文章
Nicky Zhu · 一月 11, 2021 阅读大约需 3 分钟

类、表和Globals——工作原理

当我向技术人员介绍InterSystems IRIS时，我一般会先讲其核心是一个多模型DBMS。

我认为这是其主要优势（在DBMS方面）。数据仅存储一次。您只需访问您想用的API。
- 您想要数据的概要？用SQL！
- 您想用一份记录做更多事情？用对象！
- 想要访问或设置一个值，并且您知道键？用Globals！

乍一看挺好的，简明扼要，又传达了信息，但当人们真正开始使用InterSystems IRIS时，问题就来了。类、表和Globals是如何关联的？它们之间有什么关系？数据是如何存储的？

本文我将尝试回答这些问题，并解释这些到底是怎么回事。

第一部分 模型偏见

处理数据的人往往对他们使用的模型有偏见。

开发者们把数据视为对象。对他们而言，数据库和表都是通过CRUD（增查改删，最好是基于ORM）交互的盒子，但底层的概念模型都是对象（当然这对于我们大多数使用面向对象编程语言的开发者来说没错）。

而DBA大部分时间都在搞关系型DBMS，他们把数据视为表。对象只是行的封装器。

对于InterSystems IRIS，持久类也是一个表，将数据存储在Global中，因此需要进行一些澄清。

第二部分 举例

假设您创建了类Point：

```
Class try.Point Extends %Persistent [DDLAllowed]
{
    Property X;
    Property Y;
}
```

您也可以用DDL/SQL创建相同的类：

```
CREATE Table try.Point (
    X VARCHAR(50),
    Y VARCHAR(50))
```

编译后，新类将自动生成一个存储结构，将原生存储在Global中的数据映射到列（对于面向对象开发者而言，是属性）：

```
Storage Default
{
<Data name="PointDefaultData">
    <Value name="1">
        <Value>%%CLASSNAME</Value>
    </Value>
    <Value name="2">
        <Value>X</Value>
    </Value>
    <Value name="3">
        <Value>Y</Value>
    </Value>
</Data>
<DataLocation>^try.PointD</DataLocation>
<DefaultData>PointDefaultData</DefaultData>
<IdLocation>^try.PointD</IdLocation>
<IndexLocation>^try.PointI</IndexLocation>
<StreamLocation>^try.PointS</StreamLocation>
<Type>%Library.CacheStorage</Type>
}
```

这是怎么回事？

自下向上（加粗文字很重要）：

- Type: 生成的存储类型，本例中是持久对象的默认存储
- StreamLocation - 存储流的Global
- IndexLocation - 索引Global
- IdLocation - 存储ID自增计数器的Global
- **DefaultData** - 存储将Global值映射到列/属性的XML元素
- **DataLocation** - 存储数据的Global

现在我们的DefaultData是PointDefaultData，让我们分析下它的结构。本质上Global节点有这样的结构：

- 1 - %%CLASSNAME
- 2 - X
- 3 - Y

所以我们可能期望我们的Global是这样的：

```
^try.PointD(id) = %%CLASSNAME, X, Y
```

但如果我们输出 Global 它会是空的，因为我们没有添加任何数据：

```
zw ^try.PointD
```

让我们添加一个对象：

```
set p = ##class(try.Point).%New()
set p.X = 1
set p.Y = 2
```

```
write p.%Save()
```

现在我们的Global变成了这样

```
zw ^try.PointD
^try.PointD=1
^try.PointD(1)=$lb("",1,2)
```

可以看到，我们期望的结构%%CLASSNAME, X, Y是用 $lb("",1,2)
设置的，它对应的是对象的X和Y属性（%%CLASSNAME 是系统属性，忽略)。

我们也可以用SQL添加一行：

```
INSERT INTO try.Point (X, Y) VALUES (3,4)
```

现在我们的Global变成了这样：

```
zw ^try.PointD
^try.PointD=2
^try.PointD(1)=$lb("",1,2)
^try.PointD(2)=$lb("",3,4)
```

所以我们通过对象或SQL添加的数据根据存储定义被存储在Global中（备注：可以通过在PointDefaultData
中替换X和Y来手动修改存储定义，看看新数据会怎样！）。

现在，如果我们想执行SQL查询会怎样？

```
SELECT * FROM try.Point
```

这段sql查询被转换为ObjectScript代码， 遍历^try.PointD，并根据存储定义（其中的 PointDefaultData
部分）填充列。

下面是修改。让我们从表中删除所有数据：

```
DELETE FROM try.Point
```

看看我们的Global变成什么样了：

```
zw ^try.PointD
^try.PointD=2
```

可以看到，只剩下ID计数器，所以新对象/行的ID=3。我们的类和表也继续存在。

但如果我们运行会怎样：

```
DROP TABLE try.Point
```

它会销毁表、类并删除Global。

```
zw ^try.PointD
```

看完这个例子，希望您现在对Global、类和表如何相互集成和互补有了更好的理解。根据实际需要选用正确的API会让开发更快、更敏捷、bug更少。

#对象数据模型 #SQL #Globals #关系表 #新手 #InterSystems IRIS

源URL: https://cn.community.intersystems.com/post/%E7%B1%BB%E3%80%81%E8%A1%A8%E5%92%8Cglobals%E2%80%94%E2%80%94%E5%B7%A5%E4%BD%9C%E5%8E%9F%E7%90%86