InterSystems IRIS による SQL 文の処理方法

Version 2023.1
2024-01-02

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

InterSystems IRIS による SQL 文の処理方法
InterSystems IRIS Data Platform　Version 2023.1　2024-01-02

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

目次

InterSystems IRIS による SQL 文の処理方法

ユーザが送信したすべての SQL 文について、InterSystems IRIS® では、できる限り迅速かつ効率的にその文を実行するために、いくつかの処理手順を実行します。これらの手順により、ユーザはパフォーマンスを最大化するための構造化方法を気にすることなく文を記述することができます。このトピックでは、事前解析から実行まで、インターシステムズが SQL 文を処理する方法について説明します。特に、クエリ文と、以下を含む DML (Data Manipulation Language) の一部となる文に焦点を当てて説明します。

- データを照会し、結果を取得する SELECT 文
- データを変更する INSERT、UPDATE、および DELETE 文

テーブル・スキーマを変更または定義する、CREATE TABLE や ALTER TABLE などのデータ定義言語 (DDL) 文は対象外です。

以下で説明するすべての手順は InterSystems IRIS で自動的に実行され、ユーザおよびアプリケーション・コードに対して透過的です。

1 文の事前解析

最初に文を送信すると、InterSystems IRIS がプリパーサを介してこれを実行し、正規化します。これらの正規化手順には、以下のものが含まれます。

- 関係のない空白の削除。例えば、クエリ

  SQL

  ```
  SELECT    Name     FROM     Customer
  ```

  は、以下のようになります。

  SQL

  ```
  SELECT Name FROM Customer
  ```

- SQL 標準の一部であるキーワードの大文字化。例えば、クエリ

  SQL

  ```
  Select Name From Customer
  ```

  は、以下のようになります。

  SQL

  ```
  SELECT Name FROM Customer
  ```

- リテラル置換の実行。ここでは、入力パラメータとして文に渡されたリテラル値が疑問符に置換され、別々に格納されます。例えば、クエリ

  SQL

  ```
  SELECT Name FROM Customer WHERE Zip = ‘00001’
  ```

は、以下のようになります。

SQL

```
SELECT Name FROM Customer WHERE Zip = ?
```

この正規化されたクエリは SQL サーバに渡され、文の正規化された形式と使用される SQL 言語などの環境変数に基づいて、SQL エンジンにより一意のハッシュ値が生成されます。このハッシュ値を使用して、SQL エンジンは、この文がすでにユニバーサル・クエリ・キャッシュ (UQC) に格納されているかどうか検索できます。

文がキャッシュ内にある場合、SQL エンジンはそのハッシュを使用して、正規化されたクエリを実行する ObjectScript コードを検索します。その後、その疑問符をリテラル置換の際に格納されたリテラル値に置換し、クエリを実行します。残りの文準備手順はスキップされます。これにより、文の再実行、および渡されるパラメータのみが異なる文の実行を高速化することができます。

例えば、Customers テーブルに対して初めてクエリを実行するとします。

SQL

```
SELECT Name FROM Customer WHERE Zip = '00001'
```

SQL エンジンはこのクエリの正規化された形式をキャッシュに格納します。

SQL

```
SELECT Name FROM Customer WHERE Zip = ?
```

このクエリが前のクエリと正規化された形式が同じ場合、これを実行すると、SQL エンジンは渡された入力パラメータのみを変更して格納されたクエリを実行します。

SQL

```
SELECT Name FROM Customer WHERE Zip = '00002'
```

文がキャッシュにない場合、SQL エンジンは正規化されたクエリをキャッシュに格納し、文を SQL コンパイラに渡して、文の準備を続行します。

JDBCまたはODBCクライアントを使用している場合、事前解析はクライアント側で行われ、サーバから一部の作業がオフロードされます。一部の特定のケースでは、この際にクライアント側のキャッシュを利用できるため、さらにサーバの負荷が軽減されます。

2 文のコンパイル

プリパーサから正規化された文を受け取ると、SQL コンパイラはこの文のフル解析を行います。これらの解析手順には、以下のものが含まれます。

- 文が構文的に正しく、SQL 標準に準拠していることの確認。
- この文が関与するテーブルの確認。これらのテーブルについて、この文は以下のことを行います。
  - SQL サーバに格納されているデータ・ディクショナリから、これらのテーブルのメタデータを取得します。メタデータには、テーブル内の行数など、文の生成に使用されるプランに提供される情報が含まれます。詳細は "クエリ・プランの生成" セクションを参照してください。
  - 文で要求されている情報がテーブル内のデータと一致していることを確認します。例えば、文がテーブルに存在しない列のデータを要求している場合、コンパイラはエラーを発行します。

その後、SQL コンパイラは文を SQL オプティマイザに渡し、ここで文を実行するための最適化されたプランが生成されます。

SQL コンパイラは、SQL オプティマイザで使用するため、オプション・フラグを正規化された文に追加することもできます。これらは一般的に、`/*#OPTIONS */` タグで囲まれ、ユーザは無視することができます。

3 クエリ・プランの生成

クエリ・プランは、SQL オプティマイザが文の操作を実行するのに使用する正式な戦略です。オプティマイザはさまざまなプランを生成し、それぞれの実行コストを見積もり、コストが最も低いプランを選択します。

クエリのコストを決定するために、InterSystems SQL はテーブル統計を利用します。InterSystems SQL は、TUNE TABLE ユーティリティを実行して統計を収集します。大規模なデータ・セットでは、TUNE TABLE はサンプリングを使用し、すべての行を調べるわけではありません。標準ストレージ・レイアウトのテーブルのサンプリングは、データベース・ブロックの低レベルのスキャンに基づいており、ギガバイト単位のデータを持つテーブルであっても、数秒で正確な統計を生成できます。ブロック・スキャンを使用できるテーブルでは、初めてテーブルが照会され、以前の統計が見つからない場合、InterSystems SQL は自動的に統計を収集します。

例えば、特定の郵便番号の地域に住むすべての顧客を返したいとします。考えられるクエリ・プランの 1 つは、テーブルをフル・スキャンするというものです。すべての行を調べて、郵便番号列の値が指定した値である行を保持します。このプランのコストは簡単に見積もることができます。行の総数に、ストレージから単一行を取得するのにかかる時間を掛けます。行の総数はテーブル統計に含まれており、メタデータから取得されるデータの一部です。

`ZipCode` 列にインデックスがある場合、別のクエリ・プランでこのインデックスを利用し、より低コストのプランを生成することもできます。選択性は、選択するクエリ・プランの主要な決定要因です。選択性とは、指定された 1 つの値に一致するテーブルの総数の割合です。例えば、1,000,000 行のテーブルで、`ZipCode` 列の選択性が 1% であるとします。クエリ・プランがインデックスを使用して、特定の値を持つすべての行を取得する場合、インデックスは平均して 1,000,000 * 1% = 10,000 行を返します。`ZipCode` インデックスの例において、そのインデックスを使用するクエリ・プランのコストは、選択性 * 行の総数 *（単一のインデックス・エントリを読み取るためのコスト + 単一の行を読み取るためのコスト）となります。

2 つの異なる列にフィルタ条件を満たすために使用できるインデックスがある場合、最初に選択性が低い方の列が選択されます。これは、この列の方が一致する行を迅速に絞り込むので、よりコストの低いプランが得られるためです。

場合によっては、最適なクエリ・プランは、実行時にどのパラメータが渡されるかによって異なります。例えば、米国全体の小売業者の Customer テーブルについて考えます。次のクエリを使用して、ワイオミング州のすべての顧客を選択するとします。

SQL

```
SELECT Name FROM Customer WHERE State = 'WY'
```

50 州のうち 1 州を選択すると、データの 2% を選択することになるため、`State` 列には既定として 2% の選択性が設定されています。`State` 列にインデックスがある場合、選択性はこのように低いため、このインデックスを使用するクエリ・プランは有効である可能性があります。

ワイオミング州のローカルな店舗について、別の `Customer` テーブルを考えてみましょう。先ほどと同じクエリを実行するとします。

SQL

```
SELECT Name FROM Customer WHERE State = 'WY'
```

この特定のデータセットでは、データが列の値全体に均等に分散されていないため、このインデックスのクエリ・プランはコストに見合わない可能性があります。この選択性は、ワイオミング州では行の 90% 以上と、非常に高いためです。では、クエリが別の州に対するものであったらどうでしょうか。

SQL

```
SELECT Name FROM Customer WHERE State = 'CO'
```

ワイオミング州以外の州の場合、選択性はもっと低くなります。州によっては、この値は 0.1% 未満になる場合があります。InterSystems SQL では、実行時プラン選択 (RTPC) を使用し、渡されたパラメータに基づいて適切なクエリ・プランを選択します。RTPC はシステム全体の設定で、既定でオンになっています。RTPC がパラメータに基づいてクエリ・プランを選択する方法の 1 つを以下に示します。

- パラメータ値の選択性が高い (つまり、特定の列に多く出現する) 場合、インデックスは使用しません。
- パラメータ値の選択性が低い場合、インデックスを使用します。

オプティマイザがその場で独自のインデックスを作成し、それらを使用するプランを生成して、さらにクエリを最適化することさえあり得ます。

その後、後続のコード生成に必要なすべてが含まれた最善のクエリ・プランが、文インデックスと呼ばれるすべての文のレジストリに格納されます。文インデックスには以下のものが含まれます。

- 選択したクエリ・プラン
- クエリ・キャッシュの場所
- 生成された実行コード

文インデックスには、実行回数や平均実行時間など、文の実行時統計も含まれます。これは毎日、毎時間記録されるため、データの過去ビューを表示し、クエリがどのように実行されているかを経時的に確認することができます。管理ポータルには、クエリのすべてのメタデータと実行時統計が表示されます。

指定されたクエリに対し、特定のプランの使用を固定するには、凍結プランを指定できます。クエリ・キャッシュに凍結プランが存在する場合、その文の事前解析後はクエリ・プランの生成がスキップされ、InterSystems SQL ではその凍結プランが使用されます。ユーザは、プランが凍結されているかどうかを、文インデックスから確認することができます。

4 文実行コードの生成

クエリ・プランが生成されると、InterSystems IRIS はそのプランとテーブルのストレージ定義 (テーブル・データとインデックス・データの物理的な場所) を確認し、文を実行する ObjectScript コードを生成します。このコードは、以下を持つ `Query` クラスです。

- 事前解析手順で格納されたパラメータ値を使用してこのコードをインスタンス化する手段。
- `Next` メソッド。これは、クエリによって返される結果セットのように、データの行を次々と取得するために使用されます。

クエリ・コードが生成されると、InterSystems SQL は文を実行し、結果を返します。