記事
Toshihiko Minamoto · 2021年3月22日 5m read

Outlier Selectivity (外れ値の選択性) について

Caché 2013.1 より、InterSystems
は特殊な値を持つフィールドが使われるクエリプランのセレクションを改善する目的で Outlier Selectivity
(外れ値の選択性) を導入しました。

この記事では、「Project」テーブルを例に使い、Outlier Selectivity の概要やそれが SQL
のパフォーマンスを向上させる仕組み、またクエリを書く際の注意点などについて解説したいと思います。

Selectivity (選択性)

まずは、Selectivity についてさっと説明します。 Selectivity とは、テーブル内の 1
つの列の中にある値に関するメタ情報のことです。 データが典型的なかたちで分布されていると想定した場合、「このテーブル内のこの列に特定の値を持つすべての行を要求するとしたら、通常取得できるのはテーブル内のどの程度の割合であろうか？」という疑問の答えとなる情報です。

Owener と Status という 2 つのフィールドを持つ「Project」という架空のテーブルについて考えます。 Owner
にはプロジェクトを担当する従業員が入り、Status には PREP、OPEN、REVIEW、COMPLETE という 4
つのオプションの 1 つが入ります。 Tune Table を実行すると、クラスのストレージに Selectivity
の値があるのが確認できます。

```
<Property name="Owner">
    <Selectivity>3.3726%</Selcetivity>
</Property>
<Property name="Status">
    <Selectivity>25.000%</Selcetivity>
</Property>
```

では、次の 2 つのクエリについて考えます。

```
SELECT * FROM Projects WHERE Owner = ?

SELECT * FROM Projects WHERE Status = ?
```

1 つ目のクエリが返すプロジェクトの割合は、平均で「Project」テーブルにあるすべてのプロジェクトの 3%
をわずかに超える程度です。 2 つ目のクエリの平均は 25% です。 こうしたテーブルが JOIN や複数の WHERE
条件を伴うクエリで使用されるとなれば、3% と 25% では実行時間に大きな差が生じるほか、Caché
が実行するクエリプランも変更される可能性があります。

Outlier Selectivity (外れ値の選択性)

Selectivity を見ればすべてが分かるという訳ではありません！
フィールド内の潜在的な値は特殊なかたちで分布される場合があります。 Outlier Selectivity
を使用することで、特殊な値、すなわち、外れ値を 1 つ持つフィールドを賢く取り扱うことができます。

「Project」テーブルでは、プロジェクトのステータスは先ほどふれた 4 つのうちの 1 つになりますが、数年ほど経てば COMPLETE のプロジェクトの数が他のステータスのプロジェクトよりも大分多くなります。

先ほども言いましたが、次のクエリは*平均*で「Project」テーブルの 25% を返します。

```
SELECT * FROM Projects WHERE Status = ?
```

ですが、もっと細かく推測できるはずです！ もし、WHERE 節が「WHERE Status = 'COMPLETE'」であれば、テーブルのほぼすべてを取得できますが、 「WHERE Status = 'PREP'」だと、取得できる割合はごくわずかです。

保管する前の WHERE:

```
<Property name="Status">
    <Selectivity>0.25</Selectivity>
</Property>
```

Outlier Selectivity の導入により、以下を格納できます。

```
<Property name="Status">
    <OutlierSelectivity>0.9:"COMPLETE"</OutlierSelectivity>
    <Selectivity>0.03333</Selectivity>
</Property>
```

これで、以下の 2 つのクエリを区別することができます。

```
SELECT * FROM Projects WHERE Status = 'COMPLETE'

SELECT * FROM Projects WHERE Status = 'PREP'
```

1 つ目のクエリはテーブル内にあるすべてのプロジェクトの 90% を返し、2 つ目はわずか 3% しか返さないと推測できます。

複数のテーブルや複数のインデックスから選べる選択肢があるクエリの場合、90% と 3% ではパフォーマンスに大きな差が生じるほか、この場合も SQL エンジンが選択するクエリプランが変更される可能性があります。

Outlier Selectivity を使ったクエリ

Outlier Selectivity には間違いなくメリットがあり、アプリケーションに変更を加える必要もありません。しかし、フル活用するには注意すべき点がいくつかあります。 デフォルトで、Caché は同じ形式が使われたすべてのクエリに対し、クエリプランを 1 つだけ生成します。 (先ほどの WHERE Status = 'COMPLETE' や WHERE Status = 'PREP' など)

デフォルトで、Caché は、クエリのパラメーターの値は外れ値ではないと想定します。
クエリに強制的に外れ値を考慮させるには、丸かっこを二重にして、外れ値のリテラル置換を抑制します。

```
SELECT * FROM Projects WHERE Status = (('COMPLETE'))

SELECT * FROM Projects WHERE Status = 'PREP'
```

丸かっこを二重にすると、SQL
エンジンがクエリ内のパラメーターの特定の値に対してプランを生成することを強制できます。 これで Caché は、プロジェクトの 90% が取得されるときと、3% が取得されるときが分かるため、このクエリに対して 2 種類のプランを使うことができます。

また、BiasQueriesAsOutlier の値を 1 か 0 に設定すれば、Caché
がデフォルトで外れ値以外の値を想定するかどうかも制御できます。 以下を実行すると、Caché は、外れ値を使用するクエリは稀なクエリではないと想定します。

```
<Property name="Status">
    <BiasQueriesAsOutlier>1</BiasQueriesAsOutlier>
    <OutlierSelectivity>0.9:"COMPLETE"</OutlierSelectivity>
    <Selectivity>0.03333</Selectivity>
</Property>
```

以上の例は、Outlier Selectivity の概要、およびそれがクエリのパフォーマンスを向上させる仕組みについて理解する手掛かりとしてお役に立ちましたでしょうか？ この情報の別のプレゼン資料や SQL
の他の統計に関する詳細は、Selectivity と Outlier Selectivity と題した DocBook 文書をご覧ください。

#SQL #Caché

ソースURL:https://jp.community.intersystems.com/post/outlier-selectivity-%E5%A4%96%E3%82%8C%E5%80%A4%E3%81%AE%E9%81%B8%E6%8A%9E%E6%80%A7-%E3%81%AB%E3%81%A4%E3%81%84%E3%81%A6