SQL パフォーマンス・オプションの構成

Version 2024.1
2024-06-03

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 2024.1　2024-06-03

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 1
1.1 システム全体のクエリの並列処理 1
1.2 特定のクエリの並列クエリ処理 2
1.2.1 サブクエリ内の %PARALLEL 2
1.3 クエリの並列処理の無視 2
1.4 共有メモリの考慮事項 3
1.5 SQL 文とプランの状態 3

2 実行時プラン選択の使用法 5
2.1 RTPC の適用 5
2.2 RTPC のオーバーライドまたは無効化 6

3 凍結プランの構成 7
3.1 凍結プランの使用方法 7
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 7
3.3 凍結プランのインタフェース 8
3.3.1 特権 9
3.3.2 凍結プランの相違 9
3.3.3 エラー状態の凍結プラン 10
3.4 %NOFPLAN キーワード 10
3.5 凍結プランのエクスポートとインポート 11

4 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 13
4.1 実行時プラン選択 13
4.2 システム規模の自動並列化 13
4.3 自動チューニング 14
4.4 アダプティブ・モードの無効化 14

5 クエリでの最適化ヒントの指定 15
5.1 FROM 節のキーワード 15
5.1.1 %ALLINDEX 15
5.1.2 %FIRSTTABLE 16
5.1.3 %FULL 16
5.1.4 %IGNOREINDEX 16
5.1.5 %INORDER 17
5.1.6 %NOFLATTEN 17
5.1.7 %NOMERGE 18
5.1.8 %NOREDUCE 18
5.1.9 %NOSVSO 18
5.1.10 %NOTOPOPT 18
5.1.11 %NOUNIONOROPT 18
5.1.12 %PARALLEL 19
5.1.13 %STARTTABLE 19
5.2 コメント・オプション 20
5.2.1 構文 20
5.2.2 Cosharding コメント・オプション 21
5.2.3 表示 21

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。

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。InterSystems 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。

- 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。
- ビューを参照し、ビュー ID (%VID) を返すクエリ。
- %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。

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。

1.5 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。
- 文にすべてのパラメータの値が指定されている場合、返されるプランでは RTPC を使用した最適なプランの準備でそれらの値が考慮されます。ただし、プレースホルダとして ? が指定されたパラメータがある場合、パラメータ値を使用せず、テキスト "`Different parameters may use a different plan`" が含まれるプランが返されます。
- 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® 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、既定で既存のクエリ・プランは無効になり、新しいシステムでの文の最初の実行時に新しい最適化されたクエリ・プランが生成されます。この新しいクエリ・プランでは、クエリ・オプティマイザ、実行時プラン選択、およびコード生成の改善といった SQL 処理への機能強化が導入されます。この動作は、新しいインストールの既定であるアダプティブ・モードが有効になっているインスタンスのアップグレード時に適用されます。

ただし、アップグレード後に初めてクエリを実行するときにアダプティブ・モードがオフになっている場合、そのクエリは [凍結/アップグレード] とマークされます。そのクエリの以降の呼び出しでも引き続きこのプランが使用されます。この動作によってクエリのパフォーマンスがアップグレード後も同じであることが保証されるため、SQL 処理の強化機能を活用する機会が失われる可能性があるとしても、高度に制御された環境ではこの動作が望ましい場合があります。このようなアップグレードの後には、すぐに手動でクエリ・プランを凍結解除したり、%NOFPLAN を使用してクエリ・プランのパフォーマンスをテストすることができます。以下のステップは、凍結したクエリ・プランのパフォーマンスと、新しいバージョンの InterSystems IRIS によって生成された新しいクエリ・プランのパフォーマンスを比較する方法を示しています。

1. 凍結したクエリ・プランを実行して、そのパフォーマンスを監視します。
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。このキーワードにより、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。

[凍結/アップグレード] とマークされているクエリ・プランは、"凍結プランのインタフェース" で説明しているいずれかのメソッドを使用して [凍結/明示] に昇格できます。一般にこのアップグレードは、維持したい [凍結/アップグレード] プランを選択的に昇格してから、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。

注釈　ここで説明している動作は、InterSystems 2023.3 以降へのアップグレードに適用されます。これらのバージョンからアップグレードされるインスタンスは、アダプティブ・モードが有効になっています (InterSystems IRIS 2022.2 で導入)。2023.3 より前のリリースへのアップグレードまたは 2022.2 より前のリリースからのアップグレードの場合、アダプティブ・モードはオフと見なされ、前述のようにクエリ・プランの対応する自動凍結が適用されます。

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

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

InterSystems SQL は、アダプティブ・モードでクエリの計画と実行の動作を定義するオプションをいくつか組み込んでいます。アダプティブ・モードは、幅広いユース・ケースにおける導入後、すぐに最高のパフォーマンスが得られる既定の設定です。具体的にいうと、アダプティブ・モードでは、実行時プラン選択 (RTPC) と並列処理を制御し、TUNE TABLE コマンドを自動的に実行してクエリ実行の効率を最適化します。アダプティブ・モードを無効にしない限り、アダプティブ・モードで管理されている各機能を個別に制御することはできません。

4.1 実行時プラン選択

アダプティブ・モードがアクティブであれば RTPC が有効になっています。これにより、クエリで指定した実行時パラメータ値をオプティマイザで活用して最適なプランを判断できます。RTPC の機能の詳細は、"実行時プラン選択の構成" を参照してください。

テーブルのコンテンツに関する正確な情報がある場合にのみ、RTPC が効果的である点に注意が必要です。その結果、RTPC の有効性は、TUNE TABLE を呼び出して収集するテーブル統計に左右されます。

4.2 システム規模の自動並列化

アダプティブ・モードが有効であれば、システム規模の自動並列クエリ処理が有効になっています。したがって、並列処理による利点が得られる可能性があるクエリに並列処理が適用されるように、すべての SELECT クエリには自動的に %PARALLEL によるヒントが指定されます。

ただし、この自動ヒントがあっても、すべてのクエリが並列処理で実行されるわけではありません。SQL オプティマイザによって、クエリが並列処理に適していないと判断され、ヒントが無視されることもあります。アダプティブ・モードを適用していても並列処理が適用されない事例の一覧は、"クエリの並列処理の無視" を参照してください。

自動並列処理のしきい値を構成するパラメータでも、クエリを並列実行するかどうかが判断されます。このしきい値が大きいほど、並列処理が使用される可能性が低くなります。このしきい値は複雑な最適化計算で使用される値ですが、並列クエリ処理に対する適格性がオプティマイザで検討される前にアクセス先のマップに存在する必要があるタプルの最小数としてこの値を捉えることもできます。この値の既定値は 3200 で、最小値は 0 です。
`$SYSTEM.SQL.Util.SetOption("AutoParallelThreshold",n,.oldval)` を使用すると、この設定の値を変

更できます。n は、自動並列処理のしきい値として設定する値です（.oldval は、上書きする値を保持する戻り値を表す変数です）。

シャード環境では、アダプティブ・モードが有効であれば、並列処理しきい値に関係なく、すべてのクエリが並列処理で実行されます。

並列処理によるメモリ使用の詳細は、"並列クエリ処理の構成" の "共有メモリの考慮事項" を参照してください。

4.3 自動チューニング

最も効率的なクエリ・プランがクエリ・オプティマイザで選択されるようにするには、テーブル上のデータを正確に表す最新の統計がそのテーブルに存在する必要があります。TUNE TABLE コマンドを使用した効率的なサンプリング、またはテーブル定義にハードコーディングした統計処理によって、このような統計を収集できます。一般的に、新しいテーブルには、TUNE TABLE を呼び出していない限り、このような統計が存在しません。統計がないテーブルでは、そのテーブルに発行される最初のクエリを実行する前に高速なブロック サンプリングの実行が適切であれば、アダプティブ・モードを使用することで必ず TUNE TABLE が自動的に実行されます。この機能を使用すれば、テーブル統計の収集によってクエリに得られるパフォーマンス上の大きな利点を見逃すことがなくなります。

TUNE TABLE で収集される統計のいくつかに関する詳細は、"クエリ・オプティマイザで使用するテーブル統計" を参照してください。

注釈　一般的なデータをテーブルにロードした後やそのコンテンツを大幅に変更した後でも、TUNE TABLE の実行には意味があります。自動チューニングが動作するのは、テーブルに対するクエリが初めて実行されたときの 1 回のみです。一般的なデータがテーブルに入力される前に、このような状況が発生することもあり得ます。

4.4 アダプティブ・モードの無効化

アダプティブ・モードの無効化が必要になる状況は多くありませんが、**[システム管理]→[構成]→[SQL とオブジェクトの設定]→[SQL]** に移動し、[アダプティブモードをオフにして実行時プラン選択と自動チューニングを無効にする] チェック・ボックスにチェックを入れることで、アダプティブ・モードを無効にすることができます。

アダプティブ・モードを無効にしている場合は、アプリケーションのニーズに基づいて実行時プラン選択と並列処理を手動で構成すること、およびデータのディストリビューションを大幅に変更したときは手動で TUNE TABLE を実行することをお勧めします。

5
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。

5.1 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。

5.1.1 %ALLINDEX

このオプションのキーワードを指定すると、何らかの利点をもたらすインデックスはすべて、クエリ結合順の最初のテーブルに使用されます。このキーワードは、定義された複数のインデックスがある場合にのみ使用する必要があります。オプティマイザの既定では、オプティマイザが最も有用であると判断したインデックスのみが使用されます。既定では、これには、有用なすべての等価インデックス、および選択したその他のタイプのインデックスが含まれます。%ALLINDEX では、あらゆるタイプの有用と思われるインデックスが使用されます。すべてのインデックスをテストするとオーバーヘッドが増加しますが、状況によっては、既定の最適化よりもパフォーマンスが向上する場合があります。このオプションは、複数の値域条件インデックスおよび非効率的な等価条件インデックスを使用する場合に、特に有用です。このような状況では、正確なインデックスの選択性がクエリ・オプティマイザで使用できない可能性があります。%ALLINDEX は、%IGNOREINDEX と併用することで、特定のインデックスの組み込みと除外を行うことができます。通常、%ALLINDEX は TOP 節クエリと共に使用しないでください。

%STARTTABLE を %ALLINDEX と組み合わせて使用すると、%ALLINDEX が適用されるテーブルを指定できます。

特定の条件で条件レベル・ヒント %NOINDEX を使用して、%ALLINDEX に対する例外を指定することができます。%NOINDEX ヒントは、インデックスを使用しない各クエリ選択条件の前に配置します。例えば、`WHERE %NOINDEX hiredate < ?` のようにします。通常、この方法は条件によって圧倒的多数のデータが除外されない場合に使用します。「より小さい (<)」または「より大きい (>)」条件では、通常、条件レベル・ヒント %NOINDEX を使用すると効果的です。等価条件では、条件レベル・ヒント %NOINDEX を使用しても効果はありません。結合条件では、%NOINDEX は、ON 節

の結合でサポートされます。詳細は、"SQL 最適化ガイド" の "クエリ処理でのインデックスの使用" を参照してください。

5.1.2 %FIRSTTABLE

```
%FIRSTTABLE tablename
```

このオプション・キーワードを指定すると、クエリ・オプティマイザは指定された tablename から結合を開始します。tablename は、結合シーケンスで後に指定されるテーブルに名前を付けます。残りのテーブルの結合順序は、クエリ・オプティマイザに委任されます。このヒントは機能的には %STARTTABLE と同じですが、結合テーブル・シーケンスを任意の順序で指定できます。

tablename は、テーブル・エイリアスか未修飾のテーブル名のどちらかの単純な識別子でなくてはなりません。修飾されたテーブル名 (schema.table) は使用できません。クエリでテーブル・エイリアスを指定する場合は、テーブル・エイリアスを tablename として使用する必要があります。以下に例を示します。

```
FROM %FIRSTTABLE P Sample.Employee AS E JOIN Sample.Person AS P ON E.Name = P.Name
```

%FIRSTTABLE と %STARTTABLE の両方とも、結合操作に使用する最初のテーブルを指定できます。%INORDER を使用すると、結合操作に使用するすべてのテーブルの順序を指定できます。これらの 3 つのキーワードは相互排他的です。1 つだけ指定してください。これらのキーワードが使用されない場合、クエリ・オプティマイザは、テーブルがリストされている順番に関係なく、最適と考えられる順序でテーブルの結合を実行します。

%FIRSTTABLE または %STARTTABLE を使用して、LEFT OUTER JOIN の右側 (または RIGHT OUTER JOIN の左側) で結合順序を開始することはできません。これを実行しようとすると、SQLCODE -34 エラーである "オプティマイザが使用可能な結合順序を見つけることができませんでした"" 生成されます。

詳細は、%STARTTABLE クエリ最適化オプションを参照してください。

5.1.3 %FULL

このオプション・キーワードにより、コンパイラ・オプティマイザは、すべての代替結合シーケンスを検査し、アクセス・パフォーマンスを最大限に引き出します。例えば、ストアド・プロシージャを生成する場合、コンパイル時間が長くなると最適なアクセス・パフォーマンスを引き出す場合があります。FROM 節にテーブル名が多数ある場合、既定では可能性の小さな結合シーケンスは検証しないように最適化されます。%FULL は、この既定の動作をオーバーライドします。

矢印構文でアクセスされるテーブルが FROM 節にある場合は、%INORDER キーワードと %FULL キーワードの両方を指定できます。それにより、テーブルの順序に制約がなくなります。

5.1.4 %IGNOREINDEX

このオプション・キーワードを指定すると、クエリ・オプティマイザは指定されたインデックス、またはインデックスのリストを無視します (同義の %IGNOREINDICES は非推奨ですが、下位互換性保持のためにサポートされます)。

このキーワードに続いて、1 つ以上のインデックス名を指定します。複数のインデックス名はコンマで区切る必要があります。無視されるインデックスは、以下のような方法で指定できます。

- %IGNOREINDEX * ー FROM 節に指定されたテーブルに対するクエリによって使用されるすべてのインデックスが無視されます。
- %IGNOREINDEX SchemaName.* ー SchemaName 内のすべてのインデックスが無視されます。
- %IGNOREINDEX SchemaName.TableName.* ー 特定のテーブル内のすべてのインデックスが無視されます。
- %IGNOREINDEX SchemaName.TableNameIndexName ー テーブル内の特定のインデックスが無視されます。

以下のクエリ例では、クエリで使用されるはずのすべてのインデックスが無視されます。

SQL

```
SELECT ID FROM %IGNOREINDEX * Opus.Fact WHERE Date > '2023-01-22'
```

この最適化制約を使用すると、特定のクエリに対して最適ではないインデックスを使用しないよう、クエリ・オプティマイザに指定することができます。1 つを除くすべてのインデックス名を指定することで、事実上、クエリ・オプティマイザで残りのインデックスを使用するよう強制できます。

また、最初に %NOINDEX キーワードで条件を記述することで、特定の条件式で特定のインデックスを無視することができます。詳細は、″SQL 最適化ガイド″ の “クエリ処理でのインデックスの使用” を参照してください。

5.1.5 %INORDER

このオプション・キーワードを指定すると、クエリ・オプティマイザは FROM 節にリストされるテーブルの順番で結合を実行します。コンパイル時間を最小限にします。矢印構文で参照されるテーブルの結合順には制限はありません（矢印構文の使用法の詳細は、″InterSystems SQL の使用法″ の ″暗黙結合″ を参照してください）。サブクエリを平坦化しても、インデックスを使用しても、影響を受けることはありません。

%INORDER は、CROSS JOIN または RIGHT OUTER JOIN で使用することはできません。指定されたテーブル順が外部結合に必要な順番と一致しない場合、SQLCODE -34 エラーである ″オプティマイザが使用可能な結合順序を見つけることができませんでした″ が生成されます。これを避けるために、外部結合で使用する場合は、%INORDER のみを ANSI 形式の左外部結合または FULL 外部結合で使用することをお勧めします。

シャード・テーブルに対してクエリを実行するときに %INORDER を使用することはできません。″スケーラビリティ・ガイド″ の “シャーディングによるデータ量に応じた InterSystems IRIS の水平方向の拡張” の章にある ″シャード・クラスタにおけるクエリ″ を参照してください。

ビューとテーブル・サブクエリは、FROM 節で指定された順番で処理されます。

- Streamed View：%INORDER は、ビュー内のテーブル処理順に影響を与えません。
- Merged View：%INORDER は、ビューの参照時にビューのテーブルをビューの FROM 節順に処理します。

このキーワードと %FIRSTTABLE および %STARTTABLE を比べた場合、2 つは完全な結合順序でなく最初の結合テーブルのみを指定する点が異なります。さまざまな結合順序の最適化によるテーブルのマージの動作については、″%STARTTABLE″ を参照してください。

%INORDER の最適化と %PARALLEL 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。

5.1.6 %NOFLATTEN

このオプションのキーワードは、ブーリアン値を返すサブクエリである修飾付きサブクエリの FROM 節で指定します。これにより、コンパイラ・オプティマイザでサブクエリの平坦化を抑制する必要があることを指定します。この最適化オプションは、サブクエリをクエリに効果的に統合して修飾付きサブクエリを含むクエリを最適化する “平坦化”（既定）を無効にします。この平坦化では、サブクエリのテーブルをクエリの FROM 節に追加し、サブクエリの条件をクエリの WHERE 節の結合または制限に変換します。

以下に、%NOFLATTEN を使用する修飾付きサブクエリの例を示します。

SQL

```
SELECT Name,Home_Zip FROM Sample.Person WHERE Home_Zip IN
      (SELECT Office_Zip FROM %NOFLATTEN Sample.Employee)
```

SQL

```
SELECT Name,(SELECT Name FROM Sample.Company WHERE EXISTS
            (SELECT * FROM %NOFLATTEN Sample.Company WHERE Revenue > 500000000))
 FROM Sample.Person
```

%INORDER および %STARTTABLE の最適化では、暗黙的に %NOFLATTEN が指定されます。

5.1.7 %NOMERGE

このオプションのキーワードは、サブクエリの FROM 節で指定します。これにより、コンパイラ・オプティマイザではサブクエリをビューに変換しないことを指定します。この最適化オプションは、サブクエリを含むクエリの最適化を無効にします。この最適化では、サブクエリをインライン・ビューとしてクエリの FROM 節に追加し、クエリのフィールドに対するサブクエリの比較が、結合としてクエリの WHERE 節に移動します。

5.1.8 %NOREDUCE

このオプションのキーワードは、ストリーム化されたサブクエリの FROM 節で指定します。ここのサブクエリとは、行の結果セットを返すサブクエリ、つまりクエリに含まれる FROM 節内のサブクエリです。これは、コンパイラ・オプティマイザではサブクエリ (またはビュー) とそれを含むクエリとの結合を抑制する必要があるということを指定します。

以下の例では、クエリ・オプティマイザは、通常、Sample.Person とサブクエリとのデカルト積結合を実行することで、このクエリを “削減” します。%NOREDUCE 最適化オプションは、これを防止します。InterSystems IRIS は、代わりに gname に一時インデックスを構築し、この一時インデックスに結合を実行します。

SQL

```
SELECT * FROM Sample.Person AS p,
   (SELECT Name||'goo' AS gname FROM %NOREDUCE Sample.Employee) AS e
    WHERE p.name||'goo' = e.gname
```

5.1.9 %NOSVSO

このオプションのキーワードは、ブーリアン値を返すサブクエリである修飾付きサブクエリの FROM 節で指定します。これにより、コンパイラ・オプティマイザで集合値サブクエリの最適化 (SVSO) を抑制する必要があることを指定します。

ほとんどの場合、集合値サブクエリの最適化によって [NOT] EXISTS および [NOT] IN サブクエリのパフォーマンスが向上し、特に、1 つだけの分離可能な相関条件が指定されたサブクエリのパフォーマンスが向上します。SVSO はこのために、その条件を満たすデータ値を一時インデックスに移入します。そのサブクエリを繰り返し実行する代わりに、InterSystems IRIS はこれらの値を一時インデックス内で検索します。例えば、SVSO は、`P.num` の一時インデックスを作成することで、`NOT EXISTS (SELECT P.num FROM Products P WHERE S.num=P.num AND P.color='Pink')` を最適化します。

SVSO は、ALL または ANY キーワードが比較演算子 (>、>=、<、または <=) およびサブクエリと共に使用されているサブクエリ (例 : `...WHERE S.num > ALL (SELECT P.num ...)`) を最適化します。これを行うには、サブクエリ式 sqbExpr (この例では `P.num`) を、適宜 MIN(sqbExpr) または MAX(sqbExpr) で置換します。これにより、sqbExpr に対するインデックスが存在する場合に高速の計算が実現されます。

%INORDER および %STARTTABLE の最適化では、集合値サブクエリの最適化が抑制されません。

5.1.10 %NOTOPOPT

このオプションのキーワードは、TOP 節を ORDER BY 節と共に使用する場合に指定します。既定では、TOP を ORDER BY と共に使用すると、最初の行に移動する時間が最速になるように最適化されます。%NOTOPOPT (TOP 最適化でない) を指定することでクエリが最適化され、完全な結果セットを最速で取得できるようになります。

5.1.11 %NOUNIONOROPT

このオプションのキーワードは、クエリまたはサブクエリの FROM 節で指定します。これにより、複数の OR 条件および UNION クエリ式に対するサブクエリに実行される自動最適化が無効化されます。これらの自動最適化では、適切な場

合には、複数の OR 条件が UNION サブクエリに変換されたり、UNION サブクエリが OR 条件に変換されたりします。これらの UNION/OR 変換により、EXISTS やその他の下位の述語を InterSystems IRIS クエリ・オプティマイザ・インデックスで使用できる最上位の条件に移行できます。これらの既定の変換は、ほとんどの状況に適しています。

ただし、状況によってはこれらの UNION/OR 変換によって大幅なオーバーヘッドが生じることがあります。%NOUNIONOROPT を指定すると、この FROM 節に関連付けられた WHERE 節内のすべての条件に対して自動の UNION/OR 変換が無効になります。そのため、複雑なクエリでは、この自動 UNION/OR 最適化を 1 つのサブクエリにのみ無効にしてその他のサブクエリには有効にすることができます。

UNION %PARALLEL キーワードは、自動の UNION から OR への最適化を無効にします。

%INORDER および %STARTTABLE の最適化では、OR から UNION への最適化が抑制されますが、UNION から OR への最適化は抑制されません。

5.1.12 %PARALLEL

このオプションのキーワードは、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。

個々のフィールドと集約関数の両方を指定しているが GROUP BY 節を含まないクエリは、並列処理を実行できません。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 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 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。

注釈　%PARALLEL を指定するクエリは、読み取り専用ではなく、読み取り/書き込み可能なデータベースで実行する必要があります。そうでないと、<PROTECT> エラーが発生する可能性があります。

FROM 節で %PARALLEL キーワードを指定していても、並列処理ではなく線形処理を使用するクエリがあります。特に列指向テーブルでは、DISTINCT、ORDER BY、TOP、UNION を使用するクエリは並列処理されません。また、最適化すると並列処理の利点を得られなくなるクエリもあります。並列処理のために InterSystems IRIS によってクエリが分割されたか、分割された場合はどのように分割されたかを、プラン表示を使用して確認できます。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。

詳細は、"SQL 最適化ガイド" の "並列クエリ処理の構成" を参照してください。

5.1.13 %STARTTABLE

このオプション・キーワードを指定すると、クエリ・オプティマイザは FROM 節にリストされた最初のテーブルから結合を開始します。残りのテーブルの結合順序は、クエリ・オプティマイザに委任されます。このキーワードと %INORDER を比べた場合、%INORDER では完全な結合順序を指定する点が異なります。

%STARTTABLE は、CROSS JOIN または RIGHT OUTER JOIN で使用することはできません。%STARTTABLE (または %FIRSTTABLE) を使用して、LEFT OUTER JOIN の右側 (または RIGHT OUTER JOIN の左側) で結合順序を開始することはできません。指定された先頭テーブルが外部結合に必要なテーブルと一致しない場合、SQLCODE -34 エラーである "オプティマイザが使用可能な結合順序を見つけることができませんでした" が生成されます。これを避けるために、外部結合で使用する場合は、%STARTTABLE のみを ANSI 形式の左外部結合または FULL 外部結合で使用することをお勧めします。

以下の表に、%INORDER および %STARTTABLE の最適化によりスーパークエリの親とインライン・ビューを組み合わせる場合のマージの動作を示します。

	結合オプティマイザを指定しないスーパークエリ	%STARTTABLE を指定したスーパークエリ	%INORDER を指定したスーパークエリ
結合オプティマイザを指定しないビュー	可能な場合にビューをマージする。	ビューがスーパークエリの先頭の場合、マージしない。 それ以外は、可能な場合にビューをマージする。	可能な場合にマージする。ビューの基本テーブルの順序は不規則になります。
%STARTTABLE を指定したビュー	マージしない	ビューがスーパークエリの先頭の場合、可能であればマージする。ビューの先頭テーブルはスーパークエリの先頭テーブルになります。 それ以外はマージしない。	マージしない
%INORDER を指定したビュー	マージしない	マージしない	ビューが %INORDER で制御されない場合、マージしない。 それ以外は、可能な場合にマージする。ビューの順序は、スーパークエリの結合順序に置き換えられます。

%FIRSTTABLE ヒントは、機能的には %STARTTABLE と同じですが、任意の順序で結合テーブル・シーケンスを柔軟に指定できます。

5.2 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。

5.2.1 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、`/*#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 {"NoTempFile":1} */ : 既定では、モジュールは処理を実行し、その結果を内部一時ファイル (内部一時テーブル) に入力します。NoTempFile コメント・オプションに 1 を指定することにより、内部一時ファイルを生成しないクエリ・プランを作成するようクエリ・オプティマイザに強制することもできます。

5.2.2 Cosharding コメント・オプション

SQL クエリで複数のシャード・テーブルが指定された場合、SQL プリプロセッサは Cosharding コメント・オプションを生成します。これは、クエリ・キャッシュ・テキストの末尾に追加されます。この Cosharding オプションは、指定されたテーブルがコシャードされているかどうかを示します。

注釈　Cosharding オプションは、自動的に適用されます。ユーザがこのオプションを手動で指定することはできません。

以下の例では、指定された 3 つのテーブルがすべてコシャードされています。

```
/*#OPTIONS {"Cosharding":[["T1","T2","T3"]]} */
```

以下の例では、指定された 3 つのテーブルがいずれもコシャードされていません。

```
/*#OPTIONS {"Cosharding":[["T1"],["T2"],["T3"]]} */
```

以下の例では、テーブル T1 はコシャードされていませんが、テーブル T2 および T3 はコシャードされています。

```
/*#OPTIONS {"Cosharding":[["T1"],["T2","T3"]]} */
```

5.2.3 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。

`/*#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。