SQL パフォーマンス・オプションの構成

Version 2023.1
2024-01-02

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

SQL パフォーマンス・オプションの構成
InterSystems IRIS Data Platform　Version 2023.1　2024-01-02

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください：

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

1

並列クエリ処理の構成

並列クエリ・ヒントは、マルチプロセッサ・システムで実行されている場合にクエリの並列処理を実行するようシステムに指示します。これにより、特定のタイプのクエリのパフォーマンスを大幅に向上させることができます。SQL オプティマイザは、特定のクエリに並列処理の効果があるかどうかを確認し、該当する場合は並列処理を実行します。並列クエリのヒントを指定しても、すべてのクエリの並列処理が強制されるわけではなく、並列処理によって効果が得られると考えられるクエリのみが対象になります。システムがマルチプロセッサ・システムでない場合、このオプションに効果はありません。現在のシステム上のプロセッサ数を特定するには、%SYSTEM.Util.NumberOfCPUs() メソッドを使用します。

既定では、アダプティブ・モードで並列クエリ処理が制御されます。アダプティブ・モードを無効にしている場合は、以下の 2 つの方法で並列クエリ処理を指定できます。

- 自動並列オプションを設定して、システム全体で適用します。
- 個々のクエリの FROM 節で %PARALLEL キーワードを指定して、クエリ単位で適用します。

クエリの並列処理は、SELECT クエリに適用されます。INSERT、UPDATE、DELETE の各操作には適用されません。

$job や $tlevel などのプロセス固有の関数が関係するクエリでは並列処理を避けてください。また、$ROWID のようなプロセス固有の変数のクエリでも並列処理を避けるようにします。

1.1 システム全体のクエリの並列処理

アダプティブ・モードが無効でも、以下のオプションのいずれかを使用することで、システム規模の並列クエリ処理を有効にすることができます。

- 管理ポータルから、**[システム管理]**、**[構成]**、**[SQL およびオブジェクトの設定]**、[SQL] の順に選択します。**[単一プロセスでクエリを実行する]** チェック・ボックスを確認または変更します。このチェック・ボックスには既定ではチェックが付いていません。つまり、並列処理は既定で有効になっています。
- $SYSTEM.SQL.Util.SetOption() メソッドを、`SET status=$SYSTEM.SQL.Util.SetOption("AutoParallel",1,.oldval)` のように呼び出します。既定値は 1 です（自動並列処理が有効）。現在の設定を確認するには、$SYSTEM.SQL.CurrentSettings() を呼び出します。これにより、**[%PARALLEL]** オプションが表示されます。

この構成設定を変更すると、すべてのネームスペースですべてのクエリ・キャッシュが削除されることに注意してください。

システム全体の並列クエリ処理の詳細は、"アダプティブ・モードの使用によるパフォーマンスの向上" の "システム規模の自動並列化" を参照してください。

注釈　%Runtime モードでコンパイルすると論理形式と表示形式または ODBC 形式との間で変換されるリテラル変数が使用されていて、ODBC モードでは実行されないサーバ側 SQL クエリを並列実行すると、正しくない結果が返されることが考えられます。このような状態を防止するには、該当のクエリに %NOPARALLEL を追加します。

1.2 特定のクエリの並列クエリ処理

オプションの %PARALLEL キーワードは、クエリの FROM 節で指定します。これは、InterSystems IRIS が複数のプロセッサを使用してクエリの並列処理を実行することを示しています (該当する場合)。これにより、1 つ以上の COUNT、SUM、AVG、MAX、または MIN 集約関数または GROUP BY 節 (あるいはその両方) を使用するクエリや、他の多くのタイプのクエリで、パフォーマンスを大幅に向上させることができます。一般にこれらは、大量のデータを処理し、小規模な結果セットを返すクエリです。例えば、`SELECT AVG(SaleAmt) FROM %PARALLEL User.AllSales GROUP BY Region` は、並列処理を使用する可能性が高くなります。

集約関数、式、およびサブクエリのみを指定する "1 行" のクエリは、GROUP BY 節の有無にかかわらず並列処理を実行します。ただし、個々のフィールドと 1 つ以上の集約関数の両方を指定する "複数行" のクエリは、GROUP BY 節が含まれていないときには並列処理を実行しません。例えば、`SELECT Name,AVG(Age) FROM %PARALLEL Sample.Person` は並列処理を実行しませんが、`SELECT Name,AVG(Age) FROM %PARALLEL Sample.Person GROUP BY Home_State` は並列処理を実行します。

%PARALLEL を指定しているクエリが実行時モードでコンパイルされると、すべての定数は ODBC 形式であるものとして解釈されます。

%PARALLEL を指定すると、クエリによってはパフォーマンスが低下する可能性があります。複数の同時ユーザがいるシステム上で %PARALLEL を指定してクエリを実行すると、総合的なパフォーマンスが低下する可能性があります。

ビューのクエリを実行する際に、並列処理を実行できます。ただし、%PARALLEL キーワードを明示的に指定していても、%VID を指定するクエリに対して並列処理が実行されることはありません。

詳細は、"InterSystems SQL リファレンス" の "FROM" 節を参照してください。

1.2.1 サブクエリ内の %PARALLEL

%PARALLEL は、SELECT クエリとそのサブクエリで使用するためのものです。INSERT コマンド・サブクエリは %PARALLEL を使用できません。

%PARALLEL は、括弧で囲まれたクエリと相互に関連しているサブクエリに適用された場合は無視されます。以下はその例です。

SQL

```
SELECT name,age FROM Sample.Person AS p
WHERE 30<(SELECT AVG(age) FROM %PARALLEL Sample.Employee WHERE Name = p.Name)
```

%PARALLEL は、複合述語を含むサブクエリ、または複合述語へと最適化する述語を含むサブクエリに適用された場合は無視されます。複合と見なされる述語には、FOR SOME および FOR SOME %ELEMENT 述語があります。

1.3 クエリの並列処理の無視

自動並列オプションが設定されているかどうかや、%PARALLEL キーワードが FROM 節に指定されているかどうかに関係なく、クエリによっては、並列処理ではなく線形処理が使用されることがあります。InterSystems IRIS では、他のクエリ最適化オプションが指定されていれば、それを適用した後でクエリに並列処理を使用するかどうかが決まります。Inter-

Systems IRIS は、ユーザ指定のクエリ形式が並列処理を実行すると利点があるように思える場合でも、最適化された形式のクエリが並列処理に適していないと判断することがあります。InterSystems IRIS が並列処理のためにクエリを分配したか、またはどのようにそうしたかは、プラン表示を使用して確認できます。

以下の状況では、%PARALLEL を指定しても並列処理は実行されません。クエリは正常に実行されてエラーは発行されませんが、並列処理は実行されません。

- FOR SOME 述語を含むクエリ。
- TOP 節と ORDER BY 節の両方を含むクエリ。この節の組み合わせでは、最初の行に移動する時間が最短になるように最適化されており、並列処理は使用されません。FROM 節の %NOTOPOPT optimize-option キーワードを追加することで、完全な結果セットの取得が最速になるように最適化されます。クエリに集約関数が含まれていない場合、この %PARALLEL と %NOTOPOPT の組み合わせにより、クエリの並列処理が実行されます。
- ON 節が等値条件ではない LEFT OUTER JOIN または INNER JOIN を含むクエリ。例えば、`FROM %PARALLEL Sample.Person p LEFT OUTER JOIN Sample.Employee e ON p.dob > e.dob` などです。これは、SQL 最適化により、このタイプの結合が FULL OUTER JOIN に変換されるために起きます。%PARALLEL は FULL OUTER JOIN の場合は無視されます。
- %PARALLEL の最適化と%INORDER の最適化は同時に使用できません。両方を指定すると、%PARALLEL は無視されます。
- ビューを参照し、ビュー ID (%VID) を返すクエリ。
- テーブルに BITMAPEXTENT インデックスがある場合、COUNT(*) は並列処理を使用しません。
- %PARALLEL は、標準的なデータ・ストレージ定義を使用するテーブルで使用するためのものです。カスタマイズされたストレージ形式での使用はサポートされない場合があります。%PARALLEL は、GLOBAL TEMPORARY テーブル、または拡張グローバル参照ストレージがあるテーブルではサポートされません。
- %PARALLEL は、テーブルのすべての行にアクセスできるクエリ用であり、行レベル・セキュリティ (ROWLEVELSECURITY) が定義されているテーブルは並列処理を実行できません。
- %PARALLEL は、ローカル・データベースに格納されているデータを処理するためのものです。リモート・データベースにマップされているグローバル・ノードはサポートされません。

1.4 共有メモリの考慮事項

並列処理では、InterSystems IRIS は複数のプロセス間キュー (IPQ) をサポートしています。各 IPQ が 1 つの並列クエリを処理します。これを使用することで、並列の作業ユニット・サブプロセスは、データの行をメイン・プロセスに送り返し、作業ユニットの完了をメイン・プロセスが待機する必要がなくなるようにすることができます。これにより、並列クエリは、クエリ全体の完了を待機することなく、できる限り迅速にデータの最初の行を返せるようになります。さらに、集約関数のパフォーマンスも向上します。

クエリの並列実行には、一般メモリ・ヒープ (gmheap) の共有メモリを使用します。SQL クエリの並列実行を使用するユーザは、gmheap のサイズを大きくすることが必要になる場合があります。原則として、各 IPQ のメモリ要件は 4 x 64k = 256k です。InterSystems IRIS は、並列 SQL クエリを使用可能な CPU コア数に分割します。そのため、この追加の gmheap をユーザが割り当てる必要があります。

<Number of concurrent parallel SQL requests> x <Number cores> x 256 = <required size increase (in kilobytes) of gmheap>

この式に 100% の正確性はありません。並列クエリは同じく並列のサブ・クエリを生成することがあるためです。したがって、この式で指定された値よりも多くの gmheap を割り当てることが賢明です。

適切な gmheap の割り当てに失敗すると、messages.log にエラーが報告されます。SQL クエリは失敗する可能性があります。別のサブシステムが gmheap を割り当てようとすると、その他のエラーも発生する可能性があります。

インスタンスごとの gmheap の使用率 (特に IPQ の使用率など) を確認するには、管理ポータルのホーム・ページから **[システム処理]**、**[システム使用]** の順に選択して、**[共有メモリヒープ使用状況]** リンクをクリックします。詳細は、″監視ガイド″ の “管理ポータルを使用した InterSystems IRIS の監視” の章にある ″一般 (共有) メモリ・ヒープ使用状況″ を参照してください。

gmheap (共有メモリ・ヒープまたは SMH とも呼びます) のサイズを変更するには、管理ポータルのホーム・ページで **[システム管理]**、**[構成]**、**[追加設定]**、**[メモリ詳細]** の順に選択します。詳細は、″システム管理ガイド″ の ″“InterSystems IRIS の構成”″ の章で ″メモリと開始設定″ を参照してください。

1.5 クエリ・キャッシュの考慮事項

%PARALLEL が指定されたキャッシュ済み SQL クエリを実行しているとき、クエリ・キャッシュが削除されるような処理をこの SQL クエリの初期化中に実行すると、その SQL クエリは、いずれかのワーカ・ジョブから報告された <NOROUTINE> エラーを受け取ることがあります。クエリ・キャッシュが削除される原因となる主なアクションとして、$SYSTEM.SQL.Purge() の呼び出しや、そのクエリが参照するクラスのリコンパイルなどがあります。クラスをリコンパイルすると、そのクラスに関連するクエリ・キャッシュは自動的に削除されます。

通常、このエラーが発生した場合は、もう一度クエリを実行することで正常に実行されます。クエリから %PARALLEL を削除すると、このエラーは発生しなくなります。

1.6 SQL 文とプランの状態

%PARALLEL を使用する SQL クエリは、複数の SQL 文になることがあります。これに該当する SQL 文の **[プランの状態]** は、[未凍結/並列] になります。プランの状態が [未凍結/並列] のクエリは、ユーザの操作で凍結することはできません。詳細は、“SQL 文” の章を参照してください。

2

実行時プラン選択の使用法

実行時プラン選択 (RTPC) は、SQL オプティマイザが実行時 (クエリの実行時) に異常値情報を利用できるようにするための構成オプションです。アダプティブ・モードが有効であれば RTPC が有効になりますが、アダプティブ・モードを無効にしていても RTPC を有効にすることはできます。

RTPC が有効である場合、クエリ作成では、異常値があるフィールドに関する条件がクエリで指定されているかどうかが検出されます。異常値フィールドの条件が 1 つ以上検出されると、クエリはオプティマイザに送信されず、実行時プラン選択スタブが生成されます。実行時に、オプティマイザはこのスタブを使用して、実行するクエリ・プランを選択します。このプランは、異常値ステータスを無視する標準クエリ・プランまたは異常値ステータスに対して最適化された代替クエリ・プランです。異常値条件が複数ある場合、オプティマイザは、複数の代替実行時クエリ・プランからいずれかを選択できます。

RTPC クエリ・プランの表示は、SQL コードのソースによって異なります。

- **管理ポータル**の SQL インタフェースにある [プラン表示] ボタンには、代替の実行時クエリ・プランが表示されることがあります。この [プラン表示] の SQL コードが SQL インタフェースのテキスト・ボックスから取得されるからです。
- SQL 文を選択すると、クエリ・プランを含む文の詳細が表示されます。このクエリ・プランには、代替実行時クエリ・プランは表示されませんが、代わりに “`execution may cause creation of a different plan`” というテキストが含まれています。これは、その SQL コードを文インデックスから取得するからです。
- RTPC が有効でない場合、または該当する異常値フィールド条件がクエリに含まれていない場合、オプティマイザは、標準 SQL 文および対応するクエリ・キャッシュを作成します。
- RTPC スタブが凍結されている場合、関連付けられている代替実行時クエリ・プランもすべて凍結されます。RTPC 構成オプションがオフになっていても、RTPC 処理は凍結されたクエリに対して有効のままです。
- `SELECT Name,HaveContactInfo FROM t1 WHERE HaveContactInfo=(('Yes'))` のように、括弧を指定すると、クエリを記述する際に手動でリテラル置換を抑制できます。条件で異常値フィールドのリテラル置換を抑制した場合、そのクエリには RTPC は適用されません。この場合は、リテラル変換を指定した標準のクエリ・キャッシュがオプティマイザによって作成されます。

2.1 RTPC の適用

SELECT 文と CALL 文の場合は、テーブル・チューニングによって異常値があると判断されたフィールドが、そのフィールドをリテラルと比較する条件で指定されていれば、そのようなすべてのフィールドに RTPC が適用されます。その比較条件は以下のいずれかです。

- 等値 (=)、非等値 (!=)、IN、または %INLIST 述語を使用する WHERE 節の条件

- 等値 (=)、非等値 (!=)、IN、または %INLIST 述語を使用する ON 節の結合条件

RTPC が適用された場合、オプティマイザは、標準クエリ・プランと代替クエリ・プランのどちらを適用するかを実行時に決定します。

RTPC は、INSERT、UPDATE、DELETE の各文には適用されず、以下の場合も適用されません。

- クエリに未解決の ? 入力パラメータがある場合。
- 二重括弧で囲まれたリテラル値がクエリで指定され、リテラル置換が抑制されている場合。
- 異常値フィールド条件が設定されていないサブクエリによって、異常値フィールド条件に対してリテラルが指定されている場合。

2.2 RTPC のオーバーライドまたは無効化

%NORUNTIME 制約キーワードを指定することにより、特定のクエリについて RTPC をオーバーライドできます。`SELECT Name,HaveContactInfo FROM t1 WHERE HaveContactInfo=?` というクエリが RTPC で処理される場合、`SELECT %NORUNTIME Name,HaveContactInfo FROM t1 WHERE HaveContactInfo=?` というクエリによって RTPC がオーバーライドされ、標準クエリ・プランになります。

アダプティブ・モードが無効であれば、$SYSTEM.SQL.Util.SetOption() メソッドを `SET status=$SYSTEM.SQL.Util.SetOption("RTPC",flag,.oldval)` のように使用すると、システム全体ですべてのプロセスに対して RTPC を有効または無効にすることができます。flag 引数は、RTPC を設定 (1) または設定解除 (0) するために使用するブーリアン値です。oldvalue 引数は、前の RTPC 設定をブーリアン値として返します。

現在の設定を確認するには、$SYSTEM.SQL.CurrentSettings() を呼び出します。

3

凍結プランの構成

ほとんどの SQL 文には、クエリ・プランが関連付けられています。クエリ・プランは、SQL 文が準備されるときに作成されます。既定で、インデックスの追加やクラスのリコンパイルなどの操作によってこのクエリ・プランは削除されます。次回にクエリが呼び出されたときに、クエリが再準備され、新しいクエリ・プランが作成されます。凍結プランにより、コンパイルの前後で既存のクエリ・プランを維持 (凍結) できます。クエリの実行で、新しい最適化を実行して新しいクエリ・プランを生成するのではなく、凍結プランが使用されるようになります。

システム・ソフトウェアへの変更によって、別のクエリ・プランが作成される場合もあります。通常、こういったアップグレードにより、クエリ・パフォーマンスが向上しますが、ソフトウェアのアップグレードによって特定のクエリのパフォーマンスが低下する場合があります。凍結プランを使用すると、クエリ・プランを維持 (凍結) して、クエリ・パフォーマンスがシステム・ソフトウェアのアップグレードによって変化 (低下または向上) しないようにすることができます。

3.1 凍結プランの使用方法

凍結プランの使用方法には、楽観的方法と悲観的方法という 2 つの方法があります。

- 楽観的 : この方法は、システム・ソフトウェアかクラス定義への変更でパフォーマンスが向上することを想定している場合に使用します。クエリを実行して、プランを凍結します。凍結プランをエクスポート (バックアップ) します。プランの凍結を解除します。ソフトウェアの変更を行います。クエリを再実行します。これにより、新しいプランが生成されます。2 つのクエリのパフォーマンスを比較します。新しいプランではパフォーマンスが向上しない場合は、バックアップ・ファイルから以前の凍結プランをインポートします。
- 悲観的 : この方法は、システム・ソフトウェアかクラス定義への変更で、特定のクエリのパフォーマンスが向上する可能性がほとんどないことを想定している場合に使用します。クエリを実行して、プランを凍結します。ソフトウェアの変更を行います。%NOFPLAN キーワードを指定してクエリを再実行します (凍結プランは無視されます)。2 つのクエリのパフォーマンスを比較します。凍結プランを無視してもパフォーマンスが向上しない場合は、プランを凍結したままにして、クエリから %NOFPLAN を削除します。

3.2 ソフトウェアのバージョンをアップグレードした後の凍結プラン

InterSystems IRIS® のデータ・プラットフォームを新しいメジャー・バージョンにアップグレードすると、既定では既存のクエリ・プランは自動的に凍結されます。つまり、ソフトウェアのメジャー・アップグレードによって既存のクエリのパフォーマンスが低下することがあります。ソフトウェアのバージョンをアップグレードする前に、パフォーマンスが重要なクエリをすべて手動で凍結することを検討します。アップグレード後に、これらのクエリに以下の手順を実行します。

1. プランの状態を [凍結/明示] としてクエリを実行し、パフォーマンスを監視します。これは、プランを明示的に凍結する前に作成した最適化済みクエリ・プランです。
2. %NOFPLAN キーワードをクエリに追加して実行し、パフォーマンスを監視します。これにより、ソフトウェアのアップグレードで提供された SQL オプティマイザを使用して、クエリ・プランが最適化されます。既存のクエリ・プランの凍結は解除されません。
3. パフォーマンス・メトリックを比較します。
   - %NOFPLAN のパフォーマンスが優れている場合、ソフトウェアのアップグレードによってクエリ・プランの効率が向上しています。クエリ・プランの凍結を解除します。%NOFPLAN キーワードを削除します。
   - %NOFPLAN のパフォーマンスが劣る場合、ソフトウェアのアップグレードによってクエリ・プランの効率が低下しています。クエリ・プランを凍結状態のままとして、%NOFPLAN キーワードを削除します。
4. パフォーマンスが重要なクエリのテストが終了すれば、凍結した残りのプランをすべて凍結解除してもかまいません。

自動凍結を有効にしている場合は、プランを作成した InterSystems ソフトウェアよりも新しいバージョンのソフトウェアでクエリを準備またはコンパイルしたときに自動凍結が実行されます。例えば、システム・ソフトウェア・バージョン xxxx.1 で準備またはコンパイルした SQL 文を考えます。その後でバージョン xxxx.2 にアップグレードし、その SQL 文を再び準備/コンパイルします。システムは、これが新しいバージョンで実施されるその SQL 文の最初の準備/コンパイルになることを検出し、そのプランに自動的に [凍結/アップグレード] のマークを付けて、今回の準備/コンパイルに既存のプランを使用します。これにより、前のバージョンのクエリ・プランと同程度のクエリ・プランが使用されるようになります。

メジャー・バージョンの InterSystems システム・ソフトウェアのアップグレードでのみ、既存のクエリ・プランが自動的に凍結されます。メンテナンス・リリース・バージョンのアップグレードでは、既存のクエリ・プランは凍結されません。例えば、2018.1 から 2019.1 などのようなメジャー・バージョンのアップグレードでこの処理が実行されます。2018.1.0 から 2018.1.1 などのようなメンテナンス・リリース・バージョンのアップグレードでは、この処理は実行されません。

INFORMATION.SCHEMA.STATEMENTS Frozen=2 プロパティを使用して、現在のネームスペースの凍結/アップグレード・プランをすべてリスト表示できます。

次の $SYSTEM.SQL.Statement メソッドを使用して 1 つのクエリ・プランまたは複数のクエリ・プランを凍結できます：1 つのプランの場合は FreezeStatement()、関係のすべてのプランの場合は FreezeRelation()、スキーマのすべてのプランの場合は FreezeSchema()、現在のネームスペースのすべてのプランの場合は FreezeAll()。対応する Unfreeze メソッドがあります。

- Freeze メソッドを使用すると、[凍結/アップグレード] としてフラグが設定されたクエリ・プランを [凍結/明示] に昇格 ("凍結") できます。一般に、このメソッドは適切な [凍結/アップグレード] プランを選択的に [凍結/明示的] に昇格してから、それ以外の [凍結/アップグレード] プランの凍結をすべて解除するために使用します。
- Unfreeze メソッドを使用すると、ネームスペース、スキーマ、リレーション (テーブル)、または個々のクエリといった指定した範囲内での [凍結/アップグレード] クエリ・プランを凍結解除できます。

3.3 凍結プランのインタフェース

FREEZE PLANS コマンドを使用してクエリを凍結でき、UNFREEZE PLANS コマンドを使用してクエリを凍結解除できます。どちらの操作も、テーブル別、スキーマ別、またはネームスペース別に個別のクエリに対して実行できます。個別のプランを凍結または凍結解除するには、INFORMATION_SCHEMA.STATEMENTS をクエリすることにより、目的の Statement の Hash を探し出します。つづいて、FREEZE PLANS または UNFREEZE PLANS を使用して、特定の文の Hash を指定することで、その文のプランの状態を変更できます。

Frozen プロパティに対して INFORMATION_SCHEMA.STATEMENTS をクエリすることで、現在のネームスペースにあるすべての SQL 文のプランの状態を一覧できます。Frozen 列の値は、凍結解除 (0)、凍結/明示 (1)、凍結/アップグレー

ド (2)、または未凍結/パラレル (3) のいずれかです。特定のクエリに EXPLAIN を使用して、そのクエリが凍結状態であるかどうかを判断することもできます。

1 つ以上のプランの凍結または凍結解除には、$SYSTEM.SQL.Statement Freeze および Unfreeze メソッドを使用することもできます。次の適切なメソッドを指定して、凍結または凍結解除操作の範囲を指定できます：1 つのプランの場合は FreezeStatement()、関係のすべてのプランの場合は FreezeRelation()、スキーマのすべてのプランの場合は FreezeSchema()、現在のネームスペースのすべてのプランの場合は FreezeAll()。対応する Unfreeze メソッドがあります。

3.3.1 特権

ユーザは、INFORMATION.SCHEMA.STATEMENTS クラス・クエリなど、自身が EXECUTE 特権を持っている SQL 文のみを表示できます。SQL 文へのカタログ・アクセスの場合は、文を実行する特権が付与されているか、%Development リソースに対する "USE" 特権を持っている場合に文を表示できます。

$SYSTEM.SQL.Statement Freeze または Unfreeze メソッド呼び出しの場合は、%Developer リソースに対する "U" 特権を持っている必要があります。

管理ポータルの [SQL 文] にアクセスするには、%Development リソースに対する "USE" 特権が必要です。管理ポータルで SQL 文を表示できるユーザは、その SQL 文を凍結または凍結解除できます。

3.3.2 凍結プランの相違

プランが凍結されている場合は、実際にプランの凍結を解除することなく、プランの凍結を解除したときに異なるプランが生成されるかどうかを判断できます。この情報は、プランの凍結を解除した場合にパフォーマンスが向上するかどうかを判断するために、%NOFPLAN を使用してテストする価値がある SQL 文を決定する際に役立ちます。

INFORMATION.SCHEMA.STATEMENTS FrozenDifferent プロパティを使用して、現在のネームスペースのこのタイプの凍結プランをすべてリスト表示できます。

凍結プランは、以下のいずれかの操作によって現在のプランとは異なるものになる可能性があります。

- テーブル、またはテーブルが参照するテーブルのリコンパイル。
- SetMapSelectability() を使用したインデックスの有効化または無効化。INFORMATION_SCHEMA.INDEXES カタログ・テーブルをクエリして MAP_SELECTABLE 列の値を確認することで、インデックスがアクティブかどうかを確認できます。
- テーブルに対するテーブル・チューニングの実行。
- InterSystems ソフトウェア・バージョンのアップグレード。

リコンパイルによって、既存のクエリ・キャッシュが自動的に削除されます。その他の操作の場合に新規クエリ・プランを適用するには、既存のクエリ・キャッシュを手動で削除する必要があります。

これらの操作によってクエリ・プランが異なるものになるかどうかはわかりません。すべての凍結プランをスキャンして、新しいクエリ・プランが生成されるかどうかを判断できます。

3.3.2.1 凍結プランの自動日次チェック

InterSystems SQL は、[SQL 文] リストにある凍結されたすべての文を毎晩午前 2:00 時に自動的にスキャンします。このスキャンは最大で 1 時間かかります。スキャンが 1 時間で完了しなかった場合は、中断した場所が記録され、そこから次回の日時スキャンが続行されます。

さらに、管理ポータルを使用してスキャンを強制実行できます。そのためには、[システムオペレーション]→[タスクマネージャ]→[タスクスケジュール] の順に選択し、Scan frozen plans タスクを選択します。

このスキャンの結果は、INFORMATION.SCHEMA.STATEMENTS を呼び出すことで確認できます。以下の例では、すべての凍結プランの SQL 文が返され、凍結されない場合になるプランと凍結プランが異なるかどうかが示されます。未凍結プランの場合の文は Frozen=0 または Frozen=3 となる可能性があることに注意してください。

SQL

```
SELECT Frozen,FrozenDifferent,Timestamp,Statement FROM INFORMATION_SCHEMA.STATEMENTS
WHERE Frozen=1 OR Frozen=2
```

3.3.3 エラー状態の凍結プラン

文のプランが凍結されているときに、そのプランに使用している定義に何らかの変更を加えることでプランが無効になると、エラーが発生します。例えば、文のプランに使用していたクラスからインデックスが削除されたとします。

- 文のプランは凍結された状態を維持します。
- [SQL 文の詳細] ページの **[コンパイル設定]** エリアに、[プラン・エラー] フィールドが表示されます。例えば、クエリ・プランで indxdob というインデックス名が使用されていた場合、インデックス indxdob を削除するようにクラス定義を変更すると、次のようなメッセージが表示されます:`Map 'indxdob' not defined in table 'Sample.Mytable', but it was specified in the frozen plan for the query.`
- [SQL 文の詳細] ページの **[クエリプラン]** エリアには、`Plan could not be determined due to an error in the frozen plan` と表示されます。

凍結プランがエラー状態の間にクエリが再実行された場合、InterSystems IRIS は凍結プランを使用しません。代わりに、現在の定義で動作する新しいクエリ・プランが作成されて、そのクエリが実行されます。このクエリ・プランには、前のクエリ・プランと同じクエリ・キャッシュ・クラス名が割り当てられます。

エラー状態のプランは、プランの凍結が解除されるか、プランが有効な状態に戻るように定義を変更するまでエラー状態のままになります。

プランが有効な状態に戻るように定義を変更した場合は、[SQL 文の詳細] ページに移動し、**[エラーのクリア]** ボタンをクリックして、エラーが修正されたかどうかを確認します。修正されている場合は、**[プラン・エラー]** フィールドが表示されなくなります。それ以外の場合は、**[プラン・エラー]** のメッセージが再表示されます。定義が修正されている場合は、明示的にプラン・エラーをクリアしなくても、SQL は凍結プランを使用するようになります。定義が修正されている場合は、**[エラーのクリア]** ボタンにより、[SQL 文の詳細] ページの **[凍結クエリ・プラン]** エリアに、実行プランが再度表示されます。

[プラン・エラー] は、"ソフト・エラー" の可能性があります。これは、プランがインデックスを使用していて、そのインデックスの選択可能性が SetMapSelectability() で 0 に設定されているために、現時点ではクエリ・オプティマイザが選択できない場合に発生することがあります。ほとんどの場合、この原因はインデックスが構築 (再構築) されたことによります。凍結プランがある文に InterSystems IRIS がソフト・エラーを検出すると、クエリ・プロセッサは自動的にエラーをクリアして、凍結プランを使用しようとします。プランがエラー状態のままの場合は、そのプランに再度エラー状態のマークが付けられ、クエリ実行は使用可能な最適なプランを使用するようになります。

3.4 %NOFPLAN キーワード

%NOFPLAN キーワードを使用すると、凍結プランをオーバーライドできます。%NOFPLAN キーワードを含んでいる SQL 文は、新しいクエリ・プランを生成します。凍結プランは保持されますが、使用されません。これにより、凍結プランを維持しながら、生成されたプランの動作をテストできるようになります。

%NOFPLAN の構文は、以下のとおりです。

```
DECLARE <cursor name> CURSOR FOR SELECT %NOFPLAN ...   SELECT %NOFPLAN ....   INSERT
 [OR UPDATE] %NOFPLAN ...   DELETE %NOFPLAN ...   UPDATE %NOFPLAN
```

SELECT 文では、クエリ内で最初の SELECT の直後にのみ %NOFPLAN キーワードを使用できます。これは、UNION クエリの最初の項にのみ使用可能であり、サブクエリでは使用できません。%NOFPLAN キーワードは、SELECT の直後 (DISTINCT や TOP など、その他のキーワードより前) に配置する必要があります。

3.5 凍結プランのエクスポートとインポート

SQL 文を XML 形式のテキスト・ファイルとしてエクスポートまたはインポートすることができます。これにより、凍結プランを別の場所に移動できます。SQL 文のエクスポートとインポートには、関連付けられたクエリ・プランのエンコード・バージョンと、プランが凍結されているかどうかを示すフラグが含まれます。詳細は、"SQL 文のエクスポートとインポート" を参照してください。

4

クエリでの最適化ヒントの指定

既定で、InterSystems SQL クエリ・オプティマイザは、高度で柔軟性の高いアルゴリズムを使用して、複数のインデックスを含む複雑なクエリのパフォーマンスを最適化します。大半の場合、このような既定の設定によって最適なパフォーマンスが得られます。一方、InterSystems SQL には、実行プランの手動変更に使用できるヒントが用意されています。ほとんどの場合、インターシステムズのサポート窓口の助言に従い、これらのヒントを使用して最適なクエリ実行プランを構成します。SQL のパフォーマンスについてサポート窓口の助言を得るには "レポート生成" を参照してください。

このようなヒントをクエリ・オプティマイザに提供して、特定の最適化を採用し、他を除外することを指定するには 2 つの方法があります。その 1 つは SELECT 文の FROM 節にキーワードを使用する方法で、もう 1 つはコマンド・オプションを指定する方法です。

4.1 FROM 節のキーワード

SELECT 文には FROM 節を使用できますが、特定のクエリ最適化動作を指定するキーワードをその節に記述できます。複数のキーワードを、空白で区切って任意の順序で指定できます。各キーワードの詳細は、"クエリ最適化オプション" を参照してください。

指定できるキーワードは以下のとおりです。

- %ALLINDEX – 何らかの利点が得られるすべてのインデックスを、クエリ結合順の先頭にあるテーブル向けに使用することを指定します。
- %FIRSTTABLE – 指定したテーブル名との結合処理をクエリ・オプティマイザで開始することを指定します。
- %FULL – 最大限のアクセス・パフォーマンスが得られるように、すべての代替結合シーケンスをコンパイラ・オプティマイザで検査することを指定します。
- %IGNOREINDEX – 指定したインデックスまたは指定した一覧にあるすべてのインデックスをクエリ・オプティマイザでは無視することを指定します。
- %INORDER – 複数のテーブルを、それを列挙した順序でクエリ・オプティマイザで結合することを指定します。
- %NOFLATTEN – クエリ・オプティマイザでサブクエリを平坦化しないことを指定します。
- %NOMERGE – クエリ・オプティマイザでサブクエリをビューに変換しないことを指定します。
- %NOREDUCE – クエリ・オプティマイザでサブクエリ (またはビュー) をそれを含むクエリとマージしないことを指定します。
- %NOSVSO – クエリ・オプティマイザで集合値サブクエリの最適化 (SVSO) を実行しないことを指定します。
- %NOTOPOPT – TOP 節と ORDER BY 節で使用できます。完了した結果セットを最速で取得できるようにクエリを最適化します。

- %NOUNIONOROPT – 複数の OR 条件向けと UNION クエリに対するサブクエリ向けに用意されている自動最適化を無効にします。
- %PARALLEL – InterSystems IRIS によるクエリの並列処理を提案します。
- %STARTTABLE – 先頭に列挙したテーブルとの結合処理をクエリ・オプティマイザで開始することを指定します。

4.2 コメント・オプション

SELECT、INSERT、UPDATE、DELETE、または TRUNCATE TABLE の各コマンド内で、クエリ・オプティマイザに 1 つ以上のコメント・オプションを指定できます。コメント・オプションでは、クエリ・オプティマイザが SQL クエリのコンパイル時に使用するオプションを指定します。コメント・オプションは、特定のクエリのシステム全体の構成の既定をオーバーライドするのに使用されることが多いです。

4.2.1 構文

構文 `/*#OPTIONS */` (/* と # の間にスペースなし) は、コメント・オプションを指定します。コメント・オプションはコメントではなく、クエリ・オプティマイザに対する値を指定します。コメント・オプションは、JSON 構文 (通常は、`/*#OPTIONS {"optionName":value} */` などの key:value ペア) を使用して指定します。入れ子になった値など、より複雑な JSON 構文がサポートされています。

コメント・オプションはコメントではないため、JSON 構文以外のテキストを含めることはできません。JSON ではないテキストに区切り文字として /* ...*/ を使用すると SQLCODE -153 エラーが発生します。InterSystems SQL では、JSON 文字列のコンテンツは検証されません。

#OPTIONS キーワードは、大文字で指定する必要があります。{ } 括弧で囲まれた JSON 構文内でスペースは使用できません。ダイナミック SQL 文などのように SQL コードを引用符で囲む場合、JSON 構文内の引用符は二重にする必要があります。例えば、`myquery="SELECT Name FROM Sample.MyTest /*#OPTIONS {""optName"":""optValue""} */"` のように指定します。

SQL コード内のコメントを指定できる場所であれば、任意の場所で `/*#OPTIONS */` コメント・オプションを指定できます。表示されている文テキストでは、コメント・オプションは常に、文テキスト末尾のコメントとして表示されます。

SQL コードで、複数の `/*#OPTIONS */` コメント・オプションを指定できます。複数のコメント・オプションを指定した場合、返される文テキストでは、指定した順序でそれらのコメント・オプションが表示されます。同じオプションに対して複数のコメント・オプションを指定した場合、最後に指定したオプションの値が使用されます。

以下のコメント・オプションについては説明があります。

- /*#OPTIONS {"DynamicSQLTypeList":"10,1,11"}
- /*#OPTIONS {"NoTempFile":1} */

4.2.2 表示

`/*#OPTIONS */` コメント・オプションは、SQL コマンドで指定された場所にかかわらず、SQL 文テキストの末尾に表示されます。表示される `/*#OPTIONS */` コメント・オプションの中には、SQL コマンドでは指定されず、コンパイラのプリプロセッサによって生成されるものもあります。例えば、`/*#OPTIONS {"DynamicSQLTypeList": ...}*/` と生成されます。

`/*#OPTIONS */` コメント・オプションは、[プラン表示] の **[ステートメント・テキスト]**、[クエリキャッシュ] の **[クエリ文字列]**、および [SQL 文] の **[ステートメント・テキスト]** に表示されます。

/*#OPTIONS */ コメント・オプションのみが異なる複数のクエリの場合、個別のクエリ・キャッシュが作成されます。