文章
姚 鑫 · 五月 10, 2021 阅读大约需 5 分钟

第四章 多维存储的SQL和对象使用（二）

第四章 多维存储的SQL和对象使用（二）

索引

持久化类可以定义一个或多个索引；其他数据结构用于提高操作(如排序或条件搜索)的效率。InterSystems SQL在执行查询时使用这些索引。InterSystems IRIS对象和SQL在执行INSERT、UPDATE和DELETE操作时自动维护索引内的正确值。

标准索引的存储结构

标准索引将一个或多个属性值的有序集与包含属性的对象的对象ID值相关联。

例如，假设我们定义了一个简单的持久化MyApp.Person类，该类具有两个文本属性和一个关于其Name属性的索引：

```
Class MyApp.Person Extends %Persistent
{
Index NameIdx On Name;

Property Name As %String;
Property Age As %Integer;
}
```

如果我们创建并保存此Person类的多个实例，则生成的数据和索引全局变量类似于：

```
 // data global
 ^MyApp.PersonD = 3  // counter node
 ^MyApp.PersonD(1) = $LB("",34,"Jones")
 ^MyApp.PersonD(2) = $LB("",22,"Smith")
 ^MyApp.PersonD(3) = $LB("",45,"Jones")


 // index global
 ^MyApp.PersonI("NameIdx"," JONES",1) = ""
 ^MyApp.PersonI("NameIdx"," JONES",3) = ""
 ^MyApp.PersonI("NameIdx"," SMITH",2) = ""
```

请注意有关全局索引的以下事项：
1. 默认情况下，它被放在一个全局变量中，全局变量的名称是后面附加 “ i ” (表示索引)的类名。
2. 默认情况下，第一个下标是索引名；这允许将多个索引存储在同一全局中，而不会发生冲突。
3. 第二个下标包含整理后的数据值。在这种情况下，使用默认的SQLUPPER排序函数对数据进行排序。这会将所有字符转换为大写(不考虑大小写进行排序)，并在前面加上一个空格字符(强制所有数据作为字符串进行排序)。
4. 第三个下标包含包含索引数据值的对象的对象ID值。

5. 节点本身是空的；所有需要的数据都保存在下标中。请注意，如果索引定义指定数据应与索引一起存储，则将其放置在全局索引的节点中。

该索引包含足够的信息来满足许多查询，比如按姓名列出所有Person类。

位图索引

位图索引类似于标准索引，不同之处在于它使用一系列位字符串来存储与索引值对应的一组对象ID值。

位图索引的逻辑运算

位字符串是一个包含一组特殊压缩格式的位(0和1值)的字符串。
InterSystems IRIS包含一组有效创建和使用位字符串的函数。
这些都列在下表中:

位操作

函数	描述
$Bit	在位串中设置或获取位。
$BitCount	计算位串中的位数。
$BitFind	查找位串中下一个出现的位。
$BitLogic	对两个或多个位串执行逻辑(AND, OR)操作。

在位图索引中，位字符串中的顺序位置对应于索引表中的行(对象ID号)。
对于给定值，位图索引维护一个位字符串，在给定值存在的每一行中包含1，在没有给定值的每一行中包含0。
请注意，位图索引只适用于使用系统分配的默认存储结构的对象，数值型对象ID值。

例如，假设我们有一个类似如下的表:

ID	State	Product
1	MA	Hat
2	NY	Hat
3	NY	Chair
4	MA	Chair
5	MA	Hat

如果State和Product列有位图索引，则它们包含以下值:

State列上的位图索引包含以下位字符串值:

```
MA  1   0   0   1   1
NY  0   1   1   0   0
```

注意，对于值“ MA ”，在与State等于“ MA ”的表行对应的位置(1、4和5)中有一个1。

类似地，Product列上的位图索引包含以下位字符串值(注意，这些值在索引中被排序为大写):

```
CHAIR   0   0   1   1   0
HAT 1   1   0   0   1
```

InterSystems SQL Engine可以通过对这些索引维护的位串进行迭代、计算位内位数或执行逻辑组合(AND, or)来执行许多操作。
例如，要找到State等于“ MA ”、Product等于“ HAT ”的所有行，SQL引擎可以简单地将适当的位串与逻辑and组合在一起。

除了这些索引之外，系统还维护一个额外的索引，称为“ 区段索引 ”，对于存在的每一行包含1，对于不存在的行(如已

删除的行)包含0。
这用于某些操作，如否定。

位图索引的存储结构

位图索引将一个或多个属性值的有序集合与一个或多个包含与属性值对应的对象ID值的位字符串相关联。

例如，假设我们定义了一个简单的持久MyApp。
Person类具有两个文字属性和Age属性上的位图索引:

```
Class MyApp.Person Extends %Persistent
{
Index AgeIdx On Age [Type = bitmap];

Property Name As %String;
Property Age As %Integer;
}
```

如果我们创建并保存这个Person类的几个实例，得到的数据和索引全局变量类似于:

```
 // data global
 ^MyApp.PersonD = 3  // counter node
 ^MyApp.PersonD(1) = $LB("",34,"Jones")
 ^MyApp.PersonD(2) = $LB("",34,"Smith")
 ^MyApp.PersonD(3) = $LB("",45,"Jones")

 // index global
 ^MyApp.PersonI("AgeIdx",34,1) = 110...
 ^MyApp.PersonI("AgeIdx",45,1) = 001...

 // extent index global
 ^MyApp.PersonI("$Person",1) = 111...
 ^MyApp.PersonI("$Person",2) = 111...
```

关于全局索引，请注意以下几点:
1. 默认情况下，它被放置在一个全局变量中，全局变量的名称是类名，后面附加一个 “ I ” (表示Index)。
2. 默认情况下，第一个下标是索引名;这允许多个索引存储在同一个全局中，而不会发生冲突。
3. 第二个下标包含经过整理的数据值。在这种情况下，不应用排序函数，因为这是数字数据的索引。
4.
第三个下标包含块编号；为了提高效率，位图索引被分成一系列位串，每个位串包含表中大约64000行的信息。这些位串中的每一个都被称为块。
5. 节点包含位串。

另请注意：因为该表有一个位图索引，所以会自动维护一个区索引。该盘区索引存储在索引GLOBAL中，并使用前缀有 “ $ ” 字符的类名作为其第一个下标。

位图索引的直接访问

下面的示例使用类区索引来计算存储的对象实例(行)的总数。注意，它使用$ORDER来迭代区索引的块(每个块包含大约64000行的信息)：

```
ClassMethod Count1() As %Integer
```

```
{
    New total,chunk,data
    Set total = 0

    Set chunk = $Order(^Sample.PersonI("$Person",""),1,data)
    While (chunk '= "") {
        Set total = total + $bitcount(data,1)
        Set chunk = $Order(^Sample.PersonI("$Person",chunk),1,data)
    }

    Quit total
}


DHC-APP>w ##class(PHA.TEST.SQL).Count1()
208
```

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

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E5%9B%9B%E7%AB%A0-%E5%A4%9A%E7%BB%B4%E5%AD%98%E5%82%A8%E7%9A%84sql%E5%92%8C%E5%AF%B9%E8%B1%A1%E4%BD%BF%E7%94%A8%EF%BC%88%E4%BA%8C%EF%BC%89