

SQL Server 2016

SQL Server 2016 CTP 3.2 自習書シリーズ No.1

SQL Server 2016 CTP 3.2 の新機能の概要

Published: 2016 年 1 月 31 日 有限会社エスキューエル・クオリティ



この文章に含まれる情報は、公表の日付の時点での Microsoft Corporation の考え方を表しています。市場の変化に応える必要 があるため、Microsoft は記載されている内容を約束しているわけではありません。この文書の内容は印刷後も正しいとは保障で きません。この文章は情報の提供のみを目的としています。

Microsoft、SQL Server、Visual Studio、Windows、Windows XP、Windows Server、Windows Vista は Microsoft Corporation の米国およびその他の国における登録商標です。

その他、記載されている会社名および製品名は、各社の商標または登録商標です。

この文章内での引用(図版やロゴ、文章など)は、日本マイクロソフト株式会社からの許諾を受けています。

© Copyright 2016 Microsoft Corporation. All rights reserved.

目次

STEP 1.	SQL Server 2016 CTP 3.2 の概要
1.1 1.2	SQL Server 2016 CTP 3.2 のダウンロード5 SQL Server 2016 CTP 3.2 で提供された主な新機能7
STEP 2.	Operational Analytics OLTP とデータ分析の両立21
2.1 2.2 2.3	Operational Analytics の概要 ~OLTP とデータ分析の両立~
STEP 3.	セキュリティ強化41
3.1 3.2 3.3 3.4 3.5 3.6	セキュリティの強化
STEP 4.	注目の新機能を試してみよう99
4.1 4.2 4.3 4.4 4.5	SQL Server R Services(R 統合)
STEP 5.	既存機能の強化152
5.1 5.2 5.3 5.4 5.5 5.6	SQL Server 2014 からの主な変更点153ライブ クエリ統計 (Live Query Statistics)158クエリ ストアで性能監視、プラン固定160Transact-SQL (T-SQL)の強化180AlwaysOn 可用性グループの拡張181BI 関連の強化184
5.7	付録: サンプル データベース (NorthwindJ) の作成190

STEP 1. SQL Server 2016 CTP 3.2 の概要

この STEP では、SQL Server の次期バージョンである「**SQL Server 2016**」 の CTP 3.2 で提供された主な新機能の概要を説明します。

この STEP では、次のことを学習します。

- ✓ SQL Server 2016 CTP 3.2 のダウンロード
- ✓ SQL Server 2016 CTP 3.2 の主な新機能

1.1 SQL Server 2016 CTP 3.2 のダウンロード

SQL Server の次期バージョンである「SQL Server 2016」は、製品出荷前のプレビュー版である「CTP 3.2」が、2015 年 12 月から公開されています。この CTP 3.2 は、次の URL からダウンロードすることができます。

http://www.microsoft.com/ja-jp/evalcenter/evaluate-sql-server-2016

Microsof	t									
TechNe	t Evaluation Cent	er								
マイ評価版	今すぐ評価する 🗸 技術士	コース 🖌 🏻 検証 🛙	する 試す	泛学				_		
					ここにキーワードを	入力	ρ		õ	
() S	QL Server 評個	町版ソフト	・ウェア	,						
	、 wws 10 をサービスとして提供でき 業で準備する (8 月 13 ~ 14 日)	े इ.स. Э	学ぶ MVA ジャンプ 2016 Preview	[®] スタート: Windows の新機能 (8 月 18 ^	Server ~ 19 ⊟) €	■ 検証 Ops Te いない	: する eam Show エピソ ドングル	⊦ 004	: 機能し	-
⊖ SC ≆™	QL Server 2016 Commur 版シフトウェア 180 日) ダウンロード	nity Technolog	y Preview	3.2				< *		٤
	この評価版には登録が必要です。 ファイルの 種類を選択してください ・ ISO ・ CAB ・ Azure 登録して 総行									

CTP は、**Community Technology Preview** の略で、従来の Beta 版(プレビュー版)と同じ 位置付けのものです。これまでのバージョンの SQL Server では、CTP 1、CTP 2、CTP 3 という 形で CTP が提供されていましたが、今回の SQL Server 2016 からは CTP 2、CTP 2.1、・・・、 CTP 2.4、CTP 3、CTP 3.1、CTP 3.2 のように数ヶ月ごとに最新の機能が盛り込まれた CTP が 提供されるようになりました。この自習書は、2015 年 12 月に提供された CTP 3.2 をベースに 記述していますが、CTP 3.3 などの最新版が登場した場合には、ぜひそちらで試してみてください。

この自習書では、 CTP 3.2 で提供された SQL Server 2016 の新機能を簡単に試せるように、ステップ バイ ステップ形式で画面ショット満載で紹介していますので、ぜひ、皆さんも実際に試しながらこの自習書を読み進めていただければと思います。

→ Management Studio を別途ダウンロード可能に

SQL Server 2016 からは、Management Studio の最新版を単独でダウンロードできるように変わりました。CTP が出た直後であれば、CTP に含まれている Management Studio が最新版になりますが、時間が空いた場合には、最新版が提供されていないかチェックしてみることをお勧めします。最新版の Management Studio は、次の URL からダウンロードできます。

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

- MSDN ライブラリ
 - 開発ツールと言語ドキュメント
 - デザインツール
 - モバイルおよび Embedded 開発
 - ▶ .NET 開発
 - ▶ Office の開発
 - ▶ Office ソリューション開発
 - patterns & practices
 - ▶ サーバー および エンタープライズ開発
 - Microsoft Azure
 - ▶ Web 開発
 - ▶ Windows 開発
 - Windows ドライバー開発
 - テクニカルドキュメント

SQL Server Management Studio のダウンロード

SQL Server Management Studio (SSMS) は、SQL Server のすべてのコンポーネントを構成、管理、開発 し、それらのコンポーネントへアクセスできるようにする統合環境です。 SSMS では、さまざまなグラフィック ツールと、機 能の豊富な多くのスクリプト エディターが用意されており、これらを使用して、あらゆるスキル レベルの開発者や管理者 が SQL Server にアクセスできます。

このリリースは、SQL Server 2005 から SQL Server 2016 までをサポートし、多くの新機能を備えています。たと えば、SQL Server の以前のバージョンとの互換性の強化、軽量のスタンドアロン Web インストーラー、新しいリリース が入手可能になると SSMS 内で表示されるトースト通知などがあります。このリリースは、Azure SQL Database の 最新のクラウド機能の使用も、最高レベルでサポートしています。

🗹 メモ

SQL Server Management Studio のこのバージョンは、同じコンピューターにインストールされている以前のリリ ース バージョンとの共存に対応しています。

SQL Server Management Studio プレビュー

Ĵ.

この最新プレビューのバージョン番号は 13.0.800.111 です SSMS プレビュー リリースは、入手可能な最新のサービス パックと併せて使用される場合、次のプラットフォームをサ ポートします:

Windows 10, Windows 8, Windows 8.1, Windows 7, Windows Server 2012 (64 ビット), Windows Server 2008 R2 (64 ビット)

SQL Server Management Studio November 2015 プレビューをダウンロードする

1.2 SQL Server 2016 CTP 3.2 で提供された主な新機能

SQL Server 2016 CTP 3.2 には、非常にたくさんの新機能が提供されています。これらをまとめると、次のようになります。

	SQL Server 2016 からの主な新機能
性能向上	 ・Operational Analytics (OLTPとデータ分析の両立) インメモリ OLTP と列ストア インデックスの融合 ・インメモリ OLTP の飛躍的な進化 クラスター化列ストア インデックスのサポート、最大サイズ 2TB、64コア以上での性 能向上、並列プランへの対応、TDE のサポート、ALTER のサポート、BIN2 以外の照 合順序のサポートなど ・列ストア インデックスの大幅強化 バッチ モードの性能向上、集計関数のプッシュダウンによる性能向上、 クラスター化列ストア インデックスでの追加の B-tree インデックスのサポート、 非クラスター化列ストア インデックスでの更新サポート、フィルター列インデックス のサポート、主キー/外部キー制約のサポート、AlwaysOn 可用性グループの読み取り 可能セカンダリのサポートなど
セキュリ ティ強化	 動的データ マスクによるデータのマスク(情報漏洩対策) ・行レベル セキュリティによる行レベルのアクセス制御 ・Always Encrypted による暗号化 ・TDE(透過的なデータ暗号化)の性能向上、インメモリ OLTP 対応 ・テンポラル テーブルによる Audit
注目の 新機能	 R 統合(SQL Server R Services) JSON 対応 Strech Database(ストレッチ データベース) Azure バックアップ URL の性能向上 PolyBase で Hadoop アクセス(HDFS)
可用性	 AlwaysOn 可用性グループの強化 自動フェールオーバーの台数が 2から 3 に増加、ログ転送性能の向上(マルチ スレッドで処理)、ラウンドロビン レプリカ、TDE のサポート、DTC(分散トランザクション)のサポート、ワークグループ環境でも利用可能(Windows Server 2016 を利用している場合)、Standard エディションでも利用可能 など
既存機能 の強化	 ・DROP IF EXISTS (オブジェクトが存在しているなら DROP を実行) ・tempdb の複数ファイル/初期サイズをセットアップ時に選択可能に ・トレースフラグ 1117、1118、4199 が既定でオンに ・ライブ クエリ統計 ・クエリ ストアによる性能監視、プラン固定
BI 関連 の強化	 Reporting Services の大幅強化 パラメーター ボックスのカスタマイズ、新しいグラフのサポート(サンバースト、 ツリーマップ)、PowerPoint レンダリング、印刷コントロールの変更、 Datazen 統合(モバイル レポート対応、新しいレポート マネージャー)など Integration Services の強化 Execute SQL タスクで R スクリプトのサポート、Azure Feature Pack、Hadoop (HDFS)のサポート、データ フローでのエラー時の列名のサポート、制御フローの テンプレート作成、AlwaysOn 可用性グループでの SSIS カタログ DB のサポート、 AutoAdjustBufferSize プロパティ など Analysis Services の強化 DirectQuery の新しい実装/大幅な性能向上、DBCC コマンドのサポート、 Tabular Model Scripting Language (TMSL)の提供、 AMO (Analysis Services Management Objects)の変更 など MDS の強化 性能の向上、セキュリティの強化、各種の機能強化 など

SQL Server 2016 では、**性能向上**および**セキュリティ強化**を実現できる機能が多く提供されてい るのが大きなポイントです。特に、**Operational Analytics**(OLTP とデータ分析の両立)は、今 後のデータベースのあり方を大きく変えるのではないかと思えるほど、非常に大きな可能性を感じ させるものです。

セキュリティに関しては、マイナンバーや各種の情報漏洩事件など、業界全体のセキュリティ意識 の高まりもあって、ここ数年弊社へのお問い合わせが非常に増えている分野です。SQL Server 2016 には、セキュリティを向上させることができる機能が数多く提供されているので、そういっ たニーズに答えることができます。

その他、**R 統合**(SQL Server R Servicces)や、**JSON**対応、**PolyBase**、**Hadoop**対応など、 最近のマイクロソフト社の方向性と同様、SQL Server でもオープン化が非常に進んでいます。

この章では、これらの新機能の概要を説明します。

◆ 性能向上に関する新機能 ~目玉の新機能は Operational Analytics~

SQL Server 2016 の一番の目玉の新機能は「Operational Analytics」です。Operational は 「OLTP」(オンライン トランザクション処理)、Analytics は「データ分析」と捕らえると分かり やすい言葉で、機能面で言うと、「インメモリ OLTP」と「列ストア インデックス」の進化/融合 です(インメモリ OLTP と列ストア インデックスの良いとこどりをして、OLTP とデータ分析を 両立できるようにしたものです)。

<u>Operational Analytics (OLTPとデータ分析の両立)</u>

Operational ワークロード
= OLTP (オンライントランザクション処理) などImage: SQL Server 2014 までは
ンメモリ OLTP や
通常リレーショナル テーブルを利用Analytics ワークロード
= データ集計/分析処理、DWH、夜間バッチなどImage: SQL Server 2014 までは
別ストアインデックスや
Analysis Services などを利用SQL Server 2016 ではどちらも実現(両立) 可能
次の 3パターンで実装できる
1. インメモリ OLTP + クラスター化列ストア インデックス (完全なインメモリ実装)
2. 通常リレーショナル テーブル + 非クラスター化列ストア インデックス
3. 通常リレーショナル テーブル + クラスター化列ストア インデックス

これまでの SQL Server では、「Operational ワークロード」と「Analytics ワークロード」に 対しては、別々のテクノロジー/機能を利用して、使い分ける必要がありましたが(同時に利用す るのが難しいところがありましたが)、SQL Server 2016 からは「インメモリ OLTP」と「列スト ア インデックス」が飛躍的な進化を遂げたことで、どちらも両立できるようになりました。

実際に、<u>どれぐらいの性能が出るのか</u>を検証してみたのが、次のグラフです(詳細は 2 章に記載しています)。



* ベンチマークの結果の公表は、使用許諾契約書で禁止されていますが、その効果をわかりやすく表現するため、 日本マイクロソフト株式会社の監修のもと、数値を掲載しております。

* 検証は、SQL Server 2016 CTP 3.2 を利用して行ったので、<u>RTM 版ではさらに性能が向上</u>する可能性があります。

ディスク ベースの通常テーブルでは **1.98 秒**かかった集計処理が、インメモリ OLTP にすること で **1.6 秒** (18.8%の性能向上)、さらに**列ストア インデックスを作成**することで、わずか **198 ミ リ秒**で処理できるようになり、**10 倍**もの性能向上を確認することができました。

このように、インメモリ OLTP と列ストア インデックスを組み合わせて利用すれば、完全なイン メモリで OLTP とデータ分析を両立できるようになります。もちろん、メモリをそこまで搭載でき ないという場合には、列ストア インデックスのみを単独で利用するという選択肢もあります。SQL Server 2016 の列ストア インデックスでは、今までは読み取り専用だった非クラスター化列スト ア インデックスが更新可能になったり、クラスター化列ストア インデックスに追加の B-tree イ ンデックスを作成できるようになったり、PRIMARY KEY 制約がサポートされるようになったりし ているので、既存環境から大きな変化を加えることなく利用できるようになりました。また、バッ チ モード (列ストア インデックスの内部的な動作モード)の性能向上も実現しているので、より Analytics ワークロード (データ分析/集計) に強くなっています。

これらのインメモリ OLTP と列ストア インデックスの強化ポイントについては、2 章で詳しく説 明しています。

SQL Server 2016 では 480 スレッドでもスケール

最近は、1 個の CPU あたりのコア数がどんどん増える "メニー コア" 時代になりましたが、コア 数が増えてもスケールしないデータベースが存在する中(特に 64 コアを超える K グループ環境 に対応していないデータベースがある中)、SQL Server はメニー コア環境でも、K グループをま たがったとしてもスケールします。昨年開催された SQL Server の世界最大規模のイベントであ る「PASS Summit 2015」の基調講演では、480 スレッド分もの CPU を 100%フル活用して、 スケールしているデモが行われました(以下の写真)。



* PASS Summit 2015 の基調講演中のデモを筆者が撮影した写真

1 個の CPU で 15 コア (30 スレッド) というのも時代の流れを感じますが、こうしたメニー コア に対応できるのも SQL Server の大きな特徴です。

◆ セキュリティ強化に関する新機能

SQL Server 2016 には、セキュリティを向上させることができる機能が数多く提供されています。 その主なものは、次のとおりです。

動的データ マスク(Dynamic Data Masking)

動的データ マスクは、クレジット カード番号やマイナンバーなどの**顧客情報/機密情報をマスク** (別の値に置換)して、情報漏洩を防止できる機能です(詳細は 3章で説明)。

動的データ マスクの利用例



行レベル セキュリティ(Row Level Security)

行レベル セキュリティは、行レベルのアクセス制御を実現できる機能で、ユーザーごとに、参照で きる行データを制限することができる機能です(詳細は 3 章で説明)。



Always Encrypted による列データの暗号化

Always Encrypted は、ネットワーク上を流れるデータも、データベース内に格納されるデータも、 すべて暗号化して格納できる機能です(列データを暗号化して、アプリケーションも透過的に利用 できます。詳細は 3 章で説明)。



TDE(透過的なデータ暗号化)の性能向上、インメモリ OLTP 対応

TDE (透過的なデータ暗号化) は、Intel の AES-NI (AES 暗号化の処理を高速化するための命令 セット、ハードウェア アクセラレータ) に対応して、性能向上を実現しました。また、インメモリ OLTP を利用している場合にも、TDE が利用できるようになりました。

TDE を設定したことによる性能のオーバーヘッド(暗号化処理のオーバーヘッドなど)を検証したのが次のグラフです。



* ベンチマークの結果の公表は、使用許諾契約書で禁止されていますが、その効果をわかりやすく表現するため、 日本マイクロソフト株式会社の監修のもと、数値を掲載しております。

ディスク ベースの通常テーブルでは、<u>ラッチ待ちがボトルネック</u>になってしまうので、TDE を設 定しても、設定しなくても結果は変わらない(オーバーヘッドは確認できない)、インメモリ OLTP の Durable(SCHEMA_AND_DATA)では、TDE を設定しない場合が 8 分 32 秒であるのに対 して、TDE を設定したとしても 8 分 54 秒で完了、わずか 4.3% のオーバーヘッドであること を確認できました(詳細は 3 章で説明)。

テンポラル テーブルを利用した Audit(コンプライアンスにおける監査証跡)

SQL Server 2016 では、過去の更新履歴を自動保存できる「テンポラル テーブル」機能を利用して、Audit(監査証跡)を実現できます。テンポラル テーブルは、操作ミス時のデータ復旧や、データ分析における SCD(緩やかに変化するディメンション)シナリオでも利用できます。テンポラル テーブルでは、次のように過去のデータを簡単に参照することができます。

更新前のデータ(過去の指定した日時のデータ)を確認 SELECT * FROM Products FOR SYSTEM TIME AS OF '2016-01-01' ORDER BY 商品コード 100 % ◆ < ■ 結果 Ba xytr=ジ							
	商品コード	商品名	sysstart		sysend		
1	1	果汁100% オレンジ	2016-01-01 0	0:00:00.0000000	2016-01-11 04	:18:32.2304961	
2	2	果汁100% グレープ	2016-01-01 0	0:00:00.0000000	2016-01-11 04	:18:32.2313230	
3	3	果汁100% レモン	2016-01-01 0	0:00:00.0000000	9999-12-31 23	:59:59.9999999	
4	4	果汁100%ピーチ	2016-01-01 0	0:00:00.0000000	9999-12-31 23	:59:59.9999999	
5	5	コーヒーマイルド	2016-01-01 0	0:00:00.0000000	9999-12-31 23	:59:59.9999999	

→ その他の注目の新機能

SQL Server 2016 では、R 統合や、JSON 対応、Hadoop 対応など、数多くの新機能が提供されています。その主なものは、次のとおりです。

• SQL Server R Services (R 統合)

SQL Server R Services は、Transact-SQL ステートメントを利用して、R スクリプ ト/ライブラリを実行することができる機能です。

sum EXEC sp_execute_external_script @language = N'R' ,@script = N'sum1 <- sum(input1[,1]); ret <- data.frame(sum1);' ,@input_data_1 = N'SELECT colA, colB FROM mage	R スクリプト を記述 askTestDBmaskTest1'
,@input_data_l_name = N input ,@output_data_l_name = N'ret' WITH RESULT SETS ((coll int))	SQL Server 上のテーブル データを
100 % - <	R スクリプトでの処理データとして 与えることができる
col1 R スクリプト 1 6	

● JSON 対応

SQL Server 2016 では、**JSON** (JavaScript Object Notation) に対応して、SQL Server 上のテーブル データを JSON 形式で出力したり、JSON データを SQL Server に取り 込んだりできるようになりました(**FOR JSON** や **OPENJSON、ISJSON、JSON_ VALUE、JSON_QUERY** など、JSON データを処理するための多数の関数を提供)。

DECLARE @jsonstr nvarchar(max) = N'["arr1": [["key1": "test1",]]'	"key2": 999] "key2": 777]
SELECT * FROM OPEN/SON(@jsonstr, '\$.arr1') WITH (keyl varchar(20), key2 int DO % East SON データを取得 2 test2 777	ISJSON 関数で ISON データ かどうかをチェック Dとうかをチェック COLA int FRIMARY KEY .colB nvarchar(max) CONSTRAINT chkJSON CHECK (ISJSON(colB) > 0) JSON データをテーブルに格納 INSERT INTO jsonTest1 VALUES(1, N'["key1": "test1", "key2": 999]') INSERT INTO jsonTest1 VALUES(2, N'["key1": "test1", "key2": 999]') INSERT INTO jsonTest1 VALUES(2, N'["key1": "test2", "key2": 999]') INSERT INTO jsonTest1 VALUES(2, N'["key1": "test2", "key2": 999]') INSERT INTO jsonTest1 VALUES(2, N'["key1": "test2", "key2": 777]') テーブル内の JSON データをクエリ ESELECT JSON VALUE(colB, '\$.key1') AS key1 JSON_VALUE(colB, '\$.key2') AS key2 FROM jsonTest1 IO0% - Eselect 1999 2 test2 777

● Strech Database ~アクセス頻度の低いデータをクラウドへ~

Strech Database は、テーブル データをクラウド(**Microsoft Azure SQL Database**) 上に配置することができる機能です。



● Azure バックアップ URL の性能向上(ブロック BLOB 対応)

SQL Server では、バックアップを Microsoft Azure の BLOB ストレージ上に配置す る機能が SQL Server 2012 の SP1 CU2 から利用することができましたが、これは BLOB ストレージ内の「ページ BLOB」にファイルを格納していました。SQL Server 2016 からは、従来と同様「ページ BLOB」に格納する方法に加えて、「ブロック BLOB」 に格納する方法が追加されました。

出力された sv= 以降をコピーして SECRET に貼り付け CREATE CREDENTIAL [https://matustoraze.blob.core.windows.net/cont1] WITH IDENTITY='Shared Access Signature' ,SECRET='sv=2014-02-14&sr=c&sig=KdUtNIp4a7%2F0HsIzahdF1cPcf rzwipV 100% - <	: 上の)B プ
TO URL = 'https://matustorage.blob.core.windows.net/contl/North_1.bak'	
URL = 'https://matustorage.blob.core.windows.net/contl/North_2.bak'	
WITH COMPRESSION, STATS	
	_
🗓 メッセーシ ブロック BLOB では	
10 パーセント処理されました。 ストライプ バックアップにも	5
20 パーセント処理されました。 対応することで注意向上が可	FIE .
31 ハーゼント処理されました。	
41 パービント処理されました。	

• PolyBase で Hadoop アクセス (HDFS)

PolyBase は、**Hadoop ファイル システム**(HDFS: Hadoop File System)上にある データを、SQL Server からアクセスできる機能です(この機能は、元々 SQL Server Parallel Data Warehouse: PDW で提供されていた機能ですが、SQL Server 2016 か らは、通常の SQL Server でも利用できるようになりました)。

PolyBase を利用すれば、Microsoft Azure 上の Hadoop サービスである「Microsoft Azure HDInsight」や、Hortonworks の提供する「HDP: Hortonworks Data Platform」、Cloudera の提供する「CDH: Cloudera's Distribution including Apache Hadoop」上のデータに対して、SQL Server からアクセスできるようになります。また、PolyBase では、Microsoft Azure 上の「BLOB ストレージ」にアクセスする こともできます。



➡ 既存機能の強化

SQL Server 2016 では、既存機能の強化も怠っていません。その主なものは、次のとおりです。

• DROP .. IF EXISTS (オブジェクトが存在している場合に DROP を実行)

これまでの SQL Server では、オブジェクトを削除する場合には、次のようにオブジェ クトの存在チェックを行う必要がありました。

-- 該当オブジェクトが存在するなら削除(SQL Server 2014 以前の場合)
 IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'obj1'))
 DROP TABLE obj1

SQL Server 2016 からは、「**DROP .. IF EXISTS**」が提供されたことで、上の記述を、 次のように記述することができます。

```
-- 該当オブジェクトが存在するなら削除 (SQL Server 2016 の場合)
DROP TABLE IF EXISTS obj1
```

インストール時に各種の設定が可能に

SQL Server 2016 では、これまでのバージョンではインストール後に行っていたような 設定(ベスト プラクティス値など)を、インストール時に行えるようになりました。



tempdb の複数ファイル/初期サイズもインストール時に設定できるようになりました。

1 SQL Server 2016 CTP3.1 セットアップ		×
データベース エンジンの構成	5	
データベース エンジンの認証セキュリ	ティモード、管理者、データ ディ	(レクトリ、および TempDB の設定を指定します。
プロダクト キー	サーバーの構成 データディ	レクトリ TempDB FILESTREAM
ライセンス条項 グローバル ルール	TempDB データ ファイル:	tempdb.mdf, tempdb_mssql_#.ndf .mdf のファイル数
Microsoft Update 製品の更新プログラム セットアップ ファイルのインストール	ファイルの数(<u>U</u>): 初期サイズ (MB)(<u>I</u>):	2 v 8 v 初期サイズの合計 (MB): 16
1ンストール ルール	自動拡張 (MB)(]):	64 🗧 自動拡張の合計 (MB): 128
セットアップロール 機能の選択 機能ルール 名ファイルの 初期サイズ	データ ディレクトリ(<u>D</u>):	C:¥Program Files¥Microsoft SQL Server¥MSSQL13.MSSQLSERV 調節(魚) ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・
インスタンスの構成 作成は サーバーの構成	易所	
データベース エンジンの構成 Analysis Services の構成	TempDB ログファイル:	templog.ldf templog.ldf .ldfの初期サイズ、
Reporting Services の構成 PolyBase の構成	初期サイズ (MB)(<u>S</u>): <u> </u> 自動粧程 (MB)(G):	8 ◆ 作成場所など 64 ●
機能構成ルール インストールの準備完了 インストールの進行状況	ロガ ディレクトリ(<u>し</u>):	C:¥Program Files¥Microsoft SQL Server¥MSSQL13.MSSQLSER\
		< 戻る(B) 次へ(N) > キャンセル ヘルプ

- tempdb の性能に関連するトレースフラグ 1117 と 1118 が既定でオンに
- オプティマイザーの動作に関連するトレースフラグ 4199 が既定でオンに
- Active Directory のパスワード認証/統合認証を利用可能に
- **ライブ クエリ統計** (Live Query Statistics)

ライブ クエリ統計は、現在実行中の SQL ステートメントの実行プランのうち、どの部 分を実行しているのかを確認できる機能です。



● クエリ ストア(Query Store)で性能監視、プラン固定

クエリ ストアは、**クエリの実行履歴**(実行プランを含む)を保存することができる機能 です。性能に問題が出た場合に、過去に振り返って、どういった実行プランで実行されて いたのかを確認することができます。



また、同じクエリに対して、複数の実行プランがある場合には、次のように簡単に見比べ

ることができ、かつどちらを利用させたいのかを変更(データベース管理者によるプラン 固定)することも簡単にできます。



● AlwaysOn 可用性グループの拡張

SQL Server 2016 では、AlwaysOn 可用性グループも強化されました。自動フェールオ ーバーを構成できる台数が 2 から 3 に増加や、ログ転送性能の向上(マルチ スレッド で処理)、ラウンドロビン レプリカ、TDE (透過的なデータ暗号化)のサポート、DTC (分 散トランザクション)のサポート、ワークグループ環境で利用可能(Windows Server 2016 を利用する場合)、Standard エディションのサポートなどが提供されました。

実際に、弊社環境で、ログ転送の性能を検証してみたところ、次のような結果を確認する ことができました。



コネマインコンシーネル会社の血疹のじと、気温を汚転しておりより。 また、検証は CTP 3.1 で行ったものであるため、RTM 版ではさらに性能が向上する可能性があります。

SQL Server 2014 では **300MB/sec** 弱のログ転送スピード(Bytes send to Transport/sec) であるところを、SQL Server 2016 では **450MB/sec** 近く出ている ことを確認することができました(詳細は 5章に記載)。

• Reporting Services (SSRS)の大幅強化

SQL Server 2016 では、Reporting Services も大幅に強化されました。パラメーター ボックスのカスタマイズや、新しいグラフのサポート(サンバースト、ツリーマップ)、 新しいレポート ビルダー、PowerPoint レンダリング、印刷コントロールの変更(ActiveX コントロールが不要に)、Datazen 統合(モバイル レポート対応、新しいレポート マネ ージャー)などが提供されました。



Datazen 統合では、次のような見栄えの良いレポートを簡単に作成できるようになり、 かつ**モバイル レポート**(スマートフォンやタブレット端末など向けのレポート)にも対 応しました(詳細については、別途自習書を作成する予定です)。



Integration Services (SSIS)の強化

Integration Services では、Execute SQL タスクでの R スクリプトのサポートや、 Azure Feature Pack の提供 (Azure BLOB/Azure HDInsight を操作可能なタスク)、 Hadoop (HDFS)のサポート、インポート/エクスポート ウィザードでの Azure BLOB のサポート、データ フローでのエラー時の列名のサポート、制御フローのテンプレート 作成、AlwaysOn 可用性グループでの SSIS カタログ DB のサポート、 AutoAdjustBufferSize プロパティでのバッファ サイズの自動計算、OData v4/ Excel 2013 データソースのサポート、新しい Custom Logging Level、パッケージの増 分配置などが提供されました。

● Analysis Services (SSAS)の強化

Analysis Services では、DirectQuery の新しい実装や大幅な性能向上、DBCC コマンドのサポート、AMO (Analysis Services Management Objects)の変更、Tabular Model Scripting Language (TMSL) などが提供されました。

● MDS (マスター データ サービス)の強化

MDS では、性能の向上や、セキュリティの強化、各種の機能強化(トランザクションの メンテナンス、多対多リレーションシップ、Excel アドインでのビジネス ルール管理な ど、多数の機能強化)などが提供されました。

以降では、これらの新機能について、ステップ バイ ステップ形式で画面ショット満載で紹介して いるので、ぜひ、皆さんも実際に試しながらこの自習書を読み進めていただければと思います。

STEP 2. Operational Analytics OLTP とデータ分析の両立

この STEP では、SQL Server 2016 の一番の目玉の新機能である「インメモリ OLTP と列ストア インデックスの融合」による OLTP とデータ分析の両立につ いて、概要を説明します。

この STEP では、次のことを学習します。

- ✓ **Operational Analytics**(OLTP とデータ分析の両立)
- ✓ インメモリ OLTP の飛躍的な進化 (データ分析でも利用可能に)
- ✓ **列ストア インデックスの飛躍的な進化**(OLTP でも利用可能に)

2.1 Operational Analytics の概要 ~OLTP とデータ分析の両立~

SQL Server 2016 の一番の目玉の新機能は「Operational Analytics」です。Operational は 「OLTP」(オンライン トランザクション処理)、Analytics は「データ分析」と捕らえると分かり やすい言葉で、機能面で言うと、「インメモリ OLTP」と「列ストア インデックス」の進化/融合 です(インメモリ OLTP と列ストア インデックスの良いとこどりをして、OLTP とデータ分析を 両立できるようにしたものです)。

<u>Operational Analytics (OLTPとデータ分析の両立)</u>



3. 通常リレーショナル テーブル + クラスター化列ストア インデックス

これまでの SQL Server では、「Operational ワークロード」と「Analytics ワークロード」に 対しては、別々のテクノロジー/機能を利用して、使い分ける必要がありましたが(同時に利用す るのが難しいところがありましたが)、SQL Server 2016 からは「インメモリ OLTP」と「列スト ア インデックス」が飛躍的な進化を遂げたことで、どちらも両立できるようになりました。具体的 には、次の 3 つのパターンで実現することができます。

1. インメモリ OLTP + クラスター化列ストア インデックス (完全なインメモリ実装)

2. 通常リレーショナル テーブル + 非クラスター化列ストア インデックス

3. 通常リレーショナル テーブル + クラスター化列ストア インデックス

SQL Server 2014 までのインメモリ OLTP は、OLTP 向けのインメモリ リレーショナル データ ベース エンジン、列ストア インデックスは、データ分析/集計に強いカラム指向データベースの SQL Server 実装として別々に提供されてきましたが、SQL Server 2016 からは、インメモリ OLTP と列ストア インデックスが融合して、それぞれの良いとこどりをして(OLTP にもデータ分 析にも強くなって)、完全なインメモリで Operational Analytics を実現(OLTP とデータ分析を 両立)できるようになりました。

また、これまでは読み取り専用として提供されていた「**非クラスター化列ストア インデックス**」が、 SQL Server 2016 からは<u>更新可能になった</u>ことで、通常のリレーショナル テーブルに、容易に列 ストア インデックスを追加できるようになりました。これによって、従来ながらの B-tree イン デックスを追加するのと同じような感覚で、列ストア インデックスを追加できるようになって、こ れを追加すれば、Analytics ワークロード(データ集計/分析クエリ)の性能を大幅に向上させる ことができます(Operational Analytics の実現)。

また、SQL Server 2016 では、列ストア インデックスそのものの**性能向上**(バッチ モードの強 化、プッシュダウン、更新性能の向上、パラレル Insert など)も実現しているので、現在の SQL Server で性能に問題を抱えているという方には、ぜひ試してみてほしい機能です。

↓ インメモリ OLTP の飛躍的な進化 ~Analytics ワークロードにも対応~

インメモリ OLTP は、SQL Server 2014 から提供されたものですが、SQL Server 2014 のとき には、(最初のバージョンであったこともあり)制限事項が非常に多くありました。SQL Server 2016 からは、そういった制限事項が大きく取り払われて、インメモリ OLTP が飛躍的に進化しま した。一番の進化は、前述したように**クラスター化列ストア インデックス**を追加できるようになっ たことで(完全なインメモリでの **Operational Analytics** の実現)、その他の具体的な強化内容 は、次のとおりです。

SQL Server 2016 からのインメモリ OLTP の強化ポイント

- インメモリ OLTP で列ストア インデックスのサポート(インメモリ OLTP のメモリ最 適化テーブルにクラスター化列ストア インデックスを追加可能に)
- 最大サイズが 2TB に拡張 (大量データへの対応)
- 64 コア以上での性能向上に対応(スケールアップが可能)
- **並列プラン**への対応(メニー コアの活用が可能に)
- **TDE**(透過的なデータ暗号化)のサポート(セキュリティの強化)
- ALTER/BIN2 以外の照合順序のサポート(既存環境からの移行しやすさが大幅に向上)

SQL Server 2014 のときには、制限事項の多さや、大規模環境でのスケール面(64 コア以上だと スケールしない)、集計クエリでの性能面(並列プランに未対応)など、既存環境(ディスク ベー スのテーブル)をインメモリに移行するのが難しかったところがありましたが、<u>SQL Server 2016</u> からは、そういった制限事項が大きく取り払われました。特に ALTER のサポートや BIN2 以外 の照合順序のサポートは、移行という観点で非常に大きく、SQL Server 2014 のときには、これ が原因で移行を断念しているというお客さんを見てきたので、SQL Server 2016 からは格段と移 行しやすくなりました。

また、SQL Server 2016 からはインメモリ OLTP の最大サイズが **2TB** に増えたことで(SQL Server 2014 のときは **512GB** が最大サイズ)、データ量が多くてもインメモリ OLTP に移行で きるようになりました。

◆ 検証結果:インメモリ OLTP + クラスター化列ストア インデックス

実際に、SQL Server 2016 のインメモリ OLTP とクラスター化列ストア インデックス (完全な インメモリでの Operational Analytics の実装) を利用して、<u>どれぐらいの性能が出るのか</u>を検証 してみたのが、次のグラフです。



* ベンチマークの結果の公表は、使用許諾契約書で禁止されていますが、その効果をわかりやすく表現するため、 日本マイクロソフト株式会社の監修のもと、数値を掲載しております。

* 検証は、SQL Server 2016 CTP 3.2 を利用して行ったので、<u>RTM 版ではさらに性能が向上</u>する可能性があります。

このグラフは、SQL Server 2016 の CTP 3.2 を利用して、1 億件のデータに対する GROUP BY 演算(データ集計処理)を行った結果です(テーブル構成や実行した SQL の詳細は後述しま す)。CTP 3.2 にはデバッグ コードが含まれていたり、チューニングが不十分な状態であるにも関 わらず、大きな性能向上を確認することができました(チューニングが施された RTM 版ではさら に性能が向上する可能性があります)。

ディスク ベースの通常テーブルでは 1.98 秒かかった集計処理が、インメモリ OLTP にすること で 1.6 秒 (18.8%の性能向上)、さらに列ストア インデックス (クラスター化列ストア インデッ クス)を作成することで、わずか 198 ミリ秒で処理できるようになり、<u>10 倍もの性能向上</u>を確認 することができました。これに対して、SQL Server 2014 では、インメモリ OLTP を利用するこ とで、2.12 秒かかっていた処理が 42.76 秒もかかってしまい、20.2 倍も遅い結果になってしま っています (SQL Server 2014 で遅くなる理由については後述します)。

このように、SQL Server 2016 のインメモリ OLTP は、クラスター化列ストア インデックスを 追加できるようになったここで、データ分析/集計(Analytics ワークロード)にも強くなりまし た。

◆ 列ストア インデックスの進化 ~カラム指向とリレーショナル DB の融合~

列ストア インデックス (Column-store Index) は、大量のデータを分析/集計するときに強さを 発揮するカラム指向データベースの SQL Server 実装として、SQL Server 2012 のときに提供 されたものです。列ストア インデックスは、「xVelocity 列ストア インデックス」と呼ばれるこ ともあり、PowerPivot および Analysis Services の Tabular Mode (テーブル モード)で採用 されている<u>インメモリの BI エンジン</u>である「xVelocity エンジン」を RDB (リレーショナル デ ータベース) に応用したものです。なお、このエンジンは、SQL Server 2008 R2 のときに開発さ れて (PowerPivot for Excel として実装)、その当時は VertiPaq エンジンと呼ばれていました。 **列ストア インデックス**の xVelocity エンジンでは、次のように<u>列単位でインデックスを格納し、</u> それらは高度に圧縮されています。



このインデックスによって、大量のデータに対する集計処理(GROUP BY 演算など)の性能を大きく向上させることができます。性能が向上する理由は、次のような集計クエリを考えると分かりやすいと思います。



列ストア インデックス (カラム ストア) であれば、<u>集計対象となる列データを読み込むだけ</u>で済 み、かつそのデータは高度に圧縮されているので、読み込み時間を大幅に短縮することができます。 例えば、弊社のお客様(**100 億件の DWH**)では、列ストア インデックスを作成することで、**531GB** のテーブル (Row ストア) が **90GB** (カラム ストア) へと、**約 1/6** のサイズにまで圧縮するこ とができています。このことからも、読み込み量を小さくできることが分かると思います。

列ストア インデックスは、SQL Server 2012 では、読み取り専用モードの「**非クラスター化列ス** トア インデックス」のみがサポートされていましたが、SQL Server 2014 からは<u>更新可能</u>な列ス トア インデックスとして「**クラスター化列ストア インデックス**」が提供されました。そして、今 回の SQL Server 2016 からは、これらが大幅に強化されて、その主なものは次のとおりです。

- インメモリ OLTP にクラスター化列ストア インデックスを作成可能に (=完全なインメモリでの Operational Analytics の実装)
- 列ストアインデックスの性能向上(バッチモードの性能向上、集計のプッシュダウン、
 更新性能の向上、ウィンドウ関数のバッチモード対応など)
- 非クラスター化列ストア インデックスが**更新可能**に

25

- 非クラスター化列ストア インデックスでフィルター列が利用可能に
- クラスター化列ストア インデックスでの追加の B-tree インデックスのサポート
- 主キー/外部キー制約のサポート
- AlwaysOn 可用性グループの読み取り可能セカンダリのサポート

これまでは Analytics ワークロード (データ集計/分析) に特化していた列ストア インデックス が、SQL Server 2016 からは Operational ワークロード (OLTP) にも対応して、Operational Analytics (OLTP とデータ分析の両立) を実現できるようになりました。特に、OLTP ワークロー ドで強さを発揮する「インメモリ OLTP」にクラスター化列ストア インデックスを作成できるよ うになったことが大きな進化で、その効果は前掲のグラフのとおりです。

また、従来は、読み取り専用で提供されていた「非クラスター化列ストア インデックス」が更新可 能になったことで、今まで利用していたシステムに、容易に列ストア インデックスを追加できるよ うになりました。現在、「データ分析/集計で時間がかかってしまっている」や「夜間バッチの実行 に時間がかかってしまっている」などの悩みを抱えている場合には、まず非クラスター化列ストア インデックスを作成してみることをお勧めします。夜間バッチの前に(バッチ実行時の最初の処理 として)非クラスター化列ストア インデックスを追加して、夜間バッチが完了したら削除する、と いった使い方もできるので、性能に問題を抱えている場合は、ぜひ試してみてください。

既存環境からの移行という観点では、PRIMARY KEY 制約がサポートされるようになった点は大きく、これまでの構成を大きく変更することなく移行できるようになりました。

今回の SQL Server 2016 の列ストア インデックスは、バッチ モードの性能向上や、更新性能の 向上も実現しているので、SQL Server 2014 のときに試したことがあるという方も、ぜひもうー 度試してみてください。

→ 列ストア インデックスはハイブリッドな動作が可能

完全なインメモリでの Operational Analytics(インメモリ OLTP と列ストア インデックスの 組み合わせ)を利用する場合には、インメモリ OLTP が完全にインメモリで動作するので、メモリ 上の制限(最大サイズが **1TB** までという上限)を受けます。したがって、数 TB (テラバイト)規 模になるなど、メモリに載りきらないデータ量になる場合には、インメモリ OLTP を利用すること ができません。

このような場合には、通常のリレーショナル テーブル (従来ながらのディスク ベースのテーブル) に、**列ストア インデックス**を追加して、Operational Analytics を実現することができます。列ス トア インデックスは、基本はインメモリで動作しますが (データ量が物理メモリに載りきる場合は インメモリで動作)、メモリに載りきらないデータがあった場合には、ディスクを利用して動作させ ることができるからです (ハイブリッドな動作ができます)。

このように性能向上に関するオプションが増えたことは、本当にワクワクします(今回の SQL Server 2016 の進化は、今後のデータベースのあり方を左右するのではないかと思えるほど、非常 に大きな可能性を感じています)。Operational Analytics に関しては、別途自習書を作成していま すので、そちらもぜひご覧いただければと思います。

→ DWH (データ ウェアハウス) は不要? ~リアルタイムなデータ分析へ~

最近は、データ分析/集計をできる限りリアルタイムで実現したいというニーズの高まりから、 <u>DWH(データ ウェアハウス)を構築しない</u>というケースが増えてきました。これまでは、次の図 のように、ETL(Extract/Transform/Load)を定期的なバッチ処理で実行して、DWH を構築し ているという形が主流でした。

<u>従来ながらの DWH では、データ分析に遅延がある</u>



このようなシステムの場合、ETL ツールを介すことによって(定期バッチでの実行になるため)、 データ分析には遅延が発生していました(データの鮮度が落ちていました)。

そこで、最近は、ハードウェアが進化したことも影響していますが(特に大容量メモリを安価に搭載できるようになったことも大きいのですが)、データウェアハウスを構築せずに、1 台のマシンで OLTP もデータ分析も実現してしまおうという動きが広まっています。





このように OLTP に DWH の役割も任せてしまえば、データ分析/集計をリアルタイムでできる ようになり、このような実装に最適なのが SQL Server 2016 の Operational Analytics です。 SQL Server 2016 では、インメモリ OLTP と列ストア インデックスが飛躍的に進化したことで、 こうした動きがさらに加速していくように思っています。

2.2 Operational Analytics の検証の詳細



前項で紹介した Operational Analytics の検証結果を再掲します。

* ベンチマークの結果の公表は、使用許諾契約書で禁止されていますが、その効果をわかりやすく表現するため、 日本マイクロソフト株式会社の監修のもと、数値を掲載しております。

* 検証は、SQL Server 2016 CTP 3.2 を利用して行ったので、<u>RTM 版ではさらに性能が向上</u>する可能性があります。

→ 検証の詳細

この検証で使用したテーブルの構成と1億件のデータは、次のように作成しています。

<u>検証で使用したテーブル構成/1億件のデータ</u>



3 種類のテーブル (ディスク ベースの通常テーブル、インメモリ OLTP のメモリ最適化テーブル、 メモリ最適化テーブルに列ストア インデックスを追加したもの)を同じ列構成 (col1~col5 の 5 列) で作成し、col1 (PRIMARY KEY) には IDENTITY で生成した連番、col2、col3、col4、col5 には乱数 (RAND 関数で生成した値)を格納しています。実際の 1 億件のデータは、次のように なっています。

E	SELECT	* FROM dat	:a100M				
100 %	• <						
111 紀	果 🚹	メッセージ					
	col1	col2	col3	col4	col5		
1	1	5023503	324864	8013	2010-05-28	01:12:00.000	
2	2	1159690	859684	6817	2013-09-02	02:08:00.000	
3	3	12554582	562761	7336	013-02-03	15:16:00.000	
4	4	11344397	873227	9046	col4 int	01:53:00.000	
5	5	6066207	697032	9903	0~9,999	12:58:00.000	
6	6	16121127	770155	233	の乱数	22:53:00:000	
7	7	4623271	172953	30; <mark>C(</mark>	DNVERT(int,	RAND() * 10	000)
8	81	6364128	971945	5264	2009-03-27	21:23:00.000	
9	9	·	35870	7549	2013-06-25	04:50:00:00	
10	1 п	coll int DENTITY र	825	ol3 int	11-07-24	00:31:00.000	COI5 datetime 2009/1/1 以降
11	1 1	からの連番	496 0~	/999,95 の乱数	12-07-31	13:04:00.000	の日付の乱数
12	12	124477	CONVER			000000) 00	CONVERT(int, RAND() * 2628000) + 1
13	13	12975051	CUNVER	VI (Int	, RAND() * 1	000000)	DATEADD(minute, @col5rnd, '2009/01/01')
14	14	6474654	477140	923	2012-04-09	14:07:00.000	
15	15	1967	col2 int	757	2011-12-09	05:35:00.000	
16	16	1020	19,999,99 の乱数	65	2009-10-14	19:40:00.000	
17	17				0	10:54:00.000	
18	18	CUNVERT (int, RAN	J() * :	20000000) ₁₇	05:17:00.000	

3 種類のテーブルには、同じデータが格納されるようにするために(公平な検証にするために)、 data100M という名前の中間テーブルに 1 億件のデータを格納していて、これを INSERT .. SELECT でそれぞれのテーブルにデータ複製するようにしています。

検証で使用した集計クエリ(col4 で GROUP BY)

検証では、次のように col4 列で GROUP BY (データ集計)をしたクエリを使用しました。

検証で使用したクエリ(1億件のデータを集計、col4 列で GROUP BY)
SELECT col4, COUNT(*) AS cnt FROM <テーブル名>
GROUP BY cold
ORDER BY col4

	SELECT col4, COUNT(*) AS cnt FROM InMem_HASH GROUP BY col4 ORDER BY col4								
100 '	% • <		III						
	1 結果 🔓 メッセージ								
	col4	cnt	col4 で GROUP BY を オスストズ 1万件の休用						
1	0	10210	9 るここで 1万件の相来 が返る						
2	1	9958							
3	2	10122	冬値にけ約1万件のデータがあり						
4	3	10112	1万件 * 1万件 = 1億件のデータ						
5	4	10114							
6	5	10007							

col4 列には、0~9,999 の範囲の乱数が格納されているので、1 万件の結果が返ります。ディス

ク ベースの通常テーブルで、このクエリを実行したときの結果は、次のとおりです(SET STATISTICS TIME/IO ON で計測した実行時間と I/O 数)。



また、このときの実行プランは、次のように Clustered Index Scan になっています。

SELECT col4, COUNT(*) AS cnt FROM OnDisk_CL GROUP BY col4 ORDER BY col4	
	ディスクベースでは Clustered Index Scan
13 AVE-ジョン 第7方カン フロリコ: フロリコスト (バウナ相対): 1004 SELECT col4, COUNT(*) AS cnt FROM OnDisk CL GROUP BY col4 ORDER BY col4	
BELECT Parallelism Sort Compute Scalar Hach Match Parallelism JA:: 0 % JA:: 0 % JA:: 0 % JA:: 0 %	Hash Match (Partial Aggregate) (Art: 13 t

これに対して、**インメモリ OLTP** の**メモリ最適化テーブル**で同じクエリを実行したときの実行プ ランは、次のようになります。

SELECT col4, CO ORDER BY col4	UNT(*) AS ont FRC	M InMem_HASH GROUP	BY cold	インメモリ OLTP の モリ最適化テーブルに対して実行	ī)(Table Scan (並列実行)
□ メッセージ ■ 実行フラン クエリ 1: クエリ コスト (SELECT col4, COUNT	/パッチ相対): 100% (*) AS cnt FROM :	InMem_HASH GROUP BY	col4 ORDER BY co.	14		
Parallelism ====================================	<u>ی ایک</u> Sort اکرا: ۵ ۴	Compute Scalar	Hash Match (Aggregate)	Parallelism (Repartition Streams) JAF: 0 %	Hash Match (Partial Aggregate) _↓\: 56 %	Table Scan [InMem HASH] 그人가: - 44 %

Table San で実行されて、Parallelism (並列実行) になっているのがポイントです (実行時間は、 前述のグラフのとおり、1.6 秒です)。SQL Server 2016 からは<u>並列プランに対応</u>するようになっ たので、Parallelism で処理されています。

これと同じクエリを **SQL Server 2014** で実行した場合は、次のように Parallelism にはなりま <u>せん</u> (SQL Server 2014 は、並列プランに対応していないので、シングル スレッド実行になって しまいます)。



同じクエリを、SQL Server 2016 のインメモリ OLTP(メモリ最適化テーブル)に**クラスター化 列ストア インデックス**を追加したテーブルで実行すると次のようになります。

SELECT col4, COUNT(*) AS cnt FROM InMem_HASH_ccsi GROUP BY col4 ORDER BY col4 100 % - く III インメモリ OLTP のメモリ最適化テーブルに クラスター化列ストア インデックスを追加
(10000 行処理されました) テーブル 'InMem_HASH_ccsi'。スキャン回数 24、論理読み取り数 0、物理読み取り数 0、先行読み取り数 0、LOB 論理読み取り数 テーブル 'InMem_HASH_ccsi'。セグメント読み取り 96、セグメント スキッブ 0。 テーブル 'Worktable'。スキャン回数 0、論理読み取り数 0、物理読み取り数 0、先行読み取り数 0、LOB 論理読み取り数 0、LOB
SOL Server 実行時間: SOL Server 実行時間: CPU 時間 = 3016 ミリ秒、経過時間 = 198 ミリ秒。 実行時間は わずか 198ミリ秒 Columntore インデックス スキャン (並列実行)
カビリ 1: クビリ コスト (パッチ相対): 100% SELECT col4, COUNT(*) AS cnt FROM InMem_HASH_ccsi GROUP BY col4 ORDER BY col4
SELECT Parallelism Sort Compute Scalar Hash Match Columnstore 1/5"9/J, Z#*D (NonClus_ JZA: 0 % JZA: 0 % JZA: 0 % JZA: 2 % JA: 27 %

実行プランには、Columnstore インデックス スキャンと表示されて、列ストア インデックスが 利用されていることが分かります。また、実行時間もわずか 198 ミリ秒であることが分かり、1 億 件のデータ集計を 200 ミリ秒以内で実行できるという、圧倒的な性能(ディスク ベースよりも 10 倍も速い)が出ていることを確認できます。

このように、SQL Server 2016 のインメモリ OLTP は、列ストア インデックスと融合すること によって、データ分析/集計にも強くなっています。

→ インメモリ OLTP は、もちろん OLTP に強い

ここまでは、インメモリ OLTP + 列ストア インデックス (OLTP とデータ分析の両立) が目玉の 新機能であると説明しましたが、インメモリ OLTP は、その名のとおり OLTP に強いのが最大の 特徴です。例えば、SQL Server 2014 の時点で、次のような導入効果がありました。

社名/システム概要	システムの特性	導入効果	
bwin 社 オンライン ゲームなど を提供	大量のユーザーによる同時更新。 トランザクションとしては小さい。 毎日 15万人以上のアクティブ ユー ザー。毎年 100万人以上の新規ユー ザーが増加。 ASP.NET のセッション状態データ ベースで利用。	約 16.7倍の性能向上 15,000 バッチ要求/sec が 250,000 バッチ要求/sec へ向上。 450,000 バッチ要求/sec も確認 (約 30倍の性能向上)	
SBIリクイディティ・ マーケット株式会社 FX 取引(オンライン トレード)	大量のユーザーによる同時更新。 トランザクションとしては小さい。 顧客のトレーディング データ(取引 履歴)をリアルタイムに集計	約 2.5倍の性能向上 52,080 件/sec が 131,921 件/sec へ向上。 ピーク時の Latency(遅延)は、 約 4秒だったのを、1秒以下に短縮	
Edgenet 社 商品データを提供する データ プロバイダー	DWH 環境でのステージング テーブル で利用。ETL 処理など。 バッチ処理での大量のデータ更新。 更新中の参照アクセスあり	約8~11倍の性能向上 2時間20分かかっていたバッチ処 理が、わずか20分に短縮。 更新中の参照アクセスが、 更新によってブロック(ロック待 ちなど)されることがなくなり、 読み取り性能も向上	
TPP 社 臨床ソフトウェアの提供	大量のユーザーによる同時更新。 トランザクションとしては小さい。 1秒あたり 72,000 ユーザーが 同時アクセス(ピーク時)	約 7倍の性能向上 34,700 Transaction/sec が 数十万 Transaction/sec へ向上	
弊社のお客様 A社 ポイントカード システム	大量のユーザーによる同時更新。 トランザクションとしては小さい。 ポイントカードにおける ポイントの入金や出金、残高照会処理	約 2.8倍の性能向上	

* 弊社執筆の SQL Server 2014 実践シリーズ No.1「インメモリ OLTP の実践的な利用方法」の 1.3 から引用

インメモリ OLTP は、<u>ロック フリー/ラッチ フリーのアーキテクチャ</u>なので (ロックとラッチを 利用しないアーキテクチャで、ロック待ちやラッチ待ちを回避することができるので)、**大量のユー ザーによる同時更新が発生するシステム** (いわゆる OLTP システム) で大きな効果を発揮します。

bwin 社 (bwin.party) では、ASP.NET のセッション状態データベースにインメモリ OLTP を採 用して、SQL Server 2014 のときにはラボ環境で 45万 Batch Requests/sec (1秒あたり 45 万件ものバッチ要求を処理)を達成していました。さらに、昨年開催された SQL Server の世界最 大規模のイベントである「PASS Summit 2015」では、同システムで 66万 Batch Requests/sec を達成したというアナウンスもありました(以下のスライドの最後の行に記載)。



1 秒あたりに 66 万ということは、1分(60 秒)に換算すると 66 万*60 秒=3,960 万ものバッ チ要求を処理できるようという数値です。このような驚異的な数値が出せるのは、インメモリ OLTP ならではです。

◆ 検証結果: インメモリ OLTP は、単純 INSERT にも強い

単純 INSERT の性能検証は、SQL Server 2014 の実践シリーズでも行いましたが、今回、日立様のハードウェア(詳細は後述)をお借りすることができたので、SQL Server 2016 CTP 3.2 でも検証を行ってみました。想定しているシステムは、大規模イベントでの登録や予約、投票のみを受け付けるようなシステムや、IoT などセンサー系データを常に INSERT し続けるようなシステムで、多数のユーザーが同時に INSERT を行う状況を検証しました。結果は、次のとおりです。



* ベンチマークの結果の公表は、使用許諾契約書で禁止されていますが、その効果をわかりやすく表現するため、 日本マイクロソフト株式会社の監修のもと、数値を掲載しております。

* 検証は、SQL Server 2016 CTP 3.2 を利用して行ったので、<u>RTM 版ではさらに性能が向上</u>する可能性があります。

ディスク ベースの通常テーブルでは 27分19秒かかった実行時間が、インメモリ OLTP のメモ リ最適化テーブルの Durable(永続化:SCHEMA_AND_DATA)にした場合は8分32秒(3.2 倍の性能向上)、SCHEMA_ONLY(永続化なし)にした場合はわずか5分48秒で完了し、4.7倍 もの性能向上を確認することができました。また、このときの1秒あたりのトランザクション数を 計算すると、次のようになります(実行時間を5,000万件で割り算したもの)。

		Transaction/sec	
ディスク ベース	OnDisk	30,506	
	Durable	97,657	1秒あたり 14.4万件も
イ <i>プ</i> ズモリ 0LIP	SCHEMA_ONLY	143,673 🖌	INSERT できる

SCHEMA_ONLY では、1 秒あたり 14.4 万件も INSERT できることを確認できました。

この検証は、10個の列を持ったテーブルを作成して、そのテーブルヘデータを1件ずつ INSERT を行って、5,000万件分を INSERT したときの実行時間を測定したものです。データには、実際 のアプリケーションを想定して、乱数を利用し、同じ値が格納されないようにしています。





このテストは、<u>Bulk Insert や SELECT INTO などの一括操作を利用したものではなく</u>、またネイ ティブ コンパイル ストアド プロシージャ(インメモリ OLTP の性能を向上させることができる ストアド プロシージャ機能)の中でループ処理を記述して、複数件をまとめて INSERT するよう なものでもなく、<u>実際のアプリケーションを想定して</u>、**1件ずつの INSERT 処理を行った場合の** 実行時間を測定しています。ネイティブ コンパイル ストアド プロシージャ内の処理も、前述の図 のように 1件の INSERT を行うものしか記述していません。接続に関しても、<u>1</u>件の INSERT ご とに Open と Close を行っています。

テストで利用したアプリケーション(VB+ADO.NET)

このテストで利用したアプリケーションは、実際のアプリケーションを想定して、次のように .NET (VB+ADO.NET) で作成しています (C#で作成してもほぼ同じコードになります)。



このコードは、ディスク ベースのテーブルへ INSERT を行う場合のものですが、インメモリ OLTP のネイティブ コンパイル ストアド プロシージャを利用する場合には、次のように 2 ヶ所 を修正しています。

Using on As New SqlConnection(onstr) Using ond As New SqlCommand() on.Doen() omd.Connection = on 'omd.CommandText = "INSERT INTO dbo.OnDisk_10col_CL" & "VALUES (@c2, @c3, @c4, @c5, @c6, @c7, @c8, @c9, @c10)"
'ネイティブ コンバイル ストアド ブロシージャの場合 CommandType を変更 cmd.CommandType = CommandType.StoredProcedure
cmd.Lommandlext = sp_InMem_IUcol_Dura_HASH ネイティブ コンパイル ストアド Dim p1 As SalParameter = cmd.Parameters.Add("@c2", Sa プロシージャの名前に変更
p1.Value = col2 Dim p2 As SqlParameter = cmd.Parameters.Add("@c3", SqlDbType.Int)
p2.Value = col3 Dim p3 As SqlParameter = cmd.Parameters.Add("@c4", SqlDbType.NVarChar, 20) p3 Value = col4
Dim p4 As SqlParameter = cmd.Parameters.Add("@c5", SqlDbType.NVarChar, 50) p4 Value = col5

SqlCommand の CommandType を変更して、CommandText をネイティブ コンパイル スト アド プロシージャの名前に変更するだけで、ネイティブ コンパイル ストアド プロシージャを実 行することができます。

以上のコードを、**多重実行(並行実行)**して、実行時間を計測したのが前掲のグラフです(100多 重で実行したときを測定しました)。

このように、インメモリ OLTP は、常に INSERT をし続けるようなシステムでも大きな効果を発 揮します。

ディスク ベースの通常テーブルで性能が向上しない理由 〜ラッチ待ち〜

ディスク ベースの通常テーブルは、多重度が上がれば上がるほど、性能が頭打ちになります(多重 度が上がるほど遅くなります)。性能が上がらない一番の理由は、**ラッチ待ち**(Latch Wait)や**ロッ ク待ち**(Lock Wait)などの**ブロッキング**によるものです。これに対して、インメモリ OLTP では <u>ラッチ/ロックを利用しないアーキテクチャ</u>なので、ラッチ待ち/ロック待ちに悩まされることは ありません(インメモリ OLTP であれば、多重度が上がってもスケールします)。

ディスク ベースで利用される**ラッチ**(Latch)には、主に**ページ ラッチ**(PAGELATCH_SH、 PAGELATCH_EX)と IO **ラッチ**(PAGEIOLATCH_SH、PAGEIOLATCH_EX)がありますが、 前者は、ページへの同時アクセスを制御するための、SQL Server が内部的にページに対してかけ るロックのようなもの、後者は、データ ファイル (.mdf)からメモリ内のデータ バッファへペー ジを取り出すとき/書き出すときにかけるロックのようなものです。

ページ ラッチ (**PAGELATCH_SH** や **PAGELATCH_EX**) は、次の図のように、<u>同じページに対して、多数のユーザーからの同時アクセスが発生した場合</u>を制御するためのもので、同時に同じペ ージを操作させないように、後からきたラッチを "**待ち**" にします (ページ ラッチ待ちの発生)。

同じページへの同時アクセスは、ページラッチ待ちが発生する


更新系のステートメント (INSERT/UPDATE/DELETE) では、**PAGELATCH_EX** (排他ページラ ッチ。EX は Exclusive : 排他の略) をかけにいき、SELECT ステートメントによる参照時には **PAGELATCH_SH** (共有ページラッチ。SH は Shared : 共有の略) をかけにいきます。このとき、 先にアクセスしている処理がある場合は (ラッチが既にかかっている場合には)、それが解放される まで、**"待ち"** が発生します。このような、同時アクセスによって待ちが発生する状態は、**ラッチ競** 合 (Latch Contention) とも呼ばれています。

このように、ラッチ待ちが発生すると、ユーザー数が増えれば増えるほど、ラッチ待ちも増えることになるので、スループットが低下していきます。



これでは、同時実行数が増えれば増えるほど、システムの処理能力は頭打ちになり、スケールしなくなってしまいます。

これに対して、インメモリ OLTP が採用している、ラッチを利用しない「**ラッチ フリー**」のアー キテクチャであれば、ユーザー数が増えてたとしても性能低下は発生しにくくなります。





ラッチ待ちの様子 ~インデックスの最終ページがホットスポット~

今回の検証で利用したテーブルは、col1 列を IDENTITY(1, 1) の PRIMARY KEY に設定し ているので、ディスク ベースのテーブルでは、多数のユーザーが同時にデータを追加することによ って、次のようにインデックスの最終ページにアクセスが集中します。

<u>インデックスの最終ページでページ ラッチ待ちが多発する</u>



ディスク ベースのテーブルの場合は、PRIMARY KEY 制約を作成することで、自動的に**クラスタ** ー化インデックス(B-tree)が作成されるので、データが追加されると、最終ページに値が集中す ることになります。連番系の列(IDENTITY を設定した列や、シーケンスを設定した列)など、デ ータを追加するたびに連続した値(1、2、3、…)が格納されていくような場合には、このような インデックスの最終ページでページ ラッチ待ちが多発する(最終ページがホット スポットになる) ことがよくあります。これは、シングル実行(1人のユーザーによる単体実行)では、発生しない ものですが、多数のユーザーが同時にデータを追加する場合には発生してしまいます。

ラッチ待ちが発生しているかどうかは、パフォーマンス カウンターの Latch Waits/sec (SQLServer: Latches オブジェクト)を参照することで簡単に確認することができます。実際 に、検証を行ったときのパフォーマンス カウンターは、次のようになりました。



100多重で 5,000万件を INSERT しているときのラッチ待ちの様子(ディスク ベース)

Batch Request(バッチ要求数)が 3万ぐらいを推移しているのに対して、ラッチ待ち(Latch Waits)が 7.5万(2倍以上)も発生してしまっています。このようなラッチ待ちは、多重度が 200、300 と上がっていくとさらに顕著に表れて、性能がどんどん低下していくことになります。

一方、インメモリ OLTP ではラッチ待ちは発生しないので、次のような性能が出ます。



インメモリ OLTP (SCHEMA_ONLY) の場合

このように、インメモリ OLTP は、多数のユーザーからの INSERT および常に INSERT をし続 けるようなシステムでも大きな効果を発揮します。

なお、今回の検証では、実際のアプリケーションを想定していたので、1 件の INSERT ごとに接続の Open と Close 処理を行っていましたが、もし接続をキープできるような状況(接続の Open と Close を行わずに同じ接続を再利用)であれば、次のような性能を出すことができます。



ディスク ベースでは、接続をキープしてクエリ負荷が上がる分だけ性能が逆に落ちるのに対して、 インメモリ OLTP の SCHEMA_ONLY では、1 秒あたりに 26 万件ものトランザクションを処理 できることを確認できました。

インメモリ OLTP の基本的な利用方法については、別途作成している Operational Analytics の 自習書に記載していますので、そちらもぜひご覧いただければと思います。

2.3 Operational Analytics の検証環境

前掲の Operational Analytics の検証では、株式会社 日立製作所様のご厚意でハードウェアをお借りすることができましたので、ここで環境をご紹介します。

今回利用させていただいた日立様の施設は、品川にある「**ハーモニアス・コンピテンス・センター**」 です。



検証で利用させていただいたハードウェアは、「統合サービスプラットフォーム BladeSymphony BS500」の 24 コア マシン(Xeon E5-2697v2 を 2 個搭載したサーバー)とエンタープライズ ディスク アレイ システムである「Hitachi Virtual Storage Platform G1000」です。

検証で利用させていただいたハードウェア



サーバー BladeSymphony BS500 ブレード: BS520H サーバブレード CPU: Xeon E5-2697v2 × 2 (24core) メモリ: 128G 内蔵HDD: 600GB×2 ストレージ接続: 8Gbps FC

ストレージ Hitachi Virtual Storage Platform G1000 1.6TBフラッシュモジュールドライブ×9 (RAID5:3D+1P (4915.19GB), Iミュレーション:OPEN-V×2、スペアディスク×1

STEP 3. セキュリティ強化

この STEP では、「動的データ マスク」や「行レベル セキュリティ」、「Always Encrypted」、「TDE の強化」、「テンポラル テーブルによる Audit」など、SQL Server 2016 CTP 3.2 で提供されたセキュリティ強化を実現できる機能を説明 します。

この STEP では、次のことを学習します。

- ✓ 動的データ マスクによる情報漏洩対策
- ✓ 行レベル セキュリティによるセキュリティ強化
- ✓ Always Encrypted による列データの暗号化
- ✓ TDE (透過的なデータ暗号化)の性能向上、インメモリ OLTP 対応
- ✓ テンポラル テーブルによる Audit (監査証跡)

3.1 セキュリティの強化

SQL Server 2016 には、セキュリティを向上させることができる機能が数多く提供されています。 その主なものは、次のとおりです。

- 動的データ マスク (Dynamic Data Masking)
 顧客情報(クレジット カード番号やマイナンバーなど)や機密情報をマスク(別の値に
 置換)して、情報漏洩を防止できる機能
- 行レベル セキュリティ(Row Level Security)

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

● Always Encrypted による列データの暗号化

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

● TDE (透過的なデータ暗号化)の性能向上、インメモリ OLTP 対応

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

● テンポラル テーブルを利用した Audit (コンプライアンスにおける監査証跡)

過去の更新履歴を自動保存できる「テンポラル テーブル」機能を利用して、Audit(監査 証跡)が可能に

この章では、これらの機能について、1つ1つ具体的な利用方法を説明します。

3.2 動的データ マスクによる情報漏洩対策

動的データ マスク(Dynamic Data Masking)は、クレジット カード番号やマイナンバー、メ ール アドレスなどの顧客情報/機密情報をマスク(異なる値に置換)して、セキュリティ強化(情 報漏洩の防止)を実現できる機能です。これは、次のように利用することができます。





このように、動的データ マスクでは、権限のないユーザーに対して、重要な情報を参照できないようにする(マスクされたデータしか参照できないようにする)ことができるので、顧客情報や機密 情報 (クレジット カード番号やマイナンバーなど)のデータ流出を抑えることができるようになり ます。

動的データ マスクは、次のように CREATE TABLE ステートメントでのテーブル作成時または ALTER TABLE ステートメントでのテーブル変更時に、列に対して設定することができます。

```
-- 動的データ マスクの設定例
CREATE TABLE maskTest1
( colA int
, colB varchar(200) MASKED WITH (FUNCTION = 'default()')
```

列名、データ型に続けて、MASKED WITH を付けて、FUNCTION でどのようにマスクするのか を指定します。上の例では「default」という関数を利用していますが、この場合は、文字列データ を「xxxx」のようにマスク(置換)することができます。FUNCTION には、default の他に、 数値を乱数化する「random」や、電子メール アドレスをマスクできる「email」、任意の文字列 にマスクすることができる「partial」などがあります。

Let's Try

それでは、これを試してみましょう。

1. まずは、動的データ マスクを試すためのデータベースとユーザーを作成します。

```
-- データベース「maskTestDB」の作成
CREATE DATABASE maskTestDB
go
-- データベース ユーザー「UserX」の作成
USE maskTestDB
CREATE USER UserX WITHOUT LOGIN
```

データベース名は「maskTestDB」、データベース ユーザーは、「UserX」という名前で作成 し、WITHOUT LOGIN を付けることで、簡易的なユーザーにしています。このように作成 したユーザーは、ログインするためのユーザーとしては利用することができないので、現実的 な利用方法ではないのですが、セキュリティ機能を試すときにのみ便利なユーザーになります (このようなユーザーは、ログインをすることはできませんが、後述の EXECUTE AS ステー トメントを利用して、接続をシミュレートして利用することができます。セキュリティ機能で は、複数のユーザーでの動作を試したい場面が多々あるので、こういった場合に役立つユーザ ーの作成方法です)。

 次に、CREATE TABLE ステートメントを利用してテーブルを作成します(テーブル名は maskTest1 として、colA、colB、colC、colD の 4 つの列を作成)。このときに、colB、 colC、colD の 3 つの列に対しては、次のように動的データ マスクを設定します。

```
-- テーブルの作成。colB、colC、colD 列にマスクを設定
USE maskTestDB
CREATE TABLE maskTest1
( colA int
,colB varchar(200) MASKED WITH (FUNCTION = 'default()')
,colC int MASKED WITH (FUNCTION = 'default()')
,colD datetime MASKED WITH (FUNCTION = 'default()')
```

colA はデータ型「int」の通常の列、colB は「varchar(200)」に続けて MASKED WITH を記述し、FUNCTION には default を指定してマスクを設定、colC は「int」で同様にマ スクを設定、colD は「datetime」で同様にマスクを設定します。default を指定した場合 には、文字データは「xxxx」、数値データは「0」、日付データは「1900-01-01 00:00:00.000」 にマスクできるようになります。

3. 次に、データを 2 件追加します (データは適当なものでかまいません)。

```
-- データを 2件追加
INSERT INTO maskTest1
VALUES (1, 'AAAAA', 111, '2015/10/30')
,(2, 'BBBBB', 222, '2015/11/30')
```

4. 続いて、追加したデータを確認しておきます。

```
-- データの確認
SELECT * FROM maskTest1
```

100 % - < ■ 結果 a メッセージ colA colB colC colD 1 1 AAAAA 111 2015-10-30 00:00:00.000 2 2 BBBBB 222 2015-11-30 00:00:00.000	データの確認 SELECT * FROM maskTest1							
colA colB colC colD 1 1 AAAAA 111 2015–10–30 00:00:00.000 2 2 BBBBB 222 2015–11–30 00:00:00.000	100% - <							
1 AAAAA 111 2015-10-30 00:00:00:000 2 2 BBBBB 222 2015-11-30 00:00:00:000		colA	colB	colC	colD			
2 2 BBBBB 222 2015-11-30 00:00:00000	1	1	ААААА	111	2015-10-30 00:00:00.000			
-	2	2	BBBBB	222	2015-11-30 00:00:00.000			

動的データ マスクでは、特権ユーザー(管理者アカウント)に関しては、通常どおりにデータ を参照することができます。

5. 次に、一般ユーザーからのデータ アクセスを試すために、最初に作成したユーザー「UserX」 に対して SELECT 権限を付与します。

```
    -- データベース ユーザーに、テーブルに対する SELECT 権限を付与
GRANT SELECT ON maskTest1 TO UserX
```

 次に、EXECUTE AS ステートメントを利用して、「UserX」でログインしたときの動作をシ ミュレートします(EXECUTE AS ステートメントは、USER= で指定したユーザーでログイ ンしたときをシミュレートすることができ、REVERT ステートメントを実行するまでの間、 そのユーザーでのログインをシミュレートできます)。

```
-- EXECUTE AS で UserX をシミュレート
EXECUTE AS USER = 'UserX'
SELECT * FROM maskTest1
REVERT
```

	EXECUTE AS で UserX をシミュレート EXECUTE AS USER = 'UserX' SELECT * FROM maskTest1								
1	LOO % Ⅲ 結	REVERT マベロ 果 🛅 🤉	いセージ		マスクされたデータ が表示される				
		colA	colB	colC	colD				
	1	1	××××	0	1900-01-01 00:00:00.000				
	2	2	××××	0	1900-01-01 00:00:00.000				

このように、一般ユーザーがテーブルにアクセスした場合は、動的データ マスクが有効になって、データがマスクされていることを確認できます。今回は、colB、colC、colD に対して、 FUNCTION(マスクの関数)で default を指定しているので、colB(文字データ)は「xxxx」、 colC (数値データ) は「0」、colD (日付データ) は「1900-01-01 00:00:00.000」にマス クされていることを確認できます。

このように、動的データマスクを利用すれば、一般ユーザーから簡単に情報を隠せるように なるので大変便利です。

→ FUNCTION に email を指定して電子メール アドレスをマスク

次に、マスクの関数として「email」を利用して、電子メール アドレスをマスクしてみましょう。

1. これを試すために、既存の「maskTest1」テーブルに対して、ALTER TABLE ステートメントで列を追加してみましょう(colE という名前の列を追加します)。

```
ALTER TABLE maskTest1
ADD colE varchar(200) MASKED WITH (FUNCTION = 'email()')
```

このように、動的データ マスクは、ALTER TABLE ステートメントで列を追加するときに指 定することもできます。マスクの関数に、「FUNCTION = 'email()'」と指定することで、電 子メール アドレスをマスクできるようになります。

2. 次に、データを 1 件追加(colE 列には電子メール アドレス形式の文字データを指定)して みましょう。

```
-- データを 1件追加
INSERT INTO maskTest1
VALUES (3, 'CCCCC', 333, '2015/11/30', 'ccc@test.local')
-- データの確認
SELECT * FROM maskTest1
```

データの確認 SELECT * FROM maskTest1 100% ◆ <								
		r u⊒ 2 colA	colB	colC	colD	colE	_	
1		1		111	2015-10-30 00:00:00.000	NULL		
2		2	BBBBB	222	2015-11-30 00:00:00.000	NULL		
3		3	CCCCC	333	2015-11-30 00:00:00.000	ccc@test.local		

3. 次に、一般ユーザー「UserX」からアクセスして、データがマスクされることを確認します。

EXECUTE AS USER = 'UserX'
SELECT * FROM maskTest1
REVERT

100 8	EXECUTE AS USER = 'UserX' SELECT * FROM maskTest1 REVERT に変換されている									
	colA	colB	colC	colD	colE					
1	1	××××	0	1900-01-01 00:00:00.000	NULL					
2	2	××××	0	1900-01-01 00:00:00.000	NULL					
3	3	××××	0	1900-01-01 00:00:00.000	cXXX@)	OOOX.com				

「ccc@test.local」というデータが、「cXXX@XXXX.com」という形でマスクされていること確認できます。このように、電子メール アドレスをマスクしたい場合には、email 関数を利用すると便利です。

→ FUNCTION に random を指定して数値データを乱数化

次に、マスクの関数として「random」を利用して、数値データを乱数にマスクしてみましょう。

 これを試すために、既存の colC 列(int データ型で作成した列)を利用します。既存の列に 対して、動的データ マスクを設定/変更するには、次のように ALTER TABLE ステートメ ントの ALTER COLUMN で、ADD MASKED WITH を利用します。

```
ALTER TABLE maskTest1
ALTER COLUMN colC
ADD MASKED WITH (FUNCTION = 'random(1, 100)')
```

random 関数では、2 つの引数に、乱数の範囲を指定します。上の例のように「(1, 100)」 と指定した場合は、1~100 の間の乱数にマスクすることができます。

 設定が完了したら、一般ユーザー「UserX」からアクセスして、データがマスクされることを 確認してみましょう。

```
EXECUTE AS USER = 'UserX'
SELECT * FROM maskTest1
REVERT
```



♦ FUNCTION に partial を指定して文字データを任意の文字列にマスク

次に、マスクの関数として「partial」を利用して、文字データを任意の文字列にマスクしてみます。

1. これを試すために、既存の colB 列(varchar(200) データ型で作成した列)を利用してみ ましょう。ALTER TABLE .. ALTER COLUMN ステートメントを次のように実行します。

```
ALTER TABLE maskTest1
ALTER COLUMN colB
ADD MASKED WITH (FUNCTION = 'partial(1, "zzzzz", 1)')
```

partial 関数は、3 つの引数で「接頭辞, padding, 接尾辞」という形で利用します。上の例のように「(1, "zzzzz", 1)」と指定した場合は、接頭辞が1、接尾辞も1 なので、先頭と最後の1 文字はそのまま残して、間の文字を zzzzz で埋めて (パディングして) マスクすることができます。

2. 設定が完了したら、一般ユーザー「**UserX**」からアクセスして、データがマスクされることを 確認してみましょう。

```
REVERT
   EXECUTE AS USER = 'UserX'
   SELECT * FROM maskTest1
                              先頭と最後の1文字がそのまま
   REVERT
                              残って、間が zzzzz に埋められ
100 % 👻 <
                                  てマスクされている
💷 結果 🛅 メッセージ
    colA colB colC
                      colD
                                           colE
          AzzzzA 68
                       1900-01-01 00:00:00.000 NULL
    1
1
     2
          BzzzzB 88
                       1900-01-01 00:00:00.000 NULL
2
3
     3
          Czzzzc 40
                       1900-01-01 00:00:00.000 cXXX@XXXX.com
```

3. 次に、partial 関数の引数を次のように変更(接尾辞を 1 から 0 に変更)してみましょう。

ALTER TABLE maskTest1 ALTER COLUMN colB ADD MASKED WITH (FUNCTION = 'partial(1, "zzzzz", 0)')

4. 設定後、一般ユーザー「UserX」からアクセスして、結果を確認してみます。

```
EXECUTE AS USER = 'UserX'
SELECT * FROM maskTest1
REVERT
```

EXECUTE AS USER = '**UserX**' SELECT * FROM maskTest1

EXECUTE AS USER = 'UserX' SELECT * FROM maskTest1 REVERT 100% - く 先頭の 1文字がそのまま残って、 残りが zzzzz に埋められる								
	結果	K 🛅 🤉	ハッセージ					
		colA	colB 🧹	colC	colD		colE	
1		1	Azzzz	16	1900-01	1-01 00:00:00.000	NULL	
2		2	Bzzzz	20	1900-01	1-01 00:00:00.000	NULL	
3		3	Czzzzz	45	1900-01	1-01 00:00:00.000	cXXX@XXXX.com	

今度は、接尾辞を 0 に指定しているので、最後の文字は残っていないことを確認できると思います。

このように partial 関数を利用すると、文字列データを任意の文字列にマスクすることがで

きるので大変便利です。

→ 設定の確認 ~スクリプト生成、masked_columns カタログ ビュー~

動的データ マスクの設定を確認するには、**masked_columns** カタログ ビューを利用する方法 がありますが、これよりも簡単なのが、<u>Management Studio のスクリプト生成機能</u>を利用する方 法です。これを利用するには、次のように、該当テーブルを右クリックして、[**テーブルをスクリプ ト化**] から [**新規作成**] → [**新しいクエリ エディター ウィンドウ**] をクリックします。



これで、**CREATE TABLE** ステートメントが生成されて、動的データ マスクの定義も確認することができます。

masked_columns カタログ ビューを利用する場合は、次のように「**masking_function**」列 で定義を確認することができます。

```
SELECT name, masking_function, *
FROM sys.masked_columns
WHERE object_id = OBJECT_ID('maskTest1')
```

```
SELECT name, masking_function, *
    FROM_sys.masked_columns
     WHERE object_id = OBJECT_ID('maskTest1')
    -
100 %
📰 結果 🚹 メッセージ
           masking_function
                                           name column_id
     name
                               object id
                                                            system_type_id
                                                                           user type id
     colB
            partial(1, ″zzzzz″, 0) 645577338
                                          colB
                                                  2
                                                             167
                                                                            167
            random(1, 100)
2
     coIC
                                645577338 colC
                                                  3
                                                            56
                                                                           56
            default()
                                                             61
                                                                           61
3
     colD
                                645577338 colD
                                                  4
            email()
                                645577338 colE
                                                  5
                                                             167
                                                                            167
4
     colE
```

このように、動的データ マスクを利用すれば、データを簡単にマスク(違う値に変換)することが できるので、クレジット カード番号やマイナンバー、メール アドレスなどの顧客情報/機密情報 をマスクする目的で利用することができます(セキュリティの強化、情報漏洩/データ流出の防止 を実現することができます)。

その他、動的データ マスクに関する最新情報は、オンライン ブックの次のトピックが参考になる と思います。

Dynamic Data Masking

https://msdn.microsoft.com/en-us/library/mt130841.aspx

Strong Passwords Row-Level Security	Dynamic Data Masking SQL Server 2016
SQL Server Encryption	Updated: November 19, 2015
Dynamic Data Masking	Applies To: SQL Server 2016 Preview
SQL Server Certificates and Asymmetric Keys	Topic Status: Some information in this topic is preview and subject to change in future releases. Preview information describes new features or changes to existing features in SQL Server 2016 Community Technology Preview 3 (CTP 3.0).
 SQL Server Audit (Database Engine) 	Dynamic data masking limits sensitive data exposure by masking it to non-privileged users. Dynamic data masking helps prevent unauthorized access to sensitive data by enabling customers to designate how much of the sensitive data to reveal with minimal impact on the application layer. It's a data protection feature that hides the sensitive data in the result set of a query over designated database fields, while the data in the database is not changed. Dynamic data masking is easy to use with existing applications, since masking rules are applied in the query results. Many applications can mask sensitive data without modifying existing queries.
	For example, a call center support person may identify callers by several digits of their social security number or credit card number, but those data items should not be fully exposed to the support person. A masking rule can be defined that masks all but the last four digits of any social security number or credit card number in the result set of any query. For another example, by using the appropriate data mask to protect personally identifiable information (PII) data, a developer can query production environments for troubleshooting purposes without violating compliance regulations.
	The purpose of dynamic data masking is to limit exposure of sensitive data, preventing users who should not have access to the data from viewing it. Dynamic data masking does not aim to prevent database users from connecting directly to the database and running exhaustive queries that expose pieces of the sensitive data. Dynamic data masking is complementary to other SQL Server security features (auditing, encryption, row level security) and it is highly recommended to use this feature in conjunction with them in addition in order to better protect the sensitive data in the database.
	Dynamic data masking is available in SQL Server 2016 Community Technology Preview 3 (CTP 3.0). For Azure SQL Database, see Get started with SQL Database Dynamic Data Masking (Azure Preview portal).

3.3 行レベル セキュリティによるセキュリティ強化

行レベル セキュリティは、行レベルのアクセス制御を実現できる機能です。これを利用すれば、<u>ユ</u> ーザーに対して、参照できる「行」を制限できるようになるので、次のようにユーザーごとに、見 せたい行を制限できるようになります。

<u>行レベル セキュリティ</u>



このように行レベル セキュリティを利用すれば、User1 は User1 のデータのみ、User2 は User2 のデータのみしか参照できないように制限することができます。また、すべてのデータを 参照できるユーザー(図では Manager)を設定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。

1. まずは、行レベル セキュリティを試すためのデータベースとユーザーを作成します。

```
-- データベース「rsTestDB」の作成
CREATE DATABASE rsTestDB
go
-- データベース ユーザー User1、User2、Manager の作成
USE rsTestDB
CREATE USER User1 WITHOUT LOGIN
CREATE USER User2 WITHOUT LOGIN
CREATE USER Manager WITHOUT LOGIN
```

データベース名は「rsTestDB」、データベース ユーザーは「User1」、「User2」、「Manager」 という名前で作成します(WITHOUT LOGIN は、現実的なシナリオでは利用しない使い方 になりますが、行レベル セキュリティを試すためだけの簡易的なユーザーの作成方法になり ます)。 2. 次に、CREATE TABLE ステートメントを利用してテーブルを作成します。

```
-- テーブル「rsTest1」の作成
CREATE TABLE rsTest1
( colUserName sysname
, colA int
, colB varchar(200) )
```

テーブル名は「**rsTest1**」とし、「**colUserName**」列は、アクセス可能なユーザーの名前を格納するために利用します。「**colA**」と「**colB**」には、特に意味はないので、適当な列の名前で大丈夫です。

3. 次に、データを4件追加します。

```
-- データを 4件追加
INSERT rsTest1
VALUES ('User1', 1, 'AAA')
,('User1', 2, 'BBB')
,('User2', 3, 'CCC')
,('User2', 4, 'DDD')
```

1、2件目は、colUserName 列に User1 を格納して、User1 のみが参照できるようにし、 **3、4件目**には User2 を格納するようにします。

4. 次に、追加したデータを確認します。

```
-- データの確認
SELECT * FROM rsTest1
```

```
-- データの確認
  SELECT * FROM rsTest1
100 % - <
💷 結果 🛅 メッセージ
    colUserName colA colB
    User1 1
                  AAA
1
          2
                   BBB
2
    User1
             3
    User2
                  CCC
3
              4
4
    User2
                 DDD
```

5. 続いて、データベース ユーザー(User1、User2、Manager)に対して、テーブルに対する SELECT 権限を付与します。

```
-- データベース ユーザーに、テーブルに対する SELECT 権限を付与
GRANT SELECT ON rsTest1 TO User1
GRANT SELECT ON rsTest1 TO User2
GRANT SELECT ON rsTest1 TO Manager
```

```
→ 行レベル セキュリティのためのユーザー定義関数とセキュリティ ポリシーの作成
```

行レベル セキュリティは、**ユーザー定義関数**と**セキュリティ ポリシー(フィルター**)を作成する ことによって実現できます。

6. ユーザー定義関数は、次のように作成します。

```
-- 行レベル セキュリティを実現するためのユーザー定義関数の作成例

CREATE FUNCTION rsFunc1 (@UserName AS sysname)

RETURNS TABLE

WITH SCHEMABINDING

AS

RETURN

SELECT 1 AS result

WHERE USER_NAME() = @UserName

OR USER_NAME() = 'Manager'
```

ユーザー定義関数の名前は rsFunc1 (任意の名前を設定可能)、引数に @UserName を追加して、ユーザー名を受け取れるようにします。関数の中では、USER_NAME() 関数を利用して接続中のユーザー名を取得し、引数として受け取ったユーザー名と等しいかどうか、あるいは Manager かどうかを判断して、ユーザー名が等しいまたは Manager の場合に 1 を返すようにしています。

 ユーザー定義関数を作成した後は、FILTER PREDICATE を指定したセキュリティ ポリシー (フィルター)を作成します。これを行うには、次のように CREATE SECURITY POLICY ステートメントを記述します。

```
CREATE SECURITY POLICY rsFilter1
ADD FILTER PREDICATE dbo.rsFunc1(colUserName)
ON dbo.rsTest1
WITH (STATE = ON)
```

セキュリティ ポリシーの名前は rsFilter1 (任意の名前を設定可能)、ADD FILTER PREDICATE で「dbo.rsFunc1(colUserName) ON dbo.rsTest1」と指定することで、該 当テーブル (rsTest1) に対して、rsFunc1 関数 (前の手順で作成したユーザー定義関数) で フィルターをかけて、関数へ与える引数に colUserName 列を与えることができるようにな ります。末尾の WITH (STATE = ON) では、このセキュリティ ポリシーを有効化すること ができます。

以上で、行レベル セキュリティの設定が完了です。

8. 次に、通常のユーザー(何も権限が与えられていないユーザー)で、テーブルを SELECT してみます。

```
-- 何も権限が与えられていないユーザー
SELECT * FROM rsTest1
```

何も権限が与えられていないユーザー SELECT * FROM rsTest1
100 % - <
🎟 結果 🔓 メッセージ
colUserName colA colB

結果には、何も返ってきません。このように、テーブルに対して**フィルター**を設定している場合には、ユーザー定義関数で設定したユーザー以外は、テーブル データを参照できない形になります。

9. 次に、Manager ユーザーでアクセスしてみます (EXECUTE AS USER ステートメントで別のユーザーでの接続をシミュレートします)。

-- Manager をシミュレート EXECUTE AS USER = 'Manager' SELECT * FROM rsTest1 REVERT

Manager をシミュレート EXECUTE AS USER = ' <mark>Manager</mark> ' SELECT * FROM rsTest1 REVERT								
100 %	• <							
副結	果 🛅 メッセージ							
	colUserName	colA	colB					
1	User1	1	AAA					
2	User1	2	BBB					
3	User2	3	CCC					
4	User2	4	DDD					

Manager ユーザーは、すべてのデータを参照することができます。内部的には、フィルター で指定したユーザー定義関数内の「USER_NAME() = 'Manager'」という部分が利用されて いて、Manager ユーザーであれば、すべてのデータが返る形になります。

10. 次に、User1 ユーザーでアクセスしてみます。

```
-- User1 をシミュレート
EXECUTE AS USER = 'User1'
SELECT * FROM rsTest1
REVERT
```

User1 をシミュレート EXECUTE AS USER = ' <mark>User1</mark> ' SELECT * FROM rsTest1 REVERT						
100 %	• <					
🏥 結界	果 🛅 メッセージ					
	colUserName	colA	colB			
1	User1	1	AAA			
2	User1	2	BBB			

User1 ユーザーは、colUserName 列が User1 のデータしか参照できないことを確認できます。

内部的には、フィルターで指定したユーザー定義関数内の「USER_NAME() = @UserName」 という部分が利用されていて、USER_NAME() は User1(接続しているユーザーの名前) を返し、@UserName には、colUserName 列に格納されているデータ(User1 や User2) が与えられるので、データが User1 の場合には、その結果を返るという形です。

11. 次に、User2 ユーザーでアクセスしてみます。

```
-- User2 をシミュレート
EXECUTE AS USER = 'User2'
SELECT * FROM rsTest1
REVERT
```

	User2 をシミュレート EXECUTE AS USER = 'User2' SELECT * FROM rsTest1 REVERT							
10	100 % - <							
	Ⅲ 結果							
		colUserName	colA	colB				
1	1 User2		3	CCC				
2	2	User2	4	DDD				

User2 ユーザーは、colUserName 列が User2 のデータしか参照できないことを確認できます。

このように、行レベル セキュリティを利用すれば、行レベルのアクセス制御を実現すること ができ、ユーザーごとに、見せたい行を制限できるようになります。

↓ 行レベル セキュリティを無効化したい場合

行レベル セキュリティを無効化したい場合には、作成した**フィルター(セキュリティ ポリシー**) を無効化するだけで完了します。これも試してみましょう。

 フィルターを無効化するには、次のように ALTER SECURITY POLICY ステートメントで、 STATE を OFF に設定します。

```
    -- 行レベル セキュリティを無効化する(フィルターを無効化)
    ALTER SECURITY POLICY rsFilter1
    WITH (STATE = OFF)
```

無効化をした後は、**通常のユーザー**(何も権限が与えられていないユーザー)でも、データを 参照できるようになります。

	何も権限が与えられていないユーザー(無効か後は参照可能) SELECT * FROM rsTest1								
100	100 % - <								
	結	果 🛅 メッセージ							
		colUserName	colA	colB					
1		User1	1	AAA					
2		User1	2	BBB					
3		User2	3	CCC					
4		User2	4	DDD					

その他、行レベル セキュリティに関する最新情報は、オンライン ブックの次のトピックが参考になると思います。

Row-Level Security

https://msdn.microsoft.com/en-us/library/dn765131.aspx

••• Permissions or Securables Page	Row-Level Security
Password Policy	SQL Server 2016 Other Versions -
Strong Passwords	Updated: November 30, 2015
Row-Level Security	Applies To: Azure SQL Database, SQL Server 2016 Preview Row-Level Security enables customers to control access to rows in a database table based on the
 SQL Server Encryption Dynamic Data Masking SQL Server Certificates and Asymmetric Keys SQL Server Audit (Database Engine) 	characteristics of the user executing a query (e.g., group membership or execution context). Row-Level Security (RLS) simplifies the design and coding of security in your application. RLS enables you to implement restrictions on data row access. For example ensuring that workers can access only those data rows that are pertinent to their department, or restricting a customer's data access to only the data relevant to their company. The access restriction logic is located in the database tier rather than away from the data in another application tier. The database system applies the access restrictions every time that data access is attempted from any tier. This makes your security system more reliable and robust by reducing the surface area of your security system.
	Implement RLS by using the CREATE SECURITY POLICY Transact-SQL statement, and predicates created as inline table valued functions. Applies to: SQL Server (SQL Server 2016 Community Technology Preview 3 (CTP 3.0) through current version), SQL Database V12 (Get it). In this Topic Description Use Cases

3.4 Always Encrypted による列データの暗号化

Always Encrypted は、ネットワーク上を流れるデータも、データベース内に格納されるデータ も、すべて暗号化して格納できる機能です。内部的には、次のように**列データを暗号化**して格納す ることで、これを実現しています(アプリケーションからも透過的に利用できます)。

S	Always Encrypted 設定後のデータの確認 SELECT * FROM aeTest1 列データを暗号化				
100 %	* <				
🛄 結	¥ 🛅 y	ッセージ			
	colA	colB	colC		
1	1	0x01B44A66DBF0B586803561098905B0EFEEA	0x0112FD74802543520C46C5F9135E0B80DDDEA024AE4A000		
2	2	0x0137A2B5E14A4F1E8C760651254DA459349	0x016747C02567D8F26EA69C61F35B6980715E95EFF0E9822F		

設定には、ウィザードが用意されているので、簡単に設定することができます。

🗏 SQLQuery2.sql - WIN10.ae	eTestDB (WIN10¥matumoto (71)) -	Microsoft SQL Server Mana	igement Studio (管理者)	
ファイル(F) 編集(E) 表示(V)	プロジェクト(P) デバッグ(D) ツール	ィ(T) ウィンドウ(W) ヘルプ(I	-1) 	
i 🛅 • 🔃 • 📂 🖬 🥔 😫)新しいクエリ(N) 🌓 🐴 🐴 🏠	X 10 12 19 - (P	Always Encrypted	- U X
: ഈ 한값 aeTestDB	- │ ! 実行(X) ♪ デ/	(ッグ(D) 🔲 🗸 🗓 🖻	10月 説明	
オブジェクト エクスプローラー	↓ ↓ × SOLOue	rv2.sgl - W10¥matumoto	10.00	0 ALT
接続 - 🔐 🔡 = 🍸 🖬		.,,	列の選択	
WIN10 (SQL Server 13	3.0.900 - WIN10¥matumo		マスター キーの構成	
🖃 🚞 データベース			検証	Always Encrypted は、クレジットカード番号など、SQL Server データベースに格納されている重要な情報を保護するように設計されています。クライアントによるクライアント アプリケーション内のデータの暗号化
🗉 🧰 システム データベ	-7		概要	を許可しますが、SQL Server に暗号化キーを公開することは決してありません。
	ツノショット		結果	
 ● ● ここの ● ● 新し 	ルデータベース(N)			
🗆 🚞 新し	,いクエリ(Q)			
🗉 F-1	タベースをスクリプト化(S)			0
±	2(T) ►	デタッチ(D)		
÷ +••				
E 🗀 👘	·y=(0) ►	オノラインにする(1)		
	27P(A)	ハンフ1ンIL9 @(I)		
Pow	verShell の起動(H)	Stretch(S)		
日 🗀 レポ	[−ト(P) →	列の暗号化		
日 📮 名前	前の変更(M)	 圧縮(S)		
■ <mark>■</mark> DW 削隊	余(D)	1(m/ファップ(R)		□ 次回からこのページを表示しない。(D)
	新の情報に更新(F)	バラファラフ(B) 復元(P)		
		120/0(11)		< 戻る(P) 次へ(N) > キャンセル

Always Encrypted を利用するために、アプリケーション側で変更する必要があるのは接続文字 列のみで、そのほかのプログラムは修正する必要はありません(接続文字列に Column Encryption Setting=enabled を追加するのみ。ただし、.NET Framework 4.6 以上が必要)。

Console Application 1 - Microsoft Visual Studio (管理者) アイル(E) 編集(E) 表示(V) プロジェかト(E) ビルド(E) デパッグ(E) チーム(M) ツール(I) テスト(S) 分析(N) ウィンドウ(M) ヘルプ(E) O - O 記 - C III - C IIII - C IIIIIIIIIIIIIIII	 クイック起動 (Ctri+Q) クーロ× サインイン
Program cst 9 × ConsoleApplicationt ConsoleApplicationt Using System.Data; ConsoleApplication1 Using System.Data; ConsoleApplication1 (個の参照 class Program (個の参照 class Program (個の参照 class Program (個の参照 class Program (個の参照 static void Main(string[] args) ftring cnstr = "Server=localhost;Database=aeTestDB;Integrated Security=SSP1;" + "Column Encryption Setting=enabled"; using (SqlConnection cn = new SqlConnection(cnstr))	 > ソリューション エカスプローラー ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●
<pre> i using (SqlCommand cmd = new SqlCommand()) { cmd.Connection = cn; string sqlstr = "SELEOT * FROM aeTest1"; cmd.CommandText = sqlstr; try f cn.Open(); using (SqlDataReader dr = cmd.Execute while (dr.Read()) } f } CHUbersWmatu_takMDesktopWConsoleApplicationTW max max</pre>	ソリューション I ···· チーム エカスプロ··· カラス ピュー CConsoleApplication 1¥bin¥Debug¥ConsoleApplication 1.e □ X
Console.WriteLine(dr["col0"].ToString());	

→ Always Encrypted での基本動作(アプリ側が 暗号 ←→ 復号 を自動で行う)

<u>アプリケーション側(.NET Framework 4.6 の場合)</u> SQL Server Profilerでキャプチャしたアプリから送信された SQL Rever Profile K ConsoleApplication1 - Microsoft Visual Studio (管理者) ー ファイル(F) 編集(E) 表示(V) 再生(R) ツール(T) ウィンドウ(W) ヘルプ(H) 論 🎦 🍰 🚔 🎿 📿 🕨 🎟 🦉 📓 🌆 🗐 🗐 20 🗐 🧐 ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デパッグ(D) チーム(M) ツール(T) テスト(S) 分析(N) ウィ 😋 - 💿 📸 - 🖕 💾 🚰 🎔 - 🖓 - Debug - Any CPU - 🕨 開始 - 🏓 - 🏷 🌆 📜 😕 注意 無题 - 1 (WIN10) am.cs* 🕂 🗙 onsoleApplication1
 EventClass
 TextData

 SQL:BatchCompleted
 SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
 - [©] Main(string 1 👻 🔩 Console Application1 . Program exc MessaeQueueReceive @QueueName=N'OwROIIbackQueue', -- network protocol: LPC set quoted_identifier on set exec sp_describe_parameter_encryption N'SELECT * FROM a RPC:Completed // 暗号化列をパラメーターで指定する場合 using (SqlConnection cn = new SqlConnection(cnstr)) { Audit Login RPC:Completed 4-1 exec sp_executesql N'SELECT * FROM aeTest1 WHERE colB : RPC:Completed using (SqlCommand cmd = new SqlCommand()) Trace Stop cmd.Connection = cn; string sqlstr = "SELECT * FROM aeTestl WHERE colB = @p1"; cmd.CommandText = sqlstr; SqlParameter pl = cmd.Parameters.Add("@p1", SqlDbType.Int); pl.Value = "111"; PREPK \$17.501(Cott...74) exec_sp_executesql_N'SELECT * FROM_aeTest1_WHERE_colB = @p1',N'@p1 int',@p1=0x01B44A66DBF0B586803561098905B0EFEEAA05D52C7AC3BAB92C976A33D300A9EA66 暗号化された状態に変換されて 2 暗号化された列に対しても SQL Server に送信される cn.Open(); using (SqlDataReader dr = cmd.ExecuteReader()) { 通常どおり、平文で値を指定 SQL Server 2016 側 暗号化された列データ -- Always Encrypted 設定後のデータの確認 SELECT * FROM aeTest1 while (dr.Read()) Always Encrypted で 列データを暗号化 の取得 Console.WriteLine(dr["colC"].ToString()): 🖏 結果 📑 メッセージ 3
 colA
 colB
 colA
 <thcolA</th>
 colA
 colA
 <thc C:¥Users¥matu_tak¥Desktop¥ConsoleA A × ۵Α 暗号化された列データ 暗号化されたデータを も通常どおり取得可能 プリ側が自動的に 平文に変換してくれる

Always Encrypted での基本動作は、次のようになります。

アプリケーションの接続文字列に「Column Encryption Setting=enabled」を追加しておくこ とで、アプリケーションのプログラム内に記述された平文(上の例では colB 列に与える 111 と いう値)は、アプリケーションによって、自動的に暗号化されて(Ox01B44A66DB~ 形式に自 動変換されて)SQL Server に送信されます。SQL Server 側では、暗号化された状態でデータが 格納されている(Ox01B44A66DB~ という形で格納されている)ので、この該当データをアプ リケーションに返す、という流れです。

また、アプリケーション側では、colC 列のデータ(暗号化された 0x0112FD7480~ 形式のデ ータ)を取得していますが、このデータについても、アプリケーションが自動的に復号化を行って、 AAA というデータに戻しています。

このように、Always Encrypted を利用すれば、アプリケーション コードを変更することなく (変更するのは接続文字列だけで)、SQL Server 側の列データおよびネットワーク上を流れるデ ータを暗号化することができるようになります=セキュリティを大幅に向上させることができま す。

Let's Try

それでは、これを試してみましょう。

1. まずは、Always Encrypted を試すためのデータベースとテーブルを作成します。

-- データベースの作成 CREATE DATABASE aeTestDB go

```
-- テーブルの作成
USE aeTestDB
CREATE TABLE aeTest1
( colA int PRIMARY KEY
, colB int
, colC varchar(20))
-- データを 2件追加
INSERT INTO aeTest1 VALUES(1, 111, 'AAA')
INSERT INTO aeTest1 VALUES(2, 222, 'BBB')
-- データの確認
SELECT * FROM aeTest1
```

データの確認 SELECT * FROM aeTest1					
100 %	• <				
🛄 結果	¥ 🛅 :	メッセー ジ			
	colA	colB	colC		
1	1	111	AAA		
2	2	222	BBB		

データベース名は「aeTestDB」、この中に「aeTest1」という名前のテーブルを作成して、デ ータを 2 件追加しておきます。テーブルには、「colA」と「colB」、「colC」の 3 列を作成し ていますが、この後の手順で「colB」と「colC」列を暗号化していきます。

 次に、オブジェクト エクスプローラーで「aeTestDB」データベースを右クリックして、[タ スク] メニューの [列の暗号化] をクリックします。



2. [Always Encrypted] ウィザードが起動したら、最初のページでは [次へ] ボタンをクリックします。

🗾 Always Encrypted	- 🗆 X
🗾 説明	
説明	
列の選択	
マスターキーの構成	Alwavs Encrypted は、クレジットカード番号など、SOL Server データベースに格納されている重要な情
検証	報を保護するように設計されています。クライアントによるクライアント アプリケーション内のデータの暗号化 を許可しますが、SOL Server に暗号化キーを公開することは決してありません。
概要	
結果	
	□ 次回からこのページを表示しない。(D) 【

3. 次の [列の選択] ページでは、[CEK_Auto1 (新規)] が選択されていることを確認します。 これは、列を暗号化するための「列の暗号化キー」(COLUMN ENCRYPTION KEY)になる もので、ウィザードによって自動的に作成されます。

🔛 Always Encrypted		- 🗆 ×
説明		@ ヘルプ
列の選択	列名を検索	
マスター キーの構成	✓ チェックされたすべての列に1つのキーを適用:	1) CEK_Auto1 (新規)
		暗号化の種類 暗号化キサ
結果	名前 状態	間時号化の種類 暗号化キー
	⊡ dbo.aeTest1	
	ColB 🖉	決定論的
	colC 🥒	決定論的 - CEK_Auto1(新規)
	□ 影響を受けた列のみを表示	
		2 <戻る(P) 次へ(N) > キャンセル

「aeTest1」テーブルの列の一覧からは、「colB」と「colC」列をチェックして、これを暗号 化するようにします。[暗号化の種類]では、[決定論的]または[ランダム化]を任意に設定 します(暗号化をランダム化するかどうかを設定します。画面は[決定論的]を選択)。 **4.** 次の [マスター キーの構成] ページでは、列マスター キー(COLUMN MASTER KEY)を どこに作成するのかを設定します。

Always Encrypted		-		
マスターキーの構成	Ĩ			
説明				,
列の選択 マスター キーの様成	新しい列暗号化キーを生成するには、保護するために列マスターキ キーはデータベースの外部に格納されます。	ーを選択する必要があり	ます。列マスター	
検証	列マスター キーを選択:			
概要	自動生成列マスターキー		\sim	
結果	キー ストア プロバイダーを選択			
	 Windows 証明書ストア 			
	O Azure Key Vault 🕕			
2	マスターキー ソースを選択:			
	現在のユーザー		\sim	
	証明書ストアの ユーザー ストア			
	< 戻る	3 (P) 次へ(N) >	キャンセル	

ここでは、[**列マスター キーを選択**] で [**自動生成列マスター キー**]、[**キー ストア プロバイ ダーを選択**] で [**Windows 証明書ストア**]、[**マスター キー ソースを選択**] で [**現在のユー ザー**] が選択されていること(すべて既定値)を確認します。これで、ユーザーの証明書スト アに列マスター キーを作成することができます。

5. 次の [検証ページ] では、Always Encrypted を設定するためのスクリプトを生成するのか、 今すぐ実行するのかを設定します。

🗊 Always Encrypted		-		×
し 検証				
説明			0	ヘルプ
列の選択				_
マスターキーの構成	項方化/復方化が実行中であるため、テーノルできさ込み操作を実行 ▲ き込み操作を実施すると、データが失われる可能性があります。計画され	れているメンジ	りません。 テナンス期	宇 間
検証	中に、この暗号化/復号化の操作をスケジュールすることをお勧めします。			
概要		+ 7 + 7 1	. –	
結果	晴亏化が売了 対する INSE	9 るまでに RT や UP	a、テー DATE オ	フルに などの
	Select how you would like to proceed 更新系トラン ようにしる	ザクション ましょうる	ンを実行 という警	しない 告
	○ PowerShell スクリプトを生成して後で実行する			
	 ・ ・ ・			
	9			
	< 戻る(P) 次	(∩(N) >	キャン	セル

ここでは、[続行して今すぐ完了]を選択して、すぐに実行するようにします。

6. 最後の [概要] ページでは、設定内容を確認して、[完了] ボタンをクリックします。

III Always Encrypted		_		×
説明			0	ヘルプ
列の選択	このウイザードで選択した内容を確認します。			
マスター キーの構成	[完了]をクリックして、以下のように設定された操作を実行します。			
検 <u>証</u> 施果		₹ <u>7(F)</u>	+ <i>tyy</i>	他们

以上で、Always Encrypted のすべての設定(列マスター キーや、列の暗号化キーの作成、 colB と colC 列に対する暗号化の実行)が完了です。

7. すべての実行が成功すると(暗号化が完了すると)、次のように[結果]ページで[合格]と表示されるので、[閉じる]ボタンをクリックします。



→ 暗号化されていることの確認

次に、Always Encrypted によって、列データが暗号化されたことを確認してみましょう。

1. SELECT ステートメントを実行して、「aeTest1」テーブルの中身を参照してみます。

SE	SELECT * FROM aeTest1					
	データの確認 SELECT * FROM aeTest1 ことを確認					
100 %						
	colA	colB	colC			
1	1	0x01B44A66DBF0B586803561098905B0EFEEA	0x0112FD74802543520C46C5F9135E0B80DDDEA024AE4A000			
2	2	0x0137A2B5E14A4F1E8C760651254DA459349	0x016747C02567D8F26EA69C61F35B6980715E95EFF0E9822F			

colB と **colC** 列が暗号化されていることを確認できます。このように、Always Encrypted を設定すると、列データを暗号化することができます。

次に、ウィザードによって自動作成された列マスター キー (COLUMN MASTER KEY) と列の暗号化キー (COLUMN ENCRYPTION KEY) を確認してみましょう。次のようにオブジェクト エクスプローラーで [セキュリティ]の [常に暗号化されたキー] フォルダーを展開します。



「CMK_Auto1」という名前の**列マスター キー**、「CEK_Auto1」という名前の**列の暗号化キ** ーが作成されていることを確認できます。

これらのキーは、次のように右クリックして、スクリプトを生成することで、ウィザードによってどのようにキーが作成されたのかを確認することもできます。

🧏 SQLQuery5.sql - WIN10.aeTestD8 (WIN10¥matu_tak (146)) - Microsoft SQL Server Management Studio (管理者)				
ファイル(F) 編集(E) 表示(V) クエリ(Q) プロ	ロジェクト(P) デパッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)			
: 🛅 🕶 📨 > 💕 🛃 🗿 🔔 新しいクエリ(N	9 🕞 📸 🏭 🐇 🖾 😕 - (** - 💭 - 🖳 🖉 + 💋 User01			
aeTestDB 🔹	!実行(X) ▶ テハシク(D) = ✓ 罰 目 冒 罰 暢 過 麵 @ 国 国 冨 器 場 個 麵 @ = 2 幸 幸 幅 。			
オブジェクト エクスプローラー	🝷 🕂 🗴 SQLQuery5.sql - WI10¥matu_tak (146)) 🗙 SQLQuery2.sql - WI10¥matu_tak (145)) 03_Always EncrypteN10¥matu_tak (58))			
接続 - 野 製 = 🍸 🛃 🔏	USE [aeTestDB]			
🖃 🔂 WIN10 (SQL Server 13.0.700 - WIN104	¥matu_tak) 🔨 GO			
🖃 🚞 データベース				
😠 🚞 システム データベース	USE Laelestubj //XXXXXX Dbject: ColumnMasterKey [OMK Autol] Script Date: 2015/11/30 17:56:23 XXXXXX/			
🗉 🚞 データベース スナップショット	CREATE COLUMN MASTER KEY LOWA Autol]			
🖃 🔰 aeTestDB	WITH			
	KEY_STORE_PROVIDER_NAME = N'MSSOL_CERTIFICATE_STORE',			
■ ■ □ □ □	KEY_PAIH = N CurrentUser/my/D88945A80591CF6594EZA200CTE8A69788FZA856			
■ □ プログラミング				
🗉 🧰 Service Broker				
🖃 🧰 セキュリティ				
😠 🧰 ユーザー				
😥 🗀 🗆 💷				
🗄 🗀 スキーマ				
田 □ 非対称キー □ □				
日 📄 列の暗号化主				
CEK Auto				
	PowerShell の起動(H) 構造を定(A) ようアイル…			
😠 🧰 セキュリティ ポリシー	交換(R) 削除(U) 1 (1) クリップポード			
DWConfiguration	クリーンアップ(C) 削除あよび作成(R) ト 民 エージェント ジョブ			
DWDiagnostics	検索(5)			
🗉 🔰 DWQueue	νπ−F(P) → □ 25%3(0)			

また、テーブル「aeTest1」を、次のようにスクリプト化すれば、colB と colC 列に対する 暗号化の設定も確認することができます。



生成された CREATE TABLE ステートメントの colB と colC では、ENCRYPTED WITH があり、COLUMN_ENCRYPTION_KEY (列の暗号化キー) で CEK_Auto1 が指定され、 ENCRYPTION_TYPE (暗号化の種類) で Determistiic なら「決定論的」、Randomized なら「ランダム化」になります。

→ 接続パラメーターに Column Encryption Setting=Enabled を追加

Management Studio では、接続時のパラメーターに「**Column Encryption Setting=Enabled**」 を追加することで、Always Encrypted で暗号化されたデータを参照することができます。これも 試してみましょう。

1. まずは、ツールバーの [データベース エンジン クエリ] ボタンをクリックして、新しい接続 を追加します。

🍢 SQLQuery1.sql - WIN10.master (WIN10¥matu_tak (145)) - Microsoft SQL Server Management Studio (管理者)						
ファイル(F) 編集(E) 表示(V) クエリ(Q) <u>インニクト(</u> P) デバッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)						
: 🎦 • 🕮 • 🚰 🛃 🥔 🔔 新UN/2IU(N) 🚹 🏰 🌇 🍒 🔺 🛳 🖳 🔊 - 💛 - 💭 - 🖳 🖓 🗼 🔹						
: 野 辺 master ・ 『 データベース エンジン	ידע 🔍 🎜 🗐 🖉	" 🃅 🖷 🖓 🎆 🎱 🔳 🖆 🗐 🐺 👘 🖓 🕛				
オブジェクト エクスプローラー マ 甲 × SQLQuery1.sql - \	Wl10¥matu_tak (145)) ×					
接続 - 影影 = 🝸 🔊 📓	🚽 データベース エンジンへ	の接続	\times			
🖃 🐻 WIN10 (SQL Server 13.0.700 - WIN10¥mi						
	SOL Server					
	サーバーの種類(工):	データベース エンジン	\sim			
⊞ AlwaysOn 高可用性 □ □	サーバー名(<u>S</u>):	WIN10	~			
 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	12812(<u>A</u>):	Windows 認証	\sim			
📆 SQL Server エージェント (Agent XPs がき	ユーザー名(山):	WIN10¥matu_tak	\sim			
	パスワード(<u>P</u>):					
		─ パスワードを保存する(M)				
		2				
		接続(<u>C</u>) キャンセル ヘルプ オプション(0 >>			

[データベース エンジンへの接続] ダイアログが表示されたら、[オプション] ボタンをクリ ックします。

 オプション設定ダイアログでは、[追加の接続パラメーター] タブをクリックして、「Column Encryption Setting=Enabled」と入力し、[接続] ボタンをクリックします。

■見 データベース エンジンへの接続	×
SQL Server	
ログイン 接続プロパティ 追加の接続パラメーター	
追加の接続文字列パラメーターを入力します(クリア テキストで送信されます)(E):	
Column Encryption Setting=Enabled	
(注意:接続文字列以うメーターは他のパネルでグラフィカルに選択した内容よりも優先 されます)。	
3 接続(①) キャンセル ヘルブ オブション(②	<<

これで、暗号化されたデータを参照できるようになります。

3. 接続完了後、SELECT ステートメントを実行して、「aeTest1」テーブルを参照し、暗号化さ れたデータを参照できることを確認しておきましょう。

USE aeTestDB SELECT * FROM aeTest1



なお、[追加の接続パラメーター] で設定した「Column Encryption Setting=Enabled」 は、Management Studio を終了しても設定が残っている(次の接続時も Enabled で接続に なる) ので、元に戻したい場合には、[追加の接続パラメーター]を「空」にするようにします。

◆ アプリケーションからの接続(.NET Framework 4.6 の場合)

次に、暗号化されたデータをアプリケーションから参照してみましょう。ここでは、Visual Studio 2015 (C#) および .NET Framework 4.6 を例に説明します。

ADO.NET の場合

.NET Framework アプリケーション(ADO.NET)から Always Encrypted を利用するには、.NET Framework 4.6 以上が必要になります。.NET Framework 4.6 以上を利用している場合には、次のように接続文字列に「Column Encryption Setting=enabled」を追加します。



<u>そのほかのコードは修正する必要はなく</u>、これだけで暗号化されたデータを取得することができま す。

なお、接続文字列に「**Column Encryption Setting=enabled**」を<u>追加していない場合</u>には、実 行結果は次のようになります。



復号化は行われず、暗号化されたデータをそのまま取得する形になってしまいます。また、古いバ ージョンの .NET Framework をターゲット フレームワークに指定している場合には、次のよう にサポートされていないキーワードとして扱われます。



ADO.NET でのパラメーター(SqlParameter)場合

ADO.NET のパラメーター(SqlParameter)も、通常と同じように利用することができます。



パラメーターに代入する値を暗号化する必要はなく、通常どおりに平文で(そのまま)指定します。 これで、内部的には <u>.NET Framework が自動的に暗号化</u>を行ってくれます(接続文字列に **Column Encryption Setting=enabled**指定されている場合の動作)。この動作は、**SQL Server Profiler** ツールを利用して確認することができます。次の画面は、上のアプリケーションを実行し たときに、SQL Server Profiler ツールでキャプチャしたものです。

🗃 SQL Server Profiler		>
ファイル(F) 編集(E) 表示(V) 再生	E(R) ツール(T) ウィンドウ(W) ヘルプ(H)	
🖻 🎦 📸 🗟 💣 🕷 🖉 🕨	II II 🙀 🗐 🔟 📮 🕱 📾 🔗	
a 無题 - 1 (WIN10)		
EventClass	TextData	ApplicationName
SQL:BatchCompleted	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED	TransactionManager
RPC:Completed	exec MessageQueueReceive @QueueName=N'DwRollbackQueue',@Date	TransactionManager
Audit Login	network protocol: LPC set quoted_identifier on set arit	.Net SqlClient Data Provider
RPC:Completed	exec sp_describe_parameter_encryption N'SELECT * FROM aeTest	.Net SqlClient Data Provider
RPC:Completed	exec sp_executesql N'SELECT * FROM aeTest1 WHERE colB = @p1'	.Net SqlClient Data Provider
Trace Stop		パラメーターに代入した値が
<		自動的に暗号化されている
exec sp_executesql N'SE int',@p1=0x01B44A66DBFC	LECT * FROM aeTest1 WHERE colB = @p1',N'@p1 E586803561098905B0EFEEAA05D52C7AC3BAB92C976A33D300A9EA66152F84057(013380ACE6C44FCCB5FEFE5759B7D710

パラメーターに代入した値 (コードでは colB 列に対して 111 を代入) が、.NET Framework に よって自動的に暗号化されて、OxO1B44A66DB~ という値に変換されて、SQL Server に送信 されています。SQL Server 側では、暗号化された状態でデータが格納されている (OxO1B44A66DB~ という形式で格納されている)ので、この該当データをアプリケーション に返すことができています。

このように、Always Encrypted では、アプリケーション側で(内部的に)暗号化と複合化を行っ てくれることで、アプリケーション コードを変更することなく、列データを暗号化できるようになっています。

ADO.NET での INSERT の場合

INSERT ステートメントなどの更新系のステートメントについても、通常の ADO.NET アプリケ ーションと同様に利用することができます。



パラメーターに代入した値は、次のように自動的に暗号化されて SQL Server に送信されます。

📄 SQL Server Profiler ファイル(E) 編集(E) 表示(V) 再生 記 🞦 🚰 📾 ピ 品 📿 🕨	8 ツールロ ウイドマ Mu ヘルプ田 II = 福 国 III - 1 22 24 24 9	- C	ı ×
🍰 無題 - 1 (WIN10)			
EventClass	TextData	ApplicationName	NT 🔨
Audit Losin	network protocol: TCP/IP set quoted_identifier on	.Net SqlClient Data Provider	
RPC:Completed	exec sp_describe_parameter_encryption N'INSERT INTO ae	.Net SqlClient Data Provider	
RPC:Completed	exec sp_executesq N'INSERT INTO aeTest1 VALUES(@p1, @	.Net SqlClient Data Provider	
Audit Logout		Transaction II II	いた 店が
< RPC.Completed	even sh reset connection	Transaction 自動的に暗号化さ	れている
exec sp_executesq N'IN varchar(20)',@p1=3,@p2=0	EERT INTO acTest1 VALUES(®pl, ®p2, ®p3), N'®pl int,®p2 int, ≫01FA85F4CC25391902DA170217E97991D80C4628F4518F220C98B960A1	63 Å939A740F77583179CFE44A2F9098BDB	622DD

Entity Framework を利用する場合

Entity Framework を利用する場合には、次のように [接続のプロパティ] (.NET Framework SQL Server 用データ プロバイダー) で、 [詳細設定] ボタンをクリックして、「Column Encryption Setting」を「enabled」に設定します。

■ 1ノストール)))の	並べ替え: 既定	• # E		インストール済み テンプレート の検	索 (Ctrl+E 🔑 -	
▲ Visual C# アイテム	ADO NET Entity (Data Model	Visual C# アイテム	種類: Visual C# アイテム		
Web Windows Forms		Entity D	ta Model D/ff-k	ADONETE CODI MILLI	<i>+ //</i> ++++	
WPF	EF 5.x DbContext	ジェネレーター				
データ	EF 6.x DbContext	ジェネレーター	モデルのコンテンツの選択	र		
全般 SQL Server	LINQ to SQL 257	(
Workflow	- <u> XML </u> XML Z‡−マ	モデル	含めるコンテンツ(W)			
▶ オンライン			品品			
		データ・ ロ	Cースから EF 空の EF Designer signer デル	r モ空の Code First モデ データ ル Co	パースから de First	
	₹ XSLT ファイル					
	サービス ベースのデー	タバース 既存の	データベースに基づいて EF Desic フォブジェクトを選択できます	gner でモデルを作成します。 データ アプリケーションが対話す スクラフィナ	バース接続、モデル(・ モデルから生成され	の設定、およびモデルに含めるデ
Entity Data Model ウイ	ザード		×	//////////////////////////////////////		100.3.
	接続の選択					
アプリケーションがデー	-タベースへの接続に使用すスデータ接	持(W)				
		~	新しい接続(<u>C</u>)			
この接続文字列には	データベース接続に必要な重要情報(列: パスワード) が含まれています。 接続	文字列内に重要情報を			
 保存するとセキュリティ 名前(N): いいえ、重要 	「上の問題が発生する可能性がありま 県情報を接続文字列に含めません。この	す。 接続文字列にこの里要情報を含け D情報はアプリケーション コードで設定し)ますか? ます。(<u>E</u>)			
○ はい、重要情	青報を接続文字列に含めます。(!)		データソースの選択			? ×
接続文字列:			データ ソース(<u>S</u>):			-
			Microsoft SQL Serv Microsoft SQL Serv	rer rer データベース ファイル	そ明 NET Framework S	QL Server 用データ プロ
			<その他>		バイダーを使用して № 2005 以降または Mi	/icrosoft SQL Server icrosoft SQL Azure (C
				3	接続するには、これを	を選択します。
☑ 接続設定に名前る	を付けて App.Config に保存(<u>S</u>):					
			<i>₹_8</i> 10 (\$ 48_0).			
			.NET Framework SC)L Server 用データプロ/ ~		
	< 前へ(<u>P</u>)	次へ(№) > 完了(F	.NET Framework SC	QL Server 用データプロノ 〜 る(<u>U</u>)	続行	キャンセル
	<前へ(P)	次へ(№) > 完了([NET Framework SC ☑ 常にこれを選択する	QL Server 用データ プロ/ > 3(<u>U</u>)	続行	キャンセル
接続のプロパティ	< 前へ(P))	NET Framework SC	QL Server 用データプロ/ ~	続行	÷ キャンセル :::
接続のプロパティ 選択されたデータ ソースに接続するため	< 約へ(P)	 次へN> 売了(? × をグリックし 	NET Framework SC □ 常にこれを選択する	AL Server 用データブロ/ > ういり mn Encryption Sett	- 続行 ting」	キャンセル
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースかプロパイダーまた	< 前へ(P))の情報を入力するか、または [変更] はその両方を選択します。	 次へ(N) > 見了() えて() マーン ア × をクリックし 詳細プロ/ 	NET Framework SC ア常にこれを選択する 「Colum を	21. Server 用データブロ/ ~ 5回 mn Encryption Sett :「enabled」に設定	続行 ting」	キャンセル
接続のプロパティ 選択されたデータソースに接続するため て、別のデータ ソースかプロパイダーまた データ ソース(5):	< 前へ(P の)情報を入力するか、または [変更] はその両方を選択します。	 次へ(M) > 完了() ? × をクリックし 詳細プロ/ 	NET Framework SC マ常にこれを選択する 「Colur を ティ	2L Server 用データ プロ/ > 5(リ) nn Encryption Sett :「enabled」に設定	続行 ting」 ? ×	: २ ७७७८॥ .:
接続のプロパティ 重訳されたデータソースに接続するため て、別のデータソースかプロパイダーまた データ ソース(S): [Microsoft SQL Server (SqlClient) サーパータ(F):	< 前へ(P の) () () () () () () () () () () () () ()			21 Server 用データブロ/ > 5回 mn Encryption Sett :「enabled」に設定	(続行 ting」 ? ×	: キャンセル .:
接続のプロパティ スに接続するたが て、別のデータ ソースかプロパイダーまた データ ソース(S): Microsoft SQL Server (SqlClient) サーバー名(E): 	< 前へ(P の) 情報を入力するか、または (変更) はその両方を選択します。 3 、		NET Framework SC アキにこれを選択する 「Colum を ティ リになけの Name	2L Server 用データブロ/ > 500 nn Encryption Set : 「enabled」に設定 .Net SqlClient Data Prov	雙行 ting」 ? × ider	: キャンセル .::
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースかプロパイダーまた データ ソース(S): Microsoft SQL Server (SqlClient) サーパーキ(E): localhost サーパートログオンする	< 約へ(P の))情報を入力するか、または(変更) はその両方を選択します。	 次へ(M) > 完了(f) ? × をクリックし 該幅プロ/ ご、2、 (2) (3) (4) (4)	NET Framework SC アキにこれを選択する アキスト Jication Name rktation ID コリティ	2L Server 用データブロ/ ~ 5(少) nn Encryption Set: 「enabled」 (ご設定 .Net SqlClient Data Prov	続行 ting」 ? ×	キャンセル :
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースがプロパイダーまた データ ソース(S): Microsoft SQL Server (SqlClient) サーパー名(E): localhost サーパーにログオンする ④ Windows 認証を使用する(W)	< 前へ(P の)情報を入力するか、または (変更) はその両方を選択します。 夏新の情報()	 次へ(M) > 完了() えて() そうリックし (注意) (注意)<	NET Framework SC アキにこれを選択する アイ テイ ライスト olication Name rkstation ID 2リティ hentication yum Encryotion Setting	2L Server 用データプロ/ ~ 5(リ) S(リ) .Net SqlClient Data Prov NotSpecified Enabled	続行 ting」 ? × ider	: キャンセル :
接続のプロパティ 選択されたデータソースに接続するため て、別のデータソースがプロパイターまた データソース(S): Microsoft SQL Server (Sql/Client) サーパーにログスソする ・ Windows 読証を使用する(<u>W</u> 〇 SQL Server 読証を使用する(<u>W</u>	< 前へ(P の) 使報を入力するが、または [変更] はその両方を選択します。	 次へ(M) > 完丁() ? × をクリックし (注紙丁ロ/) (注紙丁ロ/)	NET Framework SC アキにこれを選択する アイ テキスト Nication Name rkstation ID コリティ hentication zum Encryption Setting rypt	2L Server 用データ プロ/ ~ 5(リ) 5(リ) .Net SqlClient Data Prov NotSpecified Enabled False	続行 ting」 ? × ider	: २ ७७७॥ .:
接続のプロパティ 選択されたデータソースに接続するため て、別のデータソースかプロパイダーまた データソース(S): Microsoft SQL Server (SqlClient) サーパーを(E): localhost サーパーにログオンする ● Windows 認証を使用する(<u>W</u>) O SQL Server 認証を使用する(<u>C</u> ユーザー名(<u>U</u>):	< 前へ(P の) 情報を入力するか、または [変更] はその両方を選択します。	 次へ(M) > 完丁() ? × をクリックし ご便(Ω)… ご買新(B) ご買新(B) ご ご<	NET Framework SC アキにこれを選択する アイ テキスト Silcation Name rkstation ID ユリテイ hentication aum Encryption Setting rypt grated Security sword	2L Server 用データ プロ/ > 5(リ) 5(リ) .Net SqlClient Data Prov NotSpecified Enabled Talse True	続行 ting」 ? × ider	₹₩У₽₩ :
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースかプロパイダーまた データ ソース(5): Microsoft SQL Server (SqlClient) サーパー名(E): localhost サーパーにログオンする ● Windows 認証を使用する(<u>M</u> ○ SQL Server 認証を使用する(<u>M</u> 」ーザー名(<u>U</u>): 」(スワード(<u>P</u>): 」(オフート(<u>P</u>):	(前へ(P) かの情報を入力するか、または(変更) はその両方を選択します。		INET Framework SC アキにこれを選択する アティ ティ ライ ライスト slication Name rkstation ID ユリティ hentication umn Encryption Setting rypt grated Security sword sist Security Info tisteverCertificate	2L Server 用データブロ/ > 5(少) mn Encryption Set : 「enabled」(こ設定 .Net SqlClient Data Prov NotSpecified Enabled Halse True False False	続行 ting」 ? ×	: +r>tu ;;
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースかプロパイターまた データ ソース(5): Microsoft SQL Server (SqlClient) サーパー名(5): localhost サーパーにログオンする ④ Windows 認証を使用する(20 0 SQL Server 認証を使用する(20 1/スワード(9): 」パスワード(9): 」パスワード	(前へ(P))の)情報を入力するが、または(変更)はその両方を選択します。 、 (最新の情報に 、 (最新の情報に) 2) を保存する(S)		NET Framework SC アキにこれを選択する アイ マネスト Piloation Name rktattion ID 2リティ Annencyption Setting rypt grated Security sword sit Security Info atServerCertificate rID	2L Server 用データブロ/ ~ S(少) S(少) Inn Encryption Sett Fenabled」(こ設定 .Net SqlClient Data Prov NotSpecified False False False	続行 ting」 ? ×	: キャンセル :
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースかプロパイダーまた データ ソース(S): Microsoft SQL Server (Sql/Client) サーパー名(E): localhost サーパーにログオンする ● Windows 読証を使用する(M) O SQL Server 認証を使用する(M) 」パスワード(D): 」パスワード(D): データパースへの接続			NET Framework SC アイ 「Colurn を デイ テオスト Nication Name rkstation ID ユリテイ hentication um Encryption Setting rypt um Encryption Setting rypt sits Security Info tist Security Info tist Security Info tist Security Info	2L Server 用データブロ/ > S(リ) S(リ) Ann Encryption Set: 「enabled」(こ設定 .Net SqlClient Data Prov NetSpecified Enabled False False False	eting」	: २ ७७७॥ .:
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースかプロパイダーまた データ ソース(5): Microsoft SQL Server (Sql/Client) サーパー名(E): localhost サーパーにログオンする ● Windows 認証を使用する(Q) 」 SQL Server 認証を使用する(Q) 」 ムザー名(L): 」 パスワード(P): 」 パスワード(P): 「パンワード(P): 「パースもの選択または入力」 「まままた(P): 「パースもの選択または入力」	(前へ(P) (の)情報を入力するか、または(変更) はその両方を選択します。 (変更) はその両方を選択します。 (変更) (支新の情報に) (立) を保存する(S) (立) ((立) ((立) ((((((((((次へ(M) > 実了(i) ? × をクリックし 夏町(D) 詳細プロ/ 夏町(D) ジョンション 夏町(D) ジョンション 夏町(D) ジョンション 夏町(D) ジョンション マンション マンション 夏町(D) ジョンション マンション マンション マンション </td <td>NET Framework SC アキート 「Colur ティ ティ ティ ト Dication Name rkstation ID コリティ hentication ypt grated Security grated Security sword sist Security Info tisserverCertificate r ID ス Source Source</td> <td>2L Server 用データ プロ/ > S(リ) S(リ) S(リ) Inn Encryption Seti : 「enabled」 (こ設定 .Net SqlClient Data Prov NotSpecified Enabled False False False Icalhost</td> <td>続行 ting」 ? ×</td> <td>: २७७७॥ .:</td>	NET Framework SC アキート 「Colur ティ ティ ティ ト Dication Name rkstation ID コリティ hentication ypt grated Security grated Security sword sist Security Info tisserverCertificate r ID ス Source Source	2L Server 用データ プロ/ > S(リ) S(リ) S(リ) Inn Encryption Seti : 「enabled」 (こ設定 .Net SqlClient Data Prov NotSpecified Enabled False False False Icalhost	続行 ting」 ? ×	: २ ७७७॥ .:
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースかプロパイダーまた データ ソース(5): Microsoft SQL Server (SqlClient) サーパーにログオンする ● Windows 認証を使用する(G ユーザー名(1): パスワード(P): パスワード(P): データベースへの接続 ● データベーススの違訳または入す aeTestDB 〇 データベースファイルのアタッチ(H)	(前へ(P) (の)情報を入力するか、または(変更) (はその)向力を選択します。 (変更) (よその)向力を選択します。 (変更) (なの)の方を選択します。 (変更) (なの)の方を選択します。 (なの)の方を認知していたます。 (なの)のうれていたます。 (なの)のうれていたます。 (なの)のうれていたます。 (なの)のうれていたます。 (なの)のうれていたます。 (なの)のうれていたます。 (なの)のののののののののののののののののののののののののののののののの		NET Framework SC	2L Server 用データ プロ/ > S(リ) S(リ) S(リ) Inn Encryption Seti : 「enabled」 (こ設定 .Net SqlClient Data Prov NotSpecified Enabled False False False False Icaelhost	続行 ting」 ? ×	: २ १७७८॥ .:
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースかプロパイダーまた データ ソース(S): Microsoft SQL Server (SqlClient) サーパー名(E): localhost サーパー名(E): ローザー名(D): パスワード(P): パスワード(P): データベースへの接続 ・ データベースへの接続 ・ データベースへの行りの 、 アータベースへの行りの 、 、 、 、 、 、 、 、 、 、 、 、 、	(前へ(P) かの情報を入力するか、または(変更) はその両方を選択します。 3 、	ア × 東丁() マ × 東丁() 夏丁()ックし 詳細ブロ/ 夏丁() ジー 夏丁() ジー 夏丁() ジー マ ジー マー () ワー () ワー () ワー () ワー () ワー マ ワー () ワー () ワー () ワー () ワー マ ワー () ワー<	NET Framework SC ▼ #ICINE 2007 77 77 77 77 77 77 77 77 77	2L Server 用データ プロ/ ~ 5(リ) S(リ) S(リ) Inn Encryption Sett : 「enabled」(こ設定 .Net SqlClient Data Prov NotSpecified Enabled True False False localhost aeTestDB	表行 ting」 ? × ider	: २ ७७७॥ .:
 接続のプロパテイ 選択されたデータ ソースに接続するたが て、別のデータ ソースかプロパイダーまた データ ソース(な): Microsoft SQL Server (SqlClient) サーパー名(L): localhost サーパートログオンする ④ Windows 認証を使用する(Q ユーザー名(L): ゴスワード(P): ブスワード データベースへの接続 ● データベースへの接続 ● データベースへの接続 ● データベースへの接続 ● データベースへの接続 ● データベースフィグルのアタッチ(H 論理名(L): 	(前へ(P) かの情報を入力するが、または(変更) はその両方を選択します。 3 、 最新の情報に) 2) を保存する(5) た(D): ():		NET Framework SC マードにてれる選択す。 マードにてれる選択す。 マードにてれる選択す。 マードになる マードにのる マードにのでの マードにのでの マードにのでの マードにのでの マードにのでの マードにのでの マードにのでの マードにのでの マードにのでの マードにのでの マードの マードの マードの マードの マードの マードの マードの マートのの マートの マートの マートのの マートのの	21. Server 用データ プロ/ ~ 5(リ) 5(リ) Ten Encryption Sett 5(ロ) Inn Encryption Sett 5(ロ)	続行 ting」 ? × ider	÷ +τντυμ .:
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースかプロパイターまた データ ソース(気): Microsoft SQL Server (SqlClient) サーパー名(丘): localhost サーパーにログオンする ④ Windows 認証を使用する(位) 〇 SQL Server 認証を使用する(位) 「パスワード(D): 「パロード(D):			INET Framework SC INET Framework SC マネにこれを選択す。 「Colurn を ティ ティ ライ ライ ライ ライ ライ ライ ライ ライ ライ ライ	2L Server 用データブロ/ > S(リ) S(リ) S(リ) Antice Server 用データブロ/ > S(リ) Antice Server 用データブロ/ > S(リ) Antice Server Antice Server Not Specified False Fa	eting」	: τ εντελμ
接続のプロパティ 選択されたデータ ソースに接続するため て、別のデータ ソースかプロパイダーまた データ ソース(5): Microsoft SQL Server (Sql/Client) サーパー名(E): localhost サーパーにログオンする ● Windows 認証を使用する(M) SQL Server 認証を使用する(M) SQL Server 認証を使用する(M) CAワード(D): パスワード(D): パスワード(D): アータベースへの接続 ● データベースへの接続 ● データベースへの接続 ● データベースへの接続 ● データベースへの接続		ア X をクリックし 詳細プロ) 夏夏(〇)… 詳細プロ) 夏夏(〇)… 第二〇 夏新(白) 第二〇 マ 732 夏新(白) 第二〇 マ 732 夏新(白) 第二〇 マ 732 夏新(白) 第二〇 マ 732 夏新(白) 第二〇 マ 742 「四 742 〇 742 〇 743 〇 743 〇 744 〇	NET Framework SC アイ 「Colur テオスト Dication Name rkstation ID 10万イ hentication mpt grated Security more the security sword sist Security Info tisterverCertificate r ID ス achDbFilename text Connection a Source source D3チャイCのコマンドの現定の列ば urce=localhost;Initial Cataloc	2L Server 用データブロ/ > S(リ) S(リ) S(リ) Anne Encryption Sett 「enabled」(こ設定 Anti-Setting False False False False False Iocalhost aeTestDB 副号化設定です。 Sg=aeTestDB;Integrated Security	続行 ting」 ? × iider	: + τν/εμ .:

◆ SQL Server とは別のマシンからの接続(証明書のエクスポート/インポート)

Always Encrypted では、(内部的には)証明書を利用して暗号化を行っているので、SQL Server とは別のマシンのアプリケーションから暗号化データにアクセスするには、証明書を複製しておく 必要があります。これを行っていない場合には、別のマシンからアクセスした場合に、次のように 「**暗号化解除できませんでした**」エラーが通達されます。



これを回避するには、証明書を複製しておくようにします。証明書は、Always Encrypted をウィ ザードの既定値で設定した場合には、「Windows 証明書ストア」の「ユーザー ストア」に作成さ れています。これを確認するには、[スタート]メニューの[ファイル名を指定して実行]から「mmc」 と入力して、管理コンソールを起動し、[ファイル]メニューの[スナップインの追加と削除]で[証 明書]を追加します。

🧧 ファイル名を指定して実行		×			
ま行するブログジム名、または ネットリソース名を入力してく 1 名前(0): mmc ◆ このタスクは管理者特	は聞くフォルダーやドキュメント名、イン? ださい。 権で作成されます。	9- V			
2 ок	キャンセル 参照(B).				
🔚 אין אן-עעב] - און אן-אינעב					
🚰 ファイル(F) 操作(A) 表示(V) お気(こ入り(O) ウィンドウ(W) ヘルプ(I	H)			
◆ 新規作成(N)	スナップインの追加と削除				×
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□	コンピューターで利用できるスナップ どの拡張を有効にするかを構成で	インからこのコンソールに使用す さます。	するスナップインを選	択したり、 選択したスナップインを構成し	たりできます。拡張可能なスナップインでは、
名前を付けて保存(A)	利用できるスナップイン(S):			選択されたスナップイン(E):	
3 スナップインの追加と削除(M)	スナップイン	ベンダー ^	[■ コンソール ルート	拡張の編集(X)
オプション(P)	10 セキュリティ テンプレート	Microsoft Corp		🗊 証明書 - 現在のユーザー	
最近使ったファイル	💣 セキュリティが強化された	Microsoft Corp			削除(<u>R</u>)
** = 100	1 セキュリティの構成と分析	Microsoft Corp			
(X) [≎#	(一)タスク スケジューラ	Microsoft Corp			上へ移動(<u>U</u>)
	日 デバイスマネージャー	Microsoft Corp			下へ移動(<u>D</u>)
	🔕 パフォーマンス モニター	Microsoft Corp	追加(<u>A</u>) >		
	□□ フォルダー	Microsoft Corp			
	」 ポリシーの結果セット	Microsoft Corp			
	■ ローカル ユーリーとクルーク 1 印刷の管理	Microsoft Corp			
	えん 共有フォルダー	Microsoft Corp			
	20 承認マネージャー	Microsoft Corp			
	■ 証明書	Microsoft Corp 🗸			評細設定(⊻)
	説明:				
	証明書スナップインを使うとユーサ	「ー、サービス、またはコンピュータ	ターの証明書ストア	の内容を閲覧できます。	
					OK キャンセル

[証明書] スナップインを追加したら、[証明書 - 現在のユーザー] の [個人] → [証明書] フォ ルダーを展開します。



ここに「Always Encrypted Auto Certificate1」という名前の証明書が作成されていることを 確認できます。この証明書をエクスポートするには、次のように証明書を右クリックして、[すべて のタスク] メニューから [エクスポート] をクリックします。

🚰 コンソール1 - [コンソール ルート¥証明書 - 現	見在のユ−ザ−¥個人¥証	明書]					
藩 ファイル(F) 操作(A) 表示(V) お気に	こ入り(O) ウィンドウ(W) ヘルプ(H)					
🗢 🔿 📶 🧹 🗞 🗮 🖻							
	発行先	^	発行者		有効期限	目的	
◇ 🗊 証明書 - 現在のユーサー ◇ 🧰 個人	🙀 Always Encrypte	d Auto Certificate1	Always Encrypted	Auto Certificate1	2016/11/30	IP セキュリティ IKE 中	
□ 証明書		開<(O)					
> 🤗 信頼されたルート証明機関		すべてのタスク(K)	> 開く(O)				
>		切り取り(T)	新しいキーで	証明書を要求(Q)			
> 🧰 Active Directory ユーザー オブジェ		コピー(C)	新しいキーで	証明書を書き換え	<u>(N)</u>		
> 信頼された発行元 「 信頼されていない証明書		削除(D)	詳細設定指	操作(A)	>		
> 🔛 サードパーティルート証明機関		プロパティ(R)	エクスポート	(E)			
> 📫 信頼されたユーザー		ヘルプ(H)					
				×			
← 🖉 証明書のエクスポート ウィザード							×
		4	☞ 証明典のエクフポート	tvtf=k			
			© 10010101010	212 1			
証明書のエクスポート	ウィザードの開始						
			秘密キーのエクスポート				
			秘密キーを証明書	と一緒にエクスポー	トすることができます。	,	
このワイリートでは、証明書、司します。	証明音信頼リストおよい;	证明音大劝!					
			秘密キーはパスワー	ドで保護されていま	す。秘密キーを証明	書と一緒にエクスポートする場	合は、パスワードを
証明機関によって発行された	証明書は、ユーザー ID を	確認しデー	入力する必要があ	ります。			
上の領域です。	(9 8/2000) B #K 2 8/0 C	010K9 6 RE 73	証明書と一緒に移	8密キーをエクスポー	トしますか?		
			(まし、秘)	ミキーをエクスポートし	.#J()		をエクフポート
続行するには、「次へ」をクリッ	クレズください。		○いいえる	密キーをエクスポート	しません(の)	します をチ	エックする
						次へ(<u>N)</u> キャンセル

エクスポート時は、「**はい、秘密キーをエクスポートします**」をチェックして、秘密キーも含めてエ クスポートするようにします(そのほかの設定は既定値、パスワードの部分は任意のパスワードを 設定するようにします)。

エクスポートした証明書(既定では **.pfx** 形式のファイル)は、別マシンにコピーして、別マシン 上で同じく「**証明書**」スナップインを利用して、次のようにインポートすることができます。



後は、エクスポート時に設定したパスワードを入力すれば、証明書のインポートが完了です。イン ポートが完了した後は、暗号化データを処理できるようになります。

このように、ネットワークをまたがったとしても、**同じ証明書**(証明書の複製)を持っていれば、 暗号化/復号化を行うことができるので、Always Encrypted では、ネットワーク上を流れるデー タについても、暗号化されることになります(データベース内のデータは、常に暗号化されていて、 復号化あるいは暗号化はアプリケーション側で実行することになるので、ネットワーク上を流れる データは暗号化されたデータのみになります)。

→ インポート ウィザードでデータのインポート(既存のテーブルからのデータ移行 etc)

既存のテーブルやテキスト ファイルなどから、Always Encrypted を設定したテーブルにデータ をインポートしたい場合には、Integration Services の「**SQL Server インポートおよびエクス** ポート ウィザード」を利用することができます。これは、次のようにデータベースを右クリックし て、[**タスク**] メニューの [データのインポート] から起動できます。
SQL Server 2016 CTP 3.2 自習書 No.1 新機能ダイジェスト

このウィザードでは、[変換先の選択]ページで、次のように[変換先]で[.NET Framework Data Provider for SqlServer] を選択することがポイントです。

🔛 SQL Server インポートおよびエク	-		×		
変換先の選択 データのコピー先を指定します。 				Cherry Cherry	
変換先(<u>D</u>):	🔇 .Net Framewo	rk Data Provide	r for SqlServer		•
₽ 2 ↓					
 コンテキスト Application Name Workstation ID 		Net SqlClient Da	ata Provider		-
 セキュリティ Authentication Column Encryption Setting 		NotSpecified Enabled		-	
Encrypt Integrated Security Password		rue			
Persist Security Info TrustServerCertificate User ID	1	False False			
ソース AttachDbFilename Context Connection		Falce			
Data Source		ocalhost			
Initial Catalog MultiSubnetFailover TransparentNetworkIPReso	lution	aeTestDB -alse Frue			-
Column Encryption Setting 接続上のすべてのコマンドの既定	の列暗号化設定で	ਰ			
ヘルプ(円)	く戻る(<u></u>	》 次へ(N)	> 完了(E) >>	キャンセ	л <i>"</i>

これで、詳細プロパティが表示されるようになるので、[**Data Source**] に SQL Server の名前、 [**Initial Catalog**] に取り込み先のデータベース名、認証方法は[**Integrated Security**] (Windows 認証での接続の場合は True に設定)または[User ID]と[Password]を設定(SQL Server 認証での接続の場合)して、「Column Encryption Setting」を「Enabled」に設定します。これで、Always Encrypted が設定されたテーブルでも、データをインポートすることができます。

このように、Always Encrypted を利用すれば、ネットワーク上を流れるデータも、データベー ス内に格納されるデータも、すべて暗号化して格納できるようになります。データの移行も、イン ポート/エクスポート ウィザードを利用して簡単に行うことができます。

その他、Always Encrypted に関する最新情報は、オンライン ブックの次のトピックが参考になると思います。

Always Encrypted (Database Engine)

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

Choose an Encryption Algorithm

- Transparent Data Encryption (TDE)
- SQL Server and Database Encryption Keys
- Always Encrypted

...

Always Encrypted (client development)

Always Encrypted Wizard

Migrate Sensitive Data Protected by Always Encrypted

Column Master Key Rotation

Always Encrypted (Database Engine)

SQL Server 2016

Updated: November 25, 2015

Applies To: Azure SQL Database, SQL Server 2016 Preview

Always Encrypted is a feature designed to protect sensitive data, such as credit card numbers or national identification numbers (e.g. U.S. social security numbers), stored in Azure SQL Database or SQL Server databases. Always Encrypted allows clients to encrypt sensitive data inside client applications and never reveal the encryption keys to the Database Engine (SQL Database or SQL Server). As a result, Always Encrypted provides a separation between those who own the data (and can view it) and those who manage the data (but should have no access). By ensuring on-premises database administrators, cloud database operators, or other high-privileged, but unauthorized users, cannot access the encrypted data, Always Encrypted enables customers to confidently store sensitive data outside of their direct control. This allows organizations to encrypt data at rest and in use for storage in Azure, to enable delegation of on-premises database administration to third parties, or to reduce security clearance requirements for their own DBA staff.

Always Encrypted makes encryption transparent to applications. An Always Encrypted-enabled driver installed on the client computer achieves this by automatically encrypting and decrypting sensitive data in the client application. The driver encrypts the data in sensitive columns before passing the data to the Database Engine, and automatically rewrites queries so that the semantics to the application are preserved. Similarly, the driver transparently decrypts data, stored in encrypted database columns, contained in query results.

Always Encrypted is available in SQL Server 2016 Community Technology Preview 3 (CTP 3.0) and in preview in SQL Database. For Channel 9 presentations that includes Always Encrypted, see SQL Server 2016 Always Encrypted and Overview and Roadmap for Microsoft SQL Server Security.

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

TDE (Transparent Data Encryption: 透過的なデータ暗号化) は、データベースの実体となるファイル (.mdf/.ldf) およびバックアップ ファイルを暗号化することができる機能で、SQL Server 2008 から提供されました。SQL Server 2016 では、TDE のパフォーマンスを向上させるために、 Intel の AES-NI (AES-New Instructions: AES 暗号化の処理を高速化するための命令セット) に対応しました。AES-NI は、**ハードウェア アクセラレーター**で、これを利用することで、TDE に おける CPU のオーバーヘッドを軽減することができます。

→ インメモリ OLTP への対応

SQL Server 2016 からは、TDE (透過的なデータ暗号化) がインメモリ OLTP 機能に対応するようになりました。SQL Server 2014 を利用している場合には、データベース内に、インメモリ OLTP を利用したテーブル (メモリ最適化テーブル) がある場合には、次のようなメッセージが表示されていました。



TDE は設定できるものの、メモリ最適化テーブルに関しては暗号化ができないという状態でした。 これが SQL Server 2016 からは、インメモリ OLTP にも対応するようになったので、メモリ最 適化テーブルでも暗号化ができるようになりました。



◆ TDE のオーバーヘッド ~性能検証結果~

TDE(透過的なデータ暗号化)を設定したことによる性能のオーバーヘッド(暗号化処理のオーバ

ーヘッドなど)が気になる方は多いと思います。 SQL Server 2016 での TDE (透過的なデータ暗 号化)のオーバーヘッドを検証してみた結果が、次のグラフです (2 章の インメモリ OLTP での 検証と同様、100 多重で 5,000 万件の INSERT を行った場合の検証)。



* ベンチマークの結果の公表は、使用許諾契約書で禁止されていますが、その効果をわかりやすく表現するため、 日本マイクロソフト株式会社の監修のもと、数値を掲載しております。

ディスク ベースの通常テーブルでは、<u>ラッチ待ちがボトルネック</u>になってしまうので、TDE を設 定しても、設定しなくても結果は変わらない(オーバーヘッドは確認できない)、インメモリ OLTP の Durable(SCHEMA_AND_DATA)では、TDE を設定しない場合が 8 分 32 秒であるのに対 して、TDE を設定したとしても 8 分 54 秒で完了、わずか 4.3% のオーバーヘッドであること を確認できました。

なお、SQL Server 2014 と SQL Server 2016 でのオーバーヘッドの差(AES-NI の効果)を検 証しようとも試みたのですが、前述したように SQL Server 2014 では インメモリ OLTP で TDE を設定することができないので、SQL Server 2014 のときにどれぐらいのオーバーヘッドで あったのかを知ることはできませんでした。また、ディスク ベースの通常テーブルの場合は、SQL Server 2014 でも、TDE なしでも有りでも、SQL Server 2016 と同様、約 27 分の結果になり、 ラッチ待ちがボトルネックになって、TDE のオーバーヘッドを確認することはできませんでした。

このように、TDE のオーバーヘッドを検証することは難しく、ディスク ベースの通常テーブルを 利用している場合には、ラッチ待ちやロック待ち、ログ書き込み待ち、チェックポイント負荷、イ ンデックス更新のオーバーヘッドなど、さまざまな TDE 以外のオーバーヘッドが発生するので、 それらがボトルネックになっている環境であれば、TDE を設定しても、設定しなくても、それらに 足を引っ張られて、TDE のオーバーヘッドが存在しないように見えるというぐらい、オーバーヘッ ドが軽いものになっています。

TDE は、ここ数年、弊社へのお問い合わせが非常に多くなっている機能です(マイナンバーや各種の情報漏洩事件など、業界全体のセキュリティ意識の高まりが大きな要因だと思っています)。後述しますが、TDE の設定は非常に簡単で、SQL Server 2016 からはインメモリ OLTP にも対応し、前掲のグラフのようにオーバーヘッドも軽く、アプリケーションは全く変更する必要がないので、

ぜひ活用してみてください。

➡ TDE(透過的なデータ暗号化)の設定方法

TDE(透過的なデータ暗号化)の利用方法は、以前のバージョンと変わりませんが、まだ試したことがない方も多いと思うので、ここではこれも試してみましょう。

TDE を設定するおおまかな流れは、次のとおりです。

- 1. master データベースに接続して、データベース マスター キーを作成(CREATE MASTER KEY ステートメントを利用)
- 2. サーバー証明書を作成(CREATE CERTIFICATE ステートメントを利用)
- 3. Management Studio のオブジェクト エクスプローラーで、該当データベースを右クリック して、[タスク] メニューの [データベース暗号化の管理] をクリックし、TDE を有効化する (または CREATE DATABASE ENCRYPTION KEY と ALTER DATABASE .. SET ENCRYPTION ON ステートメントを利用して TDE を有効化する)

Let's Try

それでは試してみましょう。

1. まずは、TDE を試すためのデータベースを「tdeTestDB」という名前で作成します。

```
-- データベースの作成
USE master
CREATE DATABASE tdeTestDB
```

2. 次に、CREATE MASTER KEY ステートメントを利用して、master データベース内にデー タベース マスター キーを作成します。

```
    -- データベース マスター キーの作成
    USE master
    CREATE MASTER KEY
    ENCRYPTION BY PASSWORD = '<strongPassword>'
```

```
USE master
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = '<StrongPassword>2'
go
100 % - <
動 メッセージ
コマンドは正常に完了しました。
```

ENCRYPTION BY PASSWORD には、任意のパスワードを設定しますが、強固なパスワー

- ド(大文字と小文字、@ などの特殊文字を含むなど)を設定するようにします。
- 3. 次に、CREATE CERTIFICATE ステートメントを利用して、サーバー証明書を作成します(こ れも master データベース内に作成します)。



「MyServerCert」は、サーバー証明書の名前なので、任意の名前を設定できます。

 次に、Management Studio のオブジェクト エクスプローラーで、次のようにデータベース を右クリックして、[タスク] メニューから [データベース暗号化の管理] をクリックします。



5. [データベース暗号化の管理] ダイアログが表示されたら、[データベース暗号化キー] セクションで [暗号化アルゴリズム] に任意のアルゴリズム (画面は AES_128) を選択して、[サ ーバー証明書を使用] で上の手順で作成した [MyServerCert] を選択します。

🍢 データベース暗号化の管理		_		×			
▲ データベース暗号化キーの暗号化に使用される証明書 'MyServerCert' がバックアップされていません。							
ページの選択	≤ スクリプト → 📑 ヘルプ						
	データベース暗号化キー						
	暗号化アルゴリズム(E): AES 128			~			
144.64	● サーバー証明書の使用(<u>U</u>): MyServerCert			\sim			
接続 ····································	○ ++-パー非対称キーの使用(A):			~			
<u>接続のプロパティの表示</u>	データベース暗号化オブション						
進行状況							
準備完了							
	З	キャンセル	Ŋ	パ			

暗号化アルゴリズムでは、AES_128、AES_192、AES_256 などを選択できますが、キーの長さに応じて(128bit より 192bit、192bit より 256bit に変更することで)、セキュリティの強度を上げることができます(その反面、性能とのトレードオフがあります)。

後は、[**データベース暗号化をオンに設定**] をチェックして、[**OK**] ボタンをクリックすれば、 TDE の設定が完了です。

なお、Management Studio ではなく、ステートメントで TDE を設定する場合には、次のように、CREATE DATABASE ENCRYPTION KEY ステートメントを利用してデータベース暗 号化キーを作成し、ALTER DATABASE ステートメントで SET ENCRYPTION ON まを指 定して、TDE を有効化します。

```
USE tdeTestDB
```

```
    -- データベース暗号化キーの作成
    CREATE DATABASE ENCRYPTION KEY
    WITH ALGORITHM = AES_128
    ENCRYPTION BY SERVER CERTIFICATE MyServerCert
    go
    -- データベースに対して暗号化を有効
    ALTER DATABASE tdeTestDB
    SET ENCRYPTION ON
    go
```

以上で設定が完了です。

→ 暗号化されていることの確認

TDE (透過的なデータ暗号化) によって、データやバックアップ ファイルが暗号化されていること を確認するには、次のように行えます。

USE tdeTestDB -- テーブルの作成 CREATE TABLE t1 (a int , b varchar(100)) -- データを 2件追加 INSERT INTO t1 VALUES (1, 'aaaaaaaaaaa') ,(2, 'あいうえお') SELECT * FROM t1 -- データベースのバックアップ BACKUP DATABASE tdeTestDB TO DISK = 'C:¥temp¥tdeTestDB.bak'

正しく暗号化されたかどうかは、SQL Server サービスを停止して、停止が完了した後に、 「tdeTestDB.mdf」(データ ファイル)を任意のバイナリ エディター(Stirling など)で開くこ とで確認できます(.mdf は、既定では C:¥Program Files¥Microsoft SQL Server¥MSSQL13 .MSSQLSERVER¥MSSQL¥DATA フォルダーに格納されています)。

🚮 Stirling - td	eTestDB.mdf		-	
ファイル(<u>F</u>) 編集(<u>I</u>	E) 検索・移動(S) 設定(Q) ウィンドウ(W) ヘルブ(H)			
	▝▎▞▏▓▝▆▐▋▙▓▏▓▏▓▎▓▎₩▎₩▎▓,▓▝▓▏Ď▎▙▎▙▖	👑 📲 🖨 🖉		
tdeTestDB.n	ndf			• 💌
ADDRESS	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	0123456789ABCDEF		^
00025A20	16 E6 D8 35 6B 03 75 0A EA 0E F3 99 F9 7A 9A DF	.齎5k.u・・壤		
00025A30	5D 52 BE 74 C9 28 5D 02 ED 7C D6 B5 56 C4 21 47]Rtt/(].埈ヨオVト!G		
00025A40	A2 23 1B 1A 1E 13 0B AC 2C AF 4F B8 70 62 B5 79	「#v,»02pbty		
00025A50	21 D6 70 38 53 E0 1E D6 8B 30 70 92 EE 8B F1 BE	!∃p8S∃.Op悌虞セ		
00025A60	80 FB 79 75 B7 06 F9 28 A5 49 2E 9A 16 27 4E 4U	□yuf(·l`NM		
00025A70	IF IE 38 51 /A CF 66 AE F9 55 88 IB CZ F5 A4 IC	8Uz√t∃•?•.		
00025A80	UB 33 78 9F 25 77 97 98 50 0F AT TB 29 32 0A 60	E3x.‰w本J¥Y。.JZ.m Want = ol o n		
00025A90	D2 D2 2D 2D 2D 2D 20 71 43 70 F1 A3 47 E0 00 0D 20 7 D2 D2 2D 2D 2D 0E C4 0E E0 2D 00 02 71 AE 21 D4 22	M7AB木・−96 ・GM&		
00025440	DO UD OD AD UF C4 UF EU ZU 00 00 71 AF OT DA OZ AR 50 70 75 EV A1 95 DV 90 7V 90 CV 5V 22 E5 EV	Pour + 72 -	0	
00025AD0	A8 E4 98 C9 38 A9 DD 80 75 38 45 60 6E 18 84 E4	1 pu2 pzs - 7詰 /2 ` 」 & ` 。 贋		暗号化されてい
00025400	54 28 DB 5E 15 92 44 0B CE 39 EC 7E 7E 34 26 49	7+□^公 ▽9 ~:&I		ことを確認
00025AE0	1A 64 DD 4F 61 22 B2 06 DF 5C F5 13 4A 75 45 59	.d≫Na″1.`¥JuFY [∠]		
00025AF0	AF 04 82 95 D3 FB 1C 03 FD A9 6D C3 F0 FA BF 14	a, ,,モウmデ確ソ.		
00025B00	79 C4 9F C1 DC 04 39 4A AF 28 16 5A 82 1C 4A 78	yト淬ワ.9Jッ(.ZJx		
00025B10	4E 02 9F 52 E2 1D EA FA 37 BB D9 15 78 24 F9 64	N.欒・7サル.×\$・		
00025B20	F0 F0 7B F9 8E D4 8D A8 22 04 92 3C DB B8 3C FE	• [•]昏"<四外.		
00025B30	49 45 71 94 7F 45 98 6A 29 D3 8D 48 AE AA 4C 3C	IEqE亘)モ工ョェレ<		
00025B40	15 F5 BF E2 AD FA 9A 6B 1D 9E 92 77 A5 52 DE 77	.•箝埇k.梺w·R`w		
00025B50	8F 2A A0 78 5F CF AC F4 B2 A9 75 3F 75 7A 78 C5	.*.x_?r •u?uzx+		
00025B60	99 30 B7 6F E6 91 FE D8 46 43 E3 F9 51 2F D4 1B	.Ufo鞫.yFU脩Q/Y.		
00025B70	EB 37 ET 91 98 8F 39 Z7 18 3F 48 A3 19 8Z 47 70	./摟寶9.?K」.。}		
00025880	1 44 13 IF FA ZF FE UZ IC 81 DZ E5 6F 49 4E C7 CB	D/・竜INXE		~
レディ	0×000000	100 上書 44564	148 Bytes	SHIFT-JIS

バイナリ エディターで開いても、どういったデータが格納されているかを読み取ることはできず、 きちんと暗号化されていることを確認できます。同じように、バックアップ ファイル 「**tdeTestDB.bak**」についても暗号化がされていることを確認することができます。

このように、TDE を利用すれば、物理ファイル(.mdf やバックアップ ファイル)を暗号化する ことができるのがメリットです。

なお、暗号化されていないデータベース(やバックアップ ファイル)をバイナリ エディターで開いた場合には、次のようにデータを読み取ることが可能です。



別マシンでのリストアはエラー

TDE(透過的なデータ暗号化)を設定したデータベースのバックアップは、別のマシンでリストアしようとすると、次のようにエラーが発生します。



サーバー証明書が存在しないので、リストアができないという主旨です。TDE では、<u>バックアップ</u> ファイルが持ち出されたとしても、簡単にはリストアできないようになっています(かつ中身が暗 号化されているので、安全性が高いものとなっています)。

また、データ ファイル (.mdf) を別マシンでアタッチしようとしても、次のように同じエラーが 発生します。

🕵 04_TDE.sql - WIN10.master (WIN10¥matu_tak (51)) - Microsoft SQL Server M	anagement Stu	
ファイル(F) 編集(E) 表示(V) プロジェクト(P) 间 データベースのインポート	– 🗆 X	
① ▼ □ ▼ 22 日 22 新しいクエリ(N) ページの選択 型 辺 master - 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	③スクリプト マ 国へルプ	
オブジェクト エクスプローラー	アタッチするデータベース(<u>D</u>):	
接続 - 影 製 = 7 🙋 📓	MDF ファイルの場所 データベース名 次の名前でアタッチ F	
WIN10 (SQL Server 13.0.700 - WIN10¥m D T T	C¥temp¥tdeTestDB.mdftdeTestDB tdeTestDB t	
 ● ae ● ブータベースの復元(R) ● D ○ ファイルおよびファイルグル 	Microsoft SQL Server Management Studio	
 □ □<td>この要求のデータを取得できませんでした。(Microsoft SqlServer, Management, Sql., Sfc) 这加情報: し、Therapy LSC! フラートコントサナイナズの主が日本(1)(例め) 約3分化 ま) か</td><td></td>	この要求のデータを取得できませんでした。(Microsoft SqlServer, Management, Sql., Sfc) 这加情報: し、Therapy LSC! フラートコントサナイナズの主が日本(1)(例め) 約3分化 ま) か	
● Re PowerShell の起動(H) ● Fr Latt_k(m)	 ・	
Te Te E	TC077	_
接続: WIN10¥matu_tak ■】接続のプロパティの表示		

このように、<u>データ ファイルについても、簡単にアタッチできない</u>ように、セキュリティが高いものになっています。

→ サーバー証明書のバックアップとリストア

開発機から本番機へ、あるいはその逆へなど、別マシンヘデータベースを移動させたい場合には、 次のように BACKUP CERTIFICATE ステートメントを利用して、サーバー証明書(と秘密キー) のバックアップを実行しておく必要があります。

```
BACKUP CERTIFICATE MyServerCert

TO FILE = 'C:¥temp¥MyServerCert'

WITH PRIVATE KEY

( FILE = 'C:¥temp¥MyPrivKey'

, ENCRYPTION BY PASSWORD = 'P@ssword' )
```

```
-- サーバー証明書のバックアップ

USE master

BACKUP CERTIFICATE MyServerCert

TO FILE = 'C:¥temp¥MyServerCert'

WITH PRIVATE KEY

(FILE = 'C:¥temp¥MyPrivKey'

, ENCRYPTION BY PASSWORD = 'P@ssword')

100 % <</td>

    100 % <</td>

    コマンドは正常に完了しました。
```

このステートメントでは、サーバー証明書をファイルヘバックアップすることができ、WITH PRIVATE KEY オプションを付けることで、秘密キーもバックアップできます。このときに ENCRYPTION BY PASSWORD で指定するパスワードには、推測されにくい強固なパスワード を設定して、そのパスワードが簡単に漏れないように注意しておく必要があります。

バックアップしたファイル(サーバー証明書と秘密キー)は、別マシンに物理的にコピーして、次のように CREATE CERTIFICATE ステートメントを利用して、リストア(インポート)すること

ができます。

```
-- データベース マスター キーの作成
USE master
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = '<StrongPassword>'
go
-- サーバー証明書のリストア (インポート)
CREATE CERTIFICATE MyServerCert
FROM FILE = 'C:¥temp¥MyServerCert'
WITH PRIVATE KEY
(FILE = 'C:¥temp¥MyPrivKey'
, DECRYPTION BY PASSWORD = 'P@ssword')
```



サーバー証明書のリストア(ファイルからの作成)には、**データベース マスター キー**を作成して おく必要があるので、まずデータベース マスター キーを作成しています。

サーバー証明書のリストア時(CREATE CERTIFICATE ステートメントの実行時)には、 DECRYPTION BY PASSWORD に、複合化のためのパスワードを、バックアップ時に指定した のと同じものを指定します。これで、サーバー証明書のリストアが完了です。

このように同じサーバー証明書がリストアされている環境であれば、透過的なデータ暗号化(TDE) を設定したデータベースをリストアして、データを参照することができるようになります。

```
-- サーバー証明書がリストアされていれば、DB のリストアも可能

RESTORE DATABASE tdeTestDB

FROM DISK='C:¥temp¥tdeTestDB.bak'

100 % ・

国 メッセージ

データベース 'tdeTestDB' の 344 ページ、ファイル 1 のファイル 'tdeTestDB' を処理しました。

データベース 'tdeTestDB' の 2 ページ、ファイル 1 のファイル 'tdeTestDB' を処理しました。

RESTORE DATABASE により 346 ページが 0.054 秒間で正常に処理されました(49.940 MB/秒)。
```

その他、TDEの詳細(アーキテクチャなど)については、オンライン ブックの以下のトピックがお勧めです。

透過的なデータ暗号化(TDE)

http://msdn.microsoft.com/ja-jp/library/bb934049.aspx



3.6 テンポラル テーブルによる Audit (監査証跡)

テンポラル テーブル(Temporal Tables)は、テーブルに対する過去の更新履歴を自動的に履歴テ ーブルに保存できる機能です。これを利用すれば、次のような用途で利用することができます。

- タイム トラベル
- **行レベルでのデータ修復**(操作ミスなどへの対応)
- Audit (コンプライアンスにおける監査証跡)
- **SCD**(Slowly Changing Dimension:緩やかに変化するディメンション)

テンポラル テーブルを利用すれば、タイム マシンのように、過去のデータに簡単にアクセスする ことができるので、操作ミスが発生した場合のデータ修復や、コンプライアンスにおける監査証跡 (Audit Log:監査ログ)、データ分析における SCD(緩やかに変化するディメンション)として 利用することができます。

テンポラル テーブルを Audit 目的で利用すれば、J-SOX 法(日本版 SOX 法)や内部統制、PCI DSS (クレジット カード情報のセキュリティ基準) などのコンプライアンス (法令遵守) を実現す るために利用できます (なお、これらを実現するために、従来からある SQL Server Audit 機能 を利用することも、SQL Server 2016 ではもちろん可能です。違いについては以下の Note 参照)。

Note: CDC や Change Tracking との違いは「使いやすさ」、SQL Server Audit との違い

これまでの SQL Server には、CDC (Change Data Capture : 変更データ キャプチャ)や Change Tracking (変 更追跡) という、テンポラル テーブルと似たような機能 (いずれも更新履歴を自動保存できる機能) がありました。実 際、CDC や Change Tracking を利用して、タイム トラベルや行レベルのデータ修復、Audit、SCD を実現している 場合もあると思います。今回提供された「テンポラル テーブル」と CDC/Change Tracking との大きな違いは、今 回のテンポラル テーブルは、CDC/Change Tracking よりも使いやすい (履歴テーブルを操作しやすい) という点 です。CDC/Change Tracking では、タイム トラベルがしづらかったり、行レベルでのデータ修復時に、該当デー タを探すのが大変だったりしましたが、今回のテンポラル テーブルではそういったことはありません。テンポラル テ ーブルを利用すれば、簡単に過去のデータを参照することができます。

テンポラル テーブルと **SQL Server Audit** 機能との比較は、SQL Server Audit 機能が「**監査**」目的に特化した機能 で、非常に細かく監査イベントを設定できる(データの更新履歴だけでなく、ログインの成功/失敗や、アカウントの パスワード変更、データベースそのものに対する変更など、いろいろな種類のイベントをきめ細かく設定できる)こと が大きな違いです。テンポラル テーブルで行える Audit は、テンポラル テーブルを設定したテーブルに対して、過去 の更新履歴を記録/追跡できるようになることで、ログインの成功/失敗などを追跡する目的では利用することができ ません。テンポラル テーブルを利用するメリットは、Audit 目的で利用できるだけでなく、その他の用途(操作ミス時 のデータ修復や SCD シナリオなど)でも利用できる点です。

→ テンポラル テーブルの作成方法

テンポラル テーブルは、次のように作成することができます。

-- テンポラル テーブルの作成例 CREATE TABLE Products (商品コード int PRIMARY KEY, ---PRIMARY KEY は必須 商品名 nvarchar(40), sysstart datetime2 GENERATED ALWAYS AS ROW START HIDDEN NOT NULL, ---開始時刻用 sysend datetime2 GENERATED ALWAYS AS ROW END HIDDEN NOT NULL, ---終了時刻用 PERIOD FOR SYSTEM_TIME (sysstart, sysend))) WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo. ProductsHistory)) 一更新履歴を格納するテーブルの指定

テンポラル テーブルでは、PRIMARY KEY 制約(主キー制約)が必須になるので、PRIMARY KEY 制約を設定した列を設けるようにします。また、テンポラル テーブルでは、履歴/更新日時を管理 するための列が 2 つ必要になります(上の例では sysstart と sysend という名前で作成した列 で、任意の名前を付けることが可能)。この 2 つの列には、「GENERATED ALWAYS AS」で「ROW START」および「ROW END」を指定するようにして、「PERIOD FOR SYSTEM_TIME」で列 の名前を指定するようにします。また、HIDDEN 属性を付けることで、列データを隠すことがで きるようになります(SELECT * で検索しても、表示されない列に設定できるようになります)。

そして WITH 句で「SYSTEM_VERSIONING = ON」を指定することで、テンボラル テーブル を有効化して、「HISTORY_TABLE =」で更新履歴を格納するテーブルの名前を指定すれば、テン ポラル テーブルの作成が完了です。

既存のテーブルをテンポラル テーブルに変更する場合 ~ALTER TABLE~

テンポラル テーブルを既存のテーブルに対して設定したい場合には、次のように ALTER TABLE ステートメントを実行します。

```
    -- 日時を格納するための列を 2つ追加 (データがある場合には DEFAULT 制約が必須)
        ALTER TABLE Products
        ADD sysstart datetime2 GENERATED ALWAYS AS ROW START HIDDEN
        DEFAULT CONVERT(datetime2, GETUTCDATE()), -- DEFAULT 制約
        sysend datetime2 GENERATED ALWAYS AS ROW END HIDDEN
        DEFAULT CONVERT(datetime2, '9999-12-31 23:59:59.9999999'), -- DEFAULT 制約
        PERIOD FOR SYSTEM_TIME ( sysstart, sysend )
        -- テンポラリ テーブルを有効化と履歴テーブルの指定
        ALTER TABLE Products
        SET ( SYSTEM_VERSIONING = ON
        (HISTORY_TABLE = dbo. ProductsHistory ) )
```

既存のテーブルのデータが空(データ件数が 0)の場合には、DEFALUT 制約は必要ありませんが、 データが存在している場合には、更新日時を管理するための列に対して DEFAULT 制約を設定す る必要があります。上の例では、開始日時(sysstart 列)は GETUTCDATE 関数で、現在時刻を UTC(協定世界時)で指定していますが、「'2016-01-01'」のように特定の開始日時を指定するこ とも可能です。終了時刻(sysend 列)は、datetime2 データ型の最大値である「'9999-12-31 23:59:59.99999999'」を指定しておくようにします。

2 つの列(sysstart、sysend)を追加した後は、ALTER TABLE ステートメントで
 「SYSTEM_VERSIONING = ON」を指定することで、テンポラル テーブルを有効化できます。

Let's Try

それでは、これを試してみましょう。

1. まずは、テンポラル テーブルを試すためのデータベースを「temporalTestDB」という名前 で作成します。

-- データベースの作成
 CREATE DATABASE temporalTestDB

 次に、「NorthwindJ」データベース内から「商品」テーブルから「商品コード」と「商品名」 を抜き出して、テーブルを作成します(なお、NorthwindJ データベースの作成方法は、巻末 の付録に記載しています)。

USE temporalTestDB SELECT 商品コード, 商品名 INTO Products FROM NorthwindJ..商品



作成するテーブル名は「Products」にしていますが、任意の名前でかまいません。

 次に、作成した Products テーブルをテンポラル テーブルに変更していきます。テンポラル テーブルの作成には、PRIMARY KEY 制約(主キー制約)が必須になるので、次のように ALTER TABLE .. ADD CONSTRAINT ステートメントを実行して、「商品コード」列に PRIMARY KEY 制約を設定します。

-- PK を追加 ALTER TABLE Products ADD CONSTRAINT pk1 PRIMARY KEY(商品コード)



 次に、ALTER TABLE ステートメントを利用して、テンポラル テーブルで必要になる 2 つの 列(日付を格納するための列)を追加します。 ALTER TABLE Products ADD sysstart datetime2 GENERATED ALWAYS AS ROW START DEFAULT CONVERT (datetime2, '2016-01-01'), sysend datetime2 GENERATED ALWAYS AS ROW END DEFAULT CONVERT (datetime2, '9999-12-31 23:59:59.9999999'), PERIOD FOR SYSTEM_TIME (sysstart, sysend)

列の名前は「sysstart」と「sysend」にしていますが、任意の列名を付けることができます。 「sysstart」列には「GENERATED ALWAYS AS」で「ROW START」、「sysend」列には 「ROW END」を指定して、「PERIOD FOR SYSTEM_TIME (sysstart, sysend)」を記述 することで、テンポラル テーブル用の列 (更新日時/履歴を制御するための日付列) にするこ とができます。また、「sysstart」には DEFAULT 制約で「'2016-01-01'」を指定していま すが、前述したように GETUTCDATE 関数を利用して現在時刻を指定してもかまいません。

HIDDEN 付きがお勧め

上の手順では、「sysstart」と「sysend」列に HIDDEN 属性をあえて付けませんでしたが、 実際の運用では、次のように HIDDEN 属性を付けるのがお勧めです (HIDDEN 属性を付け れば、SELECT * で検索しても、表示されない列に設定することができます)。

ALTER TABLE Products ADD sysstart datetime2 GENERATED ALWAYS AS ROW START HIDDEN DEFAULT CONVERT(datetime2, '2016-01-01'), sysend datetime2 GENERATED ALWAYS AS ROW END HIDDEN DEFAULT CONVERT(datetime2, '9999-12-31 23:59:59.9999999'), PERIOD FOR SYSTEM_TIME (sysstart, sysend)

テンポラル テーブルの内部動作を理解するには、HIDDEN 属性を外しておいた方が分かり やすいので、以降の手順では、HIDDEN 属性を外した状態で説明していきます。

Note: DEFAULT 制約を設定していない場合 2 つの列 (sysstart、sysend) を追加する際に、DEFAULT 制約を指定しないで ALTER TABLE ステートメ ントを実行した場合は、次のようにエラーになります(データが存在している場合はエラーになります) ALTER TABLE Products ADD sysstart datetime2 GENERATED ALWAYS AS ROW START, sysend datetime2 GENERATED ALWAYS AS ROW START, period FOR SYSTEM_TIME (sysstart, sysend) 100 % ・ (Mage: System of the sy

5. 次に、テンポラリ テーブルとして有効化するために、次のように ALTER TABLE ステート メントを実行します。

```
ALTER TABLE Products
SET ( SYSTEM_VERSIONING = ON
( HISTORY_TABLE = dbo.ProductsHistory ) )
```



「SYSTEM_VERSIONING = ON」を指定することで、テンポラル テーブルとして有効化 することができ、「HISTORY_TABLE = dbo.ProductsHistory」と指定することで、過去 のデータを保存しておく履歴テーブルの名前を「ProductsHistory」に設定しています(任意 の名前に設定することができます)。

以上で、テンポラル テーブルの設定が完了です。

Note: PRIMARY KEY 制約を設定していない場合
テーブルに PRIMARY KEY 制約を設定していない場合には、次のようにエラーが表示されます。
PK なしでテンボラリ テーブルを有効化しようとする場合 PALTER TABLE Products SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.ProductsHistory))
100% - <
メッセージ 13553、レベル 16、状態 1、行 23 システムでバージョン管理されたテンボラル テーブル 'temporalTestDB2.dbo.Products' には主キーを定義する必要があります。

→ データを更新してみよう

次に、**Products** テーブルのデータを更新して、更新履歴が「**ProductsHistory**」テーブルに格納 されていくことを確認してみましょう。

1. まずは、データを更新する前のデータを確認しておきます。

SELECT * FROM Products

	データの確認 SELECT * FROM Products					
100	% * <					
	結果 🛅 メッセージ					
	商品コード	商品名	sysstart	sysend		
1	1	果汁100% オレンジ	2016-01-01 00:00:00.0000000	9999-12-31	23:59:59.9999999	
2	2	果汁100% グレープ	2016-01-01 00:00:00.0000000	9999-12-31	23:59:59.9999999	
3	3	果汁100% レモン	2016-01-01 00:00:00.0000000	9999-12-31	23:59:59.9999999	
4	4	果汁100%ピーチ	2016-01-01 00:00:00.0000000	9999-12-31	23:59:59.9999999	
5	5	コーヒーマイルド	2016-01-01 00:00:00.0000000	9999-12-31	23:59:59.9999999	

sysstart と **sysend** 列には、**DEFAULT 制約**で指定した値が格納されていることを確認で きます。

なお、**sysstart** と **sysend** 列に **HIDDEN 属性**を付けている場合には、**SELECT** ステート メントでデータを検索しても、次のように、これらの列が表示されない形になります。

Ē	ALTER TABLE Products ADD sysstart datetime2 GENERATED ALWAYS AS ROW START HIDDEN DEFAULT CONVERT(datetime2, '2016-01-01'), sysend datetime2 GENERATED ALWAYS AS ROW END HIDDEN					
	DEFAULT CONVERT(datetime2, '9999-12-31 23:59:59.99999999'), PERIOD FOR SYSTEM_TIME (sysstart, sysend)					
E	BALTER TABLE Products SET (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.ProductsHistory))					
	SELECT * FRO	M Products				
100 %	• •					
🛄 á	結果 🛅 メッセージ					
	商品コード	商品名				
1	1	果汁100% オレンジ				
2	2	果汁100% グレープ				
3	3	果汁100% レモン				
4	4	果汁100%ピーチ	HIDDEN 付き			
5	5	コーヒーマイルド	の列は表示されない			
6	6	コーヒービター				

2. 次に、UPDATE ステートメントを実行して、データを更新してみます。

UPDATE Products SET 商品名 = 'AAAA' WHERE 商品コード = 1 UPDATE Products SET 商品名 = 'BBB' WHERE 商品コード = 2

商品コードが「1」の商品名(果汁 100%オレンジ)を「AAAA」、商品コードが「2」の商品 名(果汁 100%グレープ)を「BBB」に変更します。

3. データの更新が完了したら、SELECT ステートメントを実行して、データを確認します。

SELECT * FROM Products

データの確認 SELECT * FROM Products 100 % ▼ < ■ 結果 1 メッセージ				
	商品コード	商品名	sysstart	sysend
1	1	АААА	2016-01-11 04:18:32.2304961	9999-12-31 23:59:59.9999999
2	2	BBB	2016-01-11 04:18:32.2313230	9999-12-31 23:59:59.9999999
3	3	果汁100% レモン	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999
4	4	果汁100%ピーチ	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999
5	5	コーヒーマイルド	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999
6	6	コーヒービター	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999

sysstart 列の値が、データを更新した日時に変更されていることを確認できます。なお、日時は UTC(協定世界時)で記録されるので、日本の場合は、+9時間(上のように 1月11日 13時18分)した値になります。

4. 次に、もう一度 UPDATE ステートメントを実行して、さらに異なる値にデータを更新してみ

ます。

SELECT * FROM Products

UPDATE Products SET 商品名 = 'AAAA_2nd' WHERE 商品コード = 1 UPDATE Products SET 商品名 = 'BBB_2nd ' WHERE 商品コード = 2

商品コードが「**1**」の商品名を「AAAA_2nd」、商品コードが「**2**」の商品名を「BBB_2nd」 に変更します。

5. データの更新後、SELECT ステートメントを実行して、データを確認します。

-- データの確認(現在のデータ) SELECT * FROM Products 100 % 🔹 < 💷 結果 🚹 メッセージ 商品コード 商品名 svsend svsstart 1 AAAA_2nd 2016-01-11 04:23:27.6951657 9999-12-31 23:59:59.9999999 1 2016-01-11 04:23:27.6971647 9999-12-31 23:59:59.9999999 2 BBB_2nd 2 2016-01-01 00:00:00:0000000 9999-12-31 23:59:59:9999999 果汁100%レモン 3 3 果汁100%ピーチ 2016-01-01 00:00:00:0000000 9999-12-31 23:59:59:9999999 4 4 コーヒーマイルド 5 5 2016-01-01 00:00:00:0000000 9999-12-31 23:59:59:9999999 コーヒービター 6 6 2016-01-01 00:00:00:0000000 9999-12-31 23:59:59:9999999

sysstart 列の値が、データを更新した日時に変更されていることを確認できます。

→ 履歴テーブル確認

次に、更新履歴が格納されている履歴テーブル(ProductsHistory)を確認してみましょう。

1. まずは、SELECT ステートメントを実行して、中身を参照してみます。

SELECT * FROM ProductsHistory					
履歴データの確認 (更新履歴) SFLFCI * FROM ProductsHistory					
100 %	• <	-			
🛄 結果	果 🛅 メッセージ				
	商品コード	商品名	sysstart	sysend	
1	1	果汁100% オレンジ	2016-01-01 00:00:00.0000000	/2016-01-11 04:18:32.2304961	
2	2	果汁100% グレープ	2016-01-01 00:00:00.0000000 /	2016-01-11 04:18:32.2313230	
3	1	АААА	2016-01-11 04:18:32.2304961	2016-01-11 04:23:27.6951657	
4 2 BBB 2016-01-11 04:18:32:2313230 2016-01-11 04:23:27:6971647					
複数のデータ更新がある場合には sysend と、次の sysstart が同じ値になる					

更新前の過去のデータ(商品コード 1は、果汁 100%オレンジと AAAA、商品コード 2は、 果汁 100%グレープと BBB) が表示されて、sysstart (開始日時) と sysend (終了日時) で有効期間 (いつからいつまでの値だったのか) が記録されていることを確認できます。また、 複数のデータ更新がある場合には、sysend と、次の sysstart が同じ値 (終了日時が、次の 開始日時)になります。

2. 次に、オブジェクト エクスプローラーで、履歴テーブルを確認してみましょう。



テンポラル テーブルを設定すると、テーブル名(Products)の隣に「システム バージョン 管理」と表示されて、テーブルを展開すると、「(履歴)」と表示される履歴テーブル (PorudtcsHistory)を確認することができます。

このように、テンポラル テーブルを設定すると、自動的に履歴テーブルが作成されて、このテ ーブルに更新履歴が格納されていくようになります。

→ 過去のデータへのアクセス ~FOR SYSTEM_TIME AS OF~

<u>テンポラル テーブルの一番のメリットは、過去のデータ(履歴)に簡単にアクセスできる</u>点にあり ます(従来の CDC や Change Tracking 機能よりも簡単に過去データを参照することができま す)。これも試してみましょう。

 まずは、過去の任意の日時を指定して、データにアクセスしてみましょう。これを行うには、 SELECT ステートメントで、次のように、テーブル名の後に「FOR SYSTEM_TIME AS OF '戻りたい日時'」を付けます。

```
SELECT * FROM Products
FOR SYSTEM_TIME AS OF '2016-01-01'
ORDER BY 商品コード
```

	更新前のデータ(過去の指定した日時のデータ)を確認 SELECT * FROM Products FOR SYSTEM_TIME AS OF '2016-01-01' ORDER BY 商品コード						
100	% - <						
	結果 🚹 メッセージ						
	商品コード	商品名	sysstart	sysend			
1	1	果汁100% オレンジ	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32:2304961			
2	2	果汁100% グレープ	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32.2313230			
3	3	果汁100% レモン	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999			
4	4	果汁100%ピーチ	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999			
5	5	コーヒーマイルド	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999			
6	6	コーヒービター	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999			

日時には、「2016-01-01」を指定していますが、この値は、DEFAULT 制約で sysstart 列 に設定した日付で、この時点(最初の時点)でのデータを参照することができます。したがっ て、商品コード 1 には 果汁 100%オレンジ、商品コード 2 には、果汁 100%グレープ(い ずれも更新する前のデータ)が表示されています。

なお、「FOR SYSTEM_TIME AS OF」は、WHERE 句を利用する場合には、次のように WHERE 句の前に記述するようにします。



2. 次に、「FOR SYSTEM_TIME AS OF」で指定する日時に、2015 年の日付(2015-12-01 など)を指定してデータを参照してみます。

SELECT * FROM Products FOR SYSTEM_TIME AS OF '2015-12-01' ORDER BY 商品コード



結果は、何もデータが返ってきません。sysstart 列の **DEFAULT 制約**で「2016-01-01」 を指定しているので、初期データは「2016-01-01」時点以降のデータという扱いになり、そ れ以前のデータは存在しない、ということでこのような動作になります。

3. 次に、1つ目の更新データ(商品コード **1** の AAAA)を参照するために、ProducttsHistory テーブルの sysstart 列の値を確認してみます。 SELECT * FROM ProductsHistory

	更新日時を確認 SELECT * FROM ProductsHistory							
100	100 % - <							
	💷 結果 🚦 メッセージ							
		商品コード	商品名	sysstart	sysend			
1		1	果汁100% オレンジ	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32.2304961			
2		2	果汁100% グレープ	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32.2313230			
3	[1	АААА	2016-01-11 04:18:32:2304961	2016-01-11 04:23:27.6951657			
4		2	BBB	2016-01-11 04:18:32:2313230	2016-01-11 04:23:27.6971647			

sysstart(開始日時)と sysend(終了日時)の間が、そのデータの有効期間(いつからいつ までの値だったのか)になります。

4. 上で調べた「AAAA」データの有効期間内の値を指定して、データを参照してみましょう。

```
SELECT * FROM Products
FOR SYSTEM_TIME AS OF '2016-01-11 04:19:00'
ORDER BY 商品コード
```

	AAAA の範囲内の日時を指定 SELECT * FROM Products FOR SYSTEM_TIME AS OF '2016-01-11 04:19:00' ORDER BY 商品コード							
100 %	100 % 👻 <							
	🖽 結果 🔓 メッセージ							
	商品コード	商品名	sysstart	sysend				
1	1	АААА	2016-01-11 04:18:32.2304961	2016-01-11 04:23:27.6951657				
2	2	BBB	2016-01-11 04:18:32:2313230	2016-01-11 04:23:27.6971647				
3	3	果汁100% レモン	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999				
4	4	果汁100%ピーチ	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999				

画面では、sysstart は「2016-01-11 04:18:32.2304961」、sysend は「2016-01-11 04:23:27.6951657」なので、その間の「'2016-01-11 04:19:00'」を指定することで、「AAAA」データを参照できています。

5. 次に、sysend (2016-01-11 04:23:27.6951657) よりも大きい値を指定して、データを 参照してみます。

SELECT * FROM Products FOR SYSTEM_TIME AS OF '2016-01-11 04:25:00' ORDER BY 商品コード

AAAA の sysend より大きい時刻を指定 SELECT * FROM Products FOR SYSTEM TIME AS OF '2016-01-11 04:25:00' ORDER BY 商品コード								
100	100 % - <							
	🎟 結果 🚺 メッセージ							
	商品コード	商品名	sysstart	sysend				
1	1	AAAA_2nd	2016-01-11 04:23:27.6951657	9999-12-31 23:59:59.9999999				
2	2	BBB_2nd	2016-01-11 04:23:27.6971647	9999-12-31 23:59:59.9999999				
3	3	果汁100% レモン	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999				

今度は、最後に更新した値である「AAAA_2nd」を参照できることを確認できます。

このように、「FOR SYSTEM_TIME AS OF」を利用することで、ある特定の時点でのデータ を簡単に参照することができます。なお、前述したように、更新日時は UTC (協定世界時)で 記録されているので、日本の場合は +9 時間を加算した日時を指定することで、任意の時点の データを参照できるようになります。

→ その他の演算子

テンポラル テーブルでは、「FOR SYSTEM_TIME」の「AS OF」の他に、「ALL」や「BETWEEN .. AND」、「FROM .. TO」、「CONTAINED IN(..)」という演算子を利用して、過去のデータを参照 することもできます。これも試してみましょう。

1. まずは、ALL を利用してみましょう。

SELECT * FROM Products FOR SYSTEM_TIME ALL ORDER BY 商品コード

	ALL SELECT * FROM Products FOR SYSTEM_TIME ALL ORDER BY 商品コード, sysstart								
100 9	∕o • <								
	結果 🚹 メッセージ								
	商品コード	商品名	sysstart	sysend					
1	1	果汁100% オレンジ	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32.2304961					
2	1	АААА	2016-01-11 04:18:32.2304961	2016-01-11 04:23:27.6951657					
3	1	AAAA_2nd	2016-01-11 04:23:27.6951657	9999-12-31 23:59:59.9999999					
4	2	【果汁100% グレープ】	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32:2313230					
5	2	BBB	2016-01-11 04:18:32.2313230	2016-01-11 04:23:27.6971647					
6	2	BBB_2nd	2016-01-11 04:23:27.6971647	9999-12-31 23:59:59.9999999					
7	3	果汁100% レモン	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999					
8	4	果汁100%ピーチ	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999					

このように、ALL を指定した場合は、過去の更新データをすべて表示することができます。

```
SELECT * FROM Products
FOR SYSTEM_TIME BETWEEN '2016-01-02' AND '2016-01-13'
ORDER BY 商品コード, sysstart
```

	BETWEEN AND SELECT * FROM Products FOR SYSTEM_TIME BETWEEN '2016-01-02' AND '2016-01-13' ORDER BY 商品コード, sysstart							
100	% • <							
	結果 🛅 メッセージ							
	商品コード	商品名	sysstart	sysend				
1	1	【果汁100% オレンジ】	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32.2304961				
2	1	АААА	2016-01-11 04:18:32.2304961	2016-01-11 04:23:27:6951657				
3	1	AAAA_2nd	2016-01-11 04:23:27.6951657	9999-12-31 23:59:59.9999999				
4	2	(果汁100% グレープ)	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32.2313230				
5	2	BBB	2016-01-11 04:18:32.2313230	2016-01-11 04:23:27:6971647				
6	2	BBB_2nd	2016-01-11 04:23:27.6971647	9999-12-31 23:59:59.9999999				
7	3	果汁100% レモン	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999				

^{2.} 次に、BETWEEN .. AND を利用してみましょう。

BETWEEN では、AND で複数の日時を指定して、その間のデータをすべて表示することができます。

3. 次に、**BETWEEN** の **AND** で指定する日時を、「AAAA」 データの sysend と同じ日時にして、データを参照してみます。

SELECT * FROM Products FOR SYSTEM_TIME BETWEEN '2015-10-21 14:24' AND **'2016-01-11 04:23:27.6951657'** ORDER BY 商品コード

BETWEEN AND で AAAA の sysend を指定 SELECT * FROM Products FOR SYSTEM_TIME BETWEEN '2016-01-02' AND '2016-01-11 04:23:27.6951657' ORDER BY 商品コード, sysstart								
100 %	• <							
🋄 結:	果 🛅 メッセージ							
	商品コード	商品名	sysstart	sysend				
1	1	果汁100% オレンジ	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32.2304961				
2	1	АААА	2016-01-11 04:18:32.2304961	2016-01-11 04:23:27.6951657				
3	1	AAAA_2nd	2016-01-11 04:23:27.6951657	9999-12-31 23:59:59.9999999				
4	2	果汁100% グレープ	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32.2313230				
5	2	BBB	2016-01-11 04:18:32.2313230	2016-01-11 04:23:27.6971647				
6	3	果汁100% レモン	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999				
7	4	果汁100%ピーチ	2016-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999				

「AAAA」データの sysend は、「AAAA_2nd」の sysstart 値でもあるので、両方のデー タを参照できています。

4. 次に、BETWEEN を FROM、AND を TO に変更して、同じように参照してみます。

SELECT * FROM Products FOR SYSTEM_TIME FROM '2015-10-21 14:24' TO '2016-01-11 04:23:27.6951657' ORDER BY 商品コード

s c	BETWEEN を FROM、AND を TO に変更して検索 SELECT * FROM Products FOR SYSTEM_TIME FROM '2015-10-21 14:24' TO '2016-01-11 04:23:27.6951657' ORDER BY 商品コード								
100 %	100 % 💌 <								
🋄 結:	果 🚹 メッセージ								
	商品コード	商品名	sysstart	sysend					
1	1	果汁100% オレンジ	2016-01-01 00:00:00.0000000	2016-01-11 04:18:32.2304961					
2	1	АААА	2016-01-11 04:18:32.2304961	2016-01-11 04:23:27.6951657					
3	2	BBB	2016-01-11 04:18:32.2313230	0 2016-01-11 04:23:27.6971647					
4	2	果汁100% グレィー	0000000	2016-01-11 04:18:32.2313230					
5	3	果汁100%レモ 🗛	A 2nd はヒットしない 0000000	9999-12-31 23:59:59.9999999					
6	4	果汁100%ピーナ	2016-01-01 00:00:00.000000	9999-12-31 23:59:59.9999999					
7	5	コーヒーマイルド	2016-01-01 00:00:00.000000	9999-12-31 23:59:59.9999999					

今度は、「AAAA」は参照できるものの、「AAAA_2nd」はヒットしないことが分かります。 このように、「BETWEEN date1 AND date2」と「FROM date1 TO date2」の違いは、 「sysstart <= date2」になるか、「sysstart < date2」になるか(= を含むかどうか)で す。BETWEEN の場合は = を含み、FROM の場合は = を含みません。

このように、テンポラル テーブルでは、過去の履歴データに対して、いろいろなデータ参照方

法が用意されているので、過去のデータを非常に扱いやすくなっています。

◆ テンポラル テーブルを無効化したい場合(通常テーブルに戻したい場合)

1. テンポラル テーブルを設定したテーブルは、DROP TABLE ステートメントで削除しようと しても、次のようにエラーが表示されます。



 テンポラル テーブルを無効化したい場合には、次のように ALTER TABLE ステートメント で「SYSTEM_VERSIONING = OFF」に設定します。



テンポラル テーブルを無効化すると、オブジェクト エクスプローラーでは、次のように表示 されます。



テンポラル テーブルの場合は「**システム バージョン管理**」と表示されていましたが、通常の テーブルに戻って、履歴テーブルに関しても、通常のテーブルに変換される(履歴テーブルも 残る)形になります。

3. このように通常テーブルに戻った後は、通常と同じように **DROP TABLE** ステートメントで 削除することができるので、不要な場合は削除しておいてください。 DROP TABLE ProductsHistory DROP TABLE Products



以上のように、テンポラル テーブルを利用すれば、テーブルに対する更新履歴を自動的に履歴テー ブルに保存、および履歴データを簡単に参照することができるので、操作ミスが発生した場合のデ ータ修復や、コンプライアンスにおける監査証跡(Audit Log:監査ログ)、データ分析における SCD (緩やかに変化するディメンション)として利用することができます。

その他、テンポラル テーブルについては、オンライン ブックの以下のトピックがお勧めです。

Temporal Tables

https://msdn.microsoft.com/en-us/library/dn935015.aspx

 Primary and Foreign Key	Temporal Tables
Constraints Unique Constraints and Check	SQL Server 2016
Constraints Table Column Properties (SQL	Updated: January 6, 2016
Server Management Studio)	Applies To: SQL Server (starting with 2016 CTP3)
Temporal Tables Getting Started with System-	Applies to: SQL Server (SQL Server 2016 Community Technology Preview 3.2 (CTP 3.2) through current version).
Versioned Temporal Tables Temporal Table System Consistency Checks Partitioning with Temporal Tables Temporal Table Considerations red Limitations	SQL Server 2016 introduces support for system-versioned temporal tables as a database feature that brings built-in support for providing information about data stored in the table at any point in time rather than only the data that is correct at the current moment in time. Temporal is a database feature that was introduced in ANSI SQL 2011 and is now supported in SQL Server 2016.
 Temporal Table Security Manage Historical Data with System-Versioned Temporal Tables System-Versioned Temporal Tables with Memory-Optimized Tables Temporal Table Metadata Views and Functions 	 Quick Start Download CTP: To download SQL Server 2016 Community Technology Preview 3.2 (CTP 3.2), go to Evaluation Center. Create Azure Virtual Machine: Have an Azure account? Spin up a virtual machine with SQL Server 2016 already installed. Download Adventure Works sample database: To get started with Temporal Tables download AdventureWorks database for SQL Server 2016 CTP3 with script samples and follow the instructions in the folder 'Temporal' Getting started: Getting Started with System-Versioned Temporal Tables System-Versioned Temporal Tables with Memory-Optimized Tables Temporal Table Usage Scenarios

STEP 4. 注目の新機能を試してみよう

この STEP では、「SQL Server R Services」や「JSON 対応」、「Strech Database」、「Azure バックアップ」、「PolyBase」など、SQL Server 2016 CTP 3.2 で提供された注目の新機能を試してみましょう。

この STEP では、次のことを学習します。

- ✓ SQL Server R Services
- ✓ JSON 対応(FOR JSON、OPENJSON、JSON_VALUE など)
- ✓ Strech Database
- ✓ Azure バックアップ(Backup To URL)の性能向上
- ✓ PolyBase で Hadoop アクセス (Hortonworks や HDInsight)

4.1 SQL Server R Services (R 統合)

SQL Server R Servicesは、Transact-SQL ステートメントを利用して、**R スクリプト/ライブ ラリ**を実行することができる、R 統合機能です。



R 言語は、オープンソースの統計解析向けのプログラミング言語で、これを <u>SQL Server</u> 環境に 統合(ビルトイン)して、実行できるようにしたのが「**SQL Server R Services**」です。この R 統合機能は、内部的には「**Revolution R Enterprise 7.5.0**」を利用していて、「**Revolution R**」 は、オープンソースの R の性能的な欠点を補うために、Enterprise 向けの性能強化を計ったプラ ットフォームです (Revolution R は、2015年にマイクロソフトが買収し、SQL Server R Services として SQL Server 2016 に統合、および 2016年1月1日以降は「**Microsoft R Server**」と いう名称に変更して、Linux 環境でも利用できる単体製品としても販売されています)。

SQL Server 2016 CTP 3.2 で、SQL Server R Services を利用するための流れは、次のとおりで す(CTP 3.2 では、SQL Server のインストーラーに統合されていませんが、RTM 版ではインス トーラーに統合される予定です)。

- SQL Server のインストール時に「Advanced Analytics Extensions」をチェック
- Revolution R Open 3.2.2 for Revolution R Enterprise 7.5.0 のインストール http://go.microsoft.com/fwlink/?LinkId=626650
- Revolution R Enterprise 7.5 のインストール http://go.microsoft.com/fwlink/?LinkId=626652
- sp_configure で「external scripts enabled」を「1」に変更
- Revolution R Open 3.2.2 の RegisterRExt.exe /install を実行

◆ SQL Server R Services のインストール

それでは、SQL Server R Services をインストールしてみましょう。

 まずは、SQL Server 2016 CTP 3.2 のインストール時に、次のように [機能の選択] ページ で、[データベース エンジン サービス] の [Advanced Analytics Extensions] をチェッ クします。

髕 SQL Server 2016 CTP3.1 セットアップ				-		×
機能の選択 インストールする Evaluation 機能を	「違沢します。					
プロダクトキー ライセンス条項 グローバルルール Microsoft Update 製品の更新プログラム セットアップファイルのインストール インストールルール セットアップロール 機能の選択 機能ルール インスタンスの構成 サーバーの機成 データベースエンジンの構成 機能構成ルール インストールの準備完了 インストールの進行状況 完了	機能(E): インスタンス機能 ダニタハース エンジン サービス SQL Server レブリケーション メロシーン 人々へれてくろいろいろいろいろいろいろいろいろいろいろいろいろいろいろいろののしてないのののしていたいちいたいろいろいろいろいろいろいたいろいたいろいろいろいろいろいろいろいろ	ns セマンティック抽出 ービス vices アドイン ン ン Program Files¥Mi C*Program Files¥Mi	機能の説明: SQL Server インスタンスのイ 成と操作が、他の SQL Server 遅されています。SQL Server 又を、同じコンピューターでサイ 選択した機能に必要なコンパ インストール済み: 	ンスタンス er インスタ の複数の ド バイサ・ - ネント(E 2.0 work 3.5	機能は、構 ソスとは分 イソスタン イドで操作 2): と に	
		< 戻る(<u>B</u>)	次へ(<u>N</u>) > キャンセ	V	ヘルプ	

この Advanced Analytics Extensions を選択した場合は、次の [機能ルール] ページで、 次のように警告が表示されますが(追加の手順が必要になるという主旨)、これは無視してイ ンストールを継続して大丈夫です。

 機能ルール イソストール処理がブロックされるかどうかを判断するために、ルールを実行しています。詳細については、[ヘルプ]をクリックしてください。 プロダクトキ- うイセンス条項 グローバル ルール	髋 SQL Server 2016 CTP3.1 セットアップ		-		\times	
インストール処理がブロックされるかどうかを判断するために、ルールを実行しています。詳細については、[ハルブ]をクリックしてください。 プロダクトキー ライセンス条項 グローバル ルール Microsoft Update 製品の更新プログラム セットアップファイルのインストール インストールルール ビットアップファイルのインストール インストールルール セットアップフロール 機能の選択 機能の選択 機能の選択 機能の選択 使用レール インスやフリンの構成 サーバーの構成 データベースエンジジンの構成 機能構成ルール インストールの学権常工	機能ルール					
プロダクトキー 操作が完了しました。成功 3、失敗 0、答告 1、スキップ 0。 ライセンス条項 アローバル ルール Microsoft Update 詳細の非表示(S) <	インストール処理がブロックされるかど	うかを判断するために、ルールを実行しています。詳細については、[ヘルプ]をクリックしてください。				
ライセンス条項 グローバル ルール Microsoft Update 詳細の非表示(S) <<	プロダクト キー	操作が完了しました。 成功 3、失敗 0、警告 1、スキップ0。				
グローバル ルール Microsoft Update 再実行(D) 製品の更新プログラム 詳細の非表示(S) <	ライセンス条項					
Microsoft Update 再実行(D) 製品の更新プログラム 詳細の非表示(S) <	グローバル ルール					
製品の更新プログラム セットアップファイルのインストール インストール ルール ビットアップロール 検能の選択 機能の選択 ペ Microsoft .NET Framework 3.5 Service Pack 1 が必要です ム ム な Advanced Analytics Extensions を使用するには、インストール後の手順を行う必要が… 留金 ペ Mik環成ルール インスタンスの構成 サーバーの構成 データベース エンジンの構成 機能構成ルール インストールの準備完了	Microsoft Update	詳細の非表示(<u>S</u>) <<		再実行	行(<u>R</u>)	
セットアップ ファイルのインストール インストール ルール セットアップ ロール 機能の選択 機能の選択 機能の選択 インスタンスの構成 サーバーの構成 データバース エンジンの構成 機能構成ルール インストールの準備完 了	製品の更新プログラム	<u>詳細レポートの表示(V)</u>				
インストール ルール 大阪 セットアップ ロール 大阪 セットアップ ロール 秋節の Visual Studio 2010 のインスタンスは、更新する必要があります。 会協 機能の選択 Microsoft .NET Framework 3.5 Service Pack 1 が必要です 会協 インスタンスの構成 Advanced Analytics Extensions を使用するには、インストール後の手順を行う必要が 登告 ゲーバーの構成 ど格2919355 インストール 会協 パール チェックの結果 ルール チェックの結果	セットアップ ファイルのインストール		11.45			
ゼッドパック ロール	インストール ルール		状態			
 (図目の増水	セットアップロール	以前の Visual Studio 2010 のインスタンスは、更新する必要があります。	合植			
	彼尼の選択	Wicrosoft .NET Framework 3.5 Service Pack 1 が必要です	合格	-		
	彼能ルール	▲ Advanced Analytics Extensions を使用するには、インストール後の手順を行う必要が…	<u> </u>		_	
データベース エンジンの構成 データベース エンジンの構成 機能構成ルール インストールの準備完了	サーバーの構成	KB2919355 インストール	<u>合格</u>		_	
現記構成ルール インストールの準備完了	データベースエンジンの構成					
インストールの準備完て	機能構成ルール	ルール チェックの結果				×
	インストールの準備完了					
インストールの進行状況 / ルール *Advanced Analytics Extensions を使用するには、インストール後の手 順を行う心要があります。* かに等率が生きれるりました。	インストールの進行状況	ルール "Advanced Analytics Extensions を 順を行う必要があります。" から警告が生成さ	使用するに	:は、インフ	ィトール後	の手
完了	完了					
Advanced Analytics Extensions の機能を使用するには、SQL Server の設定 を完了してから、インストール後の手順をいくつか行う必要があります。 次のリンクに		Advanced Analytics Extensions の機能を を完了してから、インストール後の手順をいくつ。	使用するに か行う必要	に、SQL SQL S	Server の す。次のリ	設定 ンクに
記載されている手順を行ってください。		記載されている手順を行ってください。 http://go.microsoft.com/fuligk/21ipl/d	-626645 (-1+ C+-1 -	t_£
http://go.microsoft.com/wilnk/?Linkid=620043 (ユビータ Sicila Ctrl ギーを 押しながら C キーを押します)		ーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーー	=020043 (.	10-9.50	cia etti -	1-2
OK						OK

残りの項目は、通常の SQL Server のインストールと同様進めます。

2. SQL Server 2016 CTP 3.2 のインストールが完了した後は、「Revolution R Open 3.2.2 for Revolution R Enterprise 7.5.0」をインストールします。これは、次の URL からダウ

ンロードすることができます。

Revolution R Open 3.2.2 for Revolution R Enterprise 7.5.0 のダウンロード http://go.microsoft.com/fwlink/?LinkId=626650



ダウンロード後は、「**RRO-3.2.2-for-RRE-7.5.0-Windows.exe**」をダブルクリックして、 インストールを行います。



3. 次に、「Revolution R Enterprise 7.5」をインストールします。これは、次の URL からダ ウンロードすることができます。

Revolution R Enterprise 7.5 のダウンロード http://go.microsoft.com/fwlink/?LinkId=626652

Revolution R Ente	erprise 7.5.0 (RRE-7.5.0)		
Language: English	Download		
Revolution R Enterprise class, big data big analy analytics, Revolution R E	Choose the download you wa	ant	\otimes
across complex and dive 2015 CTP 3, clusters and	File Name	Size	
download contains a fre	Revolution-R-Enterprise-Node-SQL-7.5.0-Windows.exe	81.3 MB	Download Summary:
+ Details	Revolution-R-Enterprise-SQL-7.5.0-Windows.exe	463.2 MB	SQL-7.5.0-Windows.exe
System Requirements			
Install Instructions			
			Total Size: 463.2 MB
			Next

ダウンロード後は、「**Revolution-R-Enterprise-SQL-7.5.0-Windows.exe**」をダブルクリックして、インストールを行います。

Revolution R Enterprise 7.5.0 Setup		×	Revolution R Enterprise 7.5.0 Setup		×
	Velcome to the Prerequisites Wizard The setup has determined that some of the prerequisites needed to run this program are missing. This wizard automates installing those prerequisites. Click Next to continue to the list of prerequisites.		Revolution K Enterprise 7.5.0 Setup Prerequisites These programs are needed for the a next to a prerequisite to select it for i Microsoft Visual C++ 2013 Redistrium. Check for previous installations Visual C++ 2008 SP1 Redistributa Visual Studio 2008 Isolated Shell SP1 Visual Studio 2008 Isolated Shell SP1 Microsoft MPI v6	pplication to run. Click on the check be stall or to skip it. Version Required: 12.0.21005.1 or high Required: any. Found: nothing. Required: any. Found: nothing.	x Acton Instal Instal Instal Instal Instal Instal Instal
-			Press the Next button to install the prereq	uisites.	
	Back Next Finish Cance		Back	Next Finish	Cancel

Revolution R には、Visual C++ 2013 Redistributable や Visual C++ 2008 Redistributable、Microsoft MPI などが必須コンポーネントになるので、次のようにセットアップが促されます(それぞれ使用許諾契約書の内容を確認した上で、I agree をチェックして、Installをクリックします。設定項目は、すべて既定値のままで大丈夫です)。

🕼 Microsoft Visual C++ 2013 Redistributable (x64) - 12.0.210 – 🗌 🗙	🕼 Microsoft Visual C++ 2013 Redistributable (x64) - 12.0.210 — 🗌 🗙
Microsoft Visual C++ 2013	Microsoft Visual C++ 2013
Redistributable (x64) - 12.0.21005	Redistributable (x64) - 12.0.21005
MICROSOFT SOFTWARE LICENSE TERMS	Setup Successful
MICROSOFT VISUAL C++ REDISTRIBUTABLE FOR VISUAL STUDIO 2013	
These license terms are an agreement between Microsoft Corporation (or based on where you live one of its affiliates) and you. Please read them	
<u>install</u> ose	Close

SQL Server 2016 CTP 3.2 自習書 No.1 新機能ダイジェスト

Microsoft Visual C++ 2008	Redistributable Setup - 🗆 🗙	🕼 Microsoft Visual C++ 2008 Redistributable Setup — 🗆 🗙
Welcome to Microsoft Visu Redistributable Setup	al C++ 2008	License Terms
This wizard will guide you throug	th the installation process.	Be sure to carefully read and understand all the rights and restrictions described in the license terms. You must accept the license terms before you can install the software. MICROSOFT SOFTWARE LICENSE TERMS MICROSOFT VISUAL C++ 2008 RUNTIME LIBRARIES (X86, IA64 AND X64), SERVICE PACK 1 These license terms are an agreement between Microsoft Corporation (or based on where named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft - updates. - updates. Print Press the Page Down key to see more text. Image: Additional Accept the license terms Qancel
Software Update Microsoft Visu	al Studio 2008 Shell SP1 Installation Wizard $ imes$	Software Update Microsoft Visual Studio 2008 Shell SP1 Installation Wizard
2008	Welcome to the Microsoft Visual Studio 2008 Shell SP1 software update.	Microsoft Software License Terms Please accept the license terms to continue.
tudio:	Products affected by the software update: • Web Designer Dore • Web Designer Core SP1 Patch • VS Shell	MICROSOFT SOFTWARE SUPPLEMENTAL LICENSE TERMS
Microsoft	Get <u>more information</u> about this update.	Microsoft Corporation (or based on where you live, one of its affiliates) licenses this supplement to you. If you are licensed to use Microsoft Visual Studio 2008 software ^(*) , you may use this supplement. You may not use it if you do not have a license for the software. You may use <u>Print License Terms</u> <u>Save License Terms</u> <u>[] have read and gocept the license terms</u>] <u>[] would like to improve the installation experience by sending anonymous feedback</u> .
		Please read the Microsoft Oustomer Experience Improvement Program and privacy policy.
	< Back Next >	< <u>B</u> ack <u>Next</u> <u>C</u> ancel
时 Microsoft MPI (6.0.12436.10	0) Setup — 🗆 🗙	岁 Microsoft MPI (6.0.12436.10) Setup - 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇 〇
	Welcome to the Microsoft MPI (6.0.12436.10) Setup Wizard	End-User License Agreement Please read the following license agreement carefully
	The Setup Wizard will install Microsoft MPI (6.0. 12436. 10) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.	MICROSOFT SOFTWARE LICENSE TERMS
		These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to the software named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft updates.
		I accept the terms in the License Agreement;
	Back Next Cancel	Print Back Next Cancel

必須コンポーネントのインストールが完了した後は、次のように Revolution R のセットアッ プが始まります(こちらもすべて既定値でのインストールで大丈夫です)。

SQL Server 2016 CTP 3.2 自習書 No.1 新機能ダイジェスト

邊 Revolution R Enterprise 7.5.0 Setup	K 😥 Revolution R Enterprise 7.5.0 Setup					
REVOLUTION ANALYTICS Welcome to the Revolution R	End-User License Agreement Please read the following license agreement carefully					
The Setup Wizard will install Revolution R Enterprise 7.5.0 on your computer. Click "Next" to continue or "Cancel" to exit the Setup Wizard.	REVOLUTION SOFTWARE LICENSE TERMS Revolution R Enterprise for SQL Server 2016 Community Technology Preview (CTP) 3					
< Back Next > Cancel	These license terms are an agreement between Revolution Analytics,					
Revolution R Enterprise 7.5.0 Setup	Revolution R Enterprise 7.5.0 Setup — X Select Installation Folder This is the folder where Revolution R Enterprise 7.5.0 will be installed.					
RevoScaleR operations are enhanced by the recommended packages rpart and lattice. Load these packages by default?	To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse". Eolder: C:¥Program Files¥R-Enterprise-7.5¥ V Brgwse Space required: 220MB Space available: 89GB					
Advanced Installer < Back Next > Cancel	Advanced Installer 					
Revolution R Enterprise 7.5.0 Setup	Revolution R Enterprise 7.5.0 Setup × REVOLUTION ANALYTICS Completing the Revolution R Enterprise 7.5.0 Setup Wizard					
Click "Install" to begin the installation. If you want to review or change any of your installation settings, click "Back". Click "Cancel" to exit the wizard.	Click the "Finish" button to exit the Setup Wizard. To launch Revolution R Enterprise 7.5.0, go to the Start menu and navigate as follows: All Programs Revolution R Enterprise 7.5 Revolution R Enterprise 7.5 (64)					
< Back Install Cancel	< Back Binish Cancel					

 Revolution R のインストールが完了した後は、SQL Server 2016 CTP 3.2 の Management Studio を起動して、クエリ エディターで、sp_configure を利用して「external scripts enabled」を「1」に変更します。

5. 次に、Revolution R Open 3.2.2 の「**RegisterRExt.exe /install**」を実行します。これは、 コマンド プロンプトから、次のように実行します。

%programfiles%¥RR0¥RR0-3.2.2-for-RRE-7.5.0¥R-3.2.2¥library¥RevoScaleR¥rxLibs¥x64¥RegisterRExt.exe″/install

The set of the se

最後の行に「R extensibility installed successfully」と表示されれば、インストールが完 了です。これで、SQL Server に R が統合(ビルトイン)されて、クエリ エディターから R スクリプトを実行できるようになります(スクリプトの実行には、後述の sp_execute_external_script システム ストアド プロシージャを利用します)。

→ R スクリプトの実行 ~sp_execute_external_script~

インストールが完了した後は、クエリ エディターを利用して、R スクリプトを実行してみましょう。これは、**sp_execute_external_script** システム ストアド プロシージャを利用して、次の ように記述します。

```
EXEC sp_execute_external_script
@language = N'R'
@script = N'実行したい R スクリプト'
@input_data_1 = N'R で処理したいデータ'
@input_data_1_name = N' input_data_1 に対して設定する名前'
@output_data_1_name = N'R スクリプトで処理した結果'
WITH RESULT SETS ((列名 データ型 null/not null, ・・・));
```

@language で「R」を指定、@script に「R スクリプト」を記述することで、任意の R スクリプトを実行できるようになります。また、R スクリプトに与えたい入力データは「@input_data_1」および「@input_data_1_name」で指定し、R スクリプトで処理した結果(出力データ)は「@output_data_1_name」で指定することができます。ただし、output に指定できるのは、R のデータ フレーム形式のデータ(data.frame 関数で変換可能なもの)である必要があります。

Let's Try

それでは、これを試してみましょう。ここでは、R 言語に含まれる、サンプルのデータセットである「iris」(アヤメのがく片と花弁の長さと幅が格納されたデータ)を取得してみます。このデータは、R のツールである「R コンソール」で「iris」と入力することで、次のように内容を確認できます(R コンソールは、[スタート] メニューの [すべてのアプリ] → [PRO-for-RRE] の [PRO for RRE 7.5.0] をクリックして起動することができます)。

R RGui	- Revolution R E	nterprise version 7.5 (6	4-bit)				_		×			
ファイル	編集 閲覧	その他 パッケージ	ウインドウ ヘルプ									
					R コンソールで	-						
		⊿ 🖳 🖻		R	言語に含まれて							
R Cor	nsole			[Iris]	テータセットの	+身を表示			×			
									^			
> iris	B Townshi	Course Mildels Doo		1.114.445								
1 26	epai.Length	Sepai.width Pet	ai.Length Peta	1.Width	species							
2	4.0	3.5	1.4	0.2	setosa							
2	4.9	3.0	1.7	0.2	setosa							
4	1.7	3.2	1.5	0.2	setosa							
5	5.0	3.6	1.5	0.2	setosa							
6	5.4	3.9	1 7	0.4	setosa							
7	4.6	3.4	1.4	0.3	setosa							
8	5.0	3.4	1.5	0.2	setosa							
9	4.4	2.9	1.4	0.2	setosa S							
10	4.9	3.1	1.5	0.1	setosa	あや あや	あやめの種類					
11	5.4	3.7	1.5	0.2	setosa	(setosa,	(setosa、versicolor、					
12	4.8	3.4	1.6	0.2	setosa	virginica	virginica の 3種類)					
13	4.8	13.0	1.4	40.1	setosa							
14	4.3	3.0	1.1	0.1	setosa							
15	5.8	4.0	1.2	0.2	setosa							
16	5.7		1.5	V	setosa							
17	5.4	がく片(Sepal)	1.3 花	弁(Petal)	setosa							
18	5.1	の長さと幅	1.4 0	の長さと幅	setosa							
19	5.7	3.8	1.7	0.3	setosa							
20	5.1	3.8	1.5	0.3	setosa							
21	5.4	3.4	1.7	0.2	setosa							
22	5.1	3.7	1.5	0.4	setosa							
23	4.6	3.6	1.0	0.2	setosa							
24	5.1	3.3	1.7	0.5	setosa							
25	4.8	3.4	1.9	0.2	setosa							
									. *			
<									d			

このデータは、植物生物学者である Edgar Anderson 氏が 50 の花 (アヤメ: iris) から収集した 「**がく片**」(Sepal) と「花弁」(Petal) の長さと幅になっています。Species 列には、アヤメの 種類(品種) が格納されています。

1. このデータを sp_execute_external_script システム ストアド プロシージャを利用して 取得するには、次のように記述します (Management Studio のクエリ エディターで記述)。

```
-- iris データセットを取得
EXEC sp_execute_external_script
@language = N'R'
,@script = N'ret <- iris;'
,@input_data_1 = N''</pre>
```





@script(R スクリプトを記述する場所)には、「ret <- iris;」と記述して、iris データセットを「ret」変数(データ フレーム)に代入しています(R 言語では <- で値の代入ができます)。@output_data_1_name(出力データの名前)には、この「ret」データ フレームを指定することで、SQL Server 側で結果を取得できるようになります。

@input_data_1 には、R スクリプトに与えたい入力データを指定しますが、今回は入力データを利用しないので、空文字を指定しています(この引数は、必須の引数になるので、値を 指定しない場合は空文字を与えるようにします)。

WITH RESULT SETS では、出力結果に対して、SQL Server 側で利用する際の列名とデー 夕型、NULL の可否(NULL または NOT NULL を指定。省略時は NULL)を記述します。今 回の iris データは、がく片の長さと幅(Sepal.Length と Sepal.Width)、花弁の長さと幅 (Petal.Length と Petal.Width)、アヤメの種類(Species)の5つの列が返るので、WITH RESULT SETS に5個分の列定義を記述しています。もし、結果セットの列数と、WITH RESULT SETS で定義した列数が異なる場合には、次のようにエラーが返されます。
	WITH RESULT SETS と結果の列数が異なる場合 EEXEC sp_execute_external_script @language = N'R' ,@script = N'ret <- iris;' ,@input_data_1 = N' @output_data_1_name = N'ret' WITH RESULT SETS ((coll float, col2 float))
100	% - <
	結果 📠 メッセージ
	メッセージ 11537、レベル 16、状態 2、行 47 EXECUTE ステートメントが失敗しました。WITH RESULT SETS 句で結果セット番号 1 に対して指定された列は 2 個ですが、実行時に送信された列は 5 個でした。

このように、SQL Server R Services を利用すると、R スクリプトを実行して、その実行結果を SQL Server 側で処理できるようになるので、大変便利です。

◆ SQL Server のデータを input に与える ~@input_data_1~

次に、SQL Server のデータベース内のデータを、入力値として R スクリプトに与えてみましょう。これを行うには、@input_data_1 引数に SELECT ステートメントを記述するようにします。ここでは、動的データ マスクのところで利用した「maskTestDB」データベースの「maskTest1」テーブルのデータで試してみましょう。

■SELECT * FROM maskTestDBmaskTest1								
100% - <								
	colA	colB	colC	colD	colE			
1	1	ААААА	111	2015-10-30 00:00:00.000	NULL			
2	2	BBBBB	222	2015-11-30 00:00:00.000	NULL			
3	3	00000	333	2015-11-30 00:00:00.000	ccc@test.local			

1. SQL Server のデータを入力値として与えるには、次のように記述します。

```
EXEC sp_execute_external_script
    @language = N'R'
    ,@script = N'ret <- input1;'
    ,@input_data_1 = N'SELECT colA, colB FROM maskTestDB..maskTest1'
    ,@input_data_1_name = N'input1'
    ,@output_data_1_name = N'ret'
WITH RESULT SETS ( (colA int, colB varchar(200) ) )</pre>
```

```
EXEC sp_execute_external_script
                @Tanguage = N'R'
,@script = N'ret <-
                                                   input1;'
              ,escript - N ret <- input,
,@input_data_1 = N'SELECT colA, colB FROM maskTestDB..maskTest1'
,@input_data_1_name = N'input1'
,@output_data_1_name = N'ret'
WITH RESULT SETS ( (colA int, colB varchar(200) ) )
100 %
🛄 結果 📑 メッセージ
          colA colB
         1
                     AAAAA
 1
          2
                     BBBBB
2
          3
                     ccccc
3
```

@input_data_1 引数に SELECT ステートメントを記述して、@input_data_1_name 引数に SELECT ステートメントの結果に対して任意の名前 (ここでは input1 と指定)を設 定します。ここで指定した名前で、R スクリプトを処理できるようになります (もし、 @input_data_1_name 引数を省略した場合は、InputDataSet という名前が補われま す)。

@script(R スクリプトを記述する場所)には、「ret <- input1;」と記述しているので、 SELECT ステートメントの結果をそのまま「ret」データ フレームに格納して、それを WITH RESULT SETS で定義した列定義に代入しています。

このように、SQL Server のデータを R スクリプトに与えることも簡単に行うことができます。

2. 次に、R スクリプトの関数を利用して、入力データを処理してみましょう。ここでは、sum 関数を利用して、colA 列の合計を計算してみます。



手順1 のスクリプトから変更したのは @script に記述した R スクリプトと WITH RESULT SETS で指定した列定義のみです。@script には、1 行目に「sum1 <sum(input1[,1]);」と記述して、input1 の 1 列目のデータ (SQL Server から取得した colA 列のデータ) に対して R 言語の sum 関数を利用して処理し、その結果を sum1 変 数に代入しています。

次に、「ret <- data.frame(sum1);」で、sum1 変数を data.frame 関数でデータ フレームに変換して、その結果を ret 変数(データ フレーム)に格納しています。この結果は、 1列分なので、WITH RESULT SETS で指定する列定義も 1列分(col1 int のみ)にしています。 このように、R スクリプトは、「;」で行の区切りを付けて、複数行実行することができます。 また、スクリプト内では R 言語の関数や、ライブラリを利用することができるので、さまざ まな統計解析処理を行っていくことができます。

なお、上の R スクリプトは、「**ret <- data.frame(sum(input1**[**1**]**));**」のように 1 行で 記述することもできます。

その他、SQL Server R Services については、オンライン ブックの以下のトピックがお勧めです。

SQL Server R Services

https://msdn.microsoft.com/en-us/library/mt604845.aspx



4.2 JSON 対応 (FOR JSON、OPENJSON、JSON_VALUE など)

SQL Server 2016 では、**JSON**(JavaScript Object Notation)に対応して、SQL Server 上のテ ーブル データを JSON 形式で出力したり、JSON データを SQL Server に取り込んだりできる ようになりました。具体的には、次の機能が提供されています。

● FOR JSON で SQL Server 上のテーブル データを JSON 形式で出力

JSON 形式 SELECT 商品 FOR JSON AU	∜で出力 コード,商品名 FROM NorthwindJ商品 UTO	テーブル データを
100 % 🔹 <		JSON 形式で出力
🔝 結果 🚮 メッセージ	٣	
JSON_F52E	E2B61-18A1-11d1-B105-00805F49916B	
1 【《商品コー	+*´´:1,´´商品名´´:´'果汁100% オレンジ´`},{´´商品コード´´:2,´´商	萄品名″:″果汁100% グレープ″},[″商品
2 り~}.(~商品:	コード~:69,~商品名~:~乾燥バナナ~}!(~商品コード~:70,~商。	品名~:″乾燥アッフル″}、″商品コード″:

OPENJSON 関数で JSON データの取り込み



• **ISJSON** 関数で JSON データかどうかの判別

JSON データを SQL Server 上に格納 (ISJSON 関数を CHECK 制約で利用)

• JSON_VALUE と JSON_QUERY で JSON データをクエリ



➡ FOR JSON で SQL Server 上のテーブル データを JSON 形式で出力

FOR JSON を利用すれば、SQL Server 上のテーブル データを JSON 形式で出力できます。これも試してみましょう。これを行うには、次のように SELECT ステートメントの末尾に「FOR JSON AUTO」と付けるだけです。

-- テーブル データを JSON 形式で出力
 SELECT 商品コード, 商品名 FROM NorthwindJ..商品
 FOR JSON AUTO



全体が [] で囲まれた配列として出力されて、各行のデータ(1行分のデータ)が {} で囲まれて、 それぞれ「"列名 1":"データ 1","列名 2":"データ 2"」(データが数値の場合は " なし)というペア になって出力されています。上の例では、「商品コード」と「商品名」列を指定しているので、{"商 品コード":1, "商品名":"果汁 100%オレンジ"} という形で 1行分のデータが出力されています。 なお、出力データ量が多い場合(出力する JSON 文字列が長くなる場合)には、上の結果のように 複数行に分かれて出力されます。

♦ OPENJSON 関数で JSON データの取り込み

OPENJSON は、JSON データを取り込むことができる関数です。これも試してみましょう。まずは、次のように実行してみます。

```
SELECT * FROM
OPENJSON (N' {"key1": "test1", "key2": 999}')
```

```
-- OPENJSON
SELECT * FROM
OPENJSON(N'{"key1": "test1", "key2": 999}')
100 % - <
副 結果 日 メッセージ
key value type
1 key1 test1 1
2 key2 999 2
```

各キー(key1、key2) とそれぞれの値(test1、999)が1行ずつ表示されていることを確認で きます。type 列は、キー値の種類が返り、「1」は文字列(key1 は test1 という文字データなの で1)、「2」は数値(key2 は 999 という数値データなので2)になります。 次に、OPENJSON 関数で WITH キーワードを追加して、各キーの値を列として取得してみます (前の例は、各キー値を "行" として取得しましたが、今度は "列" として取得します)。

```
-- WITH でキーを指定

SELECT * FROM

OPENJSON(N'["key1": "test1", "key2": 999]')

WITH (

key1 varchar(20),

key2 int

)

100 % - <

■ 結果 ■ メッセージ

key1 key2

1 test1 999
```

このように、WITH キーワードでは、キーの名前(key1、key2)とデータ型を指定することで、 列として取得することもできます。

次に、OPENJSON 関数に、文字列の変数を与えてみます。

JSON データを @jsonstr という文字列変数に格納して、それを OPENJSON 関数で取得して います。この JSON データは、arr1 というキーに対して、[](大カッコ)で囲んで配列を構成し ているので、type が「4」(配列を表す値)で返ってきています。なお、JSON では [要素 1, 要 素 2, ・・・] という形で、配列を表現できます。これに対して、{ }(中カッコ)で囲むものは、{" **キー名 1**":"値 1", "キー名 2":"値 2"} という形で、キー名とその値が「:」で区切られてペアになり、これはオブジェクトと呼ばれています。

OPENJSON 関数で Path を指定する ~OPENJSON(jsonstr, path)~

OPENJSON 関数では、第2引数に JSON データ内のパス(階層構造がある場合のパス)を記述 することができます。これも試してみましょう。

<pre>DECLARE @jsonstr nvarchar(max) = N' { "arr1": [</pre>
{"key1": "test1", "key2": 999}
,{"key1": "test2", "key2": 777}
]]'
SELECT * FROM OPENJSON(@isonstr. '\$.arr1')

t.	OPE DECLAR たカッコ内の シマで区切ら れた配列	NJSON で Path を指定 E @jsonstr nvarchar(max) = N'{	″, ″key ″, ″key	2″: 999] 2″: 777]	\leq	中カッコ : でキ 区切られた]で囲まれて -名と値が ミオブジェクト
	SELECT OPENJ	* FROM SON(@jsonstr, '\$.arr1')	パスを指定	Ē			
100	% ▼ <						
	結果 🚹	メッセージ					
	key	value	type				
1	0	{~key1~: ~test1~, ~key2~: 999}	5 🥆		type O	5 (+	
2	1	{~key1~: ~test2~, ~key2~: 777}	5	7:	ブジェクト	を表す	

OPENJSON 関数の第2引数に「**\$.arr1**」と指定することで、「**arr1**」の値である[](大カッコ) で囲まれた配列データを取得できています。この配列の中には、中カッコで囲まれたオブジェクト が2つがあるので、2行に分かれて結果を取得できています。**type**列の「**5**」はオブジェクトを 表す値になります。

オブジェクトに対しては、WITH キーワードでキー名を指定することで、キー値を "列" として出 力することもできるので、次のように取得することもできます。

	WITH でキーを指定 DECLARE @jsonstr nvarchar(max) = N'[″arr1": [{						
	SELECT * FROM OPENJSON(@jsonstr, '\$.arr1') WITH (key1 varchar(20), key2 int) WITH キーワードで						
100	% ▼ < キー名を指定						
	結果 💼 メッセージ						
	key1 key2						
1	test1 999 各キーの値を						
2	test2 777 列として取得できる						

このように、OPENJSON 関数を利用すれば、JSON データを簡単に取得することができます。

◆ ISJSON 関数で JSON データかどうかの判別

ISJSON は、JSON データかどうかをチェックできる関数です。JSON データであれば 1、そう でなければ 0 を返すことができます。これも試してみましょう。

```
SELECT ISJSON(N' {"key1": "test1", "key2": 999}')
```



与えた文字列は JSON データなので、1 が返ることを確認できます。次に、JSON データではない値を与えてみます。

```
SELECT ISJSON (N' aaa' )
```

	SELECT ISJSON(N'aaa')							
100	% - <							
	結果 🔓 メッセージ							
	(列名なし)							
1	0							

今度は **0** が返ることを確認できます。このように、ISJSON 関数を利用すれば、JSON データか どうかをチェックできるので便利です。

→ JSON データを SQL Server 上に格納(ISJSON 関数を CHECK 制約で利用)

<u>SQL Server</u>内に JSON データを格納したい場合には、**ISJSON** 関数を利用して **CHECK 制約** を作成するのがお勧めです。これで、JSON データのみを確実に格納できるようになります。これ も試してみましょう。

```
-- ISJSON 関数で CHECK 制約を設定
CREATE TABLE jsonTest1
( colA int PRIMARY KEY
, colB nvarchar(max) CONSTRAINT chkJSON CHECK ( ISJSON(colB) > 0 )
)
```

colB 列に対して、CHECK 制約で ISJSON 関数を利用することで、colB 列には、JSON データのみが格納できるようになります。

たとえば、次のように JSON 形式ではない値(aaa)を INSERT しようとすると、次のように CHECK 制約違反が発生して、データが INSERT されるのを防げるようになります。



次に、colB 列に JSON データを格納してみましょう。

INSERT INTO jsonTest1 VALUES(1, N' {"key1": "test1", "key2": 999}')
INSERT INTO jsonTest1 VALUES(2, N' {"key1": "test2", "key2": 777}')
SELECT * FROM jsonTest1



JSON データであれば、何の問題もなく INSERT することができます。このように、INSERT したデータは、後述の JSON_VALUE や JSON_QUERY 関数を利用して、簡単に取り出すことができます。

→ JSON_VALUE と JSON_QUERY で JSON データをクエリ

JSON_VALUE と JSON_QUERY は、JSON データ(特定のキーやオブジェクト)をクエリでき る関数です。これも試してみましょう。

```
SELECT JSON_VALUE(colB, '$.key1') AS key1
, JSON_VALUE(colB, '$.key2') AS key2
FROM jsonTest1
```

```
SELECT JSON_VALUE(coIB, '$.key1') AS key1
,JSON_VALUE(coIB, '$.key2') AS key2
FROM jsonTest1
100 % - <

副 結果 副 メッセージ

key1 key2

1 test1 999

2 test2 777
```

JSON_VALUE 関数では、第1引数に JSON 文字列(ここでは colB 列に格納されたデータ)、

第2引数にキーへのパスを指定することで、該当キーの値(value)を取得することができます。 ここでは「**\$.key1**」と「**\$.key2**」を指定しているので、各キーの値(**test1** や **999** など)を取 得することができています。

このように、SQL Server 内に格納した JSON データでも、JSON_VALUE 関数を利用すれば、 簡単に取得することができます。

JSON_QUERY 関数は、オブジェクトのパスを指定して、該当オブジェクトを取得することができます。これは次のように試すことができます。

```
TRUNCATE TABLE jsonTest1
INSERT INTO jsonTest1 VALUES(1, ' { "obj1": {"key1": "test1", "key2": 999} }')
INSERT INTO jsonTest1 VALUES(2, ' { "obj1": {"key1": "test2", "key2": 777} }')
SELECT JSON_VALUE(coIB, '$.obj1.key1') AS key1
, JSON_VALUE(coIB, '$.obj1.key2') AS key2
, JSON_QUERY(coIB, '$.obj1') AS obj1
FROM jsonTest1
```



JSON_QUERY は、多段階層になっている場合に、特定のオブジェクトのみを抜き出す場合に便利な関数です。

以上のように、SQL Server には、JSON データを扱うためのたくさんの関数が提供されて、JSON データを簡単に扱えるようになっています。

4.3 Strech Database ~アクセス頻度の低いデータをクラウドへ~

Strech Database (ストレッチ データベース) は、テーブル データをクラウド上 (**Microsoft Azure SQL Database**) に配置することができる機能です。

これを利用すれば、アクセス頻度の低いデータ(アクセス ログや、3 年以上前の古いデータなど) や、コンプライアンス(法令遵守)目的でデータ蓄積しておく必要があるもの、ディスク容量を圧 迫し得る大量データなどを、クラウド上に配置することで、運用コストを削減できる可能性がある ものです。クラウド上に配置したデータは、SQL Server 上のローカル データと同じように操作す ることができるので、ユーザーは、データがクラウド上にあるのかどうかは意識することなく、透 過的に利用することができます。

Strech Database は、次のように設定のためのウィザードが用意されているので、非常に簡単に 設定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。

1. Strech Database を利用するには、まず sp_configure で「remote data archive」を 「1」に設定しておく必要があります。

```
EXEC sp_configure 'remote data archive', 1
RECONFIGURE
```

2. 次に、Strech Database を試すためのデータベースとテーブルを作成します。

```
-- データベースの作成
CREATE DATABASE strechTestDB
go
-- テーブルの作成
USE strechTestDB
CREATE TABLE strechTest1
( colA int
, colB int )
-- データを 2件追加
INSERT INTO strechTest1
VALUES (1, 111)
, (2, 222)
-- データの確認
SELECT * FROM strechTest1
```

データの確認 SELECT * FROM strechTest1					
100 %	• <				
🋄 結:	果 🛅 🤉	いセージ			
	colA	colB			
1	1	111			
2	2	222			

3. 次に、データベースを右クリックして、[**タスク**] メニューの [Strech] から [有効化] をク リックします。

	🧏 Streach Database.sql - WIN10.strechTestD8 (WIN10¥matumoto (142)) - Microsoft SQL Server Management Studio (管理者)								
ファイル(F) 編集(E) 表示(V) プロ	ロジェクト(P) デバッグ(D) ツール(T	レ(T) ウィンドウ(W) ヘルプ(H)							
🗄 🕶 🖬 🖉 🛃 🥥 🔔 新(fしいクエリ(N) 🗋 📸 📸 🍒	X ha ha ツ - C - 戸 - 弓 224 ト 229							
: 말 많 strechTestDB	▼ 『 実行(X) ▶ デバッ	(ツク(D) ■ ✓ 認 副 副 認 誤 唱 頌 确 頌 雪 ≌ 幸 幸 結 。							
オブジェクト エクスプローラー	▼ 🕂 × Streach Database	ase.s10¥matumoto (142)) 🛛 🗙							
接続 書	00 - WIN 104m 23ット 23ット USE str CREATE 80 テー USE str CREATE (colA , colB デー INSERT VALUES LUバデータベース(N) LUバデータベース(N) LUバデータベース(N) レンデータベーススクリプト化(S) デー INSERT VALUES レンデータベーススクリプト化(S) ・ セット(A) * のの変更(M) 除(D) 新の情報に更新(F) コ/ディ(R)	-タベースの作成 E DATABASE strechTestDB - ブルの作成 trechTestDB E TABLE strechTest1 A int B int) - - - - - - - - - - - - -							

4. [データベースの Strech の有効化] ウィザードが起動したら、[次へ] ボタンをクリックして、次のページに進みます。

🕋 データベースの Stretch の有効化	- 🗆 ×
ᢇ 説明	
説明	◎ ヘルプ
データベース資格情報の作成	SQL Server の Stretch の有効化
データベース テーブルの選択	このウイザードを使って、Stretch 用の SOL Server データベースを有効にし、移行するテーブルを選択する
SQL Server の設定の検証	手順を実行します。
Azure の配置の選択	この手順を完了するための要件:
Azureの配置の構成	SQL Server の管理者権限が必要です。Microsoft Azure サフスクリプションに関連する Microsoft ア カウントが必要です。
概要	
結果	
	操作を開始するには、(次へ)をクリックします。
	□ 次回からこのハーンを表示しない。(型)
	1 (東ス(N) > キャンセル

5. 次の [データベース資格情報の作成] ページでは、データベースのマスター キーを作成するために、パスワードを入力します。

簲 データベースの Stretch の有効化	- 🗆 ×
データベース資格情	報の作成
説明	◎ ヘルブ
データベース資格情報の作成	SQL Server データベース資格情報の作成
データベース テーブルの選択	Stretch を構成するには、ソース SQL Server データベース資格情報をセキュリティで保護するデ
SQL Server の設定の検祉 Azure の配置の選択	マスターキーのパスワードを入力してください。
Azureの配置の構成	パスワード ********
概要	パスワードの確認 *******
結果	
	強力なパスワードを作成するには、英数字の大文字、小文字および特殊文字を組み合わせて使用し ます。
	2 < 戻る(P) 次へ(N) > キャンセル

ここでは、推測されにくい複雑なパスワードを入力するようにします。

6. 次の [データベース テーブルの選択] ページでは、どのテーブルをクラウド上に配置したいの かを選択します。

🍅 データベースの Stretch の有効化						-		×
データベース テーブル	の選択							
説明							0	ヘルプ
データベース資格情報の作成	Azure (C)	ストレッチする SQL Si	erver データベースからテーブルを選	択します。				
データベーステーブルの選択		名前	Stretch が実行されました	行	サイズ (KB)	移	行	1
SQL Server の設定の検証		strechTest1	False	2	16	AI	IRows	
Azureの配置の構成								
概要								
結果								
	<						>	
					2	_		
				< 戻る(P) 次へ(I	N) >	キャンセ	2JL

ここでは、前の手順で作成した「**strechTest1**」テーブルをチェックして、[**次へ**] ボタンをク リックします。

7. 次の [SQL Server の設定の検証] ページでは、該当テーブルが Strech 可能かどうか (クラ ウド上に配置できるかどうか) がチェックされます。

🏟 データベースの Stretch の有効化	-		×
🛑 SQL Serverの設	定の検証		
説明		0	ヘルプ
データベース資格情報の作成	ウィザードが、 SQL Server データベース上で Stretch を有効にするための要件を検証しました。		
データベース テーブルの選択	以下の結果を確認してください。		
QL Server の設定の検証	名前	結果	1
zureの配置の選択		合格	
zureの配置の構成	⊘ テーブル [dbo].[streachTest1] に一意のキーまたは主キーが存在しないことを検証する	<u>合格</u>	
要	🥝 テーブル [dbo].[streachTest1] の型が Stretch でサポートされていることを検証する	合格	
	図 テーブル '[dbo].[streachTest1]' の列が Stretch でサポートされている型であることを検証する	<u>合格</u>	
吉果		<u>合格</u>	
		<u>合格</u>	
		<u>合格</u>	
1 Strec 要件	h Database の を満たしている ことを確認		
	検証の再実	行	

8. 次の [Azure の配置の選択] ページでは、[サインイン] ボタンをクリックします。

🏟 データベースの Stretch の有効化		- 🗆 ×
🔒 Azure の配置の選択	R	Microsoft SQL Server にサインインする X
説明 データベース英格情報の作成 データベーステーブルの選択 SQL Server の設定の検証 Azure の配置の構成 概要 結果	Microsoft Azure サインイン: Microsoft Azure にサインインしてい、 サインイン(S) Microsoft アカウントをお持ちではない アカウントへのサインアップ	Microsoft SQL Server 職場または学校アカウントでサインインする 2 1 1 1 1 フ フ フ ト た フ ト た フ ト た フ ト た フ ト た フ ト た フ ト た し 、 た い た に し 、 し 、 、 、 、 、 、 、 、 、 、 、 、 、
		職場または学校アカウントは、このシンボルが表示されているすべての ページで使用できます。 © Microsoft 2015 利用規約 プライバシー

[Microsoft SQL Server にサインインする] ダイアログが表示されたら、Microsoft Azure を契約している Microsoft アカウントのメール アドレスとパスワードを入力して、[サイン イン] ボタンをクリックします。

9. サインインが完了すると、次のように契約している Microsoft Azure のサブスクリプション が表示されます。

论 データベースの Stretch の有効化	– 🗆 X
in Azureの配置の選	祝
説明	@ ^ルブ
テータベース資格情報の作成	Microsoft Azure サインイン:
データベーステーブルの選択	matumotak@outlook.com としてサインインしています。 ユ <u>ーザーの変更</u>
SQL Server の設定の検証	使用するサブスクリプションの選択:
Azure の配置の選択	Azure MSDN (2b960cf2-(0f40a0) V
Azureの配置の構成	
概要	Azure リージョンの違訳 North Central US V
結果 2	Azıre SQL Server を選択します: ④ 新しい Server の作成 ① 既存の Server の選択 Azure SQL Server を作成するデータセンターの選択
	Azure の配置の構成を続けるには、[次へ]をクリックします。
	4 < 戻る(P) 次へ(N) > キャンセル

ここでは、Azure SQL Database 用のサーバー (Azure SQL Server)を新規作成するか、既存のサーバーを選択するかを選択します。サーバーを新規作成する場合は、[新しい Server の作成]、既存のサーバーを選択する場合は [既存の Server の選択] をクリックします。また、[Azure リージョンの選択] で、サーバーをどのデータセンターに作成するのか/既存のサーバーを利用する場合は、そのサーバーを作成しているデータセンターの場所を選択します(画面は North Central US:米国中北部を選択)。なお、既定のデータセンターは「Brazil South」ですが、CTP 3.2 の日本語版では、このデータセンターを選択すると、作成が失敗してしまいます。ですので、North Central US などに変更しておくようにしてください)。

10. 次の [Azure の配置の構成] ページでは、サーバー(Azure SQL Server)の管理者とパスワ ードを任意に設定します。

휁 データベースの Stretch の有効	ι – Ο ×
듥 Azure の 配置の	構成
説明	الا ال
データベース資格情報の作成	Azure SQL Database サーバーの資格情報を作成する
データベース テーブルの選択	新しい Azure SQL Database の設定を完了するには、サーバー管理者用のログイン名とパスワードを入力し
SQL Serverの設定の検証	
zureの配置の選択	
zureの配置の構成	
既要	
结果	
	Azure に接続するためのファイアウォール規則を構成する
	SQL Serverの IP アドレス範囲を指定します。これで Azure SQL Database サーバー が ユーザーの SQL Server の IP アドレス範囲を指定します。これで Azure SQL Database
	開始 IP アドレス 終了 IP アドレス
	▶ 118
	3
	< 戻る(P) 次へ(N) > キャンセル

[Azure に接続するためのファイアウォール規則を構成する]では、現在利用している IP アドレスを [開始 IP アドレス] と [終了 IP アドレス] に入力して、[次へ] ボタンをクリックします (Azure SQL Database では、IP アドレスでのアクセス制限を行っていて、既定ではすべての IP アドレスからのアクセスが拒否されているので、許可をしたい IP アドレスをここで指定する形になります)。

11. 次の [概要] ページでは、設定内容を確認して [完了] ボタンをクリックします。

🖦 データベースの Stretch の有効化	-		×
被要 説明 データベース英格情報の作成 データベース テーブルの選択 SQL Server の設定の検証 Azure の配置の選択 Azure の配置の構成 概要 結果	このウィザードで選択した内容を確認します。 (元 7) をクリックして、以下のように設定された操作を実行します。 ・ソース データバースの設定: ・ソース データバースへの設定: ・ソース データバースへの設定: ・ソース データバースマスワーキーの状態:新規作成 ・ Stretch を行うテー力に: ・ テーブルキ:(Hobo)[StreachTest1] ・ Azure SQL Database サーバーの設定: ・ Azure の場所:northcentralus ・ リンース プリープ4: stretchFroup-win10-northcentralus ・ リンース プリープ4: stretchsrewerwin10-streachtestdb-20151206-192857617 ・ サーバーを運動ログソ: ma ・ ウーバーのファイアウォール規則の IP 範囲: 118: ・ サーバーのファイアウォール規則の IP 範囲: 118:	ŝ) ~JJJ
	< 反3(P) 完7(F)	キャン	'tll

これで、Strech Database の作成(Azure SQL Database の作成や、Strech Database の 設定など)が始まり、設定中は、次のように [配置状況] ページが表示されます。

🛸 データベースの Stretch の有効化		-	□ ×
ᢇ 配置状況			
説明 データベース資格情報の作成 データベーステーブルの選択 SQL Server の設定の検証 Azure の配置の選択	Azure SQL Server stretchserver-win10-streachtestdb-20151206-162636862 Ø7	プロビジョニング	לענא @ גענא @ גענא פון אינער
Azure の配置の構成 概要 結果	 名前 Azure SQL Server stretchserver-win10-streachtestdb-20151206-162636 データベース ススクー キーの作成 デーダベース スコープ英格信報の作成 ファイアウオール規則の作成 デーダベース streachTestDB に Stretch を構成する テーブル (dbo).[streachTest1] に Stretch を構成する 	状態 実行中 未開始状態 未開始状態 未開始状態 未開始状態	
	< 戻る(P)	次八(N) >	キャンセル

12. 設定が完了すると、次のように [結果] ページが表示されます。

F-ダハ-ス 決格情報の作成 デーダハ-ス テーガルの選択 SQL Server の設定の検証 Azure の配置の選択 Azure の配置の選択 Azure の配置の選択 Azure の配置の選択 Azure SQL Server stretchserver-win 10-streachtestdb-20151206-1.	🕋 データベースの Stretch の有効化		-		×
説明 データベース英格信報の作成 データベーステーブルの選択 SQL Server の設定の検証 Azure の配置の選択 Azure の配置の機成 概要 5年 100000000000000000000000000000000000	● 結果				
データベース テーガルの選択 SQL Server の設定の検証 Azure の配置の選択 Azure の配置の機成 概要 諸果 「クスク 単細 ◇ Azure SQL Server stretchserver-win 10-streachtestdb-20151206-1 点価 ※ データベース マスター キーの作成 会经 ※ データベース スコープ 突 裕 電報の作成 会经 ※ データベース スコープ 突 裕 電報の作成 会经 ※ データベース スコープ 突 裕 電報の作成 会校 ※ デークベース スコープ デークベース Streach TestDB IC Streach を構成する 会校 ※ デークベース Streach TestDI IC Streach を構成する 会校 ※ デークベース Streach TestDI IC Streach を構成する 会校	説明			0 /	い レプ
データベース テーブルの選択 SQL Server の設定の検証 Azure の配置の選択 液要 3月 ● 27 ● Azure SQL Server stretchserver-win 10-streachtestdb-20151206-1 査証 ② データベース マスター キーの作成 ③ データベース マスター キーの作成 ③ データベース マスター キーの作成 ③ データベース スコープ英格谱報の作成 ④ データベース 3上の学校 情報の作成 ④ データベース 3上の学校 情報の作成 ● 2社 00000000000000000000000000000000000	データベース資格情報の作成				
SQL Server の設定の検証 Azure の配置の違択 Azure の配置の進成 標要	データベース テーブルの選択				
Azure の配置の選択 焼栗: タスク 詳細 焼栗 タスク 詳細 ● Azure SQL Server stretchserver-win10-streachtestdb-20151206-1 査短 ● データベース マスクーキーの作成 査短 ● データベース スコープ資格 信報の(作成 全超 ● データベース オーキの各地の 全超 ● データベース streachTestDB [: Stretch を構成する 全超 ● デークバース などの 会通 ● デークバース streachTestDB [: Stretch を構成する 会超 ● デーブル (dbo)[.streachTest)] [: Stretch を構成する 会通 ● アーブル (dbo)[.streachTest)] [: Stretch を構成する 会通 ● アーブル (dbo].streachTest)] [: Stretch を構成する 会通 ● アーブル (dbo].streachTest)] [: Stretch を構成する 会通 ● アーブル (dbo].streachTest)] 学校 第回 ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● <tr< td=""><td>SQL Server の設定の検証</td><td></td><td></td><td></td><td></td></tr<>	SQL Server の設定の検証				
タスク 詳細 Azure の配置の構成 タスク 詳細 焼麦 Azure SQL Server stretchserver-win10-streachtestdb-20151206-1 全版 第三 データハース マスクーキーの作成 全版 ジ データハース スコープ技術情報の作成 会版 ジ ア・クハース スコープ技術情報の作成 会版 ジ ア・クハース オローカンド成 会版 ジ ア・クハース オローカンド成 会版 ジ ア・クハース オローカンド成 会版 ジ ア・クハース オローカンド成 会版 ジ ア・クハース まtreachTestDB [: Stretch を構成する 会版 ジ ア・フル [dbo].[streachTest1] [: Stretch を構成する 会版 ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ジ ア・クハース またるいTest1] [: Stretch を構成する 会版 ジ ア・フル [dbo].[streachTest1] [: Stretch を構成する 会版 ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ <	Azureの配置の選択	概要:			
Addre offall 豊い時の 標要		920	詳細]
 ○ データバース マスター キーの作成 会経 ジークバース スープ 支後 情報の作成 会塩 ジークバース スープ 支後 情報の作成 会塩 ジークバース streachTestDB [: Stretch を構成する 会塩 デークバース streachTestDB [: Stretch を構成する 会塩 デーフル [dbo][streachTestD] [: Stretch を構成する 会塩 Fram: C:¥Users¥matumoto¥AppDat¥Local¥SqlServer¥StretchWizardLogs*20154306-184353-WIN1	Azure on L L on L pon	Azure SQL Server stretchserver-win10-streachtestdb-20151206-1	<u>合格</u>		
結果	概要	🥝 データベース マスター キーの作成	合格		
 ○ ファイアウォール規則の作成 会述 ○ デークペース streachTestDB に Stretch を構成する 会述 ○ テーブル [dbo].[streachTest1] に Stretch を構成する 会述 ● オーブル [dbo].[streachTest1] に Stretch を構成する 会述 	結果	データベーススコープ資格情報の作成	合格		
② データベース streachTestDB に Stretch を構成する ③ デーブル [dbo].[streachTestD] に Stretch を構成する ③ 空 デーブル [dbo].[streachTest1] に Stretch を構成する ③ 合弦 ① 作成が成功 したことを確認 詳細: C:¥Users¥matumoto¥AppData¥LocsI¥SqlServer¥StretchWizardLogs¥20154306-184353-WIN1		✓ ファイアウォール規則の作成	<u>合格</u>		
● デーブル [dbo].[streachTest1] [: Stretch を構成する 全経 作成が成功 したことを確認 詳細: C:¥Users¥matumoto¥AppData¥Local¥SqlServer¥StretchWizardLogs¥20154306-184353-WIN1		🥝 データベース streachTestDB に Stretch を構成する	合格		
作成が成功 したことを確認 詳細: C:¥Users¥matumoto¥AppData¥Local¥SqlServer¥StretchWizardLogs¥20154306-184353-WIN1		🥝 テーブル [dbo].[streachTest1] に Stretch を構成する	合格		
詳細: C.¥Users¥matumoto¥AppData¥Local¥SqlServer¥StretchWizardLogs¥20154306-184353-WIN1	1 作成が成 したことを	功 確認			
		詳細: C:¥Users¥matumoto¥AppData¥Local¥SqlServer¥StretchWizardLogs¥201: 0.LOG	<u>54306-184</u>	1353-WIN1	

すべてのタスクが [合格] と表示されれば、設定が完了です。

以上で、Strech Database の設定が完了です。これで、Microsoft Azure 上に Azure SQL Database (とサーバー)が自動的に作成されるので、以降では、これを確認していきましょう。

♦ Microsoft Azure 管理ポータルで自動作成されたデータベースの確認

まずは、Microsoft Azure 上に自動作成された Azure SQL Database を確認します。

1. 自動作成された Azure SQL Database を確認するには、Azure 管理ポータルにアクセスしま す。これは、次の URL でアクセスできます。

Microsoft Azure 管理ポータル http://portal.azure.com/



2. 管理ポータルが開いたら、[**SQL Server**] をクリックして、Azure SQL Database 用のサー バー(Azure SQL Server)を確認します。

Microsoft Azure 🗸	SQL Server	م	9 🗳 🖉 😳
≡			
十 新規	777777777777777		
📦 リソース グループ	■■ ひ 列 更新		
📕 すべてのリソース	フィルター項目		
- 最近使った項目	名前	状態	場所
🔇 App Services	🐻 matusrv1	利用可能	米国中北部
👼 SQL データベース	2 a matusrvdw	利用可能	米国東部 2
👰 仮想マシン (クラシック)	stretchserver-win10-streachtestdb-20151206-184416345	利用可能	米国中北部
👰 仮想マシン			
🍪 クラウド サービス (クラシック)		/	
↑ サブスクリプション		ウィザードで指定したデ・ サーバー(Azure SQL S	ータセンターに ierver)が作成
👼 SQL Server		されている。North Cer 米国中北部と表示	ntral US なら えされる
💠 SQL 可変プール			

「strechserver-マシン名-データベース名-日付-~」という名前のサーバーが自動的に作成 されていることを確認できます。

3. 次に、このサーバーをクリックすると、サーバー内に作成されたデータベース(Azure SQL Database)を確認することができます。

	* ×	* X
十 新規	SQL Server	stretchserver-win10-streachtestdb-20151206-18441
🗊 リソース グループ	≕ ひ 列 更新	★ ★ 生 面 → + 設定 パスワード デーダベー 削除 移動 ブールの追 のNHPreil- マのイ・・・・ tra
🏭 すべてのリソース	フィルター項目	要点 ∧
🕒 最近使った項目	名前	リソース ガループ サーバーのバージョン stretchgroup-win10-northcentralus V12
🔕 App Services	🚮, matusrv1 🛛	状態 監査 利用可能 構成されていません
👼 SQL データベース	matusrvdw ····	場所 サーバー管理者 North Central US matumoto
👰 仮想マシン (クラシック)	stretchserver-win10-streachtestdb- · · · ·	サススクリプション名 ファイアウォール Azure MSDN ファイアウォール設定の表示
👰 仮想マシン		$\frac{1}{2b960cf2 \cdot ccfb} - \frac{1}{2b} \frac{1}{40} \frac{1}{2b} \frac{1}$
🛷 クラウド サービス (クラシック)		が自動的に作成されている
💡 サブスクリプション		データバース 価格レベルは、CTP 3.2 では
👼 SQL Server		SQL データベース 「Standard S3」に 自動設定される
💠 SQL 可変プール		2 17-91-2
🧮 ストレージ アカウント		データベース 状態 価格レベル
🚍 ストレージ アカウント (クラ…		RDAstreachTestDBC3800AB Online Standard: S3

[RDA データベース名~] という名前のデータベースが自動的に作成されて、[価格レベル] が [Standard S3] に設定されていることを確認できます。Azure SQL Database では、1 時間単位での課金になり、価格レベルが「S3」で、データセンターが「米国中北部(North Central US)」の場合は、1 時間あたり 20.57 円です。価格は、データセンターの場所によ って異なり、「東日本 (Japan East)」を選択している場合は、S3 だと 1 時間あたり 23.28 円になります。 Azure SQL Database の価格については、次の URL で確認できます。

SQL Database の価格

http://azure.microsoft.com/ja-jp/pricing/details/sql-database/

Azure を選ぶ理由 製品 ドキュメン	ト 価格 パートナー ブロ	<i>ば</i> リソース リポ ート	無料評価版 >
価格の詳細	Single	Database	
Elastic Database Single Database	- 	לעטי	
リージョン: 通貨:			
*国中北部 - 日7	k円 (¥) ∨		
	デー	タセンターを	
Single database -		選択	
単一データベースは、パフォーマンスの タベースは、Basic、Standard Premii	需要がある程度予測できるワー um の各サービス レベルの間で	クロード用に最適化された、完全に分離さ スケールアップやスケールダウンが可能で	されたデータベースです。 シングル デー きす。これにより、アプリのニーズに今
わせたパフォーマンスと機能を、ちょう	ど必要なときに活用することが	できます。各レベルの主な違いは、データ	パース トランザクション ユニット
(DTU) によって測定されるパフォーマン	々にあります。詳細については	は 「サービス レベル」を参照してください	•
Basic			
Basic			
Basic	DTU	DB あたりの最大ストレージ 容量	料金 *
Basic	DTU	DB あたりの最大ストレージ 容量 2 GB	利金 *
Basic B	דע 5	DB あたりの最大ストレージ 容量 2 GB	料金* ¥0.69/時間 (~¥510/月)
Basic	DTU 5	DB あたりの最大ストレージ 容量 2 GB	料金 * ¥0.69/時間 (~ ¥510/月)
Basic B Standard	DTU 5	DB あたりの最大ストレージ 容量 2 GB	料金 * ¥0.69/時間 (~ ¥510/月)
Basic B Standard	5 DTU	DB あたりの最大ストレージ 容量 2 GB DB あたりの最大ストレージ	料金 * ¥0.69/時間 (~ ¥510/月) 料金 *
Basic B Standard	DTU 5 DTU	DB あたりの最大ストレージ 育量 2 GB DB あたりの最大ストレージ 育量	料金 * ¥0.69/時間 (~ ¥510/月) 料金 *
Basic B Standard	DTU 5 DTU 10	DB あたりの最大ストレージ 容量 2 GB DB あたりの最大ストレージ 容量 250 GB	料金* ¥0.69/時間 (~¥510/月) 料金* ¥2.07/時間
Basic B Standard	דע 5 סדע 10	DB あたりの最大ストレージ 容量 2 GB DB あたりの最大ストレージ 容量 250 GB	料金* ¥0.69/時間 (~¥510/月) 料金* ¥2.07/時間 (~¥1,530/月)
Basic B Standard S0 S1	рти 5 рти 10 20	DB あたりの最大ストレージ 容量 2 GB DB あたりの最大ストレージ 容量 250 GB 250 GB	料金* ¥0.65/時間 (~¥510/月) 料金* ¥2.07/時間 (~¥1,530/月) ¥4.12/時間 (~¥3.00/月) S3 の場
Basic B Standard S0 S1	рти 5 рти 10 20	DB あたりの最大ストレージ 容量 2 GB DB あたりの最大ストレージ 容量 250 GB 250 GB	料金* ¥0.65/時間 (~¥510/月) 料金* ¥2.07/時間 (~¥1,520/月) ¥4.12/時間 (~¥3,060/月) ¥4.12/時間 (~¥3,060/月)
Basic B Standard S0 S1 S2	DTU 5 DTU 10 20 50	DB あたりの最大ストレージ 容量 2 GB DB あたりの最大ストレージ 容量 250 GB 250 GB 250 GB	料金* ¥0.69/時間 (~¥510/月) 料金* ¥2.07/時間 (~¥1.530/月) ¥4.12/時間 (~¥3.060/月) ¥10.29/時間 (~¥7.550/月)

[Single Database] をクリックして、[**リージョン**] でデータセンターを選択することで、 価格を確認することができます。

4. Azure 管理ポータルでは、該当データベースをクリックして、次のように [価格レベル (DTU のスケール)] をクリックすると、価格を確認 (ひと月あたりの価格) することもできます。

* _ □ × ■ RDAstreachTestDBC3800AB2-C618-467C-A464-4667… ◎ QL デーパース	_ ロ × 設定 RDAstreachTestDBC3800AB2-C618-467C-A46…	× (1) (1) (1) (1) (1) (1) (1) (1) (1) (1)
◆ ☆ ぱ ぱ つ つ 不 団 いはは コピー 紀元 272パー 削除 いたか		データベーススループか、コント (DTU) は、Basic データベース、Standard データベース、および Premium データベース の(ジェーマント レイルの部状内容量を定乱したう、これはは、CPU、メモリ、読み取り、および書き込みの(ジォーマンスを総合し た意志に基づいています、宇宙活用(二)
<u>π</u> , γ ζ	○ 検索設定	47.430.00 94.860.00 189.720.00
リン-ス 7/h.−プ リー(−名 stretchgroup-win10-northcentralus stretchserver-win10-streachtestdb-20 状態 リー(−の)(−ション) Online V12		JPV/HOKTH (ESTIMATED 21 P- JPV/HOKTH (ESTIMATED 21 P- P6 Premium P11 Premium S0 Standard
場所 機能文字列 North Central US データバース/単純文字列の表示 サイフロリたった	 価格レベル (DTU のスケール) 	10 DTUs 17 DTUs 10 DTUs
Azure MSDN S3 Standard (100 DTUs)	監視	Up to 500 GB Up to 1024 GB Up to 250 GB
2b960cf2-ccfb-4efc-b920-c083280f40… 構成されていません	アラートルール >	Active Geo-Replic… Active Geo-Replic
すべての設定→	🗟 データベース サイズ >	Auditing Auditing Auditing
監視 タイルを追加 ④	■ 1/3/t >	
リソース使用率	物に	
100%		379,440.00 714,011.22 1,530.09
80%	▲ インデックス アドバイザー(プレビュー) >	S1 Standard S2 Standard S3 Standard *
60%	Multiple Control Performance Insight (プレー・・ >	20 DTUS 50 DTUS 100 DTUS
40%		
20%	641)F7	Standard Geo-Rep Standard Geo-Rep Standard Geo-Rep
0%	計算 監査と骨成検出 >	Point In Time Res
18:15 18:30 18:45 19:00 DTU PERCENTAGE	「「「「「」」、「「」」、「「」」、「「」」、「「」」、「」、「」、「」、「」、	Auditing Auditing Auditing
0.07 %	 「Standard S3」の場合 の金額を確認できる 	Available only for
	-#	
監査 インデックス アドバイザー	/ \$1	3,059.87 JPY/MONTH (ESTIMATED 31 5
€=前の有効がとサットアッ 11.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1		進 段

また、このページでは、価格レベルを変更することもできます。S3 の場合は、データセンタ ーが「**米国中北部 (North Central US)**」の場合に、ひと月あたり 15,299.97 円かかるので、 Strech Database を試した後は、必ずデータベースを削除しておくするようにしておいてく ださい。削除方法については後述します。なお、データベースを削除せずに、B (Basic) レベ ルに価格を変更した場合は、ひと月利用したとしても、約 510 円で済みます。

→ 動作の確認

次に、Strech Database の動作を確認してみましょう。

 Strech Database では、テーブル データはクラウド上(Azure SQL Database 内)に自動 的に配置されますが、ユーザーは、それを意識することなく、(ローカルにデータがあるかのよ うに)データを利用することができます。次のように SELECT ステートメントを実行して、 データを参照してみましょう。

SELECT * FROM strechTest1



Strech Database を設定する前と同様、同じデータを参照できることを確認できます。

2. 次に、ツールバーの [実際の実行プランを含める] をクリックして、実行プランを確認します。



Strech Database を設定すると、[Remote Query] と表示されて、リモートのデータソー スからデータを取得したことを確認できます。

◆ Strech Database モニターで確認

Strech Database を設定すると、[**Strech Database モニター**] というツールが追加されて、状態(接続状況など)を確認することができます。

1. [Strech Database モニター] を開くには、次のようにデータベースを右クリックして、[タ スク] メニューの [Strech] から [モニター] をクリックします。



2. [Strech Database モニター] は、次のように表示されます。

💺 Stretch データベース モニター [strechTestDB] - M	licrosoft SQL Server Management S	tudio (管理者)			- 🗆 ×
ファイル(F) 編集(E) 表示(V) プロジェクト(P) う	デバッグ(D) ツール(T) ウィンドウ(W)	ヘルプ(H)			
🗄 🎦 🕶 📨 💕 🛃 🗿 🔔 新しいクエリ(N) 💧	🗅 🚯 🚳 🌇 🕹 🐁 🖃	- (* - 🗐 - 🖳 🙀	-	Church Database T	
· 嬰 說 strechTestDB - ?	実行(X) 🕨 デバッグ(D) 🔳 🗸 🖁		白柳白 三台 連連 公	streach Database モ_ が表示される	_9_
オブジェクト エクスプロ−ラ− 🔹 🕂 ×	Stretch データベース モニター [strech	TestDB] × SQLQuery1.sql	- W10¥matumoto (142)) Stre	ach Datab #matumoto (143))	-
接続 * 23 23 = 7 2 3 - 10 WIN10 (SQL Server 13.0.900 - WIN10¥m - 10 データベース - 10 ジェアモルデータバース	WIN10:strech	TestDB		最終更新: 2016/ 自動更新:	01/12 1:20:28
			• 🥭	matumotak@outlook.com とし	てサインインしています。 <u>ユーザーの</u> :
■ データベース ダイアグラム	サーバー名 WIN10		Azure サーバー名	stretchserver-win10-strechte	estdb-20160112-011255551
□□□ ジステム テーブル	データベース名 strechT	estDB	Azure データベース名	RDAstrechTestDBAC2A6ED8	-617D-4{ .
田 □ FileTables 田 □ 外部テーブル	□-カルの有効領域 5.31 M	в	サービス層	Standard/S3	Azure SQL Database
😑 🔝 dbo.strechTest1			サーバーのリージョン	North Central US	の価格レベルの確認も できる
 ■ 万 ■ 二 万 ■ 二 千 ■ 二 新約 			データベース ストレージ コン	いト 使用できません	請求明細
田 □ トリガー 田 □ インデックス ボー ボー	Stretch 構成のテーブル:			<u>Stretch デー</u>	タベース ヘルス イベントの表示
	名前	移行状態	対象になる行 ローカルイ	〒 Azure の行	詳細
 ■ 二 外部リソース ■ シノニム □ プログラミング 	✓ dbo.strechTest1	Outbound v	2 0	2	View

動作が正常の場合は、緑の矢印のアイコンが表示されます。

→ Azure 管理ポータルでデータベースを削除

Strech Database の動作を確認したら、Microsoft Azure 上からデータベース (Azure SQL Database) を削除しておきましょう (削除しておかないと、課金が続くことになるので注意してください)。

- **1.** Azure SQL Database のデータベースを削除するには、まず、Azure の管理ポータル (http://portal.azure.com/) にアクセスします。
- 管理ポータルが開いたら、[SQL Server] をクリックして、サーバー(Azure SQL Server) の一覧を表示して、Strech Database 用のサーバー(strechserver-マシン名-データベース 名-日付-~ 形式のもの)をクリックします。

三 十 新規	SQL Server	stretchserver-win10-streachtestdb-20151207-09562
📦 リソース グループ	≕ ひ 列 更新	★ ✓ 业 茴 → + 設立 パスワード データペー 削除 移動 ブールの追 のItemte - エクロペン::::::::::::::::::::::::::::::::::::
🏢 すべてのリソース	フィルター項目	要点 ^ (4) 8名 🧷
④ 最近使った項目	名前	リソース ガループ サーバーのバージョン stretchgroup-win10-japaneast V12
🔇 App Services	😹, matusrv1	 状態 監査 利用可能 構成されていません
SQL データペース	matusrvdw ····	場所 サーバー管理者 Japan East matumoto
👰 仮想マシン (クラシック)	stretchserver-win10-streachtestdb-: · · ·	サブスのリプション名 ファイアウォール Azure MSDN ファイアウォール設定の表示
🧕 仮想マシン		サンズクリフション ID 2b960cf2-ccfb-4efc-b920-c083280f40…
🤣 クラウド サービス (クラシック)		すべての設定 →
↓ 📍 サブスクリプション		データベース タイルを追加 ①
SQL Server		SQL データベース
🚸 SQL 可変プール		1 データベース 👼
ストレージ アカウント		3 データベース 状態 価格レベル
🚍 ストレージ アカウント (クラ…		RDAstreachTestD88349701 Online Basic

サーバー内のデータベースの一覧が表示されたら、Strech Database 用のデータベース(RDA データベース名~ 形式のもの)をクリックします。

3. 次のようにデータベースの構成が表示されたら [削除] をクリックします。

	hTestDB83	497011-9	917F-494	* 1A-AA4	_ □ 0-B386…	
♣ ♥ 設定 Visual I Structio ····	山 り ピー 復元	₹ 192ポ- ト	面 削除			
'RDAstreachTestD このデータベースを完全に	B83497011- 削除しますか?	917F-494/	A-AA40-B3	864F1A	E03F' の…	
2						_
サブスクリプション名	_	価格レベ		*******		-
A2dre MSDN サブスグリプション ID 2b960cf2-ccfb-4efc-b	920-c083280f4	geo レプ 10… 利用でき	3 D105) リケーション ロール きません			
					すべての設定 →	
監視					タイルを追加 🕀	
リソース使用率					編集	

[このデータベースを完全に削除しますか?]と表示されたら、[はい]をクリックします。こ

れで削除が開始されて、削除が正常に完了すると、画面の右上に、次のように[データベース が正常に削除されました]と表示されます。



これが確認できれば、データベースが完全に削除されているので、これ以上課金されることはなくなります。

データベースの削除ができれば、サーバー(Azure SQL Server)が残っていても、課金されることはなくなりますが、サーバーも削除したい場合は、次のように操作します。サーバーを表示した画面で[**削除**]をクリックします。

	Microsoft Azure 🗸	SQL Server > stretchserver-win10-streachtestdb	9-20151207-095620979 〉 設定	
		≯ X	stratchsenver-win10-streachtestdh-20151207-00562	
	十新規		SQL Server	
	🗊 リソース グループ			
	🗰 すべてのリソース	フィルター項目	要点 ^ (生 名 🖉	
	④ 最近使った項目	名前	リソース グループ サーバーのバージョン stretchgroup-win10-japaneast V12	
	🔇 App Services	🖳 matusrv1 🗤	状態 監査 利用可能 構成されていません	
	🗟 SQL データベース	matusrvdw ····	場所 サーバー管理者 Japan East matumoto	
	👰 仮想マシン (クラシック)	stretchserver-win10-streachtestdb-: · · ·	サブスクリプション名 ファイアウォール Azure MSDN ファイアウォール設定の表示	
	🖳 仮想マシン		サンパリンタン 10 2b960cf2-ccfb-4efc-b920-c083280f40…	
	🍩 クラウド サービス (クラシック)		すべての設定 →	
	💡 サブスクリプション		データベース タイルを追加 ④	
Ï	👼 SQL Server		SQL 7-9/-X	
	💠 SQL 可変プール		0 データベース 🗃	
	🧮 ストレージ アカウント		データペース 状態 価格レベル	
	🚍 ストレージ アカウント (クラ…		データバースが見つかりません	
	参照 〉			

サーバーの削除の場合は、確認のために、次のように [**サーバー名**] を入力して、[**削除**] ボタンをクリックします。

stretchserver-win10-s sqL server 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	streachtestdb-2015120 → + 移動 ナールの治 加	07-09562	stretchserver-win10-streachtestdb-20151207-095620979···
要点 ^ リース グループ stretchgroup-win10-japaneast 状態 利用可能 場所 Japan East サプスのプジュンE Azure MSON サプスのプジュンED 2006-07-07-07-07-07-07-07-07-07-07-07-07-07-		<u>19</u> R Ø	
2090002-000-4610-0520-000520014	0	すべての設定 →	
データベース		タイルを追加 🕀	影響を受ける項目
SQL データベース 0 データベース 👼			データベース名 場所 働格レベル このサーバーの下にデータベースは存在しません。
データペース 状態	価格レベル		
データベースが見つかりません			٠
			5

以上で、サーバー(Azure SQL Server)の削除が完了です。

その他、Strech Database については、オンライン ブックの以下のトピックがお勧めです。

Strech Database

Disable Stretch Database

Pause and resume Stretch

Manage and troubleshoot Stretch

Database

Database

https://msdn.microsoft.com/en-us/library/dn935011.aspx

 Databases (Database Engine) 	Stretch Data
 Databases (Database Engine) Stretch Database Identify databases and tables for Stretch Database by running Stretch Database by running Stretch Database Advisor Surface area limitations and blocking issues for Stretch Database Enable Stretch Database for a database Enable Database for Stretch Wizard Reconfigure Stretch Database Monitor Stretch Database 	Stretch Data SQL Server 2016 Applies To: SQL Server 2016 Previe Stretch Database archives your his Technology Preview 3 (CTP 3.0), St cloud. After you enable Stretch Da Database. Stretch Database provides the follo • You typically enjoy reduced • Your local queries and data
Enable Stretch Database for a table	 You don't have to change e to both local and remote days
Unmigrate remote data (Stretch Database)	What does Stretch

<

Stretch Database

ew

storical data transparently and securely. In SQL Server 2016 Community tretch Database stores your historical data in the Microsoft Azure atabase, it silently migrates your historical data to an Azure SQL

lowing benefits:

- cost and complexity.
- abase operations against current data typically run faster.
- existing queries and client apps. You continue to have seamless access ata.

Database do?

After you enable Stretch Database for a local server instance, a database, and at least one table, it silently begins to migrate your historical data to an Azure SQL Database. You can pause data migration to troubleshoot problems on the local server or to maximize the available network bandwidth.

Stretch Database ensures that no data is lost if a failure occurs during migration. It also has retry logic to handle connection issues that may occur during migration. A dynamic management view provides the status of migration.

4.4 Azure バックアップ URL の性能向上(ブロック BLOB 対応)

SQL Server では、バックアップを Microsoft Azure の BLOB ストレージ上に配置する機能が SQL Server 2012 の SP1 CU2 から利用することができましたが、これは BLOB ストレージ内 の「ページ BLOB」にファイルを格納していました。SQL Server 2016 からは、従来と同様「ペ ージ BLOB」に格納する方法に加えて、「ブロック BLOB」に格納する方法が追加されました。

ページ BLOB は「**ランダム読み取りと書き込みに最適化された 512 バイト単位のストレージ**」、 ブロック BLOB は「**複数のブロックを並列操作することで大きなファイルを効率よく管理できるス** トレージ」という特徴があります。

↓ ページ BLOB を利用したバックアップ(従来ながらの利用方法)

従来の SQL Server と同様、ページ BLOB にバックアップを行う場合は、次のように利用します。

CREATE CREDENTIAL 資格情報名 WITH IDENTITY = 'ストレージ アカウント名', SECRET = 'ストレージ アカウントのアクセス キー' BACKUP DATABASE sampleDB T0 URL = 'https://ストレージ アカウント名.blob.core.windows.net/コンテナー名/~.bak' WITH CREDENTIAL = '資格情報名', COMPRESSION

CREATE CREDENTIAL ステートメントで、**IDENTITY** に Microsoft Azure 上の**ストレージ ア カウントの名前**、**SECRET** にストレージ アカウントにアクセスするための**アクセス キー**を指定 して、資格情報を作成します。この資格情報を利用して(**CREDENTIAL** 句で資格情報名を指定し て)、**BACKUP** ステートメントを実行すれば、**TO URL** 句で指定した Microsoft Azure 上の BLOB ストレージ(コンテナー)内にページ BLOB としてバックアップを行うことができます。

WITH で指定している COMPRESSION は、バックアップ圧縮を行うための設定で、必須ではありませんが、バックアップ ファイルを圧縮することでネットワーク上を流れるデータ量を減らすことができるので、バックアップにかかる実行時間を短縮することができます。

なお、ページ BLOB では、1 つのファイルの最大サイズが「**1TB**」までという制限があるので、バックアップ ファイルのサイズは 1TB 未満に抑える必要があります。

これに対して、後述のブロック BLOB を利用したバックアップでは、1 つのファイルあたりは 「200GB」のサイズに制限されますが、<u>ストライプ バックアップ</u>(複数のファイルに並列でバッ クアップ処理)ができるので、1TB 以上のサイズでもバックアップが可能です(最大は 12.8TB までのストライプ合計サイズがサポートされます)。

ページ BLOB では、ストライプ バックアップがサポートされていないので、これを行おうとする と、次のようにエラーになります。



→ ブロック BLOB へのバックアップ(SQL Server 2016 からの新機能)

SQL Server 2016 からは、 ブロック BLOB へのバックアップおよびストライプ バックアップが サポートされたことで、ページ BLOB にバックアップするよりも性能向上、より大きなサイズへ の対応(最大サイズの上限が 1TB から 12.8TB へ)ができるようになりました。

ブロック BLOB ヘバックアップを行うには、SAS (Shared Access Signature:共有アクセス署 名)を利用して、次のように実行します。

```
CREATE CREDENTIAL [https://ストレージ アカウント名.blob.core.windows.net/コンテナー名]
WITH IDENTITY='Shared Access Signature'
, SECRET='SAS'
BACKUP DATABASE データベース名
TO URL = 'https://ストレージ アカウント名.blob.core.windows.net/コンテナー名/~.bak'
WITH COMPRESSION
```

CREATE CREDENTIAL ステートメントで、**IDENTITY** に「**Shared Access Signature**」と 記述して、**SECRET** に **SAS** (Shared Access Signature : 共有アクセス署名) を指定します (SAS の作成方法については後述します)。また、資格情報名には、ストレージ アカウントのコンテナー 名までの URL を [] で囲んで指定します。

BACKUP ステートメントでは、TO URL に指定する BLOB のコンテナー名と CREATE CREDENTIAL で指定したコンテナー名を同じにすることで、SAS (共有アクセス署名) を利用したアクセスができます。また、このように、バックアップをすることで、ブロック BLOB にバック アップすることができます。

SAS を利用したバックアップでは、ストライプ バックアップもサポートされるので、次のように TO URL で指定するファイル パスを複数指定(カンマで区切って指定)することができます。

```
BACKUP DATABASE データベース名
TO URL = 'https://ストレージ アカウント名.blob.core.windows.net/コンテナー名/~_1.bak'
,URL = 'https://ストレージ アカウント名.blob.core.windows.net/コンテナー名/~_2.bak'
,URL = 'https://ストレージ アカウント名.blob.core.windows.net/コンテナー名/~_3.bak'
WITH COMPRESSION
```

BACKUP DATABASE NorthwindJ TO URL = 'https://matustorage.blob.core.windows.ne ,URL = 'https://matustorage.blob.core.windows.ne ,URL = 'https://matustorage.blob.core.windows.ne WITH COMPRESSION, STATS	et/cont1/North_1.bak' et/cont1/North_2.bak et/cont1/North_3.bak'
100 % - <	
🌆 メッセージ	ストライプ バックアップ
10 パーセント処理されました。 20 パーセント処理されました。 31 パーセント処理されました。 41 パーセント処理されました。 51 パーセント処理されました。 60 パーセント処理されました。 70 パーセント処理されました。 81 パーセント処理されました。 91 パーセント処理されました。 データベース 'Northwind' の 384 ページ、ファイル 1 100 パーセント処理されました。 データベース 'Northwind' の 2 ページ、ファイル 1 0 BACKUP DATABASE により 386 ページが 6.141 秒間で正常	の実行例 のファイル 'NorthwindJ' を処理しました。)ファイル 'NorthwindJ_log' を処理しました。 別に処理されました (0.489 MB/秒)。

ブロック BLOB としてバックアップされたことを確認するには、Azure の管理ポータルで、該当 ストレージ アカウントのコンテナーを表示すれば、次のように確認することができます。

BLOB サービス matustorage			ront1 ⊐>テナ−		
众 設定			し 面 最新の情 削除 48(/:面4··	注 『 プロパティ アクセス ポ ^{」 ミノー}	
要点 へ		À		5 BLOB の検索 (大文字と小文)	*を区別)
ストレージ アカウント matustorage	BLOB サービス エン https://matus	ドポイント storage,blob.core.windo…	名前	変更	BLOBの種類
状態 Primary: 利用可能			North_1.bak	2015/12/8 18:32	ブロック BLOB
場所 米国中北部			North_2.bak	2015/12/8 18:32	ブロック BLOB
サブスクリプション名 Azure MSDN			North_3.bak	2015/12/8 18:32	ブロック BLOB
サブスクリプション ID 2b960cf2-ccfb-4efc-b920-c	:083280f40				
		すべての設定 →		ブロック BLOB としてファイル	
	<	テナー タイルを追加 🕀		が作成されている	
コンテナー	をク	1Uv2			
	~				
1 名前	URL	最終変更日時			
conti	https://matustorage.blob	2015/1/31 5:41:09			
cont2	https://matustorage.blob	2015/12/7 22:38:25			
cont3	https://matustorage.blob	2015/11/9 22:43:10			

このように、ブロック BLOB にストライプ バックアップを行うことで、弊社環境では、2 億件の データベースのバックアップにかかった時間が、ページ BLOB では 2分35秒だったところを、 1分 43秒に短縮することができました(33.5%の性能向上)。簡易的な検証だったので、ストラ イプ数を 10 個~50 個に変更したりしましたが、ネットワーク帯域のボトルネックとの兼ね合い もあるので、最適なストライプ数(何個のファイルに分割すれば良いか)は状況によって変わって くると思うので、ぜひいろいろなパターンを試してみてください。

◆ SAS (Shared Access Signature : 共有アクセス署名)の作成方法

SAS(共有アクセス署名)を作成する一番簡単な方法は、Codeplex で提供されている「Azure Storage Explorer」ツールを利用する方法です。これは、次の URL からダウンロードできます。 http://azurestorageexplorer.codeplex.com/

COdePlex Project Hosting for Open Source Software Register Sign						h all projects	Q
Azure	Storage E	kplorer					
HOME	SOURCE CODE	DOWNLOADS	DOCUMENTATION	DISCUSSIONS	ISSUES	PEOPLE	LICENSE
Page Info	Change History (all p	ages)				★ Follow (431)	Subscribe
					Count Milli & D		0
New: Azure Storage Explorer 6 Preview 3 (August 2014)							
New: A	zure Stora	ge Explorer 6 P	Preview 3 (Augu	ıst 2014)	Search wiki & D	ocumentation	~ ~
New: A low available lodern code review 3 bas	Azure Stora	ge Explorer 6 P ection test cloud platform, APIs, and le sunnort.	Preview 3 (Augu	ust 2014)	Search wiki at D	download	ls
New: A low available fodern code review 3 has Valk-through	e in the Downloads se base, updated for la blob, queue and tab h: http://davidpallmar	ge Explorer 6 P ection test cloud platform, APIs, and le support. an.blogspot.com/2014/08/azu	developer tools.	ust 2014) ^{3-now.html}	CURRENT	downloac Azure Storage Exp	S olorer 6 Preview 3
New: A low available todern code review 3 has /alk-through	a in the Downloads se base, updated for la blob, queue and tab h: http://davidpallmar	ge Explorer 6 P ection est cloud platform, APIs, and le support. In.blogspot.com/2014/08/azu	Preview 3 (Augu developer tools. re-storage-explorer-preview-	Jst 2014) 3-now.html	CURRENT DATE	Azure Storage Exp Fri Aug 15, 2014 a	olorer 6 Preview 3 It 5:00 PM
New: A low available lodern code review 3 has /alk-through Anue Storage for trazileccunt	e in the Downloads so base, updated for la bob, queue and tab h: http://davidpallmar prorr	ge Explorer 6 P ection est cloud platform, APIs, and le support. In.blogspot.com/2014/08/azu	Preview 3 (Augu developer tools. re-storage-explorer-preview-	Jst 2014) 3-now.html	CURRENT DATE STATUS	Azure Storage Exp Fri Aug 15, 2014 a Beta	S plorer 6 Preview 3 ht 5:00 PM
Vew: A low available Modern code treview 3 has Valk-through trailecount trailecount	e in the Downloads sr base, updated for la b blob, queue and tab h: http://davidpallman prov	ge Explorer 6 P ection est cloud platform, APIs, and le support. In.blogspot.com/2014/08/azu	Preview 3 (Augu developer tools. re-storage-explorer-preview-	Jst 2014) 3-now.html	CURRENT DATE STATUS DOWNLOADS	Azure Storage Exp Fri Aug 15, 2014 a Beta 138,634	Dolorer 6 Preview 3 It 5:00 PM
New: A low available Modern code treview 3 has Valk-through Anue Storage For Enailaccount Storage Account brazilaccount	Account Storad	ge Explorer 6 P ection test cloud platform, APIs, and le support. in.blogspot.com/2014/08/azu	Preview 3 (Augu developer tools. re-storage-explorer-preview-	Jst 2014) 3-now.html	CURRENT DATE STATUS DOWNLOADS RATING	Azure Storage Ext Fri Aug 15, 2014 a Beta 138,634	Jorer 6 Preview 3 at 5:00 PM 32 ratings e

Azure Storage Explorer ツールでは、次のように操作することで、SAS を生成することができます。



[Generate Signature] で生成した SAS (sv= の部分) を、CREATE CREDENTIAL ステートメントの SECRET に貼り付ければ、次のようにブロック BLOB へのバックアップを行えるようになります。

出力された sv= 以降をコピーして SECRET に貼り付け ECREATE CREDENTIAL [<u>bttps://matustorage.blob.core.wind</u> WITH IDENTITY='Shared Access Signature' ,SECRET='sv=2014-02-14&sr=c&sig=KdUtNIp4a7%2FOHs	ows.net/cont1] IzahdF1cPcf rzwipVJP81009gqZC1%3D&st=20
100 % - < 通 メッセージ コマンドは正常に完了しました。	生成した SAS を 貼り付け

BACKUP DATABASE NorthwindJ TO URL = ' <u>https://matustorage.blob.core.windows.ne</u> WITH COMPRESSION, STATS	at/cont1/North.bak'
100 % - <	
シッセージ	ブロック BLOB
10 バーセント処理されました。 20 バーセント処理されました。 31 バーセント処理されました。 41 バーセント処理されました。 51 バーセント処理されました。 60 バーセント処理されました。 80 バーセント処理されました。 80 バーセント処理されました。 91 バーセント処理されました。 データベース 'Northwindd' の 384 ページ、ファイル 1 100 バーセント処理されました。 データベース 'Northwindd' の 2 ページ、ファイル 1 BACKUP DATABASE により 386 ページが 5.649 秒間で正常	へのハックアッフ のファイル 'NorthwindJ' を処理しました。 ンファイル 'NorthwindJ log' を処理しました。 創に処理されました (0.533 MB/秒)。

◆ Azure PowerShell を利用して、SAS を生成する場合

SAS は、Azure PowerShell を利用して生成することもできます。Azure PowerShell は、次の URL からダウンロード/インストールすることができます。 http://go.microsoft.com/fwlink/p/?linkid=320376&clcid=0x411

Windows Azura Po	werShell					
Windows Azure Po	wersnen		Web Platform Installer 5.0		I	
	The Microsoft Azu deploying and ma 'Switch, AzuroMoo	ire PowerShell maging Micro	前提条件	インストール	構成	終
	詳細情報 発行元: Microsoft ダウンロードサイズ:	Corporation 19.50 MB	インストールされるサードハ 追加のソフトウェアと、有効 ンおよび製品は、一覧に証 ェアに対する権利をお客様 る必要があります。	ペーティ製のアプリケーション ソフトウ: Jになる Windows コンボーネントの) 3載されているサード パーティによって に付与しません。お客様自身で該	ェア、Microsoft 製品とコンボーネント 欠の一覧を確認してください。サード :提供されています。Microsoft は、t 当するサード パーティのライセンス条:	、および下で識別され」 パーティ製のアプリケーシ ナード パーティ製のソフト 項を探して読み、同意す
	バージョン: 0.9.8.1 リリース日: 2015年	10月13日	X Windows Azure <u>ライセンス条項の</u> ファイル <i>タ</i> ウンロード	PowerShell (英語) 表示 :サイズ合計:		直接ダウンロードリンク 19-5 MB
			<u>インストールされる追加のソ</u> [同意する]をクリックすると れた追加のソフトウェアのう	/ フトウェアを表示し、関連する Mic 、上に記載されているサードパーテ イセンス条項に同意したことになりま	rosoft ライセンス条項を確認するに((製のソフトウェア、Microsoft ソフト! す。すべてのライセンス条項に同意:	<u>よ、ここをクリックします。</u> ウェア、および上で識別: できない場合は、[同意
			40 J & J 99 J 99 C C 220 a		同意しない(D)	2 同意する(A)
インストールするアイテム(T) atform Installer 5.0			オブショ		終了但	
提条件	インストール		構成	終了		
		Web	Platform Installer 5.0			×
	P d n E	licrosoft latform 1 eploy an etwork c asily crea	前提条件 ノ 次の製品が正常にイン Windows Azure Powe	インストール ンストールされました。 rShell	構成	終了
レード進行状況: Windows Az	zure PowerShell	y it free				
インストールしています						

インストール完了後は、[スタート] メニューの [すべてのアプリ] から [Microsoft Azure] の

[Microsoft Azure PowerShell] をクリックして、起動することができます。



Azure PowerShell が起動したら、次のようにコマンドを記述すれば、SAS を生成することができます。

\$context = New-AzureStorageContext -StorageAccountName ストレージ アカウント名 -StorageAccountKey アクセスキー New-AzureStorageContainerSASToken -Name コンテナー名 -Permission rwdl -FullUri -Context \$context

New-AzureStorageContext で、「-StorageAccountName」にストレージ アカウントの名前 を指定して、「-StorageAccountKey」にストレージ アカウントにアクセスするためのアクセス キーを指定します。

次に、**New-AzureStorageContainerSASToken** で「**-Name**」でバックアップ先となるコンテ ナー名を入力します。

27 管理者:Microsoft Azure PowerShell Windows PowerShell Copyright (C) 2015 Microsoft Corporation. All rights reserved. For a list of all Azure cmdlets type 'help azure'. For a list of Windows Azure Pack cmdlets type 'Get-Command *wapack*'. For Node, is cmdlets type 'help node-dev'.	Azure PowerShell での SAS の生成例
For PHP cmdlets type 'help php-dev'. PS C:¥> PS C:¥> \$context = New-AzureStorageContext -StorageAccountName matustorage -Stor: wHA==	ageAccountKey giNGfC+47 zikst0
PS C:¥> PS C:¥> New-AzureStorageContainerSASToken -Name cont1 -Permission rwdl -FullUri https://matustorage.blob.core.windows.net/cont1 <mark>?sv=2015-02-21&sr=c&sig=0rdKqo5LR</mark> pc.cvX	-Context \$context OKOncXY5N8ZHcJZZhc%2Bw50eKcApobF1zIg%3D&se=2015-12
r3 6:#2 SV= の 部分をコピーす:	3

生成された SAS (sv= の部分) を、CREATE CREDENTIAL ステートメントの SECRET に貼り 付ければ、ブロック BLOB へのバックアップを行えるようになります。

Azure PowerShell でのコマンド例や、バックアップ URL の詳細については、オンライン ブック

の以下のトピックが参考になると思います。

SQL Server Backup to URL

http://msdn.microsoft.com/en-us/library/dn435916.aspx

	Create a Shared Access Signature				
SQL Server Backup to URL Use PowerShell to Backup Multiple Databases to Microsoft Azure Blob Storage Service	SQL Server Backup to URL Use PowerShell to Backup Multiple Databases to Microsoft Azure Blob Storage Service				
 SQL Server Backup to URL Best Practices and Troubleshooting 	1. Ad hoc Shared Access Signature				
File-Snapshot Backups for Database Files in Azure Managed Backup to Microsoft Azure	<pre>\$context = New-AzureStorageContext -StorageAccountName <mystorageaccountname> -StorageAccoun New-AzureStorageContainer -Name <mystorageaccountcontainername> -Context \$context New-AzureStorageContainerSASToken -Name <mystorageaccountcontainername> -Permission rwdl -Fu </mystorageaccountcontainername></mystorageaccountcontainername></mystorageaccountname></pre>				
Restoring From Backups Stored in Microsoft Azure	<pre>2. Shared Access Signature that is associated with a Stored Access Policy \$\$ubscriptionName='(mysubscriptionname)' \$StorageAccountName='(mystorageaccountcame>' \$ContainerName='(mystorageaccountcatinername>' \$Gelet-AzureSubscription-SubscriptionName \$subscriptionName \$accountKeys = Get-AzureStorageContext -StorageAccountName \$storageAccountName \$storageContext = New-AzureStorageContext -StorageAccountName \$storageAccountName \$storageContext = New-AzureStorageContext -StorageAccountName \$storageAccountName \$container = New-AzureStorageContext -StorageAccountName \$storageAccountName \$cotainer = New-AzureStorageContext -StorageAccountName \$storageAccountName \$cotainer = New-AzureStorageContext = Note \$storageContext -Name \$containerName \$cotainer = New-AzureStorageContext = StorageAccountName \$storageAccountName = StorageAccountName = \$cotainer.CloudBlobContainer \$permissions = \$cbc.GetPermissions(); \$policy = new-object 'Microsoft.WindowsAzure.Storage.Blob.SharedAccessBlobPolicy' \$policy.SharedAccessStartTime = \$(Get-Date).ToUniversalTime().AddYears(5) \$policy.SharedAccessStartTime = \$(Get-Date).ToUniversalTime().AddYears(5) \$policy.Permissions = "Read,Write,List,Delete" \$permissions.SharedAccessSignature \$storage.Blob.SharedAccessBlobPolicy' \$cbc.SetPermissions(\$permissions); \$policy = new-object 'Microsoft.WindowsAzure.Storage.Blob.SharedAccessBlobPolicy' \$sas = \$cbc.GetSharedAccessSignature(\$policy, \$policyName) Write-Host 'Shared Access Signature '\$(ssas.Substring(1))''</pre>				
	<pre>\$TSQL = "CREATE CREDENTIAL [{0}] WITH IDENTITY='Shared Access Signature', SECRET='{1}'" Write-Host \$TSQL</pre>				

4.5 PolyBase で Hadoop アクセス (Hortonworks、HDInsight)

PolyBase は、**Hadoop ファイル システム** (HDFS: Hadoop File System) 上にあるデータを、 SQL Server からアクセスできる機能です(この機能は、元々 SQL Server Parallel Data Warehouse: PDW で提供されていた機能ですが、SQL Server 2016 からは、通常の SQL Server でも利用できるようになりました)。

PolyBase を利用すれば、Microsoft Azure 上の Hadoop サービスである「Microsoft Azure HDInsight」や、Hortonworks の提供する「HDP : Hortonworks Data Platform」、Cloudera の提供する「CDH : Cloudera's Distribution including Apache Hadoop」上のデータに対し て、SQL Server からアクセスできるようになります。また、PolyBase では、Microsoft Azure 上の「BLOB ストレージ」にもアクセスすることができます。

◆ PolyBase の利用イメージ ~CREATE EXTERNAL TABLE~

PolyBase では、**CREATE EXTERNAL TABLE** ステートメントを利用することで、外部データ ソースである HDFS/Hadoop サービス上にあるファイル(1つまたは複数のファイル)を、SQL Server 上のテーブルであるかのように操作できるようになります(外部データ ソースに対して、 テーブル定義を割り当てることができ、SQL Server 上のテーブルのように扱えます)。

次の画面は、PolyBase を利用して、IP アドレス 192.168.1.36 のマシン(CentOS 7 をインス トールして、Hortonworks Data Platform: HDP で HDFS を稼働中の Linux マシン)にアクセ スしているときの様子です。

🕵 PolyBaseテスト_02_hortonworks_2_Cent.sql - WIN10.	testdb1 (WIN10¥matu_tak (53)) - Microsoft SQL Server Management Studio (管理者)
ファイル(F) 編集(E) 表示(V) クエリ(Q) プロジェクト(P)	デバッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)
: : 🛅 = 📨 - 📂 🚚 🗊 🔔 新しいクエリ(N) 📑 👒	📸 🖓 🐁 💼 🕵 🔊 • (* • 💷 • 🖳 📖 🕨 🔹 - 👘 • 👘
: 書 ట testdb1 • ! 実行(X)	▶ テノミック(D) = ✔ 認 ■ 目 認 認 吗 図 領 ◎ 目 Ξ 일 課 課 橋 -
オブジェクト エクスプローラー 🔹 🕂 🗙	PolyBaseテスト_02_ho10¥matu_tak (53)) ×
接続 📲 🛃 🔲 🍸 🧟	EVEC on configure 'hadeen connectivity' 7:
□ 10 WIN10 (SQL Server 13.0.700 - WIN10¥matu ∧ □ = データパーフ	RECONFIGURE
□ □ 「->ハ-× □ □ システム データベース	
□ データベース スナップショット	UREATE DATABASE testdb1
DWConfiguration	go USE testdb1
🗉 🧻 DWDiagnostics	COL (Establ
🗉 间 DWQueue	CREATE EXTERNAL DATA SOURCE hdp1
🗉 🔰 NorthwindJ	WITH (TYPE = HADOOP,
ReportServer	LOCATION = hdts://192.168.1.36:80201)
ReportServerTempDB Intertidb1	ODEATE EVTEDNAL ETLE CODMAT #1
■ 📴 データベース ダイアグラム	WITH (FORMAT TYPE = DELIMITEDTEXT.
	FORMAT_OPTIONS (FIELD_TERMINATOR =','
🗉 🚞 システム テーブル	,USE_TYPE_DEFAULT = TRUE))
🕀 🚞 FileTables	
🖃 🧰 外部テーブル	UKEATE EXTERNAL TABLE TESTT
🖃 📑 dbo.test1	h int)
□ □ 列	WITH (LOCATION='/input/test1',
a (Int, NULL)	DATA_SOURCE = hdp1,
	FILE_FORMAT = f1)
	SELECT & EDOM toot1
□ 🗀 外部リソース	SELECT * FROM LEST
🖂 🧰 データソース	100 % 👻 <
🗾 hdp1	TT 結果 Factoria
🖃 🧰 ファイル形式	
G Service Broker	2 2 222
□ □ ストレージ	

CREATE EXTERNAL DATA SOURCE ステートメントで「**TYPE=HADOOP**」と指定して、 **LOCATION** に IP アドレスと HDFS アクセスのためのポート番号 (HDP インストール時の既定 値の **8020** を利用)して、外部データ ソースを定義しています。

次に、CREATE EXTERNAL FILE FORMAT ステートメントでファイルのフォーマット(上の例 では FIELD_TERMINATOR に , を指定して CSV ファイルを指定)しています。

最後に、CREATE EXTERNAL TABLE ステートメントで、DATA_SOURCE に最初に作成した外 部データ ソース、FILE_FORMAT に上で定義したファイル フォーマットを指定して、 LOCATION に HDFS 上のファイルまたはフォルダーへのパス(上の例では input フォルダー 配下の test1 ファイル)を指定することで、HDFS 上のファイル(パスにフォルダーを指定した 場合は、フォルダー配下のファイル群)を、SQL Server のテーブルであるかのように利用するこ とができます。

このようにテーブル定義を設定することで、外部データ ソースであるにも関わらず、SQL Server 上に存在するテーブルであるかのように操作でき、「SELECT * FROM <外部データ ソース>」の ように SELECT ステートメントでアクセスできるようになります。

なお、この例で利用した Hortonworks Data Platform (HDP) は、次のようにインストールして います。

🌣 アプリケーション 場所 🛃 refo	xx ウェブ・ブラウザ	ja 📢 昱 水曜日 18:26 🖾 cent
home		
	Ambari - Cluster Install Wizard - Mozilla Firefox	- 0 X
🖉 Ambari - Cluster 🗙 🔶		
Ce localhost:8080/#/installer/ste	ep1 ~ C Q 核素	
Ambari		admin 🕶
CLUSTER INSTALL WIZARD Get Started	Select Stack	
Install Options Confirm Hosts Choose Services Assign Masters Assign Slaves and Clients Customize Services Review Install, Start and Test Summary	Stacks HDP 2.3 HDP 2.2 HDP 2.1 HDP 2.0 Advanced Repository Options - Back	Next -
		CENTOS
E cent@localhost:"/tmp	🥹 Ambari – Cluster Install Wizard …	1 / 4 🕚

SQL Server 2016 CTP 3.2 自習書 No.1 新機能ダイジェスト

) 🕲 localhost:8080/#/main.	/dashboard/metrics		~ C	Q. 検索	☆ 自 ♣ 侖
À Ambari 🛛 hdp1 🐻	a alerta		Dashboard	Services Hosts Alerts	Admin 🌐 🛓 admin 🚽
9 HDFS	Metrics Heatmaps Co	ontig History			
MapReduce2	Metric Actions -				
2 ZooKeeper	HDFS Disk Usage	DataNodes Live	HDFS Links	Memory Usage	Network Usage
Actions -		474	NameNode Sacondary Marco Node	No Data Available	No Data Available
	48%	1/1	1 DataNodes		
	CPU Usage	Cluster Load	NameNode Heap	NameNode RPC	NameNode CPU WIO
	No Data Available	No Data Available			
			6%	0.50 ms	n/a
	NameNode Uptime	ResourceManager	ResourceManager Uptime	NodeManagers Live	YABN Memory
	24.6 min	12%	22.4 min	1/1	0%



→ PolyBase を利用する前提条件

SQL Server 2016 CTP 3.2 で PolyBase を利用するには、次の前提条件があります。

- x64 版の SQL Server 2016 CTP 3.2 (Evaluation edition)
- .NET Framework 4.0
- 64 ビット版の Oracle JRE (Java SE RunTime Environment) version 7.51 以上
- 最小メモリ要件 4GB、最小 HDD 容量要件 2GB

PolyBase は、**64 ビット環境**のみでサポートされて、**64 ビット版の JRE 7.51**(JRE 7 Update 51)以上が必要になります。

JRE 7.51 以上をインストールしていない場合には、SQL Server 2016 のインストール時の [機 能ルール] ページで、次のようにエラーが表示されて、先に進むことができなくなります。

髋 SQL Server 2016 CTP3.1 セットアップ	î		_		×	
機能ルール						
インストール処理がブロックされるかる	どうかを判断するために、ルールを閉	実行しています。 詳細 については、 [ヘルプ] をクリックしてくだ	さい。			
プロダクトキー ライセンス条項 グローバル ルール Microsoft Update 製品の更新プログラム セットアップ ファイルのインストール インストール ルール セットアップ ロール 機能の遅れ	操作が完了しました。成功 4、 詳細の非表示(S) << 詳細レポートの表示(V) ルール ② 以前の Visual Studio 20 ④ Microsoft .NET Framew	失敗 1、警告 0、スキップ 0。 100 のインスタンスは、更新する必要があります。 vork 3.5 Service Pack 1 が必要です	状態 <u>合格</u>	再実	īτ(<u>B</u>)	
機能ルール	🔇 Oracle JRE 7 Update 51 (64 ビット) 以上が必要					
サーバーの構成 サーバーの構成 データベース エンジンの機成 PolyBase の構成 機能構成ルール	 PolyBase は各コンピュータ KB2919355 インストール 	- で 1 つの SQL Server インスタンスにのみインストールで Jルール チェックの結果 メールール "Oracle JRE 7 Update 51 (64 ビット アのコンピューターには Oracle Java SE Runt	<u>合格</u>) 以上が必要	!" は失敗しまし ment Version	×	
1ノストールの単何先了 インストールの進行状況 完了		LU Jンビューティによりなステム NUTICHE Environment version Update 51 (64 ビット) 以上がインストールされていません。Oracle Java 5 Runtime Environment はサードパーティが提供するソフトウエアです。M では、このようなサードパーティレフトウェアに対する権利を付与しておりま ザー自身がサードパーティによって適用されるライセンス条項を採し、内容 承諾する義務があります。続行するには、 http://go.microsoft.com/flwink/Linkid=526030 で Oracle SE Jav Runtime Environment をダウンロードしてください。				
		l			OK	

JRE(Java SE Runtime Environment)は、執筆時点での最新版である「**JRE 8u66**」や、Java の開発者キットである **JDK**(Java Development Kit)の「**JDK 8u66**」を入れておけば、PolyBase をインストール/動作させることができます(64 ビット版の JRE 7 Update 51 以上を利用する ことで動作させられます)。これらは、次の URL からダウンロードできます。

Java SE Runtime Environment のダウンロード URL(JRE 8u66 など) http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

Java Development Kit のダウンロード URL(JDK 8u66 など) http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

ダウンロード後は、.exe ファイルをダブルクリックすれば、JRE/JDK をインストールすること
ができます。

↓ ⊘ _ 	アブリケーション ツール ダウンロード 特有 表示 管理	- E	JDK 8u66 を インストールして いるときの様子
 マイックアクセス ⇒ ブイックアクセス ニ デスクトップ ↓ ダウンロード ◎ ジャンパー 	名前 ^] 余前 /] # [] jdk-8u66-windows-x64.exe : #]	▼ 0 9970-F00使来 更新日時 種類 サイズ 2016/01/08 13:31 アプリケーション 191,135 KB	
 □ F+1×>F ■ どクチャ ■ ど方オ ♪ ミュージック ▲ OneDrive 	Java SE Development Kit 8 Update 66のインストール・ウィザードへようこそ	 Java SE Development Kit 8 Update 66 (64-bit) - カスタ Java Java インストールするオブション機能を次のリストから選択して(ださ) の「プログラムのの自知と削除」を使用して、機能の選択を変更 	ム・セットアップ ×
	このウィザードでは、Java SE Development Kit 8 Update 66のインストール・プロ って説明します。	せ ・ ・ ・ ・ ・ ・ 、 ジール シース・コード ・ ・ ・ 、 ジース・コード ・ ・ ・ 、 パブリックJRE	機能の説明 JavaFX SDK、プライベートJREおよ UJava Mission Controlツール・ス イートを含むJava SE Development Kit 8 Update 66 (64-bit)、ハード・ドライブに180MB が必要です。
געייב א	Java Mission Controlプロファイリングおよび話を断が一ル・スイートは、JDKの一部 るようになりました。 次M >	「 インストール先: C:¥Program Files¥Java¥jdk1.8.0_66¥ <戻る(B)	変更(_) 次(!) > 取消

→ PolyBase のインストール方法

PolyBase をインストールするには、SQL Server 2016 CTP 3.2 のインストール時に、次のよう に [機能の選択] ページで、[データベース エンジン サービス] の [外部データ用 PolyBase ク エリ サービス] をチェックします。

11 SQL Server 2016 CTP3.1 セットアップ		- 🗆 ×
機能の選択 インストールする Evaluation 機能を	z運択します。	
プロダクトキー ライセンス条項 グローバルルール Microsoft Update 製品の更新プログラム セットアップファイルのインストール インストールルール セットアップロール 機能の選択 機能ルール インスタンスの構成 サーバーの構成 データベースエンジンの構成 PolyBase の構成 機能構成ルール インストールの準備完了 インストールの進行状況 完了	 機能(E): インスタンス機能 データペース エンジン サービス GSL Server レブリケーション Advanced Analytics Extensions 検索のためのフルテキスト抽出とセマンティック抽出 Data Quality Services メ卵デーク用 PolyBase クエリ サービス Analysis Services メ卵デーク用 PolyBase クエリ サービス Analysis Services Reporting Services - ネイティブ キ萄糖能 Reporting Services - ネイティブ SharePoint 製品用 Reporting Services アドイン Data Quality Client すべて輩択解除(U) インスタンス ルート ディレクトリ(B): C:¥Program Files¥N 共有機能ディレクトリ (x86)(X): C:¥Program Files (N 	機能の説明: SQL Server インスクソスのインスタンス機能は、構 成と操作が、他の SQL Server インスタンスとは分 離されています。SQL Server の複数のインスタン スを、同じコンビューターでサイドパイサイドで操作 、 選択した機能に必要なコンボーネント(D): インストール済み: Windows PowerShell 2.0 Microsoft NET Framework 3.5 く 必要なディスク領域(D) 下ライブ C: 2497 MB 必要、103191 MB 使用可 能 Alicrosoft SQL Server¥ 80)¥Microsoft SQL Server¥
	< 戻る(<u>B</u>)	次へ(<u>N</u>) > キャンセル ヘルプ

SQL Server 2016 のインストール時に PolyBase を選択した場合の注意点は、次の [サーバーの 構成] ページの [サービス アカウント] タブでのサービス アカウントの設定画面です。

1 SQL Server 2016 CTP3.1 セットアップ				-		×	
サーバーの構成							
サービス アカウントと照合順序の構成	を指定します。						
プロダクト キー ライセンス条項 グローバリーリーリ	サービス アカウント 照合順序 各 SQL Server サービスに別々のアカウ	ントを使用することをお勧めしま	≢(<u>M</u>)				
Microsoft Update	サービス	アカウント名	パスワード	スタートアッ	プの種類		
。 製品の更新プログラム	SQL Server エージェント	NT Service¥SQLSERVER		手動	~		
セットアップ ファイルのインストール	SQL Server データベース エンジン	NT Service¥MSSQLSERV	<u></u>				
インストール ルール	SQL Server Analysis Services	NT Service¥MSSQLServ	PolyBase の	サービス ア ーザーを利	7カウン 田する#	トに \ き合け	Nindows の
セットアップ ロール	SQL Server Reporting Services	NT Service¥ReportServer	ロージル 名¥ユーザ	一名」形式	で指定で	する必	、「マンン」
機能の選択	SQL Server Integration Services 13.0	NT Service¥MsDtsServer			×		
機能ルール	SQL Server PolyBase Engine	NT AUTHORITY¥NETW 1		自動	\sim		
インスタンスの構成	SQL Server PolyBase Data Movem	NT AUTHORITY¥NETW 1	1	自動	~		
サーバーの構成	SQL Server スタート パッド	NT Service¥MSSQLLaun		自動			
データベースエンジンの構成	SQL フルテキスト フィルター デーモン ラ	NT Service¥MSSQLFDLa		手動			
Analysis Services の構成	SQL Server Browser	NT AUTHORITY¥LOCAL		無効	\sim		
Reporting Services の構成 PolyBase の構成 機能構成ルール インストールの準備完了 インストールの進行状況	SQL Server データベース エンジン サー この特権が付与されると、データ ペー ます。削除されたコンテンツが承認さ つながる可能性があります。詳細につ	-ビスにボリューム メンテナンス タ ジが完全に消去されなくなり、 ~ へていない プリンシパルによってア らいては、こちらをクリックしてくだ。	スクを実行する特権 インスタント ファイルの クセスされることがあ さい。	を付与する(<u>C</u> 初期化が可 るため、情報)) 能になり 漏えいに		
		< 戻る(<u>B</u>) 次へ(j	<u>N</u>) > = ++>/t	2.11	ヘルプ		

PolyBase は、2 つのサービスとして追加されますが、既定ではサービス アカウントとして「NT AUTHORITY¥NETWORK SERVICE」が選択されています。このサービス アカウントを、 Windows のローカル ユーザーに変更する場合には、「マシン名¥ユーザー名」形式で(ユーザー名 だけでなく、¥マーク付きでマシン名を付与して)指定する必要があります(ユーザー名だけだと インストールの最後のフェーズでエラーになってしまうので注意してください)。

また、PolyBase を選択している場合は、SQL Server 2016 のインストール時に、次のように [PolyBase の構成] ページが表示されます。

髋 SQL Server 2016 CTP3.1 セットアッフ	Ŧ	- 🗆 X
PolyBase の構成 PolyBase スケールアウトのオプショ	ソとポート範囲を指定します。	PolyBase を 1台で構成するなら こちらで OK
プロダクトキー ライセンス条項 グローバルルール Microsoft Update 製品の更新プログラム セットアップファイルのインストール インストールルール セットアップロール 機能の選択 機能ルール インスタンスの構成 サーバーの構成 データベースエンジンの構成 Analysis Services の構成 Reporting Services の構成 Reporting Services の構成 機能構成ルール インストールの準備完了 インストールの進行状況	 スタンドアロンの PolyBase 対応インスタンスとして、この SQL スタンドアロンのヘッド ノードとしてこの SQL Server インスタン PolyBase スケールアウト グループの一部としてこの SQL Ser PolyBase スケールアウト グループの一部としてこの SQL Ser アクリアションを選択します。このオブションを選択すると、この ベース エンジン、SQL Server PolyBase サービスおよび SQL ファイアウォール接続が有効したが、MSDTC レジストリ設定 PolyBase サービスのボート範囲を指定します: 16450-16460 	L Server を使用します。 バスを使用するには、このオプションを選択します。 ver を使用します。 ノードとしてこの SQL Server インスタンスを使用するには、こ リンビューター上でファイアウォールが開き、SQL Server データ Browser への著信接続が可能になります。また、MSDTC を変更できます。
		< 戻る(<u>B</u>) 次へ(<u>N</u>) > キャンセル

このページでは、PolyBase を1台で構成するのか、複数台(スケールアウト)で構成するのかを

設定しますが、1 台で構成する場合には [スタンドアロンの PolyBase 対応インスタンスとして、 SQL Server を使用します。] をチェックすれば大丈夫です。

✤ PolyBase の利用方法

PolyBase を利用するには、まず **sp_configure** で「**hadoop connectivity**」構成オプションを 設定します。

EXEC sp_configure 'hadoop connectivity', 構成値 RECONFIGURE

構成値	接続可能な HDFS/Hadoop サービス
0	Hadoop 接続の無効化
1	Hortonworks HDP 1.3 on Windows Server、 Azure blob storage (WASB[S])
2	Hortonworks HDP 1.3 on Linux
3	Cloudera CDH 4.3 on Linux
4	Hortonworks HDP 2.0 on Windows Server、 Azure blob storage (WASB[S])
5	Hortonworks HDP 2.0 on Linux
6	Cloudera 5.1 on Linux
7	Hortonworks 2.1 and 2.2 on Linux、 Hortonworks 2.2 on Windows Server、 Azure blob storage (WASB[S])

PolyBase は、**HDP** (Hortonworks Data Platform) と **CDH** (Cloudera's Distribution including Apache Hadoop) の 2 つの Hadoop プロバイダーをサポートしています。この項の冒頭の例で は、**HDP 2.3** (Hortonworks Data Platform) に接続するスクリプトを紹介しましたが、これは構成値を「**7**」に設定していました。

```
EXEC sp_configure 'hadoop connectivity', 7
RECONFIGURE
```

なお、PolyBase を利用して、Azure BLOB ストレージに接続する場合も、「7」に設定しておけ ば接続することができます(7 の他に、1 と 4 でも接続可能)。

以降では、冒頭の例で **HDP 2.3**(Hortonworks Data Platform)に接続したスクリプトを、改めて 1つ 1つ説明していきます。

```
USE testdb1

/*

PolyBase で IP アドレス 192.168.1.36 のマシンに接続する例

(CentOS 7 上に HDP3.2 をインストールした Linux マシン) */

CREATE EXTERNAL DATA SOURCE hdp1 WITH

( TYPE = HADOOP,

LOCATION =' hdfs://192.168.1.36:8020'

)
```

PolyBase では、データベース内に外部データ ソース(EXTERNAL DATA SOURCE)を作成する ので、USE ステートメントでデータベースに接続してから、CREATE EXTERNAL DATA SOURCE ステートメントを実行しています。このステートメントでは、「TYPE=HADOOP」と指 定して、LOCATION に IP アドレスと HDFS アクセスのためのポート番号(HDP 3.2 をインス トールした時の既定値の 8020 ポートを利用)して、外部データ ソースを定義しています。

このように作成した外部データ ソースは、Management Studio のオブジェクト エクスプローラ ーで、次のように [**外部リソース**]の [**データ ソース**] を展開して確認することができます。



次に、**CREATE EXTERNAL FILE FORMAT** ステートメントでファイルのフォーマットを指定します。

```
CREATE EXTERNAL FILE FORMAT f1
WITH (
FORMAT_TYPE = DELIMITEDTEXT,
FORMAT_OPTIONS (FIELD_TERMINATOR =','
, USE_TYPE_DEFAULT = TRUE)
```

この例では FIELD_TERMINATOR に , を指定して CSV ファイルを指定しています。この例で アクセスするファイルは、次のように CSV ファイルとして作成しています (HDP 3.2 の Hadoop ファイル システム内に test1 という名前で、カンマ区切りのファイルとして作成)。

	ŀ	dfs@localh	ost:~		_ B X
ファイル(F) 編集(E)	表示(V) 検索(S)	端末(T) ,	ヘルプ(H)		HDFS(Hadoop ファイル システム) に CSV ファイルを配置(put)
[hdfs@localhost ~]\$ 1,111 2.222	cat test1	CSV ファ	イル	C	
[hdfs@localhost ~]\$ [hdfs@localhost ~]\$ Found 1 items	hadoop fs -put hadoop fs -ls I	test1 hdf hdfs://192	s://192.1 .168.1.36	68.1.36 :8020/ir	:8020/input/ nput/
-rw-rr 3 hdfs	hdfs 13	2 2015-11-	12 03:44	hdfs://	192.168.1.36:8020/in
[hdfs@localhost ~]\$					
					CentOS 7 ターミナル ツールで HDP を操作

外部データ ソースとファイル フォーマットを定義した後は、、CREATE EXTERNAL TABLE ス テートメントで外部テーブルを定義します。

```
CREATE EXTERNAL TABLE dbo.test1 (

a int NULL,

b int NULL

)

WITH (LOCATION='/input/test1',

DATA_SOURCE = hdp1,

FILE_FORMAT = f1,

REJECT_TYPE = VALUE,

REJECT_VALUE = 0)
```

このステートメントでは、DATA_SOURCE に最初に作成した外部データ ソース(ここでは hdp1)、FILE_FORMAT に上で定義したファイル フォーマット(ここでは f1)を指定して、 LOCATION に HDFS 上のファイルまたはフォルダーへのパス(ここでは input フォルダー配 下の test1 ファイル)を指定します。このように、作成した外部テーブルは、オブジェクト エク スプローラーでは次のように表示されます。



外部テーブルも、SQL Server のテーブルであるかのように表示されます。また、SELECT ステー トメントで操作することもできるので、次のように検索することができます。

SELECT * FROM test1

	SELECT *	FROM test	1	
100 %	• <			
🛄 ñ	課 🛅 メッ	2ージ		
	a b			
1	1 111			
0	2 222			

HDFS 上の CSV ファイル (input フォルダーの test1 ファイル) の中身を参照できることを確認できます。このように、PolyBase を利用することで、HDFS 上のファイル (パスにフォルダーを指定した場合は、フォルダー配下のファイル群)を、SQL Server のテーブルであるかのように利用することができるのが大きなメリットです。

◆ 参考: HDP (Hortonworks Data Platform)を簡単に試す方法

HDP (Hortonworks Data Platform)を試したい場合は、簡単に試すことができる仮想マシン(す べてのソフトウェアがインストール/構成されて、起動するだけで利用できる状態のもの)が Sandbox (お試し環境) として提供されています。これは、次の URL からダウンロードすること ができます。

HDP(Hortonworks Data Platform)の Sandbox(仮想マシン)のダウンロード URL http://hortonworks.com/products/hortonworks-sandbox/#install

		J	10 1 = ٢7		
Hortonworks Why Hortonworks 製品 カスタマー	- ソリューション	トレーニング	サービス	Developers	始める
Hortonworks Data P 企業向けに設計された、100%オープンソースの	Platfori Apache Hado	M pop データプΞ	ラットフ	<i>д</i> -к	
概要 2.3 における新機能 ダウンロード&インストール	ドキュメンテーショ	ン アドオン機能	能 アー	カイブ	
ダウンロード&インストール Try HDP 2.3.2 with Hortonworks Sandbox					
HDP 2.3.2 on Hortonworks Sandbox Runs on VirtualBox or VMware Try out the very latest features and functionality in Hadoop and its' ecosystem of projects with <u>HDP 2.3</u> . Follow the <u>Step by Step</u> <u>Tutorials</u> . System Requirements Installation Steps Release Notes	INSTALL GUIDES for Mac	VirtualBox <u>: & Windows</u> & VMware <u>: & Windows</u> &		for VirtualB4 (HDP 2.3.2-8.5) for VMwan (HDP 2.3.2-8.7)	DX GB) GB)
HDP 2.3.2: Ready for the enterprise					
Automated (with Ambari 2.1) RHEL/CentOS/SLES/Ubuntu/Debian (64-bit) The recommended way to set up HDP for a production environment. <u>Apache Ambari</u> simplifies the provisioning, management and monitoring of your cluster.	INSTALL GUIDES Upg Amt	omated Install Guide = <u>HTML</u> prade Guide = <u>HTML</u> pari Release Notes	Do	ownload links within o <u>Go straight to t</u> r	documentation le page

PolyBase の参考情報

Getting started with PolyBase (構成方法など) https://msdn.microsoft.com/en-us/library/mt163689.aspx



PolyBase のヘルプのトップページ

https://msdn.microsoft.com/en-us/library/mt143171.aspx



STEP 5. 既存機能の強化

この STEP では、「SQL Server 2014 からの主な変更点」や「ライブ クエリ統 計」、「クエリ ストア」、「AlwaysOn の拡張」、「T-SQL の強化」、「BI 関連機能 の強化」など、SQL Server の既存機能を強化する新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。

- ✓ SQL Server 2014 からの主な変更点
- ✓ ライブ クエリ統計 (Live Query Statistics)
- ✓ クエリ ストアで性能監視、プラン固定
- ✓ T-SQL の強化
- ✓ AlwaysOn 可用性グループの拡張(自動フェールオーバー数が 2 から 3、
 ラウンドロビン レプリカ、ログ転送性能の向上、DTC 対応など)
- ✓ BI 関連の強化

Reporting Services の強化 (Datazen 統合など)、 Analysis Services の強化、SSIS (Integration Services)の強化など

5.1 SQL Server 2014 からの主な変更点

SQL Server 2014 からの主な変更点は、次のとおりです。

- データベースの互換性レベル **130** の提供
- サービス アカウントに「**ボリュームの保守タスクを実行**」を簡単に付与可能に
- tempdb の拡張
 - tempdb の複数ファイル/初期サイズをセットアップ時に選択できるようになった
 - tempdb の性能に関連するトレースフラグ 1117 と 1118 が既定でオンになった
- トレースフラグ 4199 が既定でオンになった
- Active Directory のパスワード認証/統合認証を利用可能に

→ データベースの互換性レベル 130

SQL Server 2016 では、データベースを新しく作成したときの既定の互換性レベルは「**130**」に なりました(**130** は、SQL Server 2016 の内部バージョン番号である **13.0** という意味です)。

🧻 データベースのプロパティ - test	DB	-		\times
ページの選択	「雪 スクリプト マ 「雪 ヘルプ			
査査授				
i≦ ファイル Iff ファイル グループ	照合順序(C): Japanese_CI_AS			\sim
🚰 オプション				
🚰 変更の追跡				~
🚰 権限	互換性レベル(L): SQL Server 2016 (130)			\sim
■ 「「「「」」「「」」 「「」」 「」 「」 「」 「」 「」 「」 「」 「」	コンテイン(か)トの種類(T): SQL Server 2008 (100)			
■ ミラーリング	SQL Server 2012 (110) SQL Server 2014 (120)			
☆ トランザクション ログの配布	その他のオプション(O): SQL Server 2016 (130)			
📑 ሳ፲ሀ አኑዎ	2 2↓ □			_
	✓ FILESTREAM			•
	FILESTREAM ディレクトリ名			
	FILESTREAM 非トランザクション アクセス Off			
	 Service Broker 			
	Broker が有効 False			
	Broker の優先度の許可 False			
	Service Broker 識別子	-8018-4592	2f008d77	_
	✓ カーソル			
	コミットでカーソルを閉じる False			_
接続	既定のカーソル GLOBAL			- 11
サーハー: WTN 10	2 (1) 表記の中の基準になる中 2049			
0011410				_
接続				
WIN10¥matumoto	入れ子になったりUガーが右効 True			_
■見 接続のプロパティの表示	∨ 70 th			- 11
	ANSI NULL EE E Faise			_
·# <= +E /日	ANSI NULL 有効 False			- v
進行状況	2 桁表記の年の基準になる年			
準備完了				
		ОК	キャンセ	216

130 レベルでは、インメモリ OLTP での並列プランや、INSERT..SELECT でのマルチ スレッド、 列ストア インデックスでのバッチ モードの動作の違い(MAXDOP1 でもバッチ モードで動作す る)など、性能に関する大きな違いが出るので、130 レベルを利用することをお勧めします。SQL Server 2014 や 2012 など、古いバージョンの SQL Server で取得したバックアップを SQL Server 2016 上にリストアしたり、古いバージョンのデータベース ファイル (.mdf/.ldf)を SQL Server 2016 上にアタッチした場合には、そのバージョンの互換性レベルが保たれるので、130 レベルに上げても問題ないか、検証してみることをお勧めします。

なお、互換性レベルの詳細については、オンライン ブックの以下のトピックががお勧めです。

ALTER DATABASE Compatibility Level (Transact-SQL) http://msdn.microsoft.com/en-us/library/bb510680.aspx

<	Differences Between Compa Level 130	tibility Level 120 and
ALTER DATABASE SET Options (Transact-SQL)	This section describes new behaviors introduced	with compatibility level 130.
ALTER DATABASE Database Mirroring (Transact-SQL)	Compatibility-level setting of 120 or lower	Compatibility-level setting of 130
ALTER DATABASE SET HADR (Transact-SQL)	The Insert in an Insert-select statement is single-threaded	The Insert in an Insert-select statement is multi-threaded or can have a parallel plan.
ALTER DATABASE Compatibility Level (Transact-SQL)	Memory Optimized Table queries execute single-threaded.	Memory Optimized Table queries can now have parallel plans
	Introduced the SQL 2014 Cardinality estimator CardinalityEstimationModelVersion="120"	Further cardinality estimation (CE) Improvements with the Cardinality Estimation Model 130 which is visible from a Query plan. CardinalityEstimationModelVersion="130"
	Batch mode v/s Row Mode changes with Columnstore indexes	Batch mode v/s Row Mode changes with Columnstore indexes
	 Sorts on a table with Columnstore index are in Row mode Windowing function aggregates operate in row mode such as 	 Sorts on a table with a Columnstore index are now in batch mode Windowing aggregates now operate in batch mode such as LAG/LEAD

→ サービス アカウントに「ボリュームの保守タスクを実行」を簡単に付与可能に

SQL Server 2016 からは、インストール時に、サービス アカウントに対して、ユーザーの権利である「**ボリュームの保守タスクを実行**」を付与できるようになりました。

髋 SQL Server 2016 CTP3.1 セットアップ				-		×
サーバーの構成						
サービス アカウントと照合順序の構	成を指定します。					
プロダクト キー	サービス アカウント 照合順序					
ライセンス条項 グローバル ルール	各 SQL Server サービスに別々のアカウ:	ソトを使用することをお勧めします	≢(<u>M</u>)			
Microsoft Update	サービス	アカウント名	パスワード	スタートフ	アップの種類	1
製品の更新プログラム	SQL Server エージェント	NT Service¥SQLSERVER		手動	~	
セットアップ ファイルのインストール	SQL Server データベース エンジン	NT Service¥MSSQLSERV		自動	~	
インストール ルール	SQL Server Analysis Services	NT Service¥MSSQLServe		自動	~	
セットアップ ロール	SQL Server Reporting Services	NT Service¥ReportServer		自動	~	
機能の選択	SQL Server Integration Services 1	NT Service¥MsDtsServer		自動	~	
機能ルール	SQL Server PolyBase Engine	NT AUTHORITY¥NETW		自動	~	
インスタンスの構成	SQL Server PolyBase Data Movem	NT AUTHORITY¥NETW		自動	~	
サーバーの構成	SQL Server スタート パッド	NT Service¥MSSQLLaun		自動		
データベースエンジンの構成	SQL フルテキスト フィルター デーモン ラ	NT Service¥MSSQLFDLa		手動		
Analysis Services の構成	SQL Server Browser	NT AUTHORITY¥LOCAL		無効	~	
Reporting Services の構成 PolyBase の構成 機能構成ルール インストールの準備完了 インストールの進行状況	○ SQL Server データベース エンジン サー この特権が付与されると、データ ペー ます。削除されたコンテンツが承認さ つながる可能性があります。詳細につ	ビスにポリューム メンテナンス タ ジが完全に消去されなくなり、々 いていない プリンシパルによってア いては、こちらをクリックしてくださ	スクを実行する特 (ンスタント ファイ) クセスされることが さい。	権を付与する しの初期化が あるため、情	5(G) 可能になり 報漏えいに	
		< 戻る(<u>B</u>) 次へ(<u>1</u>	<u>v</u>) > +t;	ンセル	ヘルプ	

この権利は、**瞬時初期化**(.mdf ファイルのサイズが拡張するときに、瞬間的にサイズを拡張でき る機能)を有効化するために、以前のバージョンでは、手動で行っている作業でした(なお、サー ビス アカウントが Administrators グループのメンバーである場合には、この権利は自動的に付 与されています)。

◆ tempdb の拡張 ~ファイル数や初期サイズをセットアップ時に設定可能に~

SQL Server 2016 からは、SQL Server のインストール時に、tempdb データベースの設定を変 更できるようになりました(今までは、tempdb の設定を変更する際には、インストール後に別途 行う必要がありましたが、SQL Server 2016 からはインストール時に設定変更できるようになり ました)。これは、インストール時の[データベース エンジンの構成] ページで、次のように [TempDB] ページを開くことで設定できます。

髕 SQL Server 2016 CTP3.1 セットアップ			-		×
データベース エンジンの構成 データベース エンジンの認証セキュリラ	・ ティ モード、管理者、データ デ	イレクトリ、および TempDB の設定を指定します。			
プロダクトキー ライセンス条項 グローバルルール Microsoft Update 製品の更新プログラム セットアップ ファイルのインストール インストールルール セットアップ ロール 機能の選択 とのアイルの 初期サイズ ペロール インスタンスの構成 サーバーの構成 データベースエンジンの構成 Reporting Services の構成 Reporting Services の構成 Reporting Services の構成 Reporting Services の構成 W能構成ルール インストールの準備完了 インストールの進行状況	サーバーの構成 データディ TempDB データファイル: ファイルの数(<u>U</u>): 初期サイズ (MB)(<u>D</u>): 自動拡張 (MB)(<u>D</u>): データ ディレクトリ(<u>D</u>): TempDB ログファイル: 初期サイズ (MB)(<u>S</u>): 自動拡張 (MB)(<u>S</u>):	はクトリ「TempDB」FILESTREAM tempdb.mdf, tempdb_mssql.≢.ndf 2 ↓ 8 ↓ 初期サイズの合計 (MB): 16 64 ↓ 自動拡張の合計 (MB): 128 C+¥Program Files¥Microsoft SQL Server¥MSSQL13.MSSQLSER templog.ldf 8 ↓ C+¥Program Files¥Microsoft SQL Server¥MSSQL13.MSSQLSER C+¥Program Files¥Microsoft SQL Server¥MSSQL13.MSSQLSER	数 〕 〕 〕	加(<u>A</u>) 山除(<u>R</u>)	
		< 戻る(<u>B</u>) 次へ(<u>N</u>) > キャンセル	/	ヘルプ	

.mdf (データ ファイル) に関しては、ファイル数や初期サイズ、作成場所、.ldf (ログ ファイル) に関しては、初期サイズ、作成場所などを細かく設定できるようになりました。

なお、.mdf のファイル数は、既定では搭載されている CPU のスレッド数または 8 スレッド以上 の場合は 8 に設定されます (画面は 2 スレッドの場合の例)。また、既定の作成場所は、次のよう に SQL Server のインストール先の DATA フォルダー (以前のバージョンと同様、master など のシステム データベースが格納される場所) になります。

C: ¥Program Files ¥Microsoft SQL Server ¥MSSQL13.MSSQLSERVER ¥MSSQL ¥DATA

📙 🔄 🔜 🔻 DATA				-			
ファイル ホーム 共有 表示					~ 🕐		
 ・ へ 「」 C-¥Program Files¥Microsoft SQL Server¥MSSQL13.MSSQLSERVER¥MSSQL¥DATA ・ ひ 							
MSAS13.MSSQLSERVER	名前 ^	サイズ	更新日時	種類	^		
MSRS13.MSSQLSERVER	📴 DWDiagnostics.mdf	1,024,000 KB	2015/12/04 23:17	SQL Server Database Prima			
MSSQL13.MSSQLSERVER	DWDiagnostics_log.ldf	22,464 KB	2015/12/04 23:17	SQL Server Database Trans			
MSSQL	📴 DWQueue.mdf	4,352 KB	2015/12/04 23:17	SQL Server Database Prima			
Backup	📴 DWQueue_log.ldf	1,088 KB	2015/12/04 23:17	SQL Server Database Trans			
Pinn	📴 master.mdf	5,504 KB	2015/12/04 23:17	SQL Server Database Prima			
	📳 mastlog.ldf	2,048 KB	2015/12/04 23:17	SQL Server Database Trans			
DATA	📴 model.mdf	4,352 KB	2015/12/04 23:17	SQL Server Database Prima			
ExtensibilityData	📴 modellog.ldf	1,280 KB	2015/12/04 23:17	SQL Server Database Trans			
FTData	📴 MSDBData.mdf	16,576 KB	2015/12/04 23:17	SQL Server Database Prima			
Install	📴 MSDBLog.ldf	5, 184 KB	2015/12/04 23:17	SQL Server Database Trans			
JOBS	📴 ReportServer.mdf	4,352 KB	2015/12/04 23:17	SQL Server Database Prima			
Log	ReportServer_log.ldf	7,040 KB	2015/12/04 23:17	SQL Server Database Trans			
moldata	📴 ReportServerTempDB.mdf	4,352 KB	2015/12/04 23:17	SQL Server Database Prima			
	ReportServerTempDB_log.ldf	1,088 KB	2015/12/04 23:17	SQL Server Database Trans			
Microsoft Visual Studio 10.0	📴 tempdb.mdf	8, 192 KB	2015/12/05 12:46	SQL Server Database Prima			
Microsoft.NET	📴 tempdb_mssql_2.ndf	8, 192 KB	2015/12/05 12:46	SQL Server Database Seco			
MSBuild 🗸	🔋 templog.ldf	8, 192 KB	2015/12/05 12:46	SQL Server Database Trans	~		
19 個の項目 2 個の項目を選択 16.0 MB							

tempdb の性能に関連するトレースフラグ 1117 と 1118 が既定でオンに

SQL Server 2016 からは、tempdb データベースの性能に関連するトレース フラグである 1117 と 1118 をオンにしたときと同等の動作になるように、既定の動作が変更になります。こ の 2 つは、tempdb でラッチ待ちが発生している場合の動作を改善できるトレース フラグで、 SQL Server 2014 以前のバージョンを利用している場合のベスト プラクティスの 1 つでした。 SQL Server 2016 からは、既定でこの 2 つのトレースフラグが設定されている状態になるので、 これらを設定する必要がなくなりました。

なお、トレースフラグ **1117** と **1118** については、以下の KB 2154845(サポート技術情報) が参考になります。

Recommendations to reduce allocation contention in SQL Server tempdb database http://support.microsoft.com/en-us/kb/2154845

→ トレースフラグ 4199 がオンと同等の動作(クエリ オプティマイザーの動作)

SQL Server 2016 からは、データベースの互換性レベルで **130** を利用している場合に、**トレー ス フラグ 4199** をオンにしたときと同等の動作になるように、既定の動作が変更になります。こ れは、クエリ オプティマイザーがクエリを最適化する際の見積行数 (予測行数) などに関する変更 (hotfix) が含まれているものです。

トレースフラグ 4199 (やデータベースの互換性レベルによる動作の違いなど) については、以下の KB 974006 (サポート技術情報) が参考になります。

SQL Server query optimizer hotfix trace flag 4199 servicing model http://support.microsoft.com/en-us/kb/974006

✦ Active Directory パスワード認証/統合認証を利用可能に

SQL Server 2016 の Management Studio では、認証方法として「Active Directory パスワード認証」と「Active Directory 統合認証」が選択できるようになりました。これを利用すれば、 次のように Azure Active Directory (Azure AD)を利用して、Azure SQL Database に接続 できるようなります。

📲 データベース エンジンへの接	続		×			
	SQL Server		Azure SQL Databas Azure Active Direct	ie (C		
サーバーの種類(工):	データベース エンジン		のユーザーで接続	,		
サーバー名(<u>S</u>):	mihosrv1.database.window	snet	~			
認証(<u>A</u>):	Active Directory パスワート	*1211F	\sim			
ユーザー名(山):	miho01@hiro01sqlqualit	ty.onmicrosoft.com	~			
パスワード(<u>P</u>):	****					
	🗹 パスワードを保存する(<u>M</u>)				
	接続(C) キャンセル	ヘルプ オプ	/ション(0) >>			
Microsoft Azure 🗸	SQL データベース > AzureDB >	mihosrv1 〉 設定	〉 Active Directory 管理者	<i>117-7</i>	マの検索	○ 🗳 🖉 🕴
		* _ 🗆 ×	=0.04	_ 🗆 ×	• ·· · · · · · · · · · ·	
十 新規			說正 mihosrv1		Active Directory 音 (プバュー)	埋石
🗊 リソース グループ	→ + 移動 プールの追 ⁿⁿ				▲ 管理者の 管理者の 登定 副除 保存	★ 破棄
🗰 すべてのリソース		CH 18 🖉			Annua Antina Dina	
昼近使った項目	サーバーのバージョン V12		サポート ようブルシューティング		Azure Active Direct authentication (プ) を一元管理し、Azure	ctory レビュー版) では、ID e SQL Database
🔕 App Service	監宣 構成されていません		🗙 トラブルシューティング	>	V12 にアクセスできる。	ようになります。
🚊 仮想マシン (クラシック)	サーバー管理者 miho		🔮 新しいサポート要求	>	計構()再報 ピ	
🧕 仮想マシン	ファイアウォール ファイアウォール設定の表示		一般		プレビューの使用条件 承認されました	>
👼 SQL データベース	3d1…		11 プロパティ	>	Active Directory 管理者	0
🐵 クラウド サービス (クラシック)		9へ(の設定 →	ファイアウォール	>	< miho01@hiro01sqlqu	uality.onmicrosoft
💡 サブスクリプション	1	タイルを追加 🕀	▶ 監査	>		
参照 〉			🔀 最新の SQL Database U	Ipdate >		
Azure で Azure Ac	SQL Database tive Directory を設定		🔐 Active Directory 管理者	i (プレビュー) >	の管理者の設定	
	き 価格レベル		№ ロール	>		
	nline Basic		リソース管理			
			🎿 ユーザー	>		
			🛷 ঀঢ়	>		

なお、Azure SQL Database で Azure Active Directory を利用する方法については、Azure ド キュメントの以下のトピックが参考になります。

Azure Active Directory 認証を使用して SQL Database に接続する http://azure.microsoft.com/ja-jp/documentation/articles/sql-database-aad-authentication/

5.2 ライブ クエリ統計(Live Query Statistics)

ライブ クエリ統計は、現在実行中の SQL ステートメントの実行プランのうち、どの部分を実行しているのかを確認できる、大変便利な機能です。



ライブ クエリ統計では、完了した部分が実線で表示されて、実行中(未完了)の部分が点線で表示 されます。未完了のものは、現在の完了状況が % で表示されて、何秒(s)経過しているのかが表 示され、また、完了したものについては、100% と表示されて、どれぐらいの経過時間だったのか を確認することができます。これを見れば、どこがボトルネックでクエリが遅くなっているのかが 一目瞭然なので、パフォーマンス チューニングをする際に本当に役立ちます。

→ ライブ クエリ統計の利用方法(1クリックのみ)

ライブ クエリ統計は、次のようにツールバーの [**ライブ クエリ統計を含む**] をクリックするだけ で利用することができます。

💺 SQLQuery2.sql - WIN10.azTestDB (WIN10¥matun	noto (65))* - Microsoft SQL Server Management Studio (管理者)
ファイル(F) 編集(E) 表示(V) クエリ(Q) プロジェクト	(P) デパッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)
: 🛅 🕶 🔤 🗃 🛃 🍠 🔔 新しいクエリ(N) 🏻 🕞 י	🔁 🔁 🕹 👗 🛍 🗳 🔹 🖓 - (* - 🚰 🖳 🚧) 🕨 🚽
- 影 22 azTestDB - 🕴 実行	(X) 🕨 デバッグ(D) 💻 🗸 📅 🗐 🗐 🏹 📳 🏠 🎬 🎱 🗐 🙄 🗎
オブジェクト エクスプローラー 🔷 🕂 🗙	SQLQuery2.sql - W10¥matumoto (65))* × S ダイブクエリ統計を含む natumo
接続 📲 🛃 🔳 🍸 🏹 🏹	
□ 10 WIN10 (SQL Server 13.0.800 - WIN10¥matun □ □ データベース	SELECT coIC, COUNT(*)
🗉 🧰 システムデータベース	ON test1.colB = test2.colB
	GROUP BY colC
🖂 🧰 テーブル	

あとは、クエリを実行すれば、実行プランが表示されて、どこまで実行しているのかをグラフィカ ルに確認できるようになります。



実行中は、点線で表示されている部分が、完了後に実線に変わっていき、次のように表示されます。



実行が完了した後は、すべての実線に変わって、次のように表示されます。



全体の実行時間のうち、どの処理にどれだけの実行時間がかかったのかが簡単に分かるようになっているので、ボトルネックとなっている処理をすぐに見つけることができます。

5.3 クエリ ストアで性能監視、プラン固定

クエリ ストア(Query Store)は、「昨日までは問題なかったのに、何故か急に遅くなった」や「遅 くなった理由が分からない」といったクエリのトラブル(性能問題)を解決できる可能性がある、 大変便利な機能です。

クエリ ストアを利用すると、Store (蓄積) という名のとおり、**クエリの実行履歴** (と実行プラン) を保存することができるようになります。



これまでのバージョンでも query_stats という動的管理ビュー (DMV) を参照すれば、クエリの 実行履歴を確認することができましたが、この情報は、SQL Server を再起動したり、プロシージ ャ キャッシュのサイズが圧迫された場合には、参照できない (クリアされてしまう) という性質の ものでした。これに対して、クエリ ストアであれば、クエリの実行履歴を永続化することができる ので、SQL Server を再起動しても、過去に振り返って、クエリの実行履歴を確認できるようにな ります (遅くなった原因のクエリを過去に振り返って調べられるようになります)。

弊社のお客様の中には、SQL Server を定期的に再起動して運用しているという場合があるのです が、このように運用していると、これまでのバージョンの SQL Server では過去のクエリ実行履歴 を確認することができませんでした(∵query_stats に履歴が残っていないため)。これに対応す るには、アプリケーション側で対応したり(Log4net/Log4J などを利用して、アプリ側でログを 記録したり)、SQL Server のトレース(sp_trace_create)機能や拡張イベント(XEvents)、パフ ォーマンス データ コレクション機能を利用したりして、クエリの実行履歴を記録しておくという 方法しかとれませんでした(弊社の過去のパフォーマンス チューニングではトレースをよく利用 していました)。しかし、SQL Server 2016 からは、クエリ ストア機能が提供されるので、これ を利用すれば、簡単に(数クリックで)クエリの実行履歴を保存できるようになるので、大変便利 です。

→ クエリ ストアの最大のメリット ~実行プランの比較/プラン固定~

クエリ ストア機能では、**クエリの実行プラン**も同時に記録しておくことができるので、そのクエリ がどういった実行プランで実行されていたのかを、過去に振り返ることができます。これがクエリ ストア機能の最大のメリットです。また、同じクエリに対して、複数の実行プランがある場合には、 次のように簡単に見比べることもでき、かつどちらを利用させたいのかを変更(DB 管理者による プラン固定)することも簡単にできます。



このように、クエリストア機能を利用すれば、実行プランを簡単に確認/操作できるようになるので、次のような(多くの方々を悩ませてきた)性能問題を解決できる可能性が出てきます。

- 開発環境では問題がなかったのに、本番環境では性能が出ない
- 月イチの夜間バッチを実行した翌日にパフォーマンスが悪くなった
- 日によって性能が出るときと出ないときがある
- SQL Server を再起動した後に、性能が出なくなった

このような状況は、弊社のお客様でもよくあるのですが、これらの原因のほとんどは実行プランに よるものです。健全な性能が出ているときは、効率の良い実行プランが選択されていて、性能が出 ていないときには効率の悪い実行プランが選択されているといった状況です。

✤ Let's Try

それでは、クエリストアを試してみましょう。

1. まずは、クエリ ストアを試すためのデータベースを作成します。クエリ エディターで、次の ように CREATE DATABASE ステートメントを実行して、「QStestDB」という名前のデー タベースを作成します。

CREATE DATABASE QStestDB

 次に、クエリ ストアを有効化するために、データベースのプロパティを開きます。これを行う には、次のように「QStestDB」データベースを右クリックして、[プロパティ]をクリックし ます。

🍢 _クエリストア.sql - WIN10.master (WIN10¥matu_tak (5			_	
ファイル(F) 編集(E) 表示(V) プロジェクト(P) デバッグ(I	📙 テータベースのフロバティ - QSte	estDB —		×
: 🛅 🕶 📨 📂 🛃 🗿 🔔 新しいクエリ(N) 📑 📸	ページの選択	🔄 スクリプト 👻 📑 ヘルプ		
· · · · · · · · · · · · · · · · · · ·	☆ 主版 プライル			_
	・ 🚰 ファイル グループ			
	▲ オノション 「「」「「」「」「」」 「」」「」「」」 「」」「」」 「」」」 「 「」」 「」」 「」」 「」」 「」」 「」」 「」」 「」 「	◇ クエリストアの保有期間		
WIN10 (SOL Server 13.0.600 - WIN10¥matu tak)	▲ 権限 ▲ 本教長 - 4 - 4 - 4 - 4 - 4 - 4 - 4 - 4 - 4 -	クエリ ストア和の心かモート サイズ ベース クリーンアップ モード		
□ □ f-9<-X	「査】 孤5長フロパティ 「愛」ミラーリング	古いクエリのしきい値(日)		
🗉 🗇 システムデータベース 🛛 2	A トランザクション Q グの配布	■		
⊞ データベース スナップショット	🚰 ታェリ ストア	データのフラッシュ間隔(分)		
In Northwind		■ がおTUU4(実面)% ▼ 全般		
● ● デー 新しいデータベース(N)…	1	操作モード(実際) オフ		
田 □ テー: 新しいクエリ(Q)				
 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・				
ד בי לם: ד בי לוויע−(Ω)		カエリ フトマ形のパネルエード		_
		現在アクティブな取り込みモード。		
	-			
	-	理たのディスクの使用状況		
		PRITOD 4 V DOPUTIONE		
田 □ セキュリテイ 名前の変更(M) ロ □ サーバーオゴ オゴ	接続			
田 → ジ ハ ハン 前除(D) 田 → レプリケーショ 削除(D)	サーバー: WTN10			
AlwaysOn 最新の情報に更新(F)				
田 管理 プロパティ(R)	1致1元: WIN10¥matu_tak			
Integration Company Integrating Company Integrating Company Integrating Company				

[**データベースのプロパティ**] ダイアログでは、[**クエリ ストア**] ページを開きます。

3. [クエリ ストア] ページでは、[操作モード(要求時)] を [読み取り/書き込み] に変更して [OK] ボタンをクリックします。



<u>以上でクエリ ストアの設定が完了</u>です(ここで設定できるプロパティ項目については後述し ます)。これでクエリ ストアが有効になって、クエリの実行履歴が自動的に記録されていくよ うになります。

```
Note: SQL ステートメントでクエリ ストアを有効化する場合
SQL ステートメントを利用して、クエリ ストアを有効化したい場合には、次のように ALTER DATABASE ス
テートメントを実行します。
-- SQL でクエリ ストアを有効化する場合
ALTER DATABASE QStestDB
SET QUERY_STORE = ON
```

- SET 句で「QUERY_STORE = ON」と指定することで、クエリ ストアを有効化することができます。
- **4.** クエリ ストアを有効化すると、次のようにデータベース内に [**クエリ ストア**] フォルダーが 作成されるようになります。



5. 次に、データベース内にテーブルを作成します。

USE QStestDB CREATE TABLE QStest (coll int IDENTITY PRIMARY KEY , col2 int , col3 int)

テーブル名は「**QStest**」として、**col1** ~ **col3** の 3 つの列(いずれも **int** データ型)を持 つようにしています。

6. 次に、データを 10 万件追加します。この時点では、col3 列には「1」のデータのみを INSERT するようにします。

```
DECLARE @i int = 1
WHILE @i <= 100000
```

```
BEGIN
INSERT INTO QStest VALUES(@i, 1)
SET @i += 1
END
```

```
-- 10万件のデータを追加。col3 にはすべて「1」を INSERT

DECLARE @i int = 1

WHILE @i <= 100000

BEGIN

INSERT INTO QStest VALUES(@i, 1)

SET @i += 1

END

100 % - <

1 行処理されました)

(1 行処理されました)
```

次に、col3 列に「1」ではないデータを格納するために、UPDATE ステートメントを利用して、データを更新します。

UPDATE QStest SET col3 = 2 WHERE col1 = 99998 UPDATE QStest SET col3 = 3 WHERE col1 = 99999

8. 次に、SELECT ステートメントを実行して、データを確認しておきます。

```
-- データの個数の確認
SELECT col3, COUNT(*) FROM QStest
GROUP BY col3
100 % 👻 <
🔝 結果 🚹 メッセージ
        col1
                  col2
                            col3
99992
        99992
                  99992
                            1
99993
        99993
                  99993
                            1
        99994
                  99994
99994
                            1
        99995
                  99995
99995
                            1
        99996
                  99996
99996
                            1
        99997
                  99997
99997
                            1
                          2
        99998
                  99998
99998
99999 99999
                  99999
                            3
100000 100000
                 100000
                            1
```

SELECT * FROM QStest

99998 のデータの col3 列が「2」、99999 が「3」、その他は「1」であることを確認できます。

9. 次に、GROUP BY 演算を利用して、col3 列のデータの件数を確認しておきます。

```
SELECT col3, COUNT(*) FROM QStest
GROUP BY col3
```

データの個数の確認 SELECT col3, COUNT(*) FROM QStest GROUP BY col3					
100 %	• <				
🔝 結:	話果 11 メッセージ				
	col3	(列名なし)			
1	3	1			
2	1	99998			
3	2	1			

「3」と「2」はデータ件数が1件、「1」は99,998件であることを確認できます。

10. 次に、col3 列に非クラスター化インデックスを作成しておきます。

CREATE INDEX idx_col3 ON QStest(col3)

→ クエリの実行

次に、クエリ ストアにデータを蓄積するために、クエリをいくつか実行しておきましょう。

1. まずは、「WHERE col3 = @p1」という形のパラメーター クエリを利用して、「@p1」に「2」 を与えて、col3 列が 2 のデータのみを取得してみます。

EXEC sp_executesql N' SELECT * FROM QStest WHERE col3 = @p1', N' @p1 int', @p1 = 2



このクエリは、先ほど作成したインデックスを利用した Index Seek で実行されるので、これを確認しておきます。ツールバーの [実際の実行プランを含める] をクリックして、再度クエリを実行します。

* - Microsoft SQL Server Management Studio (管理者)
(ッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)
👸 🔏 🛍 🖄 🔊 - 🗠 📲 - 🖳 🖓 🕨 🔹 🚽 🖓 ValidTo
デバッグ(D) = 🗸 🎲 🖻 🖥 🎇 🥦 🝓 🏙 🎧 🖆 🖄 📮
× _クエリストア.sql - WIN10¥matu_tak (53))* ×
データの検索。実行ブランで Index Seek であることを確認 EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 int', @p1 = 2
100 % - <
結果 メッセージ ** 実行ブラン
フリリ 1: クリリ コスト (バッナ相対): 100% SELECT * FROM QStest WHERE col3 = 0p1 であることを確認
SELECT Nested Loops Index Seek (NonClustered)
۲۰۰۵ (۱۸۲: ۵۵ الم) ۱۸۲: ۵۵ الم)
ギー参照(Clustered) [QStest].[PF_QStest_357D0D3E701BF コスト: 50 %

クエリの実行後、[実行プラン] タブで、Index Seek になっていることを確認します。

3. 次に、同じクエリで、「@p1」に与える値を「3」に変更して実行してみます。

EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 int', @p1 = 3

EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 int', @p1	= 3						
100 % - <							
□ 結果 □ メッセージ 🚏 実行プラン	/ 3 に変更						
1 99999 99999 3							
🖽 結果 🛅 メッセージ 🎦 実行プラン							
クエリ 1: クエリ コスト (バッナ相対): 100% SELECT * FROM QStest WHERE col3 = @p1 「あることを確認」							
SELECT Nested Loops Index Seek (NonClustered)							
JAb: 0 % [QStest].[idx_col3] JAb: 0 % JAb: 50 %							
キー参照 (Clustered)							
LUSTESTJ.[PK_QStest_357DDD3E701BF コスト: 50 %							

これも Index Seek で実行されていることを確認できます。

4. 次に、同じクエリで、「@p1」に与える値を「1」に変更して実行してみます。

EXEC sp_executesql N' SELECT * FROM QStest WHERE col3 = @p1', N' @p1 int', @p1 = 1

	EXEC s	p_exec	utesql	I N'SELECT * FROM G)Stest WHERE col3 = @p1'	, N'@p1 int', @p1 = 1
100 %	• • <					
	結果 🚮	メッセージ	₽" 実	ミイテプラン		
	col1	col2	col3			ILZE
1	1	1	1			
2	2	2	1			
3	3	3	1			
4	4	4	1			
5	5	5	1			
6	6	6	1			
7	7	7	1			
フエ SEI	結果 🛅 リ 1: ク LECT *	メッセー: エリ コス FROM (기 👬 지ト (/ OStest	実行プラン ベッナ相対): 100% t WHERE col3 = @p	1	Index Seek であることを確認
sı J	ELECT		・ Nested (Inner コスト	d Loops	Index Seek (NonClustered) [QStest].[idx_col3] 」人ト: 50 年 ぞ一参照 (Clustered) test].[PK_QStest_357DDD3E コスト: 50 年	701BF

「1」に該当するデータ件数は多い(99,998 件)ので、実行には少し時間がかかります。これも Index Seek で実行されていることを確認できます。

→ SQL Server 再起動後の同じクエリの実行プランの確認

次に、SQL Server サービスを再起動して、再起動後に同じクエリを実行して、実行プランがどのように変化するのかを確認してみましょう。

1. SQL Server サービスを再起動するには、次のように操作します。

🍢 _クエリストア.sql - WIN10.QStestDB (WIN1	0¥matu_tak (53))* -	
ファイル(F) 編集(E) 表示(V) クエリ(Q) ブ	ロジェクト(P) デバック	
🗄 🚼 🕶 🗁 💕 🛃 🥥 🔔 新しいクエリ(1	v) 🛅 📸 📸 🌆	
QStestDB -	? 実行(X) ♪ デ)	
	- 1 - 1	
	· + ^	
1000 · 24 24 = 1 2 20		
□ □ 〒	#matu_tak)	
■ ■ ジステム 接続(C) 田 ■ ジステム +女体も250(C)		
 QStest 新しいクエリ(Q) 		
田 □ 7- 利用状況モニタ- 利用状況モニタ-	-(M)	t SQL Server Management Studio
		WIN10の MSSQLSERVER サービスを再起動しますか?
■ ● 外 億止の		
■ □ シ/ 一時停止(1)		2
	12	(はい(Y) (いいえ(N)
ポリシー(O)	•	
ノイゴ ファセット(A)		
E Ser PowerShell の D	L雪力(H)	
Report	-	
■ ■ Report 最新の情報に更	新(F)	
■ 🛄 レノリケーション		

2. 再起動が完了したら、再度、先ほどと同じクエリを実行します。このとき、「@p1」に与える 値には「1」を指定します。

```
EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 int', @p1 = 1
```



今度は、Clustered Index Scan (全スキャン)で実行されるように変わっていることを確認 できます。「1」に該当するデータ件数は多いので、SQL Server のクエリ オプティマイザー (クエリの最適化を行う内部エンジン)が全スキャンをしたほうが速いと判断して、このよう になっています。

3. 次に、「@p1」に与える値を「2」に変更して実行します。

EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 int', @p1 = 2

EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 in	t', @p1 = 2
100 % - <	
語 キャセージ またブラン	2 に変更
クエリ 1: クエリ コスト (バッナ相対): 100%	
SELECT * FROM QStest WHERE col3 = @p1 不足しているインデックス (影響 79.6072): CREATE N で実行されている	issing Index, sysname,
SELECT Clustered Index Scan (Clustered) [QStest].[PK_QStest_357DDD3E701BF]\lambda h: 100 %	

これも、Clustered Index Scan (全スキャン) で実行されていることを確認できます。「2」 に該当するデータ件数は 1 件のみなので、<u>Index Seek を利用した方が速く実行できる</u>にも 関わらず、全スキャンが選択されてしまっています。

このように、パラメーター クエリの場合は、一度実行プランが選択されると、パラメーターに 代入される値に関係なく、同じ実行プランが選択されてしまうという動作になります(そもそ も、パラメーター クエリは、コンパイルの負荷を軽減する=パラメーター違いのクエリでも 同じ実行プランを再利用するための機能なので、これはごくごく普通の動作になります)。

→ プロシージャ キャッシュのクリア後の実行プランの確認

次に、プロシージャ キャッシュをクリアした場合に、同じクエリを実行して、実行プランがどのように変化するのかを確認してみましょう。

1. プロシージャ キャッシュをクリアするには、次のように DBCC FREEPROCCACHE コマン ドを実行します。

DBCC FREEPROCCACHE

 クリアが完了したら、再度、先ほどと同じクエリを実行します。このとき、「@p1」に与える 値には「2」を指定します。

EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 int', @p1 = 2

データの検索。実行ブランで Index Seek に変わることを確認 EXEC sp_executesgl N'SELECT * FROM QStest WHERE col3 = @p1', N'@	p1 int', @p1 = 2
100 % - <	
田 結果 は メッセージ ま 、 ま 「 ゴ キ 「 マ 」 、 ジッセージ ま ・ マ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・	
クエリ 1: クエリ コスト (バッチ相対): 100%	2 を与える
SELECT * FROM QStest WHERE col3 = @p1	
T T M	
SELECT Nested Loops Index Seek (NonClustered)	
コスト:0 も (Inner Join) [QStest].[idx_col3] コスト:0 も コスト:50 も	Index Seek わっていることを確認
キー参照 (Clustered)	
[QStest].[PK_QStest_357DDD3E701BF コスト: 50 %	

今度は、**Index Seek**(インデックスのピンポイント検索)で実行されるように変わっている ことを確認できます。このように、プロシージャ キャッシュをクリアした場合にも、SQL Server を再起動したときと同じ動作(クエリ オプティマイザーによる実行プランの選択し直 し)になります。

3. 次に、「@p1」に与える値を「1」に変更して実行します。

EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 int', @p1 = 1



これも、Index Seek で実行されていることを確認できます。

	最初に col3 に 与えた値	実行プラン	次に col3 に 与えた値	次に col3 に 与えた値
最初	2	Index Seek	3	1
SQL Server 再起動後	1	Clustered Index Scan	2	-
DBCC FREEPROCCACHE 後	2	Index Seek	1	-

ここまでの結果をまとめると、次のようになります。

col3 内の「2」のデータ件数は「1件」、「1」のデータ件数は「99,998件」なので、パラメ ーターに「2」を与えた場合は Index Seek(ピンポイント検索)のほうが効率が良いと判断 され、「1」を与えた場合は Clustered Index Scan(全スキャン)のほうが効率が良いと判 断されています。このように、データに"偏り"がある場合には、SQL Server の再起動やプ ロシージャ キャッシュのクリアによって、選択される実行プランが定まらない(ころころ変 わってしまう、最初に与えた値を最適化するように実行プランが選択される)という状況にな ります(このような状況は、弊社のお客様でもたくさん遭遇してきました)。

クエリ ストア機能を利用すれば、このような状況が発生したかどうかを簡単に確認できるようになり、かつデータベース管理者が意図した実行プラン(この実行プランを利用して欲しいというもの)に強制(プラン固定)していくということも簡単に行うことができます。以降では、これも試してみましょう。

→ クエリ ストアを GUI で確認

まずは、クエリ ストアに蓄積されたデータを確認してみましょう。

1. クエリ ストアに蓄積されたデータを確認するには、次のようにオブジェクト エクスプローラ

ーで、[**クエリ ストア**] フォルダーを展開して、[**リソースを消費するクエリの上位**] をクリッ クします。



[合計 実行時間]の長い順にクエリが表示されていることを確認できます。

ー番実行時間の長いクエリ(グラフの一番左に表示されるクエリ)は、前の手順で実行した 「WHERE col3 = @p1」のクエリです。

2番目に長いクエリは、「SELECT * FROM CStest」(一番最初にデータを確認するために実行したもの)であることを確認できます。



3 番目に長いクエリは、「INSERT INTO CQtest VALUES(@i, 1)」(10 万件のデータを



INSERT したときのもの) であることを確認できます。

このように、クエリストア機能を有効化すると、実行時間の長いクエリを簡単に識別できるようになるので、大変便利です。なお、このグラフにリストされるのは、既定では「過去1時間」に実行されたものになっています(表示される期間を変更する方法については後述します)。

2. 次に、[総計] で [平均] を選択します。



これで、合計の実行時間ではなく、平均の実行時間(例えばクエリが 3回実行された場合は、 その 3回の実行時間を合計した値になるのではなく、3回の平均の実行時間)でクエリを並べ 替えることができるようになります。この [総計]では、[合計] と [平均] のほかに、[最大] や [最低]、[標準偏差] を選択することもできます。

3. 次に、[メトリック] で [実行回数] を選択します。



これで、実行回数の多い順にクエリを並べ替えることができます。実行回数が多いのは、(10 万回実行したので) INSERT ステートメントになることが分かると思います。このように、実 行時間ではなく、実行回数で並べ替えることができるのも、クエリ ストアのメリットです。こ こでは、[実行時間] と [実行回数] のほかに、[CPU 時間 (µs)] [論理読み取り]、[論理書 き込み]、[メモリ消費量 (KB)]、[物理読み取り] を選択することもできるので、CPU の利用 時間の長いクエリや、I/O (読み取り/書き込み) 数の多いクエリ、メモリ消費量の多いクエ リで並べ替えることができるようになっています。

 確認後、[メトリック] で [実行時間 (μs)] を選択して、実行時間で並べ替えるように戻して おきます。

なお、CTP 3.2 では、[メトリック] で [実行時間 (μs)] を選択しても、実行時間で並び変わ らないことがあり、その場合は、いったんウィンドウを閉じて、再度 [クエリ ストア] フォル ダーの [リソースを消費するクエリの上位] をクリックしてみてください。

→ 表示される期間の変更

前述したように、グラフに表示されるクエリは、既定では「過去1時間」に実行されたものです。これを変更するには、次のようにツールバーの [構成](一番右側にあるアイコン)をクリックします。

トップリソース コンシューマー [QStestDB] × _クエリストア.sql - WIN10¥matu_tak (53))*									
Ŧ	データベース QStestDB の 過去 1 時 🍢 トップリソース コンシューマー [QStestDB] ー ロ 🗙 🗖 積長口- 🔽 構成								
×	トリック 実行時間 (μs) * #	トップ リソース コンシューマ	ーの確認先	基準			3 10 1		<u>-</u>
>		○ CPU 時間 (µs)		0 4	均				
	2000000-	● 実行時間 (µs)			大		9	Í	
	1500000-	○ 実行回数		○ ह	低	-	0		
記		○ 論理読み取り		〇樗	準偏差			プランI	D
東	1000000-	○ 論理書き込み		• 合	iit			9	
ata (40	50000-	〇 メモリ消費量 (KB)	過去 5 分間	-				145	5
		○ 物理読み取り	道去 15 分f 過去 30 分f	51 19					
	⁰┙┙┙╝╝╝╝╝╝╸╸╸╸┙	期間	過去1時間				_	l l	
			過去 12 時間	9	表示す を選択	る期間 できる	0 1:10 1:2	0	
	919 id	過去 1 時間	過去1日間	5	/				
ブ	ラン 145 [強制されない]	返す	道去 2 日間 過去 1 週間	L			強制 🌆 🕄	プランを強制し	ない 🗧
クコ	[リ 1: クエリ コスト (バッチ相対)		過去 2 週間						
不	足しているインデックス (影響 79.		過去 1 か月				issing	Index, s	ysn
	-	© 112 25	過去 3 か月	8					
	Clustered Ind	時刻表示形式	過去 6 か月 昨年						
לב	[QStest].[PK_Q	 ・カル 〇 UT カスタム 							
		ОК	キャン	セル	這	i用			

ここで [期間] を [過去1週間] や [過去1ヶ月間] などに変更することで、過去に戻ること ができるようになります (どれだけ過去に戻れるかは、クエリ ストアの最大サイズによって 変わってくるので、この設定方法については後述します)。また、この [構成] ダイアログでは、 前述の [メトリック] や [総計] (基準) を変更したり、上位何件 (既定では 25 件) を表示す るのかを設定できたりします。

次に、一番実行時間の長いクエリである、前の手順で実行した「WHERE col3 = @p1」を選択します。



右側のグラフには、実行プランごとの平均実行時間が表示されますが、ここに 2 つの実行プ <u>ラン</u>が表示されていることを確認できます。この右側のグラフは、平均実行時間が Y 軸にな っていて、平均実行時間が短いほうが Index Seek の実行プラン、長いほうが Clustered Index Scan (全スキャン)の実行プランであることが分かります。



2. 次に、ツールバーの [選択したクエリのプランを、別々のウィンドウで比較します] をクリックします。

トップ リソース コ	プリソース コンシューマー [QStestDB] 🗙 _クエリストア.sql - WIN10¥matu_tak (53))* 🔹					
データベース	、QStestDB の 過去 1 時間 の間の上位 25 のトップ リソース コンシューマー	📄 縦長ビュー 🛄 横長ビュー 🏹 構成 🖕				
メトリック 実	行時間 (μs) 🔹 統計 合計 🔹 🥊 🥊 クエリ 9 のプランの概要	🗉 🐹 🌃 📰 🖂 📮				
>	>					
2000000-	・ <mark>クエリ6のプラン [QStestDB] を比較します ×</mark> トップ リソース コンシューマー [QStestDB] _	クエリストア.sql - Wl0¥matumoto (144)) 🛛 👻				
	プラン 6 [強制されない]	🛛 📅 プランの強制 🔚 プランを強制しない 🚽				
	クエリ 1: クエリ コスト (パッチ相対): 100% SELECT * FROM QStest WHERE col3 = @p1	V				
	🗖 🖻 🖏					
	SELECT Nested Loops Index Seek (NonClustered □Xト: 0 % [QStest].[idx_col3] □Xト: 0 % □Xト: 50 %	1)				
	キー参紙 (Clustered) [QStest].[PK_QStest_357DDD33 コスト: 50 %	5874DC				
	ゴニック(2米半川さわたい)	翌 ゴニンの法制 翌 ブニンを洗剤 たい				
	カナリ 1・カナリ コフト (パッチ相対)・ 100%					
	SELECT * FROM QStest WHERE col3 = @p1 不足しているインデックス (影響 79.6072): CREATE NONCLUSTERED :	INDEX [<name index,<="" missing="" of="" td=""></name>				
	SELECT Clustered Index Scan (Clustered) [QStest].[PK_QStest_357D0D3E874DC]Ah: 100 %					

これで、2つの実行プランを上下で確認できるようになります(比較しやすくなります)。

→ 実行プランの強制

クエリに対して、複数の実行プランが表示される場合は、<u>どちらかの実行プランで必ず実行される</u> ように、実行プランを強制することができます。以前のバージョンの SQL Server では、実行プラ ンの強制には、**プランガイド**という機能を利用していましたが、利用するにはいろいろと面倒なと ころがありました。しかし、クエリ ストアであれば、わずか数クリックするだけでこれを実現する ことができます。これも試してみましょう。

 まずは、col3=@p1 のクエリが Clustered Index Scan (全スキャン) で実行されるよう に、プランを強制してみましょう。次のように Clustered Index Scan の実行プランに対し て、[プランの強制] をクリックします。

クエリ 6 のプラン [QStestDB] を比較します × トップ リソース コンシューマー [QStest	DB]クエリストア.sql - WI0¥matumoto (144)) 🛛 👻
プラン 6 [強制されない]	🚼 プランの強制 🔡 プランを強制しない 🚽
フリ 1: フリ コスト (パッチ相対): 100% SELECT * FROM QStest WHERE col3 = @pl	
SELECT Nested Loops Index Seek (NonC (Inner Join) コスト: 0 * 「スト: 0 * 「(Stest].[idx コスト: 50 本一参照 (Clus [QStest].[PK_OStest_ コスト: 50	lustered) col3] % tered) 357D0D3E874DC %
プラン 8 [強制されない]	1 📅 プランの強制 📅 プランを強制しない 🝦
クエリ 1: クエリ コスト (バッチ相対): 100%	
SELECT * FROM QStest WHERE col3 = @p1 不足しているインデックス (影響 79.6072): CREATE NONCLUS	確認 × Index,
SELECT [QStest].[PK_QStest_357DDD3E874DC	クエリ 6 に対してプラン 8 を強制しますか?
	2 (まいか) いいえ(N)

[確認] ダイアログが表示されたら、[はい] をクリックします。これで、「WHERE col3 = @p1」のクエリが、必ず Clustered Index Scan で実行されるようになります。

2. これを確認するために、クエリを実行してみます。このとき、「@p1」に与える値には「2」を 指定します。

EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 int', @p1 = 2

	EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 int', @p1 = 2
100) % · • <
] 結果 🛅 メッセージ 🚏 実行プラン
ク	エリ 1: クエリ コスト (バッチ相対): 100%
SE	ELECT * FROM QStest WHERE col3 = @p1
不	「足しているインデックス (影響 79.6072): CREATE NONCLUSTERED INDEX [<name index,<="" missing="" of="" td=""></name>
<u>د</u>	SELECT Clustered Index Scan (Clustered) [QStest].[PK_QStest_357DDD3E701BF]/: 0 %

Clustered Index Scan(全スキャン)で実行されるように変わっていることを確認できます。

3. 次に、「@p1」に与える値を「1」に変更して実行します。

EXEC sp_executesql N'SELECT * FROM QStest WHERE col3 = @p1', N'@p1 int', @p1 = 1



これも、Clustered Index Scan で実行されていることを確認できます。このように、クエ リ ストアのプラン強制機能を利用すると、指定した実行プランで必ず実行するように強制で きるようになります。

次に、先ほどの実行プランを2つ表示しているウィンドウに戻って、今度は Index Seek のの実行プランに対して、[プランの強制]をクリックします。



[確認] ダイアログが表示されたら、[はい] をクリックします。

これで、「WHERE col3 = @p1」のクエリが、必ず Index Seek で実行されるようになり ます。これも試しておきましょう。



このように、クエリ ストア機能を利用すれば、クエリに対して実行プランを指定することが 簡単にできるようになるので大変便利です。

- → 全体のリソース消費量
 - 1. 次に、[クエリ ストア] フォルダーの [全体のリソース消費量] をクリックしてみます。



これで、「実行時間」や「CPU 時間」、「実行回数」、「論理読み取り」に関して、時間ごとに確認できるようになります(どの日のどの時間帯に、実行時間が長いクエリが実行されたのか、実行回数が跳ね上がったのはいつなのか、といった情報を確認できるようになります)。

グラフに何も表示されない場合は、ツールバーの [構成] ボタンをクリックして、[期間] を [過去1時間] に変更するなどしてみてください (∵既定値は、過去1ヶ月間に設定されて いるので、何も表示されていないように見えるため)。

😓 低下したクエリ [QStestDB] の構成	- 🗆 X
低下の確認先	基準
○ CPU 時間 (µs)	○ 平均
④ 実行時間 (μs)	○ 最大
○ 論理読み取り	○ 最低
○ 論理書き込み	○ 標準偏差
 メモリ消費量 (KB) 	● 合計
○ 物理読み取り	
期間	
最新 過去 1 時間	履歴 過去1週間 ~
返す	
) ತಗನ	
● 上位 25	

→ クエリ ストアの設定

クエリ ストアは、既定では、100MB のサイズになるまで蓄積できるようになっています。これは、 [データベースのプロパティ] の [クエリ ストア] ページで確認/変更することができます。



[[]最大サイズ] を超えた場合には、クエリ ストアは、自動的に「読み取り専用モード」に変更され

て、これ以上クエリの実行履歴を記録しないようになります(サイズ拡大の要因となったものを追跡できるように、読み取り専用モードとして、過去のデータに振り返られるようになっています)。 これ以上クエリを記録したい場合には、[最大サイズ]を変更するようにします。あるいは、クエリの分析が完了(性能低下の原因となっているクエリを特定できたなど)した場合には、[クエリデータの削除]をクリックして、古いデータ(クエリの実行履歴)を削除することもできます。

その他、クエリ ストアについては、オンライン ブックの以下のトピックがお勧めです。

Database Properties (Query Store Page)

http://msdn.microsoft.com/en-us/library/dn817825.aspx



not added. If the maximum allocated space of the query store has been exhausted, the

query store will change its operation mode to READ_ONLY.

5.4 Transact-SQL(T-SQL)の強化

SQL Server 2016 では、Transact-SQL も強化されています。

- **R 統合**(4章で説明)
- **JSON 対応**(4章で説明)
- **DROP .. IF EXISTS**(オブジェクトが存在している場合に DROP を実行)
- DBCC CHECKDB/CHECKTABLE/CHECKFILEGROUP で MAXDOP オプションを利用可能に

やはり、R 統合(R スクリプトを実行できるようになった点)と JSON 対応は、大きな強化ポイントです。

もう 1 つ大変便利になったのが「**DROP .. IF EXISTS**」です。これまでの SQL Server では、オ ブジェクトを削除する場合には、次のようにオブジェクトの存在チェックを行う必要がありました。

-- 該当オブジェクトが存在するなら削除 (SQL Server 2014 以前の場合) IF EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'obj1')) DROP TABLE obj1

SQL Server 2016 からは、「**DROP .. IF EXISTS**」が提供されたことで、上の記述を、次のよう に記述できるようになります。

-- 該当オブジェクトが存在するなら削除 (SQL Server 2016 の場合) DROP TABLE IF EXISTS obj1

```
-- 該当オブジェクトが存在するなら削除(SQL Server 2016 の場合)
DROP TABLE IF EXISTS obj1
100 % ▼ <

動 メッセージ
コマンドは正常に完了しました。
```

テーブルに対して実行したい場合は、「DROP TABLE IF EXISTS テーブル名」と記述することで、 該当テーブルが存在する場合には削除、そうでない場合には何もしない(エラーも発生しない)と いう使い方ができます。

そのほか、ストアド プロシージャの場合は「**DROP PROCEDURE IF EXISTS ストアド プロシ ージャ名**」、ビューの場合は「**DROP VIEW IF EXISTS ビュー名**」という形で利用することがで きます。また、CTP 3.2 では、**DROP .. IF** でサポートされるのは、AGGREGATE,、ASSEMBLY、 COLUMN、CONSTRAINT、DATABASE、DEFAULT、FUNCTION、INDEX、ROLE、RULE、SCHEMA、 SECURITY POLICY、SEQUENCE、SYNONYM、TRIGGER、TYPE、USER です。
5.5 AlwaysOn 可用性グループの拡張

SQL Server 2016 では、AlwaysOn 可用性グループ(Availability Group)も大幅に強化されました。AlwaysOn 可用性グループでは、主に次の新機能が提供されています。

- 自動フェールオーバーを構成できる台数が 2 から 3 に増加
- ログ転送性能の向上(マルチ スレッドで処理)
- ラウンドロビン レプリカ
- TDE(透過的なデータ暗号化)のサポート
- DTC (分散トランザクション)のサポート
- ワークグループ環境で利用可能(Windows Server 2016 を利用する場合)
- Standard エディションでも利用可能

◆ 自動フェールオーバーを構成できる台数が 2から 3 に増加

SQL Server 2016 からは、可用性グループで自動フェールオーバーを構成できる台数が 2から 3 に増えました。

浳 新しい可用性グループ					_	\Box ×
「こうしょう レプリカの指定						
説明						◎ ヘルプ
名前の指定	セカンダリ レプリカをホストす	る SQL Server	のインスタンスを指定	します。		
データベースの選択	レプリカ エンドポイント ノ	(ックアップの設	定 リスナー			
レプリカの指定	可用性レプリカ(⊻):					
データ同期の選択	サーバー インスタンス	初期ロ ール	自動フェールオーバ - (上限 2)	同期コミット (上限 3)	読み取	り可能なセカ
検証	SERVER1	プライマリ			はい	
概要	SERVER2	セカンダリ			はい	
結果	SERVER3	セカンダリ			はい	
	<					>
	レプリカの追加(<u>A</u>) Azure レプリカの追加(<u>Z</u>) レプリカの削除(<u>R</u>)					
	SERVER3 によってホストされているレブリカの要要 レブリカ モード:同期コミットと自動フェールオーバー レブリカは同期コミットの可用性モードを使用し、自動フェールオーバーと手動フェールオーバーの両方を サポートします。 きみ取り可能たをわっぴしたい、 >					
			<	戻3(P) 次八(I	N) >	キャンセル

➡ ログ転送転送性能の向上(マルチ スレッドで処理)

SQL Server 2016 からは、可用性グループでの**ログ転送の性能が向上**しています。ログ リーダー (ログ転送時のログを読み取るスレッド)や、REDO スレッド(レプリカ側でログを反映するスレ ッド)がマルチ スレッド対応になったことでこれを実現しています。

実際に、弊社環境で、ログ転送の性能を検証してみたところ、次のような結果を確認することができました。



日本マイクロソフト株式会社の監修のもと、数値を掲載しております。 また、検証は CTP 3.1 で行ったものであるため、RTM 版ではさらに性能が向上する可能性があります。

SQL Server 2016 では、ログ転送スピード (Bytes send to Transport/sec) が **450MB/sec** 近 く出ているのに対して、SQL Server 2014 では **300MB/sec 弱**を推移しているという状態でし た (CPU 利用率に関しては、SQL Server 2014 が 50%前後を推移して、CPU 性能を活かし切れ ていない状態であるのに対して、SQL Server 2016 では 70%近くを推移して、CPU を活用して いる状態)。このグラフは、1,000 万件のデータベースのインデックスを再構築 (REBUILD) して いるときの状況 (パフォーマンス カウンター) になりますが、実行時間は次のようになりました (3 回実行したときの実行時間を計測しました)。

バージョン	平均一	1st 💌	2nd 💌	3rd 🔽
SQL Server 2014	16.7	17	16	17
SQL Server 2016	11.0	11	11	11
差	5.7			
差%	34.0%			

SQL Server 2016 では 平均 **11 秒**で完了したのに対して、SQL Server 2014 では **16.7 秒**もか かっていて、**34%**の性能向上を確認することができました。

◆ ワークグループ環境で利用可能(Windows Server 2016 を利用する場合)

SQL Server 2016 からは、OS に Windows Server 2016 を利用している場合には、ワークグ ループ環境でも可用性グループを利用できるようになりました (Active Directory が必須ではなく なりました)。Windows Server 2016 では、ワークグループ環境でも WSFC(フェールオーバ - クラスタリング)を構成できるようになったので、この機能を利用して、可用性グループを構成 できるようになりました。

👱 ୬ステム							ı ×
🛧 💆 > วาหนาน แล้งเป็น	/ > システムとセキュリティ > シス	7L		~	- D- UV-	-ル パネルの検索	م
]ンピューターの基本的な情報	報の表示	Window	vs Server 20)16 TP4		•
	Vindows のエディション						
♥ リモートの設定	Windows Server 2016 Technie	cal Preview 4					
🗣 システムの詳細設定	© 2015 Microsoft Corporatio	on. All rights re	served.	_ W	/indows Server	2016 Technical Pi	review 4
3	1276						
	プロセッサ:	Intel(R) Core(7	M) i7-3770K CP	U @ 3.50GHz 3.50) GHz		
	実装メモリ (RAM):	32.0 GB (31.7 (GB 使用可能)				
	システムの種類:	64 ビット オペレ・	ーティング システム、	×64 ベース プロセッサ	<i>†</i>		
	ペンとタッチ:	このディスプレイア	では、ペン入力とタ	アッチ入力は利用でき	ません		
-	コンピューター名、ドメインおよびワーク	グループの設定 -					
	コンピューター名:	SARAH				♥設定の変	更
	フル コンピューター名:	SARAH.test.lo		クグループ環	境	-	
	コンピューターの説明:		7/				
Г Г	ワークグループ:	WORKGROUP					
•							
v	Vindows ライセンス認証						
	Windows はライセンス認証され	ています。 マイ	クロソフト ソフトウ:	エア ライセンス条項を	読む 		
-							
AG: SARAH - Microsoft SQL Server Manag	jement Studio (管理者)						- 🗆 ×
ファイル(F) 編集(E) 表示(V) ツール(T) ウイン	ノドウ(W) ヘルプ(H)						
🛅 • 🔝 • 🚰 🔙 🗿 🤔 新しいクエリ(N)		1) + (11 + 4	₽ • ⊑, 24 ,	•	-	- 2	Ŧ
オブジェクト エクスプローラー 🔹 🔻	무 🗙 🗚 AG: SARAH 🗙				ワークグル	レープ環境で	-
接続 - 🚏 🛃 🔳 🍸 🏹 😹					可用性クル	レープを構成	冬更新: 2015/11/23 1:26:20
😑 🐻 SARAH (SQL Server 13.0.700 - SARAH	¥Admi 🚺 AG: SARAH	によってホス	ト (レプリカ ロー,	ル: プライマリ)	7	白了	計画新 オン 川
□ □ データベース □ □ □ ジィフテル データベーフ	コ田桂ガループの世常にい	- π θ				71-	-ルオーバー ウィザードの開始
■ □ ジステム テークバース ■ □ データバース スナップショット	引用モンルーノのいた(G): プライマリインスタンス(P):	SARAH			V	Alw	aysOn ヘルス イベントの表示
🖃 🧻 AGTestDB (同期済み)	フェールオーバー モード	手動				252	スタークォーラム情報の表示
⊛ 🚞 セキュリティ	クラスターの状態:	cluster1 ((標準のクォーラム)				
■ 🗀 サーバー オフジェクト □ 📥 レゴルケーション	可用性レプリカ(R):						列の追加と削除
III → レンリン ション □ → AlwaysOn 高可用性	名前	0-ll	フェールオーバー	同期状態	間顕点		
🖂 🧰 可用性グループ	GREEN	セカンダリ	<u>王</u> 新	同期済み	1772/56471-4		
□ (1) AG (プライマリ)	SARAH	プライマリ	子助 王動	同期済み			
 回用性レノシカ	<u> </u>	22132	7 80	In 1997 A Street of the street			
🎽 SARAH (プライマリ)	グループ化(G) マ						<u>列の追加と削除</u> 🍥
□ □ 可用性データベース	名前	レプリカ		同期状態	歳	フェールオーバーの	間題点
GlestUB 国用性グループリスナー	GREEN						
	-						
🔹 🖬 🛄 見狂	🕑 AGTestDB	GREEN		同期済み	ŀ	データ損失なし	
● ■ 管理 ● ■ Integration Services カタログ	Ø AGTestDB SARAH	GREEN		同期済み	Ъ	データ損失なし	

5.6 BI 関連の強化

SQL Server 2016 では、BI 関連の機能も強化されています。

- Reporting Services (SSRS)の強化 (Datazen など)
- Integration Services (SSIS)の強化
- Analysis Services (SSAS)の強化
- MDS (マスター データ サービス) の強化

◆ Reporting Services の強化(Datazen 統合など)

SQL Server 2016 の Reporting Services は、久しぶりの大きな変更が加わっていて、主に次の 新機能が提供されています。

- パラメーター ボックスのカスタマイズ
- 新しいグラフのサポート(サンバースト、ツリーマップ)
- 新しいレポート ビルダー
- PowerPoint レンダリング (PowerPoint ファイルへのエクスポート)
- 印刷コントロールの変更(ActiveX コントロールが不要に)
- Datazen 統合(モバイル レポート対応、新しいレポート マネージャーなど)

パラメーター ボックスのカスタマイズは、待望の強化ポイントです。SQL Server 2016 からは、 次のようにパラメーターを自由に配置できるようになりました。





新しいグラフとしては、サンバースト グラフ、ツリー マップ グラフの作成が可能になりました。

レポート ビルダーも新しくなりました。



そして、Reporting Services での最も大きな変化は、**Datazen の統合**です。**Datazen** は、2015 年にマイクロソフトが買収したレポーティング ソフトウェアで、これが Reporting Services に 統合されることになりました。このソフトウェアの一番の特徴は、次のように見栄えの良いレポー トを簡単に作成でき、かつ**モバイル レポート**(スマートフォンやタブレット端末など向けのレポー ト)にも対応している点です。

	SQL S	? – 🗆 ×	
Layout Data Settings Preview	New Mobile Report	5 Grid Ro	ows 10 Grid Columns
Time Chart Category Chart Totals Chart	کیدخششا 29,821,530 ¥15 ,	054 Deta Indicator Cylinder 1	Radial Graph 2 Radial Graph 2 1,028,020 41,020,020 41,020,020 41,020,020,
Image: Comparison Time Chart Image: Comparison Time Chart Image: Comparison Time Chart Image: Chart Image: Comparison Time Chart Image: Comparison Time Chart <t< td=""><td>All (C 2) 1022 2022 2,453,800 2,453,800 4,949,750 5,863,800 4,340,500 6,585 5,863,800 7795 2,272,300 1015 3,522,800</td><td>Totals Chart 1 (秋日 21,500 二分(秋日 10,953 二分(秋日 10,953 13% 第二分(秋日 0,945 二分(秋日 0,945 13% (秋田、シリア市 10,238 12% 電子師 9,557 12% 12% 12% 12% 12% 12% 12% 12%</td><td>Pe Charl2 • Category A • Category A • Category B • Category B • Category C • Cat</td></t<>	All (C 2) 1022 2022 2,453,800 2,453,800 4,949,750 5,863,800 4,340,500 6,585 5,863,800 7795 2,272,300 1015 3,522,800	Totals Chart 1 (秋日 21,500 二分(秋日 10,953 二分(秋日 10,953 13% 第二分(秋日 0,945 二分(秋日 0,945 13% (秋田、シリア市 10,238 12% 電子師 9,557 12% 12% 12% 12% 12% 12% 12% 12%	Pe Charl2 • Category A • Category A • Category B • Category B • Category C • Cat
	왕해, 오UP가/k 3,5556,380 Bullet Graph 2 Visual Properties	5.7%	240 254 254 254 254 254 254 254 254 254 254
Gradient Heat Range Stop Heat Map Bubble Map	Title: Bullet Graph 2	Drill-Through Target: Drill-through Target	Delta Label: ValueAndPercentageFromTar 💌
DataGrids	\$25,575 + \$575 (2%) Sub-Title:	Value Direction: HigherValuesAreBetter 🔹	Range Stops: Set Ranges
In the second se	Accent: Off	Number Format: DefaultCurrency	

SQL Server 2016 CTP 3.2 自習書 No.1 新機能ダイジェスト

また、これを受けて、レポート マネージャーも、次のように変わりました。

+ttp://localhost/Reports_Previe	w/#/browse/	ク マ C 🦉 Report Manager	× 新しいレポートマネ の画面	- □ × -97- ि☆ ☺ ☺
SQL Server Reporting	g Services		\$ ₹ ?	🙂 miho ^
★ お気に入り 🗌 参照		↑ アップロード 新規	現 🗙 🛛 🔽 表示 🖌 検索	م
山ホーム				
ホーム			KPI を作成/表示 できる	
KPI 4 アイテム				
売上 KPI •••	新しい KPI ・・・ ・・・	穀類 KPI •••	菓子類KPI ••	
(+9%)	V V	(-32%)	(-17%)	
モバイル レポート 4 アイティ	6			
By Category ***	Comparison Chart ***	New Mobile Report	Tree Map	•
й язи й заи й заи й ски				
	*3,522,800	16,9% ¹⁰ 10 10 10 10 10 10 10 10 10		1
ページ分割されたレポー	ト 3 アイテム		モバイルレポートの	
★···・ ♪ サンバースト_部署別	★ *** サンバースト_部署別商品 売上	★···・ ♪ ッリーマップ_商品区分別	サムネイルを表示できる	
Datasets 6アイテム				
… 区分別クロス集計	 受注履歴	商品別売上		~

このように Reporting Services は、SQL Server 2008 R2 以来の大きな進化を遂げているので、 これらについては別途自習書を作成する予定です。

➡ Integration Services (SSIS)の強化

Integration Services は、主に次の新機能が提供されています。

- Execute SQL タスクで R スクリプトのサポート
- Azure Feature Pack の提供 (Azure BLOB/Azure HDInsight を操作可能なタスク)
- **Hadoop** (HDFS) のサポート
- インポート/エクスポート ウィザードで Azure BLOB のサポート
- データ フローでのエラー時の**列名**のサポート
- 制御フローのテンプレート作成
- AlwaysOn 可用性グループでの SSIS カタログ DB のサポート
- AutoAdjustBufferSize プロパティで、バッファ サイズの自動計算
- その他(OData v4/Excel 2013 データソースのサポートや、新しい Custom Logging Level、パッケージの増分配置 など)

その他の Integration Services の新機能については、オンライン ブックの以下のトピックが参考 になります。

What's New in Integration Services http://msdn.microsoft.com/en-us/library/bb522534.aspx

◆ Analysis Services (SSAS)の強化

Analysis Services は、主に次の新機能が提供されています。

- DirectQuery の新しい実装/大幅な性能向上
- DBCC コマンドのサポート
- Tabular Model Scripting Language (TMSL)の提供
- AMO (Analysis Services Management Objects)の変更

その他の Analysis Services の新機能については、オンライン ブックの以下のトピックが参考になります。

What's New in Analysis Services http://msdn.microsoft.com/en-us/library/bb522628.aspx

→ MDS (マスター データ サービス)の強化

MDS (マスター データ サービス) は、主に次の新機能が提供されています。

- 性能の向上
- セキュリティの強化
- 各種の機能強化(トランザクションのメンテナンス、多対多リレーションシップ、Excel アドインでのビジネス ルール管理など、多数の機能強化)

MDS には、多数の機能強化があるので、詳細については、オンライン ブックの以下のトピックを ぜひ参考にしてみてください。

What's New in Master Data Services (MDS) http://msdn.microsoft.com/en-us/library/ff929136.aspx

→ おわりに

最後まで試された皆さん、いかがでしたでしょうか? SQL Server 2016 CTP 3.2 には、たくさんの新機能が追加されていることを確認できたのではないでしょうか。インメモリ OLTP とクラスター化列ストア インデックスの融合(Operational Analytics)は、今後のデータベースのあり方を左右するのではないかと思えるほど、非常に大きな可能性を感じています(弊社のお客様でも、インメモリ OLTP に移行できるケースが多々あるのではないかとワクワクしています)。

今回は、R 統合や JSON 対応、PolyBase、Hadoop 対応など、最近のマイクロソフト社の方向性 と同様、SQL Server でもオープン化が非常に進んだ印象を持ちました(R 統合は、本当にびっく りしました)。

また、SQL Server 2016 では、セキュリティの強化や、既存機能の強化も怠ってはおらず、筆者 が特に気に入ったのは「動的データ マスク」と「透過的なデータ暗号化の性能向上&インメモリ OLTP 対応」、「可用性グループでのログ転送性能の向上」、「ライブ クエリ統計」、「クエリ ストア」 です。

セキュリティの強化と性能向上は、トレードオフなところがありますが、透過的なデータ暗号化の インメモリ OLTP 対応は、その 1 つの答え(どちらも両立できる可能性)になるかもしれません。 また、可用性グループの性能が向上した点も、大変気に入っています。これまで、弊社のお客様で は、ログ転送部分で悩みを抱えることが多かったので、今回の性能向上は本当に期待できます(本 文中で検証したものは、性能チューニングが施されていない CTP 3.2 のときに行ったものなので、 RTM 版ではさらに性能が向上している可能性もあります)。気になった機能を、皆さんも、ぜひ検 証してみていただければと思います。

5.7 付録:サンプル データベース(NorthwindJ)の作成

この自習書で利用しているサンプル データベース 「Northwind」」を作成する手順は、次のとおりです。

 まずは、Management Studio を起動するために、[スタート] メニューの [すべてのアプリ] から [Microsoft SQL Server 2016] の [Microsoft SQL Server Management Studio] をクリックします。



2. 起動後、[サーバーへの接続] ダイアログが表示されたら、[サーバー名] に SQL Server の名 前を入力して、[接続] ボタンをクリックします。

● サーバーへの接続		×
	SQL Server	
サーバーの種類(工):	データベース エンジン	SQL Server の名前を入力
サーバー名(<u>S</u>):	WIN10	~
認証(<u>A</u>):	Windows 認証	\sim
ユーザー名(山):	WIN10¥matumoto	~
パスワード(<u>P</u>):		
	パスワードを保存する(M)	
2	接続(C) キャンセル	ヘルプ オプション(<u>0</u>) >>

3. 接続完了後、次のようにツールバーの [ファイルを開く] ボタンをクリックします。

💺 Microsoft SQL Server Managemen	t Studio (管理者)						
ファイル(F) 1 写集(E) 表示(V) デパッグ(D) ツール(T) ウィンドウ(W) ヘルプ(H)							
🛅 • 🛅 🚰 📲 🎒 🕌 🗮 新しい	フエリ(N) 🔓 🐏 🏪 👗 🗐 - (Y - 💷 - 🖳 🖾 ト						
オブジェクト エクスプローラー	1県ファイルを開く	×					
接続 📲 🛃 🔳 🍸 🛃	← →	Q					
 □ WIN 10 (SQL Server 13.0.900 - ・ ・・ ・ ・・	整理 ▼ 新しいフォルター 2 サンプル スクリプトを	?					
	解凍した場所を展開して NorthwindJ.sql を選択	重類					
⊡ レプリケーション	▶ 51-ນັນ/ 2016/01/10 22:56	Microse					
Image: Imag	ローカルディスク (C:)						
・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・	PerfLogs						
📸 SQL Server エージェント (Age	Program Files						
	Program Files (x86)						
	ProgramData						
	SampleScript						
	📙 temp						
	testDB						
	Windows						
	ıザ v <	>					
	ファイル名(N): NorthwindJ.sql すべてのファイル (*.*)	~					
	開く(<u>O</u>) マ キャンセ	211					

これにより、[ファイルを開く] ダイアログが表示されるので、サンプル スクリプトを解凍したフォルダーを展開して、「NorthwindJ.sql」ファイルを選択し、[開く] ボタンをクリックします。

4. 次のようにデータベースを作成するためのスクリプトが表示されるので、ツールバーの[実行] ボタンをクリックして、スクリプトを実行します。



5. 数秒後に実行が完了して、次のように画面下に「**クエリが正常に実行されました**」と表示され ることを確認します。



以上でデータベースの作成が完了です。

→ NorthwindJ データベースの構成

NorthwindJ データベースは、Microsoft Access 2003 に付属のサンプル データベース 「Northwind」を SQL Server 上へアップサイズし、この自習書の手順を試すために、一部のデ ータを加工したものです。具体的なスキーマ構成は次のとおりです。



このデータベースは、商品の販売管理を題材として、「**商品」や「商品区分」、「受注」、「受注明細」** テーブルなどが格納されています。

執筆者プロフィール

有限会社エスキューエル・クオリティ(http://www.sqlquality.com/)

SQLQuality(エスキューエル・クオリティ)は、日本で唯一の SQL Server 専門の独立系コンサルティング 会社です。過去のバージョンから最新バージョンまでの SQL Server を知りつくし、多数の実績と豊富な経 験を持つ、OS や .NET にも詳しい SQL Server の専門家(キャリア 20 年以上)がすべての案件に対応し ます。人気メニューの「パフォーマンス チューニング サービス」は、100%の成果を上げ、過去すべてのお 客様環境で驚異的な性能向上を実現。チューニング スキルは世界トップレベルを自負、検索エンジンでは(英 語情報を含めて)ヒットしないノウハウを多数保持。ここ数年は BI/DWH システム構築支援のご依頼が多 く、支援だけでなく実際の構築も行う。

主なコンサルティング実績/構築実績

- ▶ 大手製造業の「CAD 端末の利用状況の見える化」システム構築 Oracle や CSV (Notes)、TSV ファイル、Excel からデータを抽出し、SQL Server 2012 上に DWH を構築 見える化レポートには Reporting Services を利用
- ▶ 大手映像制作会社の BI システム構築(会計/業務システムにおける予実管理/原価管理など) 従来 Excel で管理していたシートを Reporting Services のレポートへ完全移行。 Oracle や勘定奉行からデータを抽出して、SQL Server 上に DWH を構築
- ▶ 大手流通系の DWH/BI システム構築支援(POS データ/在庫データ分析/ABC 分析/ポイントカード分析)
- 大手アミューズメント企業の BI システム構築支援(人事システムにおける人材パフォーマンス管理) Reporting Services による勤怠状況の見える化レポートの作成、PostgreSQL/人事システムからのデータ抽出
- ▶ 外資系医療メーカーの BI システム構築支援(Analysis Services と Excel による販売分析システム) OLAP キューブによる売上および顧客データの多次元分析/自由分析(ユーザーによる自由操作が可能)
- ▶ 大手流通系の DWH システムのパフォーマンス チューニング データ量 100 億件の DWH、総ステップ数2万越えのストアド プロシージャのパフォーマンス チューニング
- ミッション クリティカルな金融システムでのトラブル シューティング/定期メンテナンス支援
- ▶ SQL Server の下位バージョンからの移行/アップグレード支援(32 ビットから x64 への対応も含む)
- ▶ 複数台の SQL Server の Hyper-V 仮想環境への移行支援(サーバー統合支援)
- ▶ ハードウェア リプレース時の**ハードウェア選定**(最適なサーバー、ストレージの選定)、高可用性環境の構築
- ▶ 2時間かかっていた日中バッチ実行時間を、わずか 5分へ短縮(95.8%の性能向上)
- ▶ Java 環境 (Tomcat、Seasar2、S2Dao)の SQL Server パフォーマンス チューニング etc

コンサルティング時の作業例 (パフォーマンス チューニングの場合)

- ▶ アプリケーション コード(VB、C#、Java、ASP、VBScript、VBA)の解析/改修支援
- ▶ ストアド プロシージャ/ユーザー定義関数/トリガー(Transact-SQL)の解析/改修支援
- ▶ インデックス チューニング/SQL チューニング/ロック処理の見直し
- ▶ 現状のハードウェアで将来のアクセス増にどこまで耐えられるかを測定する高負荷テストの実施
- ▶ IIS ログの解析/アプリケーション ログ(log4net/log4j)の解析
- ▶ ボトルネック ハードウェアの発見/ボトルネック SQL の発見/ボトルネック アプリケーションの発見
- ▶ SQL Server の構成オプション/データベース設定の分析/使用状況(CPU, メモリ, ディスク, Wait)解析
- ▶ 定期メンテナンス支援(インデックスの再構築/断片化解消のタイミングや断片化の事前防止策など) etc

松本美穂(まつもと・みほ)

有限会社エスキューエル・クオリティ 代表取締役 Microsoft MVP for SQL Server(2004 年 4 月~) 経産省認定データベース スペシャリスト/MCDBA/MCSD for .NET/MCITP Database Administrator SQL Server の日本における最初のバージョンである「SQL Server 4.21a」から SQL Server に携わり、現在、SQL Server を中心とするコンサルティングを行っている。得意分野はパフォーマンス チューニングと Reporting Services。著書の 『SQL Server 2000 でいってみよう』と『ASP.NET でいってみよう』(いずれも翔泳社刊)は、トップ セラー(前者は 28,500 部、後者は 16,500 部発行)。近刊に『SQL Server 2012 の教科書』(ソシム刊)がある。

松本崇博(まつもと・たかひろ)

有限会社エスキューエル・クオリティ 取締役 Microsoft MVP for SQL Server (2004 年 4 月~)

経産省認定データベース スペシャリスト/MCDBA/MCSD for .NET/MCITP Database Administrator

SQL Server の BI システムとパフォーマンス チューニングを得意とするコンサルタント。アプリケーション開発 (ASP/ ASP.NET、C#、VB 6.0、Java、Access VBA など) やシステム管理者 (IT Pro) 経験もあり、SQL Server だけでなく、 アプリケーションや OS、Web サーバーを絡めた、総合的なコンサルティングが行えるのが強み。Analysis Services と Excel による BI システムも得意とする。マイクロソフト認定トレーナー時代の 1998 年度には、Microsoft CPLS トレーナ ー アワード (Trainer of the Year) を受賞。