Article
Timothy Leavitt · Jun 28, 2022 2m read

Unique indices and null values in InterSystems IRIS

An interesting pattern around unique indices came up recently (in internal discussion re: isc.rest) and I'd like to highlight it for the community.

As a motivating use case: suppose you have a class representing a tree, where each node also has a name, and we want nodes to be unique by name and parent node. We want each root node to have a unique name too. A natural implementation would be:

```
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>
<Value name="3">
<Value>Name</Value>
</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>
}

}
```

And there we go!

But there's a catch: as it stands, this implementation allows multiple root nodes to have the same name. Why? Because Parent is not (and shouldn't be) a required property, and **IRIS does not treat null as a distinct value in unique indices**. Some databases (e.g., SQL Server) do, but the SQL standard says they're wrong [citation needed; I saw this on StackOverflow somewhere but that doesn't really count - see also @Dan Pasco 's comment below on this and the distinction between indices and constraints].

The way to get around this is to define a calculated property that's set to a non-null value if the referenced property is null, then put the unique index on that property. For example:

```
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 ];
```

This also allows you to pass $c(0) to ParentAndNameOpen/Delete/Exists to identify a root node uniquely by parent (there isn't one) and name.

As a motivating example where this behavior is very helpful, see https://github.com/intersystems/isc-rest/blob/main/cls/pkg/isc/rest/resourceMap.cls . Many rows can have the same set of values for two fields (DispatchOrResourceClass and ResourceName), but we want at most one of them to treated as the "default", and a unique index works perfectly to enforce this if we say the "default" flag can be set to either 1 or null then put a unique index on it and the two other fields.

#Indexing #SQL #InterSystems IRIS

Source URL:https://community.intersystems.com/post/unique-indices-and-null-values-intersystems-iris