文章
姚 鑫 · 十一月 22, 2021 阅读大约需 6 分钟

第八十四章 SQL命令 UPDATE（三）

第八十四章 SQL命令 UPDATE（三）

参照完整性

如果没有指定%NOCHECK, IRIS将使用系统范围的配置设置来确定是否执行外键引用完整性检查;
默认值是执行外键引用完整性检查。
可以在系统范围内设置此默认值，如外键引用完整性检查中所述。
要确定当前系统范围的设置，调用$SYSTEM.SQL.CurrentSettings()。

此设置不适用于用NOCHECK关键字定义的外键。

在UPDATE操作期间，对于每个具有更新字段值的外键引用，都会在被引用表中的旧(更新前)引用行和新(更新后)引用行上获得共享锁。
这些行在执行引用完整性检查和更新行时被锁定。
然后释放锁(直到事务结束才持有锁)。
这确保了引用的行不会在引用完整性检查和更新操作完成之间发生更改。
锁定旧行可以确保在可能的UPDATE回滚之前不会更改所引用的行。
锁定新行可以确保引用的行不会在引用完整性检查和更新操作完成之间发生更改。

如果对CASCADE、SET NULL或SET DEFAULT定义的外键字段执行了带有%NOLOCK的UPDATE操作，则相应的更改外键表的引用操作也会使用%NOLOCK。

原子性

默认情况下，UPDATE、INSERT、DELETE和TRUNCATE TABLE是原子操作。
UPDATE要么成功完成，要么回滚整个操作。
如果任何指定的行不能更新，则不更新指定的行，数据库将恢复到发出UPDATE之前的状态。

可以通过调用SET TRANSACTION %COMMITMODE来修改SQL中当前进程的这个默认值。
您可以通过调用SetOption()方法在ObjectScript中修改当前进程的这个默认值，如下SET status=$SYSTEM.SQL.Util.SetOption("AutoCommit"，intval，.oldval)。
以下intval整数选项是可用的:

- 1或IMPLICIT (autocommit on)——默认行为，如上所述。
  每个UPDATE构成一个单独的事务。
- 2或EXPLICIT (autocommit off) -如果没有事务在进行中，UPDATE会自动启动一个事务，但是你必须显式地COMMIT或ROLLBACK来结束事务。
  在EXPLICIT模式下，每个事务的数据库操作数是用户定义的。
- 0或NONE(没有自动事务)——调用UPDATE时不会启动任何事务。
  失败的UPDATE操作可能会使数据库处于不一致的状态，一些指定的行被更新，而一些未被更新。
  要在此模式中提供事务支持，必须使用START transaction来启动事务，并使用COMMIT或ROLLBACK来结束事务。

分片表始终没有自动事务模式，这意味着对分片表的所有插入、更新和删除都是在事务范围之外执行的。

可以使用GetOption(“ AutoCommit ”)方法确定当前进程的原子性设置，如下面的ObjectScript示例所示:

```
ClassMethod Update()
{
    s stat = $SYSTEM.SQL.SetOption("AutoCommit", $RANDOM(3),.oldval)
    if stat '= 1 {
        w "SetOption failed:" d $System.Status.DisplayError(stat) q
    }
    s x = $SYSTEM.SQL.GetOption("AutoCommit")
    if x = 1 {
        w "????????",!
        w "???????"
    } elseif x=0 {
        w "?????????????:",!
        w "DELETE???????????",!
        w "?????"
    } else {
        w "?????????"
    }
}
```

事务锁

如果没有指定%NOLOCK，系统将自动对INSERT、UPDATE和DELETE操作执行标准的记录锁定。
在当前事务期间，每个受影响的记录(行)都被锁定。

默认的锁阈值是每个表1000个锁。
这意味着，如果在事务期间从表中更新超过1000条记录，就会达到锁阈值，
IRIS会自动将锁级别从记录锁升级到表锁。
这允许在事务期间进行大规模更新，而不会溢出锁表。

IRIS应用以下两种锁升级策略之一:
- “ E ”类型的锁升级: IRIS使用这种类型的锁升级，如果以下条件为真:
持久性(可以从Management Portal SQL模式显示的Catalog Details中确定这一点)。
(2)类要么不指定IDKey索引，要么指定单一属性的IDKey索引。
“ E ”类型的锁升级在ObjectScript Reference中的lock命令中进行了描述。
- 传统SQL锁升级:类不使用 “ E ”类型锁升级的最可能的原因是存在一个多属性IDKey索引。
在本例中，每个%Save都会增加锁计数器的值。
这意味着如果在事务中保存单个对象1001次， IRIS将尝试升级锁。

对于这两种锁升级策略，可以使用$SYSTEM.SQL.Util.GetOption(“ LockThreshold ”)方法确定当前系统范围的锁阈值。
默认值是1000。
这个系统范围的锁阈值是可配置的:

- 使用$SYSTEM.SQL.Util.SetOption("LockThreshold")方法。
- 通过管理门户。
  进入系统管理，配置，SQL和对象设置，SQL。
  查看和编辑 “ 锁定升级阈值 ” 的当前设置。
  默认值是1000个锁。
  如果更改此设置，则更改后启动的任何新进程都将具有新设置。

需要在 “ %Admin Manage Resource ” 中具有 “ USE ” 权限才能修改锁定阈值。
IRIS会立即将对锁阈值的任何更改应用到所有当前进程。

自动锁升级的潜在后果是，当试图升级到表锁的进程与持有该表中记录锁的另一个进程冲突时，可能发生死锁情况。
有几种可能的策略可以避免这种情况:(1)增加锁升级阈值，以便锁升级不太可能在事务中发生。
(2)大幅降低锁升级阈值，以便锁升级几乎立即发生，从而减少其他进程锁定同一表中的记录的机会。
(3)在事务期间应用表锁，不执行记录锁。
这可以在事务开始时指定LOCK TABLE，然后指定UNLOCK
TABLE(没有IMMEDIATE关键字，以便表锁持续到事务结束)，然后使用%NOLOCK选项执行更新。

自动锁升级旨在防止锁表溢出。
但是，如果执行的更新数量如此之多，以致出现<LOCKTABLEFULL>错误，UPDATE将发出SQLCODE -110错误

计数器递增

- ROWVERSION
- SERIAL (%Counter)

ROWVERSION计数器增量

如果一个表有一个数据类型为ROWVERSION的字段，那么对一行执行更新将自动更新该字段的整数值。
ROWVERSION字段接受来自名称空间范围的行版本计数器的下一个顺序整数。
试图指定ROWVERSION字段的更新值将导致SQLCODE -138错误。

SERIAL (%Counter)计数器增量

UPDATE操作对SERIAL (%Library.Counter)计数器字段值没有影响。
但是，使用INSERT OR update执行的更新会导致在串行字段的后续插入操作中跳过整数序列。

权限

要执行更新，必须对指定的表(或视图)具有表级update权限，或者对指定的列具有列级update权限。
当更新一行中的所有字段时，请注意，列级特权覆盖GRANT命令中命名的所有表列;
表级权限涵盖所有表列，包括分配权限后添加的列。

- 用户必须对指定的表具有UPDATE权限，或者对更新字段列表中的所有列具有列级UPDATE权限。
- 用户必须对WHERE子句中的字段具有SELECT权限，无论这些字段是否要更新。
  如果这些字段包含在更新字段列表中，则必须同时拥有这些字段的SELECT和UPDATE权限。
  在下面的示例中，Name字段必须(至少)具有列级的SELECT权限:

```
UPDATE Sample.Employee (Salary) VALUES (1000000) WHERE Name='Smith, John'
```

在上面的示例中，Salary字段只需要列级UPDATE特权。

如果用户是该表的Owner(创建者)，则自动授予该用户对该表的所有特权。
否则，必须向用户授予该表的权限。
如果不这样做，将导致一个带有%msg的SQLCODE -99错误。
您可以通过调用%CHECKPRIV命令来确定当前用户是否具有适当的特权。
可以使用GRANT命令分配用户表权限。

当属性被定义为ReadOnly时，相应的表字段也被定义为ReadOnly。
只读字段只能使用InitialExpression或SqlComputed赋值。
尝试更新具有列级ReadOnly (SELECT或REFERENCES)权限的字段的值(即使是NULL值)将导致SQLCODE -138错误:无法为只读字段插入/更新值。
当您使用链接表向导链接一个表时，您可以选择将字段定义为只读。

源系统上的字段可能不是只读的，但是如果IRIS将链接表的字段定义为只读，那么尝试引用该字段的UPDATE将导致SQLCODE -138错误。

级安全

IRIS行级安全允许UPDATE修改任何安全允许它访问的行。
它允许更新行，即使更新创建的行安全性不允许随后访问。
为了确保更新不会阻止您对行的后续SELECT访问，建议通过具有WITH CHECK OPTION的视图执行update。

#SQL #Caché

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E5%85%AB%E5%8D%81%E5%9B%9B%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-update%EF%BC%88%E4%B8%89%EF%BC%89