記事
Toshihiko Minamoto · 2022年12月8日 5m read

InterSystems IRIS における一意のインデックスと null 値

一意のインデックスにまつわる興味深いパターンが最近持ちあがったので（isc.rest
に関する内部ディスカッション）、コミュニティ向けに強調したいと思います。

動機付けのユースケースとして: ツリーを表すクラスがあるとします。各ノードには名前があるため、名前と親ノードでノードを一意にしたいと考えています。 各ルートノードにも一意の名前を持たせます。
この場合の自然な実装は以下のようになります。

```
Class DC.Demo.Node Extends %Persistent
{

Property Parent As DC.Demo.Node;
Property Name As %String [ Required ];
Index ParentAndName On (Parent, Name) [ Unique ];
Storage Default
{
<Data name="NodeDefaultData">
<Value name="1">
<Value>%%CLASSNAME</Value>
</Value>
<Value name="2">
<Value>Parent
</Value>
<Value name="3">
<Value>Name
</Value>
</Data>
<DataLocation>^DC.Demo.NodeD</DataLocation>
<DefaultData>NodeDefaultData</DefaultData>
<IdLocation>^DC.Demo.NodeD</IdLocation>
<IndexLocation>^DC.Demo.NodeI</IndexLocation>
<StreamLocation>^DC.Demo.NodeS</StreamLocation>
<Type>%Storage.Persistent</Type>
}

}
```

以上です！

ただし、落とし穴があります。現状のこの実装では、複数のルートノードに同じ名前を付けることができます。
なぜでしょうか？ Parent は必須プロパティでない（また、そうすべきではない）ため、**IRIS は null を一意のインデックスの個別の値として処理しない**からです。 一部のデータベース（SQL Server
など）はそうしていますが、SQL
標準では、それらが誤っていると述べています（出典が必要です。このことについては StackOverflow のどこかで見かけましたが、それは出典になりません。以下にある、これについてとインデックスと制約の違いについての @Dan Pasco のコメントをご覧ください）。

これを回避する方法は、参照されたプロパティが null である場合に、非 null
値に設定される計算プロパティを定義し、そのプロパティに一意のインデックスを設定することです。

以下に、例を示します。

```
Property Parent As DC.Demo.Node;
Property Name As %String [ Required ];
Property ParentOrNUL As %String
 [ Calculated, Required, SqlComputeCode =
{Set {*} = $Case({Parent},"":$c(0),:{Parent})}, SqlComputed ];
Index ParentAndName On (ParentOrNUL, Name) [ Unique ];
```

これにより、$c(0) を ParentAndNameOpen/Delete/Exists
に渡して、親（存在しない）と名前でルートノードを一意に識別することもできます。

この動作が非常に役立つ動機付けの例として、https://github.com/intersystems/isc-rest/blob/main/cls/pkg/isc/rest/resourceMap.cls をご覧ください。 多くの行は、2
つのフィールド（DispatchOrResourceClass と
ResourceName）に対して同じセットの値を持つことができますが、最大でもその内 1 つを「デフォルト」として扱いたいと考えており、一意のインデックスはこれを行う上で最適です。「デフォルト」フラグを 1 または null に設定できると言う場合は、フラグと他の 2 つのフィールドに一意のインデックスを付けることができます。

@Dan Pasco さんのコメント

Nullや一意制約について、私の意見と、可能性のあるいくつかの事実を記載します。

IRISユニークインデックス - これは単なるインデックスだけでなく、インデックスキーに一意性制約を定義する構文的なショートカットです。多くのSQLの実装は、この2つの概念を融合せず、SQL標準はインデックスを定義していません。標準SQLは一意制約を定義しています。IDKEYとPRIMARYKEYの両方が一意制約の修飾子であることに留意してください（そして、我々の世界では、IDKEYとして定義されたインデックスもまた特別なものである）。IDKEYとしてフラグが立てられたインデックスとPRIMARYKEYとしてフラグが立てられたインデックスは、最大で1つずつ存在することができます。インデックスはPRIMARYKEYとIDKEYの両方になることができます。

かつて、"ユニークインデックス "と "ユニーク制約 "の構文を異なる規則で定義したSQL実装がありました。両者の違いは単純で、インデックスが完全に入力されていない場合（テーブルのすべての行がインデックスに表示されていない場合、これを条件付きインデックスと呼びます）、一意インデックスはインデックスで表される行の一意性をチェックするだけです。一意制約はすべての行に適用されます。

また、インデックスは、クエリのサブセットのパフォーマンスを向上させるという、唯一の目的のために存在することに留意してください。どのようなSQL制約もクエリとして表現することができます。

標準SQLは、NULLの動作に関して、少し一貫性がありません。フレームワークのドキュメントでは、次のように定義されています。

一意制約とは、テーブルの1つまたは複数の列を一意列として指定するものです。一意制約は、テーブル内の2つの行が一意列の同じ非NULL値を持たない場合にのみ満たされます。

基本文書では、F291とF292という2つのオプション機能があります。これらの機能は、ユニークな述語(291)とユニークなNULL処理(292)を定義しています。これらの機能は、ユーザがNULLの "区別性 "を定義できる構文を提供するように見えます。どちらもオプションの機能で、比較的最近(2003年?2008年?)のものです。これらの機能がサポートされていない場合のルールは、実装者に委ねられています。

IRISは、フレームワーク文書ステートメントと一致しており、すべての制約は、NULLでないキーにのみ適用されます。「NULL」キーは、すべてのキーカラム値がNULLであるキーと認識されます。

#SQL #インデックス付け #Caché #InterSystems IRIS

ソースURL:https://jp.community.intersystems.com/post/intersystems-iris-%E3%81%AB%E3%81%8A%E3%81%91%E3%82%8B%E4%B8%80%E6%84%8F%E3%81%AE%E3%82%A4%E3%83%B3%E3%83%87%E3%83%83%E3%82%AF%E3%82%B9%E3%81%A8-null-%E5%80%A4