記事
Toshihiko Minamoto · 2023年2月8日 7m read

# 2020.1 の新機能: ユニバーサルクエリキャッシュ

InterSystems IRIS 2020.1
には、重要なアプリケーションの構築を支援する新機能と機能改善が多数盛り込まれています。 2019.1 から 2020.1 までに行われた多数の大幅なパフォーマンス改善のほかに、最近の SQL
の歴史において最も大きな変更点の 1 つであるユニバーサルクエリキャッシュ（UQC）が導入されています。
この記事では、SQL
ベースのアプリケーションに対するそのインパクトについて、技術的な観点で詳しく説明しています。

## UQC とは？

スクラブルで頭字語として使用されればゲームチェンジャーとなる可能性もあるでしょうが、ユニバーサルクエリキャッシュは、あらゆる種類の SQL ステートメントを単一のキャッシュで管理することで、ボード全体の SQL 操作を単純化することを目的としています。 これまでは、動的 SQL（%SQL.Statement
インフラストラクチャを使用）や JDBC 、 ODBC にて発行されるステートメントは、Prepare 時に
キャッシュドクエリとして最適化されたコードにコンパイルされていました。 一方、埋め込み SQL
を使用する場合（&SQL( )
構文を使用）、クエリを実行するコードはインラインで生成され、アプリケーションの一部となっていました。
このため、埋め込み SQL は、IRIS がサポートする他のすべての形態の SQL
クエリとは非常に異なって処理されていました。

キャッシュドクエリクラスは、ステートメントを発行したコードから独立しており、新しいインデックスの追加やアップグレードの後に、更新されたテーブルの統計に基づいてより効率的なアクセスパスを得るなどのために、必要に応じてパージしたり生成し直したりすることが
可能です（アップグレード時に凍結されたクエリプランの注意事項もご覧ください）。 一方、埋め込み SQL
は静的であり、新しいテーブル統計とインデックスを自動的に利用することができませんでした。

2020.1 では、ユニバーサルクエリキャッシュの導入により、埋め込み SQL にもキャッシュ済みクエリクラスを生成できるようになり、このコードをアプリケーションロジックを保持するクラスに混在させることがなくなりました。 最も重要なメリットは、埋め込み SQL では、より新しいテーブル統計または新しいインデックスを利用するために、ソースクラスを再コンパイルする必要がなくなったことです。 UQC で参照されているテーブルが更新されると、新しいキャッシュ済みクエリクラスが自動的に生成されるため、生成可能な最も効率の良いコードが確実に実行されます。 埋め込み SQL をクラスに生成すると、他のすべてのクエリが既に使用しているより高速なカーソル変数ストレージを利用できます。
つまり、膨大な作業を行うクエリが以前よりも速く実行されるということです。 また、UQC
によって、クラスのコンパイル順の管理において管理者を悩ませることの多かったクラス間の SQL
依存関係がすべて取り除かれます。

これによって作業が大きく変わります。多大な労力をつぎ込んで、「想像力に溢れる」セットアップに対応するために大規模なテストを行ってはいるものの（デプロイ済みのコード、アプリケーションコード内でのネームスペースの切り替えなど）、その想像を超えるセットアップがまだまだ存在するかもしれません。 そのため、埋め込み SQL を大量に（そして創造性豊かに）使用しているアプリケーションを 2020.1
にアップグレードする際には、特に注意してください。また、通常と異なることに気づいた場合は、WRC
までご連絡ください。

## 埋め込み SQL と動的 SQL は 1 つに統一されたのですか？

動的 SQL と埋め込み SQL の使用方法については、現在でもセマンティックがある程度異なっています。
埋め込み SQL は名前付きカーソルを操作しますが、これは動的 SQL
のクエリに対する実際のオブジェクトハンドルを操作するよりも少し扱いにくいものです。 ただし、これは個人

の好みに大きく左右される問題であるため、明らかに、楽しむためだけに、あるフレーバーから別のフレーバーにアプリケーションを書き直す理由はありません。

パフォーマンスについては、埋め込み SQL には動的 SQL よりも高速だという評判がありましたが、ここ数年におけるコード生成とオブジェクト処理の改善により、その差はほとんどなくなっています。 とは言え、埋め込み SQL のカーソルの値は変数に直接取得することが可能であるため、動的 SQL
の相当するオブジェクトプロパティを通じて値にアクセスするよりも、わずかに高速ではあります。 したがって、確かにより高速ではありますが、その差はほんのわずかであるため、多くの場合、（主観的に！）開発の利便性を上回ることはありません。

## 但し書き

UQC の導入により、2020.1 の埋め込み SQL に、同じルーチンまたはクラスのコンテキストにとどまるのではなく、キャッシュ済みクエリオブジェクト内の別のクラスメソッドをバックグラウンドで呼び出してクエリを実行するという、以前にはなかった非常に小さくも固定されたオーバーヘッドが存在するようになりました。
キャッシュ済みのクエリコードの生成とコンパイルにおける作業は、以前は、&SQL( )
コンストラクトを含むクラスまたはルーチンのコンパイルの一環でしたが、現在は、動的 SQL と同様に、またより現実的なテーブル統計を利用して、クエリが初めて呼び出されるときに行われるようになっています。
開発者は、新しい `/compileembedded=1` コンパイルフラグを使って埋め込み SQL
を強制的にコンパイルできます。

これとは別に、生成されたキャッシュ済みのクエリコードで高速な i%var アクセスを使用してカーソルの状態情報を保持するのには様々なメリットがあり、ほとんどの重要なクエリで固定オーバーヘッドが相殺され、実際に埋め込み SQL が以前のバージョンよりも高速になります。 パフォーマンスが低下する可能性があるのは、2019.4 に比べれば、ほんの少数のステートメント（非常に単純な INSERT など）です。
独自に行った実験では、そのような単純なクエリでは約 6%、複雑なクエリでは 3%
のパフォーマンスの*増加*が見られました。

一方で、2019.4
より前のバージョンからアップグレー
ドするユーザーは、パフォーマンスの*改善*以外は、何も感じられない可能性があります。 2019.1 から 2019.4 までは、SQL とカーネルレイヤーにおいて、UQC
で導入される小さな固定オーバーヘッドをはるかに上回る他のパフォーマンス改善が多数行われているためです。
これらの多数の変更点の概要については、こちらの GS セッション録画をご覧ください。

## まとめ

簡単に言えば、ユニバーサルクエリキャッシュによって、最新の統計と利用可能なインデックスを考慮するようにクエリプランが素早く確実に更新されるため、DBA の業務がより楽になります。 特に、SQL
アプリケーションを別のお客様環境にデプロイする場合には、一番ピッタリな
SQLを開発者が自由に選択できる重要な機能強化と言えます。

*この記事は、@Mark.Hansonと@ Tom.Woodfin からいただいた貴重な貢献に基づいています*

#SQL #InterSystems IRIS

**ソースURL:**
https://jp.community.intersystems.com/post/20201-%E3%81%AE%E6%96%B0%E6%A9%9F%E8%83%BD-%E3%83%A6%E3%83%8B%E3%83%90%E3%83%BC%E3%82%B5%E3%83%AB%E3%82%AF%E3%82%A8%E3%83%AA%E3%82%AD%E3%83%A3%E3%83%83%E3%82%B7%E3%83%A5