文章
姚 鑫 · 二月 26, 2021 阅读大约需 9 分钟

第四十八章 Caché 变量大全 ^$LOCK 变量

第四十八章 Caché 变量大全 ^$LOCK 变量

提供锁名信息。

大纲

```
^$|nspace|LOCK(lock_name,info_type,pid)
^$|nspace|L(lock_name,info_type,pid)
```

参数

- |nspace|或[nspace] [nspace]可选-扩展SSVN引用，显式名称空间名称或隐含名称空间。必须计算为带引号的字符串，该字符串括在方括号([“ nspace ”])或竖线(| “ nspace ” |)中。命名空间名称不区分大小写；它们以大写字母存储和显示。
- lockname 计算结果为包含锁定变量名称(带下标或无下标)的字符串的表达式。如果是文字，则必须指定为带引号的字符串。
- infotype 可选-解析为所有大写字母指定为带引号字符串的关键字的表达式。infotype指定返回关于lockname的哪种类型的信息。可用选项有 “ 所有者 ” 、 “ 标志 ” 、 “ 模式 ” 和 “ 计数 ” 。作为独立函数调用^$LOCK时需要。
- pid 可选-用于 “ 计数 ” 关键字。一个整数，指定锁所有者的进程标识。如果指定，最多为 “ 计数 ” 返回一个列表元素。如果省略(或指定为0)，将为持有指定锁的每个所有者返回一个列表元素。pid对其他infotype关键字没有影响。

描述

^$LOCK结构化系统变量返回有关当前命名空间或本地系统上指定命名空间中的锁的信息。可以通过两种方式使用^$LOCK:
- infotype作为独立函数返回指定锁的信息。
- $DATA、$ORDER或$QUERY函数没有infotype作为参数。

注意:^$LOCK从本地系统的锁表中检索锁表信息。它不会从远程服务器上的锁表中返回信息。

ECP环境中的^$LOCK。

- 本地系统:为本地系统持有的锁调用^$LOCK时，^$LOCK的行为与没有ECP时相同，只有一个例外:infotype的 “ FLAGS ” 返回一个星号(*)，表示锁处于ECP环境中。这意味着一旦锁被释放，远程系统间IRIS实例就能够持有锁。
- 应用服务器:当通过ECP(数据服务器或另一个应用服务器)在一个应用服务器上为另一个服务器上持有的锁调用^$LOCK时，^$LOCK不返回任何信息。请注意，这是与锁不存在时相同的行为。
- 数据服务器:当在数据服务器上为应用服务器持有的锁调用^$LOCK时，^$LOCK的行为与本地系统略有不同，如下所示:
    - “ Owner ” ：如果锁由连接到调用^$Lock的数据服务器的应用程序服务器持有，则^$Lock(LOCKNAME，“ OWNER ”)为持有该锁的实例返回 “ ECPConfigurationName:MachineName:InstanceName ” ，但不标识持有该锁的特定进程。
    - “ FLAGS ” ：如果锁由连接到调用^$LOCK的数据服务器的应用程序服务器持有，则独占锁的^$LOCK(l

ockname，“ FLAGS ”)将返回“ Z ”标志。这个“ Z ”表示除了ECP环境之外，在InterSystems IRIS中不再使用的一种遗留锁。
- “ mode ”：如果锁由连接到调用^$lock的数据服务器的应用程序服务器持有，则独占锁的^$lock(lockname，“ mode ”)返回“ ZAX ”而不是“ X ”。ZA是一种遗留锁，除ECP环境外，在系统间IRIS中不再使用。对于共享锁，^$lock(lockname，“ mode ”)返回“ S ”，与本地锁相同。

参数

nspace

此可选参数允许您使用扩展的SSVN引用在另一个名称空间中指定全局变量。可以显式指定名称空间名称，将其命名为带引号的字符串文字或变量，或者通过指定隐式名称空间。命名空间名称不区分大小写。可以使用方括号语法[“ USER ”]或环境语法|“ USER ”|。 nspace分隔符前后不允许有空格。

可以使用以下方法测试是否定义了名称空间：

```
WRITE ##class(%SYS.Namespace).Exists("USER"),!
WRITE ##class(%SYS.Namespace).Exists("LOSER")
```

可以使用$NAMESPACE特殊变量来确定当前的名称空间。更改当前名称空间的首选方法是NEW $NAMESPACE，然后SET $NAMESPACE = “ nspacename ”。

lockname

该表达式的计算结果为包含锁定变量名称（带下标或未下标）的字符串。使用LOCK命令定义一个锁变量（通常是全局变量）。

infotype

当将^$LOCK用作独立函数时，需要一个infotype关键字；当将^$LOCK用作另一个函数的参数时，则是一个可选参数。 infotype必须以大写字母指定为带引号的字符串。

- “ OWNER ”返回锁所有者的进程ID（pid）。如果该锁是共享锁，则以逗号分隔列表的形式返回该锁的所有所有者的进程ID。如果指定的锁不存在，则^$LOCK返回空字符串。
- “ FLAGS ”返回锁的状态。它可以返回以下值：“ D ”-处于挂起挂起状态；“ P ”-处于锁定挂起状态；“ N ”-这是一个节点锁定，后代未锁定；“ Z ”-此锁处于ZAX模式；“ L ”-锁丢失，服务器不再具有此锁；“ * ”-这是一个远程锁定。如果指定的锁处于正常锁状态或不存在，则^$LOCK返回空字符串。
- “ MODE ”返回当前节点的锁定模式。对于排他锁定模式，它返回“ X ”，对于共享锁定模式，它返回“ S ”，对于ZALLOCATE锁定模式，它返回“ ZAX ”。如果指定的锁不存在，则^$LOCK返回空字符串。
- “ COUNTS ”返回锁的锁计数，指定为二进制列表结构。对于排他锁，列表包含一个元素；对于共享锁，列表包含每个锁所有者的元素。可以使用pid参数仅返回指定锁定所有者的list元素。每个元素都包含所有者的pid，独占模式增量计数和共享模式增量计数。如果独占模式和共享模式的增量计数均为0（或“ ”），则锁定处于“ ZAX ”模式。增量计数后可以跟一个“ D ”，以指示该锁已在当前事务中解锁，但是其释放被延迟（“ D ”），直到事务被提交或回滚为止。如果指定的锁不存在，则^$LOCK返回空字符串。

必须使用所有大写字母指定infotype关键字。指定无效的infotype关键字会生成<SUBSCRIPT>错误。

pid

锁所有者的进程ID。仅在使用“ COUNTS ”关键字时有意义。用于将“ ”返回值限制为（最多）一个列表元素。pid在所有平台上均指定为整数。如果pid与lockname ^$LOCK的所有者的进程ID匹配，则返回该所有者的“ COUNTS ”列表元素；如果pid与lockname

^$LOCK的所有者的进程ID不匹配，则返回空字符串。将pid指定为0表示与省略pid相同；
^$LOCK返回所有“ COUNTS ”列表元素。pid参数与“ OWNER ”，“ FLAGS ”或“ MODE ”关键字一起使用，但被忽略。

示例

下面的示例显示由infotype关键字返回的排他锁的值：

```
/// d ##class(PHA.TEST.SpecialVariables).LOCK()
ClassMethod LOCK()
{
    LOCK ^B(1,1)  ; define lock
    WRITE !,"lock owner: ",^$LOCK("^B(1,1)","OWNER")
    WRITE !,"lock flags: ",^$LOCK("^B(1,1)","FLAGS")
    WRITE !,"lock mode: ",^$LOCK("^B(1,1)","MODE")
    WRITE !,"lock counts: "
    ZZDUMP ^$LOCK("^B(1,1)","COUNTS")
    LOCK -^B(1,1) ; delete lock
}
```

```
DHC-APP>d ##class(PHA.TEST.SpecialVariables).LOCK()

lock owner: 17824
lock flags:
lock mode: X
lock counts:
0000: 0B 01 04 04 A0 45 03 04 01 02 01                          ....??E.....
```

下面的示例显示在递增和递减独占锁时，infotype“ COUNTS ”返回的值如何变化：

```
/// d ##class(PHA.TEST.SpecialVariables).LOCK1()
ClassMethod LOCK1()
{
    LOCK ^B(1,1)      ; define exclusive lock
    ZZDUMP ^$LOCK("^B(1,1)","COUNTS")
    LOCK +^B(1,1) ; increment lock
    ZZDUMP ^$LOCK("^B(1,1)","COUNTS")
    LOCK +^B(1,1) ; increment lock again
    ZZDUMP ^$LOCK("^B(1,1)","COUNTS")
    LOCK -^B(1,1) ; decrement lock
    ZZDUMP ^$LOCK("^B(1,1)","COUNTS")
    LOCK -^B(1,1) ; decrement lock again
    ZZDUMP ^$LOCK("^B(1,1)","COUNTS")
    LOCK -^B(1,1) ; delete exclusive lock
}
```

```
DHC-APP>d ##class(PHA.TEST.SpecialVariables).LOCK1()

0000: 0B 01 04 04 A0 45 03 04 01 02 01                          ....??E.....
0000: 0B 01 04 04 A0 45 03 04 02 02 01                          ....??E.....
0000: 0B 01 04 04 A0 45 03 04 03 02 01                          ....??E.....
0000: 0B 01 04 04 A0 45 03 04 02 02 01                          ....??E.....
```

```
0000: 0B 01 04 04 A0 45 03 04 01 02 01                          ....??E.....
```

下面的示例显示在递增和递减共享锁时，infotype “ COUNTS ” 返回的值如何变化

```
/// d ##class(PHA.TEST.SpecialVariables).LOCK2()
ClassMethod LOCK2()
{
    LOCK ^S(1,1)#"S"   ; define shared lock
    ZZDUMP ^$LOCK("^S(1,1)","COUNTS")
    LOCK +^S(1,1)#"S" ; increment lock
    ZZDUMP ^$LOCK("^S(1,1)","COUNTS")
    LOCK +^S(1,1)#"S" ; increment lock again
    ZZDUMP ^$LOCK("^S(1,1)","COUNTS")
    LOCK -^S(1,1)#"S" ; decrement lock
    ZZDUMP ^$LOCK("^S(1,1)","COUNTS")
    LOCK -^S(1,1)#"S" ; decrement lock again
    ZZDUMP ^$LOCK("^S(1,1)","COUNTS")
    LOCK -^S(1,1)#"S" ; delete shared lock
}
```

```
DHC-APP>d ##class(PHA.TEST.SpecialVariables).LOCK2()

0000: 0B 01 04 04 A0 45 02 01 03 04 01                          ....??E.....
0000: 0B 01 04 04 A0 45 02 01 03 04 02                          ....??E.....
0000: 0B 01 04 04 A0 45 02 01 03 04 03                          ....??E.....
0000: 0B 01 04 04 A0 45 02 01 03 04 02                          ....??E.....
0000: 0B 01 04 04 A0 45 02 01 03 04 01                          ....??E.....
```

以下示例显示如何将^$lock用作$DATA、$ORDER和$QUERY函数的参数。

作为$DATA的参数

$DATA(^$|nspace|LOCK(lockname))

^$lock作为$DATA的参数返回一个整数值，该值指定锁定名称是否作为节点存在于^$lock中。下表显示了$DATA可以返回的整数值。

Value	Meaning
0	锁信息不存在
10	锁信息存在

请注意，在此上下文中使用的$DATA只能返回0或10，其中10表示指定的锁存在。它不能确定锁是否有后代，也不能返回1或11。

下面的示例测试当前命名空间中是否存在锁名。第一次写入返回10(锁名存在)，第二次写入返回0(锁名不存在)：

```
/// d ##class(PHA.TEST.SpecialVariables).LOCK3()
ClassMethod LOCK3()
{
    LOCK ^B(1,2)  ; define lock
    WRITE !,$DATA(^$LOCK("^B(1,2)"))
    LOCK -^B(1,2) ; delete lock
    WRITE !,$DATA(^$LOCK("^B(1,2)"))
}
```

```
DHC-APP>d ##class(PHA.TEST.SpecialVariables).LOCK3()

10
0
```

作为$ORDER的参数

$ORDER(^$|nspace|LOCK(lockname),direction)

^$lock作为$ORDER的参数，按排序顺序将下一个或上一个^$lock锁名节点返回到指定的锁名。如果不存在这样的锁名作为^$lock节点，$ORDER将返回空字符串。

锁以区分大小写的字符串排序顺序返回。使用数字排序规则以下标树顺序返回命名锁的下标。

Direction参数指定是返回下一个锁名称还是返回上一个锁名称。如果不提供方向参数，InterSystems IRIS会将排序序列中的下一个锁名返回到您指定的锁名。

以下子例程在Samples名称空间中搜索锁，并将锁名称存储在名为locket的本地数组中。

```
/// d ##class(PHA.TEST.SpecialVariables).LOCK4()
ClassMethod LOCK4()
{
LOCKARRAY
    SET lname=""
    FOR I=1:1 {
    SET lname=$ORDER(^$|"SAMPLES"|LOCK(lname))
        QUIT:lname=""
        SET LOCKET(I)=lname
        WRITE !,"the lock name is: ",lname
    }
    WRITE !,"All lock names listed"
    QUIT
}
```

```
DHC-APP>d ##class(PHA.TEST.SpecialVariables).LOCK4()

the lock name is: ^%SYS("CSP","Daemon")
All lock names listed
```

作为$QUERY的参数

$QUERY(^$|nspace|LOCK(lockname))

^$lock作为$query的参数，将排序序列中的下一个锁名返回到您指定的锁名。如果没有将下一个锁名定义为^$lock中的节点，则$query将返回空字符串。

锁以区分大小写的字符串排序顺序返回。使用数字排序规则以下标树顺序返回命名锁的下标。

在下面的示例中，在当前命名空间中(按随机顺序)创建了五个全局锁名称。

```
/// d ##class(PHA.TEST.SpecialVariables).LOCK5()
ClassMethod LOCK5()
{
```

```
    LOCK (^B(1),^A,^D,^A(1,2,3),^A(1,2))
    WRITE !,"lock name: ",$QUERY(^$LOCK(""))
    WRITE !,"lock name: ",$QUERY(^$LOCK("^C"))
    WRITE !,"lock name: ",$QUERY(^$LOCK("^A(1,2)"))
}


DHC-APP>d ##class(PHA.TEST.SpecialVariables).LOCK5()

lock name: ^$LOCK("^%SYS(""CSP"",""Daemon"")")
lock name: ^$LOCK("^D")
lock name: ^$LOCK("^A(1,2,3)")
```

$QUERY将所有全局锁变量名(带下标或无下标)视为字符串，并按字符串排序顺序检索它们。因此，$QUERY(^$LOCK(“ ”))按排序顺序检索第一个锁名：^$LOCK(“ ^A ”)或排序序列中位置较高的InterSystems IRIS定义的锁。$QUERY(^$LOCK(“ ^C ”))检索排序序列中不存在的^C：^$LOCK(“ ^D ”)之后的下一个锁名。$QUERY(^$LOCK(“ ^A(1，2) ”))检索排序规则序列中它后面的^$LOCK(“ ^A(1，2，3) ”)。

#Caché #InterSystems IRIS #InterSystems IRIS for Health

源 URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E5%9B%9B%E5%8D%81%E5%85%AB%E7%AB%A0-cach%C3%A9-%E5%8F%98%E9%87%8F%E5%A4%A7%E5%85%A8-lock-%E5%8F%98%E9%87%8F