData | C:¥Program Files¥Microsoft SQL Server¥MSSQL |
| 8 | 4 | 2 | 1FDF176... | 1 | LOG | 0 | MSDBLog | C:¥Program Files¥Microsoft SQL Server¥MSSQL |

### SQL Server 2016 のインストール時に tempdb の設定を変更

SQL Server 2016 では、SQL Server のインストール時に tempdb の設定を変更できるようになりました。これは [**データベース エンジンの構成**] ページの [**TempDB**] ページから行えます。




**.mdf**（データ ファイル）に関しては、ファイル数や初期サイズ、作成場所、**.ldf**（ログ ファイル）に関しては、初期サイズ、作成場所などを設定できるので、これを移行元の SQL Server と同じように設定します。なお、.mdf のファイル数は、既定では搭載されている CPU のスレッド数または 8 スレッド以上の場合は 8 に設定されます（画面は 4 スレッドの場合の例）。

## tempdb の設定を変更する場合

SQL Server のインストール後に、tempdb データベースの場所や、サイズの変更を行いたい場合には、次のように実行します。

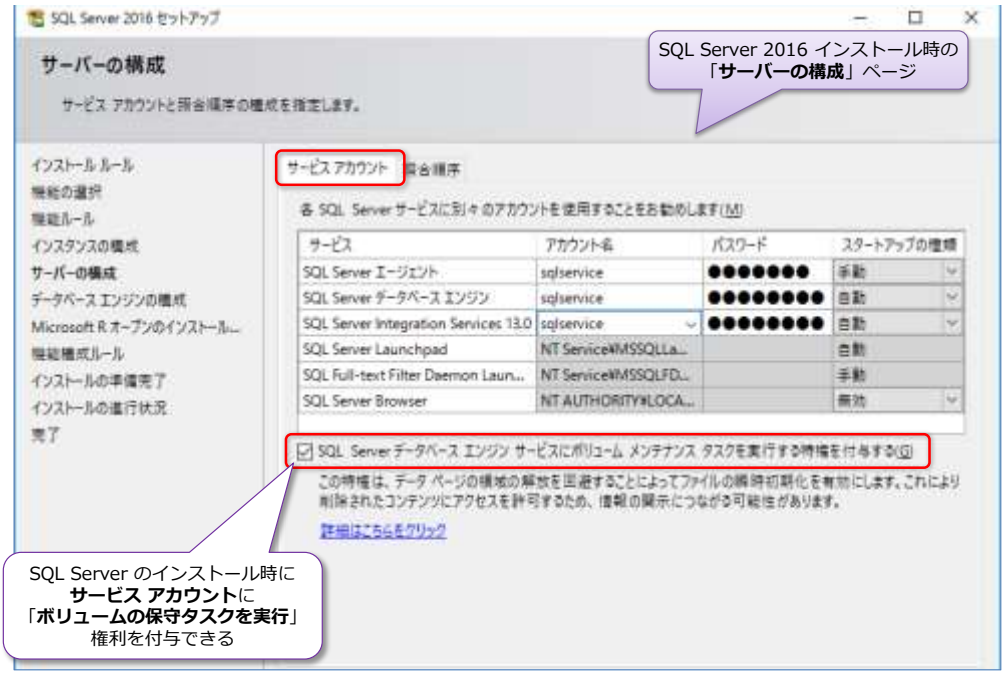
```
-- 場所を変更する場合（E ドライブの直下へ移動する場合の例）
ALTER DATABASE tempdb
 MODIFY FILE ( NAME = 'tempdev', FILENAME = 'E:¥tempdb.mdf' )
ALTER DATABASE tempdb
 MODIFY FILE ( NAME = 'templog', FILENAME = 'E:¥templog.ldf' )
go

-- 場所の変更を完了するには SQL Server の再起動が必要

-- サイズを変更する場合（mdf は 20GB、ldf は 10GB へ変更する場合の例）
ALTER DATABASE tempdb
 MODIFY FILE ( NAME = 'tempdev', SIZE = 20480MB )
ALTER DATABASE tempdb
 MODIFY FILE ( NAME = 'templog', SIZE = 10240MB )
go
```

なお、サイズの変更を行う場合は、**瞬時初期化**機能が有効になっていない場合は、実行に時間がかかることに注意してください。瞬時初期化機能が有効になっている場合は、.mdf のサイズ変更は、瞬時に行うことができます（これに対して .ldf に関しては、ストレージの書き込み性能によって、変更時間が大きく左右します）。

**瞬時初期化**機能を有効化するには、「**ボリュームの保守タスクを実行**」ユーザーの権利をサービスアカウントへ付与しますが、これについても、SQL Server 2016 からは SQL Server のインストール時に設定できるようになりました。

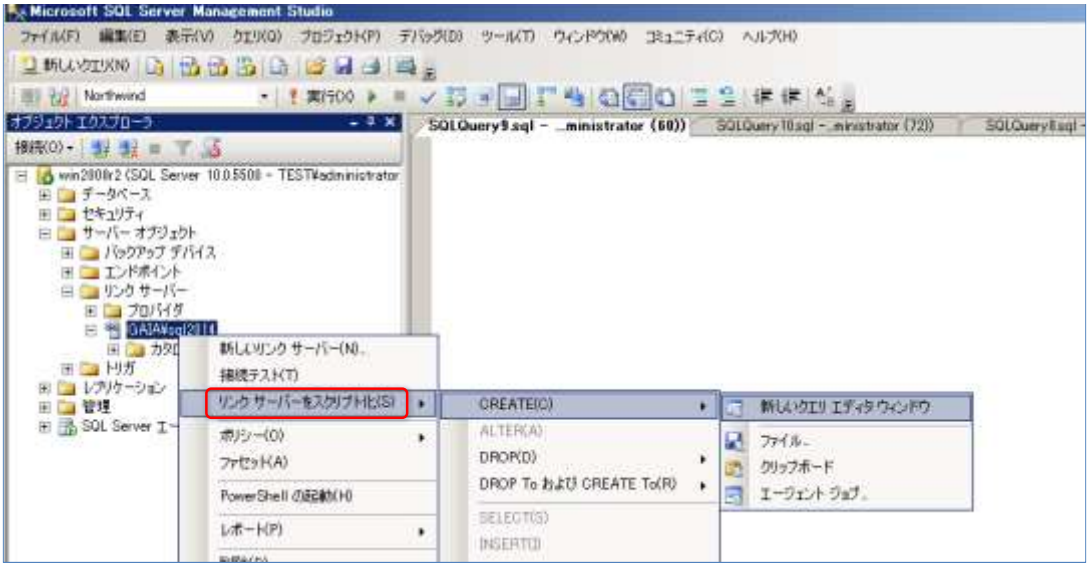


## 5.16 リンク サーバー設定の移行

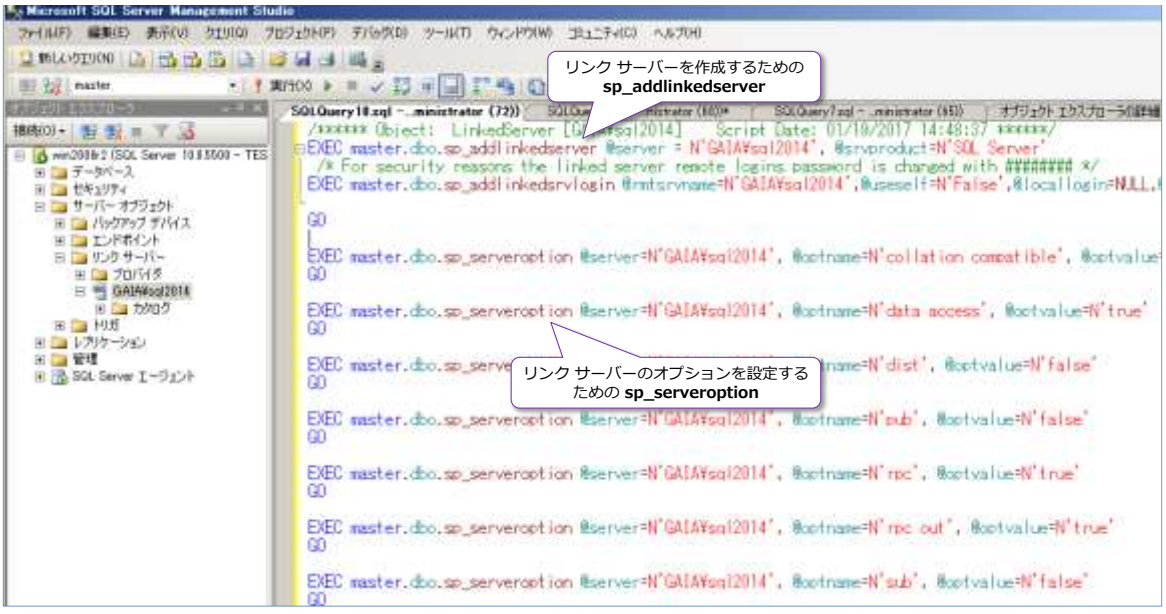
リンク サーバーの設定も **master** データベース（システム データベース）内に格納されているので、別途移行を行う必要があります。

### リンク サーバー設定のスクリプト生成

リンク サーバーの設定をスクリプト化するには、次のように［**サーバー オブジェクト**］の［**リンクサーバー**］を展開して、該当リンク サーバーを右クリックして、［**名前を付けてリンク サーバーをスクリプト化**］をクリックします（画面は SQL Server 2008 の場合ですが、他のバージョンでも同じように操作できます）。

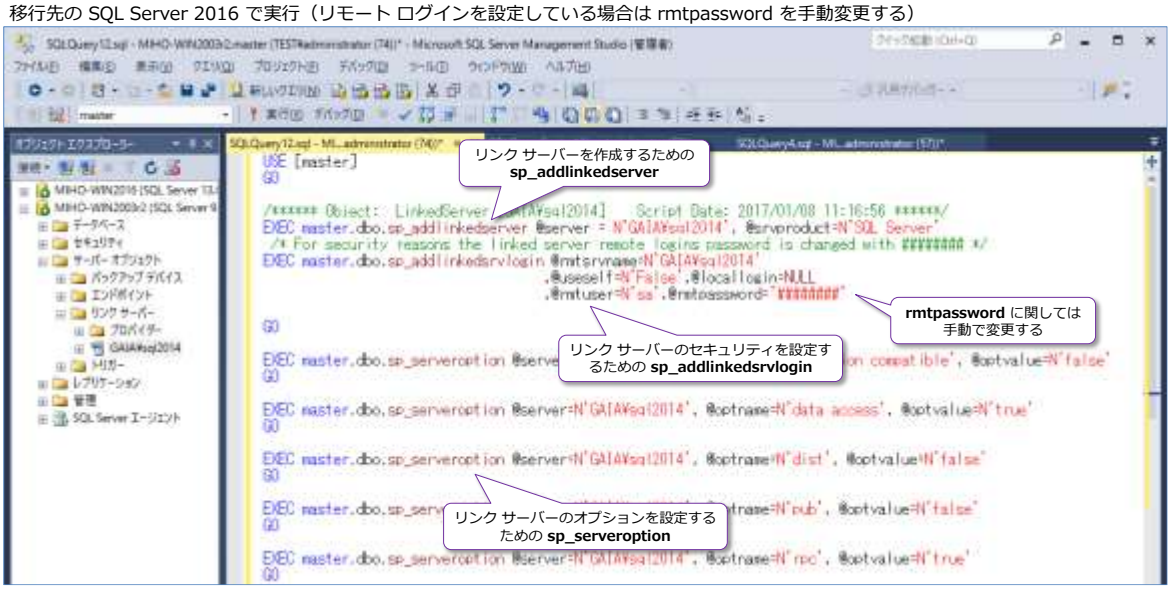



これで、リンク サーバーを作成するための **sp_addlinkedserver** とリンク サーバーのオプションを設定するための **sp_serveroption** が生成されます。



ただし、リモート ログインのパスワード（**rmtpassword**）に関しては、セキュリティ上の理由から ******* として生成されるので、この部分だけは実際の接続のためのパスワードに変更してから、移行先で実行する必要があります。

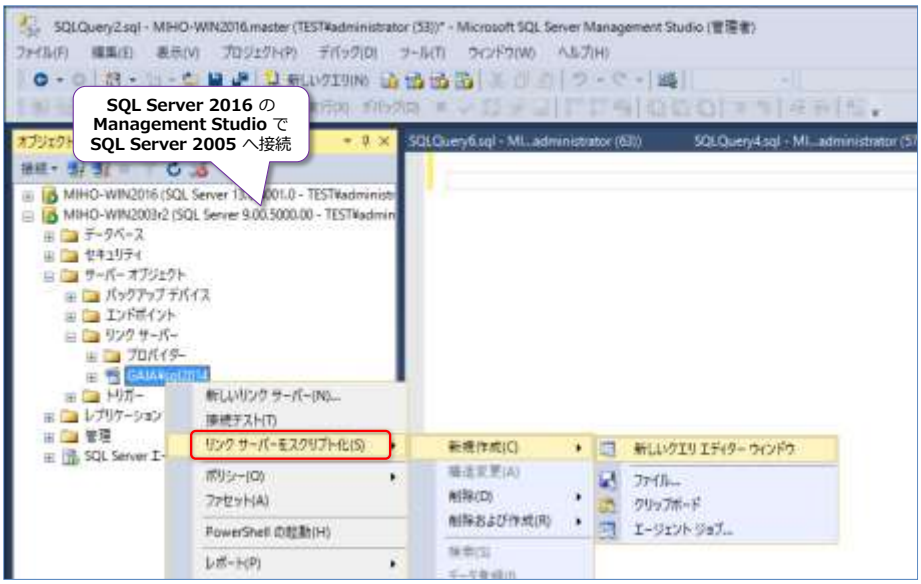
移行先の SQL Server 2016 で実行（リモート ログインを設定している場合は rmtpassword を手動変更する）




**Note： SQL Server 2005 の SP1 以前を利用している場合**

SQL Server 2005 の SP1 以前を利用している場合は、上の手順でスクリプト生成したものに、**sp_addlinkedsrvlogin**（セキュリティ ページで設定した接続情報）が含まれません。SP2 以降の Management Studio を利用している場合には、**sp_addlinkedsrvlogin** を含めてくれるのですが、SP1 以前の場合は含めてくれません。

これを回避するには、セキュリティ ページで［**スクリプト**］ボタンをクリックして、セキュリティ情報をスクリプト化するか、SQL Server 2016 の Management Studio を利用して、SQL Server 2005 にリモート接続してスクリプト生成するようにします。

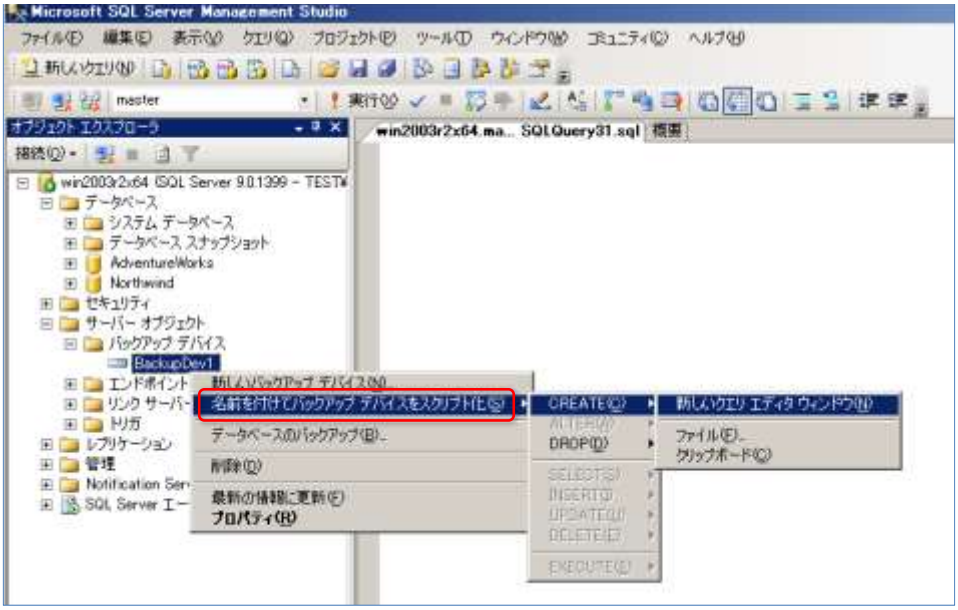


## 5.17 バックアップ デバイスの移行

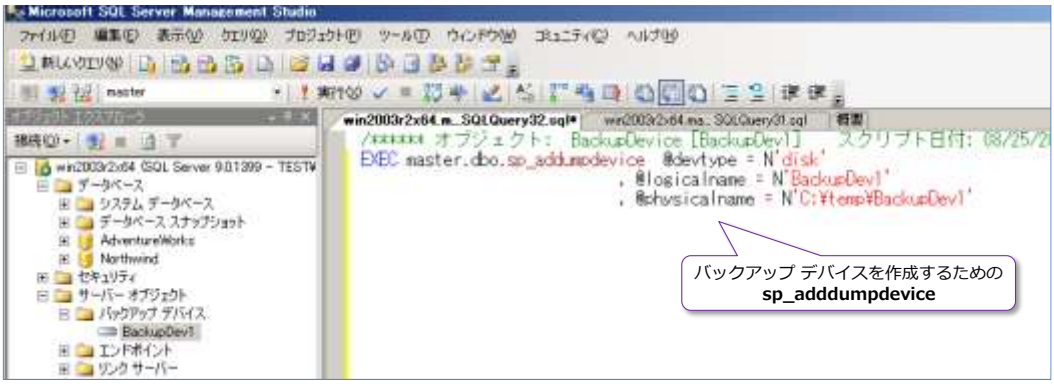
バックアップ デバイスは、利用されている方は非常に少ないかもしれません（SQL Server 6.5 の頃から SQL Server を利用しているユーザーの方にいらっしゃるかもしれません）。バックアップ デバイスの設定も **master** データベース（システム データベース）内に格納されているので、別途移行を行う必要があります。

### バックアップ デバイス設定のスクリプト生成

バックアップ デバイスの設定をスクリプト化するには、移行元の SQL Server で、次のように［**サーバー オブジェクト**］の［**バックアップ デバイス**］を展開して、該当バックアップ デバイスを右クリックして、［**名前を付けてバックアップ デバイスをスクリプト化**］をクリックします（以下の画面は SQL Server 2005 の場合ですが、他のバージョンでも同じように操作できます）。




これで、バックアップ デバイスを作成するための **sp_addumpdevice** が生成されます。




あとは、生成されたスクリプトを移行先（SQL Server 2016 上）で実行すれば、バックアップ デバイスの設定を移行することができます。

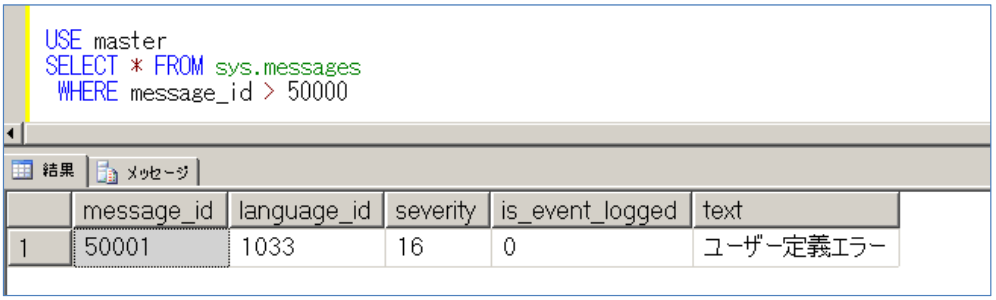
## 5.18 ユーザー定義エラーの移行

ユーザー定義エラーも利用されている方は少ないかもしれません。これも **master** データベース（システム データベース）内に格納されているので、利用している場合は別途移行を行う必要があります。

### ユーザー定義エラーのスクリプト化

ユーザー定義エラーに関しては、Management Studio からは（GUI 操作では）スクリプトを生成することができないので、ユーザー定義エラーの一覧を取得することができる **sys.messages** システム テーブルを利用します。

```
USE master
SELECT * FROM sys.messages
 WHERE message_id > 50000
```



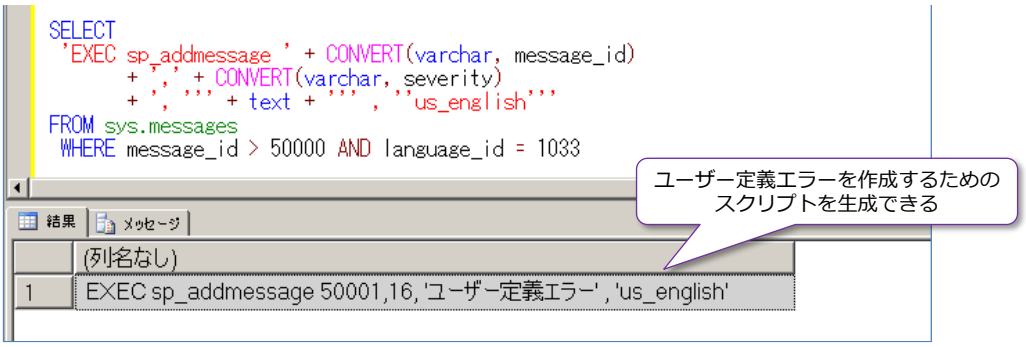

ユーザー定義エラーは、エラー番号が **50001** 以上になるので、「**WHERE message_id > 50000**」の条件を加えることでユーザー定義エラーのみを取得することができます。

ユーザー定義エラーは、**sp_addmessage** システム ストアド プロシージャを利用して、次のように作成します。

```
USE master
EXEC sp_addmessage 50001, 10, 'ユーザー定義エラー', 'us_english'
```

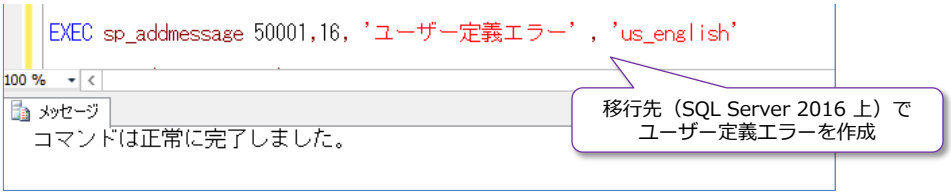
したがって、次のようにクエリを実行すれば、ユーザー定義エラーを作成するためのスクリプトを作成することができます（言語が **us_english** の場合）。

```
SELECT
 'EXEC sp_addmessage ' + CONVERT(varchar, message_id)
       + ',' + CONVERT(varchar, severity)
       + ', ''' + text + ''' , ''us_english'''
FROM sys.messages
 WHERE message_id > 50000 AND language_id = 1033
```



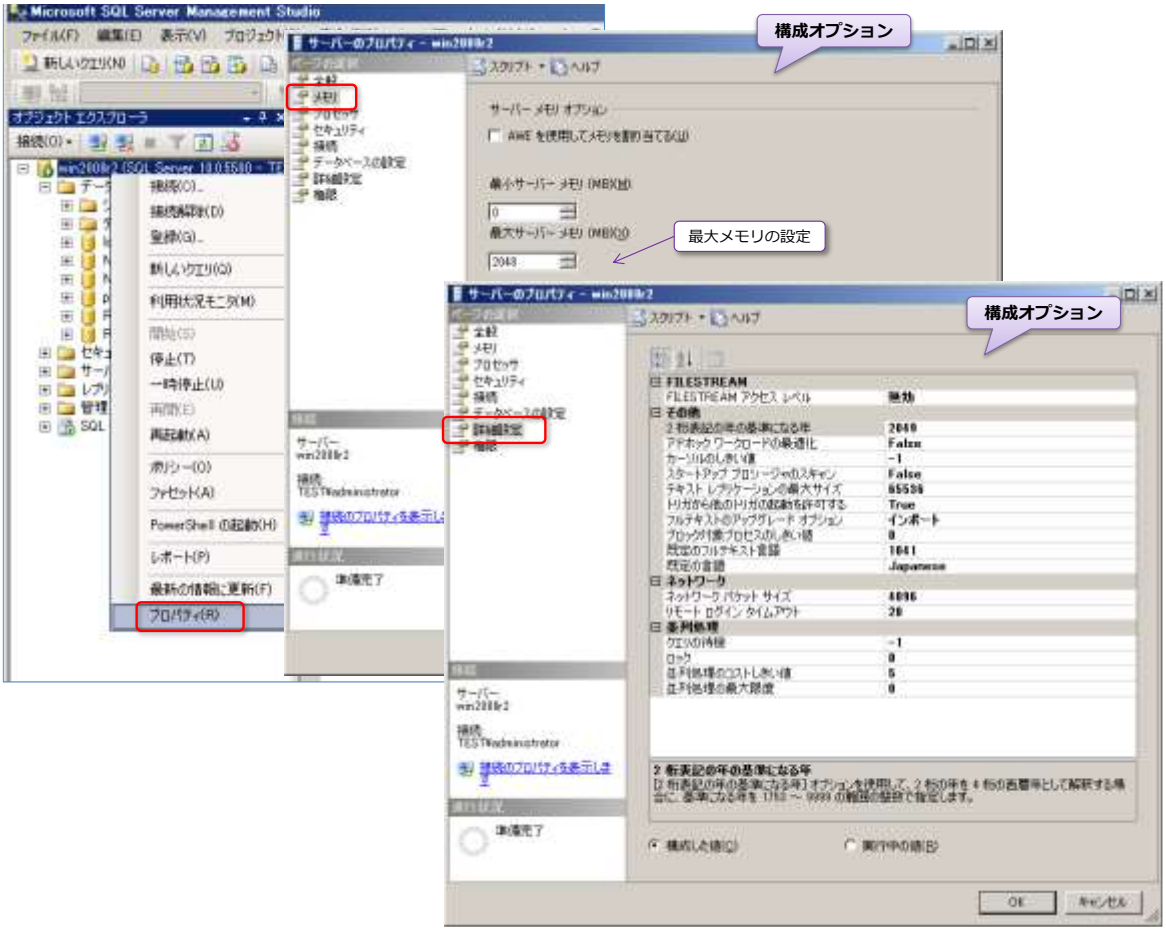

これは、「**language_id = 1033**」で **us_english**（英語版）のメッセージに絞り込んでいますが、クエリの文字列連結部分を変更すれば、ほかの言語のメッセージにも対応させることができます。

あとは、生成されたスクリプトを移行先（SQL Server 2016 上）で実行すれば、ユーザー定義エラーを移行することができます。



## 5.19 構成オプション（sp_configure）の移行

構成オプションは、次のように**サーバーのプロパティ**や **sp_configure** ストアド プロシージャで設定したサーバーのオプション（最大メモリや接続オプションの設定など）です。




現在の構成オプションは、次のように **sp_configure** システム ストアド プロシージャでまとめて確認することができます。

```
EXEC sp_configure 'show advanced options', 1
RECONFIGURE
EXEC sp_configure
```

```
EXEC sp_configure 'show advanced options', 1
RECONFIGURE
EXEC sp_configure
```

結果 | メッセージ

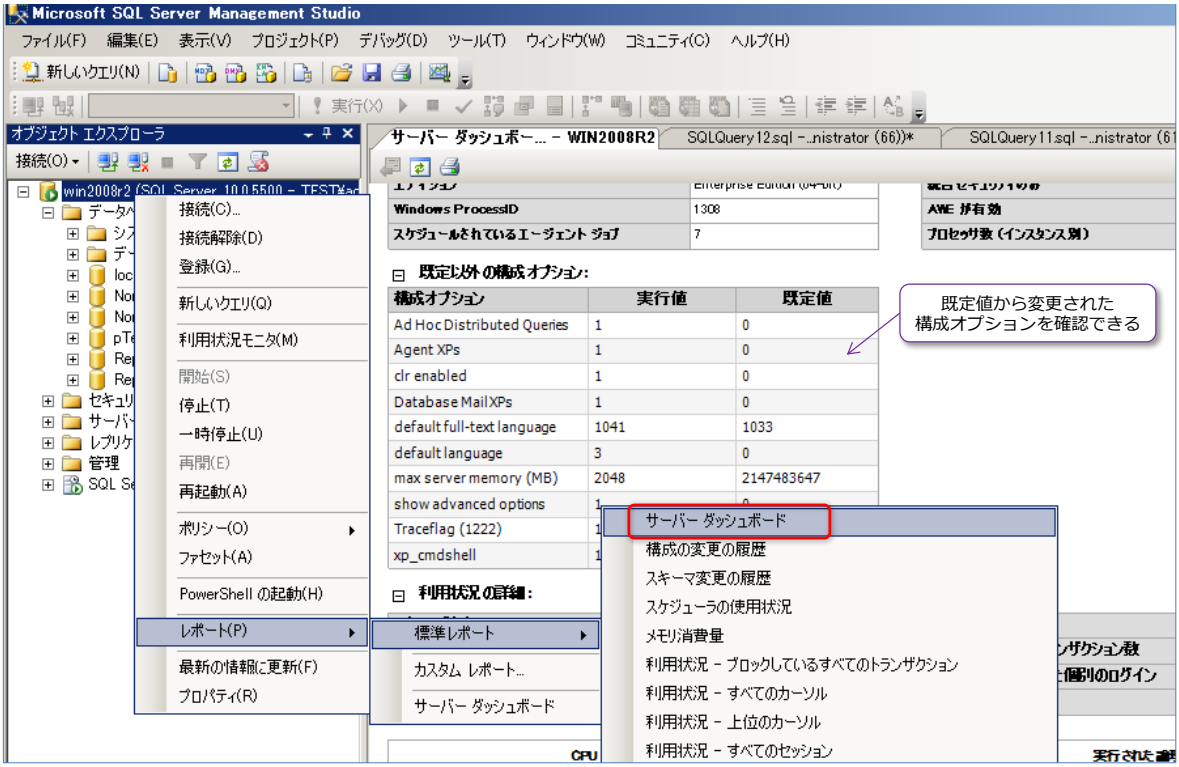
| | name | minimum | maximum | config_value | run_value |
|---|---|---|---|---|---|
| 1 | access check cache bucket count | 0 | 65536 | 0 | 0 |
| 2 | access check cache quota | 0 | 2147483647 | 0 | 0 |
| 3 | Ad Hoc Distributed Queries | 0 | 1 | 1 | 1 |
| 4 | affinity I/O mask | -2147483648 | 2147483647 | 0 | 0 |
| 5 | affinity mask | -2147483648 | 2147483647 | 0 | 0 |
| 6 | affinity64 I/O mask | -2147483648 | 2147483647 | 0 | 0 |
| 7 | affinity64 mask | -2147483648 | 2147483647 | 0 | 0 |

**show advanced options** を **1** へ設定してすべてのオプションを表示するようにして、その後「**sp_configure**」を実行することで、現在の値（**run_value**）を確認することができます。

これらの構成オプションも **master** データベース（システム データベース）内に格納されているので、設定を変更している場合には別途移行を行う必要があります。

## 既定値から変更のあった構成オプションのみを表示

構成オプションは、既定値（インストール直後の値）から変更のあったもののみを簡単に確認することができます。これは、次のように Management Studio でサーバー名を右クリックして、[**レポート**] の [**標準レポート**] から「**サーバー ダッシュボード**」をクリックします。




このように、サーバー ダッシュボードを利用すれば、どのオプションが変更されたのかが一目瞭然になるので、大変便利です。

なお、このレポートは、SQL Server 2005 の SP2 以降で利用できる機能なので、SQL Server 2005 の SP1 以前を利用している場合は、SQL Server 2016 の Management Studio を利用して、SQL Server 2005 にリモート接続してレポートを開いてみてください。

## 構成オプションのスクリプト化

構成オプションに関しては、Management Studio の GUI 操作ではスクリプト生成をすることができないので、次のように「**sp_configure**」の結果を一時テーブルに保存して、これをカーソルでループ処理することで、スクリプト生成をすることもできます。

```
-- show advanced options を1 に設定
EXEC sp_configure 'show advanced options', 1
RECONFIGURE

-- sp_configure の結果を一時テーブルへ保存
CREATE TABLE #temp1
( name sysname
 ,minimum int
 ,maximum int
 ,config_value int
 ,run_value int )
INSERT INTO #temp1 EXEC sp_configure

-- 一時テーブル内の run_value をカーソルでループ
DECLARE @name sysname, @run_value int
DECLARE cur1 CURSOR FOR
               SELECT name, run_value FROM #temp1
OPEN cur1
 FETCH NEXT FROM cur1 INTO @name, @run_value
WHILE (@@FETCH_STATUS <> -1)
BEGIN
  -- 現在の構成値 (run_value) をもとにスクリプト化
  PRINT 'EXEC sp_configure ''' + @name + ''', ' + CONVERT(varchar, @run_value) + ''
  FETCH NEXT FROM cur1 INTO @name, @run_value
END
CLOSE cur1
DEALLOCATE cur1
DROP TABLE #temp1
```


このように、ストアド プロシージャの結果と同じ列数のテーブルを作成して、**INSERT .. EXEC**

を実行することで、**EXEC** の結果をテーブルに保存することができます。このテーブルを利用して、カーソル ループを行えば、現在の設定値（**run_value**）をもとに **sp_configure** をスクリプト化することができます。

あとは、生成されたスクリプトを移行先（SQL Server 2016 上）で実行すれば、構成オプションを移行することができますが、前掲のダッシュボード レポートを利用して、変更があったオプションのみを、移行先で実行するようにします。

移行先で実行後は、次のように **RECONFIGURE WITH OVERRIDE** を実行して、構成を反映させるようにします。

```
-- 移行先で実行
EXEC sp_configure 'show advanced options', 1
RECONFIGURE

-- 変更があった構成オプションを実行
EXEC sp_configure 'Ad Hoc Distributed Queries', 1
EXEC sp_configure 'affinity I/O mask', 0
          : （中略）
EXEC sp_configure 'xp_cmdshell', 1

-- 最後に RECONFIGURE を実行する
RECONFIGURE WITH OVERRIDE
```

構成オプションによっては、**SQL Server サービスの再起動**が必要なものがありますが、再起動が必要かどうかは、オンライン ブックの以下のトピックに記載されています。

サーバー構成オプション
http://msdn.microsoft.com/ja-jp/library/ms189631.aspx


構成オプションの表

次の表は、使用可能なすべての構成オプション、設定可能範囲、および既定値を示しています。構成オプションには文字コードを付けています

- A = 詳細設定オプション。熟練したデータベース管理者または認定された SQL Server 技術者だけがこのオプションを変更するように advanced options を 1 に設定する必要があります。
- RR = データベース エンジンの再起動が必要なオプション。
- RP = PolyBase エンジンの再起動が必要なオプション。
- SC = 自己構成オプション。

| 構成オプション | 最小値 | 最大値 | 既定値 |
|---|---|---|---|
| access check cache bucket count (A) | 0 | 16384 | 0 |
| access check cache quota (A) | 0 | 2147483647 | 0 |
| ad hoc distributed queries (A) | 0 | 1 | 0 |
| affinity I/O mask (A, RR) | -2147483648 | 2147483647 | 0 |
| affinity64 I/O mask (A, 64 ビット版の SQL Server でのみ使用可能) | -2147483648 | 2147483647 | 0 |
| affinity mask (A) | -2147483648 | 2147483647 | 0 |
| affinity64 mask (A, RR, 64 ビット版のでのみ使用可能 SQL Server | -2147483648 | 2147483647 | 0 |

**RR** と表記されるものが再起動が必要なオプションで、これらをまとめると、次のようになります。

affinity64 mask、c2 audit mode、common criteria compliance enabled、
fill factor、lightweight pooling、locks、media retention、open objects、
priority boost、remote access、set working set size、user connections

これらの構成オプションを変更している場合には、スクリプト実行後に、SQL Server サービスを再起動する必要があります。

## SQL Server 2016 でサポートされなくなった構成オプション

以前の SQL Server と比較して、SQL Server 2016 でサポートされなくなった構成オプションには、次のものがあります。

- **AWE Enabled** オプション（SQL Server 2012 以降で未サポート）
- **SQL Mail XPs**（SQL Server 2012 以降で未サポート）
- **Web Assistant Procedures**（SQL Server 2008 以降で未サポート）

**AWE Enabled**（AWE の有効化）と **SQL Mail XPs**（SQL Mail 機能）は、SQL Server 2012 以降で未サポートになり、**Web Assistant** 機能は SQL Server 2008 以降で未サポートになっています。

**Note： SQL Server 2005 の構成オプションとの比較**

SQL Server 2005 の構成オプションの既定値は、次のようになっています。

| 構成オプション | 既定値 |
|---|---|
| Ad Hoc Distributed Queries | 0 |
| affinity I/O mask | 0 |
| affinity mask | 0 |
| affinity64 I/O mask | 0 |
| affinity64 mask | 0 |
| Agent XPs | 0 |
| allow updates | 0 |
| awe enabled | 0 |
| blocked process threshold | 0 |
| c2 audit mode | 0 |
| clr enabled | 0 |
| common criteria compliance enabled | 0 |
| cost threshold for parallelism | 5 |
| cross db ownership chaining | 0 |
| cursor threshold | -1 |
| Database Mail XPs | 0 |
| default full-text language | 1,041 |
| default language | 3 |
| default trace enabled | 1 |
| disallow results from triggers | 0 |
| fill factor (%) | 0 |
| ft crawl bandwidth (max) | 100 |
| ft crawl bandwidth (min) | 0 |
| ft notify bandwidth (max) | 100 |
| ft notify bandwidth (min) | 0 |
| index create memory (KB) | 0 |
| in-doubt xact resolution | 0 |
| lightweight pooling | 0 |
| locks | 0 |
| max degree of parallelism | 0 |
| max full-text crawl range | 4 |
| max server memory (MB) | 2,147,483,647 |
| max text repl size (B) | 65,536 |
| max worker threads | 0 |
| media retention | 0 |
| min memory per query (KB) | 1,024 |
| min server memory (MB) | 0 |
| nested triggers | 1 |
| network packet size (B) | 4,096 |
| Ole Automation Procedures | 0 |
| open objects | 0 |
| PH timeout (s) | 60 |
| precompute rank | 0 |
| priority boost | 0 |
| query governor cost limit | 0 |
| query wait (s) | -1 |
| recovery interval (min) | 0 |
| remote access | 1 |
| remote admin connections | 0 |
| remote login timeout (s) | 20 |
| remote proc trans | 0 |
| remote query timeout (s) | 600 |
| Replication XPs | 0 |
| scan for startup procs | 0 |
| server trigger recursion | 1 |
| set working set size | 0 |
| show advanced options | 0 |
| SMO and DMO XPs | 1 |
| SQL Mail XPs | 0 |
| transform noise words | 0 |
| two digit year cutoff | 2,049 |
| user connections | 0 |
| user options | 0 |
| Web Assistant Procedures | 0 |
| xp_cmdshell | 1 |

* common criteria compliance enabled は SP2 以降

多くのオプションが「**0**」に設定されていて、**0** は、動作に関するオプション（Affinity Mask や Max Degree of Parallelism、Locks、Max Worker Threads など）であれば "**自動調整**"、セキュリティ関連のオプション（Ad Hoc Distributed Queries や clr enabled、Database Mail XPs、xp_cmdshell など）であれば "**無効**" という意味になります。なお、Agent XPs は、SQL Server Agent サービスを自動起動に設定している場合は 1 に設定されます。

### ■ SQL Server 2016 との比較

SQL Server 2005 と SQL Server 2016 では、ほとんどの構成オプションの既定値が同じです。違いは、次の 2 つのみです。

- **min server memory (MB)** が **16** に変更されている（SQL Server 2005 では **0**）
- **remote login timeout (s)** が **10** に変更されている（SQL Server 2005 では **20**）

構成オプションの詳細については、オンライン ブックの以下のトピックが参考になると思います。

サーバー構成オプション
http://msdn.microsoft.com/ja-jp/library/ms189631.aspx

## 5.20 TDE（透過的なデータ暗号化）の移行

SQL Server 2008 からの新機能である **TDE（透過的なデータ暗号化）**を設定したデータベースは、master データベース内にある**サーバー証明書**を利用した機能なので、移行を行うには、**サーバー証明書の移行**（バックアップと復元）も必要になります。

TDE を設定したデータベースのバックアップを、別のマシン（移行先の SQL Server 2016）でリストアしようとすると、次のようにエラーが発生します。

```
-- 別マシンでのリストア
RESTORE DATABASE tdeTestDB
 FROM DISK='C:¥temp¥tdeTestDB.bak'
```

```
100 %
メッセージ
メッセージ 33111、レベル 16、状態 3、行 132
拇印 '0x37A5D9B94060FAAD5ED39FBE125ADC12A372CBA6' でサーバー 証明書 が見つかりません。
メッセージ 3013、レベル 16、状態 1、行 132
RESTORE DATABASE が異常終了しています。
```

これは、サーバー証明書が存在しないので、リストアができないというエラーになっています。TDE では、バックアップ ファイルが持ち出されたとしても、別のマシンでは簡単にはリストアできないようにするために、サーバー証明書が利用されています。

### サーバー証明書の移行（バックアップとリストア）

サーバー証明書を移行するには、移行元の SQL Server で、次のように **BACKUP CERTIFICATE** ステートメントを利用して、サーバー証明書（と秘密キー）のバックアップを実行しておきます。

```
BACKUP CERTIFICATE MyServerCert
 TO FILE = 'C:¥temp¥MyServerCert'
 WITH PRIVATE KEY
         ( FILE = 'C:¥temp¥MyPrivKey'
         , ENCRYPTION BY PASSWORD = 'P@ssword' )
```

```
-- サーバー証明書のバックアップ
USE master
BACKUP CERTIFICATE MyServerCert
 TO FILE = 'C:¥temp¥MyServerCert'
 WITH PRIVATE KEY
     ( FILE = 'C:¥temp¥MyPrivKey'
     , ENCRYPTION BY PASSWORD = 'P@ssword' )
```

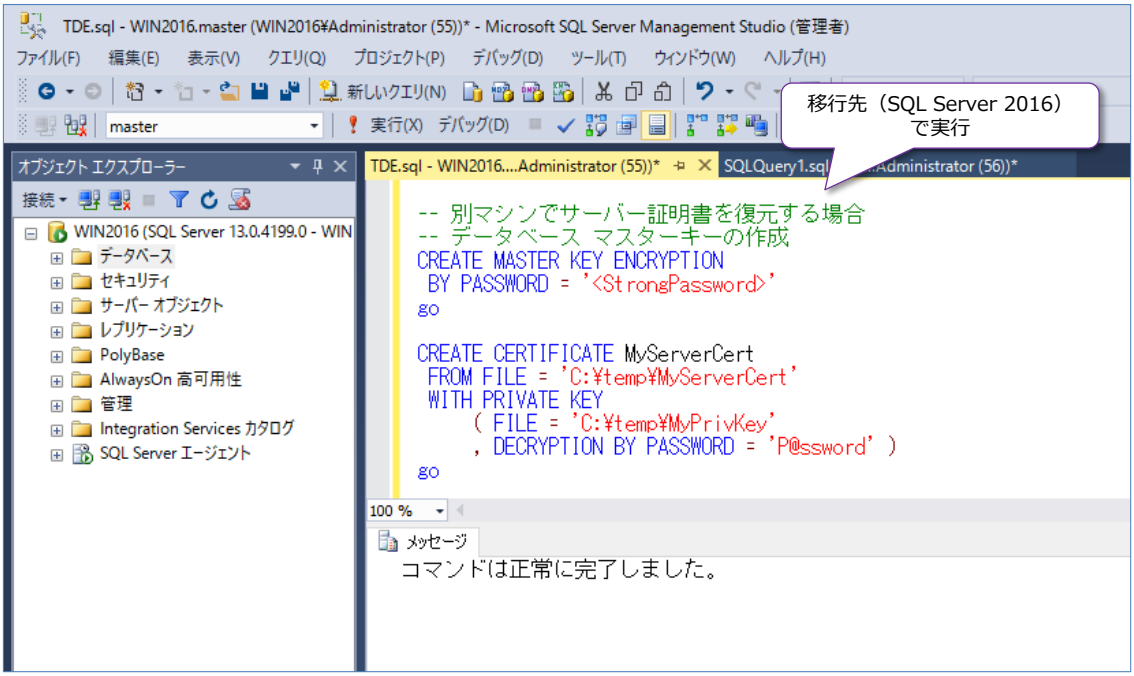
```
100 %
メッセージ
コマンドは正常に完了しました。
```

このステートメントでは、サーバー証明書をファイルへバックアップすることができ、**WITH PRIVATE KEY** オプションを付けることで、秘密キーもバックアップできます。このときに **ENCRYPTION BY PASSWORD** で指定するパスワードには、推測されにくい強固なパスワード

を設定して、そのパスワードが簡単に漏れないように注意しておく必要があります。

バックアップしたファイル（サーバー証明書と秘密キー）は、移行先となる SQL Server 2016 に物理的にコピーして、次のように **CREATE CERTIFICATE** ステートメントを利用して、リストア（インポート）することができます。

```
-- データベース マスター キーの作成
USE master
CREATE MASTER KEY
 ENCRYPTION BY PASSWORD = '<StrongPassword>'
go
-- サーバー証明書のリストア（インポート）
CREATE CERTIFICATE MyServerCert
 FROM FILE = 'C:¥temp¥MyServerCert'
 WITH PRIVATE KEY
         ( FILE = 'C:¥temp¥MyPrivKey'
         , DECRYPTION BY PASSWORD = 'P@ssword' )
```




サーバー証明書のリストア（ファイルからの作成）には、**データベース マスター キー**を作成しておく必要があるので、まずデータベース マスター キーを作成しています。

サーバー証明書のリストア時（**CREATE CERTIFICATE** ステートメントの実行時）には、**DECRYPTION BY PASSWORD** に、複合化のためのパスワードを、バックアップ時に指定したのと同じものを指定します。これで、サーバー証明書のリストアが完了です。

このように同じサーバー証明書がリストアされている環境であれば、透過的なデータ暗号化（TDE）を設定したデータベースをリストアして、データを参照することができるようになります。

```
-- サーバー証明書がリストアされていれば、DB のリストアも可能
RESTORE DATABASE enc
 FROM DISK='C:¥temp¥enc.bak'
```

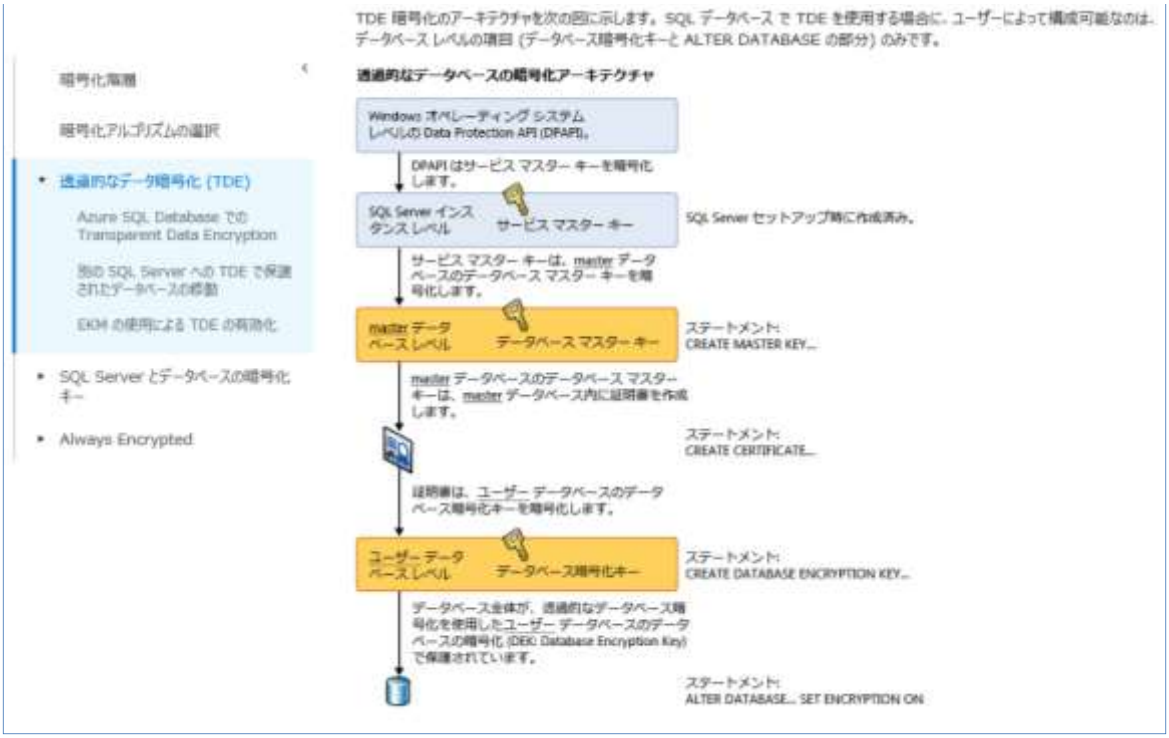
100 %

メッセージ

```
データベース 'enc' の 176 ページ、ファイル 1 のファイル 'enc' を処理しました。
データベース 'enc' の 4 ページ、ファイル 1 のファイル 'enc_log' を処理しました。
データベース 'enc' をバージョン 655 から現在のバージョン 852 に変換しています。
データベース 'enc' で、バージョン 655 からバージョン 668 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 668 からバージョン 669 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 669 からバージョン 670 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 670 からバージョン 671 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 671 からバージョン 672 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 672 からバージョン 673 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 673 からバージョン 674 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 674 からバージョン 675 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 675 からバージョン 676 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 676 からバージョン 677 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 677 からバージョン 679 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 679 からバージョン 680 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 680 からバージョン 681 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 681 からバージョン 682 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 682 からバージョン 683 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 683 からバージョン 684 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 684 からバージョン 685 へのアップグレード手順が実行されています。
データベース 'enc' で、バージョン 685 からバージョン 686 へのアップグレード手順が実行されています。
```

その他、TDE の詳細（アーキテクチャなど）については、オンライン ブックの以下のトピックがお勧めです。

透過的なデータ暗号化（TDE）
http://msdn.microsoft.com/ja-jp/library/bb934049.aspx



## 5.21 msdb データベースに含まれる情報の移行

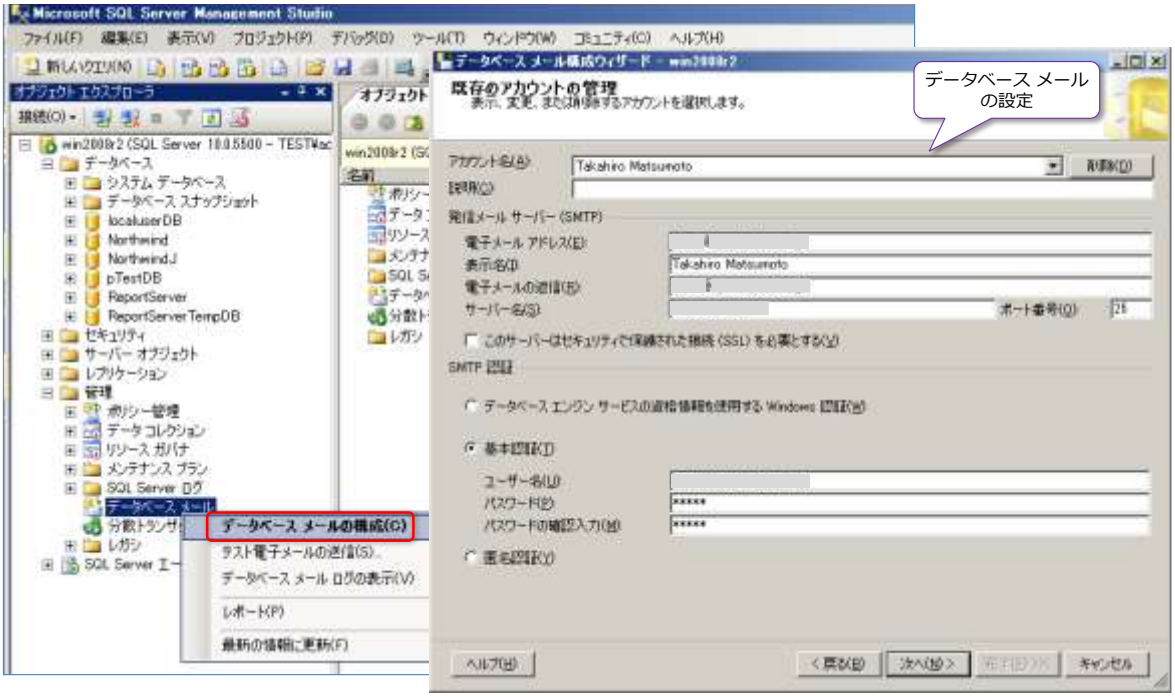
次に、システム データベースの 1 つである「**msdb**」データベース内に含まれる情報の移行について説明します。msdb データベース内には、主に次のものが含まれています。

- データベース メールの設定
- オペレーター
- ジョブ
- 警告
- メンテナンス プラン（保守計画）
- Integration Services パッケージ
- レプリケーションの設定
- ログ配布の設定
- データベース ミラーリングの設定（一部）

以降では、これらの移行方法について説明します。

## 5.22 データベース メール設定の移行

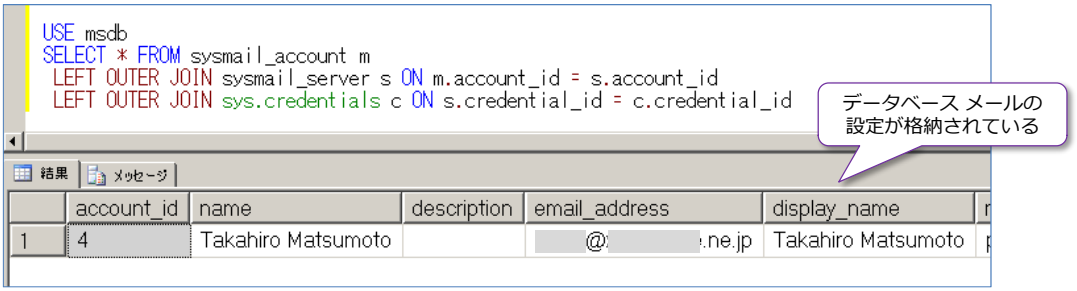
データベース メールの設定は、**msdb** データベース（システム データベース）内に格納されているので、別途移行を行う必要があります（以下の画面は SQL Server 2008 の場合）。




### データベース メール設定のスクリプト化

データベース メールの設定（メール アカウント）は、Management Studio からは（GUI 操作では）スクリプトを生成することができないので、データベース メールの設定を取得することができる **sysmail_account**、**sysmail_server**、**sys.credentials** システム テーブルを利用します。

```
USE msdb
SELECT * FROM sysmail_account m
 LEFT OUTER JOIN sysmail_server s ON m.account_id = s.account_id
 LEFT OUTER JOIN sys.credentials c ON s.credential_id = c.credential_id
```




メール アカウントの設定は、**sysmail_add_account_sp** システム ストアド プロシージャを利用して、次のように実行できます。

```
USE msdb
EXECUTE msdb.dbo.sysmail_add_account_sp
    @account_name            = 'アカウント名',
    @description             = '説明',
    @email_address           = 'メール アドレス',
    @replyto_address         = '返信先となるメール アドレス',
    @display_name            = '表示名',
    @mailserver_name         = 'メール サーバーのアドレス',
    @mailserver_type         = 'SMTP',
    @port                    = 'ポート番号',
    @username                = 'メール サーバーに認証が必要な場合のユーザー名',
    @password                = '認証が必要な場合のユーザーのパスワード'
```

**@username** と **@password** は、認証が不要な場合には省略することができます。

したがって、前出のシステム テーブル(**sysmail_account**、**sysmail_server**、**sys.credentials**)を利用して、次のようメール アカウントの設定をスクリプト化することができます。

```
USE msdb
SELECT '
  EXECUTE msdb.dbo.sysmail_add_account_sp
    @account_name            = ''' + a.name + ''',
    @description             = ''' + a.description + ''',
    @email_address           = ''' + a.email_address + ''',
    @replyto_address         = ''' + a.email_address + ''',
    @display_name            = ''' + a.display_name + ''',
    @mailserver_name         = ''' + s.servername + ''',
    @mailserver_type         = ''' + s.servertype + ''',
    @port                    = ''' + CONVERT(varchar, s.port) + ''',
    @use_default_credentials = ''0'' '
  + CASE
     WHEN c.credential_identity IS NOT NULL THEN
    ',@username                = ''' + c.credential_identity + '''
     ,@password                = ''######''' ELSE '' END
FROM sysmail_account a
  LEFT OUTER JOIN sysmail_server s ON a.account_id = s.account_id
  LEFT OUTER JOIN sys.credentials c ON s.credential_id = c.credential_id
```

**@password** は、###### と決め打ちにしているので、認証が必要な場合は、そのユーザーのパスワードへ変更します。

あとは、移行先（SQL Server 2016 上）で、次のようにデータベース メール機能を有効化して、**メール プロファイルの追加**などを行えば、データベース メールの設定を移行することができます。

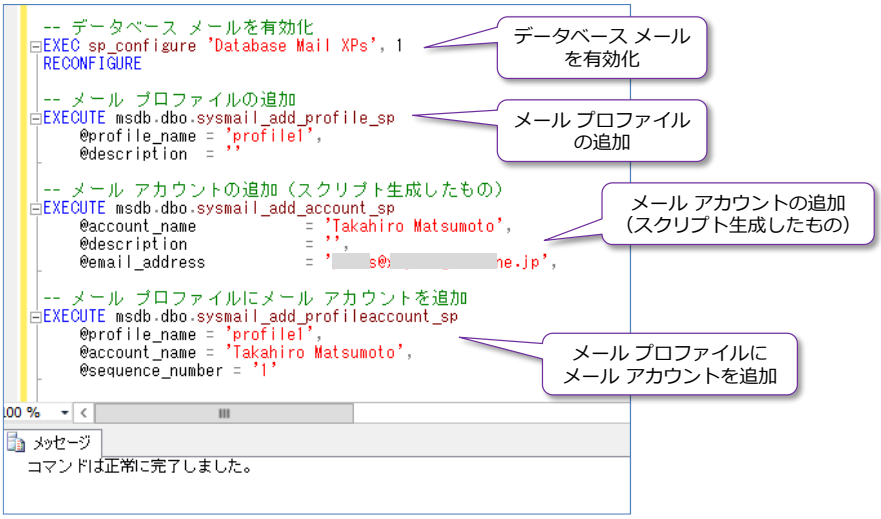
```
-- データベース メールを有効化
EXEC sp_configure 'Database Mail XPs', 1
RECONFIGURE

-- メール プロファイルの追加
EXECUTE msdb.dbo.sysmail_add_profile_sp
    @profile_name = 'プロファイル名',
```

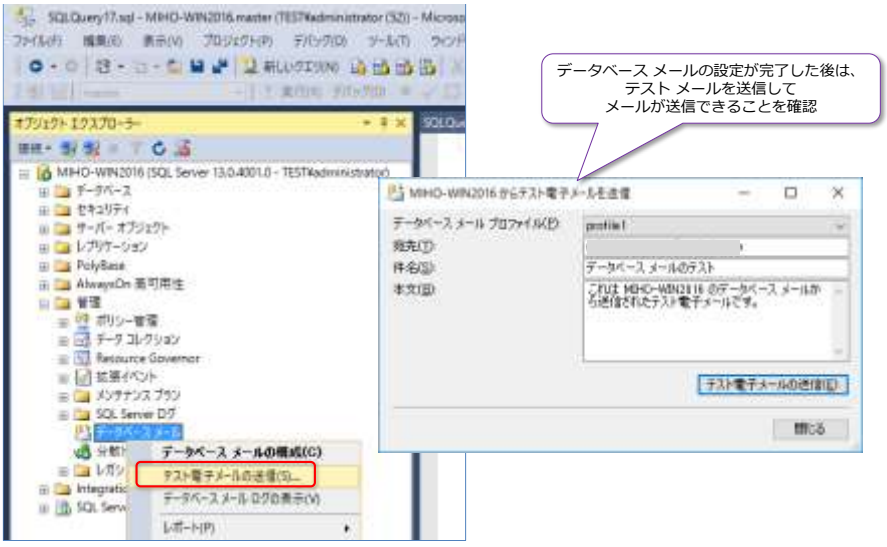
```
    @description  = ''

-- メール アカウントの追加（スクリプト生成したもの）
EXECUTE msdb.dbo.sysmail_add_account_sp
    @account_name             = 'アカウント名',
    @description              = '',
    @email_address            = '～@～～',
          : （中略）

-- メール プロファイルにメール アカウントを追加
EXECUTE msdb.dbo.sysmail_add_profileaccount_sp
    @profile_name = 'プロファイル名',
    @account_name = 'アカウント名',
    @sequence_number = '1'
```




**sysmail_add_profile_sp** でメール プロファイルを追加して（画面は **profile1** という名前で作成）、スクリプト生成したものでメール アカウントを作成、最後に **sysmail_add_profile account_sp** を利用して、作成したメール アカウントをメール プロファイルに追加すれば、データベース設定の移行が完了です。



## データベース メールの利用には .NET Framework 3.5 SP1 が必要

**SQL Server 2016** では、データベース メール機能を利用する場合のみ、**.NET Framework 3.5 SP1** が必要になります。これは、次のように**サーバー マネージャー**の［**役割と機能の追加**］ウィザードから簡単にインストールすることができます。

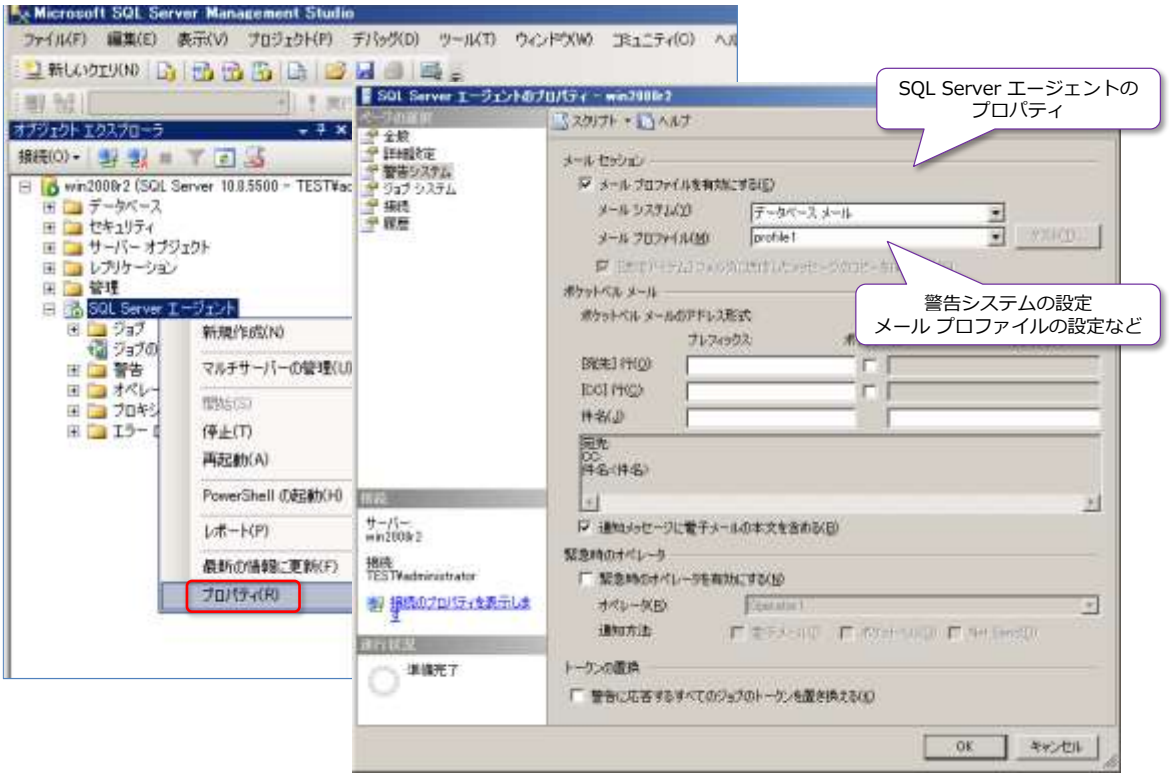
Windows Server 2012／2012 R2 での .NET Framework 3.5 SP1 のインストール

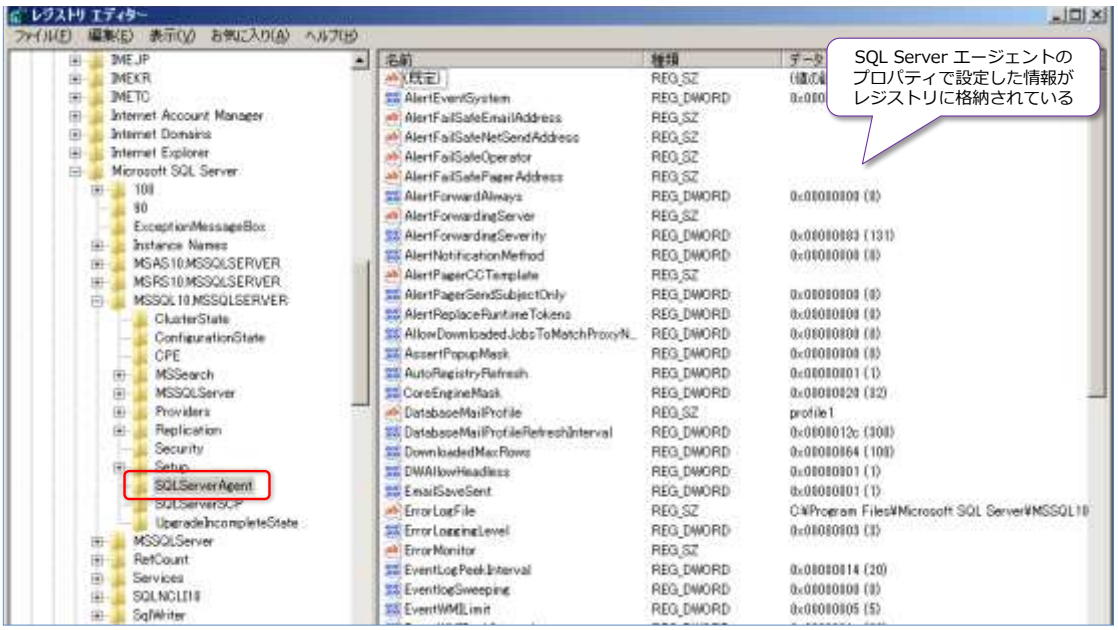

Windows Server 2016 での .NET Framework 3.5 SP1 のインストール


［**.NET Framework 3.5 Features**］の［**.NET Framework 3.5**］をチェックすることで、**.NET Framework 3.5 SP1** をインストールすることができます。これでデータベース メール機能を利用できるようになります。

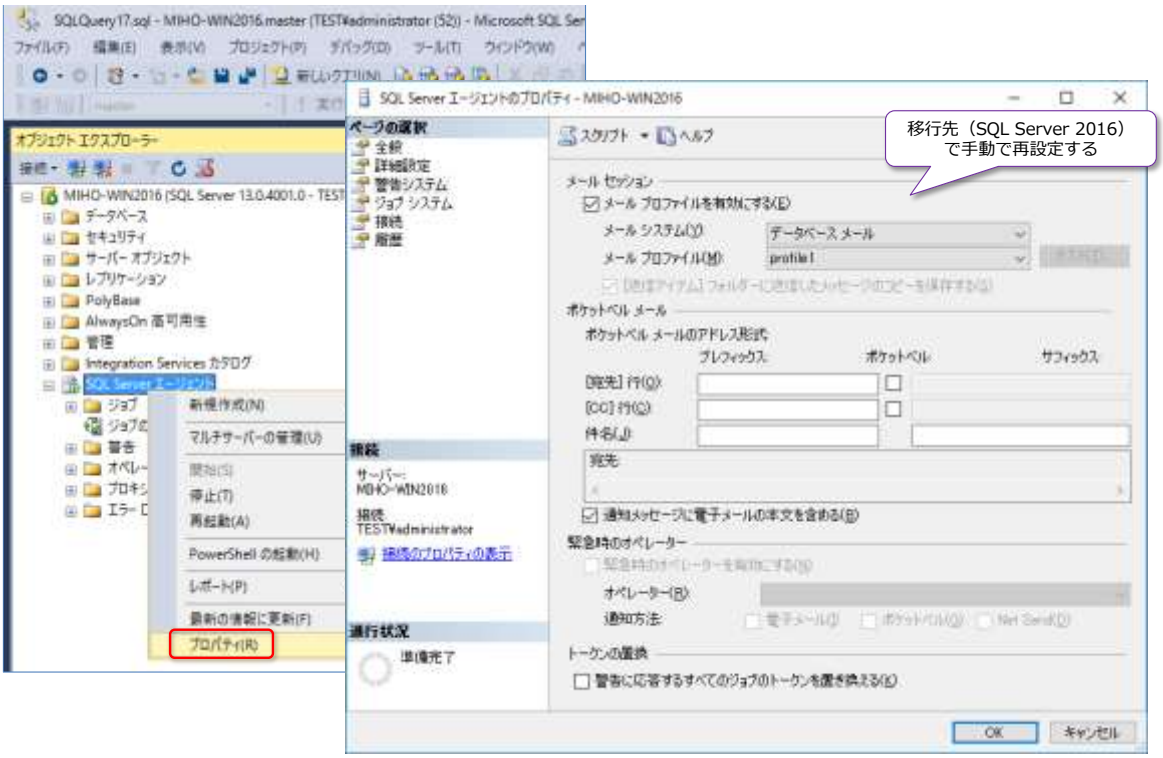
## 5.23 SQL Server エージェントのプロパティ設定の移行

SQL Server エージェントの［**プロパティ**］で設定した情報は、レジストリに格納されるので、手動で移行先（SQL Server 2016 上）で再設定しなければなりません（以下の画面は SQL Server 2008 の場合）。



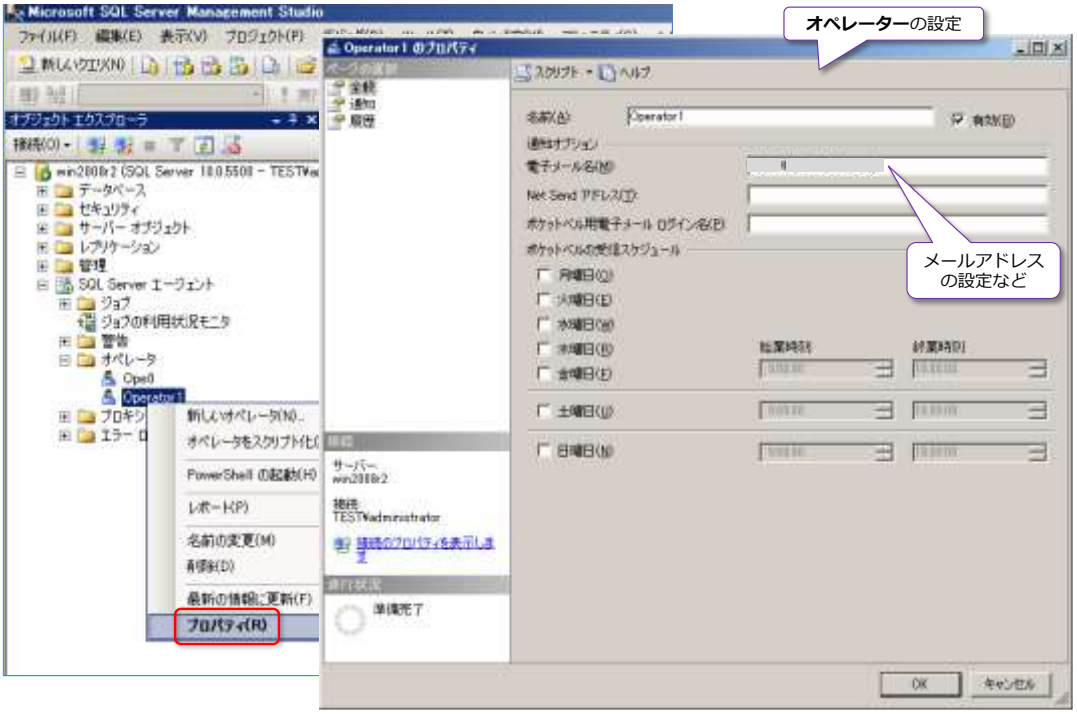

［**警告システム**］ページでの［**メール プロファイルを有効にする**］の設定は、データベース メール機能を利用して、オペレーターに設定したメール アドレスにメールを送信するために必要な設定になるので、これを利用している場合は、移行先（SQL Server 2016 上）でも再設定をしておく必要があります（手動で設定をし直す必要があります）。




なお、レジストリに格納される情報は他にもありますが、これらについては後述します。

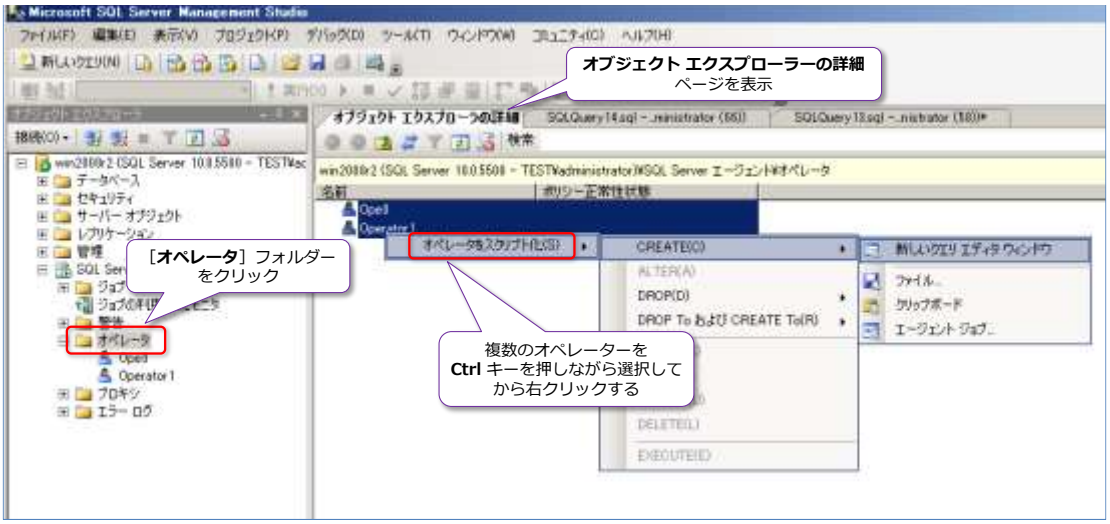
## 5.24 オペレーターの移行

オペレーターは、**msdb** データベース（システム データベース）内に格納されているので、別途移行を行う必要があります（以下の画面は SQL Server 2008 の場合）。




### オペレーターのスクリプト化

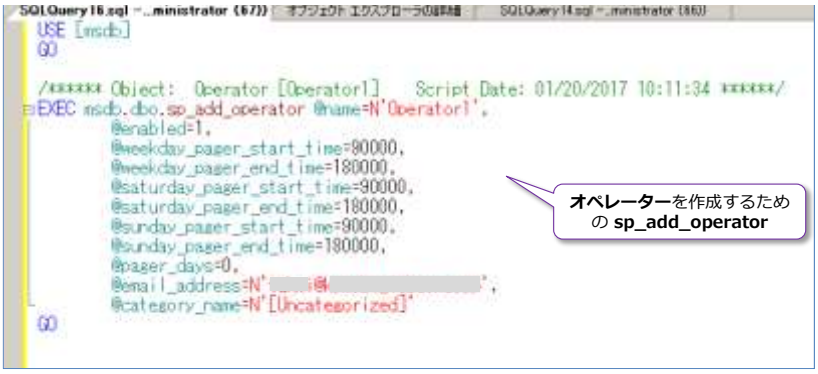
オペレーターをスクリプト化するには、移行元の SQL Server の Management Studio で［**表示**］メニューから［**オブジェクト エクスプローラーの詳細**］をクリックして詳細ページを表示し、次のように［**オペレータ**］フォルダーで複数のオペレーターを Ctrl キーを押しながら選択して、右クリックし、［**オペレータをスクリプト化**］から行います。



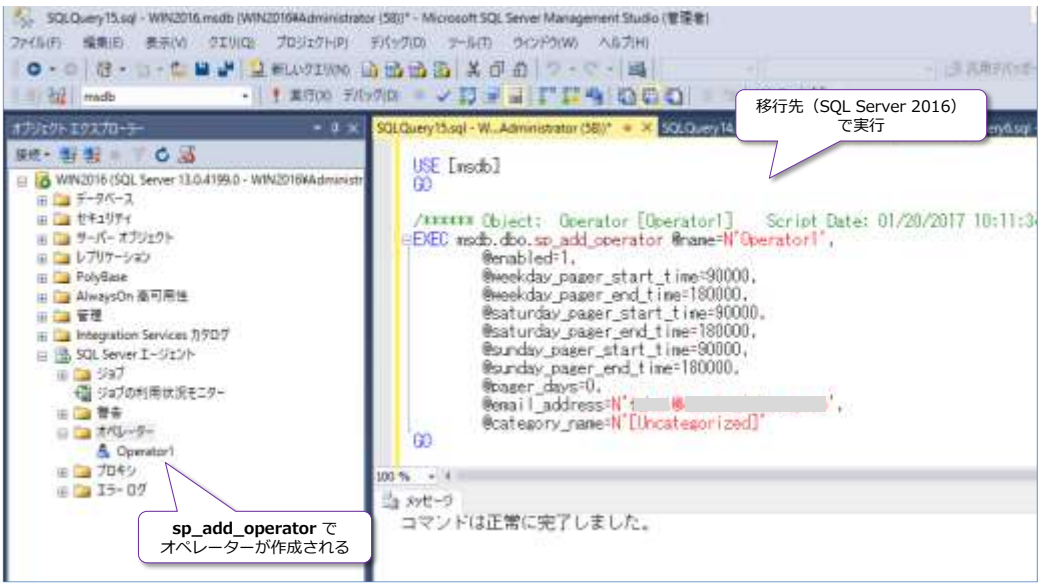

この画面は SQL Server 2008 の場合ですが、他のバージョンでも同じように操作できます（なお、

SQL Server 2005 の SP1 以前を利用している場合は、[**概要**] をクリックして概要ページを開くことで、同じ操作で行えます)。

スクリプト化によって、オペレーターを作成するための **sp_add_operator** が生成されます。

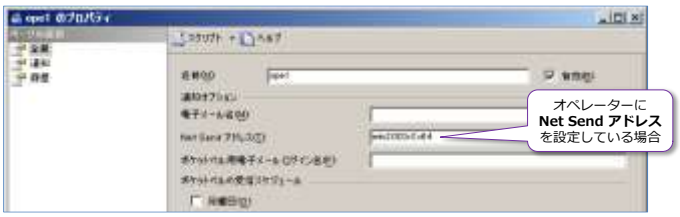



あとは、生成されたスクリプトを移行先（SQL Server 2016 上）で実行すれば、オペレーターを移行することができます。




**Note： net send は Windows Server 2008 以降未サポート**

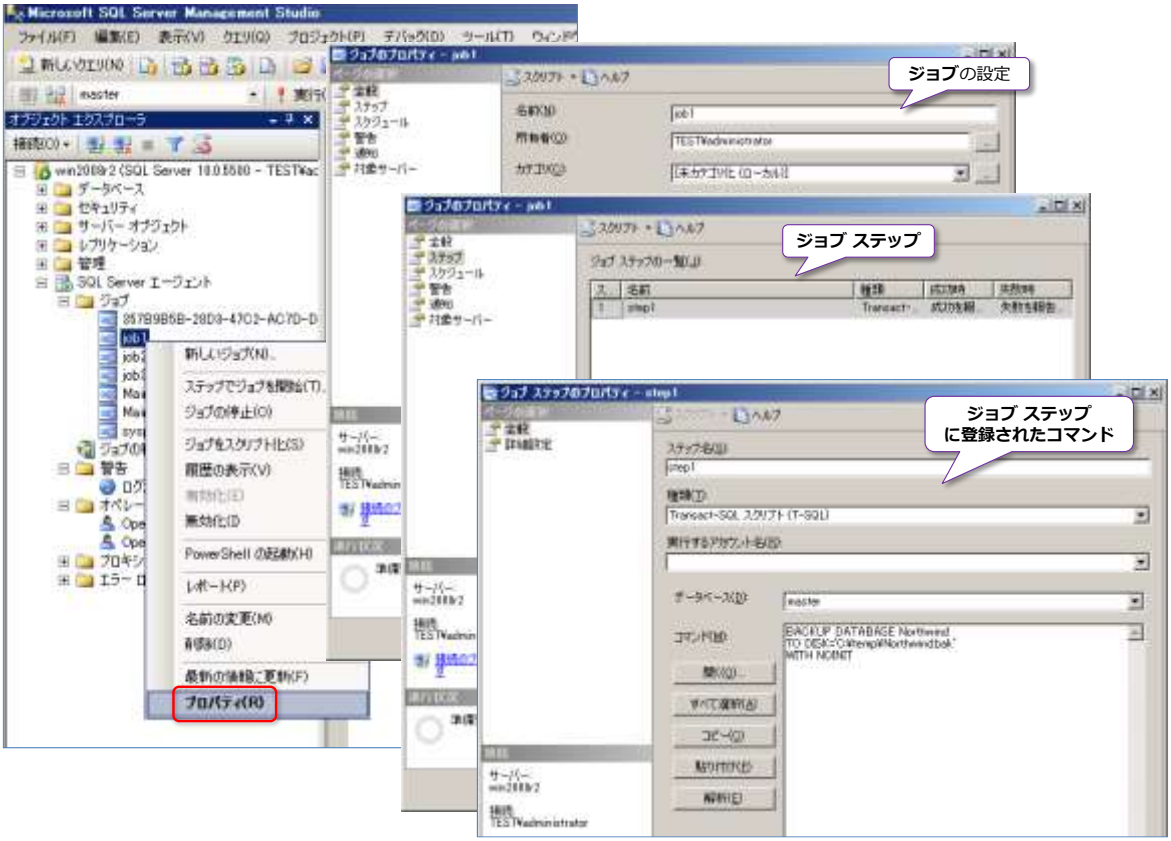
オペレーターの通知オプションで、次のように [**Net Send アドレス**] を設定している場合は、SQL Server 2016 では動作させることができません。




Windows Server 2008 以降の OS では **net send コマンド**および **Messenger サービス**（net send のメッセージを受け取るためのサービス）がサポートされなくなったためです。したがって、オペレーターへの通知に関しては、データベース メール機能を利用（電子メール名を設定）したり、ジョブなどで通知を実装（通知を行うコマンドをジョブ ステップとして追加）したりするようにします。

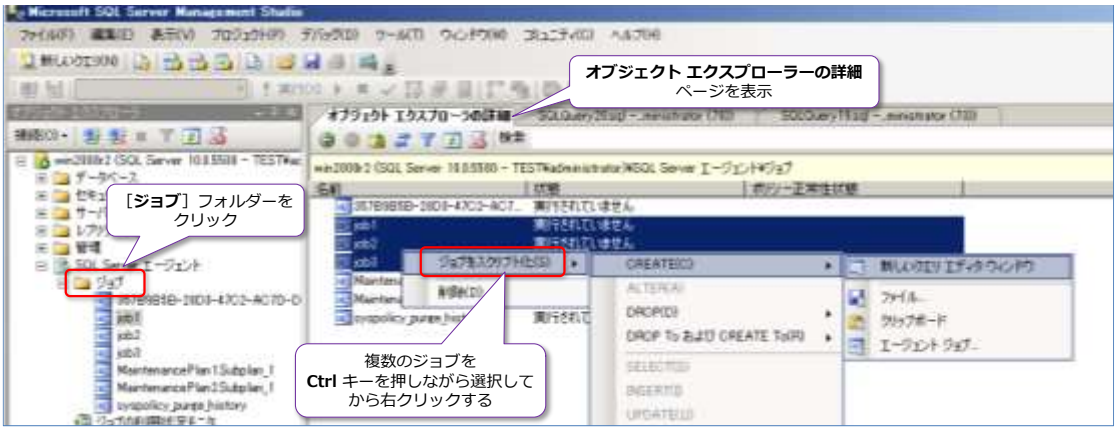
## 5.25 ジョブの移行

ジョブも、**msdb** データベース（システム データベース）内に格納されているので、別途移行を行う必要があります（以下の画面は SQL Server 2008 の場合）。



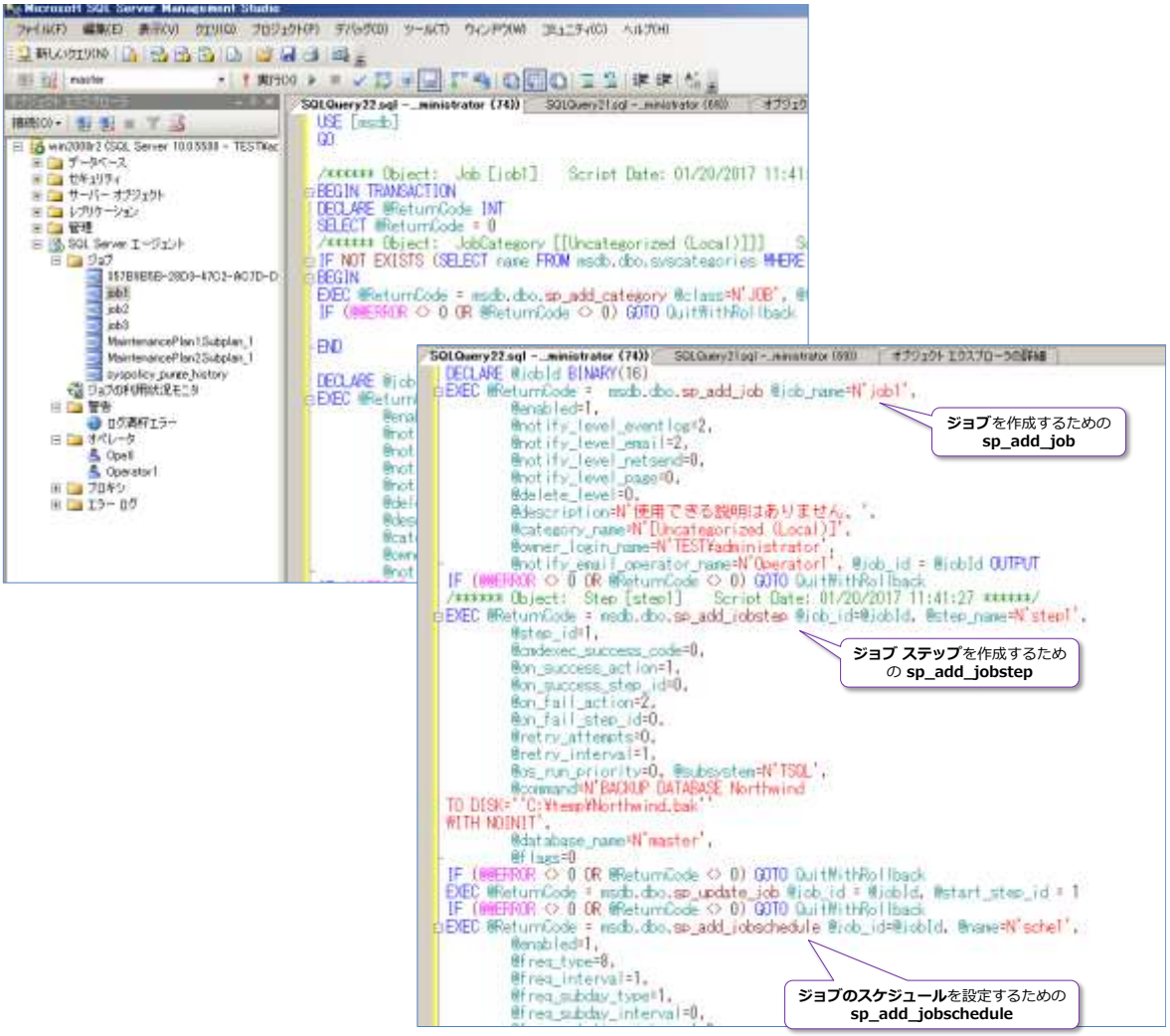

### ジョブのスクリプト化

ジョブをスクリプト化するには、移行元の SQL Server の Management Studio で［**表示**］メニューから［**オブジェクト エクスプローラーの詳細**］をクリックして詳細ページを表示し、次のように［**ジョブ**］フォルダーで複数のジョブを Ctrl キーを押しながら選択して、右クリックし、［**ジョブをスクリプト化**］から行えます。



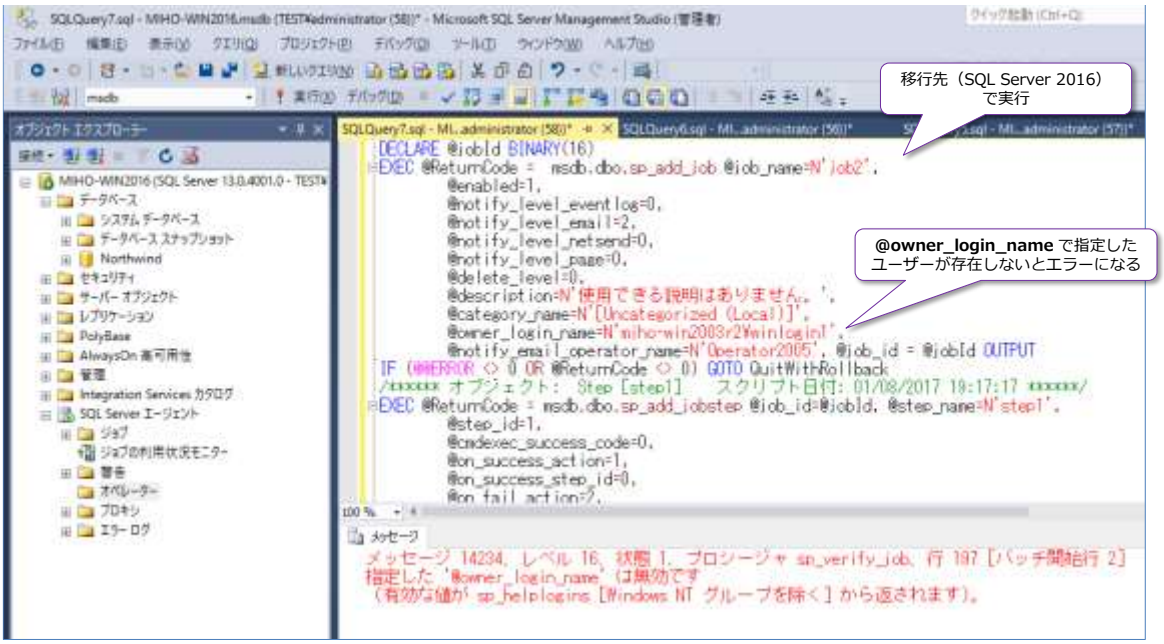
この画面は SQL Server 2008 ですが、他のバージョンでも同じように操作できます（なお、SQL Server 2005 の SP1 以前を利用している場合は、[**概要**] をクリックして概要ページから行います）。

これで、ジョブを作成するための **sp_add_job**、ジョブ内のステップを作成するための **sp_add_jobstep**、ジョブのスケジュールを作成するための **sp_add_jobschedule** が生成されます。




あとは、生成されたスクリプトを移行先（SQL Server 2016 上）で実行すれば、ジョブを移行することができます。

ただし、ジョブの所有者（**@owner_login_name**）に、Active Directory ドメイン ユーザーや Windows ローカル ユーザーを利用している場合は、移行先が同じドメインではなかったり、異なるワークグループ環境である場合には、そのユーザーが存在しないことになるので、次のようにエラーが発生します。




この場合は、**@owner_login_name** を移行先に存在するログイン アカウントの名前に変更する必要があります（**サーバー名¥Administrator** や **sa** などに変更します）。

**Note： 保守計画、SSIS パッケージ、レプリケーション、ログ配布などは除外が必要**

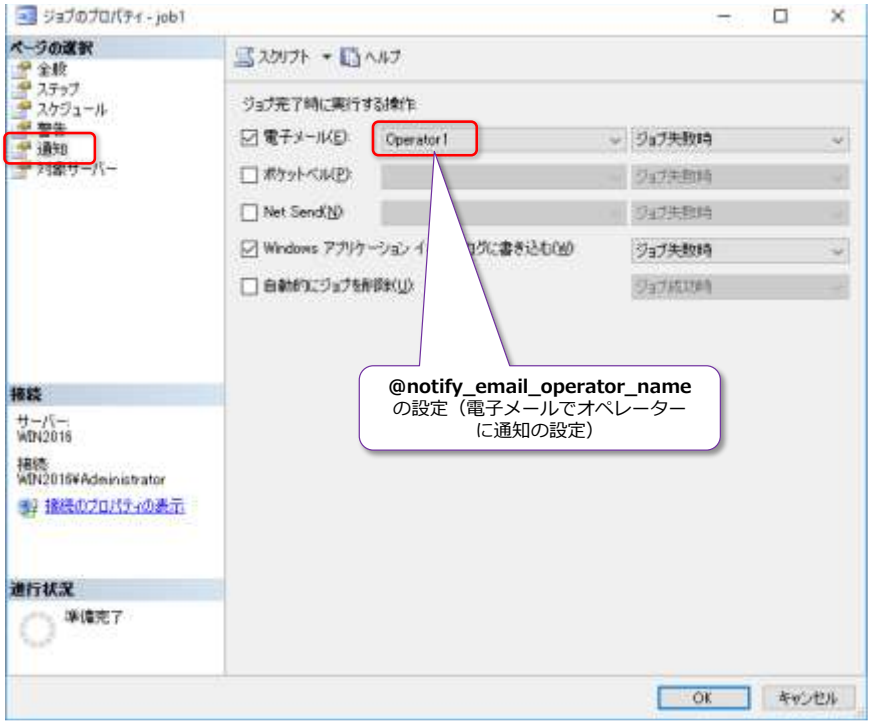
移行元の SQL Server で、**メンテナンス プラン（保守計画）**や **Integration Services の SSIS パッケージ**のスケジュールを設定している場合や、**レプリケーション**や**ログ配布**、**データベース ミラーリング**機能を利用している場合は、これらの機能を動作させるために**ジョブ**が作成されています。

ジョブをスクリプト化するにあたっては、これらのジョブを除外しておく必要があります。これらのジョブは、このまま SQL Server 2016 上へ作成すると、ジョブ実行時にエラーが発生してしまうためです。メンテナンス プランや SSIS パッケージ、レプリケーション、ログ配布、データベース ミラーリングは、SQL Server 2016 上で動作するように別途修正／再作成しなければなりません。

また、**Reporting Services** で**サブスクリプション**を作成している場合も、定期実行を実現するためにジョブ（ID が振られたジョブ）が作成されています。これについても、スクリプト化にあたって除外しておくようにします（∵Reporting Services では、データベースのバックアップと復元によってサブスクリプションを移行することができるため。詳しくは後述）。

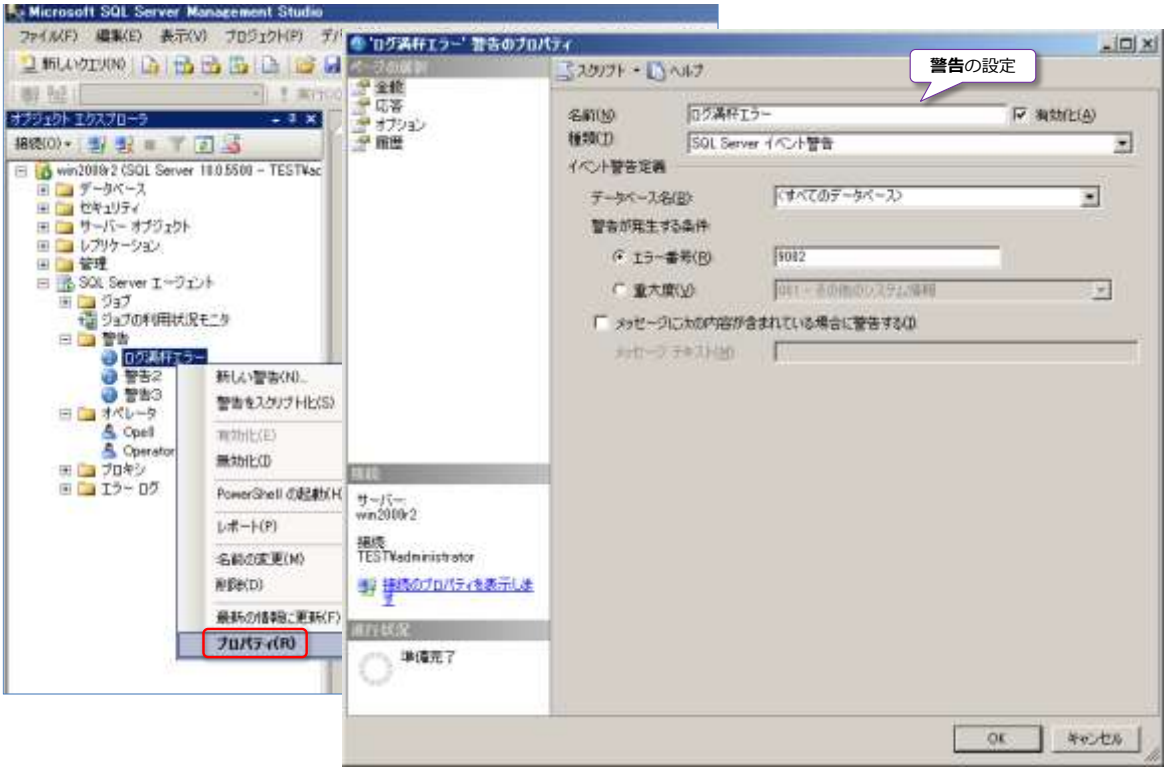
なお、ジョブでオペレーターへの通知設定を行っている場合に、オペレーターを事前に作成していない場合は、次のようにエラーが発生するので、ジョブを作成する前に、オペレーターを作成しておくようにします。


この **@notify_email_operator_name** は、次のようにジョブのプロパティの［**通知**］タブでオペレーター（電子メールでの通知）を設定している場合です。



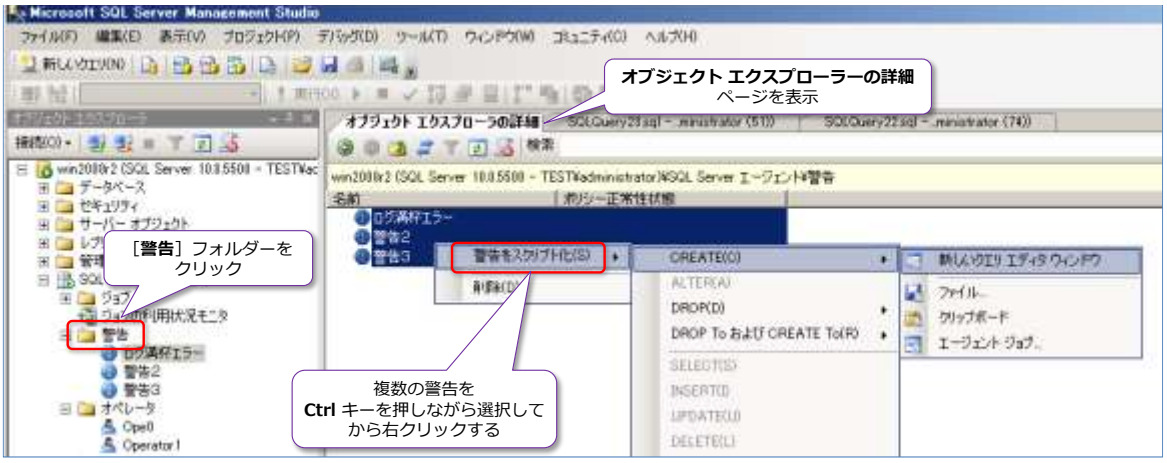
## 5.26 警告の移行

警告も、**msdb** データベース（システム データベース）内に格納されているので、別途移行を行う必要があります（以下の画面は SQL Server 2008 の場合）。




### 警告のスクリプト化

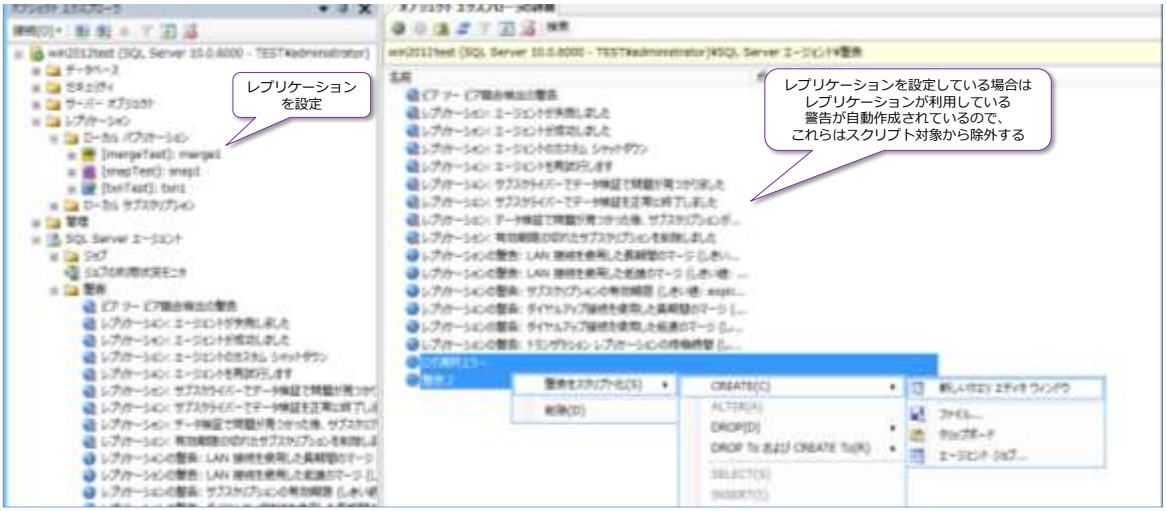
警告をスクリプト化するには、移行元の SQL Server の Management Studio で［**表示**］メニューから［**オブジェクト エクスプローラーの詳細**］をクリックして詳細ページを表示し、次のように［**警告**］フォルダーで複数の警告を Ctrl キーを押しながら選択して、右クリックし、［**警告をスクリプト化**］から行えます。



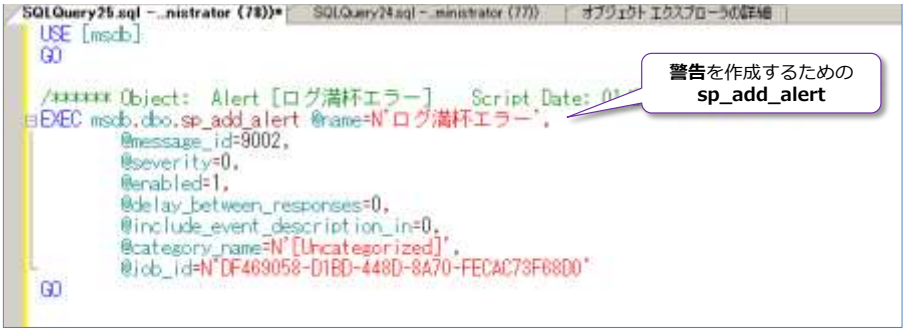

この画面は SQL Server 2008 ですが、他のバージョンでも同じように操作できます（なお、SQL

Server 2005 の SP1 以前を利用している場合は、[**概要**] をクリックして概要ページから行います)。

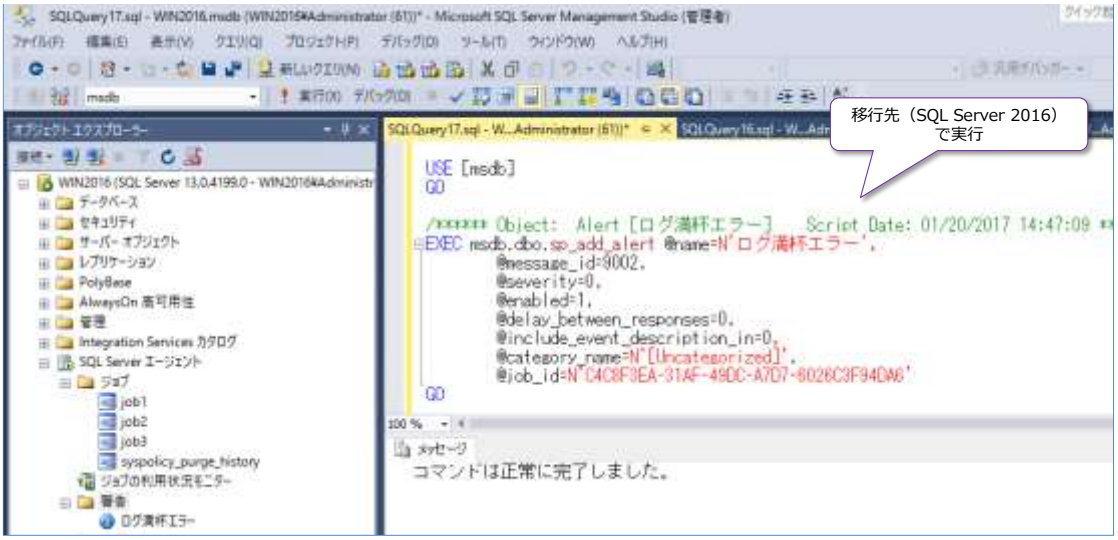
なお、**レプリケーション**を設定している場合には、次のようにレプリケーション関連の警告が自動的に作成されているのですが、これについては別途移行する必要がある（移行先で再作成をする必要がある）ので、レプリケーション関連の警告は、スクリプト生成から除外しておくようにします。




スクリプト化を行うと、次のように警告を作成するための **sp_add_alert** が生成されます。

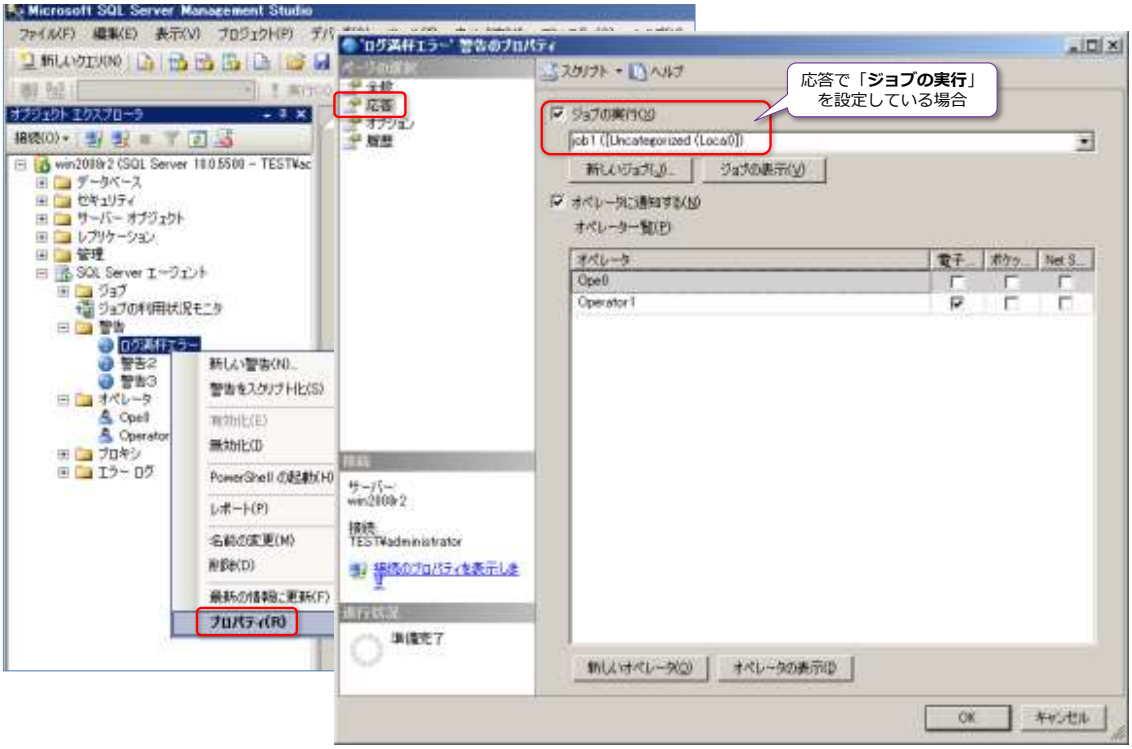



あとは、生成されたスクリプトを移行先（SQL Server 2016 上）で実行すれば、警告を移行することができます。

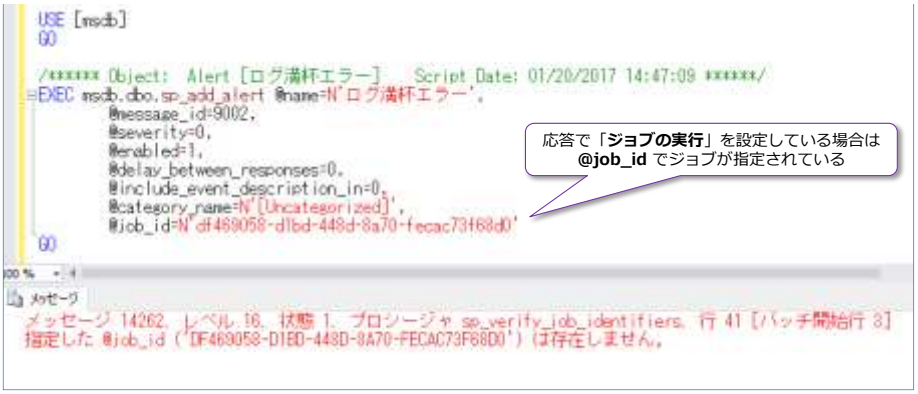


## 警告の応答で「ジョブの実行」を設定している場合の追加作業

警告の応答で、次のように［**ジョブの実行**］を設定している場合は、スクリプトの実行時に追加の作業が必要になります（以下の画面は SQL Server 2008 の場合）。



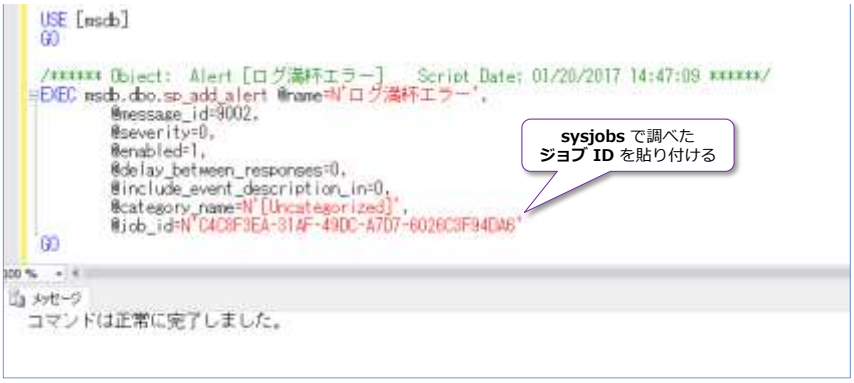

この場合は、警告の作成のスクリプト（**sp_add_alert**）を移行先で実行すると、次のようにエラーになります。




［**ジョブの実行**］を設定している場合は、**@job_id** でジョブの内部的な ID が設定されていて、移行先では、ジョブを作成し直しているので、この ID が違うものが設定されてしまっているためです。したがって、この **@job_id** で指定するジョブ ID を移行先の環境で設定されたものに変更する必要があります。ジョブ ID は、次のように **sysjobs** システム テーブルを参照することで確認することができます。

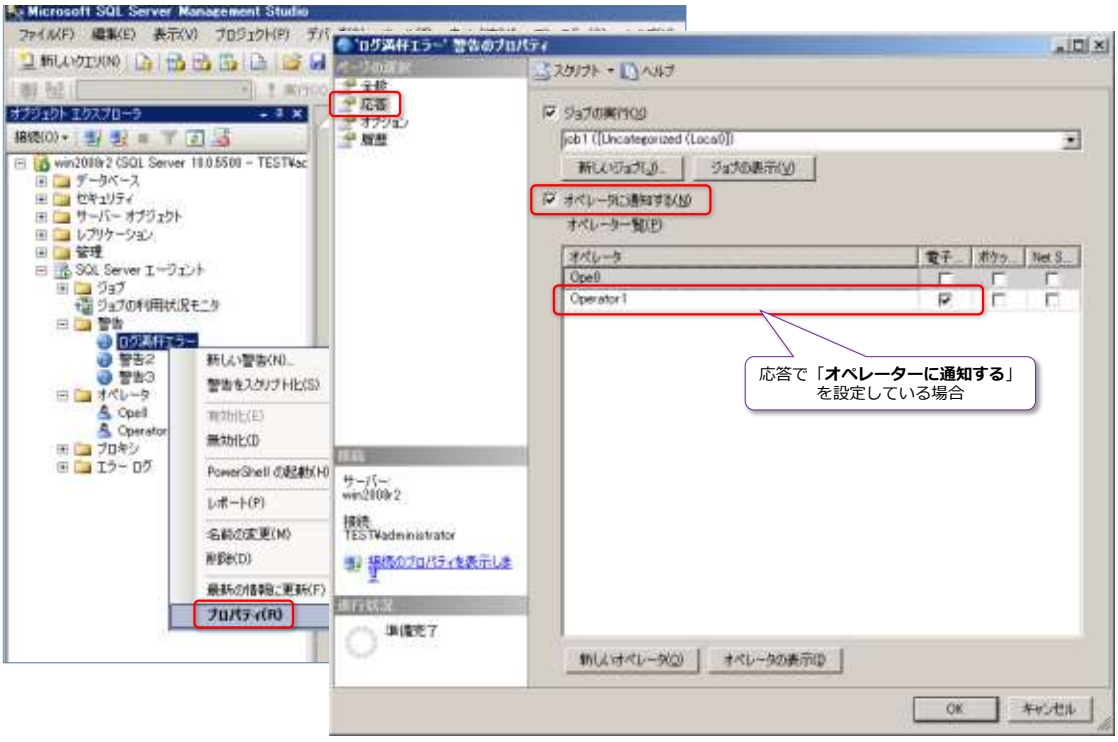
```
USE msdb
SELECT * FROM sysjobs
```


**job_id** 列でジョブ ID を確認できるので、**sp_add_alert** の **@job_id** をこの値に変更します。



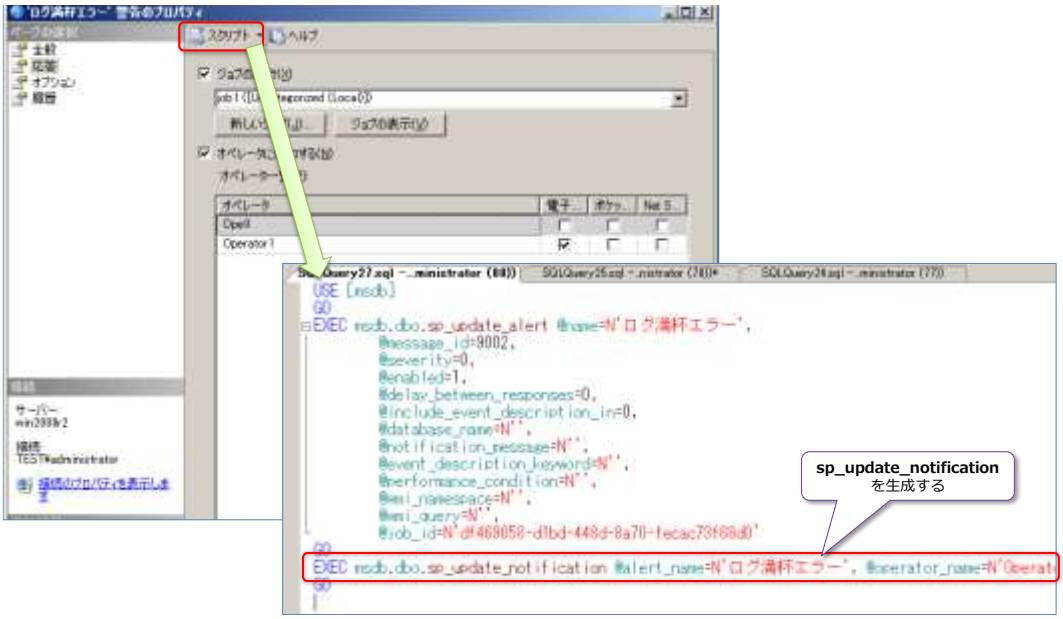

## 警告でオペレーターへの通知を設定している場合の追加作業

警告で、次のようにオペレーターへの通知（警告の応答）の設定をしている場合は、追加の作業が必要になります（以下の画面は SQL Server 2008 の場合）

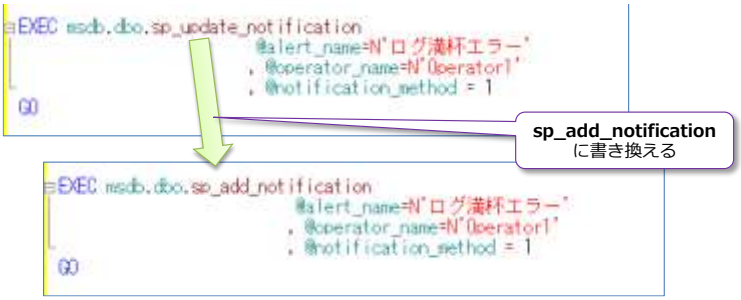



この設定は、**sp_add_notification** を実行すれば良いのですが、SQL Server 2005 の SP1 以前を利用している場合は、警告をスクリプト化したときに、自動的にこれを生成してくれていまし

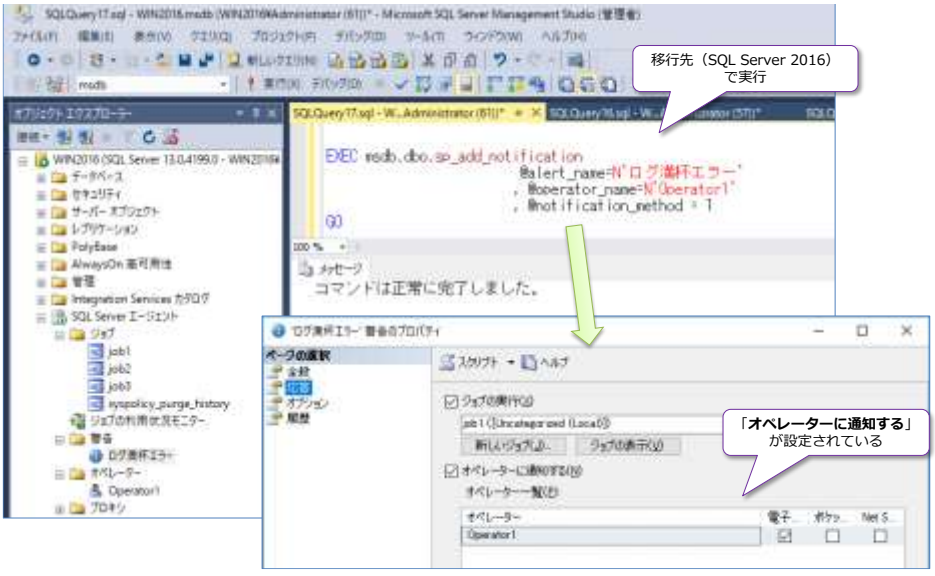
た。しかし、SP2 以降は、これを生成してくれないので、上記の画面で、[**スクリプト**] ボタンをクリックして、**sp_update_notification** を生成します。




**sp_update_notification** は、既存の通知を変更するものなので、この "**update**" という部分を、次のように "**add**" に変更して、新しく通知を作成するように変更します。

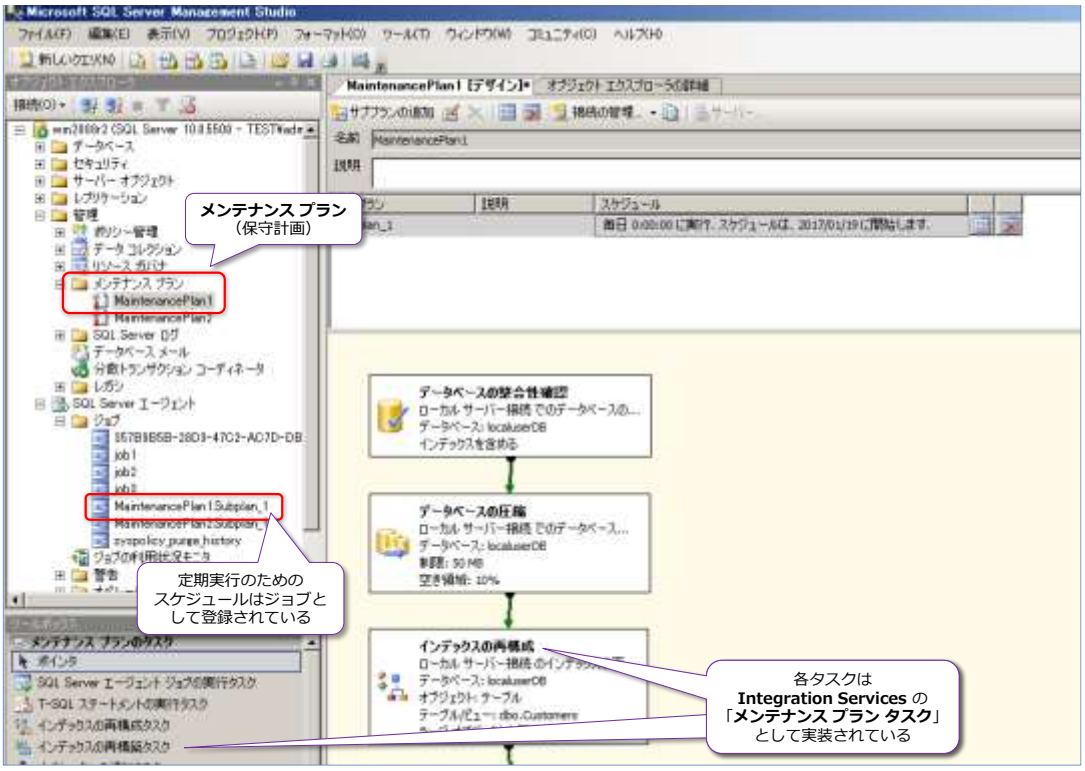



あとは、修正した **sp_add_notification** を移行先（SQL Server 2016 上）で実行すれば、オペレーターへの通知を設定することができます。

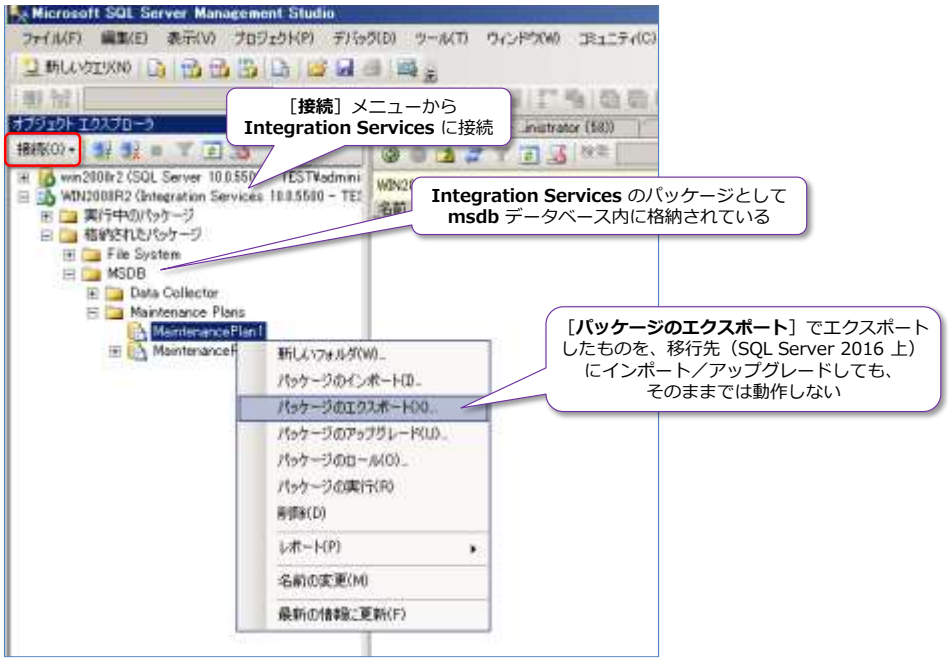


## 5.27 メンテナンス プラン（保守計画）の移行

移行元の SQL Server 上で作成した**メンテナンス プラン**（保守計画）は、スクリプト生成を行うことができず、移行のためのツールも用意されていないので、移行先（SQL Server 2016 上）で再作成する（手動で作り直す）のがお勧めです（以下の画面は SQL Server 2008 の場合）。




メンテナンス プランの実体は、**Integration Services** の**パッケージ**（メンテナンス プランのタスク）として実装されているので、Management Studio で **Integration Services** へ接続すれば、次のように確認することもできます。



Integration Services 上のパッケージは、上の画面のように［**パッケージのエクスポート**］でメンテナンス プランをエクスポートすることも可能ですが、これを移行先（SQL Server 2016 上）にインポート／アップグレードしても、そのままでは動作しないので、メンテナンス プランは、移行先で再作成するのがお勧めです。

**Note： パッケージのエクスポートとインポートで移行したい場合**

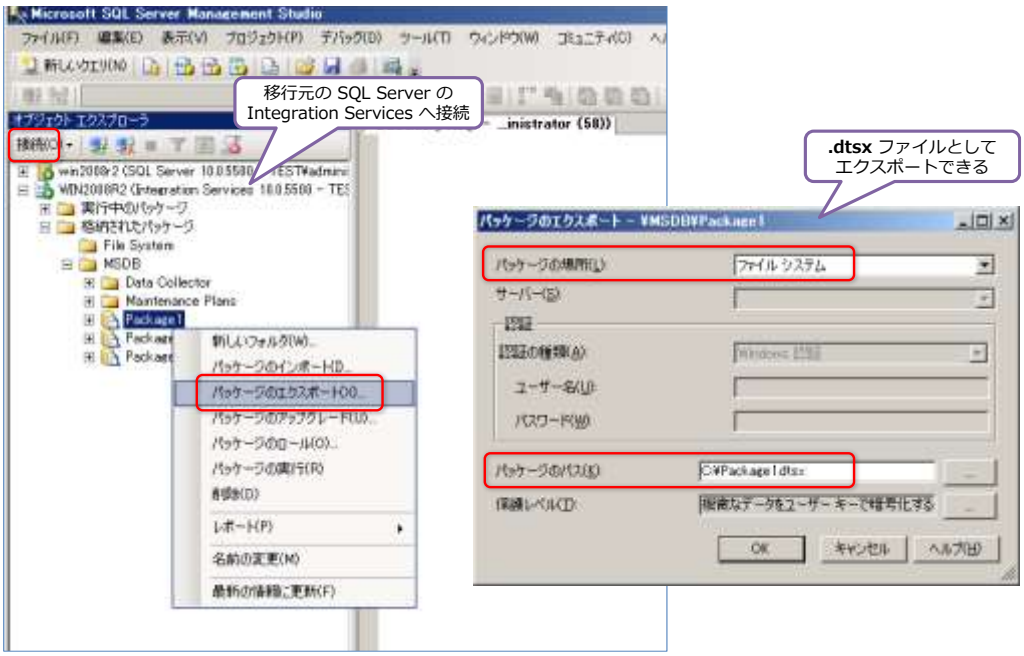
メンテナンス プランは、再作成をして移行するのがお勧めですが、パッケージのエクスポートとインポートを利用して、移行したい場合は、次のように操作します。

1. 移行元でメンテナンス プランのパッケージをエクスポートする（Integration Services に接続）
2. エクスポートしたパッケージ（.dtsx）を任意のテキスト エディターで開いて、移行元のマシン名が埋め込まれている場所を、移行先のマシン名に置換して、ファイルを上書き保存する
3. 移行先で、手順 2 で修正したパッケージをインポートする（Integration Services に接続）
4. 移行先で、インポートしたメンテナンス プランを一度開いて保存する（データベース エンジンに接続して、「**管理**」フォルダーの「**メンテナンス プラン**」フォルダーからインポートしたものを開く）
5. レポートの保存先や、バックアップの出力先、クリーンアップ対象のフォルダー パスなどを、移行先の環境に合わせて修正する

という手順になります。このように、パッケージのエクスポート／インポートを利用した方法は、少し面倒で、メンテナンス プランを再作成するのと手間があまり変わりません。

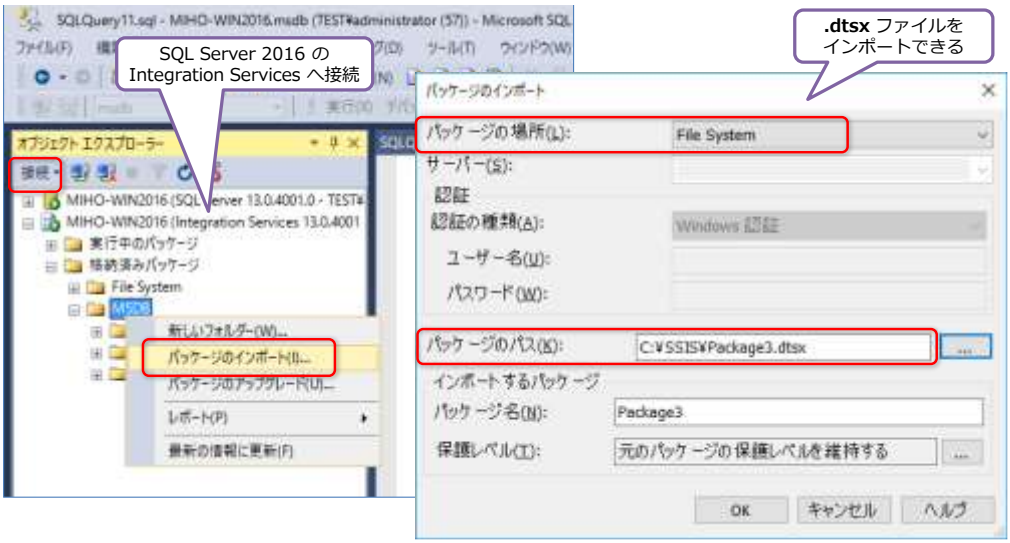
## 5.28 Integration Services の SSIS パッケージの移行

**Integration Services** を利用している場合は、**SSIS パッケージ**も別途移行を行う必要があります。ファイル（**.dtsx**）として利用している SSIS パッケージであれば、そのファイルを移行先へコピーし、**msdb** データベース（システム データベース）内に格納している SSIS パッケージであれば、次のように Management Studio の［**接続**］メニューから **Integration Services** へ接続して、［**パッケージのエクスポート**］をクリックします（画面は SQL Server 2008 ですが、他のバージョンでも同じように操作できます）。

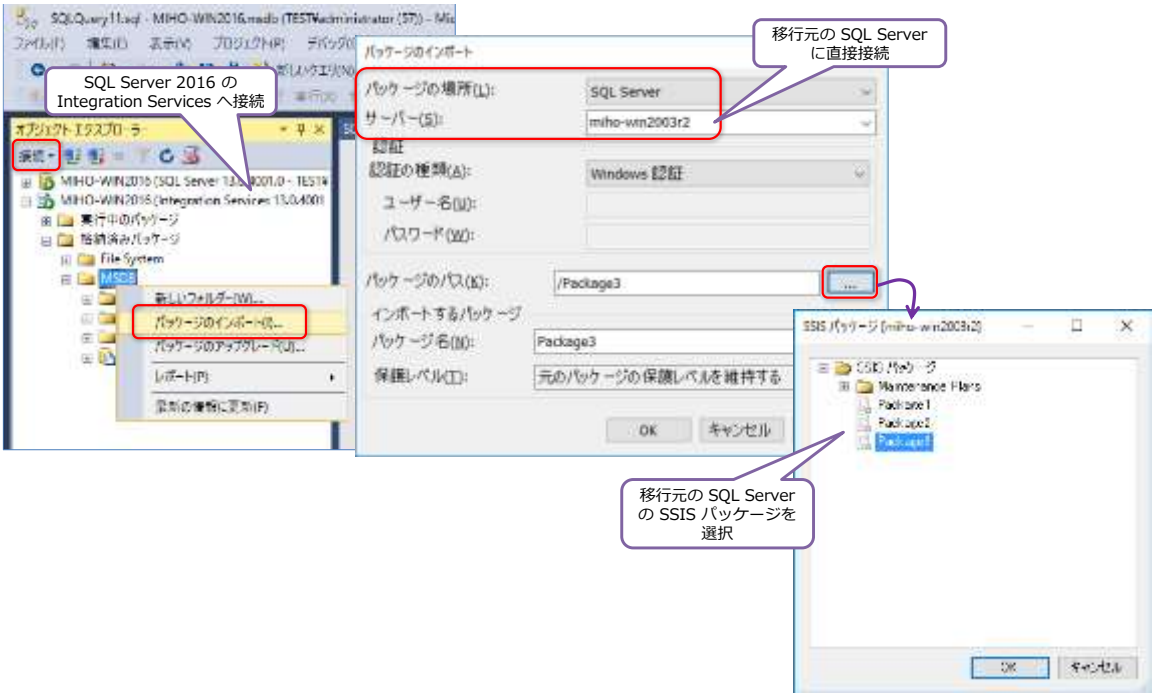



［**パッケージの種類**］で「**ファイル システム**」を選択することで、ファイル（**.dtsx**）としてエクスポートすることができます。これを移行先のサーバーに、ファイル コピーします。

移行先の SQL Server 2016 では、次のように Management Studio で Integration Services へ接続して、［**パッケージのインポート**］をクリックして、［**パッケージの場所**］で「**File System**」を選択することで、ファイル（**.dtsx**）をインポートすることができます。



もし、移行先の **SQL Server 2016** から、移行元の SQL Server にリモート接続できる環境の場合は、次のように［**パッケージの場所**］で「**SQL Server**」を選択することで、ネットワークを介して SSIS パッケージをインポートすることも可能です。



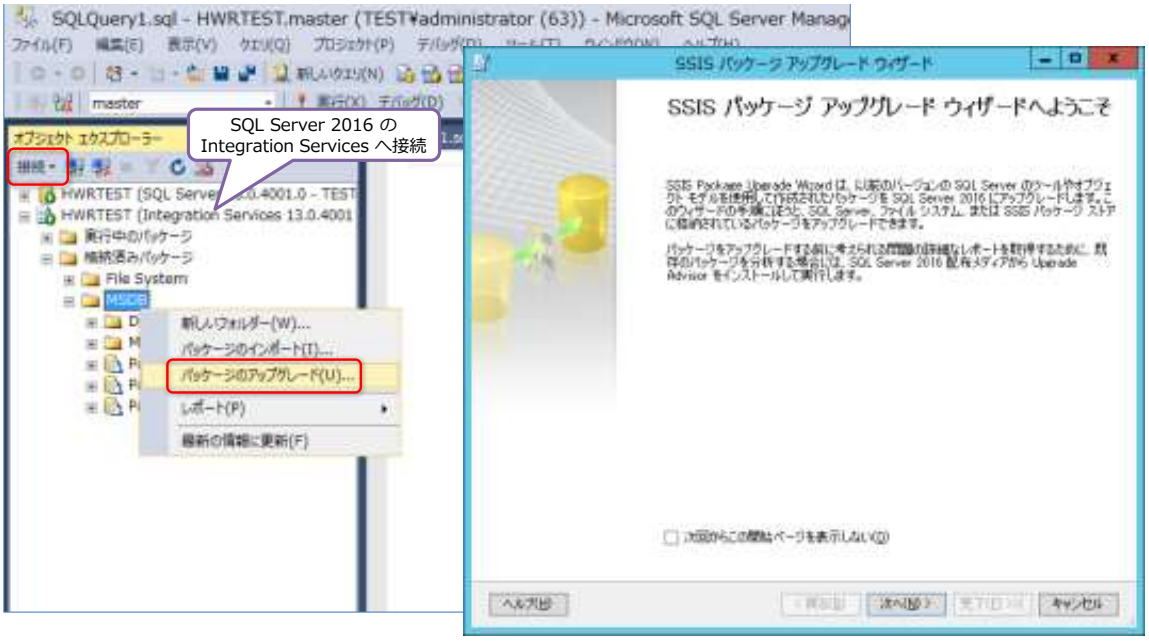

## SSIS パッケージのアップグレード

パッケージのインポートでは、**SSIS パッケージ**のアップグレード（SQL Server 2016 形式のパッケージへの変換）は行われません。アップグレードを行わなかった影響については、オンラインブックの次のトピックを一読することをお勧めします。

Integration Services パッケージのアップグレード
http://msdn.microsoft.com/ja-jp/library/cc280546.aspx

古い形式のまま実行できる SSIS パッケージの場合は、このトピックに記載されているように、パフォーマンスのオーバーヘッドがある（∵内部的にパッケージの変換を行った上で実行しているため）ことに注意する必要があります。また、古い形式のままでは実行できないもの（例えば、SQL Server 2012 以降に廃止された ”DTS 2000 パッケージ実行タスク” を含んでいるパッケージなど）がある場合は、それらを修正していく必要があります。

SSIS パッケージのアップグレードは、「**アップグレード ウィザード**」を利用することで簡単に SQL Server 2016 形式に変換することができます。アップグレード ウィザードは、SSDT や、Management Studio から起動することができますが、Management Studio を利用する場合は、次のようにオブジェクト エクスプローラーの [**接続**] メニューで Integration Services へ接続して、[**MSDB**] フォルダーを右クリックして、[**パッケージのアップグレード**] をクリックします（これで SQL Server 内に格納されたすべての Integration Services をまとめてアップグレードすることができます）。




このウィザードの手順や、SSIS パッケージのその他の情報については、第 3 章の 3.23 で説明しているので、こちらもぜひご覧いただければと思います。

## 5.29 再作成が必要なもの

移行元の SQL Server 上で、レプリケーションやログ配布、データベース ミラーリングなど、サーバー間の連携機能を利用している場合には、移行先で再作成する（手動で作り直す）必要があります。再作成が必要なものをまとめると次のようになります（移行元で利用している場合のみ）。

- データベース メールの設定（前掲）
- 警告の応答で「ジョブの実行」や「オペレータに通知」を設定している場合（前掲）
- メンテナンス プラン（前掲）
- レジストリに格納された設定（後述）
- NTFS アクセス許可やユーザー権利などの OS の設定（後述）
- SQL Server Audit やリソース ガバナー、ポリシーベースの管理、パフォーマンス データ コレクションなどのサーバーの管理機能（SQL Server 2008 からの新機能）
- レプリケーションやログ配布、可用性グループ、データベース ミラーリングなどのサーバー間の連携機能
- WSFC（Windows Server フェールオーバー クラスタリング）の SQL Server インスタンス

これらについては、移行元で利用している場合には、移行先で再作成をする必要があります。

## 5.30 レジストリに格納されている情報の再設定

移行時は、**レジストリ**に格納されている情報も再設定する必要がありますが、レジストリに格納されている情報はほとんどありません（∵SQL Server の動作に関する設定は、ほとんどがシステムデータベース内に格納されているため）。

レジストリに格納されている主な情報は、次のとおりです。

| | **レジストリに格納される主な情報** |
|---|---|
| **SQL Server サービス関連**<br>構成マネージャー（Configuration Manager）ツールで設定したものなど | ・サービスの自動起動の設定<br>・サービス起動時のトレース フラグ（起動時のパラメーター）<br>・サービス アカウントの設定<br>・セキュリティ モード（認証モード）<br>・TCP ポート番号<br>・監査レベル（成功／失敗のログインをログへ記録するかどうか） |
| **SQL Server Agent サービス関連**<br>SQL Server エージェントのプロパティで設定したものなど | ・サービスの自動起動の設定<br>・サービス アカウントの設定<br>・［警告システム］ページの設定（メール セッションでのデータベース メールの設定、緊急時のオペレーターなど）<br>・ジョブ履歴の最大行数<br>・予期しない停止時の自動再起動の設定<br>・パフォーマンス条件警告でのパフォーマンス カウンターの更新頻度<br>・CPU をアイドルとみなす秒数／使用率（ジョブのスケジュールで利用している場合） |

サービスの自動起動や、サービス アカウントの設定、セキュリティ モード（認証モード）については、SQL Server 2016 のインストール時に、移行元と同じになるように設定すれば大丈夫です。

弊社のお客様では、デッドロックをログへ記録するためのトレース フラグ「**1222**」の設定や、TCP ポート番号の設定、データベース メールの設定（SQL Server エージェントのプロパティの［警告システム］タブの設定）あたりが利用しているものです。

以上のように、レジストリに格納される情報は少なく、あまり変更することのない設定が多いと思います。レジストリに関しては、第４章の 4.3 でも詳しく説明しているので、こちらもご覧いただければと思います。

## 5.31 OS の設定を再設定する

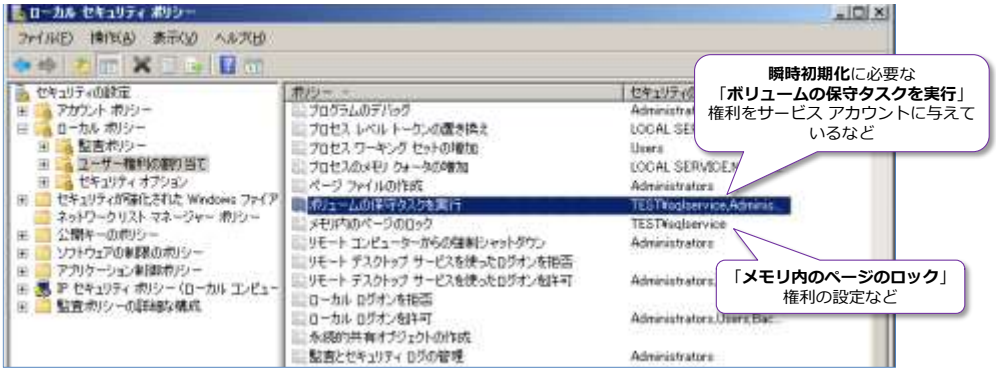
**OS の設定**についても、移行元で設定を変更しているものがある場合は、それらを再設定する必要があります（フォルダー構成や NTFS アクセス許可、ユーザーの権利、共有フォルダーなど）。

### NTFS アクセス許可の再設定

移行後に、よくありがちなのが NTFS アクセス許可が足りないために、エラーが発生しているというケースです。特に、**サービス アカウント**への NTFS アクセス許可が設定されていない場合には、多くの場面でエラーに遭遇する可能性があるので、注意してみてください。OS が変わることによって、ドライブに対するセキュリティが強化されていて、ドライブ直下のファイルへの NTFS アクセス許可が変わっていたりするので、うまく動作しない場合は、NTFS アクセス許可を確認してみてください。

### ユーザーの権利（瞬時初期化など）の再設定

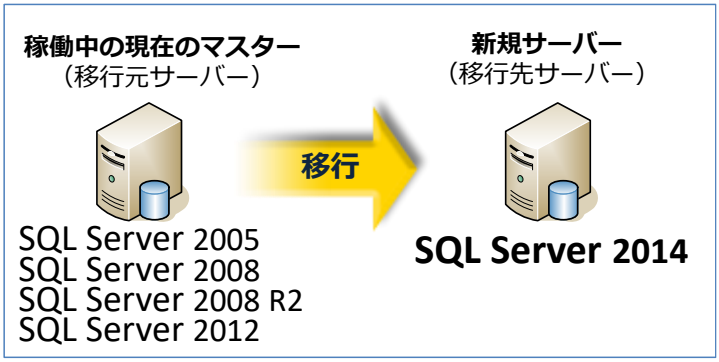
移行元で、**ユーザーの権利**を変更している場合には、移行先でも変更するようにします。例えば、次のように、**瞬時初期化**に必要となる「**ボリュームの保守タスクを実行**」権利をサービス アカウントに設定していたり、「**メモリ内のページのロック**」権利を設定しているという場合には、移行先でも同じように設定するようにします。




この設定は、[**管理ツール**] メニューの [**ローカル セキュリティ ポリシー**] ツールから行えます。

## 5.32 ケース3「新規サーバーへの移行」のまとめ

ケース3 での操作手順をまとめると、次のようになります。




1. **Data Migration Assistant** による移行チェックを行う
2. **新規サーバー**へ **Windows Server 2012**（X64 版）以上の OS をインストールする（移行元と同じ OS である必要はありません）
3. **Active Directory ドメイン**環境の場合は、**新規サーバー**をドメインに参加させる
4. **SQL Server 2016** を**インストールするためのソフトウェア要件**を確認する

   OS は、Windows Server 2012 以降の X64 版をサポート、Windows Server 2012 R2 の場合は KB 2919355 が必要など。データベース メール機能を利用している場合は、.NET Framework 3.5 SP1 をインストールしておく必要がある
5. **新規サーバー**へ **SQL Server 2016** をインストールする
6. **新規サーバー**へ **SQL Server 2016** の**最新の修正プログラム**（CU や Service Pack など）をインストールする

   CU2 には、性能向上に関する修正が入っているので、できる限り最新の修正プログラムを適用しておくことをお勧めします。

   Reporting Services を利用している場合は、SP1 に対する重要な更新プログラムがあるので、CU4（SP1＋CU1）以上を適用しておくことをお勧めします。
7. **Management Studio**（管理ツール）の最新版をダウンロードして、インストールする（オプション）
8. **移行元サーバー**（SQL Server 2005／2008／2008 R2／2012／2014）の**データベース**を、**新規サーバー**（SQL Server 2016）へ移行する（バックアップと復元機能を利用）
9. **統計**（Statistics）を更新する
10. **フルテキスト インデックス**を再構築する（フルテキスト検索機能を利用している場合）
11. **データベースの互換性レベルを 130** へ上げる（オプション）

互換性レベルの影響については**クエリ ストア**機能を利用することで簡単にチェック可能（実行プランの比較や、プラン強制もできる）。

12. **データベースの所有者**を確認／設定する（∵所有者が空の場合には、**データベース ダイアグラム**と後述の **SQL CLR オブジェクト**が動作しないため）
13. **SQL CLR オブジェクト**の権限セットで「**UNSAFE**」または「**外部**」を利用している場合は、**TRUSTWORTHY** オプションを有効化する
14. **システム データベース**関連のオブジェクトを移行する（ログイン アカウントやサーバー ロール、tempdb の設定、リンク サーバー、ユーザー定義エラー、構成オプション、データベース メール、ジョブ、警告、オペレーターなどのうち、移行元で設定を変更／利用しているものがある場合は、それらを移行する）
15. **レジストリ**に格納されている情報を再設定する（サービスの自動起動やサービス アカウント、認証モード、TCP ポート番号、起動時パラメーターでのトレースフラグの設定などのうち、移行元で設定を変更しているものがある場合は、それらを再設定する）
16. **OS の設定**で、移行元で変更しているものがある場合は、それらを再設定する（フォルダー構成や、NTFS アクセス許可、ユーザーの権利、共有フォルダーなど）
17. **メンテナンス プラン**（保守計画）を利用している場合は、メンテナンス プランを再作成する
18. **BIDS**（Business Intelligence Development Studio）または **SSDT-BI**（SQL Server Data Tools - Business Intelligence）を利用している場合は、**SSDT** の最新版をダウンロードして、インストールする（オプション）
19. **Integration Services** を利用している場合は、SSIS パッケージを移行する
20. **SQL Server Audit** や**リソース ガバナー**、**パフォーマンス データ コレクション**などのサーバー管理機能を利用している場合は、これらを再設定する
21. **レプリケーション**や**ログ配布**、**可用性グループ**、**データベース ミラーリング**などのサーバー間の連携機能を利用している場合は、これらを再設定する

新規サーバーへの**データベースの移行**は、**標準のバックアップと復元**機能を利用して、簡単に行うことができます。

この移行方法を利用するメリットは、**OS** を簡単に変更できること（**Windows Server 2003／2003 R2** や **Windows Server 2008／2008 R2** から **Windows Server 2012** や **2012 R2**、**Windows Server 2016** へ変更するなど）、クロス プラットフォーム（移行元が **32 ビット**で、移行先が **64 ビット**など）でも関係がないこと、**Microsoft Azure** などの**クラウド環境**であったとしてもデータベースの移行が行えること、**SQL Server 2005 からの移行**にも対応していることです。

**各種の設定**（**システム データベース**や**レジストリ**に格納されている設定、**メンテナンス プラン**な

ど）は、移行元で設定を変更していたり、該当オブジェクトを利用したりしている場合には、再設定／再作成が必要になります。**システム データベース**に格納されているものに関しては、ほとんどのものが GUI 操作で**スクリプト生成**することができるので、簡単に移行することができます。**レジストリ**に格納されているものに関しては、SQL Server はレジストリをほとんど利用していないので、こちらも再設定は簡単です。

ケース１ やケース２ との大きな違いとしては、**サーバー管理に関する機能**（リソース ガバナーや SQL Server Audit、ポリシー ベースの管理など）や、**サーバー間の連携の機能**（ログ配布やレプリケーション、データベース ミラーリング、可用性グループなど）に関しては、再設定をする必要がある点です。

## 5.33 移行にかかる時間（ダウンタイム）の見積もり

移行にかかる時間（移行時のダウンタイム）は、次のように考えることができます。

| | 移行作業 | 作業時間の見積もり | 説明 |
|---|---|---|---|
| 1 | Data Migration Assistant による互換性のチェックを行う | - | 事前作業。<br>データベースの互換性を事前にチェックしておく |
| 2 | 新規サーバーへ OS をインストール<br>（SQL Server 2016 のインストールには Windows Server 2012 以上が必要） | 20分～2時間 | OS として Windows Server 2012 R2 を利用する場合は **with Update のパッケージ**を利用することで、OS と Update（**KB 2919355**）のインストールを同時に行うことが可能。<br>作業時間は、ハードウェア環境によって作業時間が前後する。OS の再起動に 10分程度かかるサーバー機もあるため、機種によっては 1時間以上になることも。 |
| 3 | Active Directory ドメイン環境の場合は、<br>新規サーバーをドメインに参加させる | 数分～15分 | ドメインの参加には、OS の再起動が伴い、OS の再起動にかかる時間は、サーバー機によって大きく異なる |
| 4 | 新規サーバーへ SQL Server 2016 をインストールする | 15分～<br>1時間30分 | ハードウェア環境や、以前のバージョンでインストールされていた機能の種類によって作業時間が前後する。<br>**with SP1** など、Service Pack 付きのパッケージを利用することで、SQL Server 2016 と SP のインストールを同時に行うことも可能 |
| 5 | 新規サーバーへ SQL Server 2016 の最新の修正プログラム（CU や SP）をインストールする | 15分～<br>1時間30分 | ハードウェア環境や、インストールしている機能の種類によって作業時間が前後する。<br>**Reporting Services** を利用する場合は、SP1 に対する重要な更新プログラムがあるので **SP1+CU1** または **RTM+CU4** の適用を推奨 |
| 6 | **データベースを移行**する。<br>移行元サーバー（SQL Server 2005／2008／2008 R2／2012／2014）のデータベースを、<br>新規サーバー（SQL Server 2016）へ移行する | 大きく変動 | 標準のバックアップと復元機能を利用して、データベースを移行する。<br>データベース サイズやストレージの性能、ネットワーク速度によって大きく変化する |
| 7 | Management Studio の最新版をインストールする（オプション） | 10分～1時間 | 事前にインストーラーをダウンロードしておくことで<br>インストール時間を短縮できる |
| 8 | 統計（Statistics）を更新する | 数分～ | テーブル サイズが大きい場合は実行に時間がかかることがある |
| 9 | フルテキスト インデックスを再構築する<br>（フルテキスト検索機能を利用している場合のみ） | 数分～ | テーブル サイズが大きい場合は実行に時間がかかる。<br>フルテキスト インデックスの作成時と同等の時間がかかることに注意 |
| 10 | データベースの互換性レベルを 130 へ上げる（オプション） | 数分 | 必要に応じて、クエリストア機能を利用して、<br>互換性レベルの影響（特に性能面）をチェックする |
| 11 | データベースの所有者を確認／設定する | 数分 | 所有者が空の場合には、データベース ダイアグラムと、SQL CLR オブジェクトで権限セットを「UNSAFE」または「外部」を利用しているものが動作しないため |
| 12 | SQL CLR オブジェクトの権限セットで「UNSAFE」または「外部」を利用している場合は、TRUSTWORTHY オプションを有効化する | 数分 | |
| 13 | システム データベース関連のオブジェクトを移行する | 変動 | ログイン アカウントやサーバー ロール、tempdb の設定、リンク サーバー、ユーザー定義エラー、構成オプション、データベース メール、ジョブ、警告、オペレーターなどのうち、移行元で設定を変更／利用しているものがある場合は、それらを移行する。<br>設定している数によって、設定時間が変動する |
| 14 | レジストリの設定を移行元と同様にする | 数分 | サービスの自動起動やサービス アカウント、認証モード、TCP ポート番号、起動時パラメーターでのトレースフラグの設定などのうち、移行元で設定を変更しているものがある場合は、それらを再設定する |
| 15 | OS の設定で、移行元で変更しているものがある場合は、それらを再設定する | 数分～数十分 | フォルダー構成や、NTFS アクセス許可、ユーザーの権利、共有フォルダーなど、状況によって作業時間が左右する |
| 16 | メンテナンス プラン（保守計画）を利用している場合は、メンテナンス プランを再作成する | 数十分 | |
| 17 | SSDT のインストール（オプション） | 10分～1時間 | 事前にダウンロード版のインストーラー（iso）をダウンロードしておくことで、インストール時間を短縮できる |
| 18 | Integration Services を利用している場合は、SSIS パッケージを移行する | 変動 | 利用している SSIS パッケージの数によって変動 |
| 19 | サーバー管理機能を利用している場合は、これらを再設定する | 変動 | SQL Server 2008 からの新機能（SQL Server Audit やリソース ガバナー、ポリシー ベースの管理、パフォーマンス データ コレクションなど）など、移行元で利用しているものがある場合は、それらを再設定する。<br>利用している機能によって、設定時間が変動する |
| 20 | サーバー間の連携機能を利用している場合は、これらを再設定する | 変動 | レプリケーションやログ配布、データベース ミラーリング、可用性グループなど、移行元で利用しているものがある場合は、それらを再設定する。<br>利用している機能によって、設定時間が変動する |
| 21 | 動作チェックを行う | 1時間～ | アプリケーションやストアド プロシージャが問題なく動作するかをチェックする。当日のサービス インまでに確認しておくべき最重要クエリの動作チェックを行う |

この手順の中で、移行時間を大きく左右するのは、**データベースの移行**にかかる時間です。これは、**データベース サイズ**が大きい場合には、非常に時間がかかることになります。また、**ストレージの**

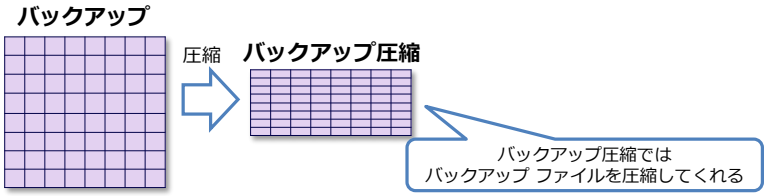
**性能**（読み取りおよび書き込み速度）と**ネットワーク速度**にも大きな影響を受けます。

## データベースの移行にかかる時間の見積もり

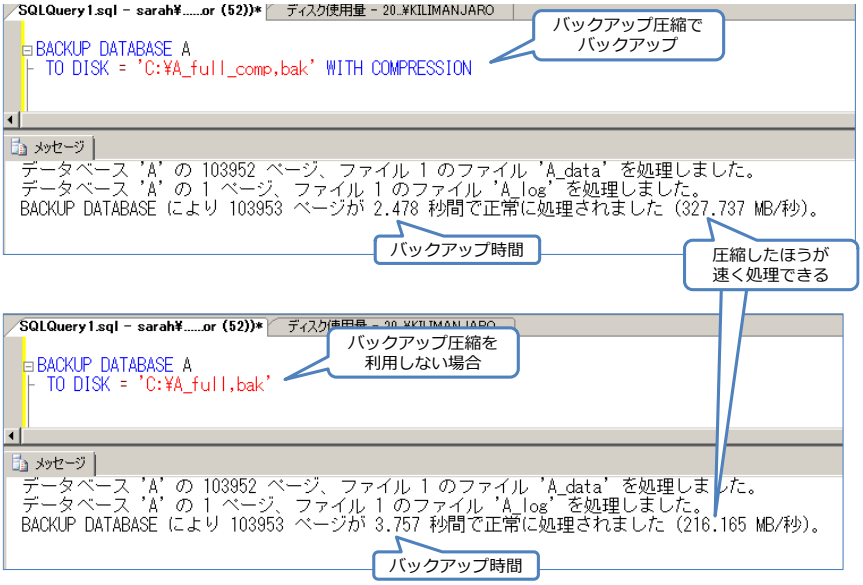
データベースの移行にかかる時間は、次のとおりです。

**データベースの移行にかかる時間**
**＝ バックアップ（BACKUP ステートメントの実行）にかかる時間**
**＋ バックアップ ファイルをコピーするのにかかる時間**
**＋ バックアップ ファイルの復元（RESTORE ステートメントの実行）にかかる時間**

第４章の 4.11 で説明したように、バックアップ ファイルを圧縮するかどうかは、低速回線かどうかにもよりますが、低速回線の場合は、圧縮したほうが速くコピーが完了できる場合が多くなります。SQL Server 2008 以降であれば、**バックアップ圧縮**機能を利用して、（バックアップ時に）バックアップ ファイルを圧縮することができます。

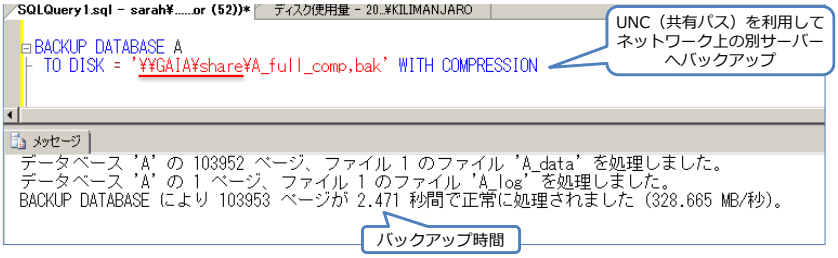



バックアップ圧縮機能は、SQL Server 2008 Enterprise または SQL Server 2008 R2 Standard エディション以上であれば利用することができます（SQL Server 2008 のときは Enterprise エディションが必要でしたが、SQL Server 2008 R2 以降では Standard エディションでも利用できるようになりました）。バックアップ圧縮は、次のように **WITH COMPRESSION** キーワードを付けるだけで実行できます。



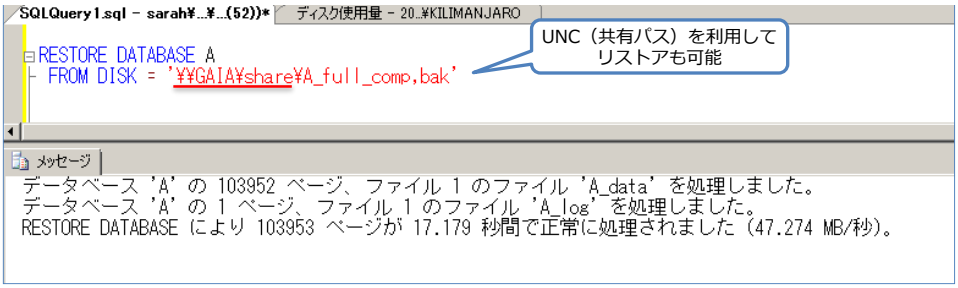

このように、バックアップ圧縮を利用すると、バックアップ時間を短縮できる可能性があるだけでなく、ファイル サイズを小さくできることによって、ファイルをコピーする時間の短縮も実現することができます。これは移行時のダウンタイムを小さくするという意味で、非常に重要です。

**BACKUP** ステートメントでは、次のように **UNC**（共有フォルダー）を指定して、ネットワーク上のサーバーへ直接バックアップすることも可能です。




これを利用すれば、バックアップをしつつ、ネットワーク コピーも行ってしまうということができます。LAN 環境などのネットワークが安定している場合には、お勧めのバックアップ方法です。一方、WAN 環境などのネットワークが安定していない場合には、万が一ネットワークが切れてしまった場合にバックアップをイチからやり直さなければならなくなるので、お勧めではありません。

オンライン バックアップからの復元時（**RESTORE DATABASE** ステートメント）も、次のように **UNC** を指定して実行することができます。




このように UNC を直接利用することでも、実行時間を短くできる場合があるので、検討してみることをお勧めします（ネットワークを介す場合は**圧縮**も重要になります）。

データベースの移行は、移行時のダウンタイムでの大きな時間を占めることになるので、事前に入念なテストを行って、しっかりと実行時間を見積もっておくことが重要になります。

なお、バックアップ（**BACKUP** ステートメント）に関しては、日々の定期バックアップにかかる時間を測定している場合は、そこからバックアップにかかる時間を見積もることができます。また、完全バックアップとログ バックアップを定期的に取得している環境の場合は、データベースの複製時には、最後のログ バックアップ以降の（未取得の）ログ バックアップのみを取得するだけで、最新の状態へ複製していくことができます。

## 5.34 Reporting Services の新規サーバーへの移行

**Reporting Services** を移行する手順は、ケース２ の場合とほとんど同じで、次のように行います（移行元は、SQL Server 2005／2008／2008 R2／2012／2014 のすべてで共通で、SQL Server 2005 の Reporting Services でも SQL Server 2016 へ移行することができます）。

1. 移行元で、**ReportServer** と **ReportServerTempDB** データベースをオンライン バックアップする（クエリ エディターから BACKUP DATABASE ステートメントを実行）
2. 移行元で、**暗号化キー**をバックアップする（Reporting Services 構成マネージャーから実行）
3. 移行先で、**Reporting Services をインストール**するときに、「**インストールと構成**」ではなく、「**インストールのみ**」を選択する
4. 移行先で、**データベースを復元**する（RESTORE DATABASE）
5. 移行先で、**暗号化キーを復元**する
6. 移行先で、**Reporting Services を構成**する（Reporting Services 構成マネージャーを利用）
7. 移行先で、**電子メールの設定**を行う（移行元で設定している場合）

### 1. ReportServer、ReportServerTempDB データベースのバックアップ

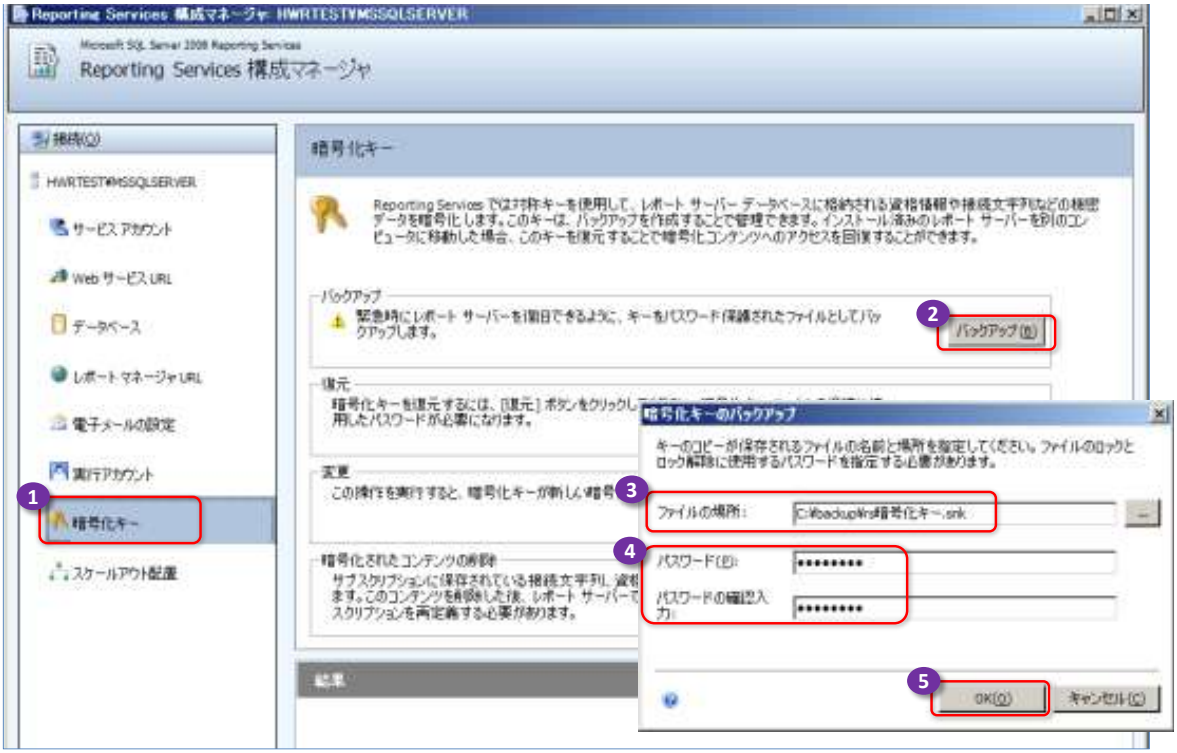
まずは、移行元で、**Reporting Services の実体**（レポート本体や、レポートに関する各種の設定）となる、SQL Server 上の **ReportServer** および **ReportServerTempDB** データベースをオンライン バックアップします。次のように、Management Studio のクエリ エディターで、**BACKUP DATABASE** ステートメントを実行します（ファイル名やバックアップ先のパスは皆さんの環境に合わせて変更してください）。

```
BACKUP DATABASE ReportServer
 TO DISK = 'C:¥backup¥ReportServer_full.bak'

BACKUP DATABASE ReportServerTempDB
 TO DISK = 'C:¥backup¥ReportServerTempDB_full.bak'
```

## 2. 移行元で暗号化キーをバックアップする

次に、移行元で、**暗号化キー**をバックアップします。暗号化キーのバックアップは、「**Reporting Services 構成マネージャー**」ツールを利用して、次のように［**暗号化キー**］ページから行います（画面は、SQL Server 2008 ですが、他のバージョンでもほとんど同じ操作で行えます）。



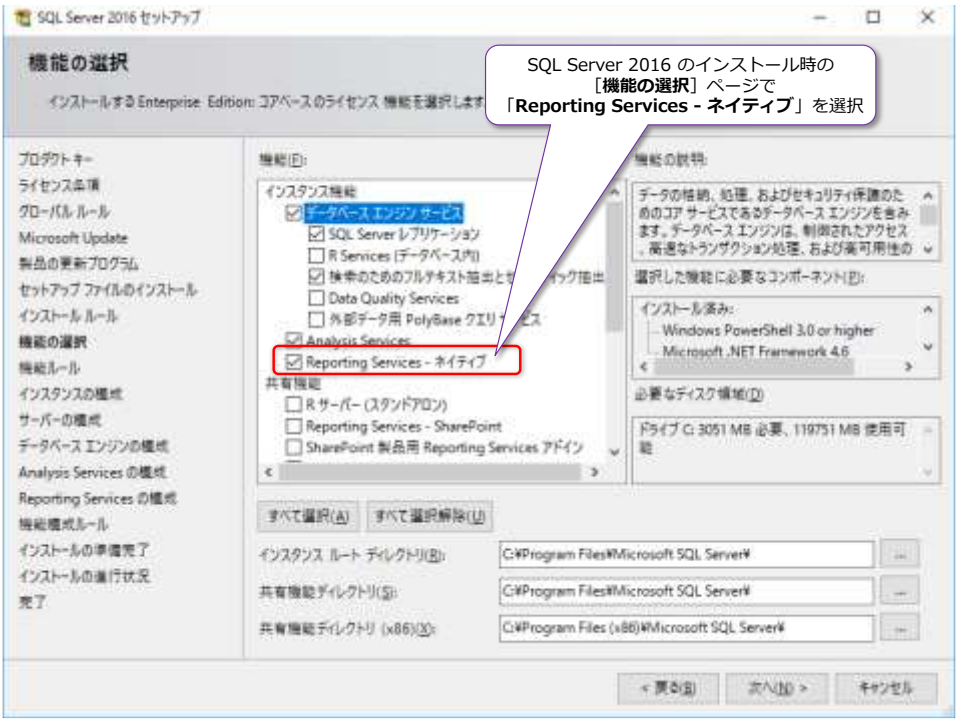

このページでは、［**バックアップ**］をクリックして、［**暗号化キーのバックアップ**］ダイアログが表示されたら、［**パスワード**］と［**パスワードの確認**］に任意のパスワードを設定し、［**ファイルの場所**］にバックアップ先のファイル名を指定します（画面は **C:¥backup** フォルダーの下に **rs 暗号化キー.snk** というファイル名を指定）。ここで設定するパスワードは、**複雑なパスワード**（大文字・小文字と @などの特殊文字を入れたりするなど）を指定する必要があり、これを行っていない場合は、次のように失敗します。


バックアップが成功した場合は、次のように表示されます。

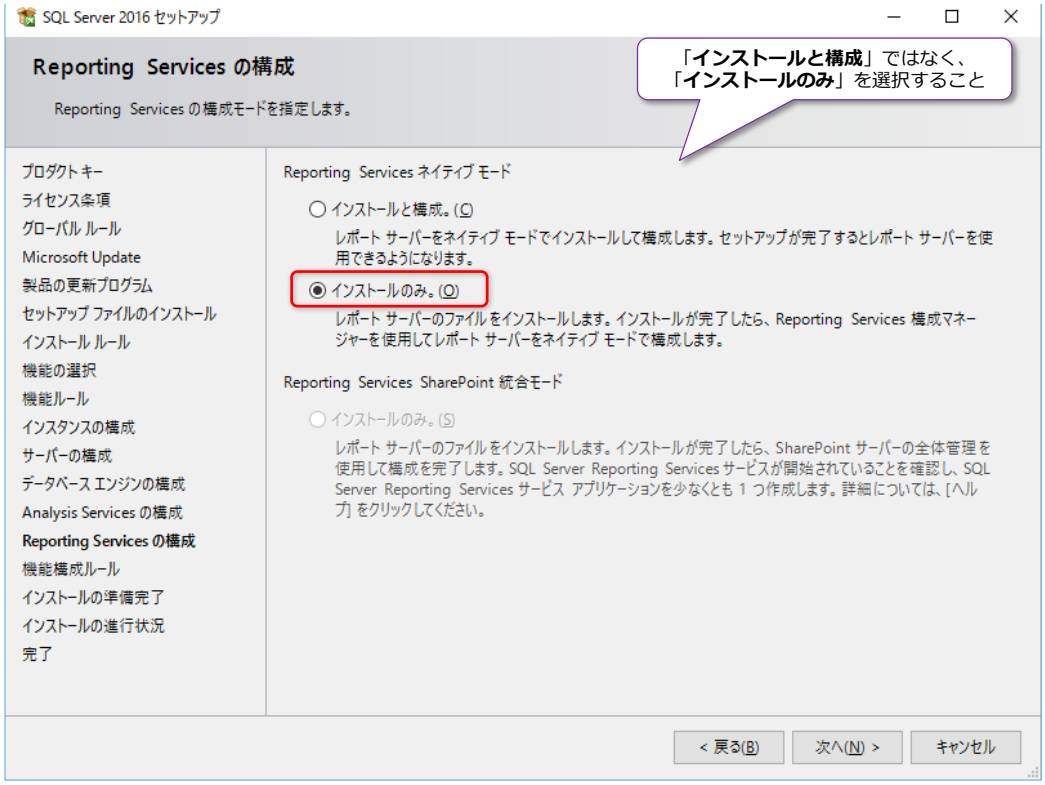

また、設定したパスワードは、新規サーバーで復元する際に必要になるので覚えておいてください。バックアップしたファイル（**～.snk**）は、新規サーバーからアクセス可能なファイル サーバーや USB HDD など持ち運び可能なメディアに保存します。

## 3. 移行先で Reporting Services をインストール。「インストールのみ」を選択

次に、移行先で Reporting Services をインストールします。



インストール時の［**Reporting Services の構成**］ページでは、次のように「**インストールのみ**」を選択するようにします。

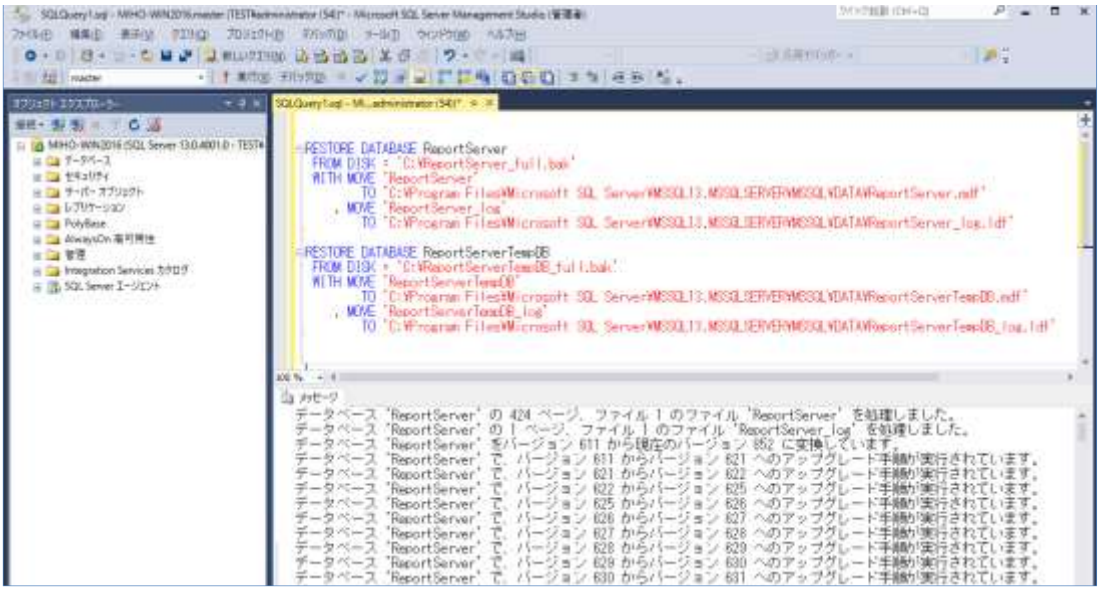



## 4. 移行先で、データベースを復元する

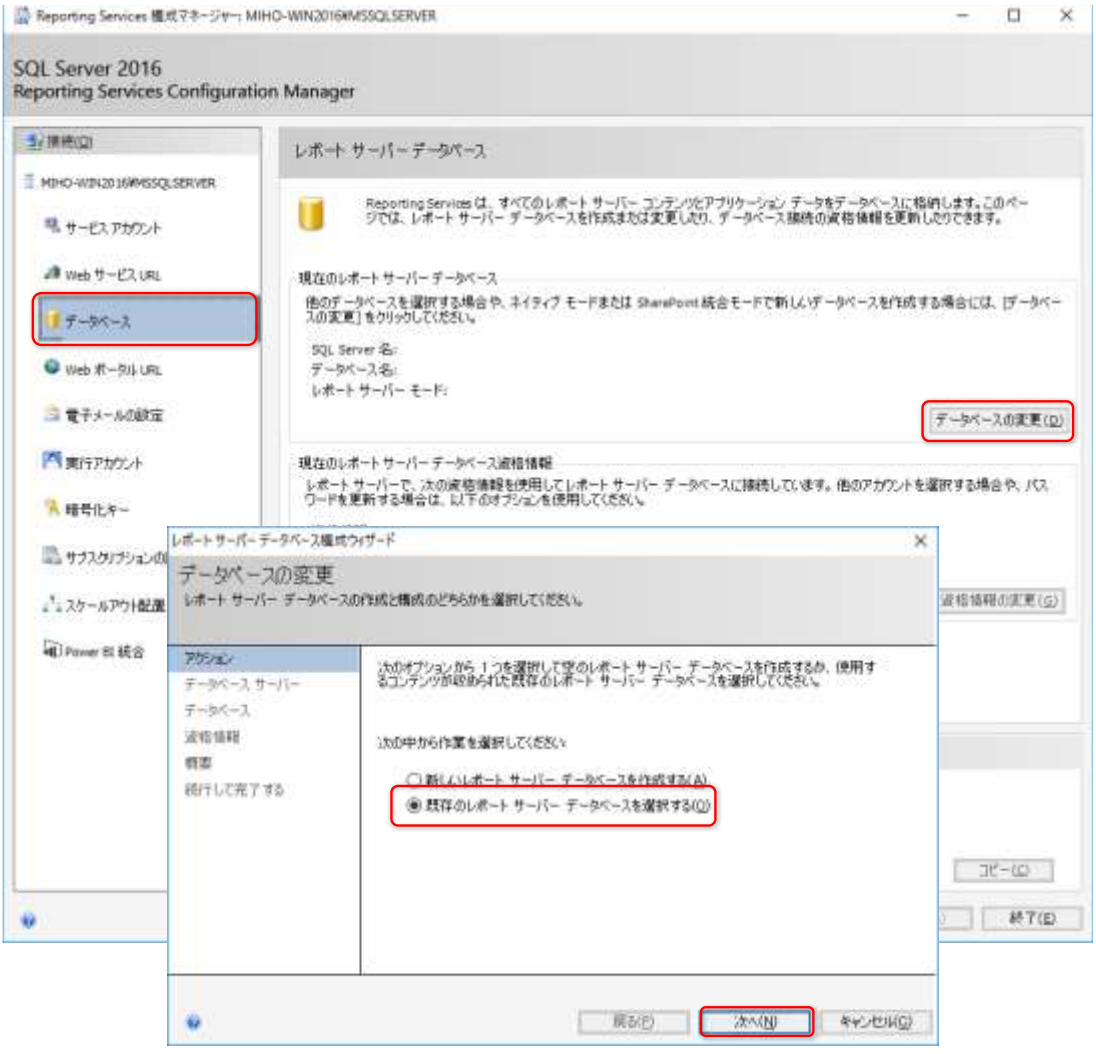
移行先での Reporting Services のインストールが完了したら、次に、**ReportServer** および **ReportServerTempDB** データベースをバックアップから復元します。復元には、**RESTORE DATABASE** ステートメントを次のように実行します（復元先のパスには、SQL Server 2016 の Reporting Services の既定のインスタンスが、既定でインストールされるフォルダーを指定していますが、任意の場所に変更することも可能です）。

```
RESTORE DATABASE ReportServer
 FROM DISK = 'C:¥ReportServer_full.bak'
 WITH MOVE 'ReportServer'
        TO 'C:¥Program Files¥Microsoft SQL Server¥MSSQL13.MSSQLSERVER¥MSSQL¥DATA¥ReportServer.mdf'
    ,MOVE 'ReportServer_log'
        TO 'C:¥Program Files¥Microsoft SQL Server¥MSSQL13.MSSQLSERVER¥MSSQL¥DATA¥ReportServer_log.ldf'

RESTORE DATABASE ReportServerTempDB
 FROM DISK = 'C:¥ReportServerTempDB_full.bak'
 WITH MOVE 'ReportServerTempDB'
        TO 'C:¥Program Files¥Microsoft SQL Server¥MSSQL13.MSSQLSERVER¥MSSQL¥DATA¥ReportServerTempDB.mdf'
    ,MOVE 'ReportServerTempDB_log'
        TO 'C:¥Program Files¥Microsoft SQL Server¥MSSQL13.MSSQLSERVER¥MSSQL¥DATA¥ReportServerTempDB_log.ldf'
```



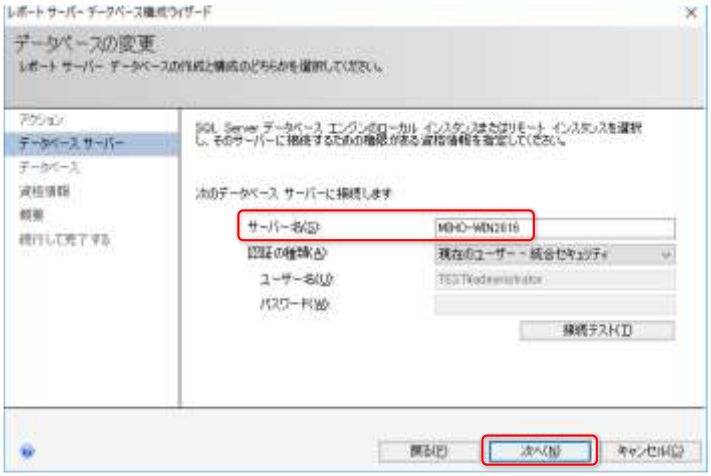
データベースの復元が完了した後は、**Reporting Services 構成マネージャー**を利用して、Reporting Services にデータベースを認識させます。これを行うには、次のように［**データベース**］ページで、［**データベースの変更**］をクリックします。



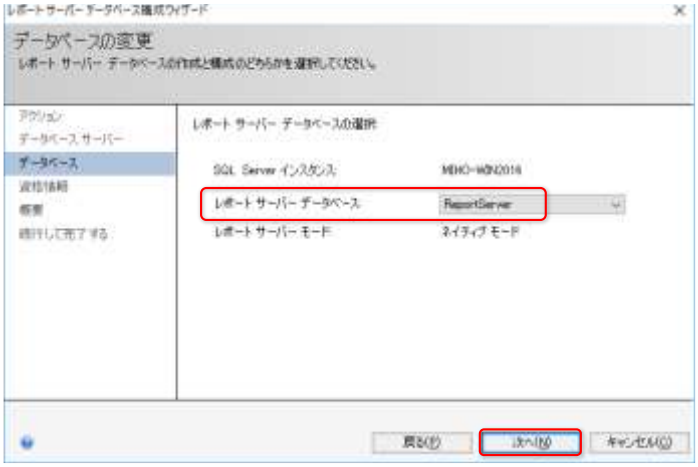

［**データベースの変更**］ウィザードが起動したら、［**既存のレポート サーバー データベースを選択**

**する**］を選択して、［**次へ**］ボタンをクリックします。

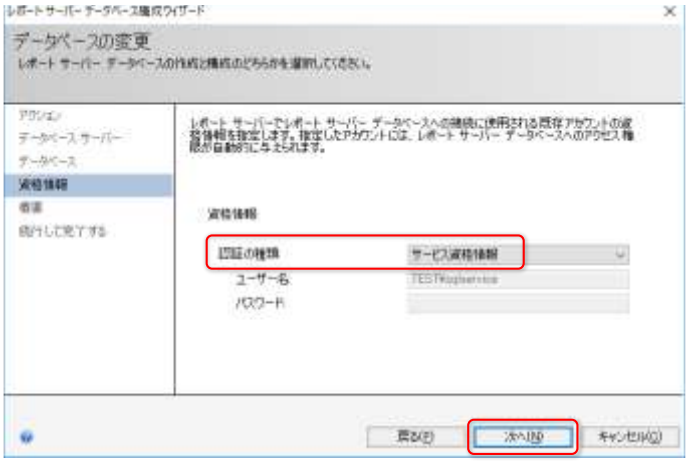
次のページでは、［**サーバー名**］に移行先の SQL Server 2016 のサーバー名を選択して、［**次へ**］ボタンをクリックします。



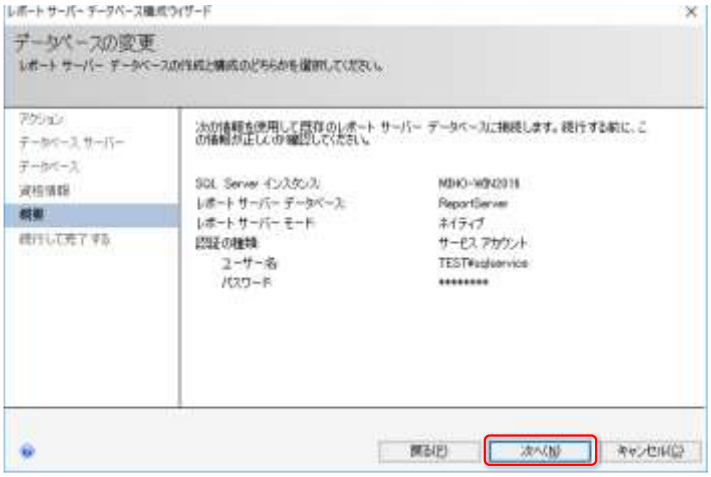

次のページでは、［**レポート サーバー データベース**］で「**ReportServer**」データベースを選択して、［**次へ**］ボタンをクリックします。



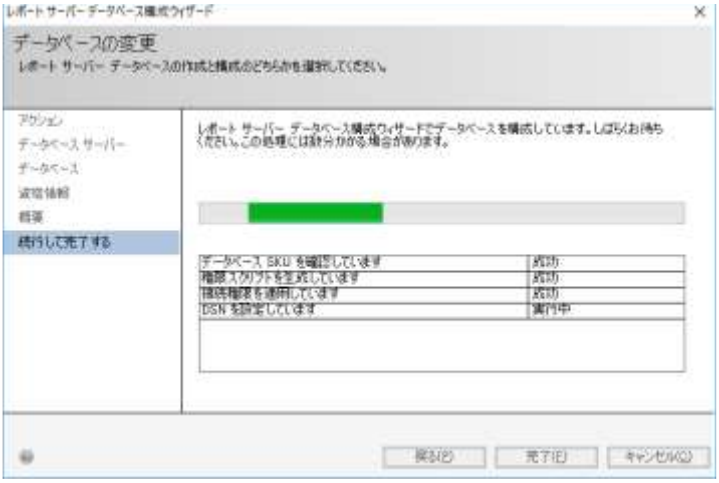

次のページでは、［**認証の種類**］でレポート サーバーがデータベースへアクセスするときの認証方法を選択して（既定は「**サービス資格情報**」で、サービス アカウントが接続しに行く設定）、［**次へ**］ボタンをクリックします。



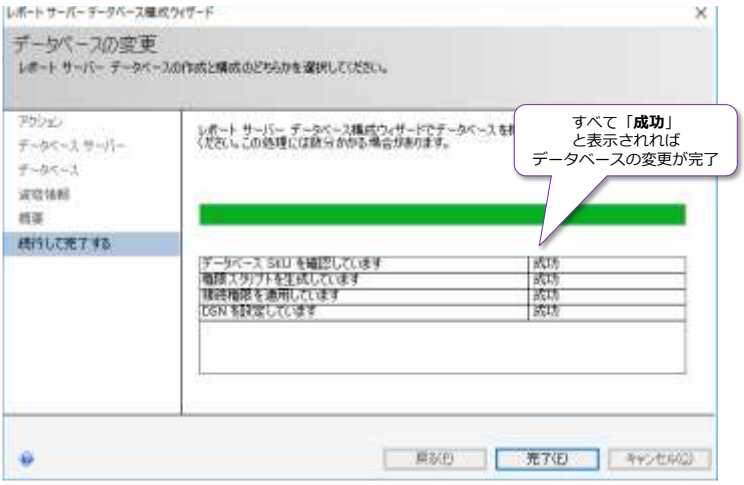
次のページでは、[**次へ**] ボタンをクリックすると、データベースの変更が開始されます。




データベースの変更中は、次のように表示されます。

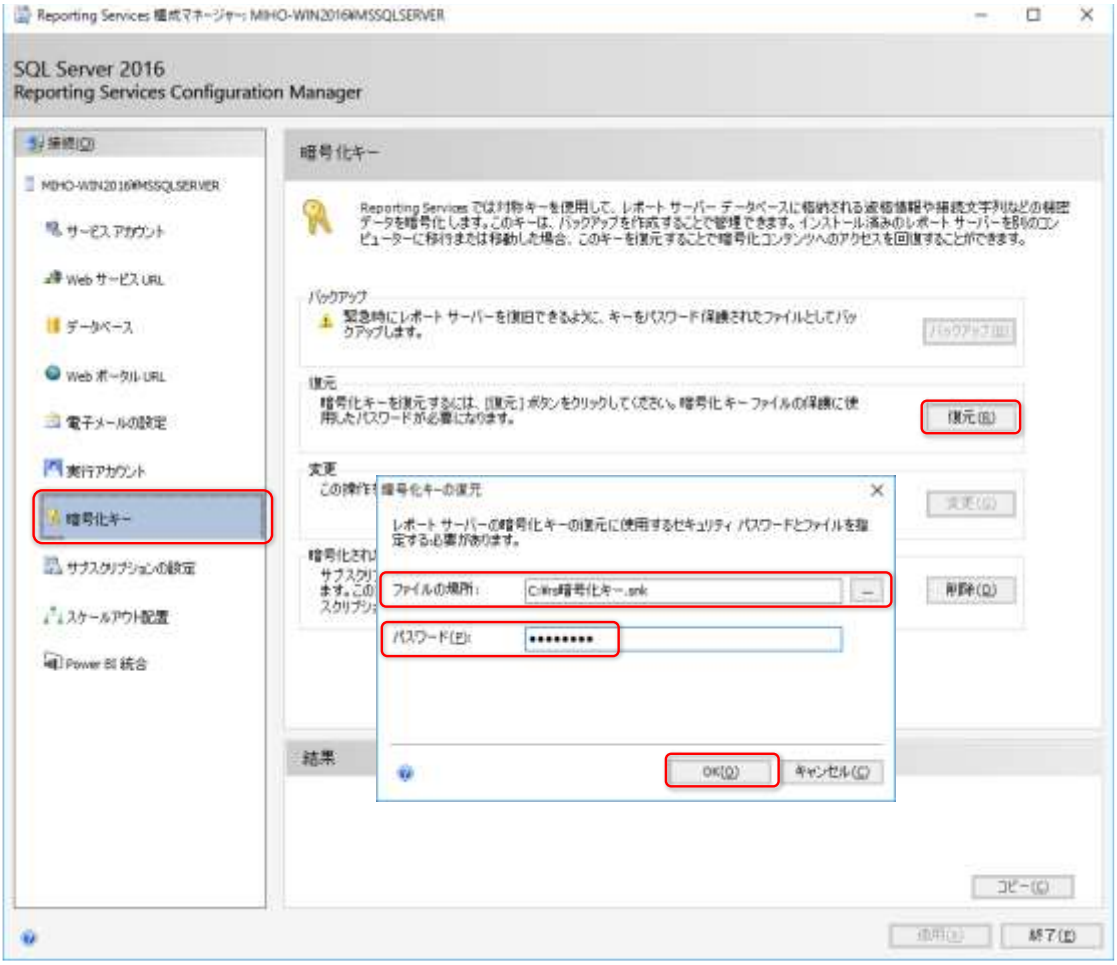



データベースの変更が完了すると、次のように表示されます。



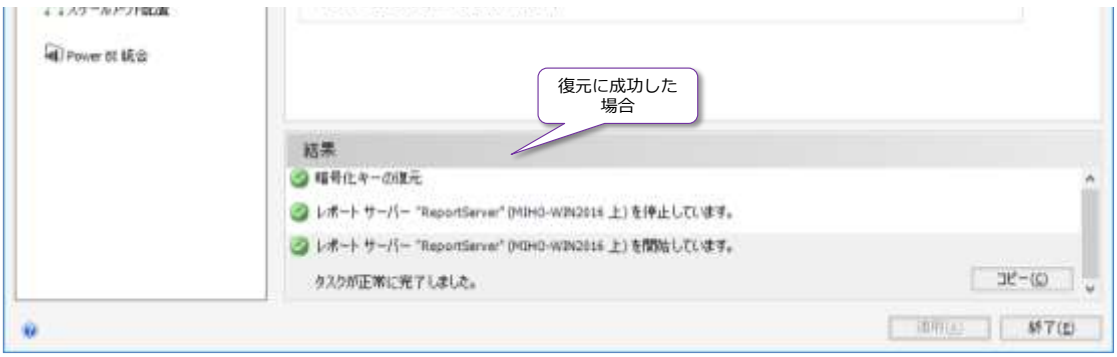
## 5. 移行先で、暗号化キーを復元する

次に、**暗号化キー**をバックアップから復元します。これは、次のように［**暗号化キー**］ページで［**復元**］ボタンをクリックします。。



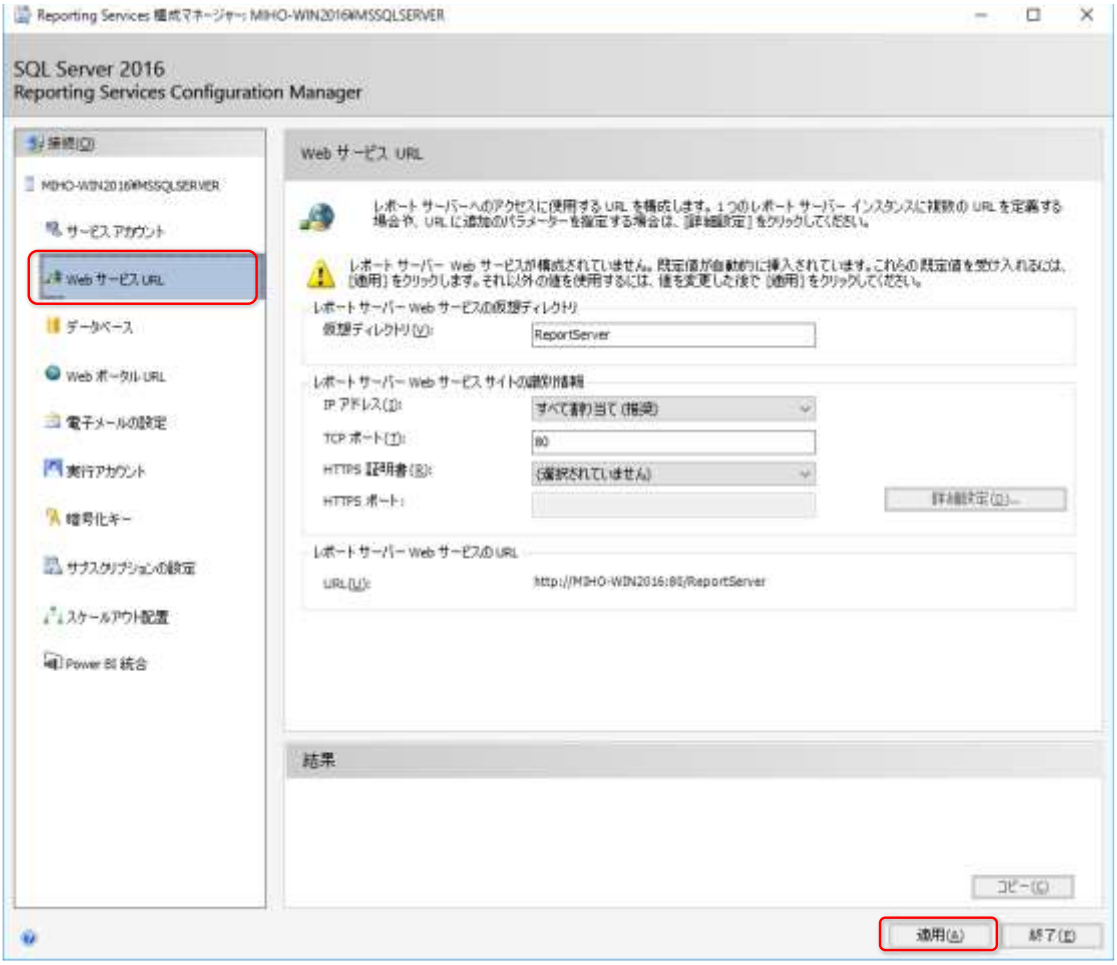

［**暗号化キーの復元**］ダイアログでは、［**パスワード**］にバックアップ時に指定したパスワードを入力して、［**キー ファイル**］にバックアップした暗号化キー（**.snk**）を指定します。

復元に成功すると、次のように表示されます。



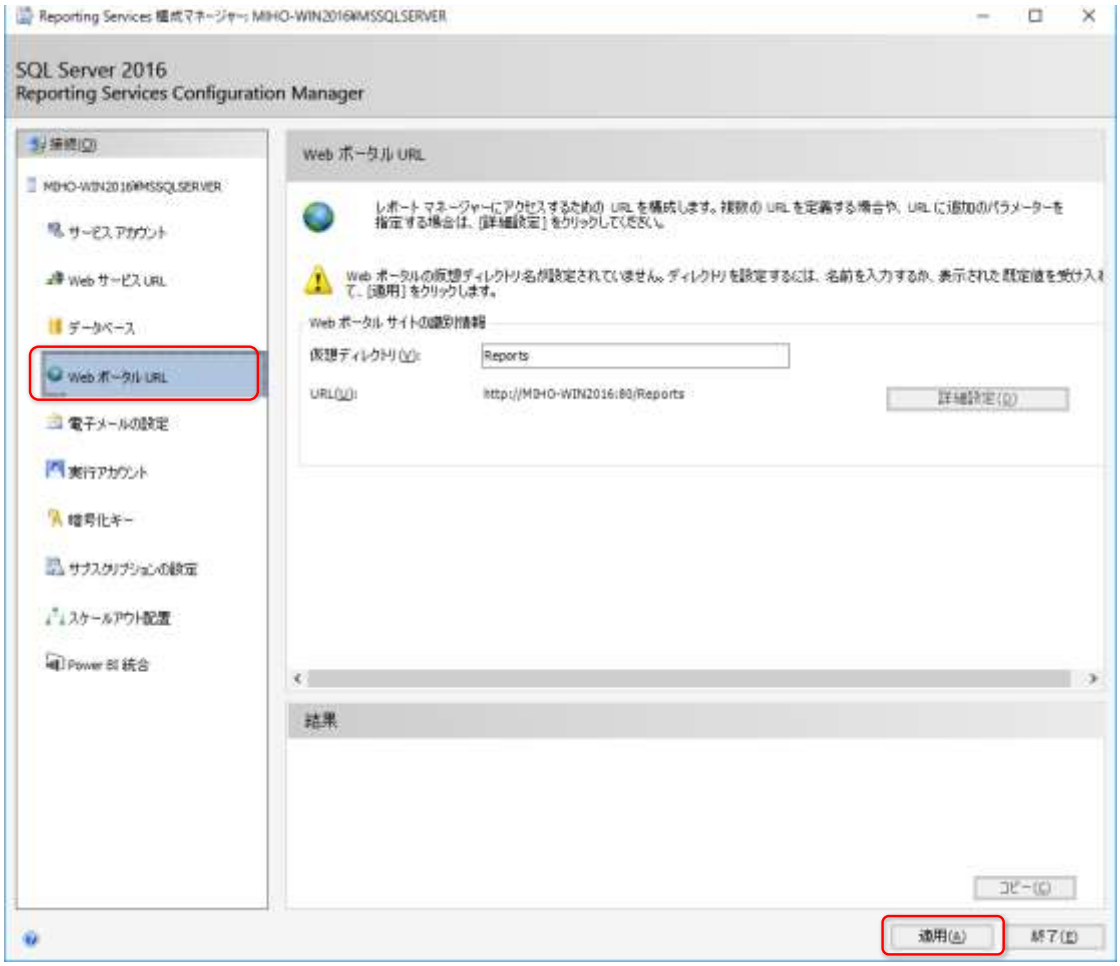
## 6. 移行先で、Reporting Services を構成する

暗号化キーの復元が完了したら、次は Reporting Services を構成します。引き続き Reporting Services 構成マネージャーを利用して、次のように［**Web サービス URL**］ページを開きます。




このページでは［**適用**］ボタンをクリックします。これで**レポート サーバー**（**http://～/ReportServer**）が構成されます。構成が成功した場合には、次のように表示されます。

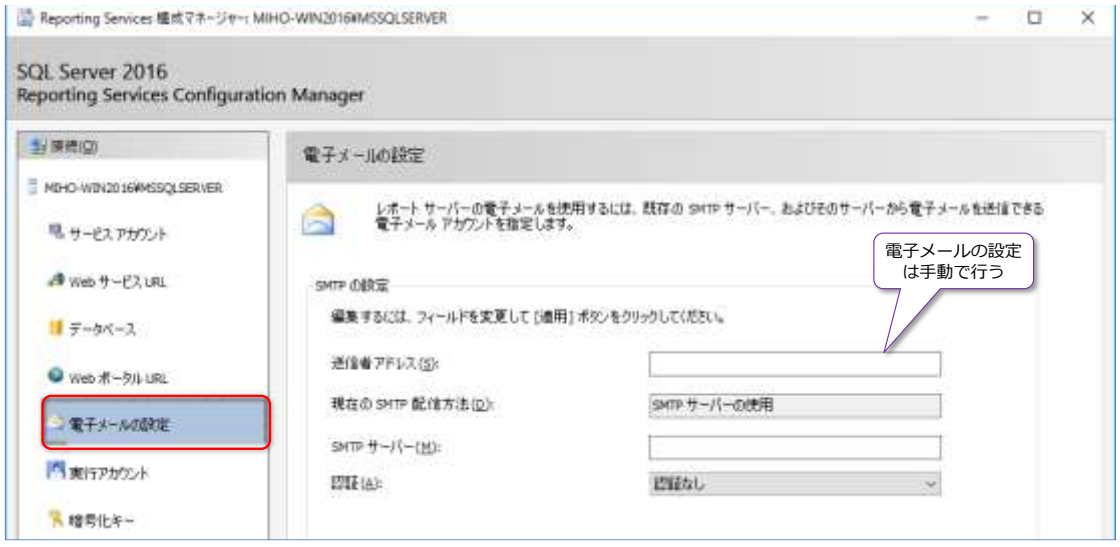
次に、[**Web ポータル URL**] ページを開いて、[**適用**] ボタンをクリックします。



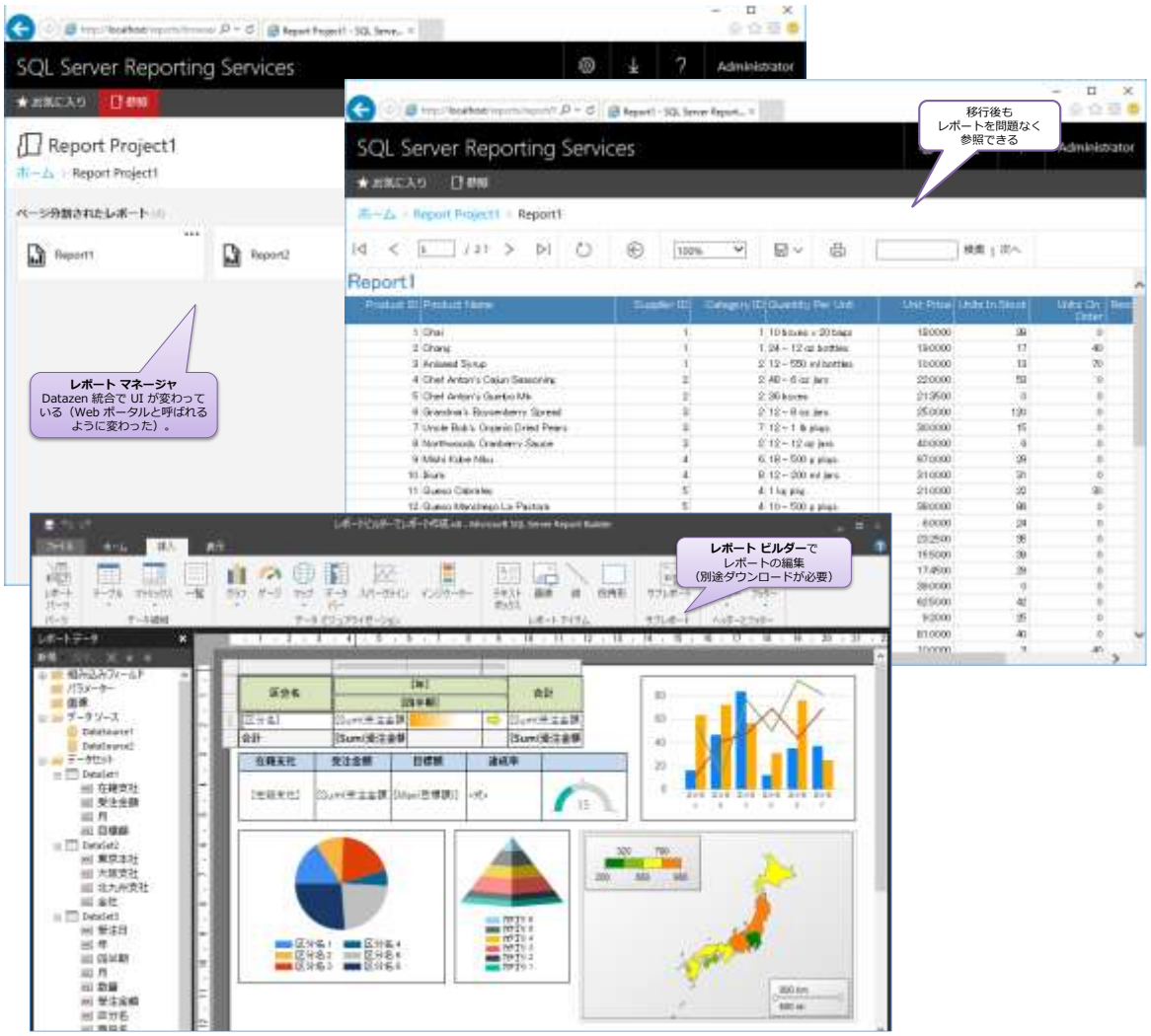

これで**レポート マネージャー**（**http://～/Reports**）が構成されます。

## 7. 電子メールの設定

Reporting Services の電子メールの設定に関しては、手動で再設定をする必要があります。



以上で、作業が完了です。これで移行元と同じようにレポートを参照できるようになります。




ただし、レポート内のデータソースや、共有データソースに、**移行元のマシン名**が埋め込まれている場合には、そこを**移行先のマシン名**に修正する必要があります（マシン名ではなく、**localhost** などを指定している場合は修正は不要です）。

レポート マネージャー（**/Reports**）のポータルは、SQL Server 2016 からは Datazen というソフトウェア（モバイル レポート機能）を統合したことによって、見栄えが変わっていますが、レポート サーバー（**/ReportServer**）に関しては、以前の SQL Server と同様に利用できます。

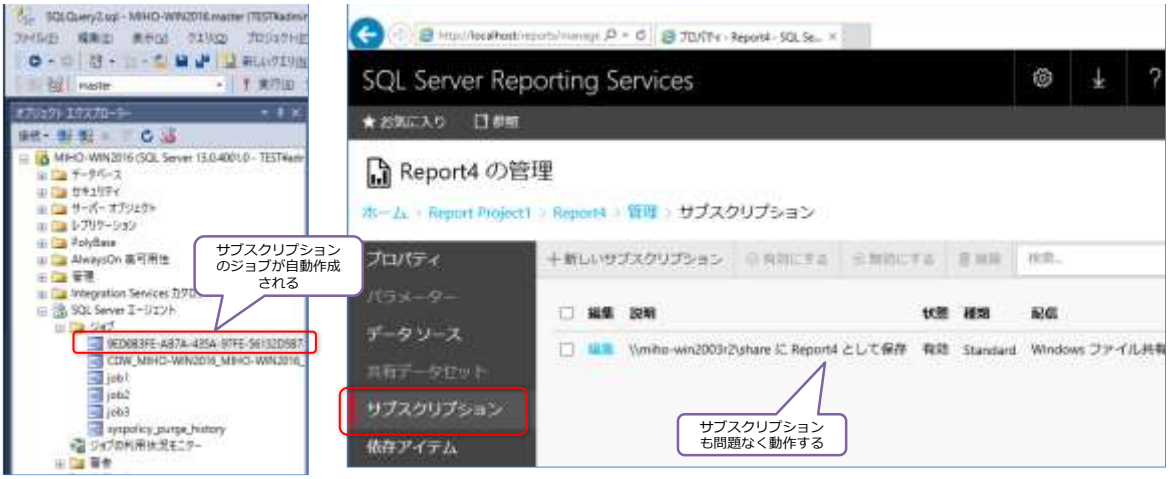
なお、SQL Server 2016 からは ClickOnce のレポート ビルダーの提供はなくなって、別途ダウンロード／インストールする形に変わりました。

レポート ビルダーのダウンロードは、次の URL から行うことができます。

Microsoft SQL Server 2016 レポート ビルダー
https://www.microsoft.com/ja-JP/download/details.aspx?id=53613

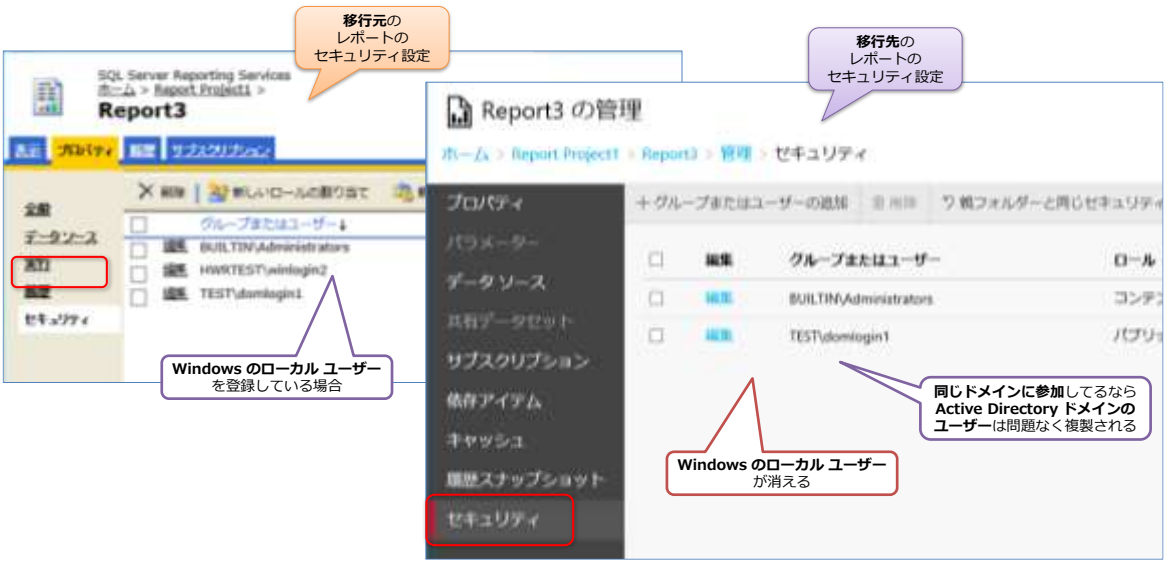
SQL Server 2016 からは、Datazen ソフトウェアとの統合によって、モバイル レポート機能が強化されていますが、これについては、SQL Server 2016 自習書シリーズの No.3「Reporting Services の新機能」編で詳しく説明しているので、こちらもぜひご覧いただければと思います。

構成後は、レポートを参照できるだけでなく、**レポートのプロパティ**で設定した**セキュリティ**（ロール）や、**キャッシュ設定**、**サブスクリプション**についても移行元と同じように利用できます。サブスクリプションに関しては、次のように SQL Server Agent のジョブも自動作成されています（Reporting Services を構成すると、ジョブも自動作成されます）。




## Windows のローカル ユーザーを利用している場合の注意点

移行後の注意点としては、**レポートのセキュリティ設定**で **Windows のローカル ユーザー**を利用している場合です。これは、次のような状況です。




Windows のローカル ユーザーの場合は、マシン（OS）が変わったとすると、同じ名前のユーザーを作成したとしても、**内部的な SID（Security ID）**が異なってしまうので、Reporting Services にとっては、認識できないユーザーとなってしまいます。このため、新規サーバーでは、Windows のローカル ユーザーが全く表示されない形になります。

これを回避する方法はないので、Windows のローカル ユーザーを利用している場合は、再設定をする必要があります。

**Active Directory ドメインのユーザー**に関しては、**移行元と移行先が同じドメインに参加**している場合には、問題なくセキュリティ設定が引き継がれますが、**移行先が異なるドメイン**や、**ワークグループ環境**の場合には、ドメイン ユーザーに関しても、認識できないユーザーとなって、表示されない形になります。この場合も再設定が必要になります。

## サブスクリプションの所有者が Windows のローカル ユーザーの場合

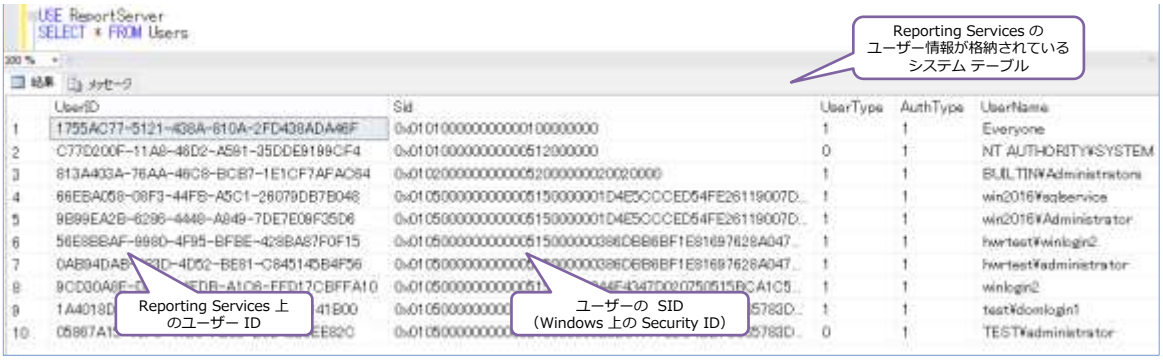
サブスクリプションの所有者が **Windows のローカル ユーザー**の場合や、**Active Directory ドメインのユーザー**で、**移行先が異なるドメイン**や、**ワークグループ環境**の場合には、サブスクリプションが"**編集**"できないということが発生します。これは次のような状況です。


サブスクリプションを編集しようとすると、古い所有者（移行元でのサブスクリプションの所有者）が存在しないので、「**認識できません**」というエラーとなってしまいます。

これを回避するには、サブスクリプションを削除して、再作成をするか、**Subscriptions** テーブル（Reporting Services が利用しているシステム テーブル）を修正するようにします。**Subscriptions** テーブルは、**ReportServer** データベース内に存在します。


このテーブルには、**OwnerID** 列があり、ここにサブスクリプションの所有者の ID が格納されて

います。この ID は、Reporting Services 上の ID で、**Users** テーブルで管理されています。




**Users** テーブルでは、**UserID** 列が Reporting Services 上の ID（Reporting Services で管理するユーザーを識別するための ID）、**Sid** 列に Windows 上の **SID**（Security ID）が格納されています。この **Users** テーブルにユーザーを追加するには、次のように Reporting Services の Web ポータル（レポート マネージャー）での［**サイトの設定**］ページや、各レポートの［**セキュリティ**］ページから行えます。


このように移行先で登録した新しいユーザーを利用して、サブスクリプションの所有者を変更するようにすれば、サブスクリプションを編集できない問題を解決できます。これを行うには、次のように **UPDATE** ステートメントを実行します。

```
USE ReportServer
DECLARE @OldUserID uniqueidentifier
DECLARE @NewUserID uniqueidentifier
SELECT @OldUserID = UserID FROM Users WHERE UserName = '古い所有者名'
SELECT @NewUserID = UserID FROM Users WHERE UserName = '新しい所有者名'
UPDATE Subscriptions SET OwnerID = @NewUserID WHERE OwnerID = @OldUserID
```

**Subscriptions** テーブルの **OwnerID** 列を、新しい所有者の **UserID** に変更しています。これを行えば、移行先でもサブスクリプションを編集できるようになります。

以上のように、Reporting Services の移行は、非常に簡単です。SQL Server 2005 からの移行も可能で、SQL Server 2005 のときには、Reporting Services の動作に IIS と ASP.NET が必須でしたが、SQL Server 2008 以降では IIS を必要としないアーキテクチャに変わっています（SQL Server 2016 に移行後は、IIS が不要になります）。

## Reporting Services の移行に関するその他の情報

その他の Reporting Services の移行に関する情報は、オンライン ブックの以下のトピックが参考になると思います。

Reporting Services の旧バージョンとの互換性
http://msdn.microsoft.com/ja-jp/library/ms143251.aspx


SQL Server 2016 で廃止された SQL Server Reporting Services の機能
http://msdn.microsoft.com/ja-jp/library/ms144231.aspx

SQL Server 2016 における SQL Server Reporting Services の非推奨機能
http://msdn.microsoft.com/ja-jp/library/ms143509.aspx

SQL Server 2016 における SQL Server Reporting Services の重大な変更
http://msdn.microsoft.com/ja-jp/library/ms143380.aspx

SQL Server 2016 における SQL Server Reporting Services の動作変更
http://msdn.microsoft.com/ja-jp/library/ms143200.aspx

## 5.35 Analysis Services の新規サーバーへの移行

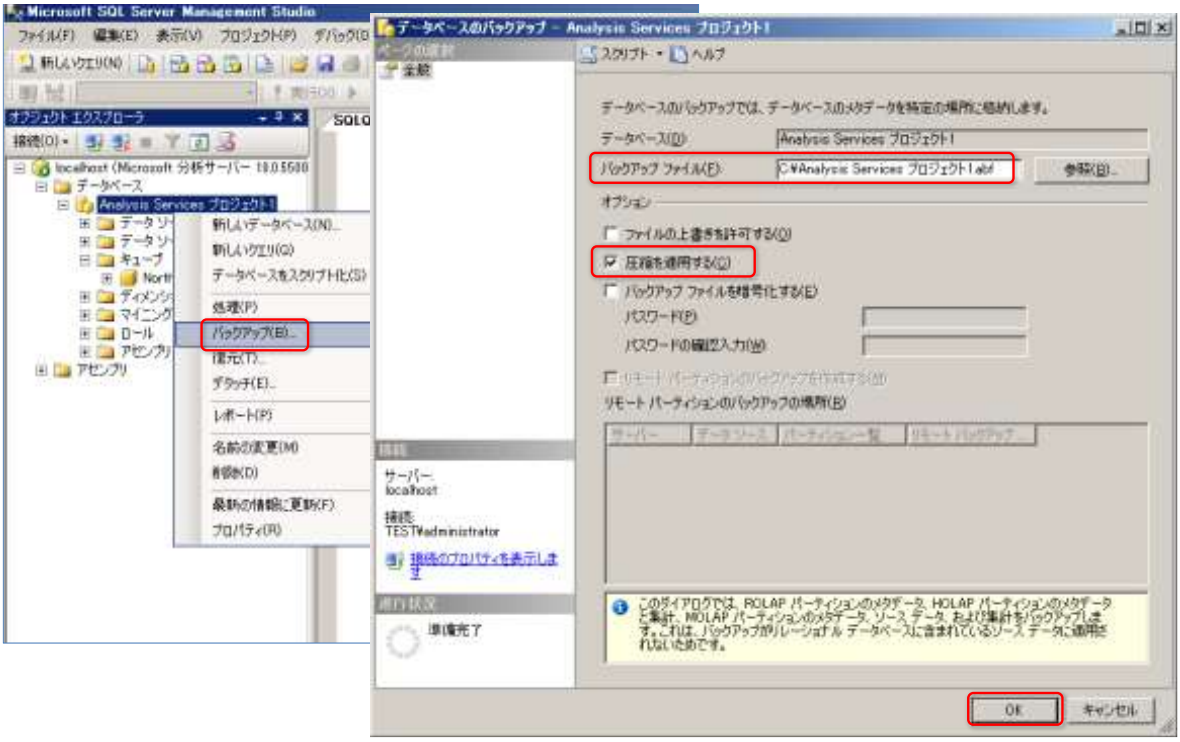
**Analysis Services** を移行する手順は、ケース２ の場合とほとんど同じで、次のように行います（移行元は、SQL Server 2005／2008／2008 R2／2012／2014 のすべてで共通で、SQL Server 2005 の Analysis Services でも SQL Server 2016 に移行することができます）。

1. 移行元で、**Analysis Services のデータベース**を**オンライン バックアップ**する
2. 移行先で、**オンライン バックアップ**から復元する
3. **Analysis Services の構成オプション**を変更している場合は、再設定する

なお、SQL Server 2012 からは、インメモリ BI を実現できる「**表形式モード**」（**Tabular Mode**）も登場していますが、ここでは従来ながらの「**多次元モード**」（Multi Dimensional Mode、OLAP キューブ）での移行方法を説明します。

### 移行元でデータベースのオンライン バックアップ

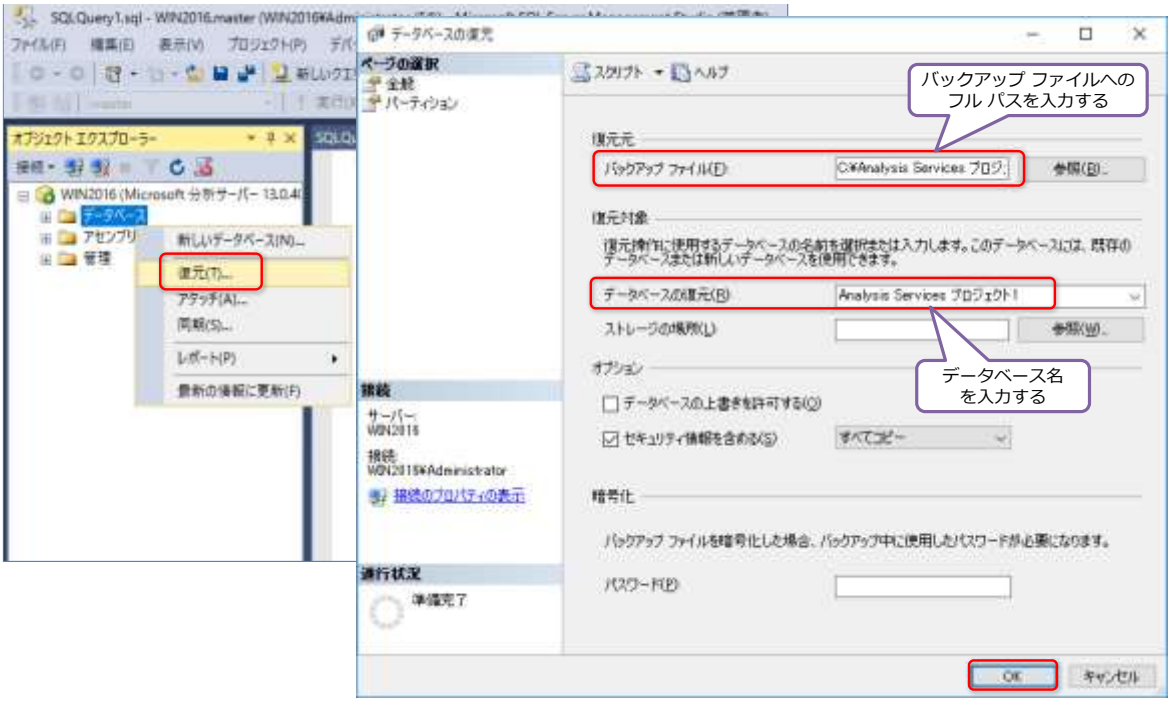
まずは、移行元で、**Analysis Services** でデータベースの**オンライン バックアップ**を実行します。オンライン バックアップは、次のように Management Studio で Analysis Services に接続して行います（画面は SQL Server 2008 ですが、他のバージョンでもほとんど同じ操作で行えます）。




データベースを右クリックして、[**バックアップ**] をクリックし、[**バックアップ ファイル**] へバックアップ先のファイル名をフル パスで入力します。[**圧縮を利用する**] はチェックを付けておくことをお勧めします（バックアップ ファイルを圧縮することができます）。

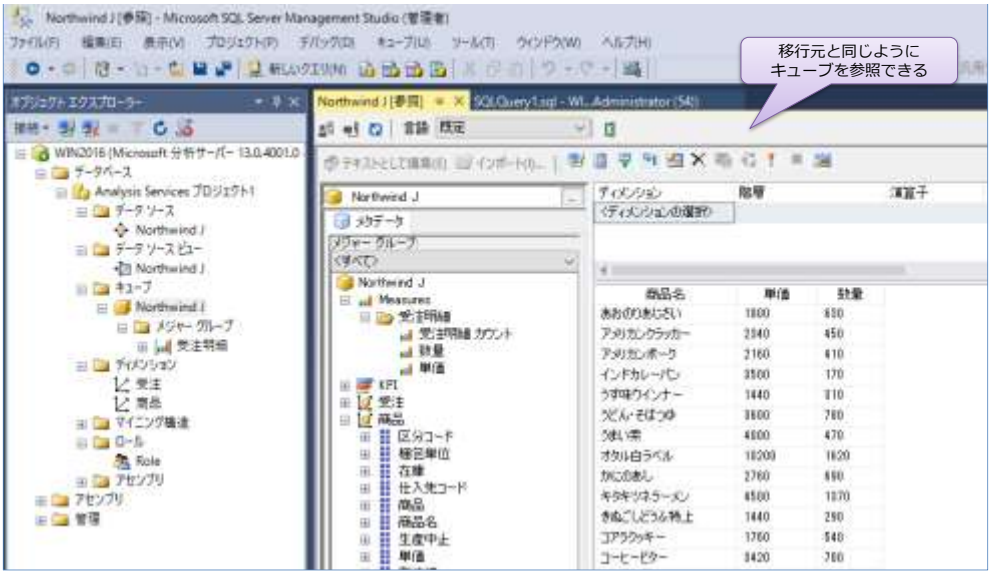
## 移行先でオンライン バックアップから復元

移行元で取得したオンライン バックアップは、移行先（SQL Server 2016）で次のように復元できます。




Management Studio で Analysis Services に接続して、[**データベース**] フォルダーを右クリックして、[**復元**] をクリックします。[**データベースの復元**] ダイアログでは、[**バックアップ ファイル**] へバックアップ ファイルをフル パスで入力します。[**データベースの復元**] には復元後のデータベース名を入力して、[**OK**] ボタンをクリックします。

復元が完了すると、旧マスターと同じようにデータベースを参照できるようになります。

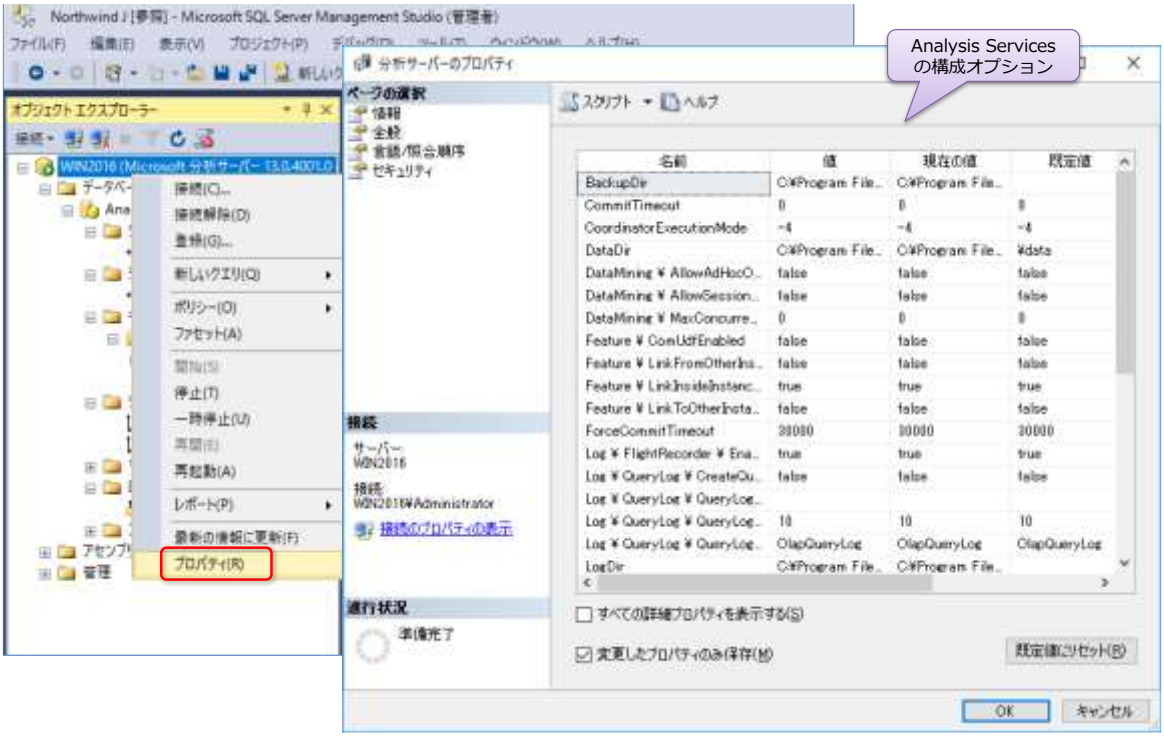



ただし、データソースに、**移行元のマシン名**が埋め込まれている場合には、そこを**移行先のマシン**

**名**に修正する必要があります（マシン名ではなく、**localhost** などを指定している場合は修正は不要です）。

## Analysis Services の構成オプション

移行元で、**Analysis Services の構成オプション**を変更している場合は、移行先でも再設定をする必要があります。




なお、これらの設定は、「**msmdsrv.ini**」という設定ファイルに格納されていて、SQL Server 2008 の場合は、既定で次のフォルダーに作成されています。

**C:¥Program Files¥Microsoft SQL Server¥MSAS10.MSSQLSERVER¥OLAP¥Config**

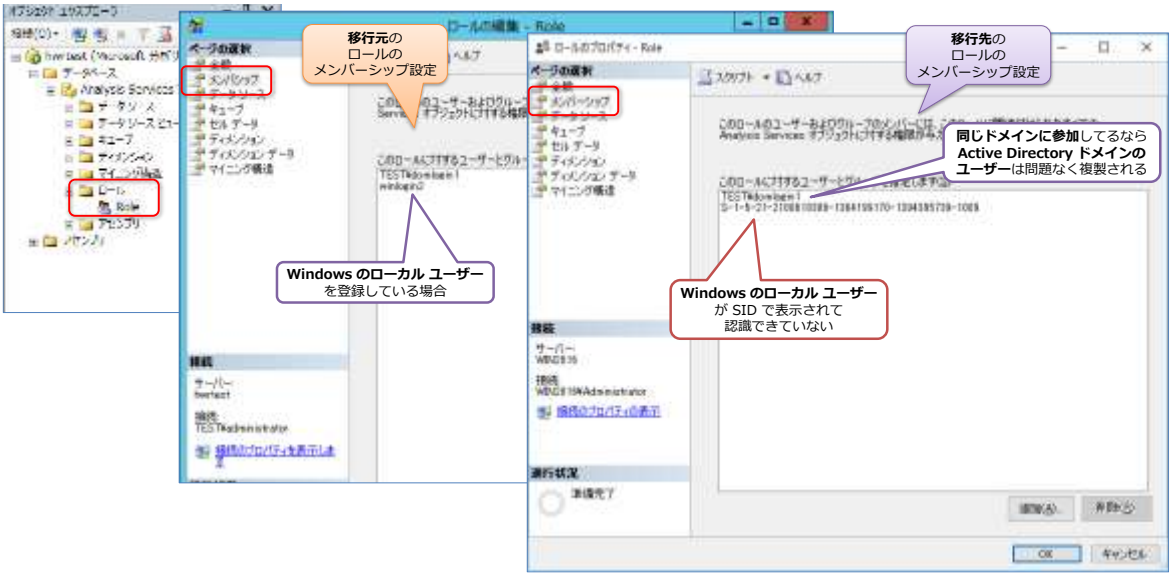
SQL Server 2008 R2 の場合は「**MSAS10.MSSQLSERVER**」の部分が「**MSAS10_50.～**」、SQL Server 2012 では「**MSAS11.～**」、SQL Server 2014 では「**MSAS12.～**」、SQL Server 2005 の場合は「**MSSQL.2**」に代わります。

多くの構成オプションを変更している場合は、このファイルをもとに修正すると、作業が楽になると思います。

以上で、作業が完了です。これで移行元と同じように Analysis Services を利用できるようになります。

## Windows のローカル ユーザーを利用している場合の注意点

移行後の注意点としては、**Analysis Services のセキュリティ設定（ロール）**で **Windows のローカル ユーザー**を利用している場合です。これは、次のような状況です。




Windows のローカル ユーザーの場合は、マシン（OS）が変わったとすると、同じ名前のユーザーを作成しても、**内部的な SID（Security ID）**が異なってしまうので、Analysis Services にとっては、認識できないユーザーとなってしまいます。このため、新規サーバーでは、Windows のローカル ユーザーの名前が不明になるので、SID で表示される形になります。

これを回避する方法はないので、Windows のローカル ユーザーを利用している場合は、再設定をする必要があります。

**Active Directory ドメインのユーザー**に関しては、**移行元と移行先が同じドメインに参加**している場合には、問題なくセキュリティ設定が引き継がれますが、**移行先が異なるドメイン**や、**ワークグループ環境**の場合には、ドメイン ユーザーに関しても、認識できないユーザーとなって、SID で表示されてしまいます。この場合も再設定が必要になります。

以上のように、Analysis Services の移行は、非常に簡単です。SQL Server 2005 からの移行も可能です。

## Analysis Services の移行に関するその他の情報

その他の Analysis Services の移行に関する情報は、オンライン ブックの以下のトピックが参考になると思います。

Analysis Services の旧バージョンとの互換性
http://msdn.microsoft.com/ja-jp/library/ms143479.aspx


SQL Server 2016 で提供が中止された Analysis Services の機能
http://msdn.microsoft.com/ja-jp/library/ms143229.aspx

SQL Server 2016 における Analysis Services 非推奨機能
http://msdn.microsoft.com/ja-jp/library/ms143346.aspx

SQL Server 2016 における Analysis Services 機能の動作の変更
http://msdn.microsoft.com/ja-jp/library/ms143682.aspx

SQL Server 2016 の Analysis Services 機能における重大な変更
http://msdn.microsoft.com/ja-jp/library/ms143742.aspx

多次元データベースの互換性レベルの設定
http://msdn.microsoft.com/ja-jp/library/gg471593.aspx

Analysis Services のアップグレード
http://msdn.microsoft.com/ja-jp/library/ms143686.aspx

## おわりに

最後までこのドキュメントを読まれた皆さま、いかがでしたでしょうか？ SQL Server 2016 への移行／アップグレードは非常に簡単に行えることを確認していただけたのではないでしょうか。筆者自身も多くのお客様の移行／アップグレードをサポートしてきましたが、本文中で説明した細かい注意点はいくつかありますが、アップグレード インストールの失敗などの大きな問題は発生したことがありません。

これをきっかけに SQL Server 2016 のパワフルな新機能（インメモリ OLTP と列ストア インデックスの融合やセキュリティ強化、INSERT..SELECT のパラレル処理、R 統合、Reporting Services のモバイル レポートなど）をぜひ試してみてください。多くの環境で性能向上を確認できると思います。

SQL Server 2016 からの新機能に関しては、SQL Server 2016 自習書シリーズで詳しく説明しているので、こちらもぜひご覧いただければと思います。

SQL Server 2016 自習書シリーズ
https://www.microsoft.com/ja-jp/cloud-platform/products-SQL-Server-2016-Evaluate.aspx

No.1「SQL Server 2016 の新機能の概要」
No.2「Operational Analytics ～ OLTP とデータ分析の両立～」
No.3「Reporting Services の新機能 ～ Datazen の統合～」
No.4「Analysis Services の新機能」

## 執筆者プロフィール

### 有限会社エスキューエル・クオリティ（http://www.sqlquality.com/）

SQLQuality（エスキューエル・クオリティ）は、**日本で唯一の SQL Server 専門の独立系コンサルティング会社**です。過去のバージョンから最新バージョンまでの SQL Server を知りつくし、多数の実績と豊富な経験を持つ、OS や .NET にも詳しい **SQL Server の専門家（キャリア 20 年以上）がすべての案件に対応します**。人気メニューの「**パフォーマンス チューニング サービス**」は、100%の成果を上げ、過去すべてのお客様環境で驚異的な性能向上を実現。チューニング スキルは**世界トップレベル**を自負、検索エンジンでは（英語情報を含めて）ヒットしないノウハウを多数保持。ここ数年は **BI／DWH システム構築支援**のご依頼が多く、支援だけでなく実際の構築も行う。

主なコンサルティング実績／構築実績

- 大手製造業の「**CAD 端末の利用状況の見える化**」システム構築
  Oracle や CSV（Notes）、TSV ファイル、Excel からデータを抽出し、SQL Server 2012 上に DWH を構築
  見える化レポートには Reporting Services を利用
- 大手映像制作会社の **BI システム構築**（会計／業務システムにおける予実管理／原価管理など）
  従来 Excel で管理していたシートを Reporting Services のレポートへ完全移行。
  Oracle や勘定奉行からデータを抽出して、SQL Server 上に DWH を構築
- 大手流通系の **DWH／BI システム構築支援**（POS データ／在庫データ分析／ABC 分析／ポイントカード分析）
- 大手アミューズメント企業の **BI システム構築支援**（人事システムにおける人材パフォーマンス管理）
  Reporting Services による勤怠状況の見える化レポートの作成、PostgreSQL／人事システムからのデータ抽出
- 外資系医療メーカーの **BI システム**構築支援（Analysis Services と Excel による販売分析システム）
  OLAP キューブによる売上および顧客データの多次元分析／自由分析（ユーザーによる自由操作が可能）
- 大手流通系の **DWH システム**のパフォーマンス チューニング
  **データ量 100 億件**の DWH、**総ステップ数 2 万**越えのストアド プロシージャのパフォーマンス チューニング
- ミッション クリティカルな**金融システム**でのトラブル シューティング／定期メンテナンス支援
- SQL Server の下位バージョンからの**移行／アップグレード**支援（32 ビットから x64 への対応も含む）
- 複数台の SQL Server の **Hyper-V 仮想環境**への移行支援（サーバー統合支援）
- ハードウェア リプレース時の**ハードウェア選定**（最適なサーバー、ストレージの選定）、**高可用性環境**の構築
- **2 時間**かかっていた日中バッチ実行時間を、わずか **5 分**へ短縮（**95.8%** の性能向上）
- **Java 環境**（Tomcat、Seasar2、S2Dao）の SQL Server パフォーマンス チューニング etc

コンサルティング時の作業例（パフォーマンス チューニングの場合）

- アプリケーション コード（VB、C#、Java、ASP、VBScript、VBA）の解析／改修支援
- ストアド プロシージャ／ユーザー定義関数／トリガー（Transact-SQL）の解析／改修支援
- インデックス チューニング／SQL チューニング／ロック処理の見直し
- 現状のハードウェアで将来のアクセス増にどこまで耐えられるかを測定する高負荷テストの実施
- IIS ログの解析／アプリケーション ログ（log4net／log4j）の解析
- ボトルネック ハードウェアの発見／ボトルネック SQL の発見／ボトルネック アプリケーションの発見
- SQL Server の構成オプション／データベース設定の分析／使用状況（CPU, メモリ, ディスク, Wait）解析
- 定期メンテナンス支援（インデックスの再構築／断片化解消のタイミングや断片化の事前防止策など）etc

### 松本美穂（まつもと・みほ）

有限会社エスキューエル・クオリティ 代表取締役 Microsoft MVP for Data Platform（2004 年 4 月～）
経産省認定データベース スペシャリスト／MCDBA／MCSD for .NET／MCITP Database Administrator
SQL Server の日本における最初のバージョンである「SQL Server 4.21a」から SQL Server に携わり、現在、SQL Server を中心とするコンサルティングを行っている。得意分野はパフォーマンス チューニングと Reporting Services。著書の『SQL Server 2000 でいってみよう』と『ASP.NET でいってみよう』（いずれも翔泳社刊）は、トップ セラー（前者は 28,500 部、後者は 16,500 部発行）。近刊に『SQL Server 2016 の教科書』（ソシム刊）がある。

### 松本崇博（まつもと・たかひろ）

有限会社エスキューエル・クオリティ 取締役 Microsoft MVP for Data Platform（2004 年 4 月～）
経産省認定データベース スペシャリスト／MCDBA／MCSD for .NET／MCITP Database Administrator
SQL Server の BI システムとパフォーマンス チューニングを得意とするコンサルタント。アプリケーション開発（ASP／ASP.NET、C#、VB 6.0、Java、Access VBA など）やシステム管理者（IT Pro）経験もあり、SQL Server だけでなく、アプリケーションや OS、Web サーバーを絡めた、総合的なコンサルティングが行えるのが強み。Analysis Services と Excel による BI システムも得意とする。マイクロソフト認定トレーナー時代の 1998 年度には、Microsoft CPLS トレーナー アワード（Trainer of the Year）を受賞。