文章
Louis Lu · 五月 30, 2021 阅读大约需 7 分钟

如何保存、查询 List 类型数据

本文主要总结了在InterSystems IRIS 中如何保存、查询List类型数据

假设我们设计的对象中包含姓名，同时每个姓名下可以包含多个电话。我们可以使用下面方法进行处理。

1. 传统方式

我们可以把每一个姓名和电话放在不同列中。

```
Class Test.Person Extends %Persistent
{

Property Name As %String;

Property Phone As %String;

}
```

我们使用SQL语句插入数据：

```
insert into Test.Person values ('a','111-111-1111');

insert into Test.Person values ('b','222-111-1111');

insert into Test.Person values ('a','111-222-1111');

insert into Test.Person values ('c','333-111-1111');

insert into Test.Person values ('b','222-222-1111');
```

数据在表中是这样的：

Name	Phone
a	111-111-1111
b	222-111-1111
a	111-222-1111
c	333-111-1111
b	222-222-1111

这种情况下，我们可以使用下面的sql语句将结果返回：

```
SELECT
   distinct %exact(Name) Name,
   LIST(phone %foreach(Name)) Phonestr
FROM test.person
```

Name	Phonestr
a	111-111-1111,111-222-1111
b	222-111-1111,222-222-1111
c	333-111-1111

我们可以为电话号码创建索引，以提高搜索速度，如下：

```
Index IdxP On Phone;
```

使用这种方式保存list数据比较简单，当是当list数据非常多时，这种方法会使表格臃肿。

2. 保存在一个字符串字段中，使用分隔符区分

这里我们将所有电话号码保存在一个字符串字段中，每个号码之间用逗号区分

```
Class Test.Person2 Extends %Persistent
{

Property Name As %String;

Property PhoneStr As %String;

}
```

填充数据后，类似于这样

Name	PhoneStr
a	111-111-1111,111-222-1111
b	222-111-1111,222-222-1111
c	333-111-1111
d	333-111-1111,222-222-1111

这种情况下我们可以用下面方法实现对每个电话的索引

Index idxP On PhoneStr(ELEMENTS);

```
ClassMethod PhoneStrBuildValueArray(value, ByRef array) As %Status
{
  if value="" {
    s array(0)=value
  }else{
    s list=$lfs(value,","),ptr=0
    while $listnext(list,ptr,item){
      s array(ptr)=item
    }
  }
  q $$$OK
}
```

这里用到了一个函数

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

其中：

- value – 需要拆分的内容；
- valueArray – 返回array类型的值，其中包含拆分的内容，格式为 array(key1)=value1, array(key2)=value2...

这时候我们的数据是这样：

```
USER>zw ^Test.Person2D
^Test.Person2D=4
^Test.Person2D(1)=$lb("","a","111-111-1111,111-222-1111")
^Test.Person2D(2)=$lb("","b","222-111-1111,222-222-1111")
^Test.Person2D(3)=$lb("","c","333-111-1111")
^Test.Person2D(4)=$lb("","d","333-111-1111,222-222-1111")
```

索引是这样：

```
USER>zw ^Test.Person2I
^Test.Person2I("idxP"," 111-111-1111",1)=""
^Test.Person2I("idxP"," 111-222-1111",1)=""
^Test.Person2I("idxP"," 222-111-1111",2)=""
^Test.Person2I("idxP"," 222-222-1111",2)=""
^Test.Person2I("idxP"," 222-222-1111",4)=""
^Test.Person2I("idxP"," 333-111-1111",3)=""
^Test.Person2I("idxP"," 333-111-1111",4)=""
```

这种情况下我们可以通过下面的SQL 语句查找 包含电话号码 333-111-1111 的姓名

```
select Name from test.person2 where phonestr ['333-111-1111'
select Name from test.person2 where phonestr like '%333-111-1111%'
```

但是当你检查查询计划的时候，却发现它**并没有使用任何索引**

我们只能通过类似于下面SQL语句的写法才能使用该索引

```
select Name from test.person2 where for some %element(Phonestr) (%value = '333-111-11
11')
```

类似的还有下面的写法

```
(%Value %STARTSWITH '?')
(%Value [ 'a' and %Value [ 'b')
(%Value in ('c','d'))
(%Value is null)
```

3. 使用 %List 类型

```
Class Test.Person3 Extends %Persistent
```

```
{

Property Name As %String;

Property PhoneList As %List;

Index idxP On PhoneList(ELEMENTS);

ClassMethod PhoneListBuildValueArray(value, ByRef array) As %Status
{
  if value="" {
    s array(0)=value
  }else{
    s ptr=0
    while $listnext(value,ptr,item){
      s array(ptr)=item
    }
  }
  q $$$OK
}

}
```

插入数据

```
insert into Test.Person3 (Name,PhoneList) select 'a', $LISTBUILD('111-111-1111','111-222-1111')

insert into Test.Person3 (Name,PhoneList) select 'b', $LISTBUILD('222-111-1111','222-222-1111')

insert into Test.Person3 (Name,PhoneList) select 'c', $LISTBUILD('333-111-1111')

insert into Test.Person3 (Name,PhoneList) select 'd', $LISTBUILD('333-111-1111','222-222-1111')
```

数据和索引保存为

```
USER>zw ^Test.Person3D
^Test.Person3D=4
^Test.Person3D(1)=$lb("","a",$lb("111-111-1111","111-222-1111"))
^Test.Person3D(2)=$lb("","b",$lb("222-111-1111","222-222-1111"))
^Test.Person3D(3)=$lb("","c",$lb("333-111-1111"))
^Test.Person3D(4)=$lb("","d",$lb("333-111-1111","222-222-1111"))

USER>zw ^Test.Person3I
^Test.Person3I("idxP","111-111-1111",1)=""
^Test.Person3I("idxP","111-222-1111",1)=""
^Test.Person3I("idxP","222-111-1111",2)=""
^Test.Person3I("idxP","222-222-1111",2)=""
^Test.Person3I("idxP","222-222-1111",4)=""
^Test.Person3I("idxP","333-111-1111",3)=""
^Test.Person3I("idxP","333-111-1111",4)=""
```

同样可以使用下面的SQL语句查找包含电话333-111-1111的姓名

```
select Name from test.person2 where for some %element(phonelist) (%value = '333-111-1
111')
```

4 使用 List Of、Array Of 保存

不需要定义propertynameBuildValueArray函数

```
Class Test.Person4 Extends %Persistent
{

Property Name As %String;

Property PhoneList As list Of %String;

Index idxP On PhoneList(ELEMENTS);

}
```

使用同样的方式插入数据

```
insert into Test.Person4 (Name,PhoneList) select 'a', $LISTBUILD('111-111-1111','111-
222-1111')
insert into Test.Person4 (Name,PhoneList) select 'b', $LISTBUILD('222-111-1111','222-
222-1111')
insert into Test.Person4 (Name,PhoneList) select 'c', $LISTBUILD('333-111-1111')
insert into Test.Person4 (Name,PhoneList) select 'd', $LISTBUILD('333-111-1111','222-
222-1111')
```

数据和索引保存为

```
USER>zw ^Test.Person4D
^Test.Person4D=4
^Test.Person4D(1)=$lb("","a",$lb("111-111-1111","111-222-1111"))
^Test.Person4D(2)=$lb("","b",$lb("222-111-1111","222-222-1111"))
^Test.Person4D(3)=$lb("","c",$lb("333-111-1111"))
^Test.Person4D(4)=$lb("","d",$lb("333-111-1111","222-222-1111"))

USER>zw ^Test.Person4I
^Test.Person4I("idxP"," 111-111-1111",1)=""
^Test.Person4I("idxP"," 111-222-1111",1)=""
^Test.Person4I("idxP"," 222-111-1111",2)=""
^Test.Person4I("idxP"," 222-222-1111",2)=""
^Test.Person4I("idxP"," 222-222-1111",4)=""
^Test.Person4I("idxP"," 333-111-1111",3)=""
^Test.Person4I("idxP"," 333-111-1111",4)=""
```

使用同样的SQL查询可以得到结果

```
select Name from test.person4 where for some %element(Phonelist) (%value = '333-111-1
111')
```

引申话题：针对日期字段的索引

日期格式通常是yyyy-mm-dd，我们经常要求按照某年或者某月查询数据，我们可以使用propertynameBuildValueArray函数设定保存的索引方式实现这个目的

```
Class Test.Person5 Extends %Persistent
{

Property Name As %String;

Property DOB As %Date;

Index idxD On (DOB(KEYS), DOB(ELEMENTS));

ClassMethod DOBBuildValueArray(value, ByRef array) As %Status
{
  if value="" {
    s array(0)=value
  }else{
    s d=$zd(value,3)
    s array("yy")=+$p(d,"-",1)
    s array("mm")=+$p(d,"-",2)
    s array("dd")=+$p(d,"-",3)

  }
  q $$$OK
}

}
```

插入数据

```
insert into Test.Person5 (Name,DOB)
  select 'a', {d '2000-01-01'} union all
  select 'b', {d '2000-01-02'} union all
  select 'c', {d '2000-02-01'} union all
  select 'd', {d '2001-01-02'} union all
  select 'e', {d '2001-01-01'} union all
  select 'f', {d '2001-02-01'}
```

查看数据以及索引保存的内容

```
USER>zw ^Test.Person5D
^Test.Person5D=6
^Test.Person5D(1)=$lb("","a",58074)
^Test.Person5D(2)=$lb("","b",58075)
^Test.Person5D(3)=$lb("","c",58105)
^Test.Person5D(4)=$lb("","d",58441)
^Test.Person5D(5)=$lb("","e",58440)
^Test.Person5D(6)=$lb("","f",58471)

USER>zw ^Test.Person5I
^Test.Person5I("idxD","dd",1,1)=""
^Test.Person5I("idxD","dd",1,3)=""
^Test.Person5I("idxD","dd",1,5)=""
^Test.Person5I("idxD","dd",1,6)=""
^Test.Person5I("idxD","dd",2,2)=""
```

```
^Test.Person5I("idxD","dd",2,4)=""
^Test.Person5I("idxD","mm",1,1)=""
^Test.Person5I("idxD","mm",1,2)=""
^Test.Person5I("idxD","mm",1,4)=""
^Test.Person5I("idxD","mm",1,5)=""
^Test.Person5I("idxD","mm",2,3)=""
^Test.Person5I("idxD","mm",2,6)=""
^Test.Person5I("idxD","yy",2000,1)=""
^Test.Person5I("idxD","yy",2000,2)=""
^Test.Person5I("idxD","yy",2000,3)=""
^Test.Person5I("idxD","yy",2001,4)=""
^Test.Person5I("idxD","yy",2001,5)=""
^Test.Person5I("idxD","yy",2001,6)=""
```

执行下面 SQL 可以显示所有2月出生的信息

```
select * from Test.Person5 where for some %element(DOB) (%key='mm' and %value = 2)
```

这篇文章源自 这里，作者 Vitaliy Serdtsev

#InterSystems IRIS

源 URL: https://cn.community.intersystems.com/post/%E5%A6%82%E4%BD%95%E4%BF%9D%E5%AD%98%E3%80%81%E6%9F%A5%E8%AF%A2-list-%E7%B1%BB%E5%9E%8B%E6%95%B0%E6%8D%AE