SQL Server テクニカル e-Book シリーズ

Microsoft

SQL Server パフォーマンス : SQL Server による クエリの高速化

SQL Server パフォーマンス : SQL Server による クエリの高速化

目次

本書の対象読者

この e-Book は、最も要求の厳しいデータ主導アプリケーションをサポートするためにクエリ処理機能を高速化するデータベース設計者、管理者、開発者を対象としています。この e-Book をお読みいただくことで、インメモリのパフォーマンス、セキュリティ、分析、柔軟性を含む高度なビルトイン処理機能を利用して SQL Server を最大限活用する方法をご理解いただけます。この e-Book では、列ストア インデックスやアダプティブ クエリ処理などのツールや機能について取り上げており、それらの機能の実行方法の技術的詳細を説明しています。

データの扱い方次第でビジネスを差別化することができます。

あらゆる場所でデータのやりとりが増えている今日、データ量に対応するだけでなく、それを活用することが重要です。

このニーズにより、組織のアプリケーションを近代化し、より優れた分析機能を搭載したデジタル トランスフォーメーションを推進する機会が作り出されます。最先端のビジネス インテリジェンス機能を使用することで、膨大で多様なデータを最大限に活用し、スマートな意思決定と迅速な実行によってビジネスのスピードを加速し、競争で優位に立つことができます。

お客様のデータやアプリケーションに使用可能な最適かつ最速のプラットフォームによって、**Microsoft SQL Server は、この目標達成を支援します。** SQL Server には、次のような重要な機能が搭載されています。

- 業界トップの [1] インメモリのパフォーマンス
- 確実で安全
- データベース内での画期的なアドバンスド アナリティクス
- あらゆるデータを含むあらゆる環境で完全なデータ資産を実行する柔軟性

[1] Gartner 社は、あらゆる運用データベース管理システムのうち、3 年連続でマイクロソフトが最も完全なビジョンと最高の能力を備えた「リーダー」であると評価しています。SQL Server ブログ、マイクロソフトは 3 年連続で ODBMS Magic Quadrant のリーダーとなる、2017 年 11 月 3 日。

SQL Server は高速なだけでなく、幅広い選択肢を提供します。SQL Server の力を Linux、Linux ベースのコンテナ、Windows に取り込むことで、お客様がそれぞれの状況に応じて開発言語、データの種類、環境 (オンプレミスまたはクラウド)、オペレーティング システムを決定できようにします。

IDC は、データ分析の対象となる世界のデータの量が

2025 年に今の 50 倍の 5.2 ZB にまで増えると予測しています。

出典 : IDC、Data Age 2025、2017 年 4 月

SQL Server には、分析パフォーマンスとクエリ処理を高速化する機能が搭載されており、データベース アプリケーションをピーク時のスピードに保ちます。

SQL Server は、主要ビジネス アプリケーションを使用したトランザクション処理で、複数の最高性能ベンチマークを保有しています。

ヒューレット・パッカード・エンタープライズ (HPE) 社は、SQL Server 2017 と Windows Server 2016 の使用による、世界新記録の TPC-H 10TB ベンチマーク[1] の結果を発表しました。これは、価格と性能において SQL Server がトップにいることを示しています。

HPE はさらに、TPC-H 3TB の結果が 1 番だったこと[2] も発表しており、データウェアハウスを含む分析クエリ ワークロードを処理する SQL Server 2017 の能力を示しました。

SQL Server は、オンライン トランザクション処理 (OLTP) ワークロードの実績を誇るリーダーです。Lenovo は最近、SQL Server 2017 と Windows Server 2016 の使用による、世界新記録の TPC-E ベンチマークの結果を発表しました[3]。これは現時点で、性能および価格 / 性能の両方でトップの TPC-E の結果です。

SQL Server は、Red Hat Enterprise Linux 上の SQL Server で、世界記録の 1TB TPC-H ベンチマーク[4] の結果 (非クラスター) を保有しています。 ■

[1] 10TB TPC-H の非クラスターでの結果 (2017 年 11 月 9 日現在)。
[2] 3TB TPC-H の非クラスターでの結果 (2017 年 11 月 9 日現在)。
[3] TPC-E のベンチマークの結果 (2017 年 11 月 9 日現在)。
[4] RHEL での TPC-H のベンチマークの結果 (2017 年 4 月現在)。

データベースは、単純なトランザクションと複雑なトランザクションの両方を実行します。

トランザクションを実行する際の複雑さの種類やレベルによっては、データベースから結果が返されるまでの時間が大幅に増える可能性があります。データベースの最適化を検討する場合は、トランザクションの種類と、データベースが十分に応答可能とみなされるために必要なスループットの量を把握することが重要です。

SQL Server には、トランザクションのスループットを向上させるためのいくつかの機能があります。その 1 つがインメモリ データ処理という機能です。トランザクションの種類によっては、これを使用することで大幅に時間を節約できる可能性があります。また、列ストア インデックスという機能もあります。これは、列ベースのデータ ストレージを活用して、膨大な量の分析可能情報を圧縮形式で整理します。これにより、行ベースのデータ ストレージで行われる B ツリー インデックス検索と比べて、検索が劇的に高速化されます。

インメモリ データ

概して、インメモリ データ処理技術では、ディスク シークやディスク読み取りの時間が節約されるため、高速化を実現できます。また、インメモリ データ処理では同時実行ロックアウトがなくなるので待ち時間が減少しますが、永続性がないために RAM の一時的な性質に対しては脆弱になります。つまり、突然の停電によってデータが失われる可能性があります。

SQL Server には、リスクを軽減しつつインメモリ データ処理技術を最大限に活用するための機能が追加されました。これにより、インメモリ OLTP をより安全に、より一層活用できるようになり、スループットが飛躍的に向上します。

トランザクションの種類によっては、インメモリ OLTP に特に向いているものがあります。トランザクションが短くて量が多い場合には、より高いパフォーマンスが得られます (特に INSERT ステートメントの処理の割合が高い場合)。たとえば、売買取引の記録、大量のモノのインターネット (IoT) またはリモート センサーのレポート、広告のクリックなどです。

SQL Server のインメモリ OLTP のいくつかの機能を組み合わせて、処理しているデータの種類に応じてパフォーマンスを最適化することができます。

メモリ最適化テーブル。インメモリ OLTP のかぎは、メモリ最適化テーブルの使用です。これは、ディスク上ではなくメモリー内に作成される特殊なテーブル データ構造です。メモリ最適化テーブルは、同時トランザクション処理にオプティミスティックなアプローチを取っていると言われています。2 つの UPDATE トランザクションが同じ行のデータに影響を与える可能性が非常に低いからです。したがって、行は更新される前にロックされることはありません。その代わりに、システムはコミット時にいくつかの簡単な検証を実行し、発生する可能性のある競合にフラグを付けます。これにより、ミリ秒ではありますが、貴重な処理時間が節約されます。

非永続テーブル。SQL Server の既定ではメモリ最適化テーブルは永続的ですが、データ損失のリスクが許容できるのであれば、非永続的に設定することもできます。これらは、クエリ結果またはキャッシングのための一時ストレージとして使用されます。

メモリ最適化テーブル変数。この機能では、インメモリ テーブルとして宣言する変数を利用できます。これらの変数は、クエリ結果を、たとえばネイティブにコンパイルまたは解釈されたストアド プロシージャなど、他のステートメントやプロシージャに渡しやすいように格納します。

ネイティブ コンパイル ストアド プロシージャ。ストアド プロシージャはネイティブ コードにコンパイルされており、メモリ最適化テーブルにアクセスできます。ストアド プロシージャは作成時にコンパイルされるため、エラー検出は実行時ではなく作成時に行われるという点で、解釈されたストアド プロシージャに比べて優れています。ネイティブ コンパイル ストアド プロシージャが処理するロジックが複雑でデータの行が多いほど、パフォーマンスが向上します。詳細については、「ネイティブ コンパイル ストアド プロシージャの使用に関するベストプラクティス」を参照してください。

ネイティブ コンパイル済みのスカラー ユーザー定義関数。UDF とも呼ばれるこれらのユーザー定義関数はネイティブ コードにコンパイルされており、メモリ最適化テーブルで高速に実行できます。このように定義された UDF はメモリ最適化されたテーブルでのみ実行可能で、従来のディスク ベース テーブルでは実行できません。

SQL Server では、以前の製品バージョンで見られたテーブルやストアド プロシージャの多くの制限が取り除かれ、インメモリ OLTP ワークロードのパフォーマンスが改善されました。2017 年に導入された機能では、アプリケーションの移行が容易になり、インメモリ OLTP の利点が活かされています。また、並列処理の結果としてスループットが向上したため、メモリ最適化テーブルではさらに高速な OLTP ワークロードがサポートされるようになりました。

SQL Server には、次の利点もあります。

メモリ最適化テーブルでの 8 つのインデックスの制限がなくなりました。ディスク ベース テーブルと同じ数のインデックスをメモリ最適化テーブルでも作成できるようになりました。以前はこの制限が原因で移行できなかったデータベース内のディスク ベース テーブルも、メモリ最適化できるようになりました。

メモリ最適化テーブルのトランザクション ログの再実行も同時に実行できます。これにより復旧時間が短縮され、AlwaysOn 可用性グループ構成の持続スループットが大幅に向上します。

データベース復旧中に `MEMORY_OPTIMIZED` テーブルに対して再構築される Bw-Tree (非クラスター化) インデックスのパフォーマンスが大幅に向上しています。この改善により、非クラスター化インデックスが使用される場合のデータベース復旧時間が大幅に短縮されます。

`sp_spaceused` がメモリ最適化テーブルでサポートされるようになりました。これは、現在のデータベース内のテーブル、インデックス付きビュー、または Service Broker キューに使用されている行数、予約済みディスク領域、およびディスク領域を表示します。または、データベース全体の予約済みおよび使用済み領域を表示します。

`sp_rename` が、メモリ最適化テーブルとネイティブ コンパイル T-SQL モジュールでサポートされるようになりました。

メモリ最適化ファイル グループのファイルを Azure Storage に保存できるようになりました。Azure Storage でのメモリ最適化ファイルのバックアップ / 復元がサポートされます。

メモリ最適化テーブルは、計算列をサポートするようになりました。ネイティブ モジュールのクエリ サーフェス領域が改良され、JSON 関数がフル サポートされるようになりました。`CROSS APPLY`、`CASE`、`TOP (N) WITH TIES` などのクエリ コンストラクトのネイティブ サポートが追加されました。

詳解 :
インメモリ OLTP

アプリケーションでインメモリ OLTP を使用できるようにするには、メモリ最適化テーブルを作成します。

```
 CREATE TABLE SupportEvent
(
    SupportEventId   int NOT NULL
        PRIMARY KEY NONCLUSTERED,
    ...
)      WITH (
        MEMORY_OPTIMIZED = ON,
        DURABILITY = SCHEMA_AND_DATA);
```

`MEMORY_OPTIMIZED = ON` 句は、テーブルをメモリ最適化と指定し、またテーブルへのすべての変更がログに記録されテーブル データがメモリに格納されることを定める `SCHEMA_AND_DATA` を指定します。すべてのメモリ最適化テーブルには、少なくとも 1 つのインデックスが含まれている必要があります。

➔ **メモリ最適化テーブルの**より詳細な情報については、「メモリ最適化テーブル」を参照してください。

ネイティブ コンパイル ストアド プロシージャを作成して、メモリ最適化テーブルのデータにアクセスすることもできます。次に構文の例を示します。

```
CREATE PROCEDURE dbo.usp_add_kitchen @dept_id int, @kitchen_
count int NOT NULL
WITH EXECUTE AS OWNER, SCHEMABINDING, NATIVE_COMPILATION
AS
BEGIN ATOMIC WITH (
TRANSACTION ISOLATION LEVEL = SNAPSHOT, LANGUAGE = N' us_
english' )

  UPDATE dbo.Departments
  SET kitchen_count = ISNULL(kitchen_count, 0) +
@kitchen_count
  WHERE id = @dept_id
END;
GO
```

NATIVE_COMPILATION なしで作成されたプロシージャは、ネイティブ コンパイル ストアド プロシージャに変更することはできません。

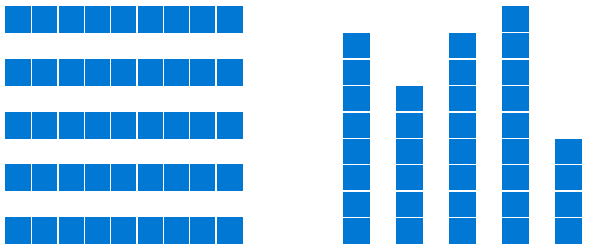
➔ **ネイティブ コンパイル ストアド プロシージャ**、サポートされているクエリのセキュリティ、および演算子のプログラマビリティの詳細については、「ネイティブ コンパイル T-SQL モジュールでサポートされる機能」を参照してください。

列ストア インデックス

列ストア インデックスは、大量のデータをカラム型形式で格納およびクエリするための技術です。これは、高速な分析クエリと大規模データベース向けの、SQL Server の最も強力な機能の 1 つです。列ストア インデックスは、カラム型データを圧縮してメモリとディスクのフットプリントを削減し、行グループの消去によりスキャンを自動的にフィルタリングし、クエリをバッチ処理することでパフォーマンスを向上させます。SQL Server の列ストア インデックスを使用すると運用分析が可能になり、トランザクション ワークロードに対してリアルタイムの分析ができるようになります。

SQL Server では、列ストア インデックス向けの機能がいくつか提供されます。

- 非クラスター化列ストア インデックス (NCCI) のオンライン再構築。
- ラージ オブジェクト (LOB) は、列ストア インデックスをサポートします。
- クラスター化列ストア インデックス (CCI) の計算列。
- Machine Learning サービスなどのクエリ オプティマイザー機能。



行として格納されたデータ **列として格納されたデータ**

詳解 : 列ストア インデックス

列ストア インデックスは、クラスター化または非クラスター化のどちらかです。クラスター化列ストア インデックス (CCI) は、テーブル全体の物理ストレージです。CCI を使用して、データ ウェアハウス ワークロードのためのファクト テーブルと大きなディメンション テーブルを格納します。非クラスター化列ストア インデックス (NCCI) は、行ストア テーブルで作成されるセカンダリ インデックスです。NCCI を使用して、OLTP ワークロードについてリアルタイムで分析を実行します。クラスター化列ストア インデックスには、1 つ以上の非クラスター化 B ツリー インデックスを含めることができます。

➔ **CCI と NCCI の**詳細については「列ストアインデックス – 概要」を参照してください。

詳解 : データウェアハウジングのための列ストアインデックス

分析クエリでは特定の値を検索するのではなく、広範囲の値に対して処理を実行することが多いため、CCI は分析クエリに適しています。CREATE TABLE ステートメントでテーブルを作成する場合、CLUSTERED COLUMNSTORE INDEX オプションを作成することによりテーブルを CCI として指定できます。

```
--Create the table
CREATE TABLE t_account (
    AccountKey int NOT NULL,
    AccountDescription nvarchar (50),
    AccountType nvarchar(50),
    UnitSold int
);
GO
--Store the table as a columnstore.
CREATE CLUSTERED COLUMNSTORE INDEX taccount_cci ON t_account;
GO
```

非クラスター化 B ツリー インデックスを、CCI 上のセカンダリ インデックスとして作成できます。データ ウェアハウスでのテーブル シークを最適化するために、次のような検索に対して最適化されたクエリを実行できる NCCI を作成できます。

```
CREATE NONCLUSTERED COLUMNSTORE INDEX taccount_nc1 ON t_
account (AccountKey);
```

詳解 : ハイブリッド トランザクション / 分析処理 (HTAP) の列ストア

ハイブリッド トランザクション / 分析処理 (HTAP) は、更新可能な行ストア テーブルの列ストア インデックスを使用します。列ストア インデックスはデータのコピーを保持するため、OLTP および分析のワークロードは、データの個別のコピーに対して実行されます。これにより、パフォーマンスを低下させることなく、トランザクション データ ワークロードに対するリアルタイムの分析処理が可能になります。各テーブルに対して、主に OLTP ワークロードについての既存の分析を高速化するように設計された B ツリー インデックスをすべて削除します。それらを孤立した列ストア インデックスで置き換えます。フィルタリングされた状態の OLTP テーブルに非クラスター化列ストアを作成するには、次の例を参照してください。

```
CREATE TABLE t_account (
    accountkey int PRIMARY KEY,
    accountdescription nvarchar (50),
    accounttype nvarchar(50),
    unitsold int
);
--Create the columnstore index with a filtered condition
CREATE NONCLUSTERED COLUMNSTORE INDEX account_NCCI
ON t_account (accountkey, accountdescription, unitsold)   ;
```

インメモリ テーブル上の列ストア インデックスにより、インメモリ OLTP とインメモリ列ストアの技術を統合することで運用分析が可能になり、これらの両方のワークロードのパフォーマンスが向上します。インメモリ テーブル上の列ストア インデックスには、すべての列を含める必要があります。この例では、列ストア インデックスを持つメモリ最適化テーブルを作成します。

```
CREATE TABLE t_account (
    accountkey int NOT NULL PRIMARY KEY NONCLUSTERED,
    Accountdescription nvarchar (50),
    accounttype nvarchar(50),
    unitsold int,
    INDEX t_account_cci CLUSTERED COLUMNSTORE
)
    WITH (MEMORY_OPTIMIZED =
ON );
GO
```

このインデックスを作成すれば、アプリケーションに変更を加えることなく HTAP を実装することができます。分析クエリは列ストア インデックスに対して実行され、OLTP の処理は引き続き OLTP B ツリー インデックスに対して実行されます。

非クラスター化列ストアのオンライン再構築

列ストア インデックスは、クエリの効率を高める上で重要な要素です。パフォーマンスを維持するにはこれらのインデックスを定期的に再構築する必要がありますが、非常に大きなインデックスの場合は時間がかかることがあります。オンライン ビジネスの場合、アプリケーションが 1 時間ダウンすると収益に影響を及ぼし、もっと悪い結果が生じることもあります。アプリケーションを実行し続けるために、SQL Server ではオンライン バックアップ、整合性チェック、インデックスの再構築がサポートされています。

SQL Server ではインデックスの構築を一時停止し、障害発生後を含め、いつでも再開できます。インデックスは使用中でも再構築が可能で、再構築を一時停止して、後で停止したところから再開することもできます。以前のリリースでのインデックス再構築操作に比べ、使用するログ領域は小さくなるという利点があります。 ■

クエリの記述が不適切な場合、アプリケーションのパフォーマンスが低下し、重要なビジネス情報の利用に影響を及ぼす可能性があります。

また、CPU、メモリ、ネットワークなどのリソースの使用効率が下がることもあります。クエリ実行プランの回帰もパフォーマンスに大きな影響を与えます。これらは、アプリケーションの変更や古いデータベースの統計がある場合、行数の見積もりが不正確な場合に発生します。極めて強力なハードウェア上でデータベース サーバーを実行している場合でも、適切に動作しない少しのクエリがパフォーマンスに悪い影響を与えることがあります。実のところ、1 つの不正なクエリがデータベースのパフォーマンスに重大な問題を発生させることもあります。

クエリのパフォーマンスは多くの要因の影響を受けますが、その 1 つはクエリ プランです。不適切なクエリをチューニングおよび最適化する場合、DBA は通常そのクエリの実行プランの調査から始めてそれを評価し、データの予測に基づいて最善で最も効率的なプランを決定します。このプロセスを支援するために、SQL Server はクエリ プランの動作方法を変更するクエリ処理機能とパフォーマンス機能を提供します。

Query Store の改善により、待機統計の概要情報を追跡でき、トラブルシューティングにかかる時間を短縮できます。

アダプティブ クエリ処理 (AQP) は、実行結果に基づいてクエリ プランのエラーを軽減して実行プランを適応させることにより、SQL Server の実行プランを最適化する方法です。

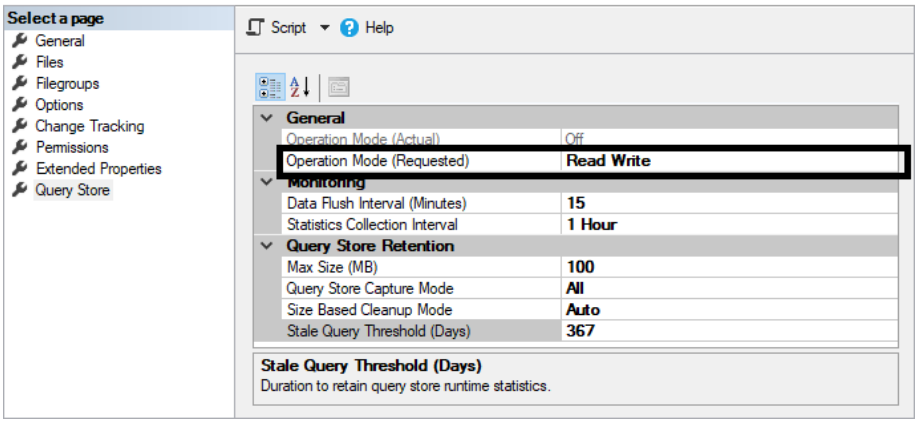
Query Store

Query Store はコンパイル時および実行時の統計に関するテレメトリを収集します。また、クエリとプランの履歴を確認のために取得します。データは時間枠で区切られているため、データベースの使用パターンを確認し、サーバー上でクエリ プランの変更がいつ発生したかを把握できます。

待機統計は、SQL Server のパフォーマンスの問題をトラブルシューティングするのに役立つもう 1 つの情報源です。過去には、待機統計はインスタンス レベルでしか利用できなかったため、実際のクエリの詳細を調べることは困難でした。Query Store では概要情報が提供されるため、待機統計をより効率的に追跡するのに役立ちます。待機統計はクエリ プランと関連付けられており、実行時統計と同様、時の経過とともに収集されます。これにより、Query Store の利点を残しつつワークロードのパフォーマンスとボトルネックに関するインサイトが得られます。

詳解 : SQL Server Management Studio または Query Store 向け Transact-SQL 構文の使用

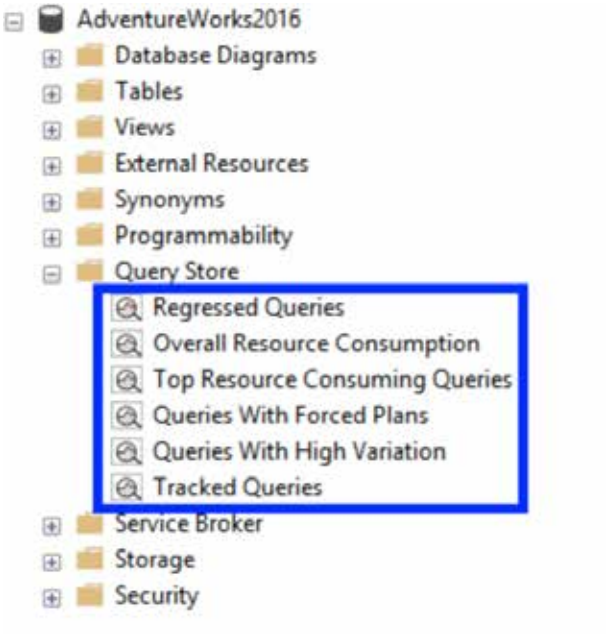
SQL Server Management Studio (SSMS) は、Query Store を構成するため、またワークロードに関する収集されたデータを消費するために設計された一連のユーザー インターフェイスをホストします。SSMS のオブジェクト エクスプローラーでは、**[処理モード (要求済み)]** ボックスを選択することにより、Query Store を有効にできます。



また、`ALTER DATABASE` ステートメントを使用して Query Store を実装することもできます。例 :

```
ALTER DATABASE AdventureWorks2012 SET QUERY_STORE = ON;
```

これらのオプションのいずれかを選択した後、[オブジェクト エクスプローラー] ペインのデータベースの部分を更新して **Query Store** セクションを追加します。Query Store のレポートを表示できるようになります。



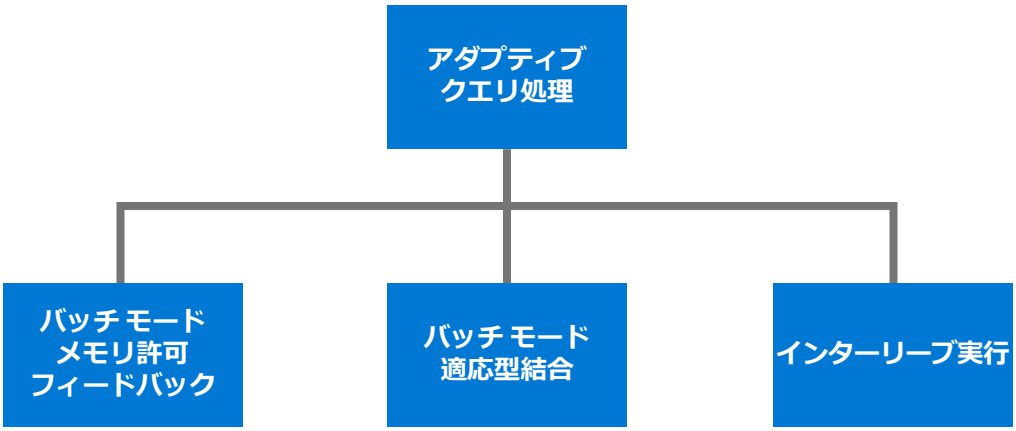
➔ **Query Store を使用して**パフォーマンスを監視する方法の詳細については、Microsoft ドキュメントを参照してください。

アダプティブ クエリ処理

クエリ処理と最適化の実行時に、基数見積 (CE) プロセスは、実行プランの各ステップで処理される行数の見積もりを行います。見積りが不正確な場合、クエリ応答時間の遅れ、過剰なリソースの使用 (CPU、メモリ、IO)、スループットと同時実行の低下を招く場合があります。

CE プロセスを改善するために、SQL Server はアダプティブ クエリ処理 (AQP) と呼ばれる機能ファミリーを提供します。AQP は、クエリ プロセッサが実行時の特性に基づいてクエリ プランの選択を調整できるようにすることで、SQL Server によるワークロードの処理を大幅に高速化します。AQP は、クエリ プランと実際の実行との間の壁をなくします。最適化は、クエリの実行中に実施することもできますし、以降のクエリ実行に役立つように実行完了後に実施することもできます。AQP には、アプリケーションのワークロード特性に適応するための 3 つの手法があります。

- バッチ モード メモリ許可フィードバック。
- バッチ モード適応型結合。
- 複数ステートメントのテーブル値関数のインターリーブ実行。



バッチ モード メモリ許可フィードバック

クエリの実行後のプランでは、SQL Server は特定の T-SQL バッチの基数推定を実行し、実行に必要な最小メモリ許可と、バッチのすべての行をメモリに保持するために必要な理想的なメモリ許可を推定します。CE に問題があると、パフォーマンスが低下し、使用可能なメモリが制限されます。過剰なメモリ許可は、メモリの無駄遣いと同時実行の減少を招きます。不十分なメモリ許可は、弊害の多いディスクへのスピルを引き起こします。

バッチ モード メモリ許可フィードバックにより、SQL Server はクエリに実際に必要なメモリ量を再計算し、キャッシュされたプランの許可値を更新します。同一のクエリ ステートメントが実行される場合は、クエリは修正されたメモリ許可サイズを使用します。tempdb へのスピルが減ってバッチへのメモリ許可がより適切になるため、メモリを最も必要としているバッチに追加のメモリを付与することが可能となり、パフォーマンスが向上します。

過剰に許可した場合、許可されたメモリが使用されたメモリの 2 倍を超えるサイズであれば、メモリ許可フィードバックは再計算してキャッシュされたプランを更新します。メモリ許可が 1 MB 未満のプランは、超過分のための再計算をしません。バッチ モード演算子に対してディスクへのスピルの引き起こす、**不十分なサイズ**のメモリ許可の場合、メモリ許可フィードバックは再計算を実行します。スピル イベントは、メモリ許可フィードバックに報告されます。このイベントは、プランからのノード ID と、そのノードのスピルされたデータ サイズを返します。

バッチ モード 適応型結合

SQL Server は通常、ネスト ループ結合、マージ結合、ハッシュ結合の 3 種類の物理的な結合演算子から選択します。それぞれの種類の結合には、データ パターンとクエリ パターンの特性に応じて長所と短所があります。各クエリで最適なアルゴリズムは、結合入力の基数推定によって異なります。入力された CE が不正確だと、不適切な結合アルゴリズムが選択される可能性があります。

バッチ モード適応型結合機能により、SQL Server では、最初の入力のスキャンが**完了する**まで、ハッシュ結合法またはネスト ループ結合法が選択されないようにすることができます。適応型結合演算子は、ネスト ループ プランに切り替えるタイミングの決定に使用されるしきい値を定義します。結果として、プランは実行中に、より良い結合方法に動的に切り替わります。

複数ステートメントのテーブル値関数のインターリーブ実行

複数ステートメントのテーブル値関数 (MSTVF) は開発者の間では一般的ですが、その初期実行ではパフォーマンスが低下する可能性があります。AQP を活用して、SQL Server は MSTVF 機能のインターリーブ実行によってこの問題を解決します。この機能により、単一クエリ実行の最適化フェーズと実行フェーズの間の一方向境界が変更され、修正された基数推定に基づくプランの適応が可能になります。インターリーブ実行では、MSTVF からの実際の行数が、MSTVF 参照から下流のプラン最適化に使用されます。その結果、実際のワークロードの特性に基づいて、より詳細なプランが作成され、最終的にはクエリのパフォーマンスが向上します。

MSTVF があると、クエリ オプティマイザーは以下の処理を行います。

- 一時停止の最適化。
- 正確な CE を取得するために、MSTVF サブツリーを実行します。
- 一連の正確な前提により、後続の操作の処理を継続します。

実行結果 (推定行数) に基づいて、クエリ オプティマイザーはより良いプランを検討し、修正されたプランでクエリを実行します。

一般に、推定行数と実際の行数のずれが大きいほど、下流のプラン処理の数と相まって、パフォーマンスへの影響が大きくなります。**次の両方が当てはまる場合、インターリーブ実行はクエリにとって利点となります。**

- 中間結果セット (この場合は MSTVF) の推定行数と実際の行数とのずれが大きい場合。
- 全体的なクエリが、中間結果のサイズの変化の影響を受けやすい場合。これは通常、クエリ プランのサブツリー上に複雑なツリーがある場合に発生します。MSTVF からの単純な「`SELECT *`」では、インターリーブ実行にメリットはありません。

詳解 :
AQP の有効化

データベースの互換性レベル 140 を有効にすることにより、ワークロードを自動的に AQP に適応させることができます。次に、T-SQL を使用してこれを設定する方法の例を示します。

```
ALTER DATABASE [YourDatabaseName]
SET COMPATIBILITY_LEVEL = 140;
```

➔ **これらの機能の**使い方の詳細については、「SQL データベースにおけるアダプティブ クエリ処理」を参照してください。■

ほかにも、機能の構成オプション、監視機能やチューニング機能など、パフォーマンスの監視と最適化のための SQL Server でサポートされているツールや機能があります。

パフォーマンスのための機能構成オプション

SQL Server では、パフォーマンスをさらに向上させるためのディスク、サーバー、テーブル、およびクエリに関するさまざまな構成オプションがデータベース エンジン レベルで提供されます。

ディスクの構成。SQL Server では、レベル 0、1、5 の RAID (Redundant Array of Independent Disks) を設定できます。SQL Server では、ディスク ストリップには RAID 0、ディスク ミラーリングには RAID 1、パリティ付きストライピングには RAID 5 を設定します。

Tempdb の構成。tempdb のパフォーマンスを最適化するには、データベース ファイルの瞬時初期化、自動拡張、ディスク ストライピングなどのオプションを使用して、共有ネットワーク ドライブではなくローカル ドライブに tempdb を保存します。詳細については、「SQL Server での tempdb パフォーマンスの最適化」を参照してください。

サーバーの構成。プロセッサ、メモリ、インデックス、バックアップ、およびクエリの設定を構成して、SQL Server でのサーバー パフォーマンスを向上させることができます。これらの設定には、MAXDOP、最大サーバー メモリ、アドホック ワークロードの最適化、ネストされたトリガーなど、最大限の並列処理のためのオプションが含まれます。詳細については、「パフォーマンスのための設定オプション」を参照してください。

データベースの構成。データベースのパフォーマンスを最適化するために、行ストアおよび列ストアのテーブルおよびインデックスに対してデータ圧縮機能を使用して、行圧縮とページ圧縮を設定します。また、要件に基づいてデータベースの互換性レベルを変更することもできます。

テーブルの構成。テーブルのパフォーマンスを向上するために、パーティション テーブルとパーティション インデックスを使用できます。

クエリ パフォーマンスのオプション。クエリ レベルのパフォーマンスを向上させるために、インデックス、パーティション、ストアド プロシージャ、UDF、および統計を使用できます。メモリ最適化テーブルやネイティブ コンパイル ストアド プロシージャなどの前述の機能を使用して、インメモリ OLTP パフォーマンスを向上させます。詳細については、「クエリ パフォーマンスのオプション」を参照してください。

パフォーマンスのための監視機能とチューニング機能

Query Store、実行プラン、ライブ クエリ統計、およびデータベース エンジン チューニング アドバイザーのようなツールを使用すると、SQL Server イベントを監視できます。また、エンジン プロセス イベントを追跡する sp_trace_setfilter や、トレース フラグを有効にするコマンドである DBCC TRACEON などのさまざまな T-SQL コマンドを使用できます。また、sp_configure を使用してパフォーマンスのベースラインを設定して、SQL Server システムが最適に動作しているかどうかを判断することもできます。詳細については、「パフォーマンスの監視ツールとチューニング ツール」を参照してください。

Resource Governor

Resource Governor は、システムのリソース消費量に対する制限を管理したり指定したりするのに役立つ SQL Server のツールです。受信アプリケーション要求が使用できる CPU、物理 I / O、およびメモリのリソース制限を単純に定義できます。Resource Governor を使用すると、リソースの使用パターンを継続的に観察し、それに応じてシステム設定を調整して効率を最大化できます。使用方法と仕組みについての詳細は、Resource Governor のドキュメントを参照してください。 ■

SQL Server 2017 は最も要求の厳しいデータ主導のアプリケーションをサポートするために、高速な処理機能を提供します。これには、業界をリードする高度なパフォーマンス機能を基盤とした、独自の機能セットが含まれています。

インメモリ OLTP は、より高速なオンライン トランザクション処理のワークロードと優れたスループットを実現します。列ストア インデックスは、データベースのパフォーマンスを向上させ、高速な分析を可能にします。さらに、アダプティブ クエリ処理などの機能により、DBA はクエリ処理機能をさらに最適化できます。SQL Server 2017 は、企業がユーザーの要求に応え続けるために必要なスピード、機能、スケーラビリティを提供します。■

使い慣れたプラットフォームで SQL Server を実行する方法をご覧ください。

SQL Server の最新機能について学んでください。

SQL 業界のベンチマークとパフォーマンスをご覧ください。

SQL Server をダウンロードする。