文章
姚 鑫 · 六月 22, 2022 阅读大约需 4 分钟

第七章 操作位和位串（三）

第七章 操作位和位串（三）

操作位串

要创建新的位串，请使用 $bit 函数将所需位设置为 1：

```
kill bitstring

set $bit(bitstring, 3) = 1

set $bit(bitstring, 6) = 1

set $bit(bitstring, 11) = 1
```

使用 $bit 将现有位串中的位设置为 1：

```
set $bit(bitstring, 5) = 1
```

使用 $bit 将现有位串中的位设置为 0：

```
set $bit(bitstring, 5) = 0
```

由于位串中的第一位是位 1，因此尝试设置位 0 会返回错误：

```
set $bit(bitstring, 0) = 1

SET $BIT(bitstring, 0) = 1
^
<VALUE OUT OF RANGE>
```

测试位是否已设置

要测试是否在现有位串中设置了位，还可以使用 $bit 函数：

```
write $bit(bitstring, 6)
1
write $bit(bitstring, 5)
0
```

如果测试未明确设置的位，则 $bit 返回 0：

```
write $bit(bitstring, 4)
0
write $bit(bitstring, 55)
0
```

显示位

要显示位串中的位，请使用 $bitcount 函数获取位串中位的计数，然后遍历位：

```
for i=1:1:$bitcount(bitstring) {write $bit(bitstring, i)}
00100100001
```

还可以使用 $bitcount 来计算位串中 1 或 0 的数量：

```
write $bitcount(bitstring, 1)
3
write $bitcount(bitstring, 0)
8
```

查找设置位

要查找在位串中设置了哪些位，请使用 $bitfind
函数，该函数返回指定值的下一位的位置，从位串中的给定位置开始：

```
Class User.BitStr
{

ClassMethod FindSetBits(bitstring As %String)
{
   set bit = 0
   for {
      set bit = $bitfind(bitstring, 1, bit)
      quit:'bit
      write bit, " "
      set bit = bit + 1
   }
}

}
```

此方法搜索字符串并在 $bitfind 返回 0 时退出，表示没有找到更多匹配项。

```
do ##class(User.BitStr).FindSetBits(bitstring)
3 6 11
```

在测试位串的比较时要非常小心。

例如，可以有两个位串 b1 和 b2，它们具有相同的位集：

```
do ##class(User.BitStr).FindSetBits(b1)
3 6 11
do ##class(User.BitStr).FindSetBits(b2)
3 6 11
```

然而，如果你比较它们，你会发现它们实际上并不相等：

```
write b1 = b2
0
```

如果你使用 zwrite，你可以看到这两个比特环的内部表示是不同的：

```
zwrite b1
b1=$zwc(405,2,2,5,10)/*$bit(3,6,11)*/

zwrite b2
b2=$zwc(404,2,2,5,10)/*$bit(3,6,11)*/
```

在这种情况下，b2 将第 12 位设置为 0：

```
for i=1:1:$bitcount(b1) {write $bit(b1, i)}
00100100001
USER>for i=1:1:$bitcount(b2) {write $bit(b2, i)}
001001000010
```

此外，还可能存在其他内部表示，例如由 $factor 创建的表示，它将整数转换为位串：

```
set b3 = $factor(1060)

zwrite b3
b3=$zwc(128,4)_$c(36,4,0,0)/*$bit(3,6,11)*/

for i=1:1:$bitcount(b3) {write $bit(b3, i)}
00100100001000000000000000000000
```

要比较可能具有不同内部表示的两个位串，可以使用 $bitlogic 函数直接比较设置的位，如执行按位算术中所述。

执行按位算术

使用 $bitlogic 函数对位串执行按位逻辑运算。

在此示例中，有两个位串 a 和 b。

```
for i=1:1:$bitcount(a) {write $bit(a, i)}
100110111
for i=1:1:$bitcount(b) {write $bit(b, i)}
001000101
```

使用 $bitlogic 函数对位执行逻辑或：

```
set c = $bitlogic(a|b)

for i=1:1:$bitcount(c) {write $bit(c, i)}
101110111
```

使用 $bitlogic 函数对位执行逻辑与：

```
set d = $bitlogic(a&b)

for i=1:1:$bitcount(d) {write $bit(d, i)}
000000101
```

此示例说明如何使用 $bitlogic 函数执行逻辑 XOR 以测试两个位串 b1 和 b3
是否具有相同的设置位，而不管内部表示如何：

```
zwrite b1
b1=$zwc(405,2,2,5,10)/*$bit(3,6,11)*/

zwrite b3
b3=$zwc(128,4)_$c(36,4,0,0)/*$bit(3,6,11)*/

write $bitcount($bitlogic(b1^b3),1)
0
```

逻辑异或可以快速表明两个位串的设置位没有差异。

转换为位串整数

要将常规位串转换为存储为整数的位串，请使用 $bitfind 函数查找设置的位并将它们的 2
次方相加。请记住将结果除以 2 以将位向左移动，因为常规位串中的位 1 对应于位串中的位 0。

```
ClassMethod BitstringToInt(bitstring As %String)
{
   set bitint = 0
   set bit = 0
   for {
      set bit = $bitfind(bitstring, 1, bit)
      quit:'bit
      set bitint = bitint + (2**bit)
      set bit = bit + 1
   }
   return bitint/2
}
```

将位串转换为位串为整数：

```
for i=1:1:$bitcount(bitstring) {write $bit(bitstring, i)}
00100100001
set bitint = ##class(User.BitStr).BitstringToInt(bitstring)

write bitint
1060
```

#SQL #Caché

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E4%B8%83%E7%AB%A0-%E6%93%8D%E4%BD%9C%E4%BD%8D%E5%92%8C%E4%BD%8D%E4%B8%B2%EF%BC%88%E4%B8%89%EF%BC%89