Article
Eduard Lebedyuk · Aug 3, 2020 3m read

Creating classes/tables with more than 999 properties in InterSystems IRIS

InterSystems IRIS currently limits classes to 999 properties.

But what to do if you need to store more data per object?

This article would answer this question (with the additional cameo of Community Python Gateway and how you can transfer wide datasets into Python).

The answer is very simple actually - InterSystems IRIS currently limits classes to 999 properties, but not to 999 *primitives*. The property in InterSystems IRIS can be an object with 999 properties and so on - the limit can be easily disregarded.

Approach 1.

Store 100 properties per serial property. First create a stored class that stores a hundred properties.

```
Class Test.Serial Extends %SerialObject
{

Property col0;
...
Property col99;
}
```

And in your main class add as much properties as you need:

```
Class Test.Record Extends %Persistent
{
Property col00 As Test.Serial;

Property col01 As Test.Serial;
...

Property col63 As Test.Serial;
}
```

This immediately raises your limit to 99900 properties.

This approach offers uniform access for all properties via SQL and object layers (we always know property reference by it's number).

Approach 2.

One $lb property.

```
Class Test.Record Extends %Persistent
{
Property col As %List;
}
```

This approach is simpler but does not provide explicit column names.

Use SQL $LIST* Functions to access list elements.

Approach 3.

Use Collection (List Of/Array Of) property.

```
Class Test.Record Extends %Persistent
{
Property col As List Of %Integer;
}
```

This approach also does not provide explicit column names for individual values (but do you really need it?). Use property parameters to project the property as SQL column/table.

Docs for collection properties.

Approach 4.

Do not create properties at all and expose them via SQL Stored procedure/%DispatchGetProperty.

```
Class Test.Record Extends %Persistent
{

Parameter GLVN = {..GLVN("Test.Record")};

/// SELECT Test_Record.col(ID, 123)
/// FROM Test.Record
///
/// w ##class(Test.Record).col(1, )
ClassMethod col(id, num) As %Decimal [ SqlProc ]
{
    #define GLVN(%class) ##Expression(##class(Test.Record).GLVN(%class))
    quit $lg($$$GLVN("Test.Record")(id), num + 1)
}

/// Refer to properties as: obj.col123
Method %DispatchGetProperty(Property As %String) [ CodeMode = expression ]
{
..col(..%Id(), $e(Property, 4, *))
}


/// Get data global
/// w ##class(Test.Record).GLVN("Test.Record")
ClassMethod GLVN(class As %Dictionary.CacheClassname = {$classname()}) As %String
{
    return:'$$$comClassDefined(class) ""
    set strategy = $$$comClassKeyGet(class, $$$cCLASSstoragestrategy)
    return $$$defMemberKeyGet(class, $$$cCLASSstorage, strategy, $$$cSDEFdatalocation
)
}
```

The trick here is to store everything in the main $lb and use unallocated schema storage spaces to store your data. Here's an article on global storage.

With this approach, you can also easily transfer the data into Python environment with Community Python Gateway via the ExecuteGlobal method.

This is also the fastest way to import CSV files due to the similarity of the structures.

Conclusion

999 property limit can be easily extended in InterSystems IRIS.

Do you know other approaches to storing wide datasets? If so, please share them!

#Globals #Object Data Model #Python #Relational Tables #SQL #Tips & Tricks #InterSystems IRIS

Source URL:https://community.intersystems.com/post/creating-classestables-more-999-properties-intersystems-iris