Microsoft

SQL Server 2008 自習書シリーズ No.12

バックアップと復元

Published: 2008 年 5 月 31 日
改訂版: 2008 年 10 月 27 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

STEP 1. バックアップの概要と自習書を試す環境について

この STEP では、バックアップの概要と自習書を試す環境について説明します。

この STEP では、次のことを学習します。

- ✓ バックアップの概要
- ✓ 自習書を試す環境について

1.1　バックアップの概要

バックアップの概要

SQL Server のバックアップには、次の 2 種類があります。

- **オフライン バックアップ**
- **オンライン バックアップ**

オフライン バックアップは、SQL Server を停止してから実行するバックアップで、オンライン バックアップは、SQL Server を実行したままの状態で実行できるバックアップです。どちらも一長一短があるので、その使い分け方法について、この自習書で説明します。

そのほかに、この自習書で説明するトピックは、次のとおりです。

- **トランザクション ログの肥大化の防止**
- **ハードウェア リプレイス時のデータ移動**
- **別のマシンへデータを移動する場合の注意事項**
- **バックアップ圧縮（SQL Server 2008 からの新機能）**
- **障害発生直前までのデータ復旧**

SQL Server では、トランザクション ログの管理を行っていないと、ディスク容量が満杯になるまで、どんどん肥大化していきます。満杯になって、ディスクの空き領域がなくなった場合は、それ以上トランザクションが実行できなくなります。トランザクション ログの管理は、SQL Server を利用する上で必須の管理作業になるので、確実に覚えておくことが重要です。

1.2　自習書を試す環境について

必要な環境

この自習書で実習を行うために必要な環境は次のとおりです。

OS

Windows Server 2003 SP2（Service Pack 2）以降 または

Windows XP Professional SP2 以降 または

Windows Vista または

Windows Server 2008

ソフトウェア

SQL Server 2008 Enterprise / Developer / Standard / Workgroup Edition

この自習書内での画面やテキストは、OS に Windows Server 2003 SP2、ソフトウェアに SQL Server 2008 Enterprise Edition を利用して記述しています。

STEP 2. オフライン バックアップ

この STEP では、オフライン バックアップの実行方法と、別マシンへの復元方法（アタッチ機能）、システム データベースのバックアップ方法などを説明します。

この STEP では、次のことを学習します。

- ✓ オフライン バックアップ
- ✓ 障害のシミュレート
- ✓ オフライン バックアップからの復元
- ✓ 別マシンでの復元（アタッチ）
- ✓ システム データベースのオフライン バックアップ
- ✓ ハードウェア リプレイス時のデータ移動

2.1　オフライン バックアップ（Offline Backup）

オフライン バックアップ

オフライン バックアップは、SQL Server を停止した状態で実行するバックアップです。オフライン バックアップは、開発機から本番機へのデータ移動や、マシン リプレイス時のデータ移動などにも利用できる、大変便利なバックアップです。

オフライン バックアップの実行は、非常に簡単で、次のように操作するだけで完了します。

1. **SQL Server 関連のサービスを停止する**
2. **バックアップしたいデータベースの実体となるファイル（.mdf と.ldf）をコピーする**

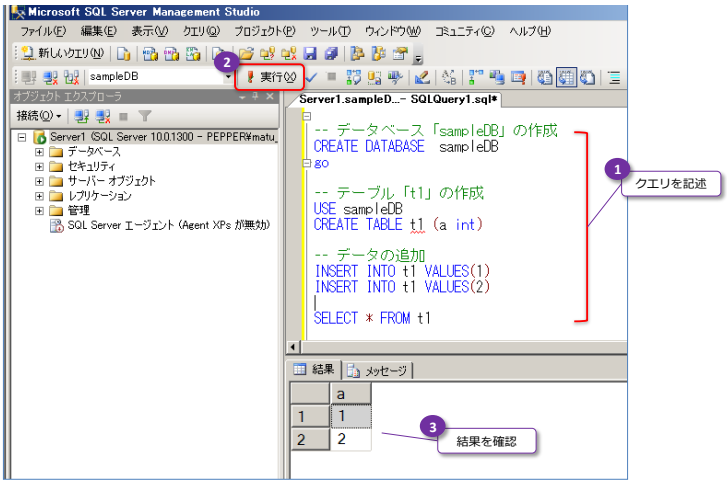
ファイルのコピーは、エクスプローラからの単純なコピーで大丈夫です。また、Windows に付属のバックアップ ツール（**ntbackup.exe**）やサードパーティ製のバックアップ ソフトを利用することもできます。

Let's Try

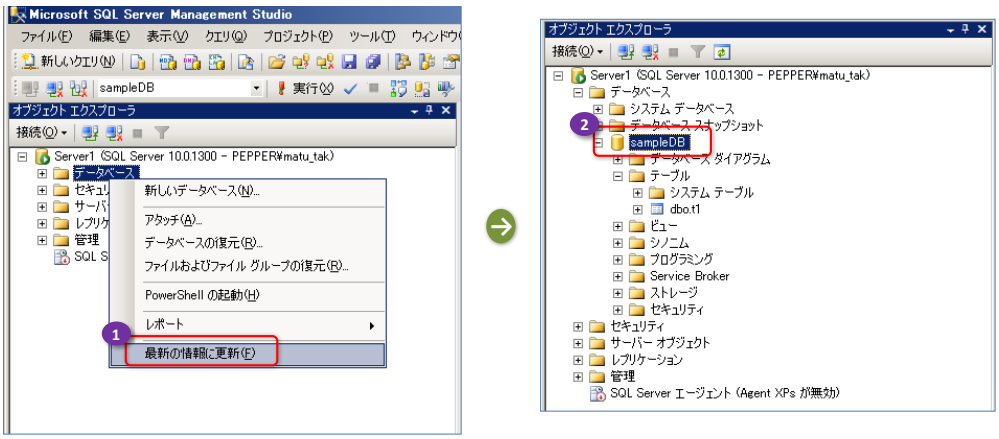
それでは、これを試してみましょう。まずは、バックアップをテストするためのデータベースを作成しましょう。

1. 最初に、[スタート] メニューから **Management Studio** を起動して、SQL Server へ接続します。
2. 続いて、**クエリ エディタ**を起動して、次のように「**sampleDB**」という名前のデータベースを作成し、その中へ「**t1**」テーブルを作成します。

```
-- データベース「sampleDB」の作成
CREATE DATABASE  sampleDB
go

-- テーブル「t1」の作成
USE sampleDB
CREATE TABLE t1 (a int)

-- データの追加
INSERT INTO t1 VALUES(1)
INSERT INTO t1 VALUES(2)

SELECT * FROM t1
```



3. 次に、オブジェクト エクスプローラで［**データベース**］フォルダを右クリックして［**最新の情報に更新**］をクリックし、作成した sampleDB データベースが追加されていることを確認します。

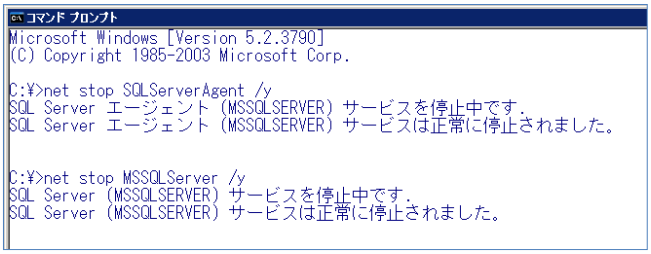


SQL Server 関連のサービスの停止

次に、オフライン バックアップを実行してみましょう。最初の手順は、SQL Server 関連のサービスの停止です。

1. SQL Server 関連のサービスを停止するには、［**スタート**］メニューの［**すべてのプログラム**］→［**アクセサリ**］から［**コマンド プロンプト**］を起動し、「**net stop**」コマンドを次のように実行します。

```
net stop SQLServerAgent /y
net stop MSSQLServer /y
```



「**SQLServerAgent**」と指定することで SQL Server Agent サービス、「**MSSQLServer**」と指定することで SQL Server サービスを停止することができます。SQL Server Agent サービスを利用していない場合は、サービスの停止は不要です。

サービスの停止は、Management Studio や SQL Server Configuration Manager（構成マネージャ）、管理ツールの［サービス］アプレットなどからも行えますが、オフライン バックアップの自動化（バッチ ファイルによる無人化）には、net stop コマンドが必須になるので、このコマンドを覚えておくことをお勧めします。

Note： 名前付きインスタンスの場合

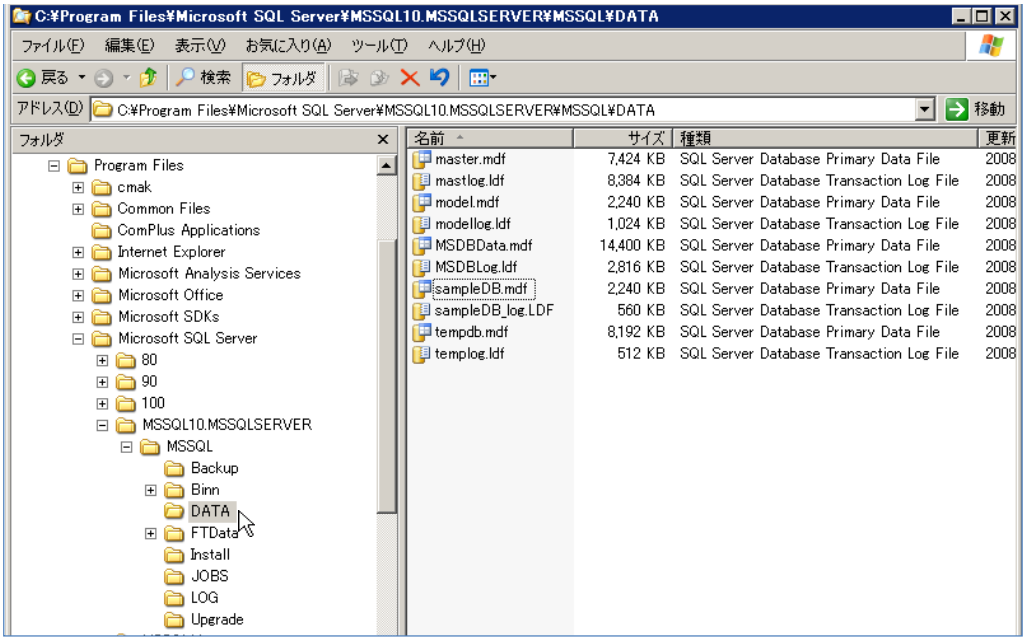
SQL Server を名前付きインスタンスとしてインストールしている場合は、次のように「**SQLAgent**」および「**MSSQL**」の後に「**$**」マークを付けて、インスタンス名を指定します。

```
net stop SQLAgent$インスタンス名 /y
net stop MSSQL$インスタンス名 /y
```

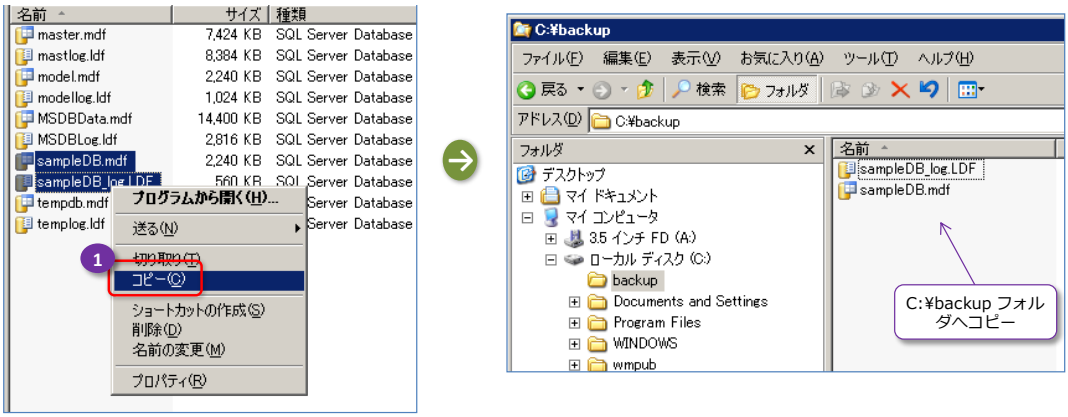
2. サービスの停止が完了したら、次は、Windows エクスプローラを起動して、「**sampleDB**」データベースの**データ ファイル（.mdf）**と**トランザクション ログ ファイル（.ldf）**が格納されているフォルダを開きます。前の手順では、データベースの作成先のパスを指定せずに作成したので、次のフォルダへ格納されています。

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

   名前付きインスタンスとしてインストールしている場合は、「**MSSQL10.MSSQLSERVER**」の部分が「**MSSQL10.インスタンス名**」になります。

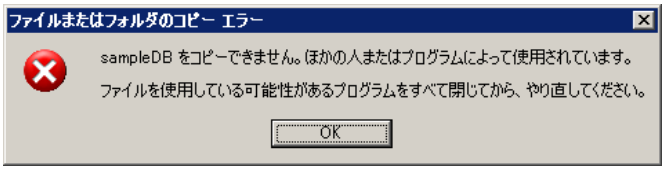


3. 次に、「**sampleDB.mdf**」と「**sampleDB_log.LDF**」ファイルを「**C:¥backup**」フォルダなど、任意のフォルダを作成して、そこへコピーします。



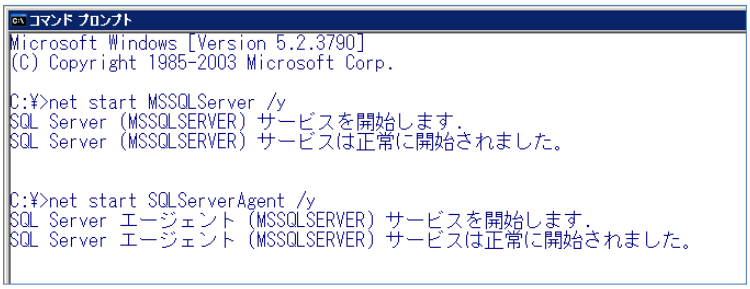
Note： SQL Server サービスが起動している場合のエラー

もし、SQL Server サービスを停止せずにファイルのコピーを実行した場合は、次のエラーが表示されます。



4. ファイルのコピーが完了したら、SQL Server 関連のサービスを開始します。サービスを開始するには、コマンド プロンプトから「**net start**」コマンドを次のように実行します。

```
net start MSSQLServer /y
net start SQLServerAgent /y
```



サービスの停止時は、SQL Server Agent サービスから停止しましたが、サービスを開始する場合は、逆になり、SQL Server サービスから起動します。

名前付きインスタンスを利用している場合は、「MSSQL$インスタンス名」および「SQLAgent$インスタンス名」と指定します。

障害のシミュレート

次に、データベース（ファイル）の破損をシミュレートしてみましょう。

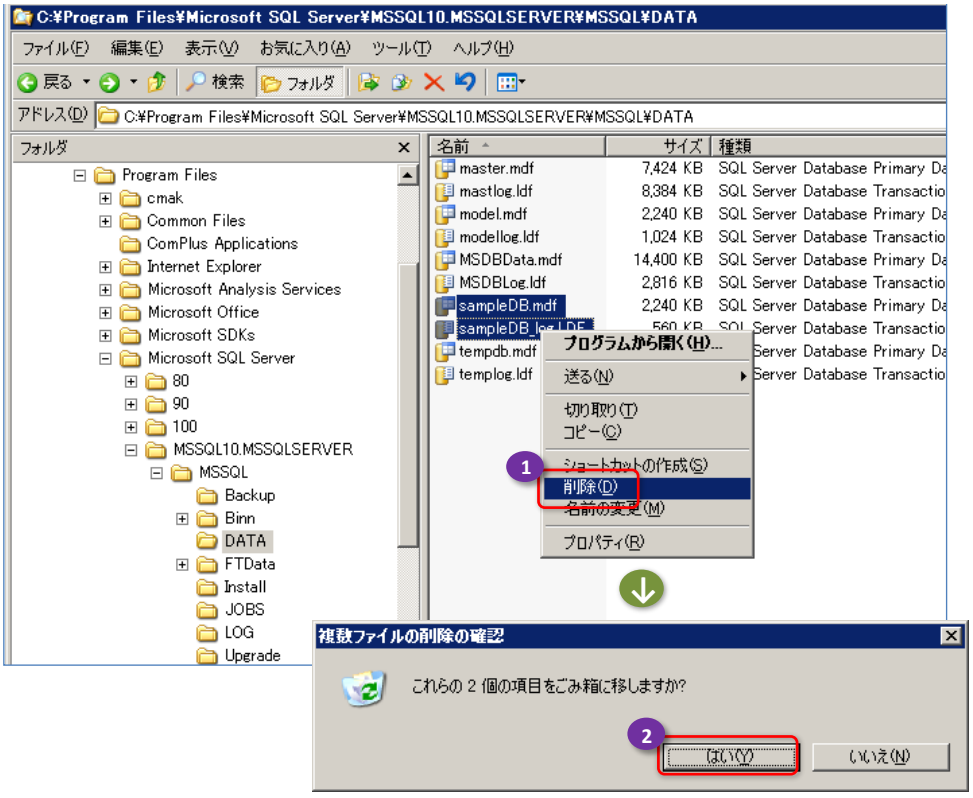
5. ファイルの破損をシミュレートには、まず SQL Server サービスを停止します。コマンド プロンプトから「**net stop**」コマンドを利用して、停止します。

```
net stop SQLServerAgent /y
net stop MSSQLServer /y
```

6. 次に、Windows エクスプローラを起動して、「**sampleDB**」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）**が格納されている以下のフォルダを開きます。

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

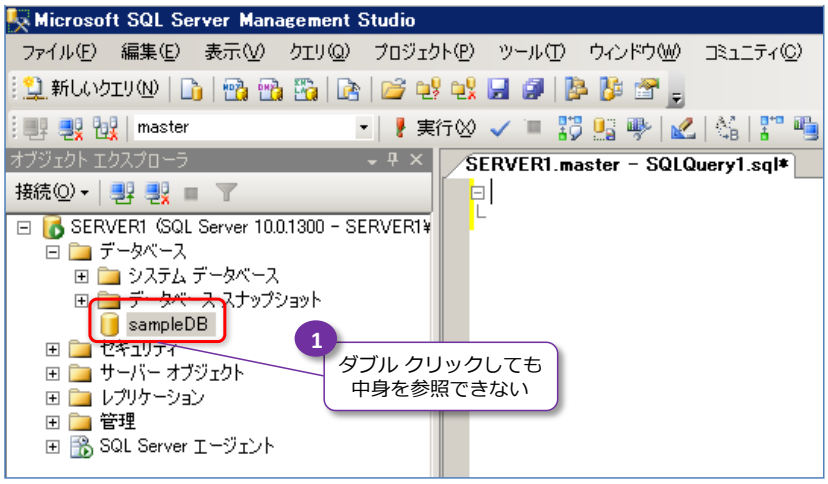
7. このフォルダ内の「**sampleDB.mdf**」と「**sampleDB_log.LDF**」ファイルを右クリックして、［**削除**］をクリックし、ファイルを削除します。



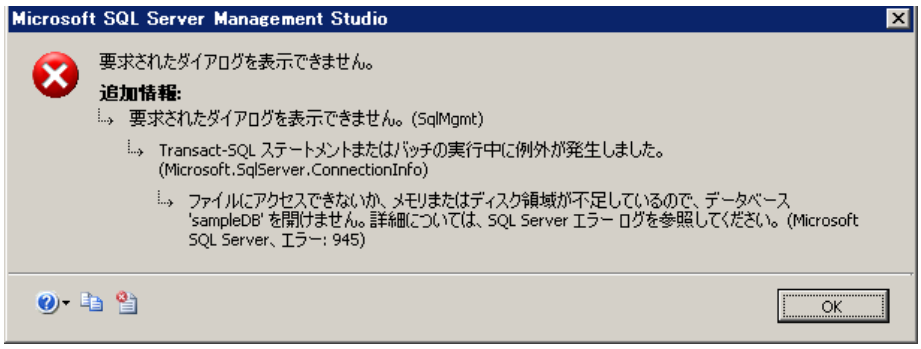
8. 削除後、コマンド プロンプトから「**net start**」コマンドを実行して、SQL Server サービスを開始しまます。

```
net start MSSQLServer /y
net start SQLServerAgent /y
```

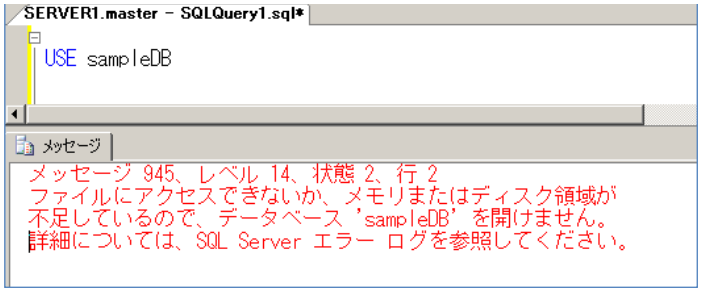
9. サービスが開始されたら、Management Studio を起動します。次のように「**sampleDB**」データベースの名前は一覧されますが、ダブル クリックしても中身を参照できないことを確認できます。



10. データベースを右クリックして、[**プロパティ**] をクリックしても、次のエラーが発生します。

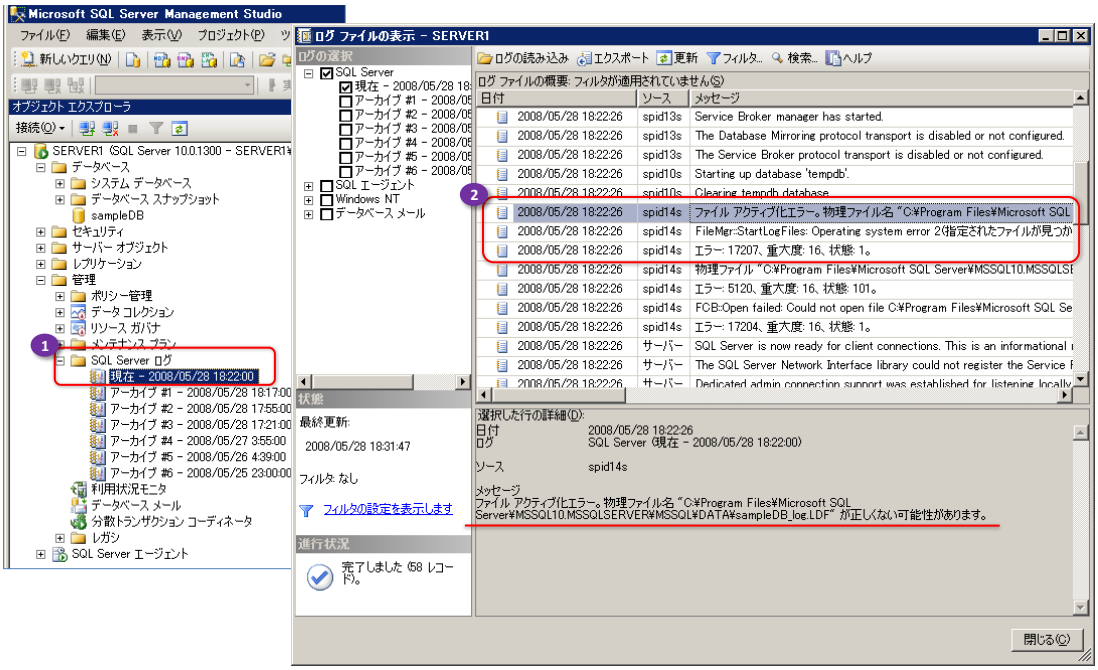


11. また、クエリ エディタから「**USE**」ステートメントでデータベースへ接続しようとしても、次のエラーが発生します。



SQL Server ログの確認

12. 次に、オブジェクト エクスプローラで［**管理**］フォルダの［**SQL Server ログ**］を展開して、［**現在 ～**］をダブル クリックして、SQL Server ログを表示します。



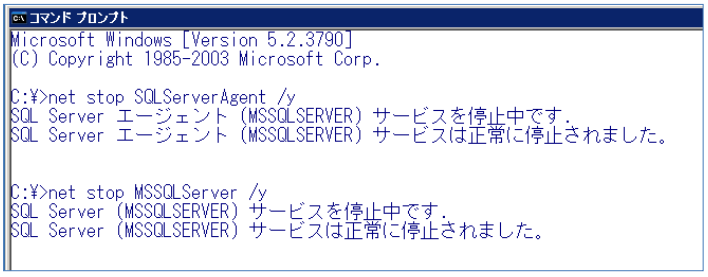
重大度レベルが「**16**」の「**ファイルのアクティブ化エラー**」が記録されていることを確認できます。このようにファイルが破損している場合は、SQL Server ログへエラーが記録されます。

オフライン バックアップからの復元

次に、オフライン バックアップ（ファイル コピー）で取得したファイルを復元して、障害から復旧していきましょう。

13. ファイルを復元するには、まず SQL Server サービスを停止します。コマンド プロンプトから「**net stop**」コマンドを利用して停止します。

```
net stop SQLServerAgent /y
net stop MSSQLServer /y
```

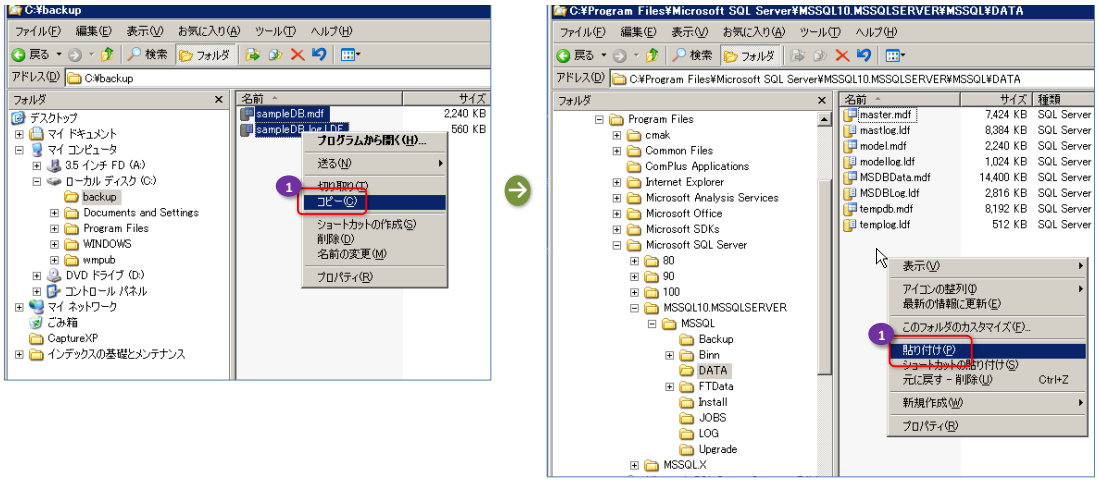


14. サービスの停止後、ファイルをバックアップしたフォルダ「**C:¥backup**」から「**sampleDB.mdf**」と「**sampleDB_log.LDF**」ファイルを、次のフォルダ（元々、ファイルが格納されていた場所）へコピーします。

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

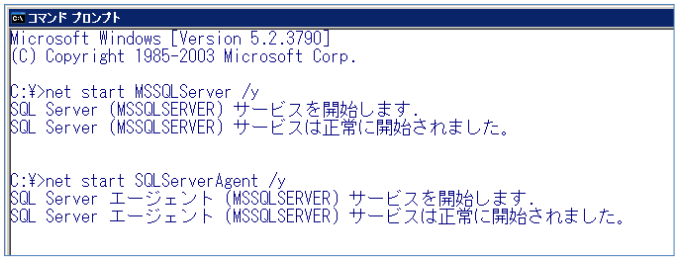
名前付きインスタンスとしてインストールしている場合は、「**MSSQL10.MSSQLSERVER**」

の部分を「**MSSQL10.インスタンス名**」とします。



15. ファイルのコピーが完了したら、コマンド プロンプトから「**net start**」コマンドを実行して、SQL Server サービスを開始します。

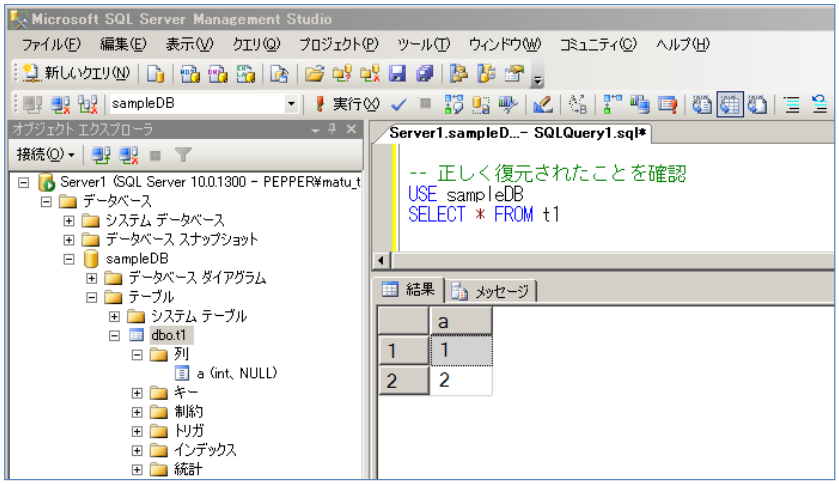
```
net start MSSQLServer /y
net start SQLServerAgent /y
```



以上で、オフライン バックアップからの復元が完了です。

16. 次に、正しく復元できたことを確認するために、Management Studio を起動して、クエリ エディタから、データベースへ接続してみましょう。

```
USE sampleDB
SELECT * FROM t1
```



データが正しく復元されていることを確認できます。

このように、オフライン バックアップは、非常に簡単にデータをバックアップできる機能です。SQL Server サービスを停止しなければいけないというデメリットがありますが、次に説明する別のマシンへデータを移動する場合や、マシン リプレイス時などで大変役立つので、ぜひ活用してみてください。

2.2　オフライン バックアップを別のマシンへ復元（アタッチ）

オフライン バックアップを別のマシンへ復元（アタッチ）

次は、オフライン バックアップで取得した「**sampleDB.mdf**」と「**sampleDB_log.LDF**」ファイルを別のマシン（別の SQL Server）に復元する方法を説明します。別のマシンで復元する場合には、アタッチ（Attach）機能を利用します。

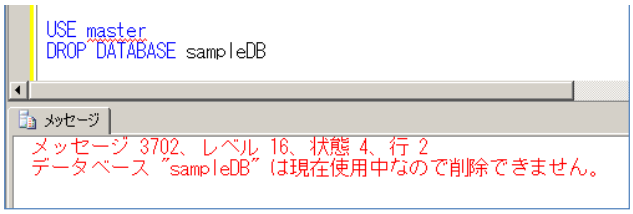
Let's Try

それでは、これを試してみましょう。

1. 別のマシンへの移動をシミュレートするために、まずはクエリ エディタから「**sampleDB**」データベースを削除します。

```
USE master
DROP DATABASE sampleDB
```

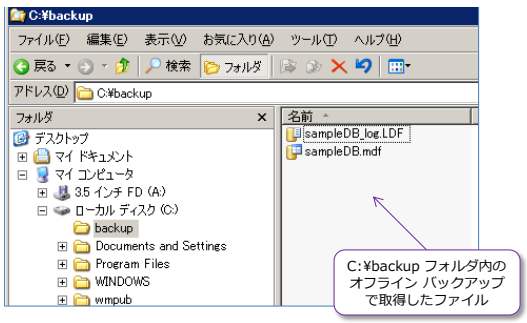
実行後、次のエラーが出る場合は、「sampleDB」データベースへ接続しているクエリ エディタを一度すべて閉じてから、再度実行してください。



それでも、「使用中」エラーが出る場合は、次のように ALTER DATABASE ステートメントを実行して、データベースへ接続しているユーザーを強制終了してから、データベースを削除します。

```
USE master
ALTER DATABASE sampleDB
 SET SINGLE_USER WITH ROLLBACK IMMEDIATE
```

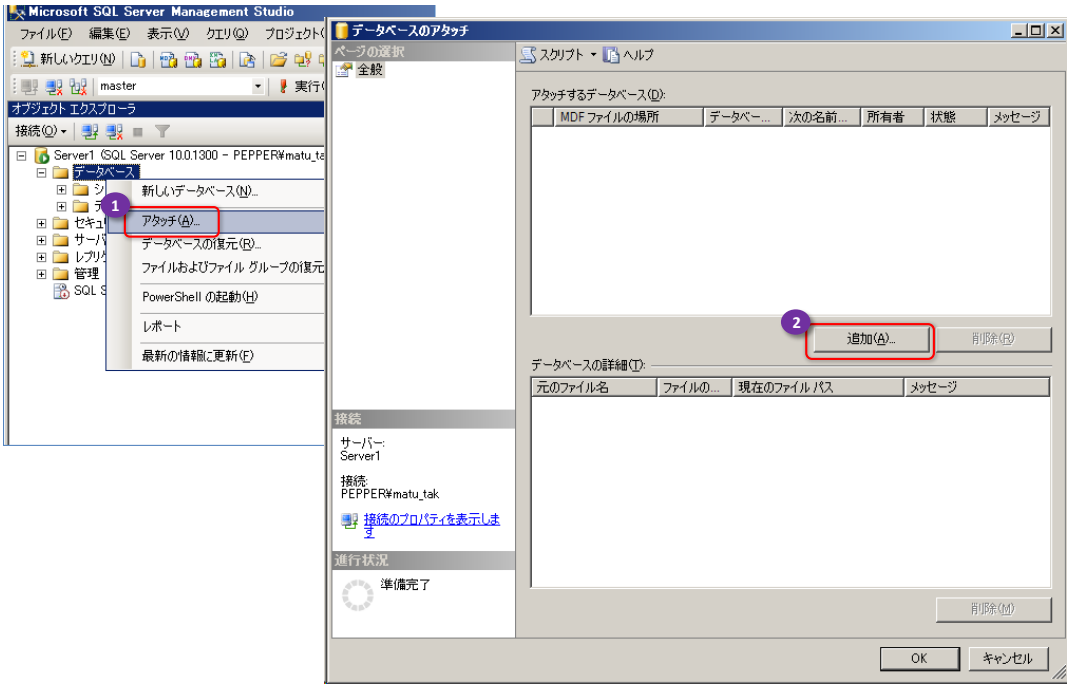
2. データベースの削除後、オフライン バックアップを取得したフォルダ「**C:¥backup**」を開きます。



データベースのアタッチ

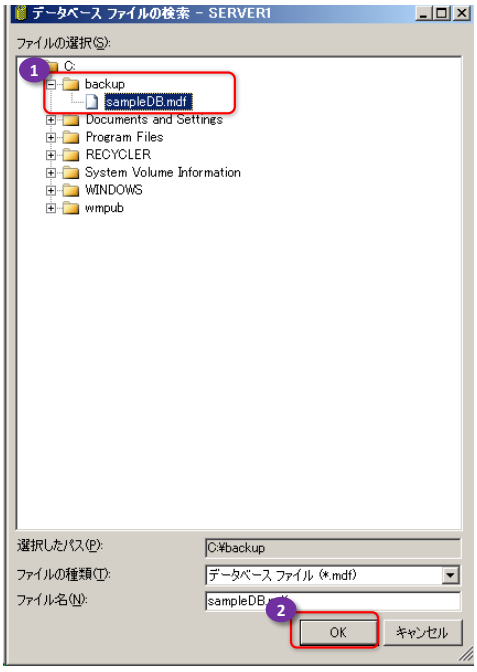
次に、このフォルダのデータ ファイル（.mdf）とログ ファイル（.ldf）を SQL Server へ認識させるために、アタッチを実行します。

3. アタッチを実行するには、Management Studio から［**データベース**］フォルダを右クリックして、［**アタッチ**］をクリックします。

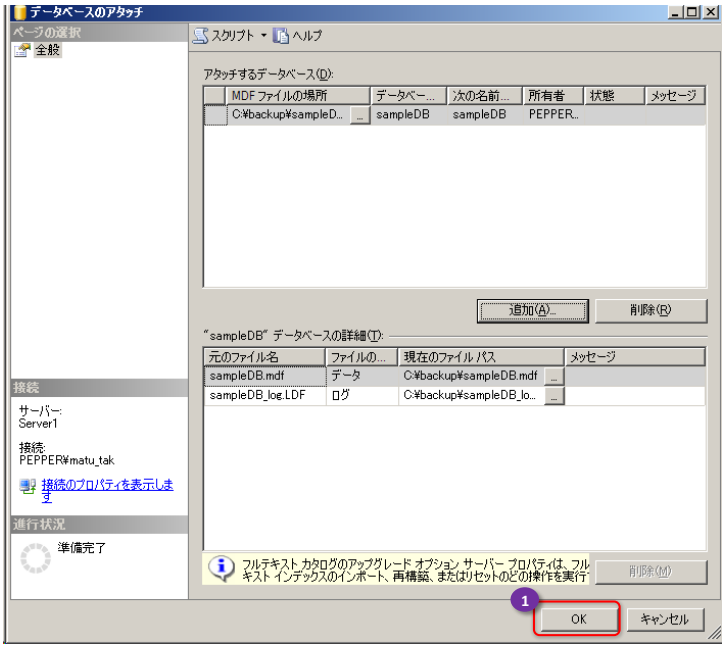


［データベースのアタッチ］ダイアログが表示されたら、［**追加**］ボタンをクリックします。

4. ［データベース ファイルの検索］ダイアログでは、「**C:¥backup**」フォルダを展開して、「**sampleDB.mdf**」ファイルを選択し、［OK］ボタンをクリックします。



5. ［データベースのアタッチ］ダイアログへ戻ったら、［OK］ボタンをクリックします。



以上でアタッチが完了です。このようにアタッチ機能を利用すれば、データベースを簡単に別の SQL Server へ移動させることができます。

Note： サービス アカウントに対して NTFS アクセス許可が必要

［データベース ファイルの検索］ダイアログで .mdf ファイルが表示されなかったり、アタッチが失敗する場合は、SQL Server のサービス アカウントに対して NTFS アクセス許可が与えられているかどうかをチェックしてください。

SQL ステートメントでアタッチ

SQL ステートメントを利用して、アタッチを実行するには、**CREATE DATABASE** ステートメントで、次のように **FOR ATTACH** キーワードを指定します。

```
USE master
go
CREATE DATABASE sampleDB
ON ( FILENAME = N'C:¥backup¥sampleDB.mdf' ),
   ( FILENAME = N'C:¥backup¥sampleDB_log.LDF' )
FOR ATTACH
```

アタッチ機能で複製できる／できないオブジェクト

アタッチ機能は、あくまでもデータベースのみの移動なので、複製できるオブジェクトは、データベース内へ作成したオブジェクトのみになります。具体的には、次の表のオブジェクトを移動することができます。

複製可能なオブジェクト	
・テーブル ・制約 ・インデックス ・トリガ ・ビュー ・シノニム ・データベース ユーザー ・スキーマ ・データベース ロール ・アプリケーション ロール	・ストアド プロシージャ ・ユーザー定義関数 ・ユーザー定義データ型 ・SQLCLR オブジェクト ストアド プロシージャ ユーザー定義関数 ユーザー定義データ型 集計関数 ・データ パーティション ・フルテキスト インデックス ・データベース対称キー / 非対称キー / 証明書

これに対して、ログイン アカウントや環境設定オプション（sp_configure で設定したもの）、ジョブ、警告など、システム データベースへ格納される情報と、レジストリへ格納される情報については、データベースのアタッチだけでは複製することができないので、別途複製する必要があります。複製できないものは、主に次のとおりです。

master	msdb	レジストリ
・ログイン アカウント ・環境設定オプション（sp_configure） ・リンク サーバー設定 ・ユーザー定義エラー	・ジョブ ・警告 ・オペレータ ・Integration Services パッケージ	・セキュリティ モード（認証モード） ・TCP ポート番号

これらの複製方法については、次の項以降で説明します。

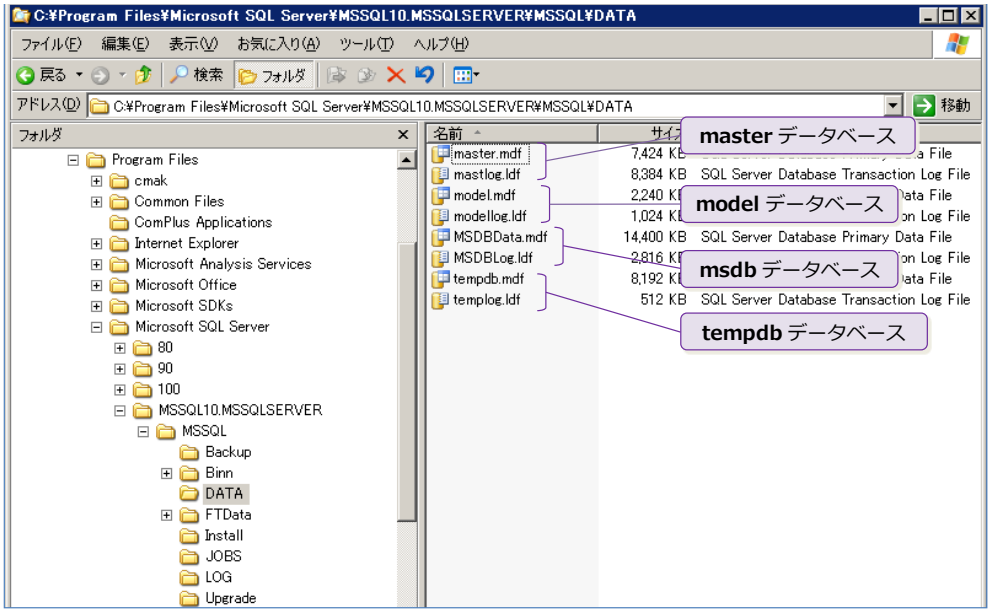
2.3　システム データベースのオフライン バックアップ

システム データベースのオフライン バックアップ

SQL Server のシステム データベース（**master**、**msdb**、**model**、**tempdb**）の実体となるファイル（.mdf と .ldf）は、次のフォルダへ格納されています。

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

名前付きインスタンスとしてインストールしている場合は、「**MSSQL10.MSSQLSERVER**」の部分が「**MSSQL10.インスタンス名**」になります。



これらのシステム データベースも、通常のデータベースのオフライン バックアップと同様、SQL Server サービスを停止した状態で、バックアップ（ファイル コピー）することが可能です。

Note： tempdb のバックアップは不要

tempdb データベース（tempdb.mdf と tempdb.ldf）は、SQL Server の起動時に、model データベースをもとに毎回再作成されているので、バックアップを取得しておく必要はありません。

Note： システム データベースのバックアップのタイミング

システム データベース（master、msdb、model）をバックアップするタイミングは、それぞれのデータベースへ格納される情報（ログイン アカウントや環境設定オプション、ジョブ、警告など）を変更した後や、１週間に １回など定期的に実行することをお勧めします。また、レプリケーション機能を利用している場合は、**distribution** データベースが作成されて、利用されるので、このデータベースについても、１週間に １回など定期的にオフライン バックアップを取得することをお勧めします。

別のマシンへのシステム データベースの移動は基本的に不可

別のマシン（コンピュータ名が異なるマシン）へのシステム データベースの移動は、基本的にはしてはいけません。移動後も、SQL Server サービスは問題なく起動しますが、いくつかの制限を受けます。この制限は、次のコマンドを実行することで、回避できるものもあります。

```
EXEC sp_dropserver '古いサーバー名'
EXEC sp_addserver '新しいサーバー名', @local='local'
```

しかし、このコマンドを実行しても、すべての機能が動作するという保証はありません。何が動作しなくなるのかは、オンライン ブックには明確な記載はないので、トラブルが発生する可能性がゼロではありません。したがって、コンピュータ名が異なるマシンへのシステム データベースの移動は、行わないことをお勧めします。

なお、同じ名前のコンピュータ名を設定した別マシンであれば、システム データベースを移動して、何の問題もなく SQL Server を動作させることができます（移動元の SQL Server とまったく同じように動作させることが可能です）。したがって、次の項で説明する「ハードウェア リプレイス時」などでは、オフライン バックアップで取得したシステム データベースをコピーすると、非常に簡単にリプレイス作業を実施することができます。

2.4　ハードウェア リプレイス時のデータ移動

ハードウェア リプレイス時のデータ移動

ハードウェアの老朽化に伴って、マシンをリプレイスする場合には、システム データベースのオフライン バックアップが大変役立ちます。これを利用すれば、非常に簡単にリプレイス作業を実施することができます。

ハードウェアをリプレイスする場合のおおまかな手順は、次のとおりです。

1. 新マシンへ OS をインストールする。このときのマシン名は任意でも可能
2. 現在のマスタの SQL Server サービスを停止
3. **現在のマスタでオフライン バックアップを取得**（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ファイル コピーで取得）
4. 現在のマスタを停止する（OS のシャットダウン）
5. Active Directory ドメインのコンピュータ アカウントの一覧から現在のマスタを削除する（この手順は、ワークグループ環境の場合は不要）
6. **新マシンの名前を旧マシンと同じ名前へ変更する**（この作業は SQL Server をインストールする前に必ず行っておくこと）
7. 新マシンの OS を再起動する
8. 新マシンを Active Directory ドメインへ参加させる
9. 新マシンへ SQL Server をインストールする（このとき、インストール先のフォルダは、現在のマスタをインストールしたときのパスとまったく同じにすること。また、照合順序などの設定も、現在のマスタを同じに設定すること）
10. 現在のマスタへインストール済みの Service Pack および修正モジュールを、新マシンへインストールする（このときにインストールするモジュールは、現在のマスタへインストールしているものと同じモジュールのみであることに注意すること）
11. SQL Server を停止する
12. **オフライン バックアップで取得したすべてのデータベースを新マシンへコピーする**（システム データベースは、上書きコピーすれば OK）
13. SQL Server を開始する
14. 手順 10 でインストールしたものよりも、最新の Service Pack や修正モジュールがある場合は、それをインストールする

以上でリプレイス作業が完了です。このように、オフライン バックアップを利用すると、非常に簡単にハードウェア リプレイス作業を行うことができます。

2.5　オフライン バックアップの利点と欠点

オフライン バックアップの利点と欠点

最後にオフライン バックアップの利点と欠点をまとめておきます。

まず、オフライン バックアップの利点は、次のとおりです。

- 操作が簡単
- **システム データベースのバックアップに最適**
- **ハードウェア リプレイス時に最適**

これに対して、欠点は、次のとおりです。

- SQL Server を停止しなければならない
- **トランザクション ログの切り捨てを定期的に実行しなければならない。**または復旧モデルを「単純」で運用し、トランザクション ログが肥大化するのを防ぐ必要がある
- **毎回ファイルのサイズ分だけバックアップしなければならない**（後述のオンライン バックアップよりもサイズが大きくなる。また差分や増分バックアップ機能がない）
- 障害発生直前までの復旧ができない（トランザクション ログ バックアップと組み合わせた運用ができない）

これらの欠点は、オンライン（Online）バックアップを利用すれば、すべて解決します。したがって、実際の運用では、オフライン バックアップとオンライン バックアップを併用していくことが重要になります。オフライン バックアップは、システム データベースのバックアップやマシン リプレイス時のデータ移動用途として最適で、オンライン バックアップは、ユーザーが作成したデータベースのバックアップ時に最適です。

次の Step では、オンライン バックアップの基本操作や差分バックアップ機能、増分バックアップが可能になるトランザクション ログ バックアップ機能などを説明します。

STEP 3. オンライン バックアップの基本操作

この STEP では、オンライン バックアップの基本操作と定期バックアップの実行方法、バックアップ圧縮の実行方法、トランザクション ログの肥大化の防止方法などを説明します。

この STEP では、次のことを学習します。

- ✓ オンライン バックアップの概要
- ✓ 完全バックアップ
- ✓ スケジュール機能による定期バックアップの実行（ジョブ）
- ✓ バックアップ セットとは
- ✓ バックアップ圧縮
- ✓ トランザクション ログの管理（肥大化防止）

3.1　オンライン バックアップ（Online Backup）の概要

オンライン バックアップの概要

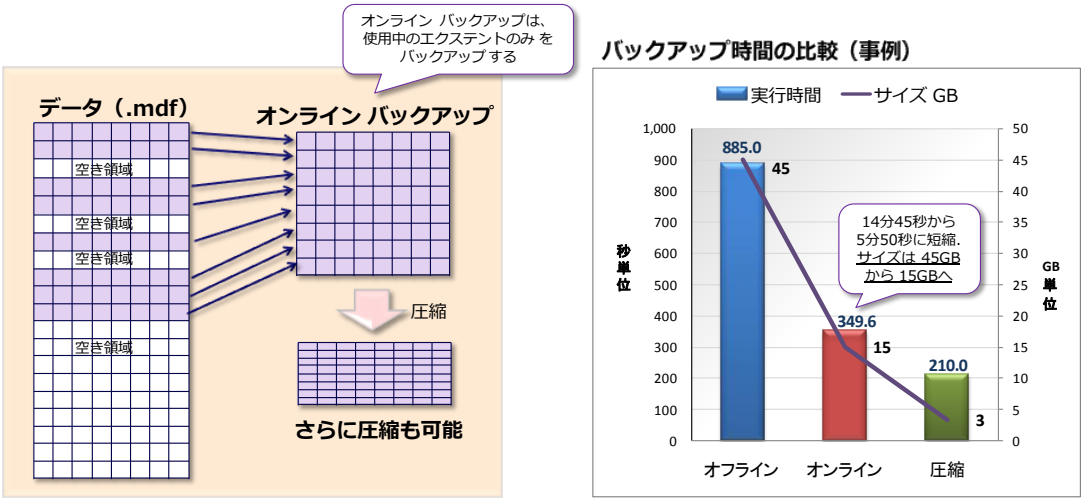
オンライン バックアップは、SQL Server を起動したままの状態でバックアップが取得できる機能で、オフライン バックアップと比べて、次の利点があります。

- SQL Server を停止する必要がない
- トランザクション ログのバックアップを定期実行することで、**ログの肥大化を防げる**
- オフライン バックアップよりもファイル サイズを小さくできる = オフライン バックアップよりもバックアップ時間を短くできる
- 差分または増分のみをバックアップできる
- バックアップ圧縮機能によって、さらにバックアップ時間を短縮できる（SQL Server 2008 からの新機能で Enterprise Edition でのみ利用可能）
- 障害発生直前までの復旧が可能

これらの利点については、以降の Step で実際に試していきます。

オフライン vs. オンライン

オフライン バックアップとオンライン バックアップの内部動作を比較すると、次の図のようになります。



オフライン バックアップは、データ ファイル（.mdf）のコピーになるので、ファイルのサイズ分のコピーが必要になる（空き領域もコピーしなければならない）のに対して、オンライン バックアップでは、使用中のデータ（エクステント）のみをバックアップするだけで済みます。

バックアップにかかる時間は、デバイス（テープやディスク）の書き込み速度に大きく依存するので、書き込む量を減らせると、バックアップ時間を短縮できます。右のグラフは、筆者のお客様のデータ（45GB のデータベース）でオフライン バックアップを実行した場合と、オンライン バッ

クアップを実行した場合の実行時間とサイズを比較したものです。オフライン バックアップでは **45GB** 分のファイル コピーが必要なのに対して、オンライン バックアップでは **15GB**（使用しているデータ サイズ）分のバックアップで済んでいるので、バックアップ時間を **14 分 45 秒**から **5 分 50 秒**に短縮（約 9 分の短縮）することができています。

また、このグラフでは、後述する「**バックアップ圧縮**」機能を利用した場合の速度結果も掲載しています。バックアップ圧縮では、バックアップ サイズを **3GB** まで縮小することができ、さらに書き込み量を減らすことができるので、バックアップ時間を **3 分 30 秒**にまで短縮できています。

このように、オフライン バックアップは、データ サイズが大きなユーザー データベースでは、バックアップ時間が長くなってしまうので、オンライン バックアップを利用したほうがバックアップ時間を短縮できます。また、バックアップ圧縮機能を利用すれば、さらにバックアップ時間を短縮することもできます。

この Step では、オンライン バックアップの実行方法やバックアップ圧縮の実行方法について説明します。また、差分バックアップやログ バックアップについても説明します（これらもバックアップ時間の短縮に役立つ機能です）。

オンライン バックアップの種類

オンライン バックアップには、次の 3 種類があります。

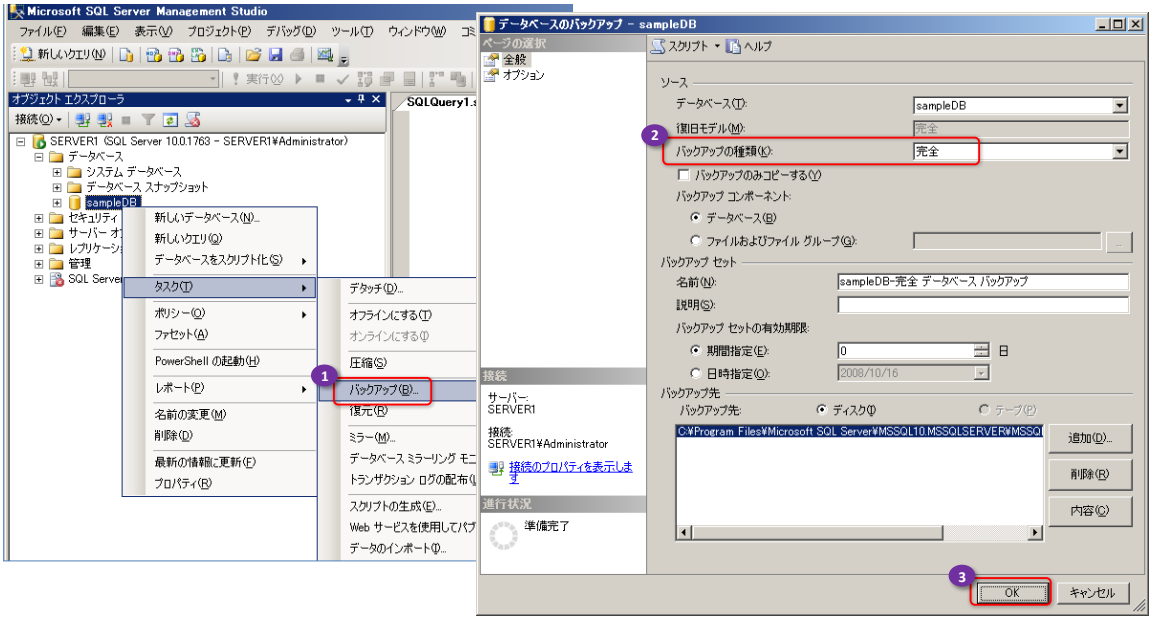
- **完全バックアップ（フル バックアップ）**
- **差分バックアップ**
- **ログ バックアップ（トランザクション ログ バックアップ）**

完全バックアップは、データベース全体を丸ごとバックアップし、差分バックアップは、前回の完全バックアップ以降の差分のみ、ログ バックアップは、前回のログ バックアップ以降の差分（増分）のみをバックアップできる機能です。それぞれの使い分けについては、以降で説明します。

3.2　完全バックアップ（フル バックアップ）

完全バックアップ

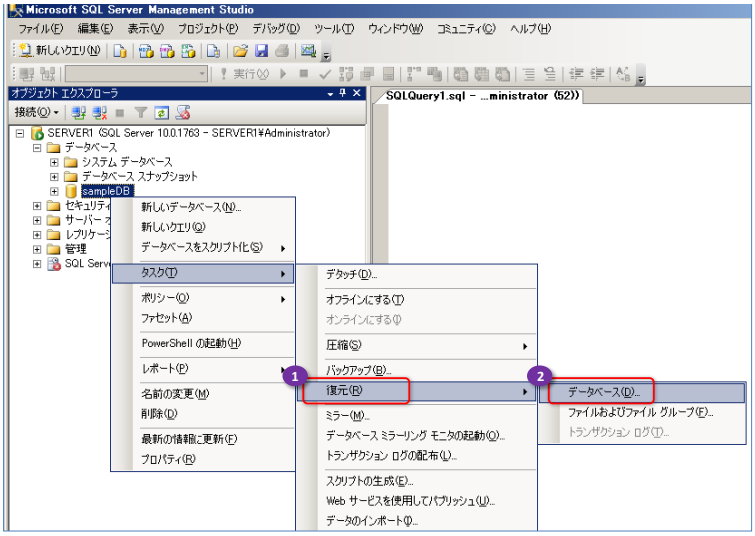
完全バックアップは、データベース全体を丸ごとバックアップする機能です。完全バックアップを実行するには、Management Studio から、次のように該当データベースを右クリックして、[**タスク**] メニューの [**バックアップ**] をクリックします。

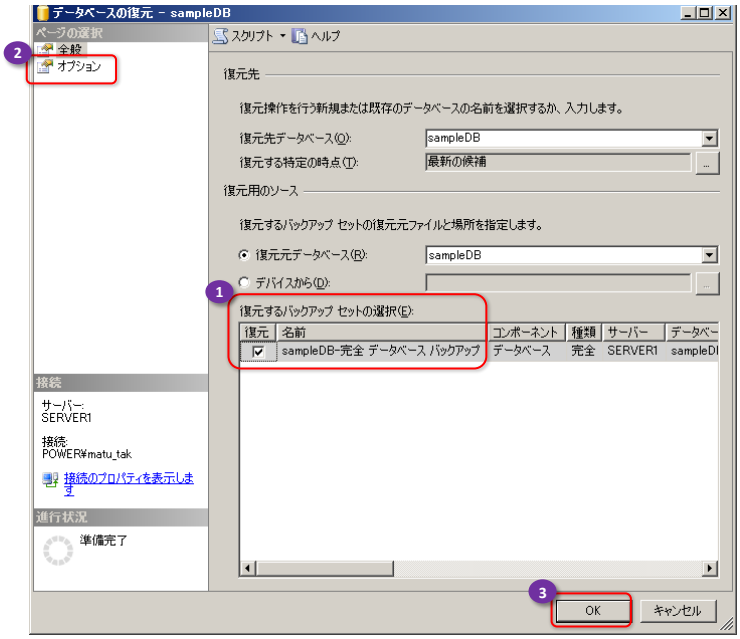


[データベースのバックアップ] ダイアログで、[**バックアップの種類**] で「**完全**」を選択すれば、完全バックアップを実行できます。

完全バックアップからのリストア（復元）

完全バックアップからリストアをするには、次のように [**タスク**] メニューの [**復元**] から [**データベース**] をクリックします。





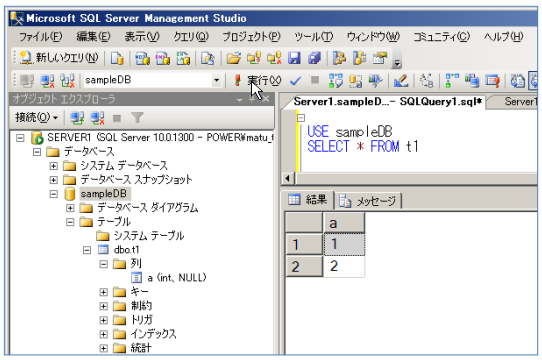
［データベースの復元］ダイアログでは、復元したいバックアップを選択し、［**オプション**］ページで必要に応じてオプションを設定し、［OK］ボタンをクリックします。オプション設定については、この後、説明していきます。

Let's Try

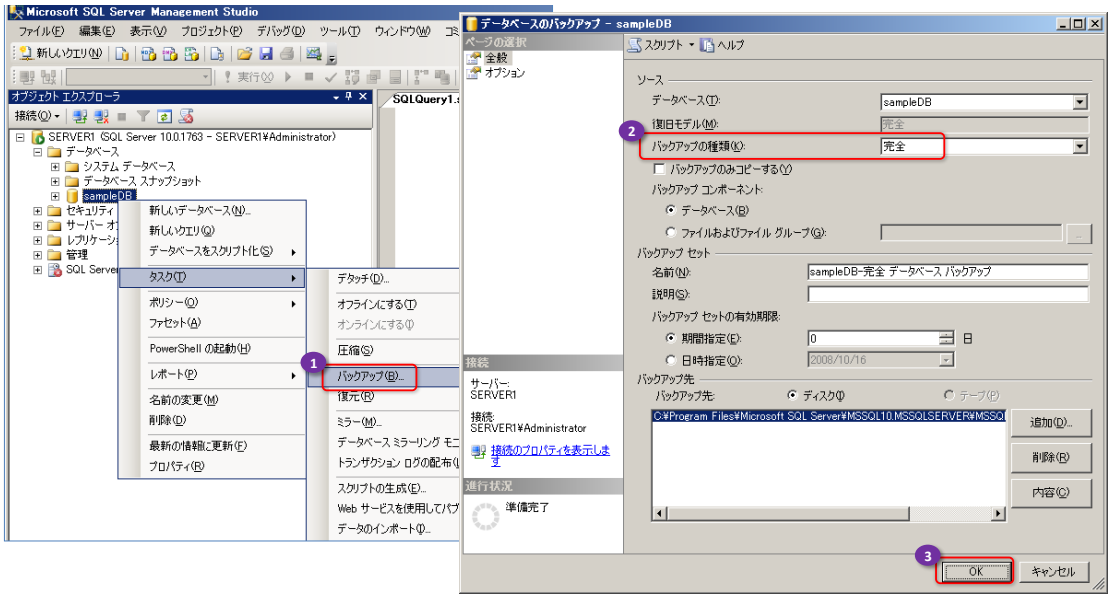
それでは、完全バックアップとリストアを試してみましょう。

1. まずは、クエリ エディタから、前の Step で作成した「**sampleDB**」データベースへ接続して、バックアップ前の「**t1**」テーブルの中身を確認します。

```
USE sampleDB
SELECT * FROM t1
```

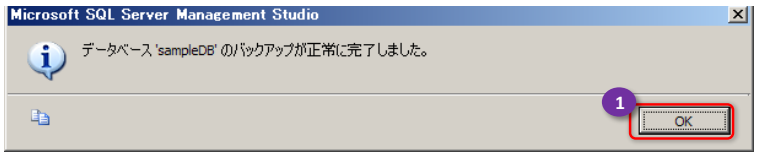


2. 次に、オブジェクト エクスプローラから「**sampleDB**」データベースを右クリックして、［**タスク**］メニューの「**バックアップ**」をクリックします。



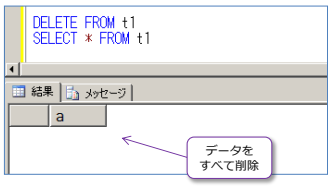
［データベースのバックアップ］ダイアログで、［**バックアップの種類**］で「**完全**」を選択して、［OK］ボタンをクリックします。

これで完全バックアップが開始され、実行が完了すると、次のダイアログが表示されます。



3. バックアップが完了したら、クエリ エディタから、次のように **DELETE** ステートメントを実行して、すべてのデータを削除します。

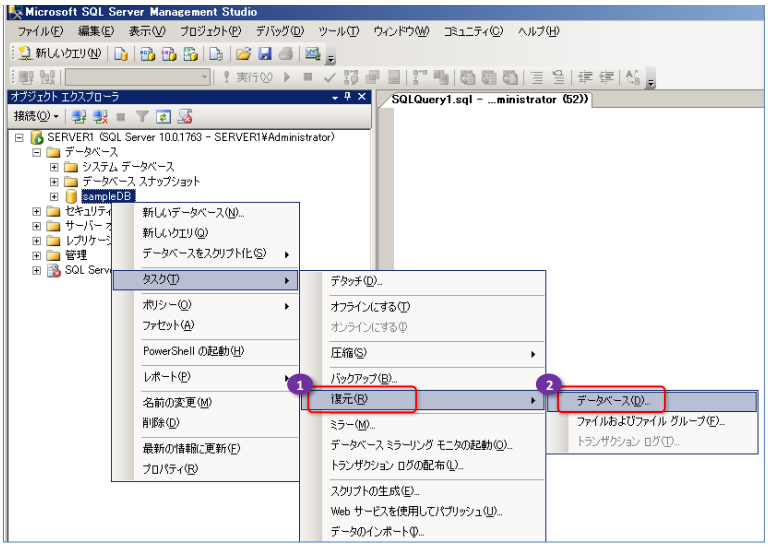
```
DELETE FROM t1
SELECT * FROM t1
```



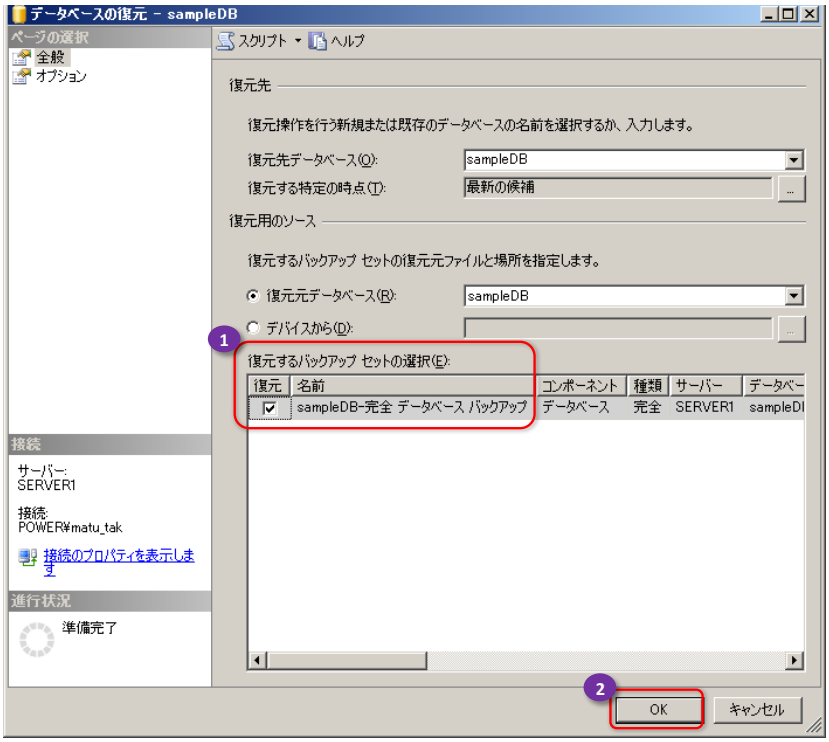
完全バックアップからのリストア

次に、完全バックアップからリストアを実行してみましょう。

4. リストアを実行するには、次のように［**タスク**］メニューの［**復元**］から［**データベース**］をクリックします。



［データベースの復元］ダイアログが表示されたら、［**復元するバックアップ セットの選択**］から完全バックアップが選択されているのを確認して、［**OK**］ボタンをクリックします。

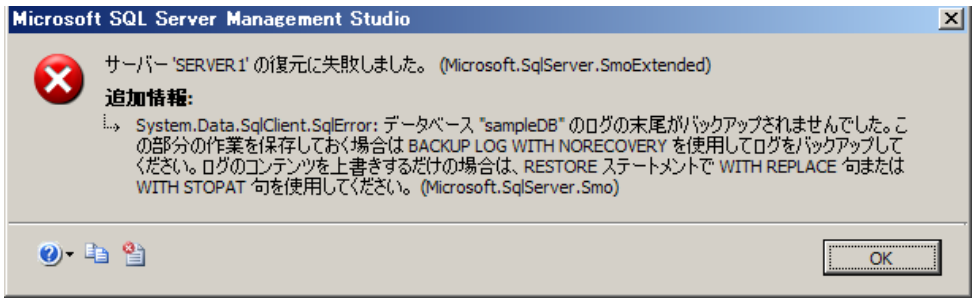


これにより、リストアが開始されますが、数秒後に、次のエラーが発生します。

「データベースは使用中なので、排他アクセスを獲得できませんでした」とエラーが表示され、リストアに失敗してします。エラー メッセージのとおり、データベースへ接続しているユー

ザーが １人でもいる場合は、リストアに失敗します。したがって、「sampleDB」データベースへ接続しているクエリ エディタをすべて閉じる必要があります。

5. クエリ エディタをすべて閉じたら、もう一度、「**sampleDB**」データベースのリストアを同じように実行します。すると、今度は次のエラーが発生します。



エラー メッセージの中で、「**WITH REPLACE**」というキーワードがあることに注目してください。これは、Replace（置換する）という言葉のとおり、既存のデータベースに対して、上書き復元するためのオプションです。デフォルトでは、簡単にデータベースが上書き復元されないように配慮されています。

そのほかのエラー内のキーワード「**ログ末尾**」や「**WITH NORECOVERY**」、「**WITH STOPAT**」などについては、後述します。

Note： 「排他アクセスを獲得できませんでした」エラーが引き続き表示される場合

もし、「排他アクセスを獲得できませんでした」エラーが引き続き表示される場合は、SQL Server の内部的なプロセスがデータベースへ接続している可能性があります。この場合は、ユーザーが一切接続していないことを確認してから、次のように **ALTER DATABASE** ステートメントを実行して、シングル ユーザー モードへ設定し、データベースへの接続をすべて強制終了します。

```
-- データベース接続を強制終了する場合
USE master
ALTER DATABASE sampleDB SET SINGLE_USER WITH ROLLBACK IMMEDIATE
```

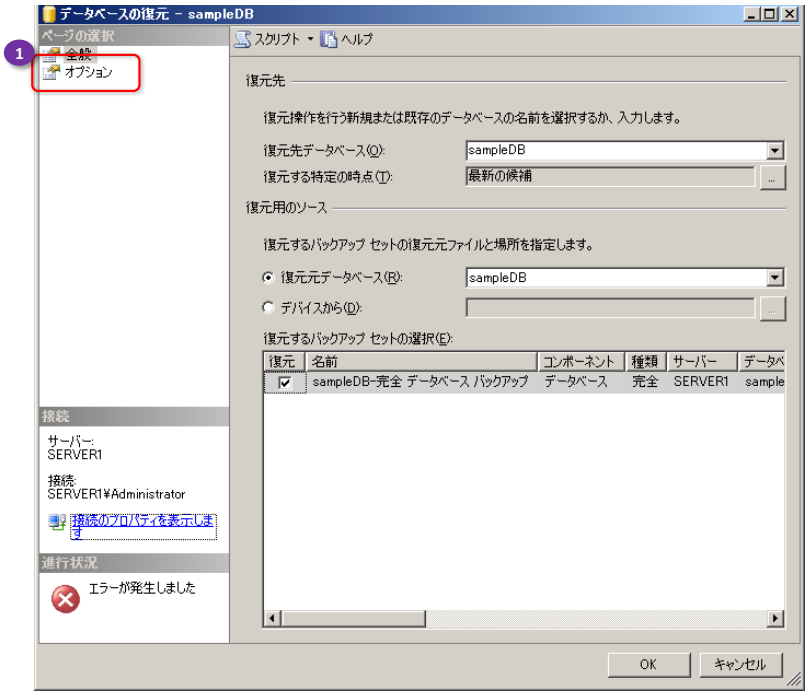
```
メッセージ
限定されないトランザクションをロールバック中です。推定ロールバック完了率: 100%。
```

なお、このモードへ設定した場合は、データベースのリストアが完了した後に、通常のマルチ ユーザー モードへ戻しておく必要があります。この場合は、次のように実行します。

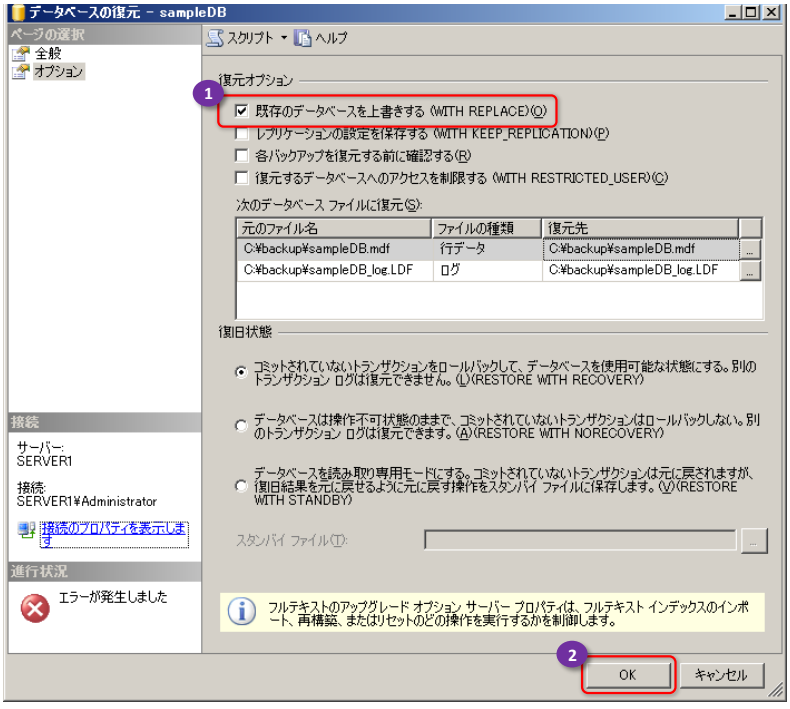
```
-- リストア終了後、もとへ戻す
ALTER DATABASE sampleDB SET MULTI_USER
```

```
メッセージ
コマンドは正常に完了しました。
```

6. エラーのダイアログを閉じた後、［データベースの復元］ダイアログへ戻ったら、次のように［**オプション**］をクリックします。

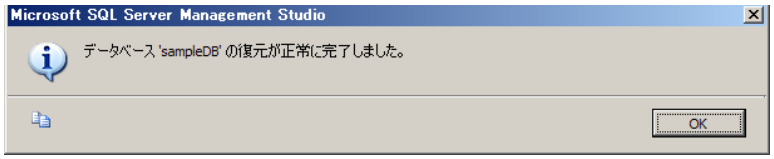


7. ［オプション］ページでは、次のように［**復元オプション**］で［**既存のデータベースを上書きする**］をチェックします。このオプションを設定することで、上述のエラーにあった「WITH REPLACE」による上書き復元が可能になります。



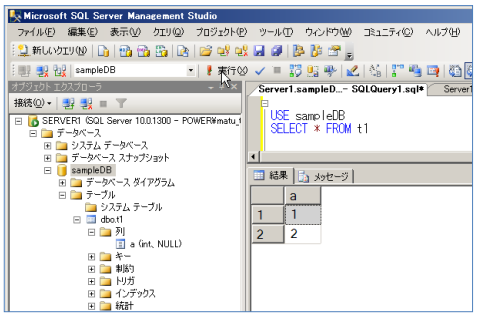
オプション設定後、［OK］ボタンをクリックすると、リストアが開始されます。

リストアが完了すると、次のダイアログが表示されます。



8. リストアが完了したら、クエリ エディタを開いて、データが復元されたかどうかを確認してみましょう。

```
USE sampleDB
SELECT * FROM t1
```



削除したデータが復活して、完全バックアップを取得したときの状態へ復元できたことを確認できます。このように、オンライン バックアップは、SQL Server を起動したままの状態で実行することができ、非常に手軽にバックアップと復元を行うことができます。

3.3　BACKUP / RESTORE ステートメントによるバックアップと復元

BACKUP ステートメントによる完全バックアップ

GUI 操作ではなく、SQL ステートメントを使用してオンライン バックアップを実行する場合には、「**BACKUP DATABASE**」ステートメントを利用します。構文は、次のとおりです。

```
BACKUP DATABASE データベース名
 TO { DISK | TAPE } = 'パス' [ WITH オプション ]
```

ディスクへバックアップする場合は「**TO DISK=**」に続けて、バックアップ先のファイル パスを記述し、テープ装置へバックアップする場合は「**TO TAPE=**」に続けて、テープへのパスを記述します。WITH で指定できるオプションについては、後述します。

RESTORE ステートメントによるリストア

オンライン バックアップからリストア（復元）するには、「**RESTORE DATABASE**」ステートメントを利用します。構文は、次のとおりです。

```
RESTORE DATABASE データベース名
 FROM { DISK | TAPE } = 'ファイルパス' [ WITH オプション ]
```

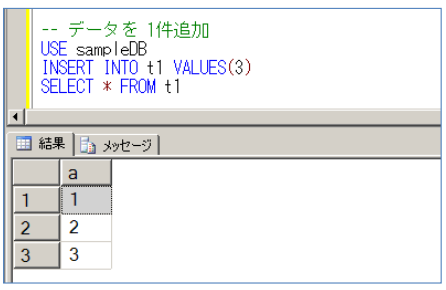
BACKUP との違いは、「**TO**」の部分が「**FROM**」へ変わるだけです。また、WITH で指定できるオプションについては後述しますが、前の手順で試した既存のデータベースを上書きするためのオプションは「**REPLACE**」です。

Let's Try

それでは、BACKUP と RESTORE ステートメントを試してみましょう。

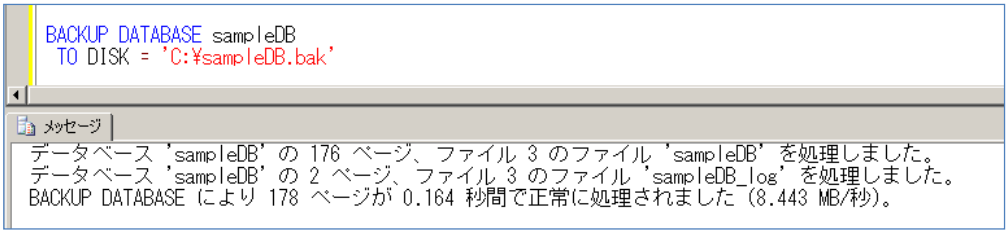
1. まずは、バックアップの効果を試すために、「**t1**」テーブルへデータを 1 件追加しましょう。

```
USE sampleDB
INSERT INTO t1 VALUES(3)
SELECT * FROM t1
```



2. 次に、**BACKUP** ステートメントを利用して完全バックアップを実行してみましょう。次のようにパスを指定して、**C:** ドライブの直下へ「**sampleDB.bak**」という名前でバックアップを取得します。

```
BACKUP DATABASE sampleDB
 TO DISK = 'C:¥sampleDB.bak'
```



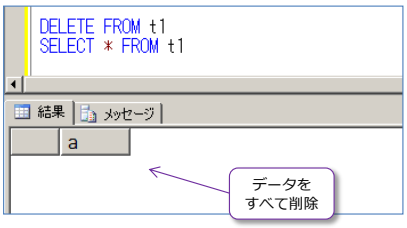
「正常に処理されました」と表示されれば、バックアップが完了しています。

Note： サービス アカウントに対して NTFS アクセス許可が必要

バックアップは、内部的には、SQL Server のサービス アカウントが実行者になるので、バックアップ先のフォルダに対して、SQL Server のサービス アカウントへの NTFS アクセス許可が与えられていないと、バックアップがエラーになって失敗します。したがって、C: ドライブの直下へバックアップを取得したい場合は、サービス アカウントに対して、「**変更**」NTFS 許可を付与しておく必要があります。

3. バックアップが完了したら、次に、**DELETE** ステートメントを実行して、すべてのデータを削除します。

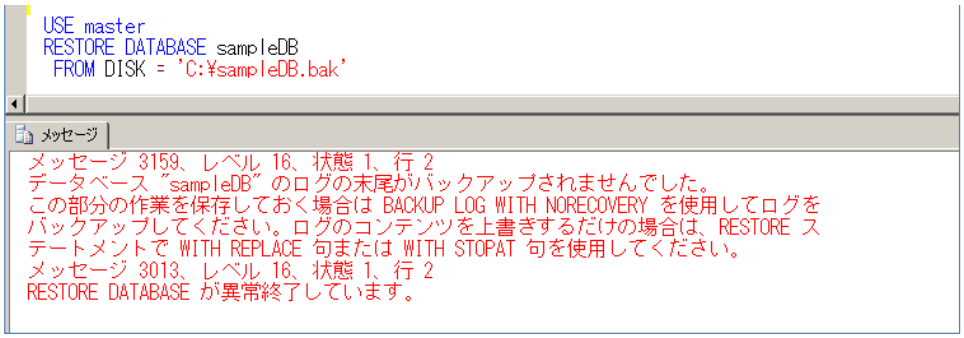
```
DELETE FROM t1
SELECT * FROM t1
```



RESTORE ステートメントによるリストア

4. データの削除後、**RESTORE** ステートメントを実行して、バックアップをリストアしてみましょう。

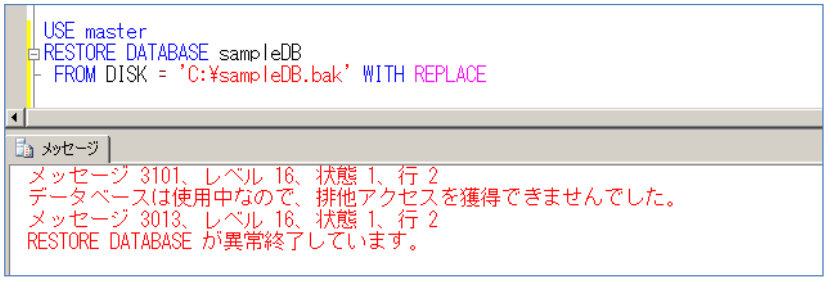
```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥sampleDB.bak'
```



前の Step で試したときと同じように、「**WITH REPLACE** オプションを指定しないと上書き復元できない」という主旨のエラーが表示されます。

5. 次に、「**WITH REPLACE**」オプションを指定して RESTORE ステートメントを実行します。

```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥sampleDB.bak' WITH REPLACE
```

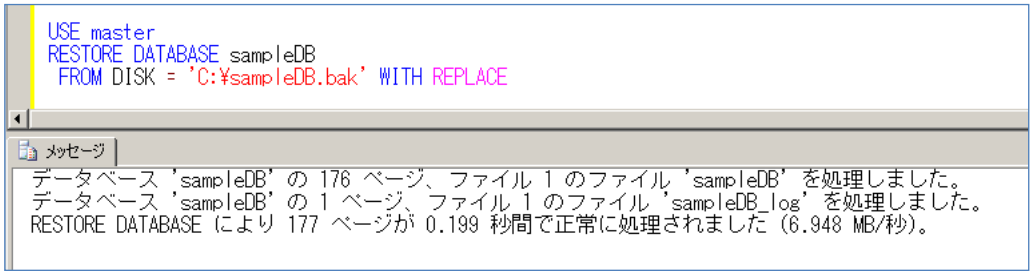


今度は、数秒後に「データベースは使用中なので、排他アクセスを獲得できませんでした」とエラーが表示されます（エラーが表示されない場合は、そのまま手順 8 へ進んでください）。

6. このエラーを回避するには、一度クエリ エディタをすべて閉じます。

7. すべてのクエリ エディタを閉じたら、新しいクエリ エディタを開いて、もう一度 RESTORE ステートメントを実行します。

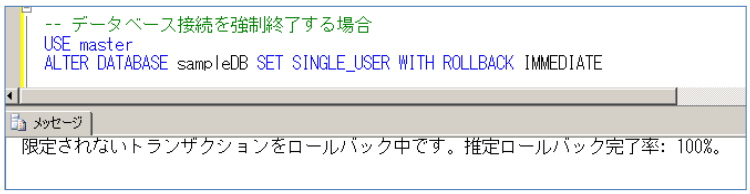
```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥sampleDB.bak' WITH REPLACE
```



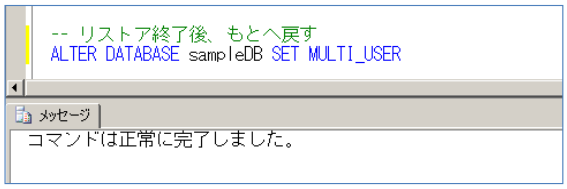
今後は、リストアが正しく実行されて、「正常に処理されました」と表示されることを確認できます。

Note： 「排他アクセスを獲得できませんでした」エラーが引き続き表示される場合

前述の Note にも記載しましたが、「排他アクセスを獲得できませんでした」エラーが引き続き表示される場合は、データベースに対してユーザーが一切接続していないことを確認してから、次のように **ALTER DATABASE** ステートメントを実行して、シングル ユーザー モードへ設定し、データベースへの接続をすべて強制終了します。

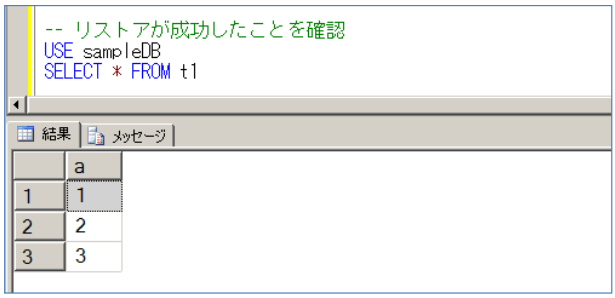


また、このモードへ設定した場合は、データベースのリストアが完了した後に、通常のマルチ ユーザー モードへ戻す必要があります。この場合は、次のように実行します。



8. リストアが完了したら、データが正しく復元されたことを確認してみましょう。

```
USE sampleDB
SELECT * FROM t1
```



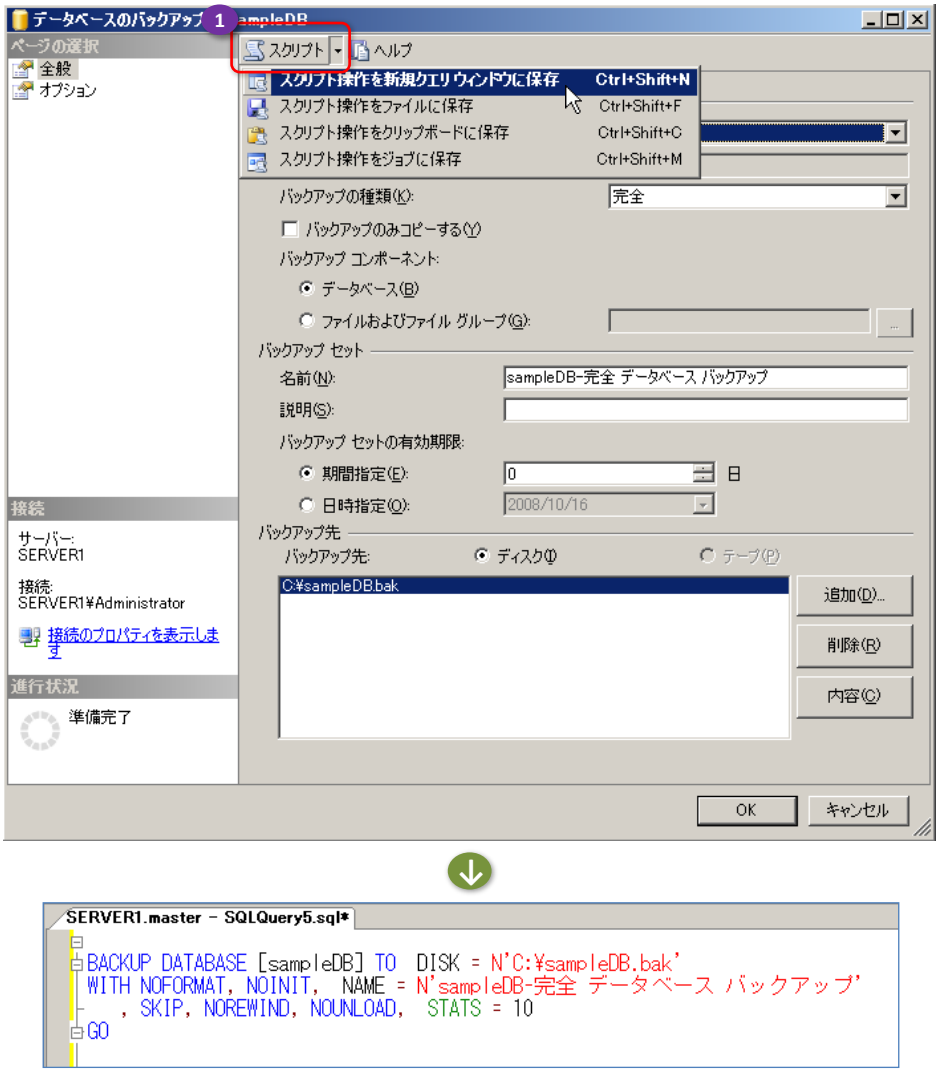
削除したデータが復活して、BACKUP ステートメントで完全バックアップを取得したときの状態へ復元できたことを確認できます。

このように、オンライン バックアップは、GUI を利用しなくても BACKUP ステートメントを使用して簡単に取得することができ、また RESTORE ステートメントによる復元も簡単に実行できます。これらのステートメントは、次の項で説明する「**定期バックアップ**」には欠かせない機能になるので、ぜひ覚えておくことをお勧めします。

3.4　スクリプト（BACKUP / RESTORE）の生成機能

スクリプト（BACKUP / RESTORE）の生成機能

GUI 操作でのバックアップ画面の [データベースのバックアップ] および [データベースの復元] ダイアログでは、次のようにツールバーの [**スクリプト**] ボタンをクリックすることで、BACKUP ステートメントおよび RESTORE ステートメントを生成できる機能があります。

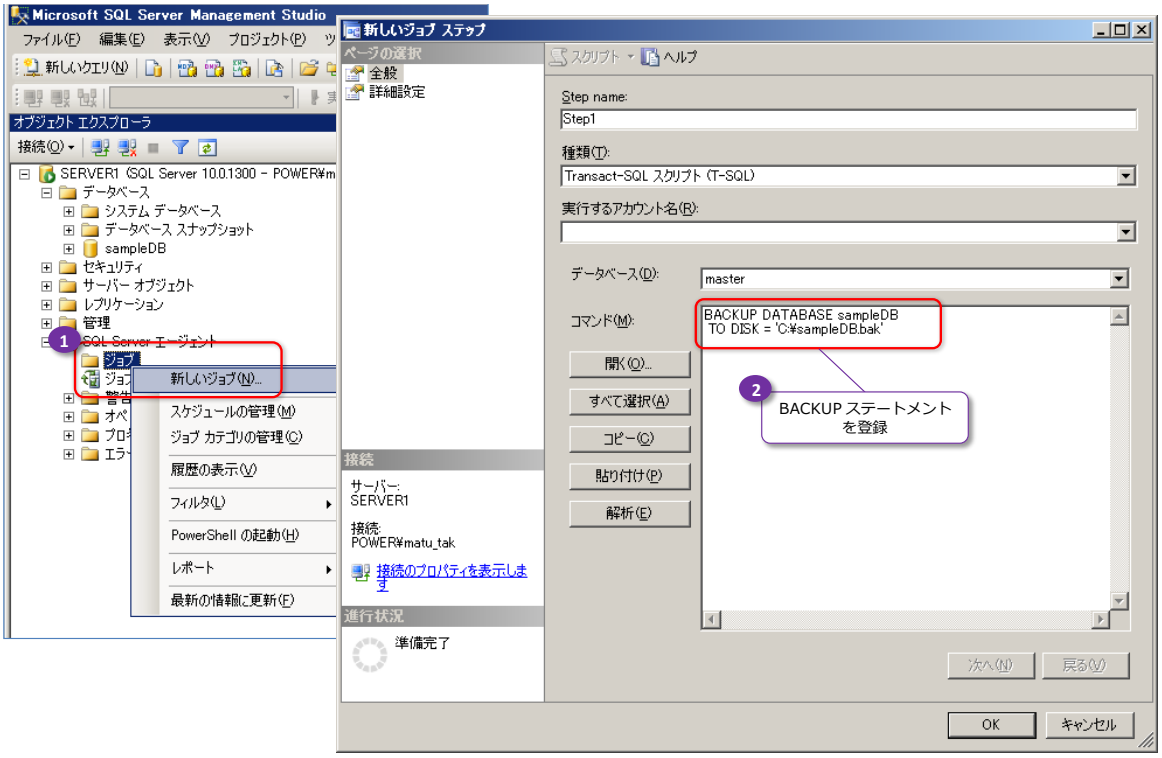


これを利用すると、GUI 操作で設定したオプションをそのまま生成できるので、大変便利です。

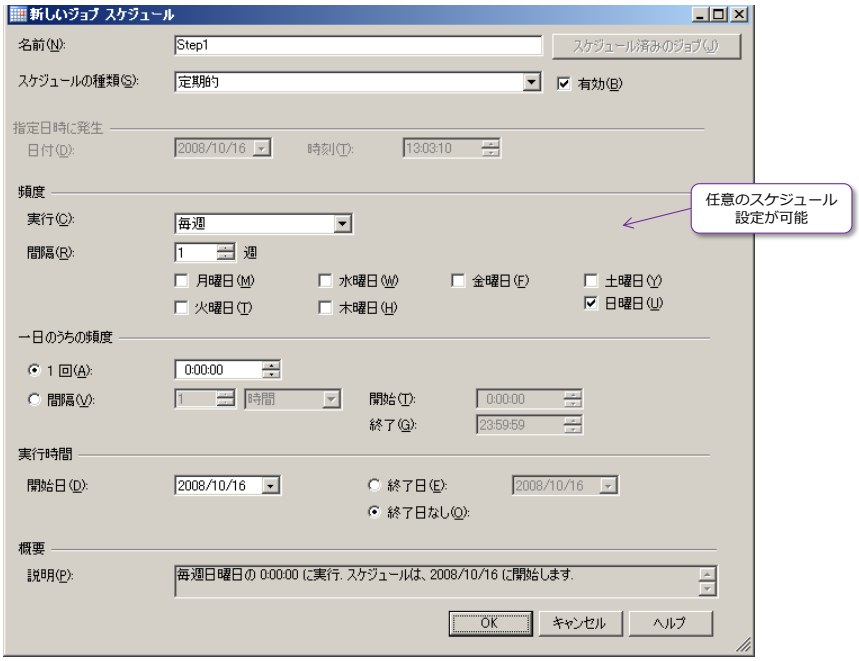
3.5　スケジュール設定によるバックアップの定期実行（ジョブ機能）

スケジュール設定による定期実行（SQL Server Agent ジョブ）

ここでは、バックアップを定期実行する方法について説明します。バックアップを定期実行するには、SQL Server Agent サービスの「**ジョブ**」機能（スケジュール実行機能）を利用して、BACKUP ステートメントを登録します。



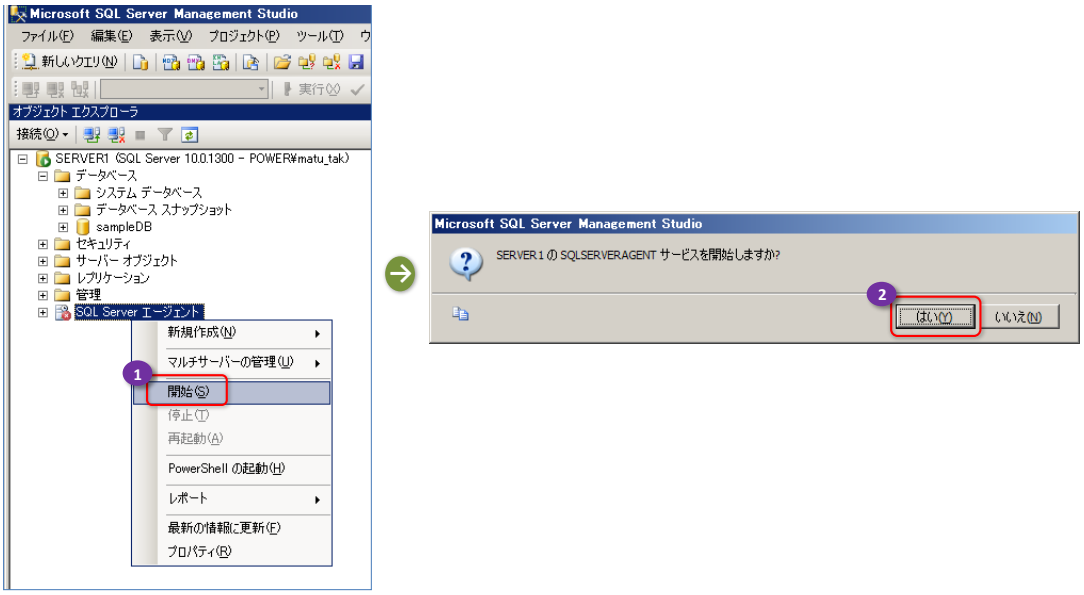
ジョブ機能では、次のように任意のスケジュール設定が可能です。



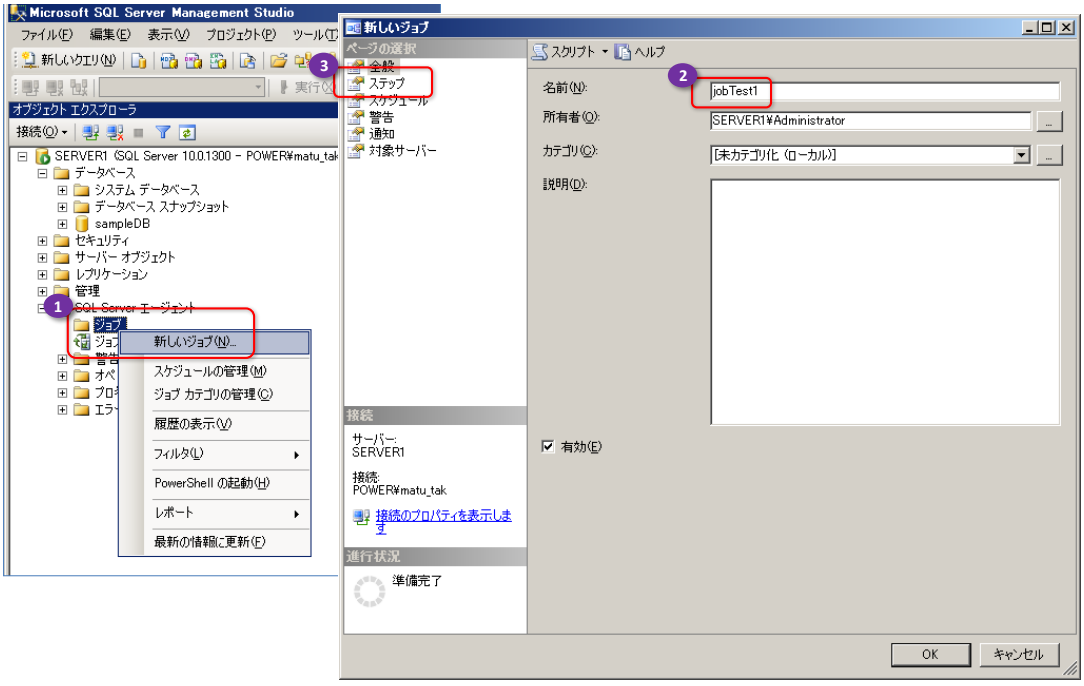
Let's Try

それでは、これを試してみましょう。

1. まずは、ジョブ機能を利用するために、**SQL Server Agent サービス**を開始します。SQL Server Agent サービスを開始するには、オブジェクト エクスプローラで［**SQL Server エージェント**］を右クリックして［**開始**］をクリックします。

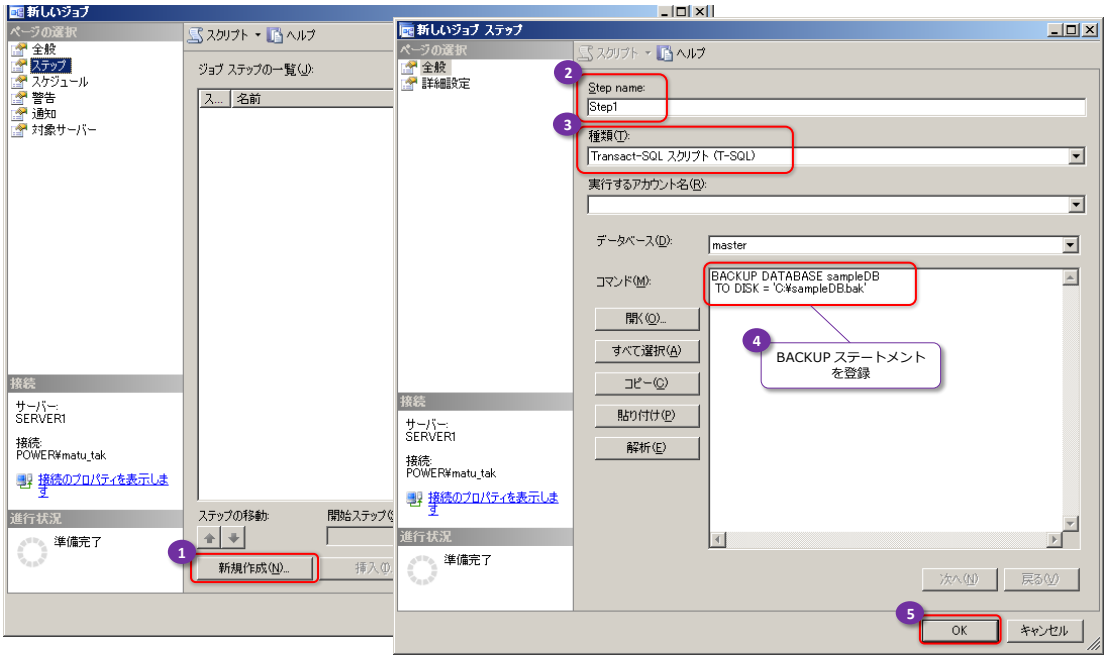


2. サービスが開始されたら、新しいジョブを作成します。ジョブを作成するには、次のように［SQL Server エージェント］を展開して、［**ジョブ**］フォルダを右クリックして［**新しいジョブ**］をクリックします。



［新しいジョブ］ダイアログが表示されたら、［名前］へ任意のジョブ名（**jobTest1** など）を入力して、［ページの選択］で［**ステップ**］をクリックします。

3. ［ステップ］ページが開いたら、［**新規作成**］ボタンをクリックして、新しいステップを作成します。

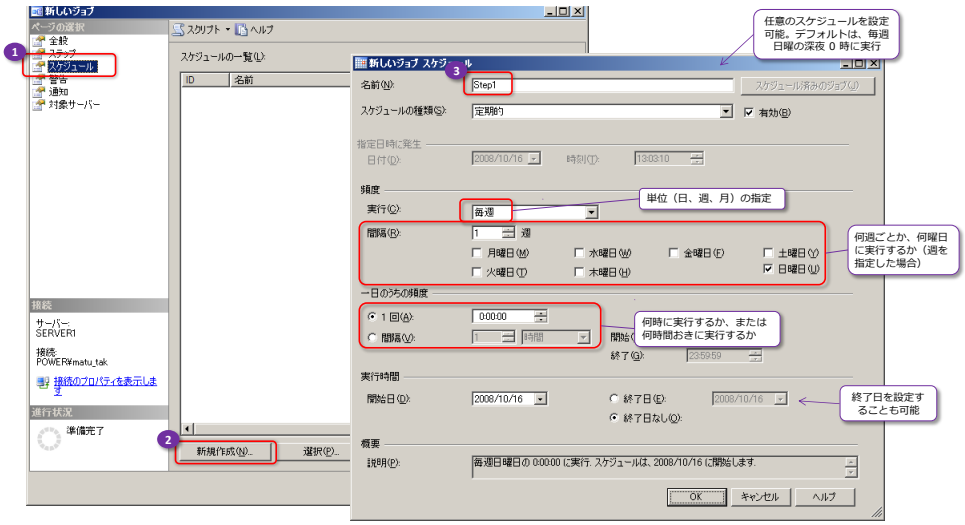


［新しいジョブ ステップ］ダイアログでは、任意のステップ名（**Step1** など）を入力して、［**種類**］で「**Transact-SQL スクリプト（T-SQL）**」が選択されていることを確認し、［**コマンド**］へ前の Step で実行した **BACKUP** ステートメントを記述します。

```
BACKUP DATABASE sampleDB
 TO DISK = 'C:¥sampleDB.bak'
```

ステートメントを記述後、［OK］ボタンをクリックして、ダイアログを閉じます。

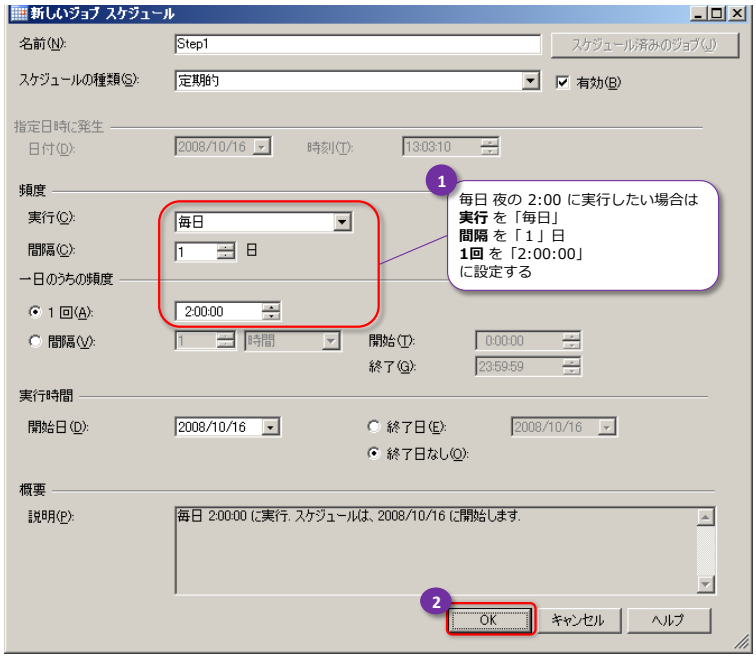
4. ［新しいジョブ］ダイアログへ戻ったら、［**スケジュール**］ページを開きます。



［スケジュール］ページでは、［**新規作成**］ボタンをクリックして、新しいスケジュールを作成します。［新しいジョブ スケジュール］ダイアログが表示されたら、任意のスケジュール名（**sche1** など）を入力して、［**実行**］や［**間隔**］、［**曜日**］などを任意に設定します。デフォル

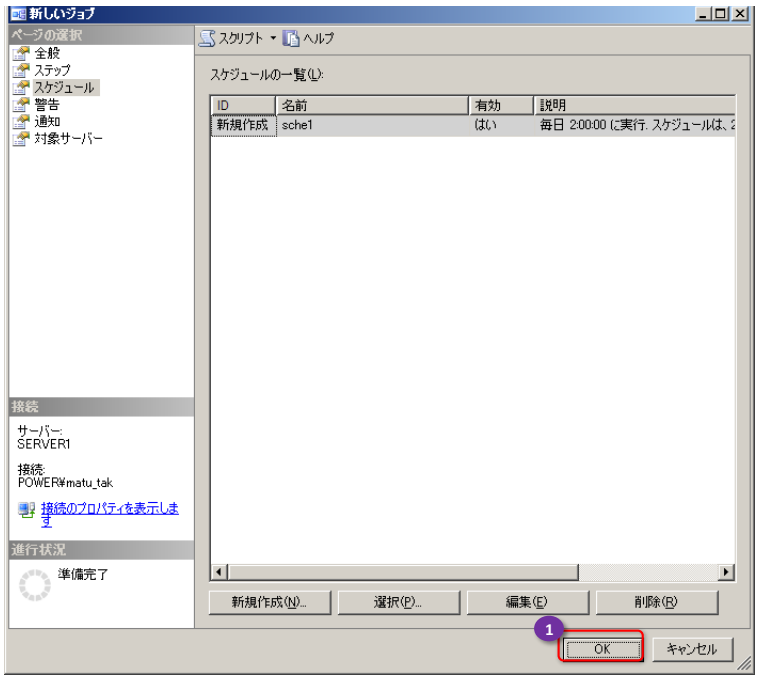
トでは、毎週日曜の深夜 0:00 に実行するようにスケジュールが設定されます。

スケジュールは、次のように設定すると、毎日深夜 2:00 に実行できるようになります。



スケジュールは、任意に設定して、[OK] ボタンをクリックして、ダイアログを閉じます。

5. [新しいジョブ] ダイアログへ戻ったら、[OK] ボタンをクリックします。

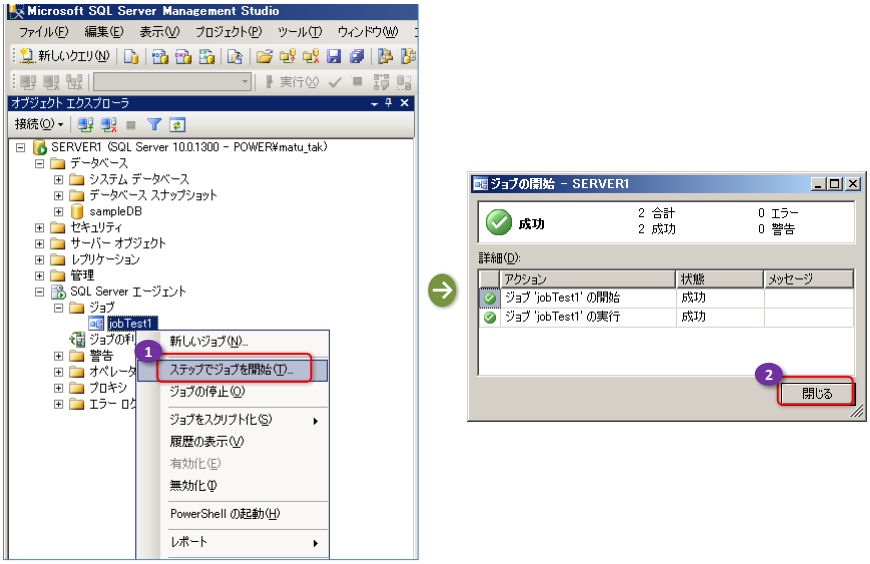


以上で、定期バックアップを実行するためのジョブが完成しました。あとは、SQL Server Agent サービスが起動していれば、設定したスケジュールのタイミングで、バックアップが実行されるようになります。

ジョブの手動実行

次に、ジョブを手動実行して、作成したジョブが正しく動作することを確認してみましょう。

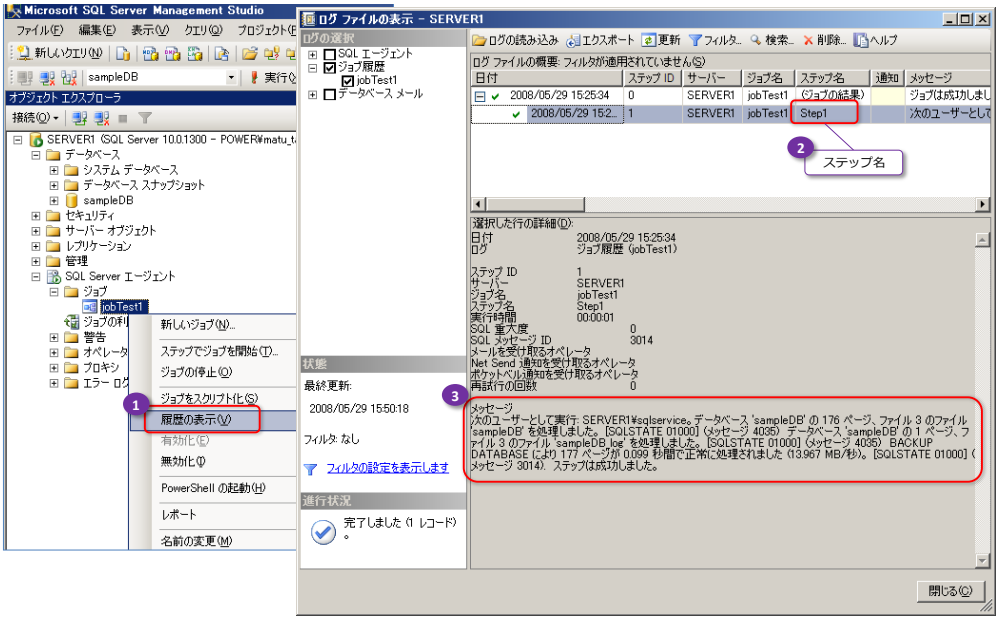
6. ジョブを手動実行するには、次のように作成したジョブ「**jobTest1**」を右クリックして、[**ステップでジョブを開始**] をクリックします。



[状態] がすべて [**成功**] と表示されれば、ジョブの実行が成功しています。これで、ステップへ登録した完全バックアップの実行が完了しています。

もし、ジョブが失敗する場合は、サービス アカウントに対して、バックアップ先フォルダへの「変更」NTFS アクセス許可が付与されているかどうかを確認してみてください。

7. 次に、ジョブの実行履歴を確認してみましょう。ジョブの実行履歴を確認するには、ジョブを右クリックして、[**履歴の表示**] をクリックします。



ステップ名が表示されて、バックアップが正常に処理されていることを確認できます。

3.6　バックアップ セットとは

バックアップ セットとは

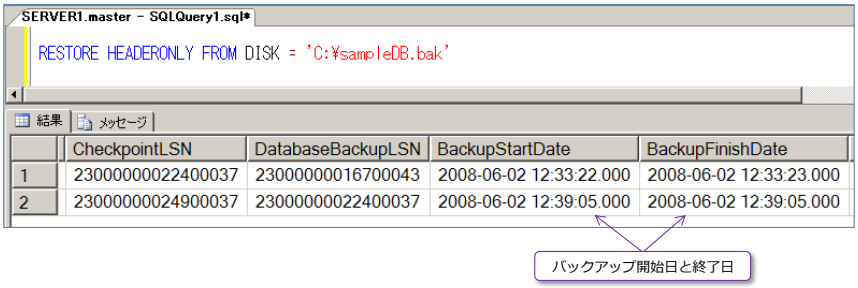
1 回のバックアップでバックアップされるデータは、正確には「**バックアップ セット**」と呼ばれます。前の Step では、「**C:¥sampleDB.bak**」ファイルへバックアップしましたが、このファイルへは、その前の前の Step でもバックアップをしています。したがって、1 つのファイル内へ複数のバックアップが取得されている状態になっています。

C:¥sampleDB.bak ファイル内

バックアップ セット #1 Step 3.3 で BACKUP ステートメントを実行して取得したもの	バックアップ セット #2 Step 3.5 でジョブの手動実行で取得したもの

バックアップ ファイル内へ複数のバックアップ セットがあることを確認するには、次のように **RESTORE HEADERONLY** ステートメントを利用します。

```
RESTORE HEADERONLY FROM DISK = 'C:¥sampleDB.bak'
```



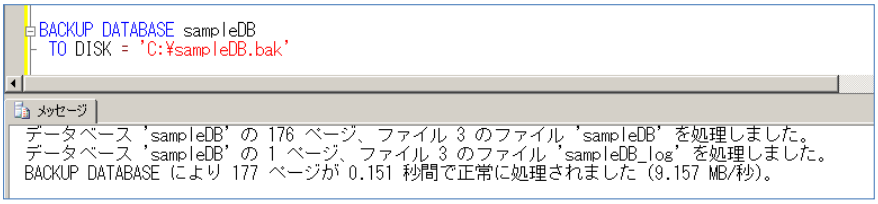
ステートメントの実行結果を右へスクロールすると、BackupStartDate と BackupFinishDate が表示されて、バックアップの開始日と終了日を確認できます。また、結果が 2 件であることから、このファイル内に 2 つのバックアップ セットが含まれていることを確認できます。

Let's Try

それでは、これを試してみましょう。

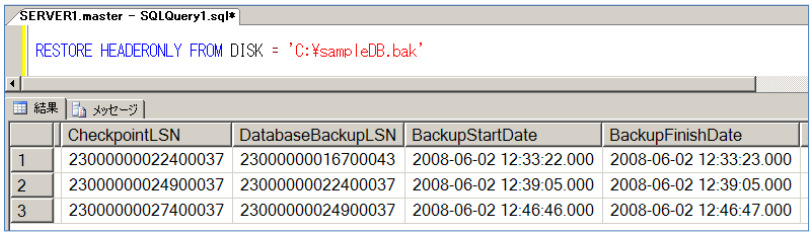
1. まずは、クエリ エディタから、もう一度「**C:¥sampleDB.bak**」ファイルへバックアップを取得してみましょう。

```
BACKUP DATABASE sampleDB
 TO DISK = 'C:¥sampleDB.bak'
```



2. 次に、**RESTORE HEADERONLY** ステートメントを実行して、バックアップ ファイル内のバックアップ セット数が増えていることを確認しましょう。

```
RESTORE HEADERONLY FROM DISK = 'C:¥sampleDB.bak'
```



	CheckpointLSN	DatabaseBackupLSN	BackupStartDate	BackupFinishDate
1	23000000022400037	23000000016700043	2008-06-02 12:33:22.000	2008-06-02 12:33:23.000
2	23000000024900037	23000000022400037	2008-06-02 12:39:05.000	2008-06-02 12:39:05.000
3	23000000027400037	23000000024900037	2008-06-02 12:46:46.000	2008-06-02 12:46:47.000

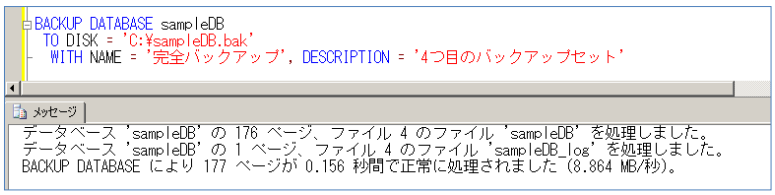
結果が 3 件に増えて、同じファイル内へバックアップが取得されたことを確認できます。

NAME、DESCRIPTION オプション

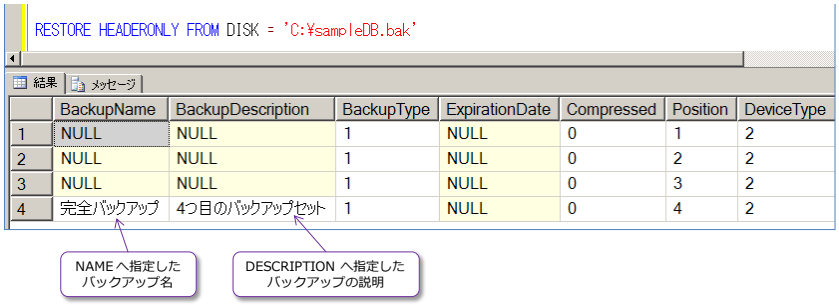
BACKUP ステートメントでは、WITH のオプションで **NAME**（バックアップ名）と **DESCRIPTION**（説明）を追加することができます。これを設定しておくと、同じファイル内へ複数のバックアップを取得する場合に、見分けが付きやすくなります。では、これを試してみましょう。

3. 今度は、**NAME** と **DESCRIPTION** を指定して、もう一度「**C:¥sampleDB.bak**」ファイルへバックアップを取得してみましょう。

```
BACKUP DATABASE sampleDB
 TO DISK = 'C:¥sampleDB.bak'
  WITH NAME = '完全バックアップ', DESCRIPTION = '4つ目のバックアップセット'
```



4. 次に、**RESTORE HEADERONLY** ステートメントを実行して、バックアップ ファイル内のバックアップ セットを確認してみます。



	BackupName	BackupDescription	BackupType	ExpirationDate	Compressed	Position	DeviceType
1	NULL	NULL	1	NULL	0	1	2
2	NULL	NULL	1	NULL	0	2	2
3	NULL	NULL	1	NULL	0	3	2
4	完全バックアップ	4つ目のバックアップセット	1	NULL	0	4	2

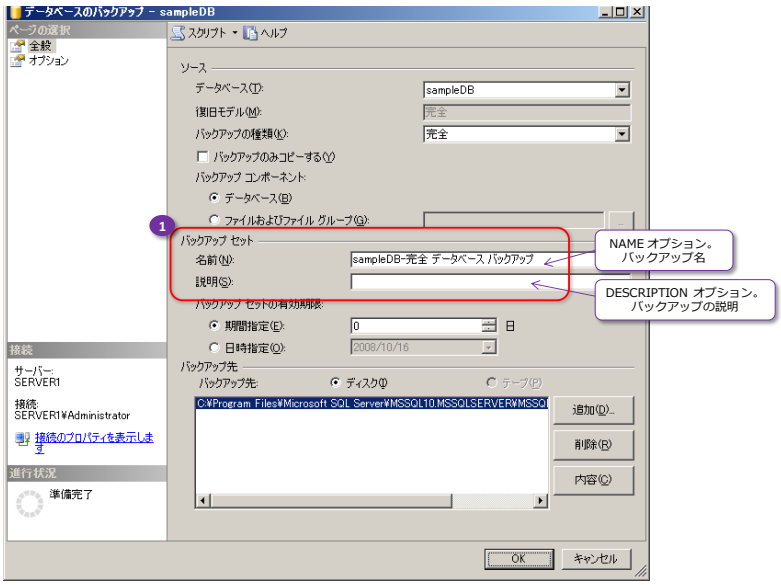
NAME へ指定したバックアップ名

DESCRIPTION へ指定したバックアップの説明

結果の **BackupName** と **BackupDescription** 列へ、オプションで指定したバックアップ名と説明が取得できることを確認できます。このように、NAME と DESCRIPTION を指定

してバックアップを実行すると、複数のバックアップを同じファイル内へ取得する場合に見分けがつけやすくなります。

なお、GUI 操作でバックアップの名前と説明を設定する場合は、[データベースのバックアップ] ダイアログで、次のように設定します。

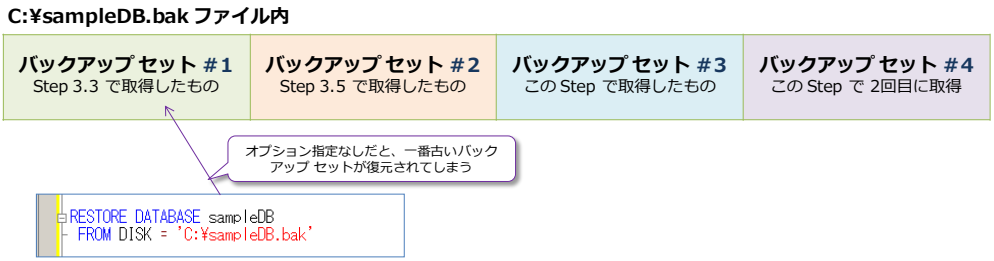


複数バックアップ セットがある場合のリストア（FILE オプション）

同じファイル内に複数のバックアップ セットがある場合は、リストア時に注意する必要があります。デフォルトでは、1 つ目（一番古いバックアップ セット）が復元されてしまうからです。たとえば、次のように RESTORE を実行したとします。

```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥sampleDB.bak'
```

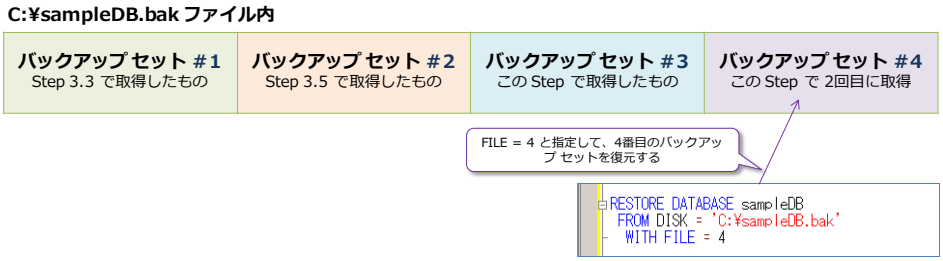
この場合は、「**C:¥sampleDB.bak**」ファイル内の一番古いバックアップ セット（Step 3.3 で取得したもの）が復元されてしまいます。



RESTORE 時にどのバックアップ セットを復元するかは、次のように **FILE** オプションを指定します。

```
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥sampleDB.bak' WITH FILE = n
```

n には、バックアップ セット番号（ファイル内の何個目のバックアップ セットか）を指定します。したがって、4 つ目の一番最新のバックアップ セット（この Step で取得したバックアップ）を復元する場合には、n へ 4 を指定します。

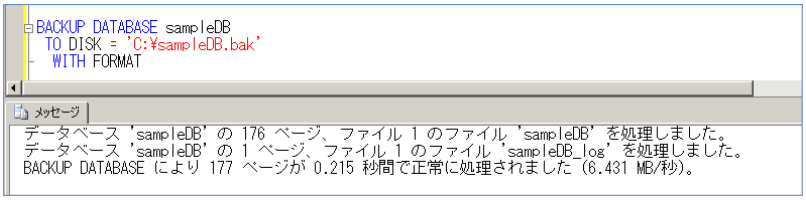


このように同じファイル内へ複数のバックアップを取得している場合は、リストア時に注意するようにしましょう。

FORMAT オプションでバックアップ ファイルのフォーマット

BACKUP ステートメントでは、**FORMAT** オプションを指定してバックアップを実行すると、バックアップ ファイルをフォーマットすることができます。これは、次のように利用します。

```
BACKUP DATABASE sampleDB
 TO DISK = 'C:¥sampleDB.bak' WITH FORMAT
```



これにより、ファイルがフォーマットされて、ファイル内には、1 つのバックアップ セット（今、取得したバックアップ）のみが含まれるようになります。

Note： 1 つのファイルには、1 つのバックアップ セットを格納することをお勧め

1 つのファイル内へ複数のバックアップ セットを格納する方法は、リストア時に誤って違うバックアップ セットを復元してしまう可能性があります。したがって、1 つのファイルには、1 つのバックアップ セットを格納したほうが管理がしやすく、間違いも起こりにくいので、このようにバックアップを取得していくことをお勧めします。この際には、次の Step で説明する日付と時刻をファイル名へ入れるようにすると、さらに管理がしやすくなるのでお勧めです。

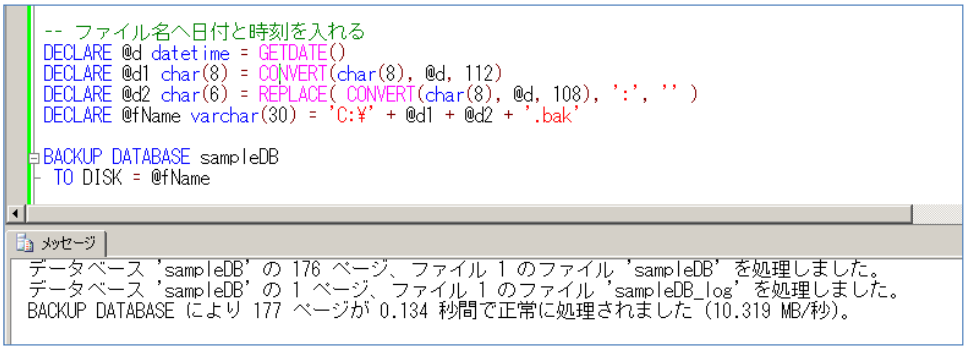
3.7　ファイル名へ日付／時刻を入れる方法

ファイル名へ日付／時刻を入れる方法

バックアップ ファイル名には、バックアップを実行したときの日付と時刻を入れておくようにすると、管理がしやすくなり、リストアの際にも、どのバックアップを戻せば良いのかが一目瞭然になるのでお勧めです。

バックアップ ファイル名へ日付と時刻を入れるには、次のように実行します。

```
DECLARE @d datetime = GETDATE()
DECLARE @d1 char(8) = CONVERT(char(8), @d, 112)
DECLARE @d2 char(6) = REPLACE( CONVERT(char(8), @d, 108), ':', '' )
DECLARE @fName varchar(30) = 'C:¥' + @d1 + @d2 + '.bak'

BACKUP DATABASE sampleDB
 TO DISK = @fName
```

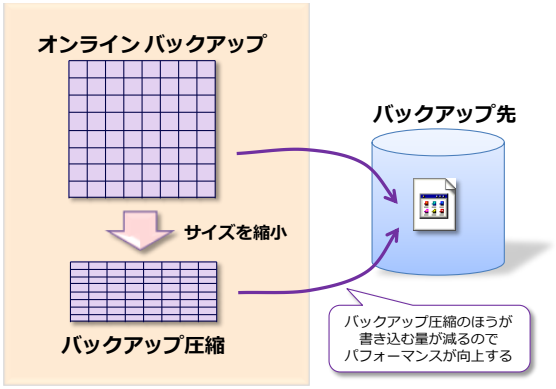


これにより、現在時刻が「**2008 年 5 月 3 日 18 時 08 分 46 秒**」なら 「**C:¥20080503180846.bak**」というファイル名にすることができます。**CONVERT** 関数を利用して、日付と時刻を文字列へ変換している部分の、第 3 引数へ指定した「**112**」や「**108**」については、Transact-SQL リファレンスの「CAST および CONVERT」を参考にしてください。「**112**」は yymmdd 形式、「**108**」は hh:mm:ss 形式の結果を返します（間のコロン : は REPLACE 関数で取り除いています）。

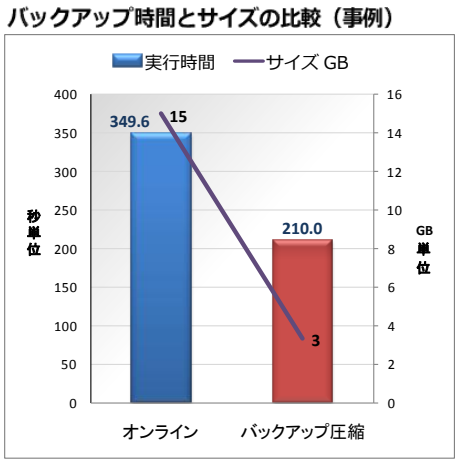
3.8　バックアップ圧縮（Backup Compression）

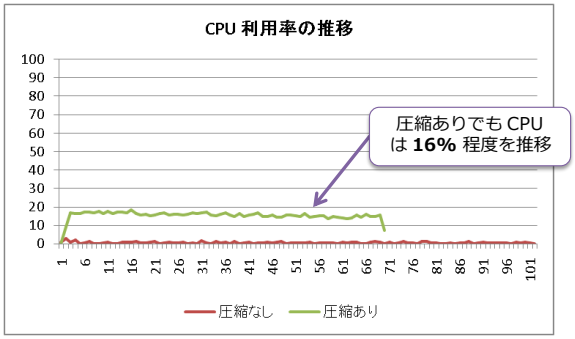
バックアップ圧縮

バックアップ圧縮は、SQL Server 2008 から提供された新機能です。その名のとおり、バックアップ データを圧縮できる機能で、これによりバックアップとリストアのパフォーマンスを向上させることができます。圧縮することによって、バックアップおよびリストア時のデバイス（テープ装置やディスクなどのバックアップ先デバイス）への書き込み / 読み取り量（I/O）を減らすことができるので、パフォーマンスが向上します。



バックアップにかかる時間は、デバイス（テープやディスク）の書き込み速度に大きく依存するので、書き込む量を減らせると、バックアップ時間を短縮できます。次のグラフは、筆者のお客様のデータ（45GB のデータベース）でオンライン バックアップを実行した場合と、バックアップ圧縮を実行した場合での、実行時間とサイズ、CPU 使用時間を比較したものです。





オンライン バックアップでは **15GB** 分のファイル コピーが必要なのに対して、バックアップ圧縮では **3GB** 分（4 分の 1 以下）のバックアップで済んでいるので、バックアップ時間を **5 分 50 秒**から **3 分 30 秒**に短縮（2 分 20 秒の短縮。40%のパフォーマンス UP）することができています。

また、このときの CPU 利用率は、圧縮なしではほぼゼロ近辺を推移しているのに対し、圧縮ありでは、16%前後を推移していました。圧縮は、デバイスへの書き込みと CPU 利用率のトレード オフになりますが、夜間など CPU リソースが十分に余っている時間帯では、積極的にバックアップ

圧縮を利用することをお勧めします。

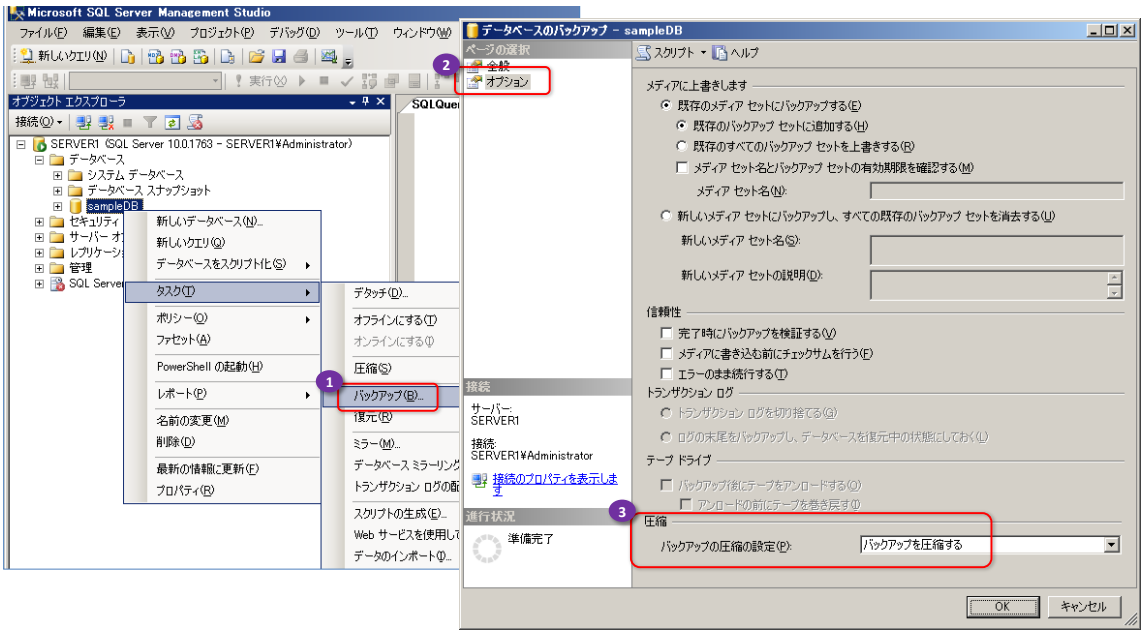
COMPRESSION オプション

バックアップ圧縮を実行するための構文は、次のとおりです。

```
BACKUP DATABASE sampleDB
 TO { DISK | TAPE } = 'パス'
   WITH COMPRESSION
```

COMPRESSION オプションを指定するだけで、バックアップ圧縮を実行することができます。

BACKUP ステートメントではなく、GUI 操作でバックアップ圧縮を実行する場合は、次のように操作します。



［データベースのバックアップ］ダイアログで、［**オプション**］ページを開き、［**バックアップの圧縮の設定**］で「**バックアップを圧縮する**」を選択すれば、バックアップ圧縮を実行できるようになります。

Let's Try

それでは、バックアップ圧縮を試してみましょう。

1. sampleDB データベースに対して、次のようにバックアップ圧縮を実行してみましょう。

```
-- バックアップ圧縮
BACKUP DATABASE sampleDB
 TO DISK = 'C:¥sampleDBcomp.bak'
   WITH COMPRESSION
```

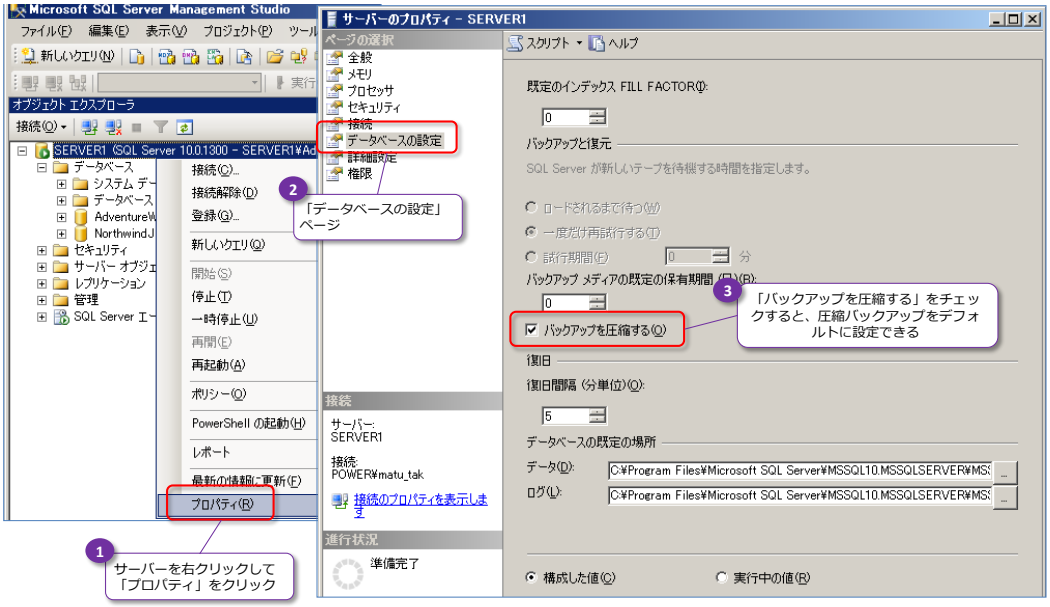
```
-- バックアップ圧縮
BACKUP DATABASE sampleDB
  TO DISK = 'C:¥sampleDBcomp.bak'
    WITH COMPRESSION
```

```
メッセージ
データベース 'sampleDB' の 176 ページ、ファイル 1 のファイル 'sampleDB' を処理しました。
データベース 'sampleDB' の 1 ページ、ファイル 1 のファイル 'sampleDB_log' を処理しました。
BACKUP DATABASE により 177 ページが 0.079 秒間で正常に処理されました (17.503 MB/秒)。
```

このように、通常の BACKUP ステートメントに WITH COMPRESSION を付けるだけで、バックアップ ファイルを圧縮できるようになります。

サーバーの設定としてバックアップ圧縮を有効化する

WITH COMPRESSION や GUI での圧縮の指定を行わなくても、バックアップ圧縮を実行させるようにすることもできます(デフォルトをバックアップ圧縮に設定することができます)。これは、サーバー設定を次のように変更します。



［**データベースの設定**］ページで、［**バックアップを圧縮する**］をチェックすれば、デフォルトをバックアップ圧縮へ設定することができます。

バックアップ圧縮ファイルからのリストア

バックアップ圧縮した場合のリストアは、通常のリストアとまったく同じように実行できます。

```
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥sampleDBcomp.bak'
```

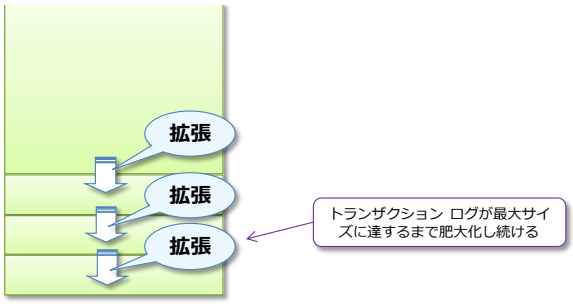
Note： リストアのパフォーマンスも向上

バックアップ圧縮したファイルからのリストアは、通常のバックアップからリストアするよりもパフォーマンスが向上します。これは、デバイスからの読み取り量（Read I/O）が削減されるためです。

3.9　トランザクション ログの管理

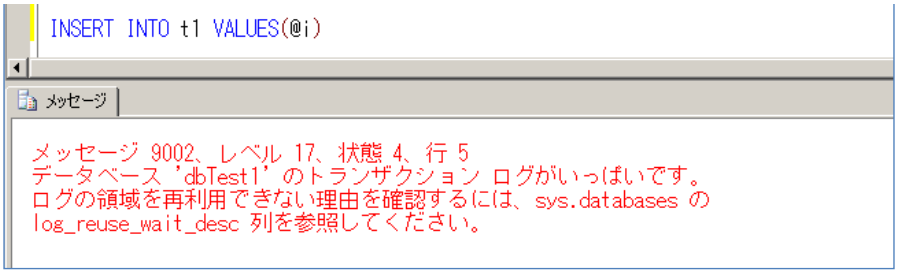
トランザクション ログの肥大化

SQL Server では、デフォルトでは、トランザクション ログは、ハード ディスクの空き容量がなくなるまで、またはファイルの最大サイズに達するまで、どんどん肥大化し続けます。



9002： ログが満杯の場合のエラー

ディスクの空き領域がなくなって、トランザクション ログが満杯になった場合は、エラー **9002** が発生します。



このエラーは、満杯になった後のトランザクションの実行時に発生し、そのトランザクションがエラーになることに注意してください。ログが満杯の状態では、それ以上トランザクションを実行することができません。

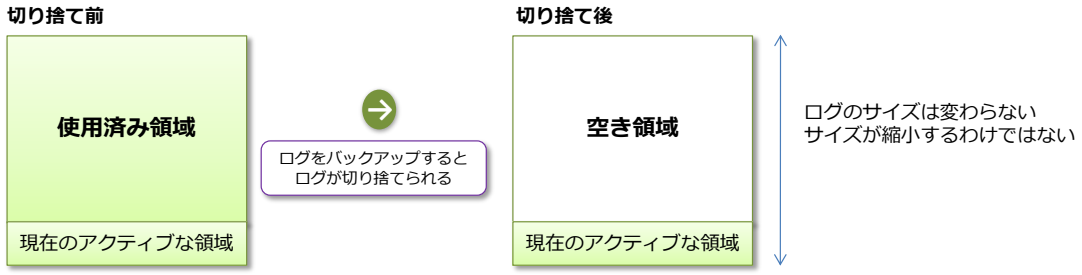
したがって、ログが満杯にならないようにトランザクション ログを管理しておくことは、SQL Server を利用する上で必須の作業になります。

ログの肥大化を防止するには

トランザクション ログの肥大化を防止するには、次の 2 つの方法があります。

1. **トランザクション ログを定期的にバックアップする**
2. **復旧モデルを「単純」にする**

1 つめのトランザクション ログのバックアップは、次の Step で説明しますが、このバックアップを実行すると、トランザクション ログの使用済み領域を切り捨てることができます。



このように、ログのバックアップを実行すると、使用済み領域を解放して、空き領域へ変換してくれるので、その領域が再利用できるようになります（肥大化を防止できます）。

2 つ目の復旧モデルを「**単純**」へ変更する方法は、Step 5 で説明しますが、障害発生直前までの復旧ができなくなるので、お勧めの方法ではありません。

したがって、ログの肥大化を防止するには、ログ バックアップを定期実行することが最善策になります。次の Step で説明するログのバックアップ手順を確実にマスターしておくことが重要です。

Note： ファイル サイズを縮小したい場合は DBCC SHRINKFILE コマンド

ログ バックアップ時に実行されるログの切り捨ては、再利用可能な空き領域を作るだけで、ファイル サイズを縮小するわけではありません。しかし、ディスク容量を圧迫するほど、大きく肥大化してしまったファイルの場合は、サイズを小さくしたい場合があります。この場合は、**DBCC SHRINKFILE** コマンドを次のように使用します。

```
DBCC SHRINKFILE(論理名, 縮小後のサイズ)
```

Note： 以前のバージョンの TRUNCATE_ONLY オプションは廃止

以前のバージョンでは、次のように TRUNCATE_ONLY オプションを利用すると、ログを切り捨てることができました。

```
BACKUP LOG データベース名 WITH TRUNCATE_ONLY
```

SQL Server 2008 では、このオプションは廃止されたので、実行することができなくなっています。SQL Server 2008 では、ログのバックアップを実行して、ログを切り捨てる必要があります。

ログの使用量を確認する： DBCC SQLPERF(LOGSPACE) コマンド

現在のトランザクション ログの使用量を確認するには、**DBCC SQLPERF(LOGSPACE)** コマンドを使用します。

```
-- トランザクション ログのサイズと使用率の確認
DBCC SQLPERF(LOGSPACE)
```

結果 | メッセージ

	Database Name	Log Size (MB)	Log Space Used (%)	Status
1	master	8.179688	36.81948	0
2	tempdb	0.4921875	50.59524	0
3	model	0.9921875	50	0
4	msdb	2.742188	38.46154	0
5	sampleDB	0.5390625	73.00725	0
6	dbTest1	3.5625	7.593202	0

ログのサイズ　ログの使用率

STEP 4. ログ バックアップと差分バックアップ

この STEP では、ログ バックアップと差分バックアップ機能について説明します。

この STEP では、次のことを学習します。

- ✓ ログ バックアップ（トランザクション ログ バックアップ）
- ✓ 差分バックアップ
- ✓ 復旧状態（RECOVERY、NORECOVERY、STANDBY）
- ✓ 複数のバックアップから復元する場合の復旧状態

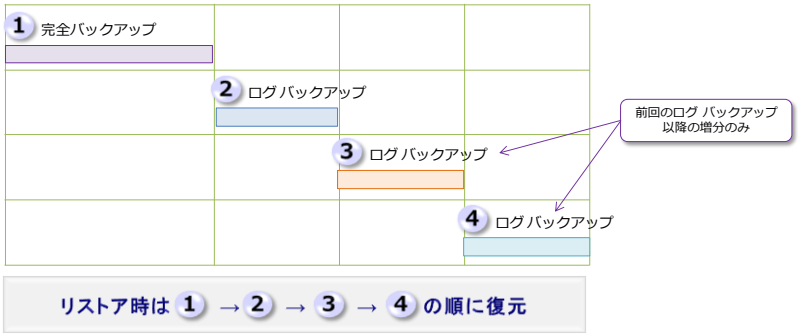
4.1　ログ バックアップと差分バックアップ

ログ バックアップと差分バックアップ

完全バックアップは、データベースを丸ごとバックアップする必要があるので、データベースのサイズが大きい場合には、バックアップに時間がかかってしまいます。そこで、バックアップ時間を短縮するための機能として、「**ログ バックアップ**」と「**差分バックアップ**」の 2 種類のバックアップが用意されています。

- **ログ バックアップ（トランザクション ログ バックアップ）**

  ログ バックアップは、前回のログ バックアップ以降の増分情報のみをバックアップできる機能です。これにより、バックアップにかかる時間を短縮することができます。

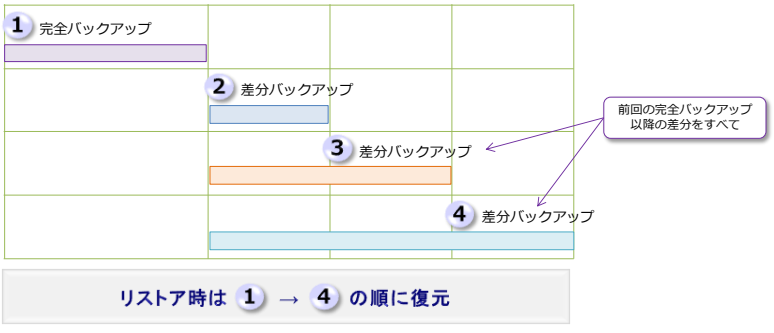


  ログ バックアップ定期的に取得している場合は、この図のように、リストア時は、完全バックアップをリストアした後に、その後のすべてのログ バックアップをリストアすることで、最新の状態まで復元することができます。

  また、前の Step で説明したように、ログ バックアップ時には、トランザクション ログ内の不要になった領域が切り捨てられるので、トランザクション ログの肥大化を防ぐ効果もあります。

- **差分バックアップ**

  差分バックアップは、前回の完全バックアップ以降の差分情報のみをバックアップできる機能です。これにより、バックアップにかかる時間を（完全バックアップを実行するよりも）短縮することができます。



  この図のように、差分バックアップは、実行のたびに、完全バックアップ以降の差分情報を

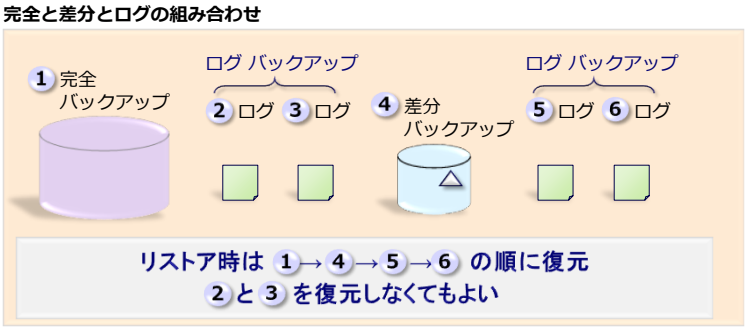
すべてバックアップしていくので（増分のみではないので）、ログ バックアップよりもバックアップ時間がかかります。

この反面、差分バックアップでは、リストア時間をログ バックアップよりも短縮することができます。完全バックアップと最後の差分バックアップのみをリストアすれば良いからです。

このように、差分バックアップとログ バックアップには、一長一短があるので、次のように組み合わせて利用すると、それぞれの長所を活かせるようになります。

ログ バックアップと差分バックアップを組み合わせたバックアップ計画

ログ バックアップと差分バックアップは、次のように組み合わせて利用することができます。

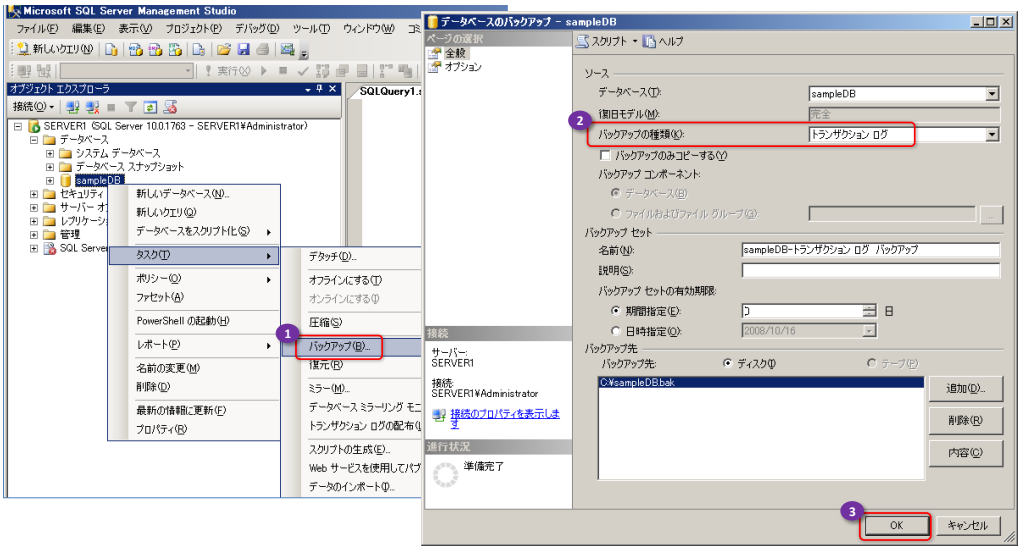


このように組み合わせると、バックアップ時間もリストア時間も短縮できるようになります。たとえば、日中のトランザクションが多く発生する時間帯は、バックアップ時間が短くて済むログ バックアップのみを実行し、1 日の終わりなどに差分バックアップを取得、そして 1 週間に 1 回完全バックアップを取得、という形で利用して、バックアップ時間を短縮することができます。

また、リストア時は、完全バックアップと差分バックアップを復元した後に、差分バックアップ以降のログ バックアップのみを復元するだけで済むので、リストア時間を短縮することもできます。

ログ バックアップの実行方法

ログ バックアップを実行するには、完全バックアップのときと同様、Mangement Studio から、次のように該当データベースを右クリックして、[**タスク**] メニューの [**バックアップ**] をクリックします。



［データベースのバックアップ］ダイアログで、［**バックアップの種類**］で「**トランザクション ログ**」を選択すれば、ログ バックアップになります。

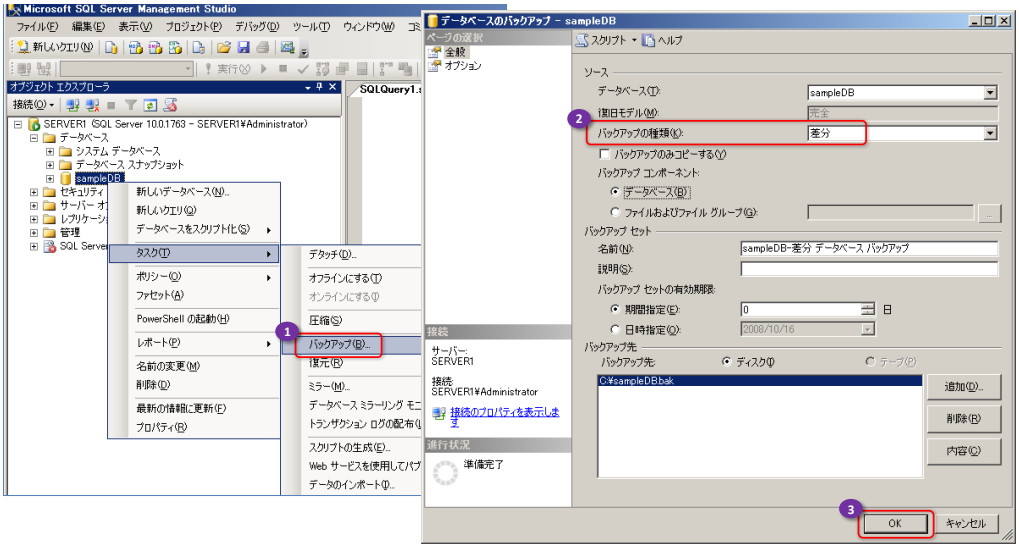
GUI 操作ではなく、SQL ステートメントを使用してログ バックアップを実行する場合は、「**BACKUP LOG**」ステートメントを利用します。構文は、次のとおりです。

```
BACKUP LOG データベース名
 TO { DISK | TAPE } = 'ファイルパス' [ WITH オプション ]
```

完全バックアップを実行するときの BACKUP DATABASE が BACKUP LOG へ変わっただけです。

差分バックアップの実行方法

差分バックアップを実行するには、他のバックアップと同様 Mangement Studio から、次のように該当データベースを右クリックして、［タスク］メニューの［バックアップ］をクリックします。



［データベースのバックアップ］ダイアログで、［**バックアップの種類**］で「**差分**」を選択すれば、

差分バックアップになります。

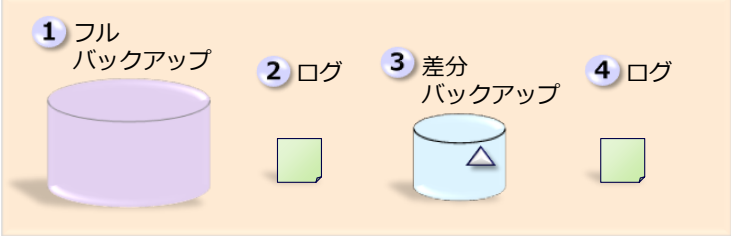
GUI 操作ではなく、SQL ステートメントを使用して差分バックアップを実行する場合は、完全バックアップと同様、次のように「**BACKUP DATABASE**」ステートメントを利用します。

```
BACKUP DATABASE データベース名
 TO { DISK | TAPE } = 'ファイルパス' WITH DIFFERENTIAL [,オプション ]
```

完全バックアップとの違いは、**WITH** オプションへ「**DIFFERENTIAL**」と指定する点です。

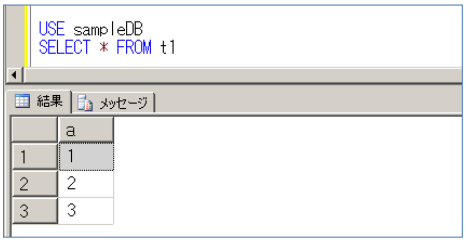
Let's Try

それでは、ログ バックアップと差分バックアップを試してみましょう。ここでは、次のような順番でバックアップを実行してみましょう。



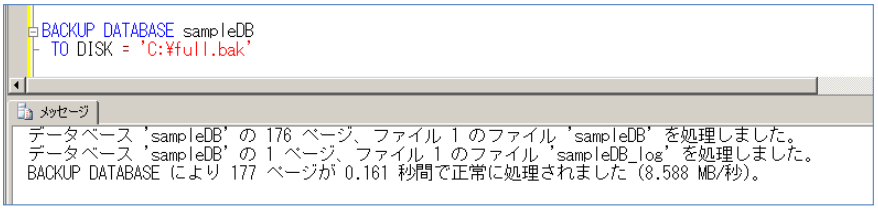
1. まずは、クエリ エディタから、「**sampleDB**」データベースへ接続して、バックアップ前の「**t1**」テーブルの中身を確認します。

```
USE sampleDB
SELECT * FROM t1
```



2. 次に、**BACKUP DATABASE** ステートメントを利用して完全バックアップを実行します。次のようにパスを指定して、**C:** ドライブの直下へ「**full.bak**」という名前でバックアップを取得します。

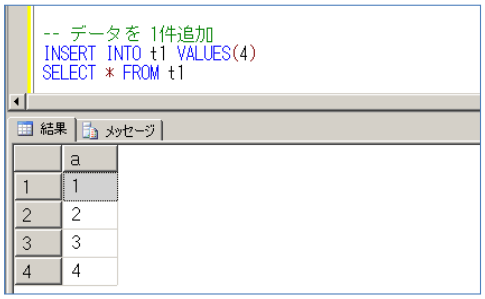
```
BACKUP DATABASE sampleDB
 TO DISK = 'C:¥full.bak'
```



「正常に処理されました」と表示されれば、バックアップが完了しています。

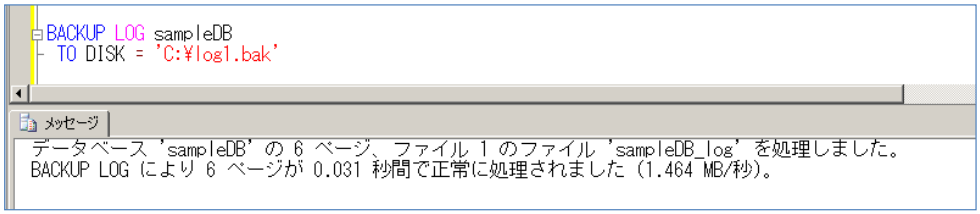
3. 完全バックアップが完了したら、バックアップの効果を試すために、「**t1**」テーブルへデータを 1 件追加しましょう。

```
INSERT INTO t1 VALUES(4)
SELECT * FROM t1
```



4. 次に、**BACKUP LOG** ステートメントを利用して「**ログ バックアップ**」を実行します。次のようにパスを指定して、**C:** ドライブの直下へ「**log1.bak**」という名前でバックアップを取得します。

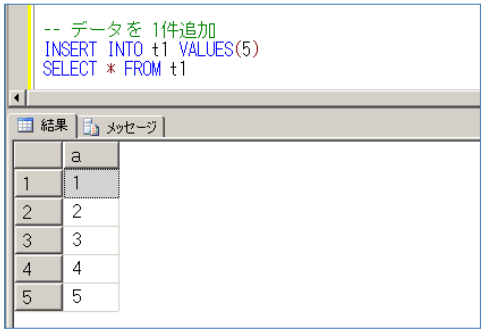
```
BACKUP LOG sampleDB
 TO DISK = 'C:¥log1.bak'
```



「正常に処理されました」と表示されれば、ログ バックアップが正しく完了しています。

5. ログ バックアップが完了したら、「**t1**」テーブルに対してデータをもう 1 件追加しましょう。

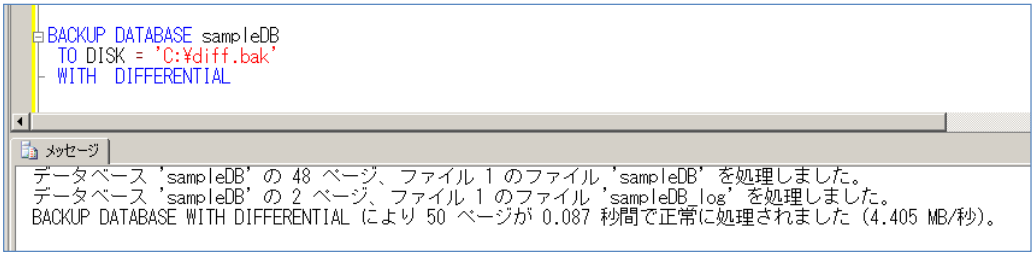
```
INSERT INTO t1 VALUES(5)
SELECT * FROM t1
```



6. 次に、**BACKUP DATABASE** ステートメントを利用して「**差分バックアップ**」を実行します。次のようにパスを指定して、**C:** ドライブの直下へ「**diff.bak**」という名前でバックアップを

取得します。

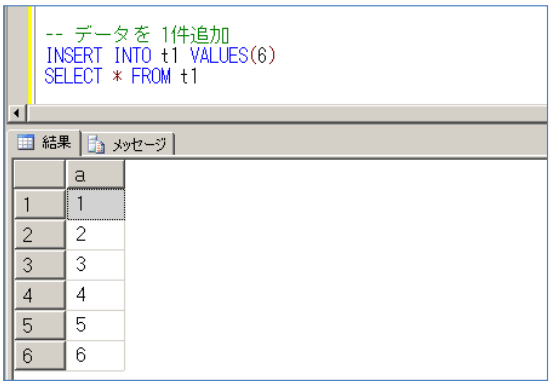
```
BACKUP DATABASE sampleDB
 TO DISK = 'C:¥diff.bak'
  WITH  DIFFERENTIAL
```



「正常に処理されました」と表示されれば、差分バックアップが正しく完了しています。

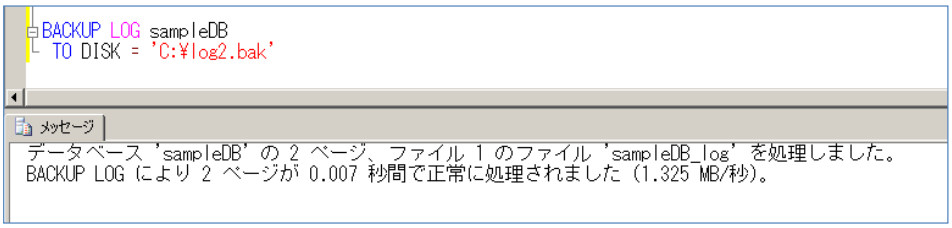
7. 差分バックアップが完了したら、「**t1**」テーブルに対してデータをもう 1 件追加しましょう。

```
INSERT INTO t1 VALUES(6)
SELECT * FROM t1
```



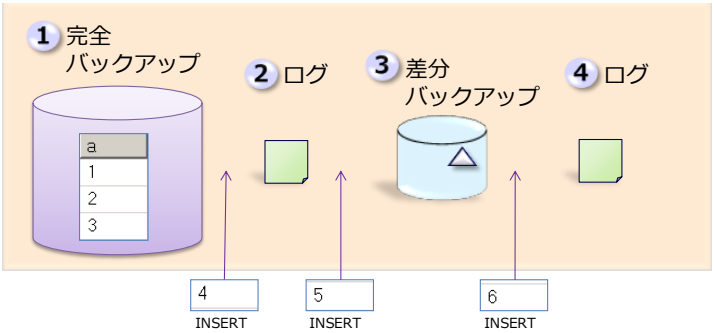
8. 次に、**BACKUP LOG** ステートメントを利用して、もう一度「**ログ バックアップ**」を実行します。次のようにパスを指定して、**C:** ドライブの直下へ「**log2.bak**」という名前でバックアップを取得します。

```
BACKUP LOG sampleDB
 TO DISK = 'C:¥log2.bak'
```



「正常に処理されました」と表示されれば、バックアップが完了しています。

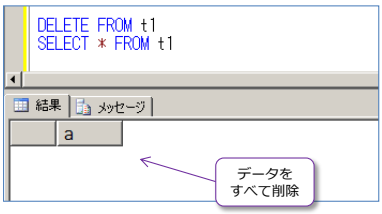
ここまでの操作をまとめると、次の図のようになります。



データの全削除

9. バックアップが完了したら、クエリ エディタから、次のように **DELETE** ステートメントを実行して、すべてのデータを削除します。

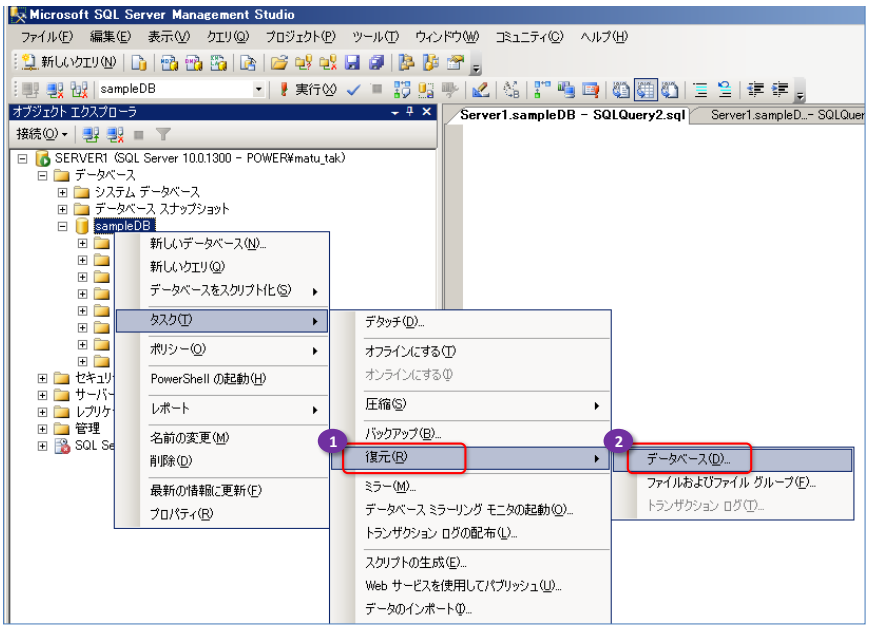
```
DELETE FROM t1
SELECT * FROM t1
```



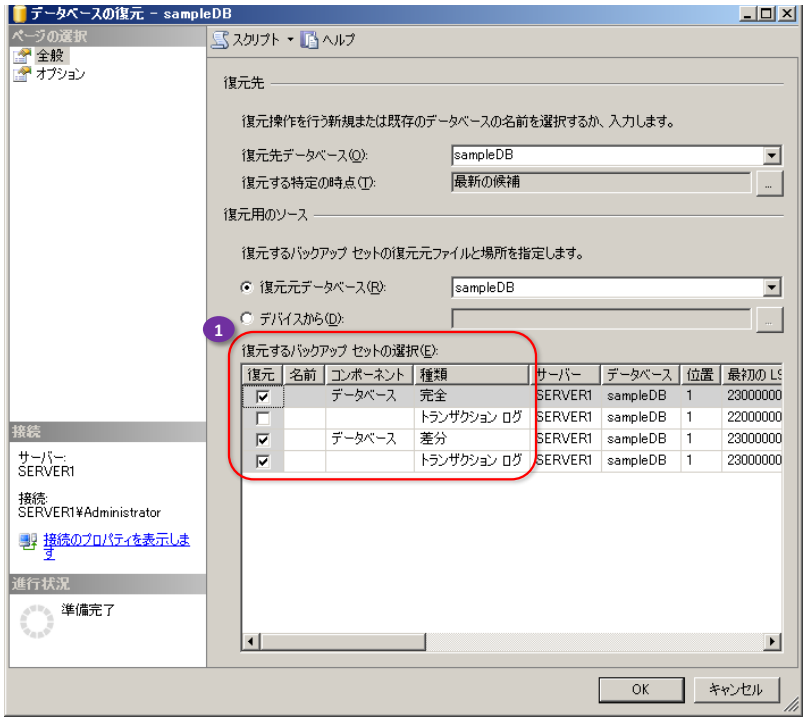
GUI からのリストア

次に、取得したバックアップを Management Studio からリストアしてみましょう。

10. リストアを実行するには、次のように［**タスク**］メニューの［**復元**］から［**データベース**］をクリックします。



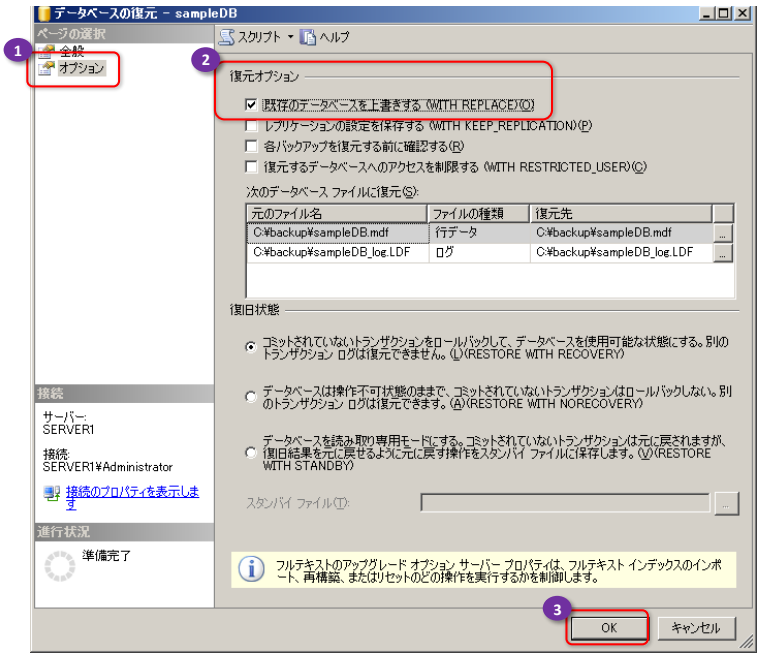
［データベースの復元］ダイアログが表示されたら、［**復元するバックアップ セットの選択**］に注目します。



sampleDB データベースに対して実行した完全バックアップ以降のバックアップ履歴が表示されて、最新の状態へ戻すために必要なバックアップに対して自動的にチェック マークが付いていることを確認できます。

差分バックアップは、前回の完全バックアップ以降の差分を含むので、差分バックアップ以前のログ バックアップは復元する必要がないので、チェックされていません。

11. 続いて、次のように［**オプション**］ページをクリックして開き、［**復元オプション**］で［**既存のデータベースを上書きする**］をチェックします。

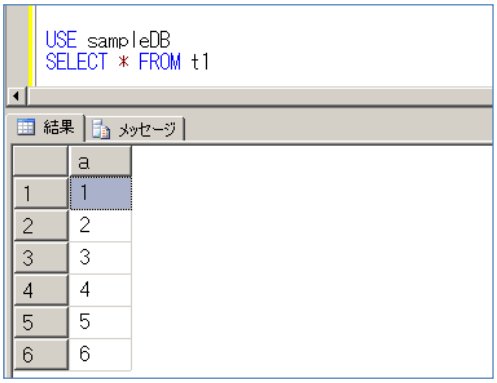


オプション設定後、[OK] ボタンをクリックすると、リストアが開始されます。数秒後に、次のエラー「データベースは使用中なので、排他アクセスを獲得できませんでした」が表示される場合は、クエリ エディタを一度すべて閉じてから、もう一度リストアを実行してください。

リストアが完了した場合は、次のダイアログが表示されます。

12. リストアが完了したら、クエリ エディタを開いて、データが復元されたかどうかを確認してみましょう。

```
USE sampleDB
SELECT * FROM t1
```

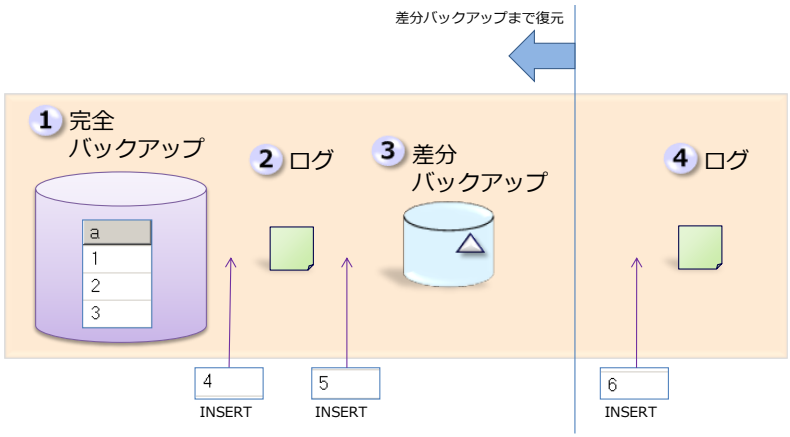


削除したデータが復活して、最新の状態へ復元できたことを確認できます。

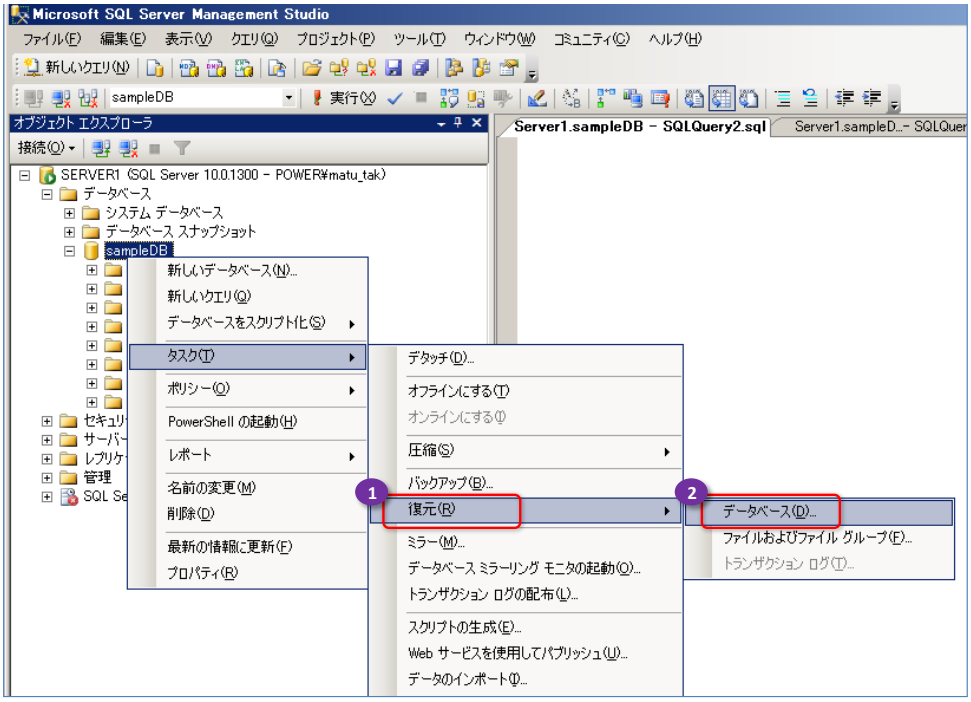
13. 確認後、次の手順のためにクエリ エディタを終了して、sampleDB データベースへの接続を切ります。

差分バックアップまでの復元

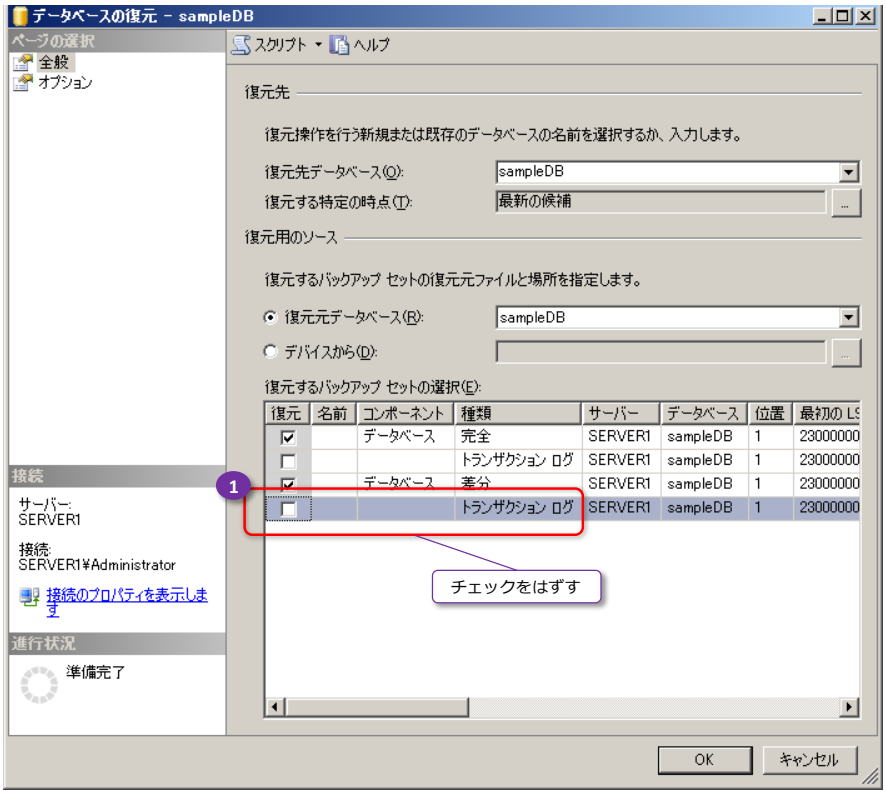
次に、すべてのバックアップを戻さずに、差分バックアップまでのリストアを実行して、差分バックアップを取得した時点のデータへ戻してみましょう。



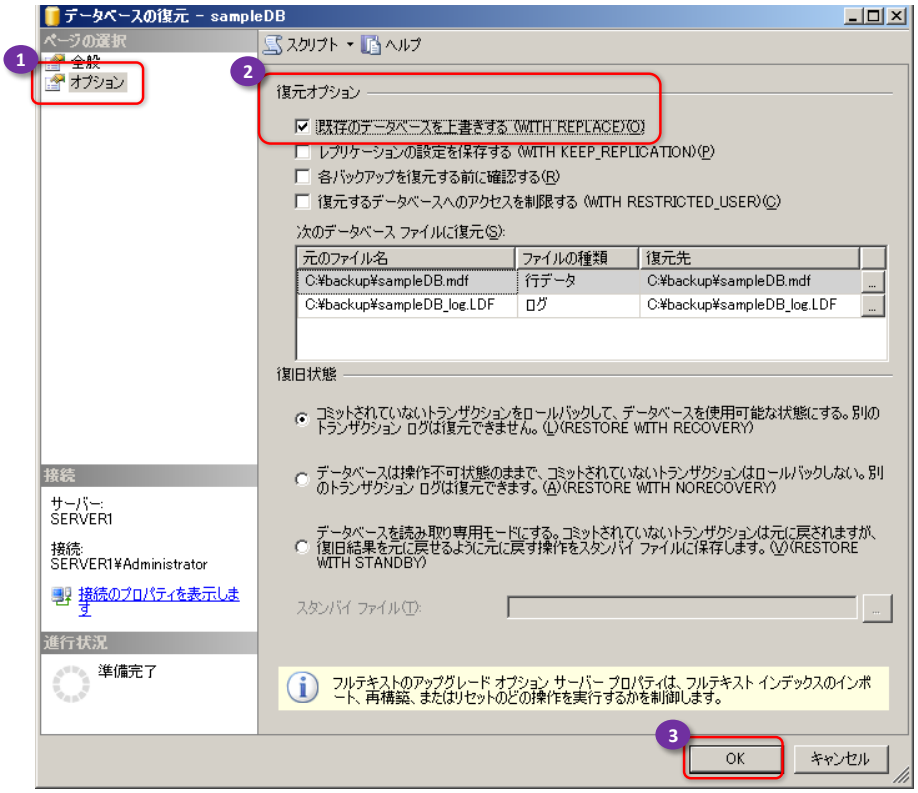
1. まずは、前回の手順と同じように［**タスク**］メニューの［**復元**］から［**データベース**］をクリックします。



［データベースの復元］ダイアログでは、次のように［**復元するバックアップ セットの選択**］で、最後のログ バックアップのチェックをはずして、差分バックアップまでのリストアを実行するようにします。

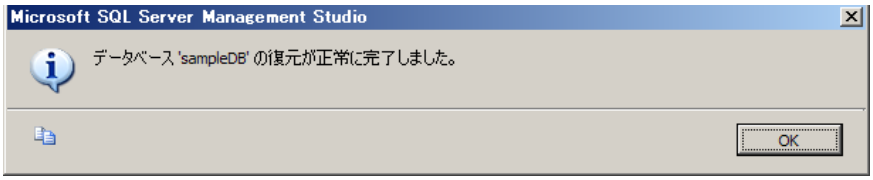


2. 続いて、次のように［**オプション**］ページをクリックして開き、［**復元オプション**］で［**既存のデータベースを上書きする**］をチェックします。



オプション設定後、［OK］ボタンをクリックすると、リストアが開始されます。

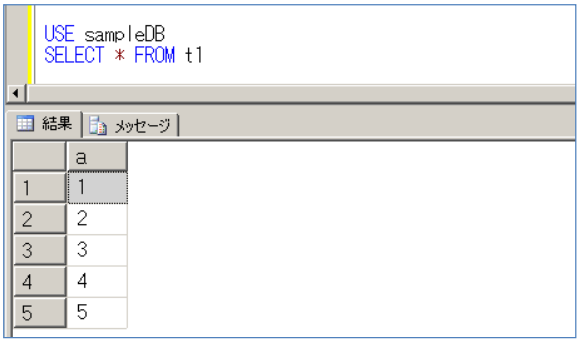
リストアが完了すると、次のダイアログが表示されます。



「データベースが使用中」エラーが表示される場合は、すべてのクエリ エディタを閉じてから、再度リストアを実行してください。

3. リストアが完了したら、クエリ エディタを開いて、データがどこまで復元されたかを確認してみましょう。

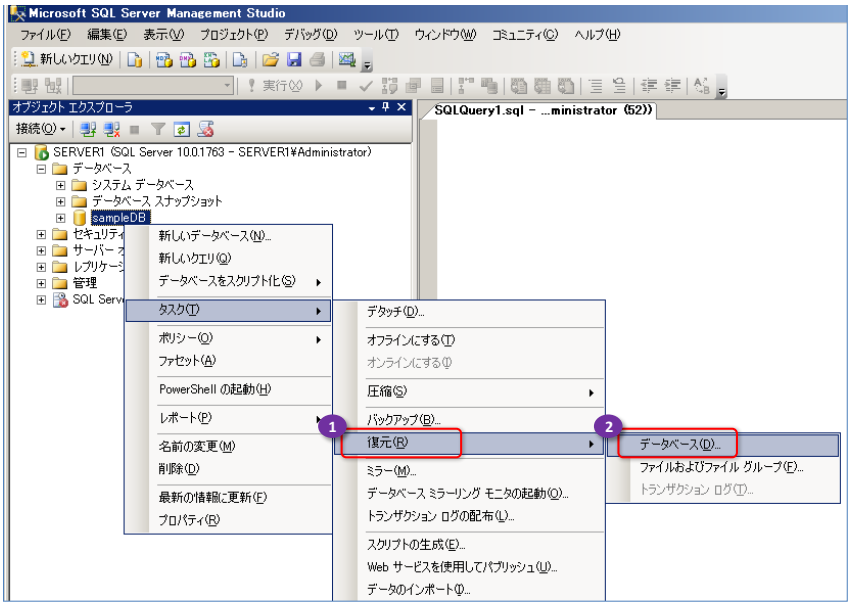
```
USE sampleDB
SELECT * FROM t1
```



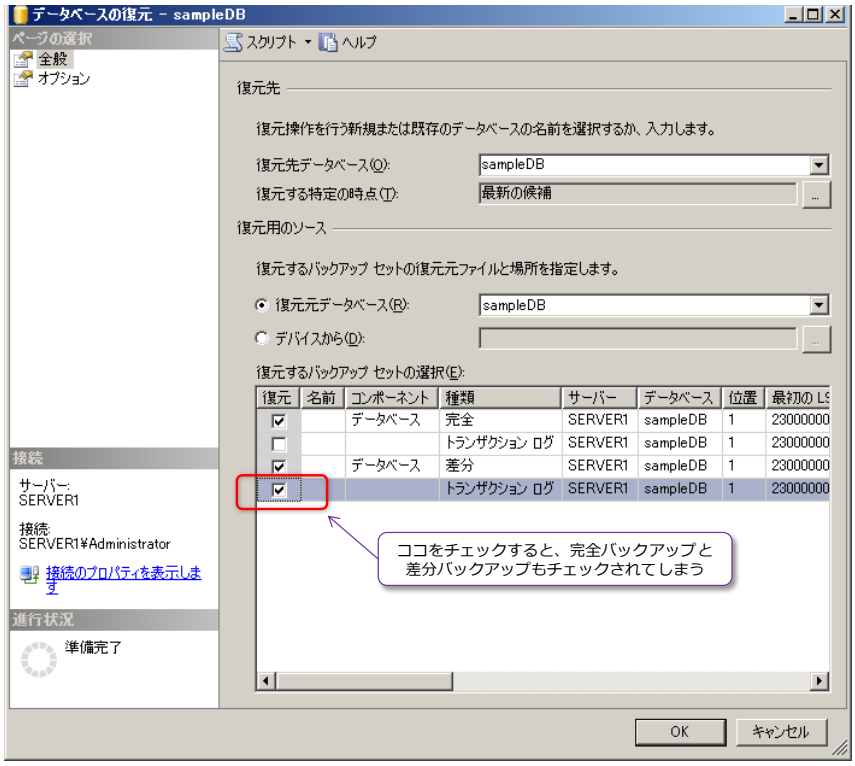
差分バックアップを取得した時点まで復元できたことを確認できます。

残りのログ バックアップの復元

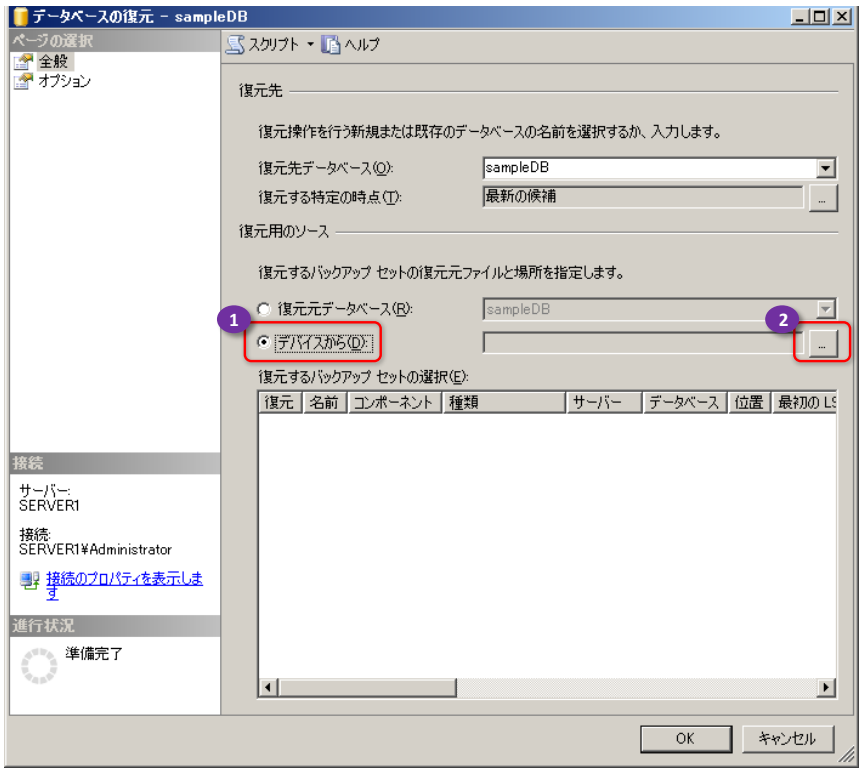
4. 続いて、前の手順の [データベースの復元] ダイアログでチェックをはずした、最後のログ バックアップからのリストアも実行してみましょう。[**タスク**] メニューの [**復元**] から [**データベース**] をクリックします。



5. ［データベースの復元］ダイアログでは、［**復元するバックアップ セットの選択**］で、最後のログ バックアップのみを選択できないことを確認します。

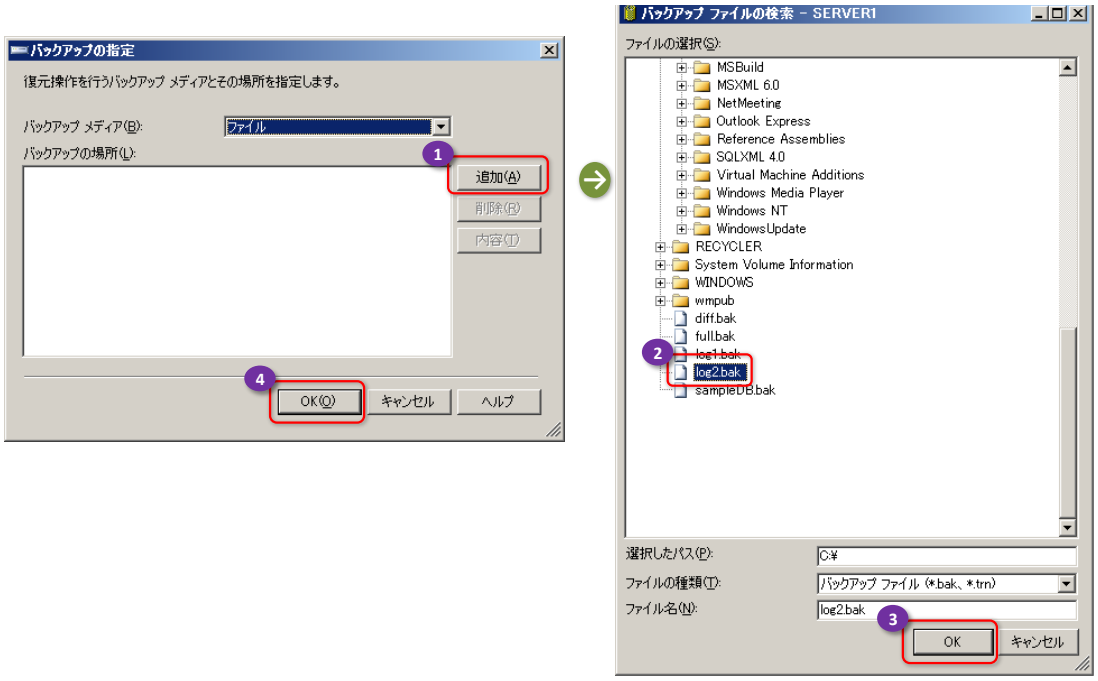


6. 続いて、最後に取得したログ バックアップ ファイル「**log2.bak**」を直接指定するために、次のように［**復元用のソース**］で、［**デバイスから**］を選択して［**...**］ボタンをクリックします。



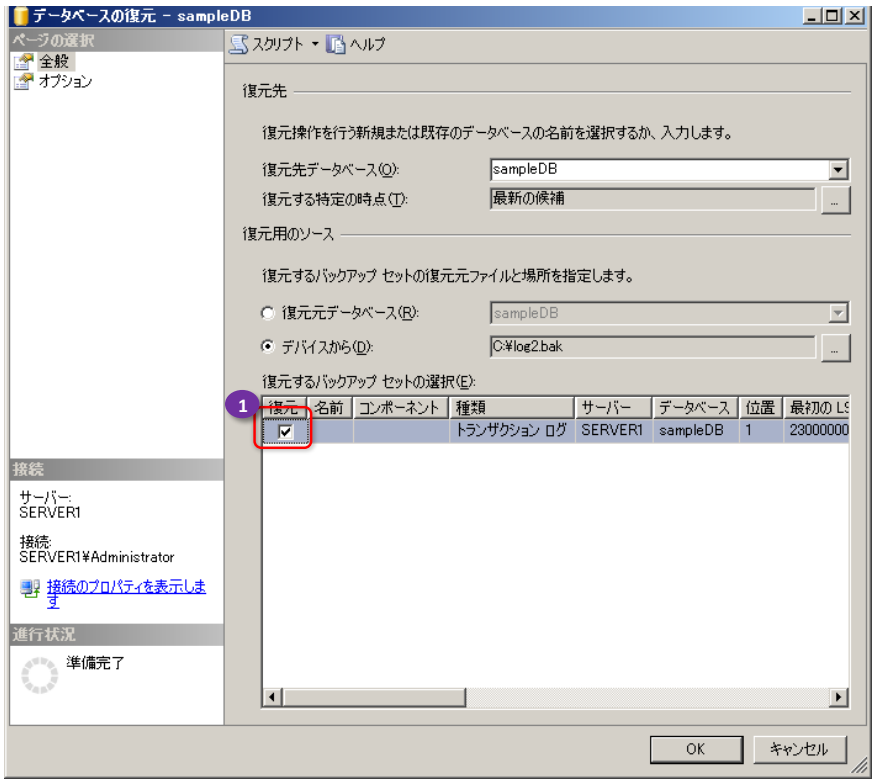
［バックアップの指定］ダイアログが表示されたら、［**追加**］ボタンをクリックして、［バック

アップ ファイルの検索］ダイアログを表示し、「**C:¥log2.bak**」ファイルを選択して、［OK］ボタンをクリックします。

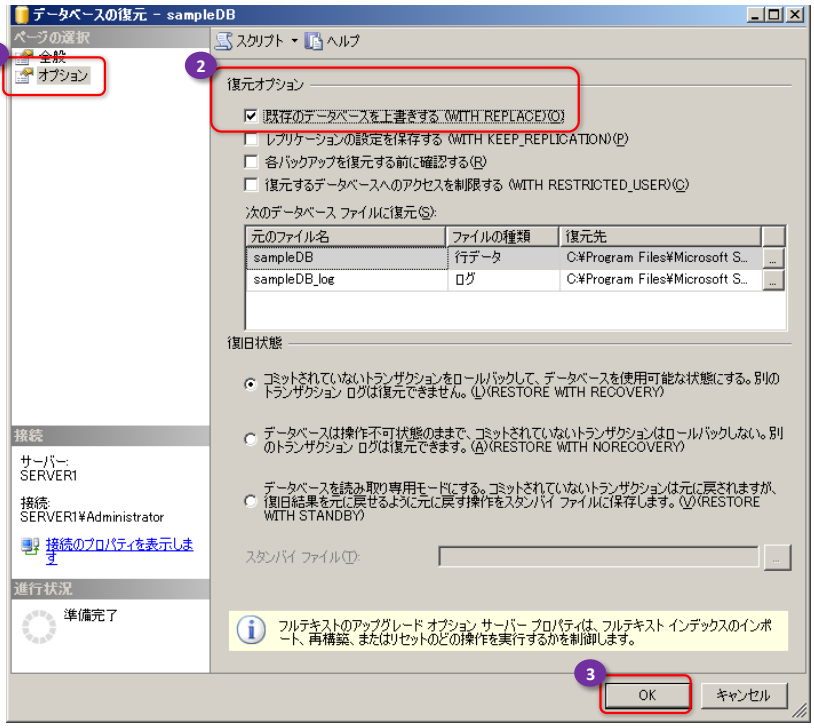


［バックアップの指定］ダイアログへ戻ったら、［OK］ボタンをクリックします。

［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の選択］で、「トランザクション ログ」をチェックします。



続いて、［ページの選択］で［**オプション**］をクリックして、［オプション］ページを開き、［復元オプション］で［**既存のデータベースを上書きする**］をチェックします。



オプション設定後、［OK］ボタンをクリックすると、次のエラーが表示されます。

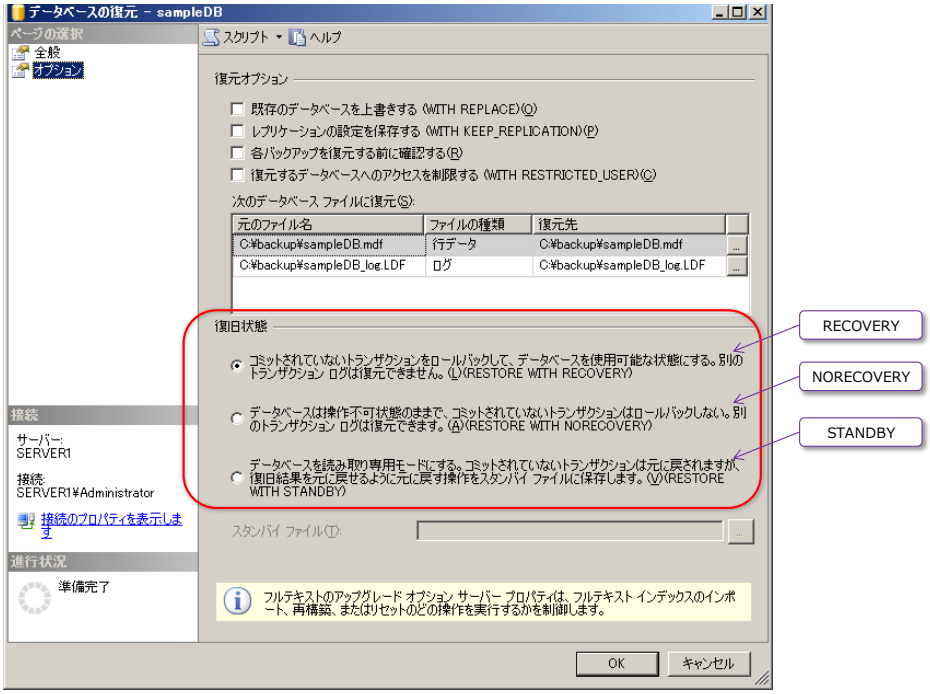
これは、復旧が完了したデータベースに対して、ログ バックアップや差分バックアップが復元できないという主旨のエラーです。これを回避するには、次に説明する「**復旧状態**」オプションを設定して、以前のバックアップを復元しておく必要があります。

4.2　復旧状態オプション

復旧状態オプション

リストアしたいバックアップが複数ある場合には、「**復旧状態**」オプションに注意する必要があります。前の手順で試したように、途中のバックアップをリストアするときに、デフォルトの復旧状態でリストアしてしまうと、残りのバックアップがリストアできなくなるからです。

「復旧状態」オプションは、リストア時の［データベースの復元］ダイアログで設定できます。



復旧状態オプションのそれぞれの意味は、次の通りです。

- **RECOVERY（デフォルトの復旧状態）**

  RECOVERY では、リストア後にデータベースを完全復旧した状態にします。これは、リストアが完了したことの合図になり、残りのバックアップ（差分バックアップやトランザクション ログバックアップ）をリストアすることはできません。したがって、最後のバックアップを復元する場合にのみ、このオプションを選択するようにします。

- **NORECOVERY（復旧中）**

  NORECOVERY は、まだリストアすべきバックアップ（差分バックアップやログ バックアップ）がある場合に使用するオプションです。このオプションを指定してリストアを行った場合は、データベースが「**復旧中**」（リストア中）となり、ユーザーからの接続を一切できないようにブロックします。リストア中にデータベースが更新されては、データの不整合が発生してしまうからです。

- **STANDBY（スタンバイ）**

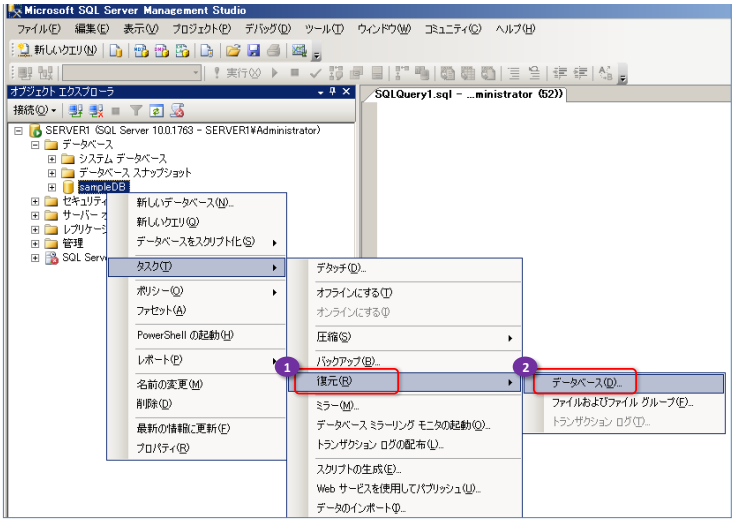
  STANDBY は、NORECOVERY と同様、まだリストアすべきバックアップがある場合に

使用するオプションです。NORECOVERY との違いは、NORECOVERY がユーザーからの接続をすべて拒否するのに対して、STANDBY では、ユーザーからの接続を拒否せずに、読み取り操作 (SELECT) のみを実行可能にしている点です。データが更新されずに、読み取られるだけなら、データの不整合が発生しないからです。

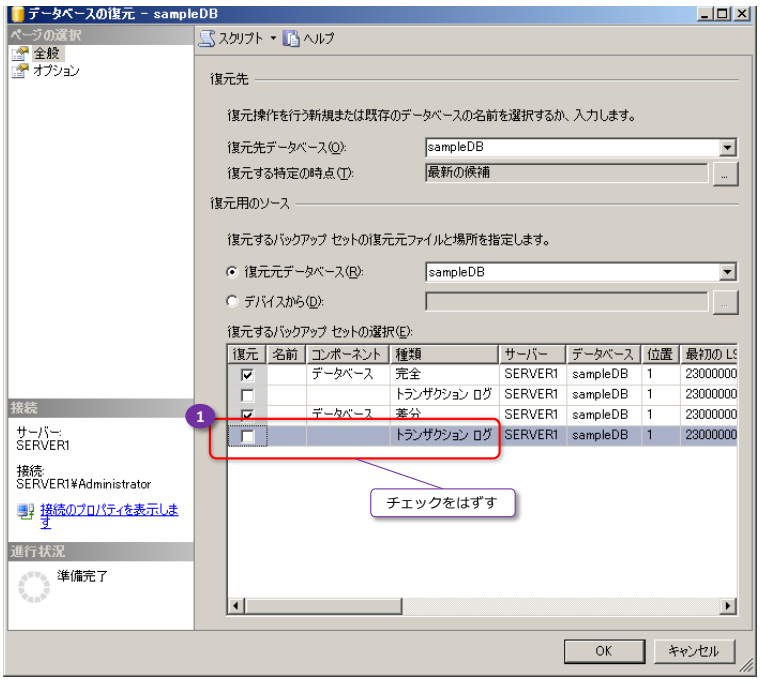
Let's Try : STANDBY 状態で差分まで復元

それでは、復旧状態オプションを試してみましょう。ここでは、**STANDBY** 状態を指定して、前の手順で行った差分バックアップまでのリストアを実行してみましょう。

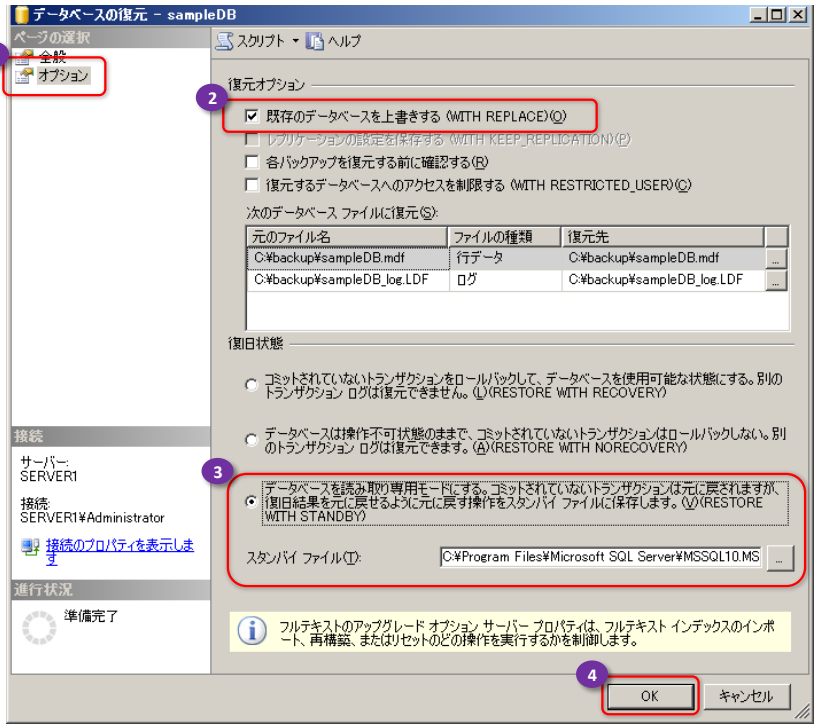
1. まずは、次のように［**タスク**］メニューの［**復元**］から［**データベース**］をクリックします。



2. ［データベースの復元］ダイアログでは、次のように［**復元するバックアップ セットの選択**］で、最後のログ バックアップのチェックをはずして、差分バックアップまでのリストアを実行するようにします。

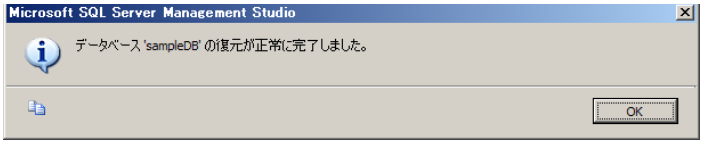


続いて、次のように［**オプション**］ページをクリックして開き、［**復元オプション**］で［**既存のデータベースを上書きする**］をチェックし、［**復旧状態**］で 3 つ目の「**STANDBY**」オプションを選択します。



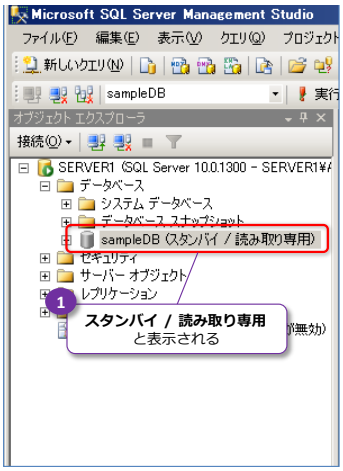
オプション設定後、［OK］ボタンをクリックすると、リストアが開始されます。

リストアが完了すると、次のダイアログが表示されます。

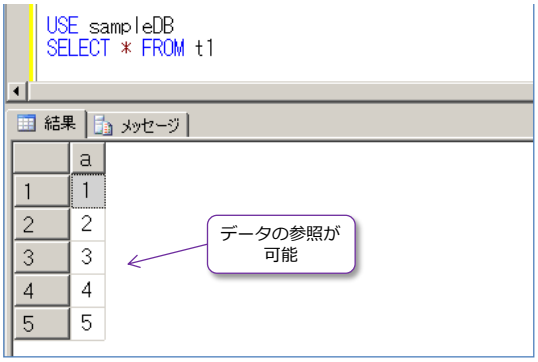


「データベースが使用中」エラーが表示される場合は、すべてのクエリ エディタを閉じてから、再度リストアを実行してください。

3. リストアが完了したら、次のように sampleDB データベースの状態が「**スタンバイ / 読み取り専用**」と表示されることを確認します。



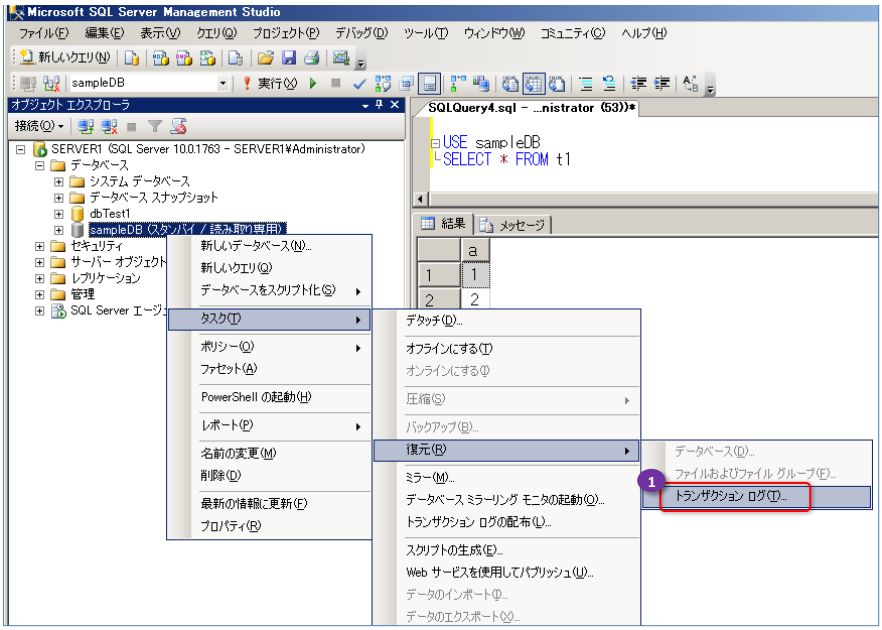
4. 次に、クエリ エディタを開いて、データが参照できることを確認してみましょう。



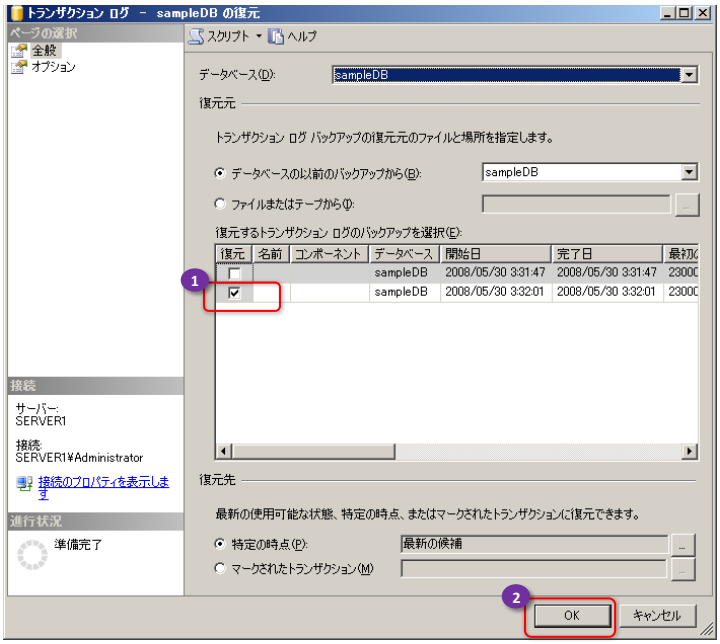
差分バックアップを取得した時点へ復元できたことを確認できます。このように STANDBY オプションでは、リストア中のデータを読み取ることが可能です。

残りのログ バックアップの復元

5. 続いて、前の手順の [データベースの復元] ダイアログでチェックをはずした、最後のログ バックアップのみをリストアしてみましょう。残りのログ バックアップを復元するには、[**タスク**] メニューの [**復元**] から [**トランザクション ログ**] をクリックします。



6. [トランザクション ログの復元]ダイアログが表示されたら、[**復元するトランザクション ログのバックアップを選択**] で、ログ バックアップの履歴が表示されることを確認して、2 つ目のバックアップをチェックします。



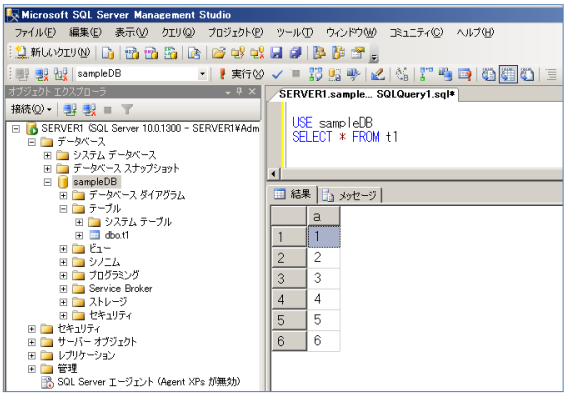
［OK］ボタンをクリックすると、リストアが開始されます。

リストアが完了すると、次のダイアログが表示されます。

「データベースが使用中」エラーが表示される場合は、すべてのクエリ エディタを閉じてから、再度リストアを実行してください。

7. リストアが完了したら、sampleDB が通常の表示に戻っていることを確認できます。クエリ エディタを開いて、データがどこまで復元されたかを確認してみましょう。

```
USE sampleDB
SELECT * FROM t1
```



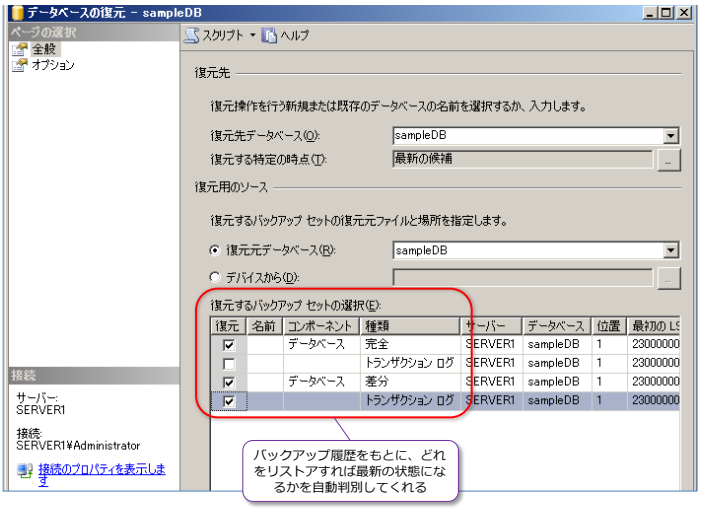
最後のログ バックアップを取得した時点へ復元できたことを確認できます。

このように、復旧状態オプションを「**STANDBY**」へ設定した場合には、リストア中にデータを参照して、かつ残りのバックアップもリストアできるようになります。

4.3　RESTORE ステートメントによる複数バックアップの復元

バックアップ履歴機能

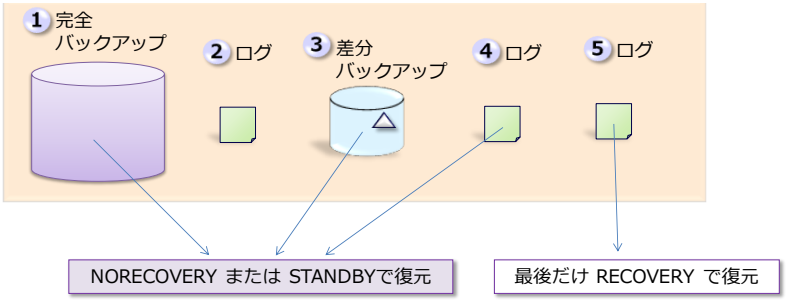
GUI からのリストア時は、今までにバックアップを実行した履歴をもとに、何を戻せば良いのかを指示してくれるので、「**復旧状態**」オプションを意識しなくても、最新の状態へリストアすることが簡単に行えました（前の Step では、あえて途中まで復元して、途中のデータを参照することで復旧状態オプションの効果を試しました）。



このバックアップ履歴は、内部的には **msdb** システム データベースへ記録されています。したがって、もし msdb データベースが破損してしまった場合や、ほかのマシンへデータベースを復元する場合、SQL Server を再インストールしたマシンへデータベースを復元する場合には、バックアップ履歴が存在しないことになります。

バックアップ履歴がない場合のリストア

バックアップ履歴が存在しない場合は、複数のバックアップを １つ １つ、「復旧状態」オプションに注意しながらリストアする必要があります。複数のバックアップをリストアするには、次のように最後のバックアップを除いて、**NORECOVERY** または **STANDBY** オプションを指定してリストアをする必要があります。



途中のバックアップは、NORECOVERY または STANDBY オプションを指定してリストアし、リストア中のデータが更新されないようにブロックして、データの不整合が発生するのを防ぐ必要が

あるからです。最後のバックアップをリストアするときは、RECOVERY オプションを指定してリストアし、これがリストア完了の合図になります（ユーザーが利用できる状態になります）。

RESTORE ステートメントでの「復旧状態」オプションの指定

RESTORE ステートメントで「復旧状態」オプションを設定するには、次のように **WITH** オプションを利用します。

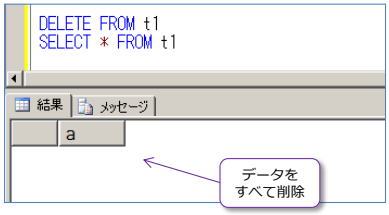
```
RESTORE DATABASE データベース名
 FROM { DISK | TAPE } = 'ファイルパス'
 [ WITH RECOVERY | NORECOVERY | STANDBY ]
```

Let's Try： 復旧状態オプションを設定しない場合の動作

それでは、RESTORE ステートメントで「復旧状態」オプションを試してみましょう。まずは、オプションを指定しない場合のデフォルトの動作を確認してみましょう。

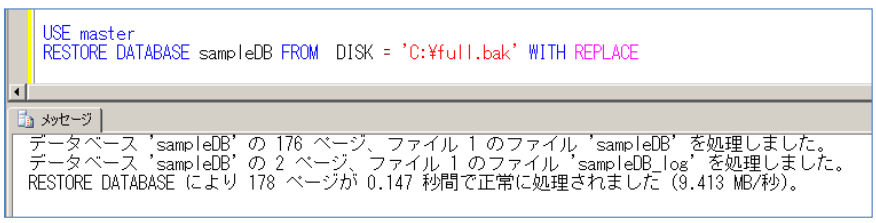
1. 最初に、リストアの効果を確認しやすくするために、クエリ エディタから、「**t1**」テーブルデータをすべて削除します。

```
USE sampleDB
DELETE FROM t1
SELECT * FROM t1
```



2. データの削除後、次のように RESTORE ステートメントを実行して、完全バックアップ「**full.bak**」からリストアします。

```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥full.bak'
  WITH REPLACE
```

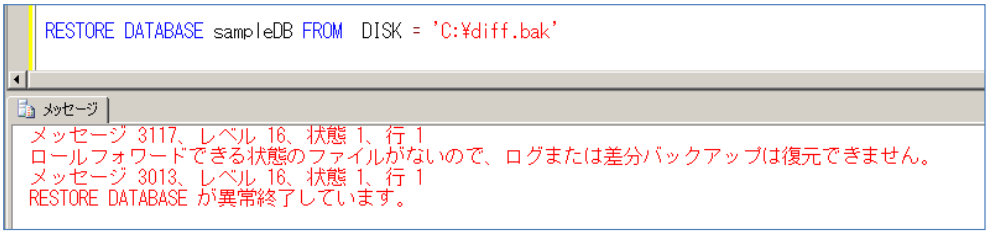


「データベースが使用中」エラーが表示される場合は、すべてのクエリ エディタを閉じてか

ら、再度 RESTORE ステートメントを実行してください。

3. 続いて、差分バックアップ「**diff.bak**」を、RESTORE ステートメントでリストアします。

```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥diff.bak'
```



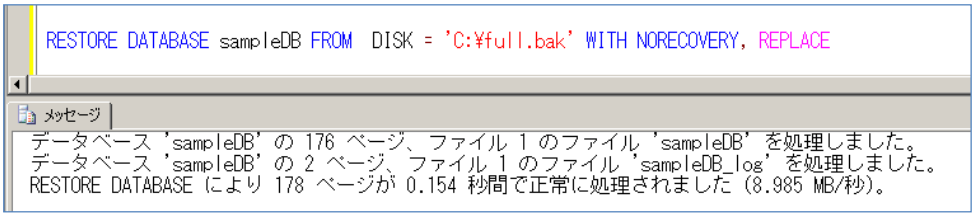
「ログまたは差分バックアップは復元できません」というエラーが表示され、リストアができないことを確認できます。

このように、復旧状態オプションを指定しない場合は、デフォルトの復旧状態「**RECOVERY**」で復元されるので、復旧完了と見なされてしまい、残りのバックアップを続けてリストアすることができません。

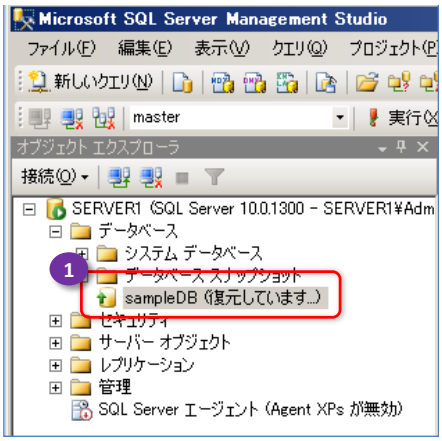
NORECOVERY 復旧状態オプションの指定

1. 次に、「**NORECOVERY**」復旧状態オプションを利用して、もう一度完全バックアップからリストアし直してみましょう。

```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥full.bak'
  WITH NORECOVERY, REPLACE
```



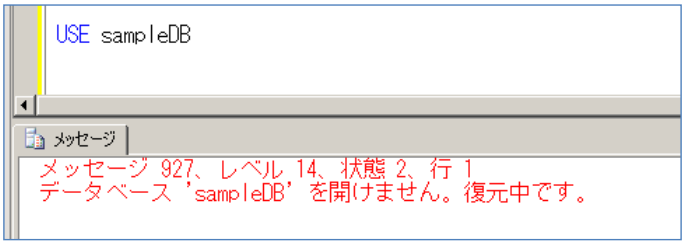
2. 完全バックアップからのリストア後、オブジェクト エクスプローラで［データベース］フォルダを右クリックして［**最新の情報に更新**］をクリックします。



データベースの状態が「**sampleDB（復元しています）**」と表示されることを確認できます。

3. 次に、クエリ エディタから sample DB へ接続してみます。

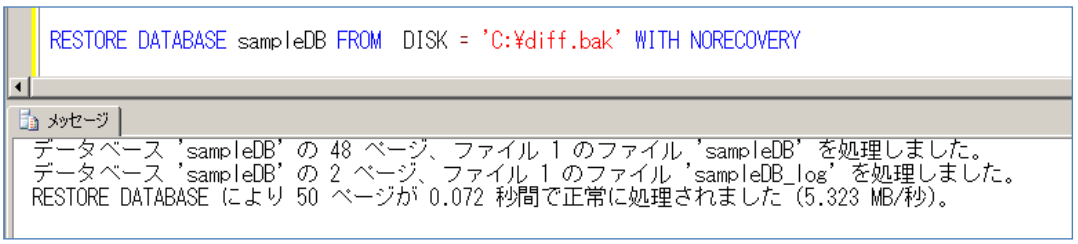
```
USE sampleDB
```



「**開けません。復元中です**」とエラーが表示されて、データベースへ接続できないことを確認できます。このように NORECOVERY オプションを指定した場合は、データベースの状態が復旧中（リストア中）になり、続けて、後続のバックアップをリストアできるようになります。

4. 次に、差分バックアップ「**diff.bak**」を NORECOVERY オプションを指定してリストアしてみましょう。

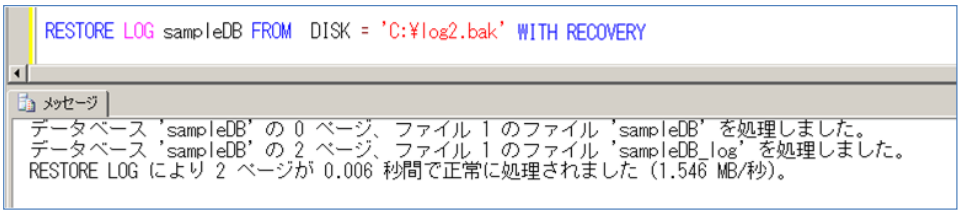
```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥diff.bak'
  WITH NORECOVERY
```



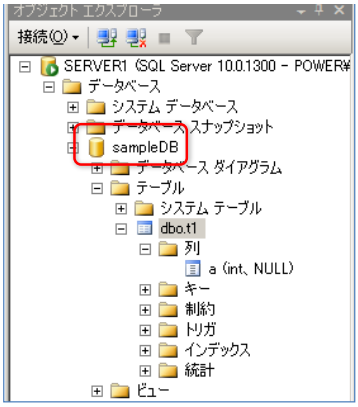
問題なくリストアが実行できたことを確認できます。

5. 次に、ログ バックアップ「**log2.bak**」をリストアしましょう。これは、リストアする最後のバックアップになるので、「**RECOVERY**」オプションを指定します。

```
USE master
RESTORE LOG sampleDB
 FROM DISK = 'C:¥log2.bak'
  WITH RECOVERY
```

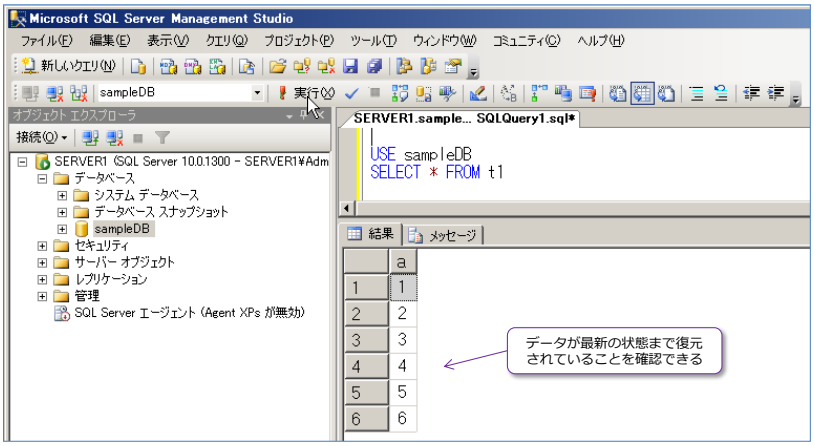


6. ログ バックアップからのリストア後、オブジェクト エクスプローラで「sampleDB」データベースを右クリックして［**最新の情報に更新**］をクリックすると、sampleDB から、「復元しています」という文字が消えていて、通常通りに表示されることを確認できます。



7. クエリ エディタから sampleDB に接続して、データを参照してみます。

```
USE sampleDB
SELECT * FROM t1
```



今度はエラーが表示されず、最後のログ バックアップを取得した時点へ復元できたことを確認できます。

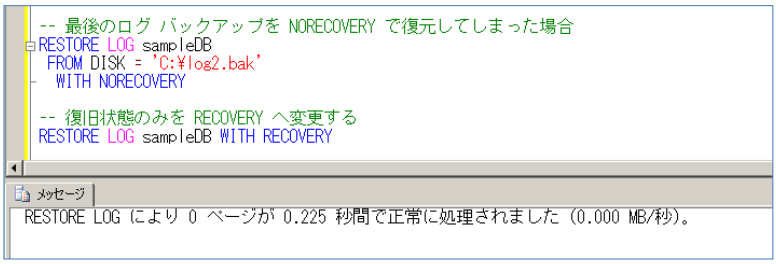
このように、複数のバックアップからリストアする場合には、復旧状態オプションを「**NORECOVERY**」へ指定することで、後続のバックアップをリストアできるようになりま

す。また、最後のバックアップを戻す際には、「**RECOVERY**」オプションを指定することで、データベースが復旧完了（すべてのリストアが完了）状態へすることができます。

Note： RECOVERY をし忘れてしまった場合

最後のバックアップを戻すときに RECOVERY オプションと誤って、NORECOVERY オプションでリストアしてしまう場合もあると思います。この場合は、次のように RESTORE LOG ステートメントを実行することで、「復旧状態」オプションのみを RECOVERY へ変更することができます。

```
RESTORE LOG sampleDB WITH RECOVERY
```

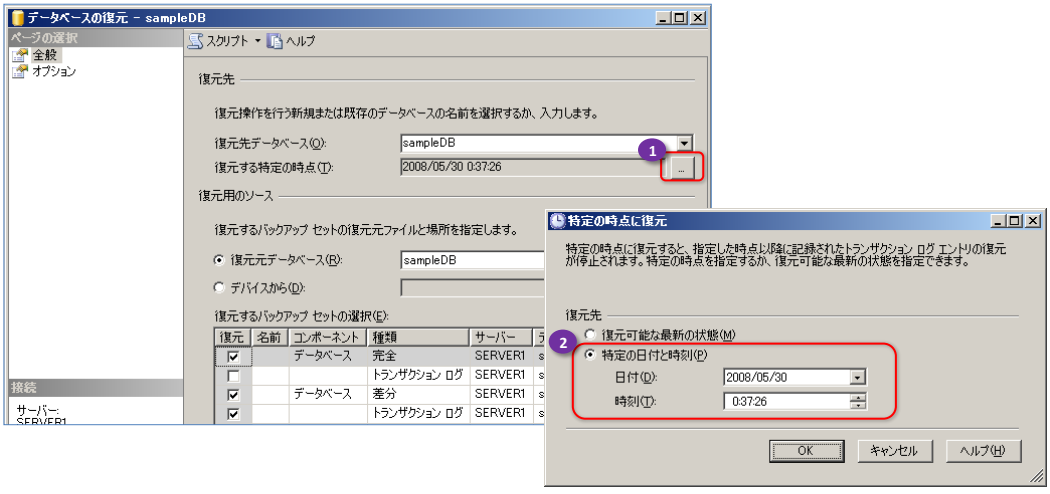


Note： 時刻を指定した復元（STOPAT オプション）

トランザクション ログのリストア時は、時刻を指定してデータを復元することもできます。これを利用することで、特定の時間までのデータへ復元したいということが実現できます。これを行うには、次のように **STOPAT** オプションを指定して、**RESTORE LOG** ステートメントを実行します。

```
RESTORE LOG testDB1
 FROM DISK = 'C:¥testDB1_log.bak'
  WITH RECOVERY, STOPAT = '2008-05-30 00:37:03.363'
```

GUI から時刻指定で復元する場合は、次のように［データベースの復元］ダイアログで設定します。



STEP 5. 障害発生直前までの復旧

この STEP では、障害発生直前までの復旧方法について説明します。データ ファイルが破損しても、トランザクション ログ ファイルが残っていれば、データを障害発生の直前まで復旧することができます。このことを理解するには、トランザクション ログの仕組みと役割を理解することが重要です。

この STEP では、次のことを学習します。

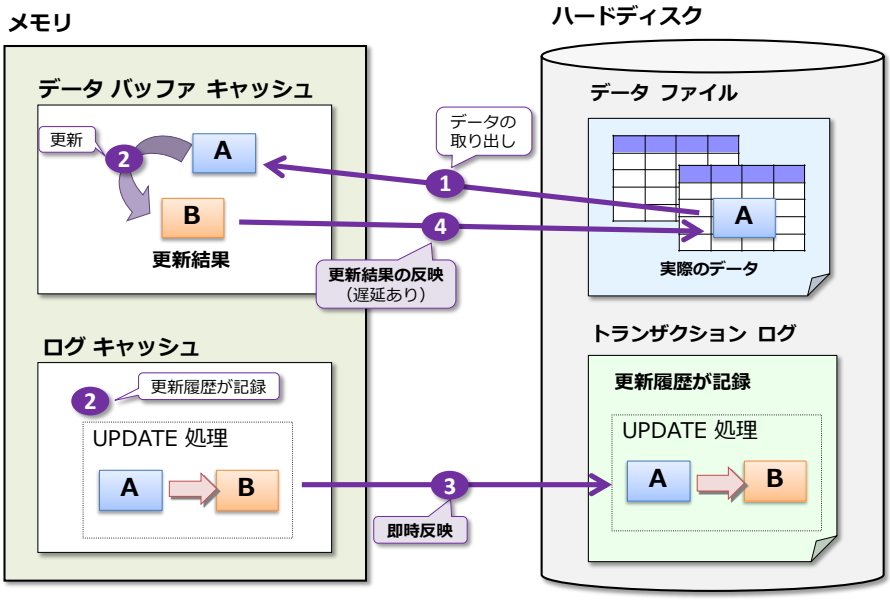
- ✓ トランザクション ログの仕組み
- ✓ 自動復旧処理
- ✓ チェックポイント
- ✓ 障害発生直前までの復旧
- ✓ 復旧モデルとトランザクション ログ

5.1　トランザクション ログの仕組み

トランザクション ログの仕組み

データベースを障害直前まで復旧できるようにするには、トランザクション ログの仕組みと役割を理解することが重要です。そこで、まずはトランザクション ログの仕組みを説明します。

次の図は、テーブル内のデータ「**A**」を「**B**」へ更新（UPDATE）するときの内部動作を示したものです。



この図の動作は、次のようになります。

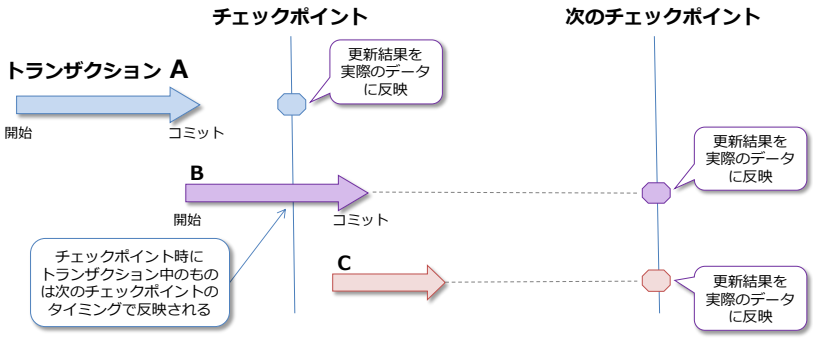
1. 更新対象となるデータがメモリ上のデータ バッファ キャッシュに存在しない場合には、ディスク上のデータ ファイルから読み取る
2. メモリ内の**ログ キャッシュ**へ更新履歴が書き込まれ、**データ バッファ キャッシュ**内のデータが更新される
3. ログ キャッシュへ記録された更新履歴が、ディスク上の**トランザクション ログ ファイル**に書き込まれる
4. **チェックポイント**というプロセスが発生したときに、データ バッファ キャッシュ内の更新結果が**データ ファイル**に書き込まれ、更新結果が実際のデータに反映される

この動作のポイントは「**3**」と「**4**」です。トランザクション ログへの更新履歴の書き込みは即時に行われるのに対して、実際のデータへの反映には遅延が発生しています。この遅延は意図的なもので、データへの書き込みを遅らせて、チェックポイントというタイミングでまとめて書き込みを行うことで、ディスク アクセスを減らしています（パフォーマンスを向上させています）。

5.2　チェックポイント（Checkpoint）とは

チェックポイントとは

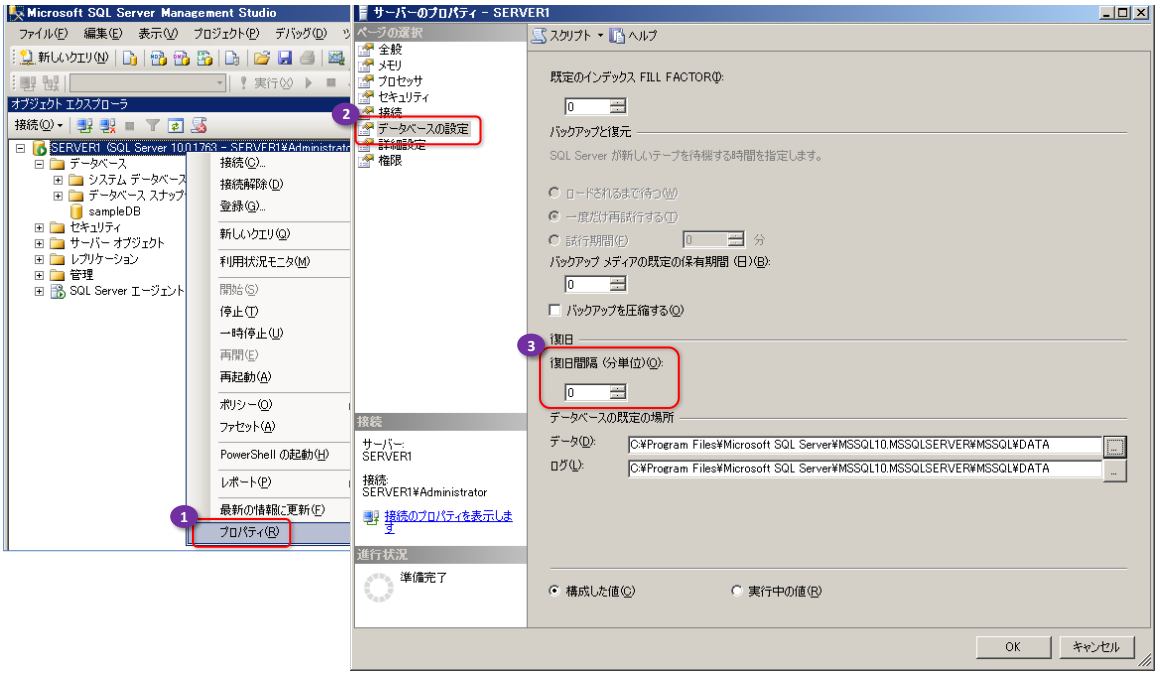
チェックポイントは、SQL Server が自動的に実行し、このときに実際のデータ（ディスク内の .mdf ファイル）への反映が行われます。このとき、ディスクへ反映されるデータは、完了（コミット）済みのトランザクションのみです。これは、次のような状況です。



チェックポイントが発生した時に実行中のトランザクションの結果は、ディスクへは反映されず、次回のチェックポイントが発生した時に反映されます。

チェックポイントが発生するタイミング

チェックポイントが発生するタイミングは、SQL Server の環境設定オプションの「**復旧間隔**」の設定によって決まります。この設定は、オブジェクト エクスプローラで、サーバー名を右クリックして、［プロパティ］をクリックし、［**データベースの設定**］ページで確認できます



このページの［**復旧間隔**］オプションが、チェックポイントのタイミングを制御するオプションになります。［復旧間隔］オプションは、分単位で設定しますが、ここで指定した時間がそのままチ

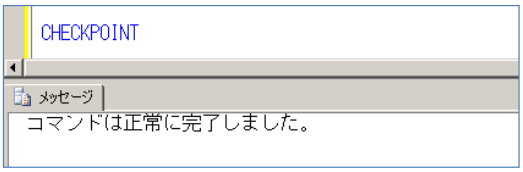
ェックポイントが発生する間隔になるというわけではありません。”復旧間隔”という名のとおり、このオプションにはデータベースの復旧（後述の自動復旧処理）にかかる時間を指定します。復旧にかかる時間は、障害前に実行されていたトランザクションの量によって変化します。

たとえば、復旧間隔オプションを「**5**」分に設定した場合、復旧にかかる時間が 5 分以内に収まるようにチェックポイントが発生します。絶え間なくトランザクションが実行されているような場合には、約 5 分ごとにチェックポイントが発生しますが、ほとんどトランザクションが実行されていないような場合には、チェックポイントはほとんど発生しません。

また、デフォルトでは、復旧間隔オプションが「**0**」へ設定されていますが、これは復旧にかかる時間が 1 分未満と設定されます。

Note： チェックポイントの手動実行

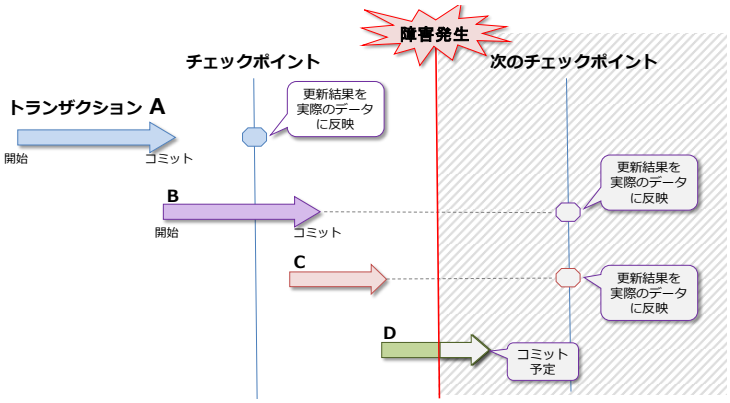
チェックポイントは、手動で実行することもできます。これを行うには、「**CHECKPOINT**」ステートメントを実行します。



5.3　自動復旧処理

自動復旧処理

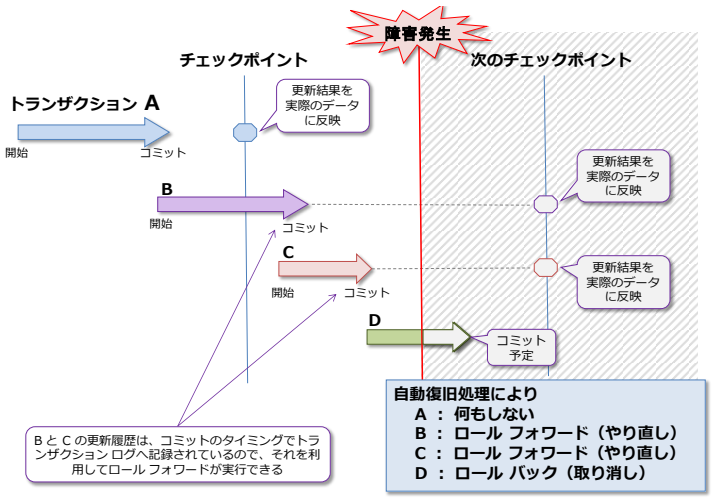
トランザクション ログへ書き込まれた更新履歴は、障害時のデータ復旧のために利用されます。前述したように、チェックポイントは、デフォルトでは、約 1 分ごと（トランザクションが絶えず実行されている場合）に発生しているので、その間に障害発生することは当然あり得ます。これは次のような状況です。



チェックポイントとチェックポイントの間で障害（停電など）が発生した場合は、実際のデータへの反映が行われていないトランザクションが存在することになります。これでは、データの一貫性が保たれないので、このような障害が発生した場合は、SQL Server は、障害から復旧した後の起動時に、次のように動作します。

1. トランザクション ログを読み、障害前に**コミット**済みのトランザクションを探す。
2. コミットされたトランザクションを見つけた場合には、ログに記録された更新履歴をもとにトランザクションを**ロール フォワード**する（やり直す）。
3. コミットされていないトランザクションを見つけた場合は、トランザクションを**ロールバック**する（取り消す）。

この動作は、「**自動復旧処理**」と呼ばれます。したがって、上の図のように障害が発生した場合は、トランザクション **B** と **C** がロール フォワードされ、**D** がロールバックされます。



このように、トランザクション ログは、障害時の自動復旧処理に利用されます。

自動復旧処理で対応可能な障害

自動復旧処理で対応可能な障害は、障害からの復旧後に OS と SQL Server が正常に動作するもので、主に次のとおりです。

- 停電などの電源断
- SQL Server の異常終了
- OS の異常終了（ブルー スクリーン）
- ディスクを除いたハードウェア障害

これらの障害が発生しても、基本的には OS と SQL Server が動作します（動作しない場合は、自動復旧処理では対応不可能な障害になります）。4 つ目の「ディスクを除いたハードウェア障害」は、障害のあったハードウェアを新しいものへ交換すれば、OS と SQL Server が正常に動作するハードウェアのことを指しています。たとえば、メモリや CPU、ネットワーク カード、マザーボードなどがあります。これらは、故障しても新しいものと交換すれば、基本的には OS と SQL Server が正常に動作します。

自動復旧処理が動作すれば、障害が発生する前にコミットされていたトランザクションは、ロールフォワード（やり直し）をすることで、障害発生の直前の状態まで復旧することができます。

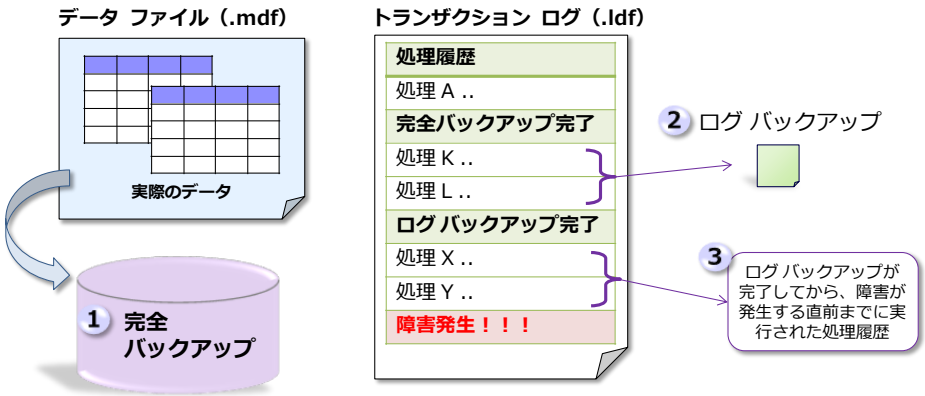
5.4　障害発生直前までの復旧

障害発生直前までの復旧

前述の自動復旧処理は、障害からの復旧後に OS と SQL Server が動作するという条件がありましたが、データ ファイル（.mdf）が格納されているディスクが破損しても、障害が発生する直前まで復旧することが可能です。この場合の条件は、次の 3 つです。

- 完全バックアップ（必要に応じてログ バックアップ）が実行されていること
- トランザクション ログが破損していないこと
- 後述の復旧モデルが「**完全**」であること

バックアップを実行した時点ではなく、障害が発生した直前まで復旧できるところがポイントです。これは、バックアップとトランザクション ログを組み合わせることで実現しています。

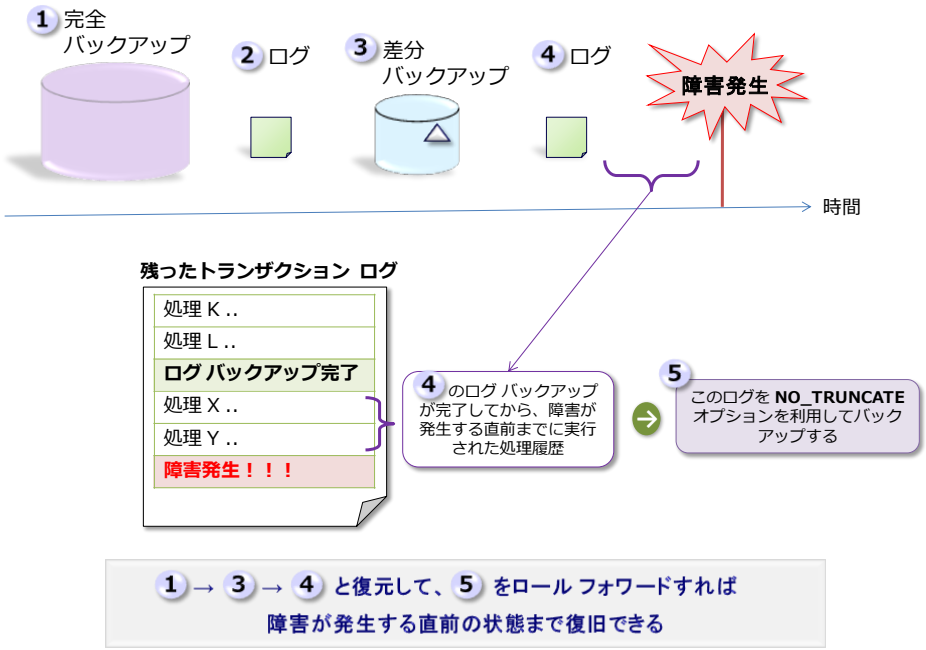


1 と 2 を復元して、3 をロール フォワードすれば
障害が発生する直前の状態まで復旧できる

トランザクション ログには、バックアップが完了してから障害が発生する直前までの処理の履歴が格納されているので、この処理をロール フォワード（やり直し）をすることで、障害が発生した直前までデータを復旧することができます。

復旧手順： NO_TRUNCATE オプションの利用

障害が発生する直前までデータを復旧するには、まず最初に、残ったログ（まだバックアップしていないログ）を **NO_TRUNCATE** オプションを利用してバックアップする必要があります。これは、次のような状況です。

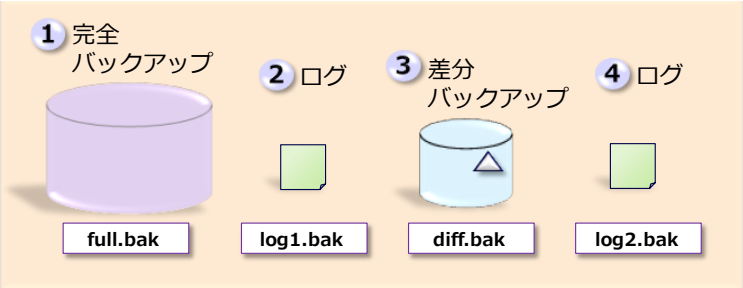


障害が発生した場合は、まず、残ったトランザクション ログを **NO_TRUNCATE** オプションを使用してバックアップします。このオプションは、ログを切り捨てない（truncate は切り捨てという意味）という効果があります。

あとは、通常通りにバックアップから復元し（NORECOVERY オプションを付けてリストアし）、最後に、NO_TRUNCATE オプションを使用してバックアップしたログ バックアップをリストアすれば、障害発生直前まで復旧することができます。

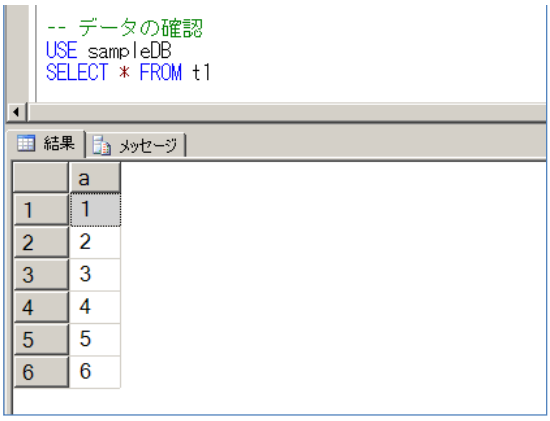
Let's Try

それでは、これを試してみましょう。ここでは、前の Step で実行したバックアップを利用して、障害発生直前までの復旧を試してみます。



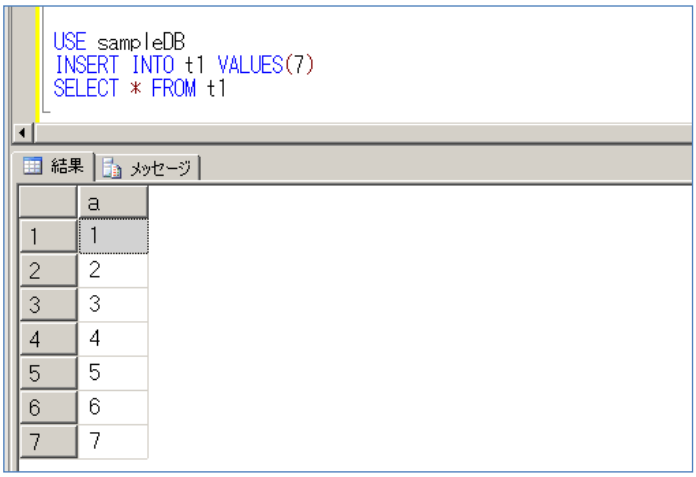
1. まずは、現在のデータを確認しておきましょう。

```
USE sampleDB
SELECT * FROM t1
```



2. 次に、障害発生直前のデータ復旧を試すために、データを 1 件追加しておきます。

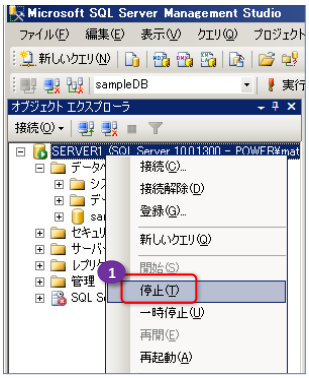
```
USE sampleDB
INSERT INTO t1 VALUES(7)
SELECT * FROM t1
```



障害シミュレート（データ ファイルの破損）

次に、データ ファイル（.mdf）の破損をシミュレートしてみましょう。

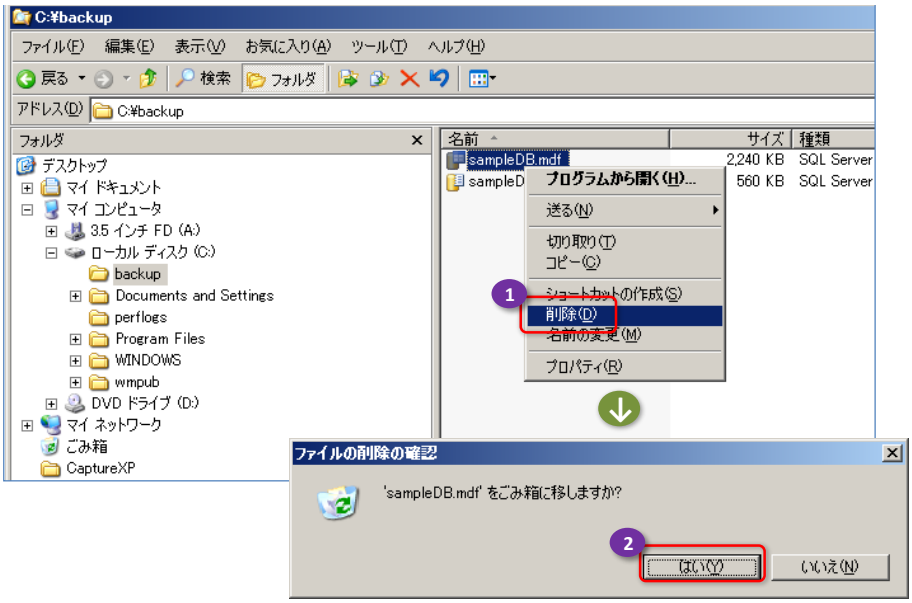
1. ファイルの破損をシミュレートには、まず SQL Server サービスを停止します。SQL Server を停止するには、オブジェクト エクスプローラでサーバー名を右クリックして［**停止**］をクリックします。



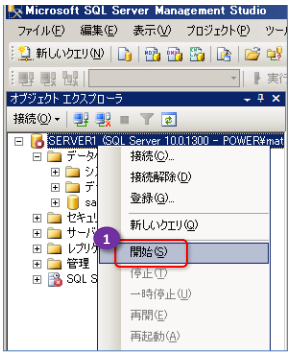
2. 次に、Windows エクスプローラを起動して、「**sampleDB**」データベースの**データ ファイル（.mdf）**が格納されている以下のフォルダを開きます。

   C:¥backup

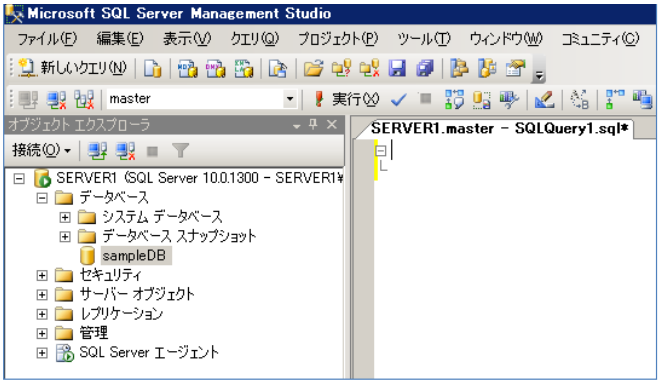
3. このフォルダ内の「**sampleDB.mdf**」ファイルを右クリックして、［**削除**］をクリックし、ファイルを削除します。



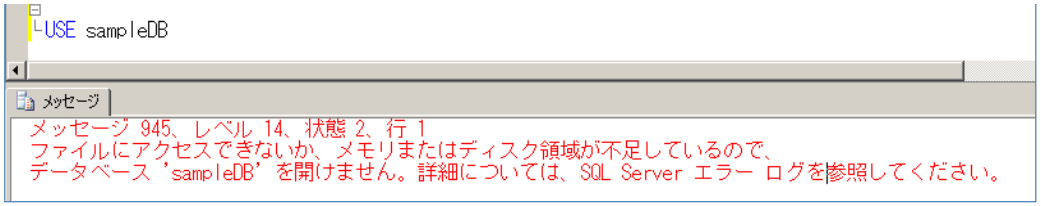
4. 削除後、SQL Server サービスを開始します。SQL Server を開始するには、オブジェクト エクスプローラでサーバー名を右クリックして［**開始**］をクリックします。



5. サービスが開始されたら、「**sampleDB**」データベースをダブル クリックしても中身を参照できないことを確認できます。



6. また、クエリ エディタから「**USE**」ステートメントでデータベースへ接続しようとしても、次のエラーが発生することを確認できます。

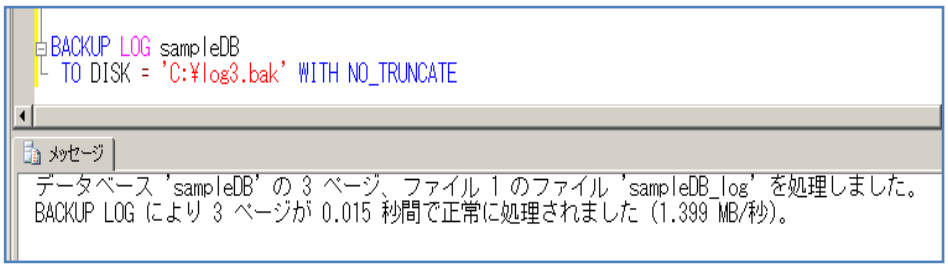


障害発生直前までの復旧

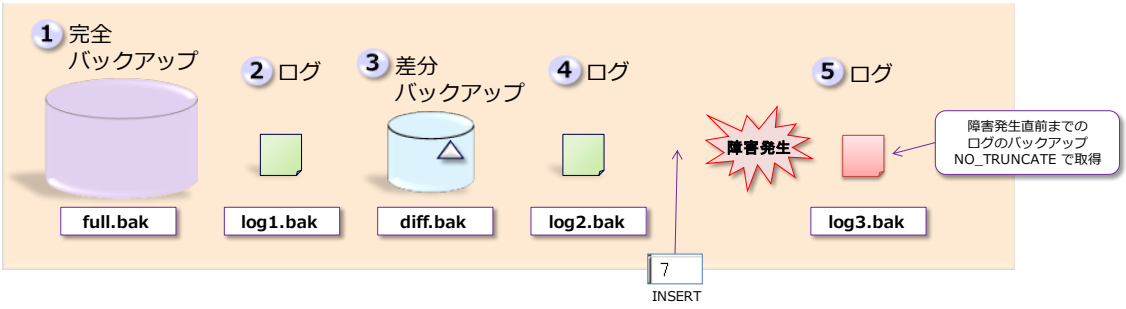
障害が発生する直前まで復旧をするには、まず **NO_TRUNCATE** オプションを使用して、トランザクション ログのバックアップを実行します。

1. 次のように、ログ バックアップを実行します。

```
BACKUP LOG sampleDB
 TO DISK = 'C:¥log3.bak'
  WITH NO_TRUNCATE
```



ここまで取得したバックアップをまとめると、次のようになります。

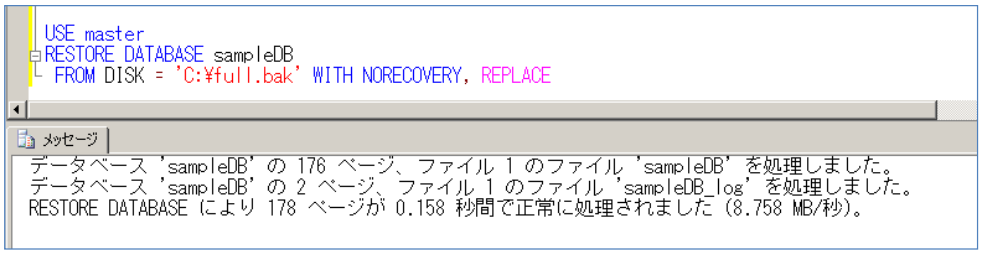


データベースの復元

続いて、今まで取得したバックアップをリストアして、障害が発生する直前まで復旧していきましょう。リストア時は、最後のバックアップをリストアするまでは、**NORECOVERY** オプションを指定することに注意してください。

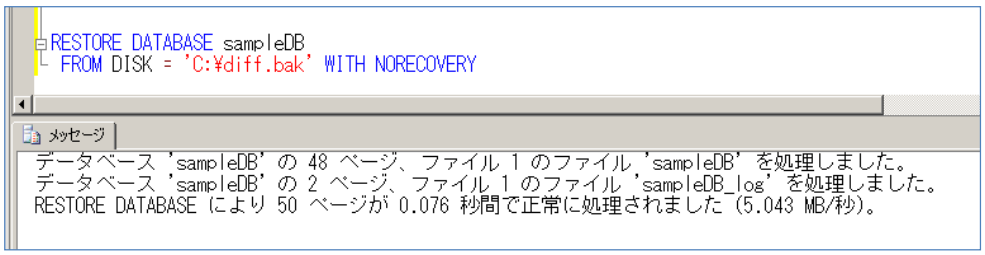
2. まずは、完全バックアップ「**full.bak**」を NORECOVERY オプションを指定してリストアします。

```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥full.bak'
  WITH NORECOVERY, REPLACE
```



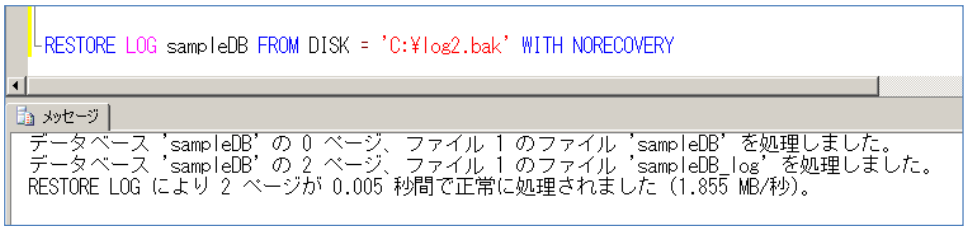
3. 次に、差分バックアップ「**diff.bak**」を NORECOVERY オプションを指定して、リストアします。

```
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥diff.bak'
  WITH NORECOVERY
```



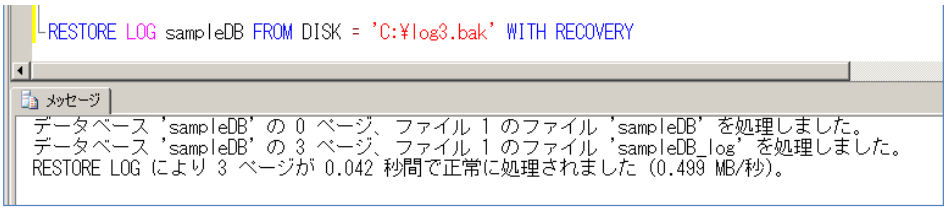
4. 次に、ログ バックアップ「**log2.bak**」を NORECOVERY オプションを指定してリストアします。

```
RESTORE LOG sampleDB
 FROM DISK = 'C:¥log2.bak'
  WITH NORECOVERY
```



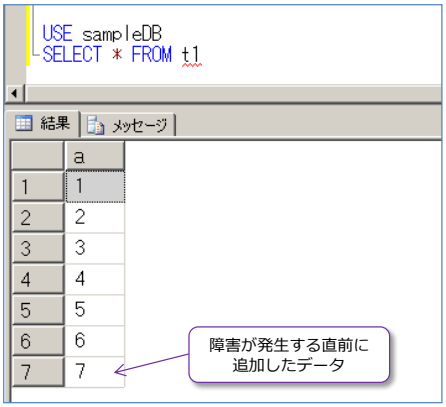
5. 最後に、**NO_TRUNCAE** オプションを使用して取得したログ バックアップ「**log3.bak**」をリストアします。これはリストアする最後のバックアップなので、**RECOVERY** オプションを指定します。

```
RESTORE LOG sampleDB
 FROM DISK = 'C:¥log3.bak'
  WITH RECOVERY
```



6. リストア後、障害が発生する直前に追加したデータ「**7**」が復元されていることを確認しておきましょう。

```
USE sampleDB
SELECT * FROM t1
```



このように、データ ファイル（.mdf）へ障害が発生しても、トランザクション ログが残っていれば、障害が発生する直前の状態まで、復元することができます。

もう一度、障害発生直前まで復旧するための条件の 3 つを確認しておきましょう。

- 完全バックアップ（必要に応じてログ バックアップ）が実行されていること
- トランザクション ログ（.ldf）が破損していないこと
- 後述の復旧モデルが「**完全**」であること

この機能は、あくまでもバックアップと残存したログを組み合わせた機能であることに注意してください。また、復旧モデルが「**完全**」（Standard Edition 以上のデフォルト値）である必要もあります。これについては、次の項で説明します。

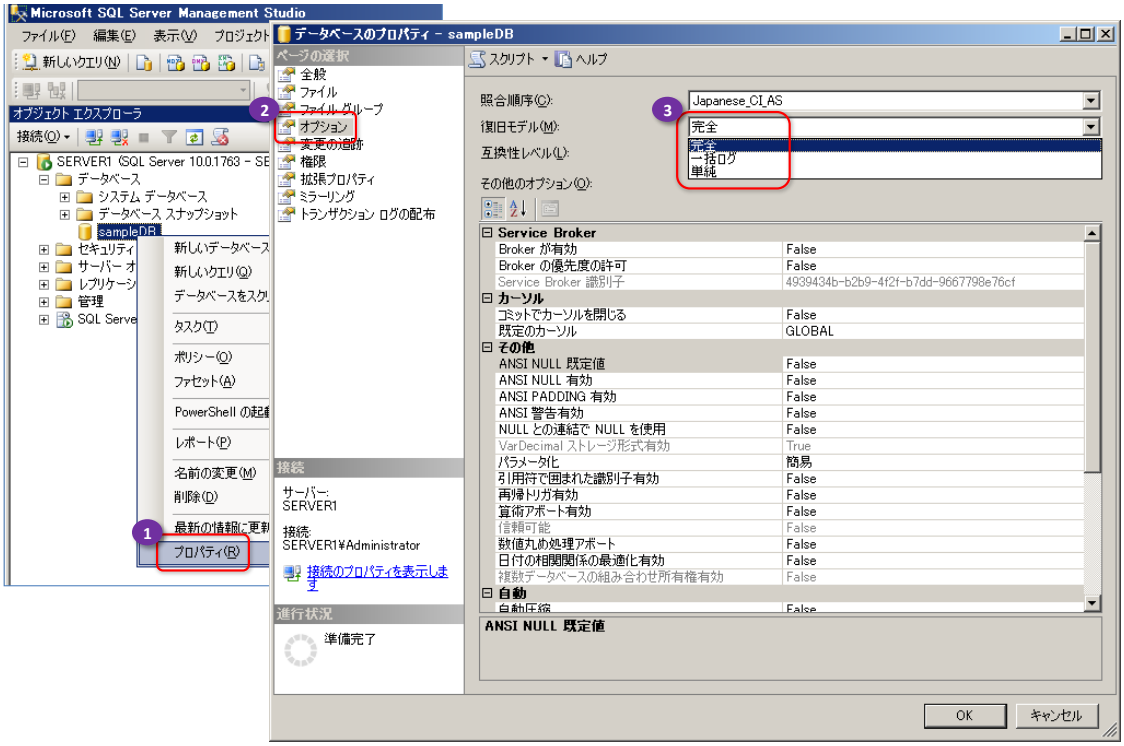
また、「**トランザクション ログが破損していない**」という条件も非常に重要です。これを実現するには、データ ファイル（.mdf）とログ ファイル（.ldf）を別々のディスク（別々の RAID セット）へ配置しておく必要があります。同じディスクだと、ディスクが壊れた場合に、両方とも破損してしまうからです。また、このように別々のディスクへ配置しておくことは、パフォーマンス向上というメリットも得られます。ディスクへの書き込みを分散できるからです。

したがって、障害対策およびパフォーマンス向上を考慮して、データ ファイル（.mdf）とログ ファイル（.ldf）を別々のディスク（別々の RAID セット）へ配置しておくことをお勧めします。

5.5　復旧モデルとトランザクション ログ

復旧モデルとトランザクション ログ

トランザクション ログの内部動作は、データベースの「**復旧モデル**」の設定によって変化します。復旧モデルには、「**完全**」と「**一括ログ**」、「**単純**」の 3 種類があり、現在の復旧モデルを確認／変更するには、次のように操作して、データベースのプロパティを開きます。



SQL ステートメントを使用して復旧モデルを設定する場合は、**ALTER DATABASE** ステートメントを次のように実行します。

```
USE master
ALTER DATABASE データベース名
SET RECOVERY { FULL | BULK_LOGGED | SIMPLE }
```

復旧モデルの違い

3 つの復旧モデルの違いは、次のとおりです。

- **完全（Full）モデル**

  完全モデルは、トランザクション ログへすべての処理履歴を完全に記録するモデルです。これは、Standard Edition 以上のデフォルトの復旧モデルです。このモデルでは、前の手順で試したように、障害発生時に障害が発生した直前までデータを復旧することができます。また、時刻を指定した復旧も可能で、ほとんどの環境で最適なモデルです。

- **一括ログ（Bulk Logged）モデル**

  一括ログ モデルは、一括（bulk：バルク）操作（インデックスの作成や再構築、bcp コマンド、BULK INSERT ステートメント、Integration Services パッケージ、SELECT INTO など）のパフォーマンスを向上するために、トランザクション ログへ記録する情報を最小限に抑えるモデルです（この動作は、**最小ログ記録**とも呼ばれます）。

  しかし、ログへ記録される情報が少なくなるということは、障害発生時に一部の操作の復旧ができない可能性があることを意味します。また、このモデルでは、時刻を指定した復旧もできません。障害復旧よりも一括操作のパフォーマンス向上を重視したい場合には、このモデルを使用します。

- **単純（Simple）モデル**

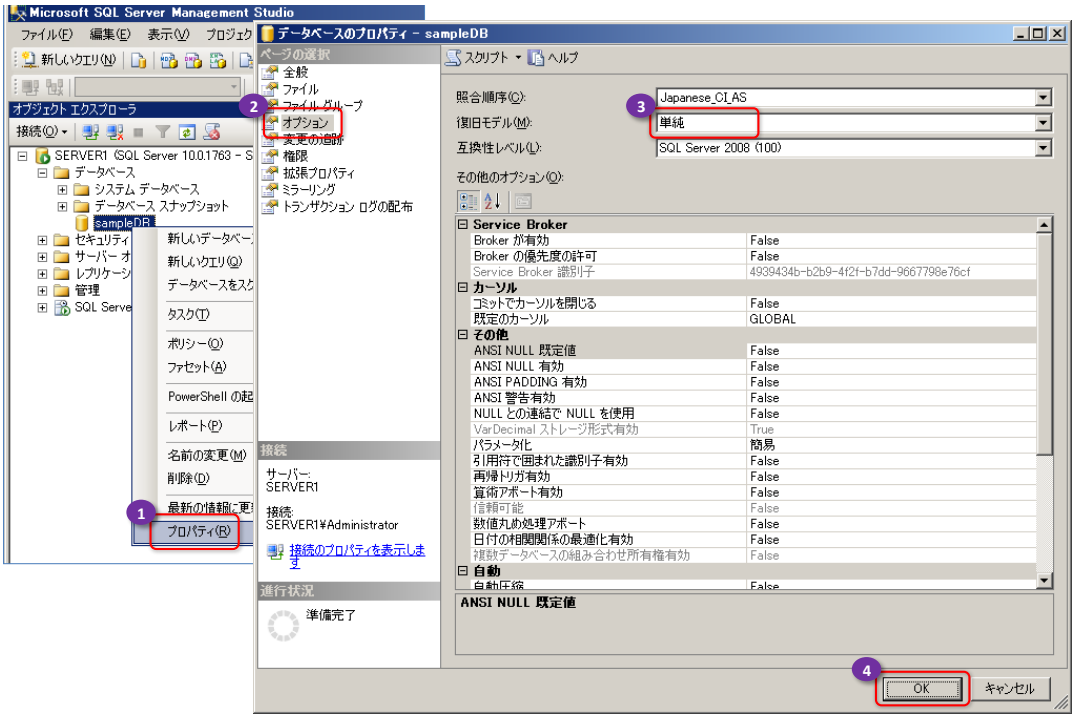
  単純モデルは、SQL Server 7.0 以前のバージョンでは「**チェックポイント時のログ切り捨て**」（trunc. log on chkpt）と呼ばれていました。その名のとおり、チェックポイントが完了するごとに、現在実行されているトランザクションを除いて、トランザクション ログを切り捨てます。これによって、トランザクション ログの使用領域を最小に抑えて、ログの肥大化を防ぐことができます。

  このモデルでも、自動復旧機能は正常に動作しますが、障害発生時に障害が発生した直前までデータを復旧することはできません。また、トランザクション ログのバックアップも実行することができません。したがって、開発時などディスク領域を節約したい場合や、障害時にバックアップを取得した時点までの復元ができれば良い場合にのみ利用するようにします。

Let's Try： 復旧モデルを「単純モデル」へ変更

それでは、復旧モデルを試してみましょう。ここでは、**sampleDB** データベースの復旧モデルを「**完全モデル**」（デフォルト）から「**単純モデル**」へ変更してみましょう。

1. 復旧モデルを変更するには、次のようにオブジェクト エクスプローラで、sampleDB 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、［**オプション**］ページを開きます。



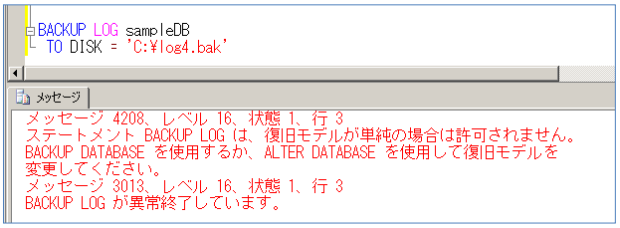
［オプション］ページでは、［**復旧モデル**］で「**単純**」を選択して、［**OK**］ボタンをクリックします。これで、復旧モデルの変更が完了です。

単純モデルとログ バックアップ

次に、単純モデルへ変更した sampleDB データベースに対して、ログ バックアップを実行してみましょう。

2. クエリ エディタで、次のように入力して、ログ バックアップを実行します。

```
BACKUP LOG sampleDB
 TO DISK = 'C:¥log4.bak'
```



結果は、「復旧モデルが単純の場合は許可されません」とエラーが表示されて、ログ バックアップが失敗することを確認できます。このように、単純復旧モデルでは、ログ バックアップを実行することができません。

5.6　一括ログ モデルの障害直前までの復旧

一括ログ モデルの障害直前までの復旧

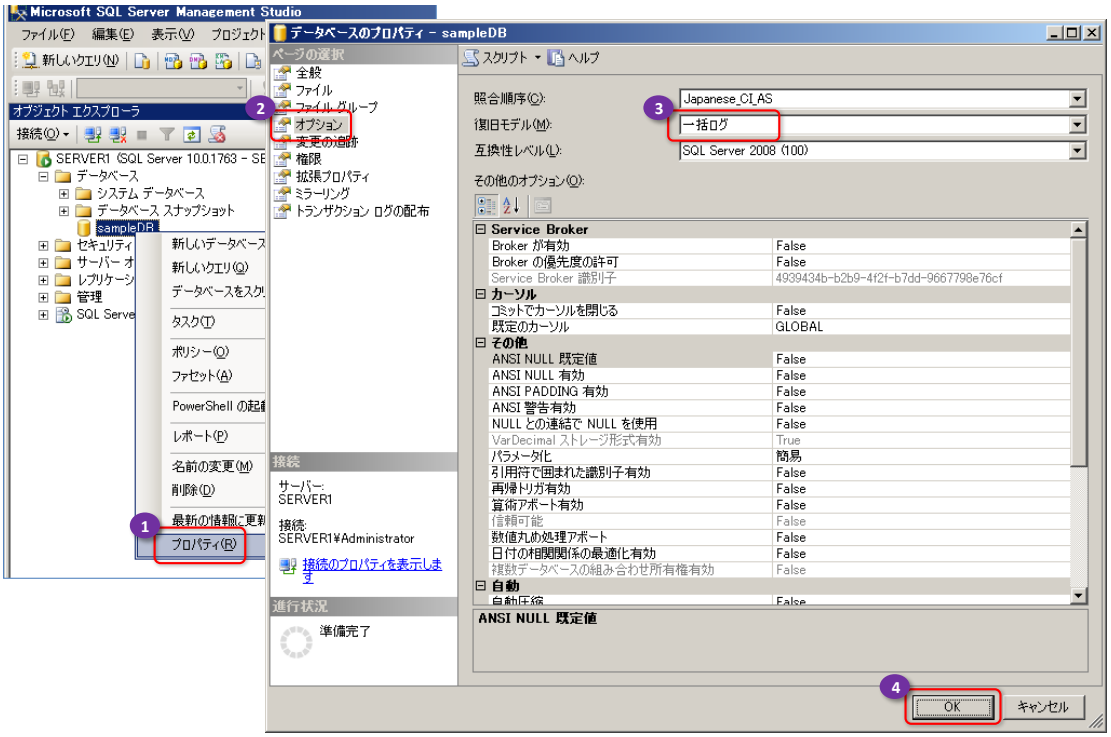
一括ログ モデルは、前述したように、一括操作のパフォーマンスを向上させたい場合に利用する復旧モデルです（一括操作の実行履歴を、ログへほとんど記録しないようにすることで、パフォーマンスを向上させています）。しかし、ログへの記録を最小限に抑えることによって、障害発生時への復旧へは次の制限がかかります。

- バックアップ以降に一括操作が実行された場合は、障害が発生する直前まで復旧できない
- 日付と時刻を指定した復旧が行えない（STOPAT オプションが利用できない）

Let's Try： 一括ログ モデルへの変更

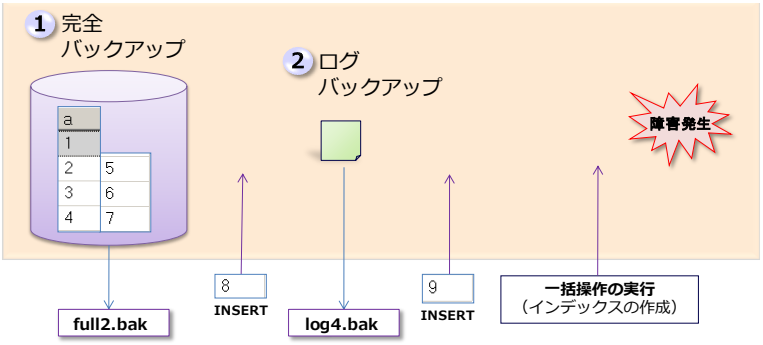
それでは、これを試してみましょう。sampleDB データベースの復旧モデルを**一括ログ モデル**へ変更し、一括操作が実行された場合に、障害発生直前まで復旧できないことを確認してみましょう。

1. まずは、sampleDB データベースの復旧モデルを「一括ログ」へ変更します。これは、次のように操作します。



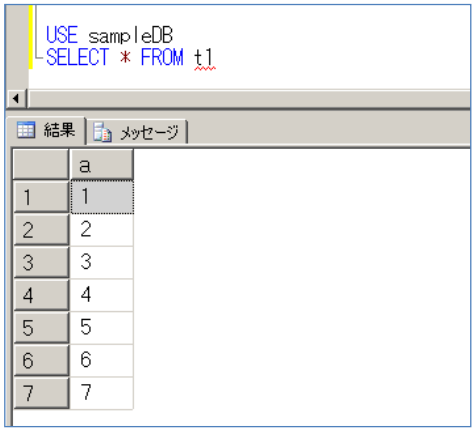
一括ログ モデルの動作確認

次に、一括ログ モデルへ変更した sampleDB データベースに対して、次の順番で操作していきましょう。



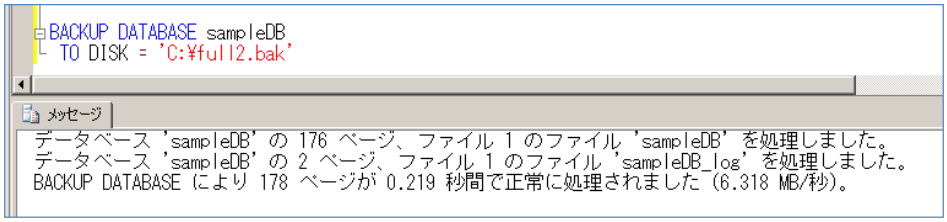
2. まずは、バックアップ前のデータを確認しておきます。

```
USE sampleDB
SELECT * FROM t1
```



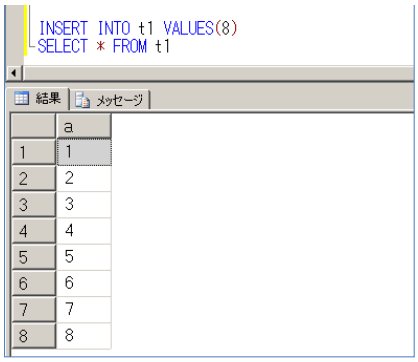
3. 次に、**BACKUP DATABASE** ステートメントを利用して、**完全バックアップ**を実行します。次のようにパスを指定して、**C:** ドライブの直下へ「**full2.bak**」という名前でバックアップを取得します。

```
BACKUP DATABASE sampleDB
 TO DISK = 'C:¥full2.bak'
```



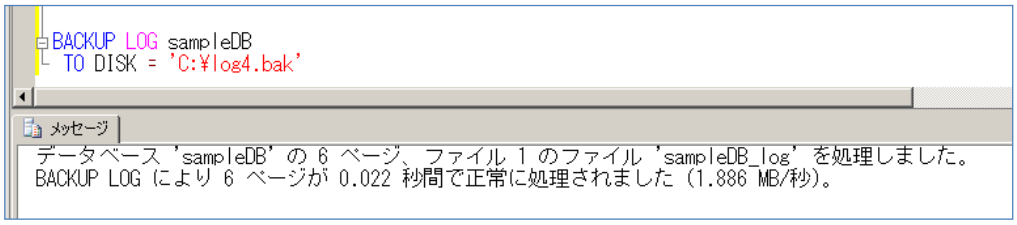
4. 完全バックアップが完了したら、「**t1**」テーブルへデータを 1 件追加します。

```
INSERT INTO t1 VALUES(8)
SELECT * FROM t1
```



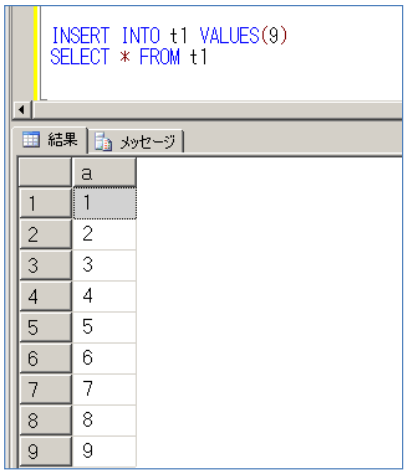
5. 次に、**BACKUP LOG** ステートメントを利用して、**ログ バックアップ**を実行します。次のようにパスを指定して、**C:** ドライブの直下へ「**log4.bak**」という名前でバックアップを取得します。

```
BACKUP LOG sampleDB
 TO DISK = 'C:¥log4.bak'
```



6. ログ バックアップが完了したら、「**t1**」テーブルに対してデータをもう 1 件追加します。

```
INSERT INTO t1 VALUES(9)
SELECT * FROM t1
```

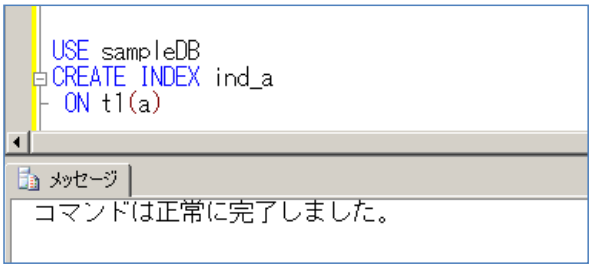


一括操作の実行

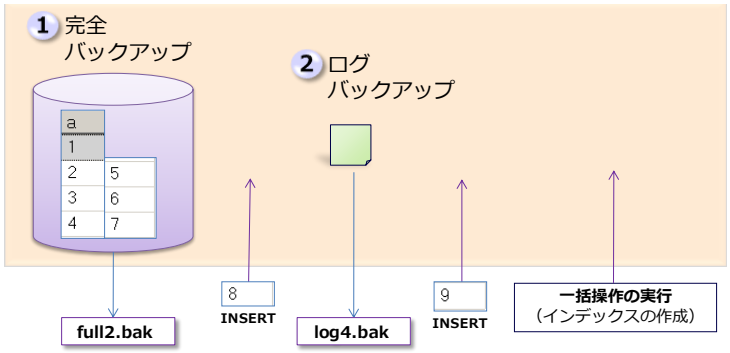
続いて、sampleDB データベースに対して、一括操作を実行してみましょう。

7. 次のように **CREATE INDEX** ステートメントを実行して、インデックスを作成します（インデックスの作成は、一括操作です）。

```
USE sampleDB
CREATE INDEX ind_a
 ON t1(a)
```



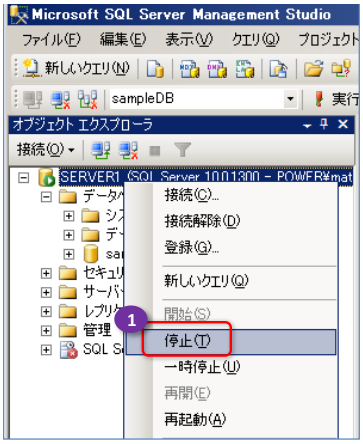
ここまでの操作をまとめると、次のようになります。



障害シミュレート

次に、データ ファイル（.mdf）の破損をシミュレートしてみましょう。

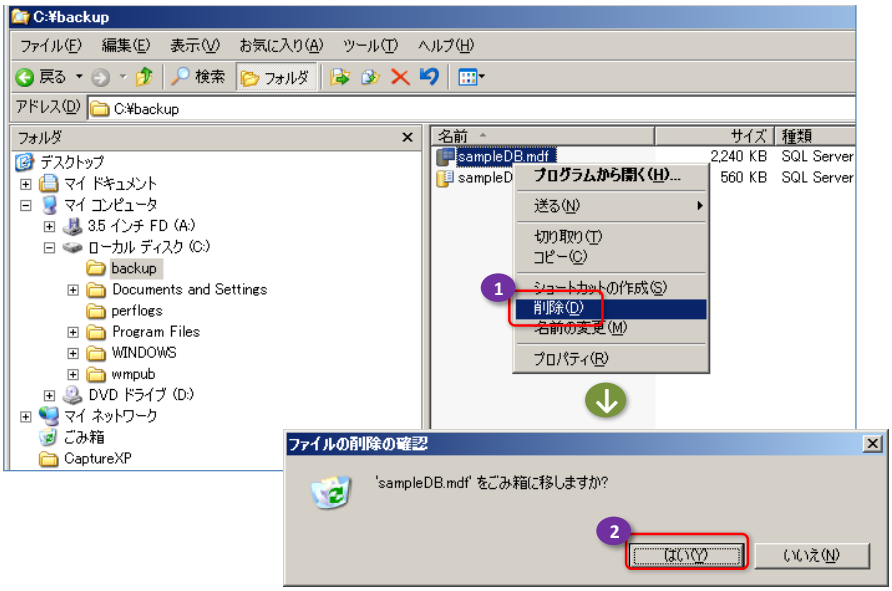
1. 破損をシミュレートには、まず SQL Server サービスを停止します。



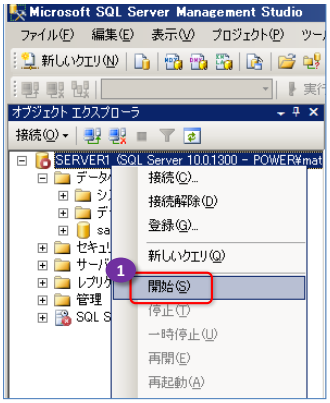
2. 次に、Windows エクスプローラを起動して、「**sampleDB**」データベースの**データ ファイル（.mdf）**が格納されている以下のフォルダを開きます。

   C:¥backup

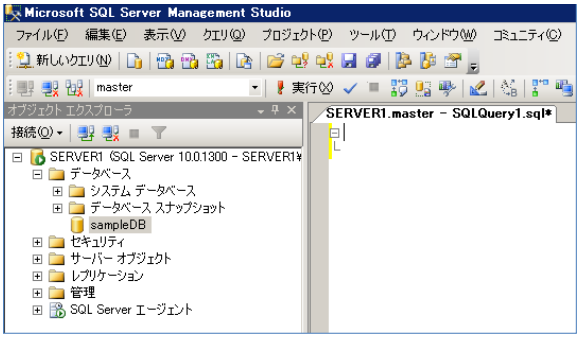
3. このフォルダ内の「**sampleDB.mdf**」ファイルを右クリックして、［**削除**］をクリックし、ファイルを削除します。



4. 削除後、SQL Server サービスを開始します。



5. サービスが開始されたら、**sampleDB** データベースの中身を参照できないことを確認します。



一括ログ モデルでの障害発生直前までの復旧

1. 次に、障害発生直前まで復旧するための最初の手順である、**NO_TRUNCATE** オプションを指定した、トランザクション ログのバックアップを実行してみましょう。

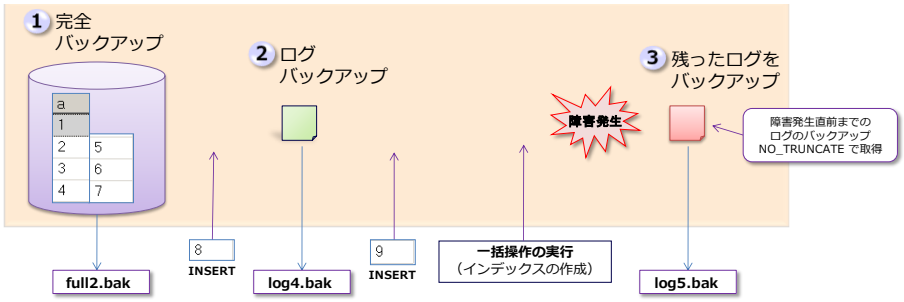
```
USE master
BACKUP LOG sampleDB TO DISK = 'C:¥log5.bak'
 WITH NO_TRUNCATE
```

```
USE master
BACKUP LOG sampleDB
 TO DISK = 'C:¥log5.bak' WITH NO_TRUNCATE
```

```
メッセージ
データベース 'sampleDB' の 5 ページ、ファイル 1 のファイル 'sampleDB_log' を処理しました。
BACKUP WITH CONTINUE_AFTER_ERROR により、破損したデータベースのバックアップが正しく生成されました。
発生したエラーの詳細については、SQL Server エラー ログを参照してください。
BACKUP LOG により 5 ページが 0.022 秒間で正常に処理されました (1.509 MB/秒)。
```

結果には、いつもと違うメッセージが表示されていることを確認できます。これは、一括操作によって行われた変更が、正しくバックアップできていないという主旨のメッセージです。

ここまでの操作をまとめると、次のようになります。



2. 続いて、これらのバックアップをリストアしていきましょう、まずは、**NORECOVERY** オプションを指定して、完全バックアップ「**full2.bak**」をリストアします。

```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥full2.bak'
  WITH NORECOVERY, REPLACE
```

```
USE master
RESTORE DATABASE sampleDB
 FROM DISK = 'C:¥full.bak' WITH NORECOVERY, REPLACE
```

```
メッセージ
データベース 'sampleDB' の 176 ページ、ファイル 1 のファイル 'sampleDB' を処理しました。
データベース 'sampleDB' の 2 ページ、ファイル 1 のファイル 'sampleDB_log' を処理しました。
RESTORE DATABASE により 178 ページが 0.158 秒間で正常に処理されました (8.758 MB/秒)。
```

3. 次に、NORECOVERY オプションを利用して、ログ バックアップ「**log4.bak**」をリストアします。

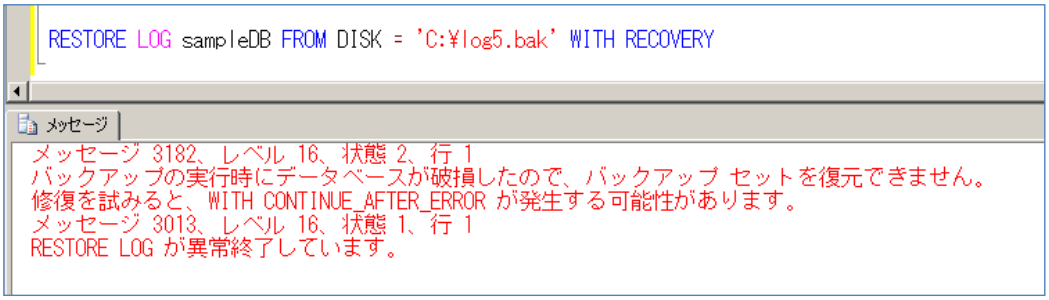
```
RESTORE LOG sampleDB
 FROM DISK = 'C:¥log4.bak'
  WITH NORECOVERY
```

```
RESTORE LOG sampleDB FROM DISK = 'C:¥log2.bak' WITH NORECOVERY
```

```
メッセージ
データベース 'sampleDB' の 0 ページ、ファイル 1 のファイル 'sampleDB' を処理しました。
データベース 'sampleDB' の 2 ページ、ファイル 1 のファイル 'sampleDB_log' を処理しました。
RESTORE LOG により 2 ページが 0.005 秒間で正常に処理されました (1.855 MB/秒)。
```

4. 最後に、NO_TRUNCAE オプションを使用して取得したログ バックアップ「**log5.bak**」をリストアします。これは、**RECOVERY** オプションを指定して実行します。

```
RESTORE LOG sampleDB
 FROM DISK = 'C:¥log5.bak'
  WITH RECOVERY
```



結果は、エラーになって、リストアが失敗します。このように一括ログ モデルの場合は、ログ バックアップ以降に一括操作が実行された場合、障害発生直前まで復旧できないという制限があります。

Note： 一括ログ モデルと完全モデルを併用する

一括ログ モデルと完全モデルは、切り替えて運用していくことが可能です（切り替え時にサービスを再開始する必要もなく、またバックアップ操作に影響なく切り替えが可能です）。
したがって、夜間などの一括操作（インデックスの再構築や Bulk Insert 処理など）が多い時間帯では、一括ログ モデルを利用し、日中は完全モデルを利用する、という運用が可能です。一括ログ モデルは、一括操作のパフォーマンスを向上させる場合に大変役立つモデルなので、ぜひ活用してみてください。

STEP 6. その他の TIPS

この STEP では、開発機から本番機へ、あるいはその逆へデータベースを移動する場合（別のマシンへデータベースを移動する場合）の注意点と、システム データベースのオンライン バックアップ／復元について説明します。

この STEP では、次のことを学習します。

- ✓ 別マシンへのデータベース移動時の注意点
- ✓ ログイン アカウントの複製（不明なデータベース ユーザー）
- ✓ システム データベースのオンライン バックアップ／復元

6.1　別マシンへのデータベース移動時の注意点

別マシンへのデータベース移動時の注意点

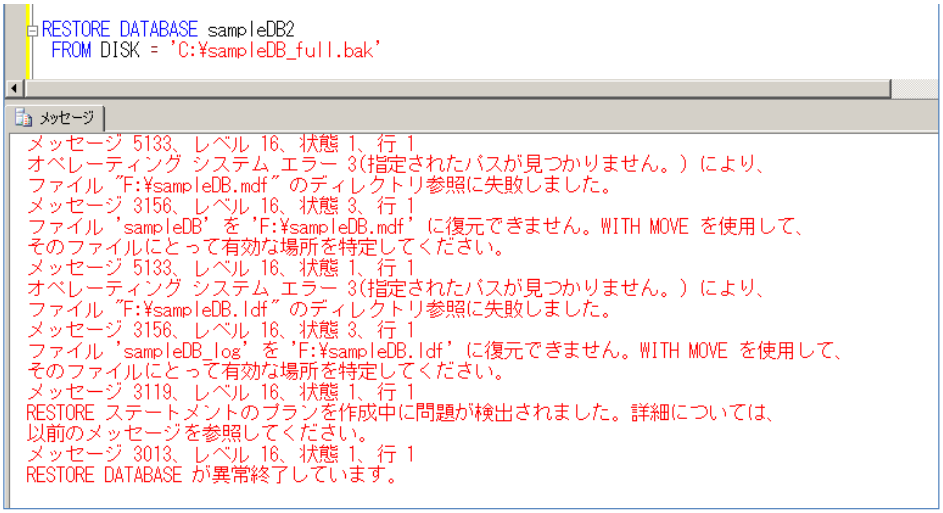
Step 2.2 の「オフライン バックアップを別のマシンへ復元（アタッチ）」で説明したように、別マシンへデータベースを移動する場合には、オフライン バックアップを利用できます。

また、オンライン バックアップを利用しても、同じように移動することができます。この場合は、**BACKUP** ステートメントでバックアップを取得し、それを移動先のマシンで **RESTORE** ステートメントでリストアすれば完了です。オフライン バックアップ（アタッチ機能）と比較して、ファイル サイズ（移動するファイルのサイズ）を小さくできる点がメリットです。

復元場所を変更する MOVE .. TO オプション

別のマシンへデータベースを移動する場合は、RESTORE ステートメントが、もともとデータベースが配置してあった場所（BACKUP ステートメントでバックアップを実行したときに配置されていた場所）へリストアしようとする点に注意する必要があります。マシンが異なると、ドライブ名が違ったり、フォルダ構成が違ったりするからです。

たとえば、元々データベースを作成してあった場所が「**F:**」ドライブで、移動先の別のマシンには「**F:**」が存在しなかったとすると、この場合は、次のエラーが発生します。



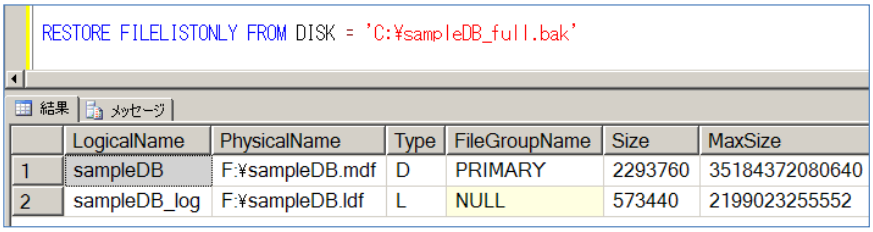
エラー内の「**指定されたパスが見つかりません**」というキーワードがポイントです。また、「**WITH MOVE を使用して、～有効な場所を指定してください**」というキーワードもあります。この MOVE オプションを利用すると、復元先を変更してリストアを実行することができます。

したがって、このエラーを回避するには、次のように MOVE .. TO オプションを利用します。

```
RESTORE DATABASE データベース名
  FROM DISK='パス'
    WITH  MOVE 'データファイルの論理名' TO '新ファイルパス'
         ,MOVE 'ログファイルの論理名'   TO '新ファイルパス'
```

論理名は、データベースの作成時に設定したものですが、デフォルトではデータ ファイルは「データベース名」、トランザクション ログ ファイルは「データベース名_Log」という名前です。また、論理名は、次のように **RESTORE FILELISTONLY** ステートメントを使用して確認することもできます。

```
RESTORE FILELISTONLY
 FROM DISK = 'C:¥sampleDB_full.bak'
```



```
RESTORE FILELISTONLY FROM DISK = 'C:¥sampleDB_full.bak'
```

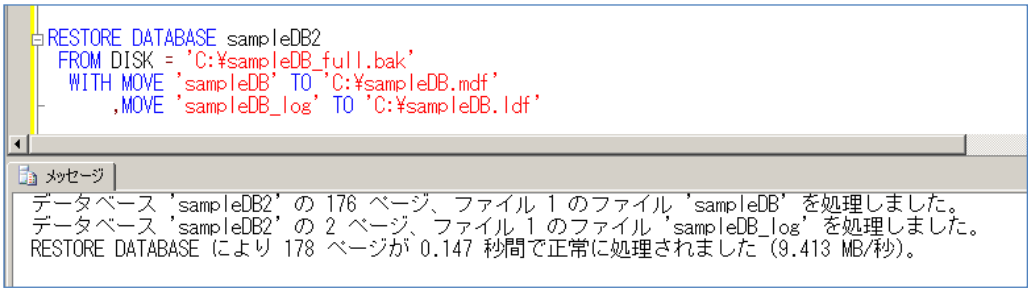
結果 | メッセージ

	LogicalName	PhysicalName	Type	FileGroupName	Size	MaxSize
1	sampleDB	F:¥sampleDB.mdf	D	PRIMARY	2293760	35184372080640
2	sampleDB_log	F:¥sampleDB.ldf	L	NULL	573440	2199023255552

結果のうち、LogicalName が論理名です。

これを利用して、MOVE .. TO オプションを利用すると次のようになります。

```
RESTORE DATABASE sampleDB2
 FROM DISK = 'C:¥sampleDB_full.bak'
  WITH MOVE 'sampleDB' TO 'C:¥sampleDB.mdf'
      ,MOVE 'sampleDB_log' TO 'C:¥sampleDB.ldf'
```

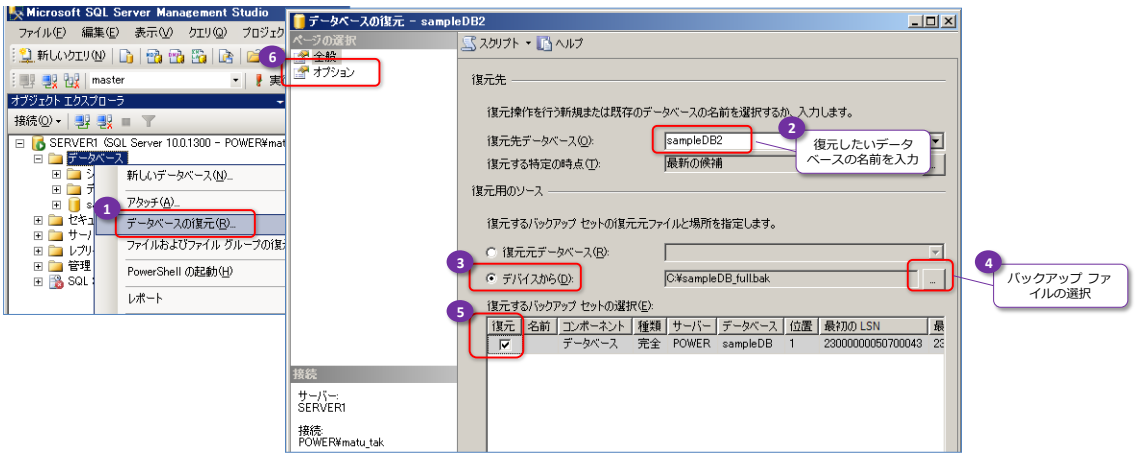


```
RESTORE DATABASE sampleDB2
 FROM DISK = 'C:¥sampleDB_full.bak'
  WITH MOVE 'sampleDB' TO 'C:¥sampleDB.mdf'
      ,MOVE 'sampleDB_log' TO 'C:¥sampleDB.ldf'
```

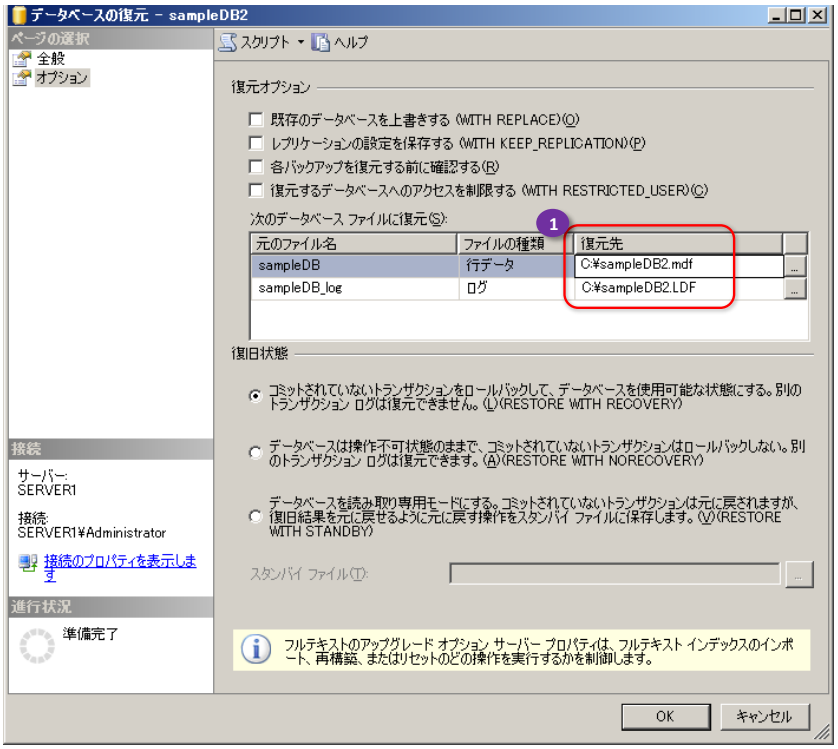
メッセージ

```
データベース 'sampleDB2' の 176 ページ、ファイル 1 のファイル 'sampleDB' を処理しました。
データベース 'sampleDB2' の 2 ページ、ファイル 1 のファイル 'sampleDB_log' を処理しました。
RESTORE DATABASE により 178 ページが 0.147 秒間で正常に処理されました (9.413 MB/秒)。
```

また、GUI から復元先を変更したい場合は、次のように操作します。



［**オプション**］ページでは、次のように［**復元先**］を設定します。



データベース移動時の注意点

Step 2.2 でも説明したように、データベースの移動で複製できるのは、あくまでもデータベース内のオブジェクトのみです（次の表のとおり）。

複製可能なオブジェクト	
・テーブル	・ストアド プロシージャ
・制約	・ユーザー定義関数
・インデックス	・ユーザー定義データ型
・トリガ	・SQLCLR オブジェクト ストアド プロシージャ ユーザー定義関数 ユーザー定義データ型 集計関数
・ビュー	
・シノニム	
・データベース ユーザー	
・スキーマ	・データ パーティション
・データベース ロール	・フルテキスト インデックス
・アプリケーション ロール	・データベース対称キー / 非対称キー / 証明書

ログイン アカウントや環境設定オプション（sp_configure で設定したもの）、ジョブ、警告など、システム データベースへ格納される情報と、レジストリへ格納される情報については、データベースの移動だけでは複製できないので、別途複製しなければなりません。複製できないものは、主に次のとおりです。

master	msdb	レジストリ
・ログイン アカウント ・環境設定オプション（sp_configure） ・リンク サーバー設定 ・ユーザー定義エラー	・ジョブ ・警告 ・オペレータ ・Integration Services パッケージ	・セキュリティ モード（認証モード） ・TCP ポート番号

これらのシステム データベースの情報は、”同じ名前” の別マシンへ移動する場合（ハードウェア リプレイス時）は、オフライン バックアップで丸ごと移動することができますが、別のマシンへ

移動する場合には、基本的に手動で複製する必要があります。ジョブや警告などは、スクリプト生成機能があるので、これを利用して移動することができます。

したがって、開発機から本番機へ、あるいはその逆へデータの移動をスムーズに行うには、それぞれのオブジェクトの作成をスクリプト ベース（SQL ステートメント）で行って、それを移動先で再実行するように運用／管理することをお勧めします。なお、Management Studio では、多くの管理画面で、スクリプトを生成する機能が備わっているので、それを利用して多くのオブジェクトを作成するためのスクリプトを生成しておくことができます。

また、データベースを移動する際に、最も注意する必要があるのが、ログイン アカウントの複製です。これについては、次の項で説明します。

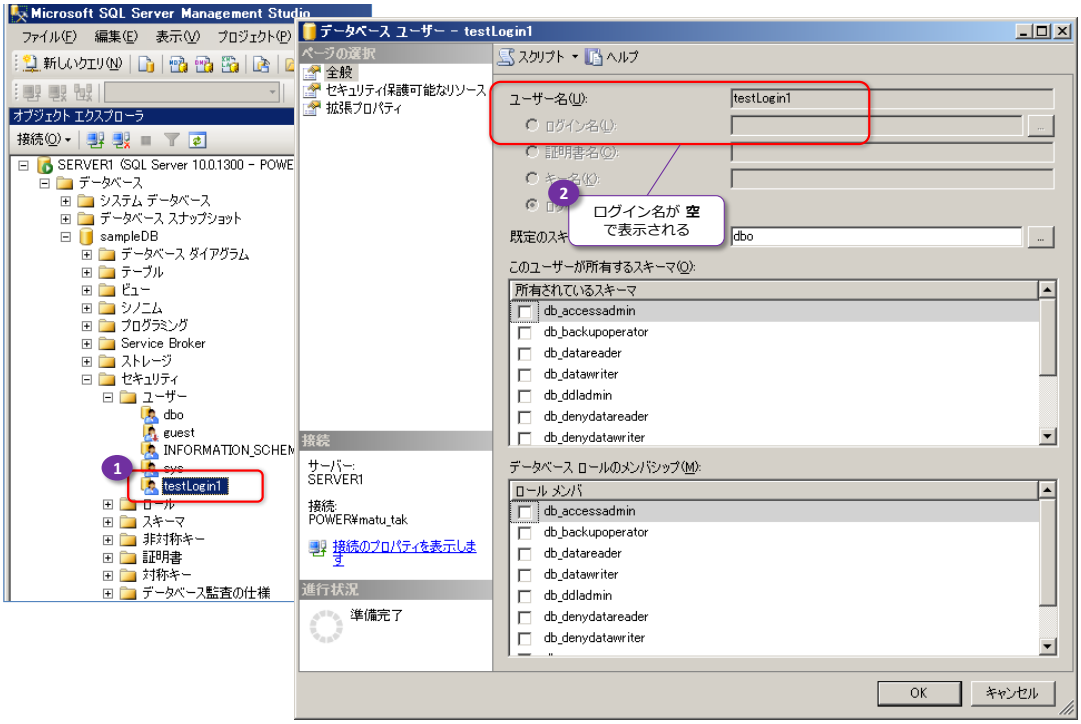
6.2　ログイン アカウントの複製

ログイン アカウントの複製

ログイン アカウントの複製は、SQL Server で標準で提供されるツールでは、パスワードと内部的に設定される SID をそのまま転送することができないので、注意する必要があります。もし、同一のパスワード／SID のログイン アカウントがきちんと転送されていない場合は、データベース ユーザーが利用できなくなり、そのユーザーへ設定されたオブジェクト権限も機能しなくなってしまいます。このようなユーザーは、「**不明なデータベース ユーザー**」と呼ばれます。

不明なデータベース ユーザーの発生

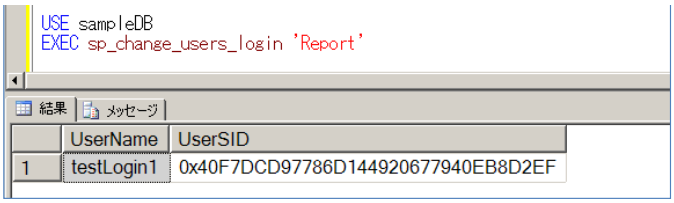
不明なデータベース ユーザーは、次のように表示されるユーザーです。



これは、「**testLogin1**」という名前のデータベース ユーザーのプロパティを表示していますが、［**ログイン名**］が「**空**」になっているのがポイントです。これは、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）に発生します。この状態では、**testLogin1** ユーザーはデータベースへアクセスすることができません。

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、移動元と移動先で、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。

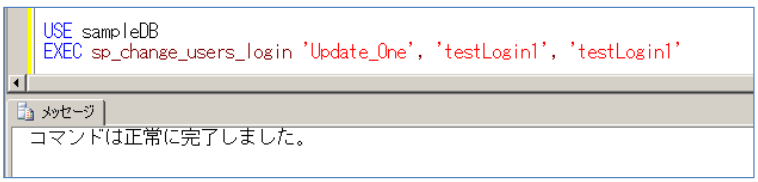
ログイン アカウントの転送

ログイン アカウントの転送が可能な標準ツールには、「**データベース コピー ウィザード**」と Integration Services の「**ログイン転送タスク**」、「**SQL Server オブジェクトの転送タスク**」の 3 つがありますが、ログイン アカウントが Windows 認証用で、かつ Active Directory ドメイン アカウントの場合は、これらのツールを利用してログイン アカウントを転送すれば、不明なデータベース ユーザーが解消され、データベース ユーザーを移動元と同じように利用することができます。

しかし、ログイン アカウントが Windows のローカル ユーザーの場合には、注意が必要です。この場合は、移行元と移行先でそれぞれ異なる Windows ユーザーとなるので、ログイン アカウントの転送に失敗してしまいます。しかも、これを回避する方法は、ログイン アカウントとデータベース ユーザーを移動先で手動で作り直すしかありません。

また、ログイン アカウントが SQL Server 認証用の場合も注意が必要です。これらの転送ツールでは、ログイン アカウントのパスワードと SID （ログイン アカウントへ内部的に割り当てられた Security ID）を転送することができないからです（パスワードと SID は、転送先で新しいものが再作成されます）。データベース ユーザーは、ログイン アカウントの名前とではなく、SID とマッピングされるので、SID が異なる場合は、不明なデータベース ユーザーが解消されません。これを解決するには、次のように sp_change_users_login を実行します。

```
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'
```



このよう **Update_One** を指定して sp_change_users_login を実行すると、不明なデータベース ユーザーの古い SID（移動元の SID）を、新しいログイン アカウントの SID（移動先の SID）へ更新し、再マッピングをしてくれるようになります。これで不明なデータベース ユーザーが解消され、移動元と同様にデータベース ユーザーが利用できるようになります。

同じパスワード／SID のログイン アカウントの作成： sp_help_revlogin

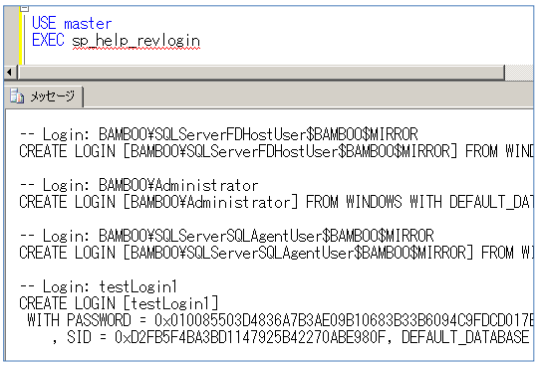
sp_change_users_login によるログイン アカウントの再マッピングは、データベース ユーザーの数が多い場合には、大変な作業になります。そこで、同じパスワード／SID のログイン アカウントを一括で作成できる方法があります。これは、**sp_help_revlogin** という名前のストアド プロシージャで、マイクロソフトのサポート技術情報（KB：Knowledge Base）の文書番号 **918992** で提供されています。

KB918992： ログインとパスワードを SQL Server 2005 のインスタンスの間、転送する方法
http://support.microsoft.com/kb/918992/ja

この文書は、SQL Server 2005 用ですが、SQL Server 2008 にもそのまま適用可能です。この文書で提供されるスクリプトを丸ごとコピーして、Management Studio のクエリ エディタに貼り付けて実行すれば、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。

作成後、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が生成されます。

```
USE master
EXEC sp_help_revlogin
```



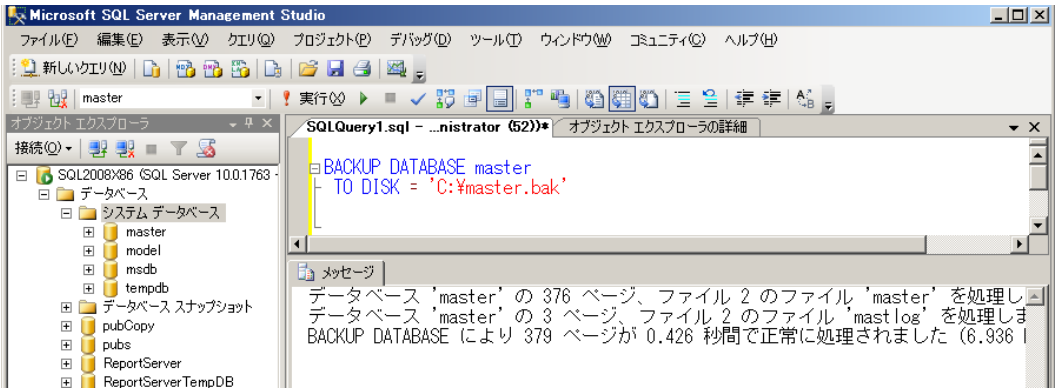
あとは、生成された CREATE LOGIN ステートメントを移動先で実行すれば、同一のパスワード／SID のログイン アカウントを作成できようになり、不明なデータベース ユーザーを解消することができます。このように sp_help_revlogin は、ログイン アカウントのバックアップと復元用途として利用できる大変便利なストアド プロシージャなので、ぜひ活用していただければと思います。

6.3　システム データベースのオンライン バックアップ／復元

システム データベースのオンライン バックアップ

SQL Server サービスを停止できないような 24 時間 365 日の運用が求められる環境では、システム データベースのオフライン バックアップが実行できないので、オンラインでバックアップを取得しなければなりません。

システム データベースをオンライン バックアップする方法は、通常のデータベースと同じで、Management Studio または BACKUP ステートメントから行えます。たとえば、master データベースを BACKUP ステートメントでバックアップする場合は、次のように実行します。



システム データベースのオンライン バックアップからの復元

システム データベースの復元は、通常のデータベースと同様、Management Studio または RESTORE ステートメントから行うことができます。ただし、master データベースの復元には、後述の追加の手順が必要になることと、次の順番で復元する必要があることに注意が必要です。

システム データベースの復元順序

1. master データベース
2. msdb データベース（復元前に Agent サービスを停止しておくこと）
3. distibution データベース（レプリケーションを利用している場合）
4. model データベース

システム データベースは、まず master をリストアしてから、ほかのデータベースを復元するようにします。また、msdb と distribution データベースを復元するまでは、SQL Server Agent サービスなど、SQL Server 関連の他のサービスをすべて停止しておく必要があります。

model データベースの復元後は、SQL Server サービスを再起動して、SQL Server 関連の他のサービスを起動します。以上で、システム データベースの復元が完了です。tempdb データベースは、自動的に再構築されるので、復元は不要です。

master データベースのリストア手順

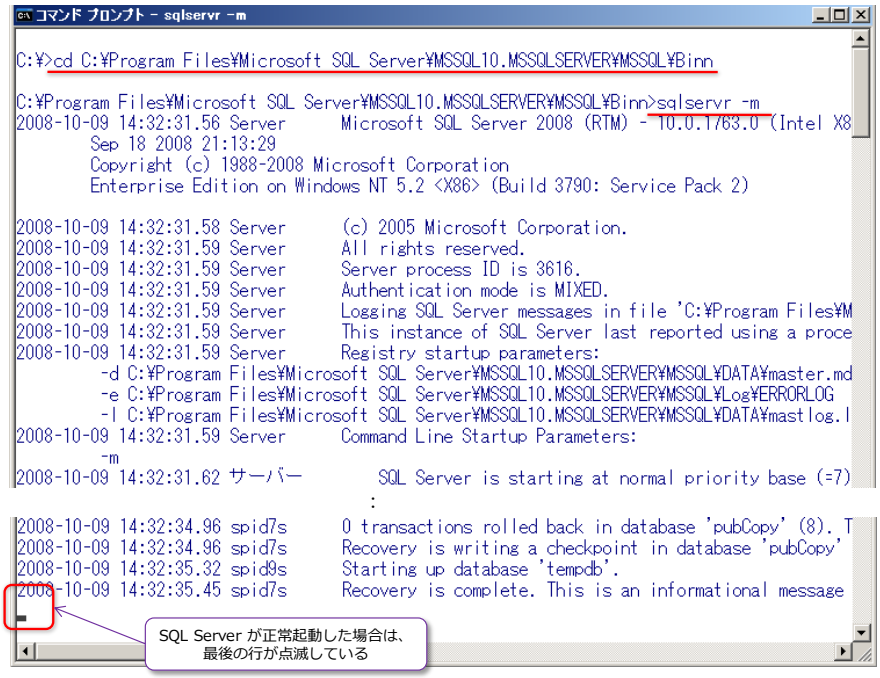
master データベースのリストアは、次の手順で実行する必要があります。

1. まず、SQL Server 構成マネージャから、SQL Server サービスを停止します。
2. 次に、コマンド プロンプトを起動して、次のようにコマンドを入力します。

```
cd C:¥Program Files¥Microsoft SQL Server¥MSSQL10.MSSQLSERVER¥MSSQL¥Binn
sqlservr -m
```

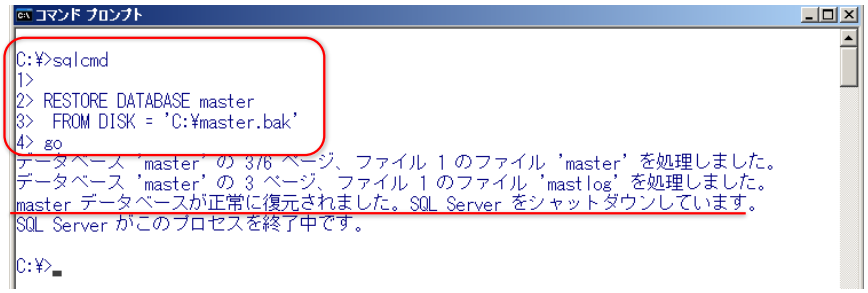
cd コマンドで SQL Server 2008 をインストールした **Binn** フォルダへ移動し、SQL Server サービスの実体である **sqlservr.exe** を **-m** オプションを付けて実行しています。これにより、SQL Server をシングル ユーザー モードで起動することができます。

なお、名前付きインスタンスとして、SQL Server 2008 をインストールしている場合は、フォルダ パスの「**MSSQL10.MSSQLSERVER**」を「**MSSQL10.インスタンス名**」へ変更する必要があります。



3. 次に、もう 1 つコマンド プロンプトを起動して、**sqlcmd** ユーティリティから、**RESTORE** ステートメントを実行して、master データベースを復元します。

```
sqlcmd

RESTORE DATABASE master
 FROM DISK = 'バックアップ ファイルへのパス'
go
```



4. 復元が成功した場合には、手順 2 でコマンド プロンプトから起動した SQL Server が、自動的にシャットダウン（停止）されます。

5. 最後に、SQL Server 構成マネージャから、SQL Server サービスを開始します。
以上で、master データベースの復元が完了です。

別のマシンへのシステム データベースの復元は基本的に不可

Step 2.3 で説明したように、別のマシン（コンピュータ名が異なるマシン）へのシステム データベースの復元は、基本的にはしてはいけません。復元後も、SQL Server サービスは問題なく起動しますが、いくつかの制限を受けます。この制限は、次のコマンドを実行することで、回避できるものもあります。

```
EXEC sp_dropserver '古いサーバー名'
EXEC sp_addserver '新しいサーバー名', @local='local'
```

しかし、このコマンドを実行しても、すべての機能が動作するという保証はありません。何が動作しなくなるのかは、オンライン ブックには明確な記載はないので、トラブルが発生する可能性がゼロではありません。したがって、コンピュータ名が異なるマシンへのシステム データベースの移動は、行わないことをお勧めします。

なお、同じ名前のコンピュータ名を設定した別マシンであれば、システム データベースを復元して、何の問題もなく SQL Server を動作させることができます（ただし、復元先の SQL Server のインストール先のパスは、復元元と同じパスにしておく必要があります）。

おわりに

最後までこの自習書の内容を試された皆さま、いかがでしたでしょうか？
SQL Server 2008 のバックアップと復元機能は、非常に簡単に利用できることを確認できたのではないでしょうか。今回は、バックアップとリストアの基本的な操作方法のみの紹介になりましたが、応用的な利用方法については、オンライン ブックに詳しく記載されていますので、ぜひ参考にしてみてください。

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/）

SQL Server と .NET を中心とした「コンサルティング サービス」と「メンタリング サービス」を提供。

主なコンサルティング実績

- 9 TB データベースの物理・論理設計支援 （パーティショニング対応など）
- 1 秒あたり 1,000 Batch Request の ASP（アプリケーション サービス プロバイダ）サイトのチューニング（ピーク時の CPU 利用率 100% を 10% まで軽減）
- 高負荷テスト（ラッシュテスト）実施のためのテスト アプリの作成支援
- 大手流通系システムの夜間バッチ実行時間を 4 時間から 1 時間半へ短縮
- 大手インターネット通販システムの夜間バッチ実行時間を 5 時間から 1 時間半へ短縮
- 宅配便トラッキング情報の日中バッチ実行時間を 2 時間から 5 分へ短縮
- 検索系 Web サイトのチューニング（10 倍以上のパフォーマンス UP を実現）
- 10 Server によるレプリケーション環境のチューニング
- 3 TB のセキュリティ監査アプリケーションのチューニング
- 大手家電メーカーの制御系アプリケーション（100GB）のチューニングと運用管理設計
- 約 3,000 本のストアド プロシージャとユーザー定義関数のチューニング
- ASP.NET / ASP（Active Server Pages）アプリケーションのチューニング
- 大手アミューズメント企業の BI システム設計支援
- 外資系医療メーカーの Analysis Services による「販売分析」システムの設計支援
- 大手企業の Analysis Services による「財務諸表分析」システムの設計支援
- Analysis Services OLAP キューブのパフォーマンス チューニング 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 SQL Server / PASSJ 理事
MCDBA（Microsoft Certified Database Administrator）
MCSD for .NET（Microsoft Certified Solution Developer）

現在、SQL Server を中心とするコンサルティング、企業に対するメンタリング サービスなどを行っている。今までに手がけたコンサルティング案件は、テラバイト クラスの DB から少人数向け小規模 DB までと 幅広く多岐に渡る。得意分野はパフォーマンス チューニング。コンサルティング業務の傍ら、講演や執筆も行い、Microsoft 主催の最大イベント Tech・Ed や、PASSJ が主催するカンファレンスなどでスピーカーとしても活躍中。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 15,500 部発行）。のびのびになっている SQL Server の新書籍は、もうじき刊行予定。

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 SQL Server / PASSJ 理事
MCDBA（Microsoft Certified Database Administrator）
MCSD for .NET（Microsoft Certified Solution Developer）

SQL Server のパフォーマンス チューニングを得意とするコンサルタント。過去には、約 3,000 本のストアド プロシージャのチューニングや、テラバイト級データベースの論理・物理設計、運用管理設計、高可用性設計などを行う。また、過去には、実際のアプリケーション開発経験（ASP／ASP.NET、VB 6.0、Java、Access VBA など）と、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。最近は、Analysis Services と Excel 2007 による BI（ビジネス インテリジェンス）システムも得意とする。執筆時のペンネームは「百田昌馬」。月刊 Windows Developer マガジンの 『SQL Server でど～んといってみよう！』、DB マガジンの『SQL Server トラの穴』を連載。マイクロソフト公開のホワイトペーパー（技術文書）のゴースト ライターとして活動することもあり。過去、マイクロソフト認定トレーナー時代には、SMS（Systems Management Server）や、Proxy Server、Commerce Server、BizTalk Server、Application Center、Outlook CDO などの講習会も担当。1998 年度には、Microsoft CTEC（現 CPLS）トレーナー アワード（Trainer of the Year）を受賞。