文章
姚 鑫 · 四月 14, 2021 阅读大约需 8 分钟

第二章 定义和构建索引（二）

第二章 定义和构建索引（一）

定义索引

使用带有索引的Unique、PrimaryKey和IdKey关键字

与典型的SQL一样，InterSystems IRIS支持惟一键和主键的概念。
InterSystems IRIS还能够定义IdKey，它是类实例(表中的行)的唯一记录ID。
这些特性是通过Unique、PrimaryKey和IdKey关键字实现的:
- Unique -在索引的属性列表中列出的属性上定义一个唯一的约束。
也就是说，只有这个属性(字段)的唯一数据值可以被索引。
唯一性是根据属性的排序来确定的。
例如，**如果属性排序是精确的，则字母大小写不同的值是唯一的;**
如果属性排序是SQLUPPER，则字母大小写不同的值不是唯一的。
但是，请注意，对于未定义的属性，不会检查索引的惟一性。
根据SQL标准，未定义的属性总是被视为唯一的。
- PrimaryKey -在索引的属性列表中列出的属性上定义一个主键约束。
- IdKey -定义一个唯一的约束，并指定哪些属性用于定义实例(行)的唯一标识。
IdKey总是具有精确的排序规则，即使是数据类型为string时也是如此。

这些关键字的语法出现在下面的例子中:

```
Class MyApp.SampleTable Extends %Persistent [DdlAllowed]
{
  Property Prop1 As %String;
  Property Prop2 As %String;
  Property Prop3 As %String;

  Index Prop1IDX on Prop1 [ Unique ];
  Index Prop2IDX on Prop2 [ PrimaryKey ];
  Index Prop3IDX on Prop3 [ IdKey ];
}
```

注意:IdKey、PrimaryKey和Unique关键字只在标准索引下有效。
不能将它们与位图或位片索引一起使用。

同时指定IdKey和PrimaryKey关键字也是有效的语法，例如:

```
 Index IDPKIDX on Prop4 [ IdKey, PrimaryKey ];
```

这个语法指定IDPKIDX索引既是类(表)的IdKey，也是它的主键。
这些关键字的所有其他组合都是多余的。

对于使用这些关键字之一定义的任何索引，都有一个方法允许打开类的实例，其中与索引关联的属性有特定的值;

定义SQL搜索索引

可以在表类定义中定义SQL搜索索引，如下所示:

```
Class Sample.TextBooks Extends %Persistent [DdlAllowed]
{
 Property BookName As %String;
 Property SampleText As %String(MAXLEN=5000);

 Index NameIDX On BookName [ IdKey ];
 Index SQLSrchIDXB On (SampleText) As %iFind.Index.Basic;
 Index SQLSrchIDXS On (SampleText) As %iFind.Index.Semantic;
 Index SQLSrchIDXA On (SampleText) As %iFind.Index.Analytic;
}
```

用索引存储数据

可以使用index Data关键字指定一个或多个数据值的副本存储在一个索引中:

```
Class Sample.Person Extends %Persistent [DdlAllowed]
{
 Property Name As %String;
 Property SSN As %String(MAXLEN=20);

 Index NameIDX On Name [Data = Name];
}
```

在本例中，索引NameIDX的下标是各种Name值的排序(大写)值。名称的实际值的副本存储在索引中。当通过SQL更改Sample.Person表或通过对象更改对应的Sample.Person类或其实例时，将维护这些副本。

在经常执行选择性(从许多行中选择一些行)或有序搜索(从许多列中返回一些列)的情况下，在索引中维护数据副本会很有帮助。

例如，考虑以下针对Sample.Person表的查询：

SQL引擎可以通过读取NameIDX而从不读取表的主数据来决定完全满足此请求。

注意：不能使用位图索引存储数据值。

索引null

如果一个索引字段的数据为NULL(没有数据存在)，相应的索引使用索引NULL标记来表示这个值。
默认情况下，索引空标记值为-1E14。
使用索引空标记可以使空值排序在所有非空值之前。

%Library.BigInt数据类型存储小于-1E14的小负数。默认情况下，%BigInt索引空标记值为-1E14，因此与现有BigInt索引兼容。如果索引的%BigInt数据值可能包括这些极小的负数，则可以使用INDEXNULLMARKER属性参数更改特定字段的索引NULL标记值，作为特性定义的一部分，如下例所示：

```
Property ExtremeNums As %Library.BigInt(INDEXNULLMARKER = "-1E19");
```

还可以在数据类型类定义中更改索引NULL标记的默认值。

此参数属性在IRIS里有，Cache里没有。

索引集合

为属性编制索引时，放在索引中的值是整个已整理属性值。对于集合，可以通过将(Elements)或(Key)附加到属性名称来定义与集合的元素和键值相对应的索引属性。(元素)和(键)允许指定从单个属性值生成多个值，并对每个子值进行索引。当属性是集合时，Elements令牌通过值引用集合的元素，Key令牌通过位置引用它们。当元素和键都出现在单个索引定义中时，索引键值包括键和关联的元素值。

例如，假设有一个基于Sample.Person类的FavoriteColors属性的索引。对此属性集合中的项进行索引的最简单形式是以下任一种：

```
INDEX fcIDX1 ON (FavoriteColors(ELEMENTS));
```

或

```
INDEX fcIDX2 ON (FavoriteColors(KEYS));
```

其中，FavoriteColor(Elements)是指FavoriteColors属性的元素，因为它是一个集合。一般形式是PropertyName(元素)或PropertyName(键)，其中该集合的内容是定义为某个数据类型的列表或数组的属性中包含的一组元素)。

若要索引文本属性，可以创建一个由PropertyNameBuildValueArray()方法生成的索引值数组(在下一节中介绍)。与集合本身一样，(Elements)和(Key)语法对索引值数组有效。

如果属性集合被投影为数组，则索引必须遵守以下限制才能被投影到集合表。索引必须包括(键)。索引不能引用集合本身和对象ID值以外的任何属性。如果投影索引还定义了要存储在索引中的数据，则存储的数据属性也必须限制为集合和ID。否则，不会投影索引。此限制适用于投影为数组的集合属性上的索引；不适用于投影为列表的集合上的索引。

与集合的元素或键值对应的索引还可以具有所有标准索引功能，例如将数据与索引一起存储、特定于索引的排序规则等。

InterSystems SQL可以通过指定FOR SOME%ELEMENT谓词来使用集合索引。

使用(Elements)和(Key)索引数据类型属性

为了索引数据类型属性，还可以使用BuildValueArray()方法创建索引值数组。此方法将属性值解析为键和元素的数组；它通过生成从与其关联的属性的值派生的元素值集合来实现这一点。使用BuildValueArray()创建索引值数组时，其结构适合索引。

BuildValueArray()方法的名称为PropertyNameBuildValueArray()，其签名为：

```
ClassMethod propertynameBuildValueArray(value, ByRef valueArray As %Library.String) As %Status
```

- BuildValueArray()方法的名称以组合方法的典型方式派生于属性名。
- 第一个参数是属性值。
- 第二个参数是通过引用传递的数组。
  这是一个包含键-元素对的数组，键下标的数组等于元素。
- 该方法返回一%Status 值。

这个例子:

```
/// DescriptiveWords?????????????
Property DescriptiveWords As %String;

/// ????????
Index dwIDX On DescriptiveWords(ELEMENTS);

/// ??????:???????????????
///
/// (??DescriptiveWords???????????????)
ClassMethod DescriptiveWordsBuildValueArray(
     Words As %Library.String = "",
     ByRef wordArray As %Library.String)
   As %Status {
 If Words '= "" {
   For tPointer = 1:1:$Length(Words,",") {
     Set tWord = $Piece(Words,",",tPointer)
     If tWord '= "" {
       Set wordArray(tPointer) = tWord
     }
   }
 }
 Else {
   Set wordArray("TODO") = "Enter keywords for this person"
 }
 Quit $$$OK
}
```

在本例中，dwIDX索引基于DescriptiveWords属性。
DescriptiveWordsBuildValueArray()方法接受由Words参数指定的值，基于该值创建一个索引值数组，并将其存储在wordArray中。
InterSystems IRIS在内部使用BuildValueArray()实现;
不调用此方法。

注意:没有必要将任何元素/键值建立在属性值的基础上。
唯一的建议是，每次向该方法传递给定值时，都创建相同的元素和键数组。

为各种实例的描述性词所属性设置值和检查这些值的属性涉及活动（如以下）：

```
SAMPLES>SET empsalesoref = ##class(MyApp.Salesperson).%OpenId(3)

SAMPLES>SET empsalesoref.DescriptiveWords = "Creative"

SAMPLES>WRITE empsalesoref.%Save()
1
SAMPLES>SET empsalesoref = ##class(MyApp.Salesperson).%OpenId(4)

SAMPLES>SET empsalesoref.DescriptiveWords = "Logical,Tall"

SAMPLES>WRITE empsalesoref.%Save()
1
```

这 sample index内容，例如：
DescriptiveWords(ELEMENTS)　　ID　　Data

```
" CREATIVE"                    3                              ""
" ENTER KEYWORDS FOR THIS      1                              ""
PERSON"
" ENTER KEYWORDS FOR THIS      2                              ""
PERSON"
" LOGICAL"                     4                              ""
" TALL"                        4                              ""
```

注意：此表显示抽象中的索引内容。磁盘上的实际存储形式可能会有所变化。

将数组(元素)上的索引投影到子表

要在嵌入式对象中索引属性，需要在引用该嵌入式对象的持久化类中创建索引。
属性名必须指定表(%Persistent类)中的引用字段的名称和嵌入对象(%SerialObject)中的属性的名称，如下面的示例所示:

```
Class Sample.Person Extends (%Persistent) [ DdlAllowed ]
{  Property Name As %String(MAXLEN=50);
   Property Home As Sample.Address;
   Index StateInx On Home.State;
}
```

此处Home是Sample.Person中引用嵌入对象Sample.Address的属性，该对象包含State属性，如下例所示：

```
Class Sample.Address Extends (%SerialObject)
{  Property Street As %String;
   Property City As %String;
   Property State As %String;
   Property PostalCode As %String;
 }
```

**只有与持久类
属性引用相关联的嵌入对象
的实例中的数据值被索引。不能直接索引%SerialObject
属性**。%Library.SerialObject(以及%SerialObject的所有未显式定义SqlCategory的子类)的SqlCategory为字符串。

还可以使用SQL CREATE INDEX语句在嵌入式对象属性上定义索引，如下例所示：

```
CREATE INDEX StateIdx ON TABLE Sample.Person (Home_State)
```

类中定义的索引注释

当在类定义中使用索引时，需要记住以下几点:

- **索引定义仅从主(第一个)超类继承。**
- **如果使用Studio添加(或删除)数据库中存储数据的类的索引定义，则必须使用 “ 构建索引 ” 中描述的过程之一来手动填充索引。**

使用DDL定义索引

如果你使用DDL语句来定义表，也可以使用以下DDL命令来创建和删除索引:

- CREATE INDEX
- DROP INDEX

DDL index命令执行以下操作:

1. 它们更新在其上添加或删除索引的相应类和表定义。
   重新编译修改后的类定义。
2. 它们根据需要在数据库中添加或删除索引数据:CREATE
   index命令使用当前存储在数据库中的数据填充索引。
   类似地，DROP INDEX命令从数据库中删除索引数据(即实际索引)。

#SQL #Caché #InterSystems IRIS #InterSystems IRIS for Health

源
URL:
https://cn.community.intersystems.com/post/%E7%AC%AC%E4%BA%8C%E7%AB%A0-%E5%AE%9A%E4%B9%89%E5%92%8C%E6%9E%84%E5%BB%BA%E7%B4%A2%E5%BC%95%EF%BC%88%E4%BA%8C%EF%BC%89