文章
姚 鑫 · 十一月 15, 2021 阅读大约需 6 分钟

第七十七章 SQL命令 TRUNCATE TABLE

第七十七章 SQL命令 TRUNCATE TABLE

从表中删除所有数据并重置计数器。

大纲

```
TRUNCATE TABLE [restriction] tablename
```

参数

- restriction - 可选—以下限制关键字中的一个或多个，用空格隔开:%NOCHECK， %NOLOCK。
- tablename - 要从中删除所有行的表。
  还可以指定一个可更新视图，通过该视图可以删除表中的所有行。
  表名(或视图名)可以是限定的(schema.table)或非限定的(table)。
  使用模式搜索路径(如果提供的话)或默认模式名将非限定名称匹配到其模式。

描述

TRUNCATE TABLE命令从表中删除所有行，并重置所有表计数器。
可以直接截断表，也可以通过视图截断表。
通过视图截断表会受到删除要求和限制，如CREATE view中所述。

TRUNCATE
TABLE重置用于生成RowID字段、IDENTITY字段和SERIAL(%Library.Counter)字段连续整数值的内部计数器。
IRIS为插入到TRUNCATE表后的表中的第一行中的这些字段赋值为1。
对表的所有行执行DELETE操作不会重置这些内部计数器。

TRUNCATE TABLE重置用于在数据插入到流字段时生成流字段OID值的内部计数器。
对表的所有行执行DELETE操作不会重置此内部计数器。

TRUNCATE TABLE总是将%ROWCOUNT本地变量设置为-1;
它没有将%ROWCOUNT设置为删除的行数。

TRUNCATE TABLE不会重置ROWVERSION计数器。

TRUNCATE TABLE禁止提取基表触发器，否则在DELETE处理期间提取基表触发器。
因为TRUNCATE TABLE执行的是带有%NOTRIGGER行为的删除，用户必须被授予%NOTRIGGER权限(使用GRANT语句)才能运行TRUNCATE TABLE。
TRUNCATE TABLE的这方面在功能上是相同的:

```
DELETE %NOTRIGGER FROM tablename
```

注意:DELETE命令也可以用来删除表中的所有行。

DELETE提供了比TRUNCATE TABLE更多的功能，包括返回%ROWCOUNT中已删除的行数。
DELETE不会重置内部计数器。

TRUNCATE TABLE为从其他数据库软件迁移代码提供了兼容性。

截断一个表:

- 表必须存在于当前(或指定)命名空间中。
  如果无法找到指定的表，IRIS将发出SQLCODE -30错误。
- 即使没有定义触发器，用户也必须具有%NOTRIGGER管理权限。
  如果没有此权限，则会出现%msg User does not have %NOTRIGGER权限的SQLCODE -99错误。
- 用户必须对表具有DELETE权限。
  如果没有此权限，将导致带有%msg的SQLCODE -99。
  可以通过调用%CHECKPRIV命令来确定当前用户是否具有DELETE权限。
  可以通过调用$SYSTEM.SQL.Security.CheckPrivilege()方法来确定指定的用户是否具有DELETE权限。
- 该表不能定义为READONLY。
  试图编译引用只读表的TRUNCATE TABLE会导致SQLCODE -115错误。
  注意，这个错误现在是在编译时发出的，而不是只在执行时发生。
- 如果通过视图删除，视图必须是可更新的;
  不能定义为WITH READ ONLY。
  尝试这样做会导致SQLCODE -35错误。
- 所有的行必须是可删除的。
  默认情况下，如果不能删除一行或多行，则TRUNCATE TABLE操作失败，不会删除任何行。

如果表被其他进程以EXCLUSIVE模式或SHARE模式锁定，则TRUNCATE TABLE失败。
试图在一个锁定的表上执行TRUNCATE TABLE操作将导致SQLCODE -110错误，并带有%msg，如下所示:
MyStuff' on row with RowID = '3'(其中指定的RowID是表中的第一行)。

如果删除一行会违反外键引用完整性，那么TRUNCATE TABLE将失败。
未删除任何行，因此TRUNCATE TABLE发出SQLCODE -124错误。
这个默认行为是可以修改的，如下所述。

Fast Truncate

如果可能，SQL优化器将执行高效的Fast Truncate表操作。
Fast Truncate操作删除表的范围，而不是单独删除每条记录。
在可能的情况下，快速截断将自动应用。
当无法实现快速截断时，将执行标准的Truncate TABLE操作。

注意:如果没有删除行，或者使用Fast TRUNCATE删除行，则TRUNCATE TABLE不会初始化或设置%ROWID。
因此，应该避免在TRUNCATE表之后使用%ROWID值。

Fast Truncate 限制

快速截断可以应用于标准表或分片表。

不能应用快速截断:

- 如果用户无法获得表级锁(除非指定了%NOLOCK)。
- 如果表是外键约束的目标。
- 如果表包含带有指定LOCATION参数的流字段。

当所有流字段没有指定可选的LOCATION参数时，可以应用快速截断。

Atomicity

TRUNCATE TABLE不会在自动启动的事务中发生，因此不提供日志记录或回滚选项。

如果需要日志记录和回滚TRUNCATE TABLE选项，则必须显式指定START TRANSACTION，并以显式COMMIT或rollback结束。

这与SET TRANSACTION %COMMITMODE= NONE或0(没有自动事务)相同——调用TRUNCATE TABLE时不会启动任何事务。
失败的TRUNCATE TABLE操作可能会使数据库处于不一致的状态，一些行被删除，一些行没有被删除。
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。

分片表的TRUNCATE TABLE总是使用SET TRANSACTION %COMMITMODE NONE执行，即使用户显式地设置了SET TRANSACTION %COMMITMODE EXPLICIT。

限制参数

要使用constraint参数，必须拥有当前名称空间对应的admin-privilege。

指定约束参数限制处理如下:
- %NOCHECK - 禁止对引用被删除行的外键进行引用完整性检查。
- %NOLOCK - 抑制被删除行的行锁定。
这应该只在单个用户/进程更新数据库时使用。
如果不指定%NOLOCK，则快速截断将尝试获取表级锁。
如果TRUNCATE TABLE不能获得表级锁，它将执行一个标准的截断表，在表的每一行上获取行级锁。

可以以任何顺序指定多个限制参数。
多个参数由空格分隔。

如果在删除父记录时指定了约束参数，则在删除相应的子记录时将应用相同的约束参数。

TRUNCATE TABLE总是使用隐式的%NOTRIGGER行为执行删除操作，并且需要相应的admin-privilege。

参照完整性

IRIS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()。

在TRUNCATE TABLE操作期间，对于每个外键引用，都会在引用表中相应的行上获得一个共享锁。
这一行将被锁定，直到事务结束。
这确保了在可能的TRUNCATE表回滚之前不会更改引用的行。

事务锁

IRIS对TRUNCATE TABLE操作执行标准锁定。
唯一的字段值在当前事务期间被锁定。

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删除超过1000个惟一字段值，就会达到锁阈值，IRIS会自动将锁级别从惟一字段值锁提升到表锁。
这允许在事务期间进行大规模删除，而不会溢出锁表。

可以使用$SYSTEM.SQL.Util.GetOption(“ LockThreshold ”)方法确定当前系统范围的锁阈值。
这个系统范围的锁阈值是可配置的:
- 使用$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。

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。

Imported SQL代码

ImportDDL("IRIS")和Run()方法不支持TRUNCATE TABLE命令。
在这些方法导入的SQL代码文件中发现的TRUNCATE TABLE命令将被忽略。
这些导入方法确实支持DELETE命令。

示例

下面两个动态SQL示例比较了DELETE和TRUNCATE表。
每个示例都创建一个表，向表中插入行，删除表中的所有行，然后向现在为空的表中插入一行。

第一个示例使用DELETE删除表中的所有记录。
注意，DELETE不会重置RowID计数器:

```
ClassMethod TruncateTable()
{
    s tcreate = "CREATE TABLE SQLUser.MyStudents1 (StudentName VARCHAR(32),StudentDOB
 DATE)"
    s tinsert = "INSERT INTO SQLUser.MyStudents1 (StudentName,StudentDOB) "_
                "SELECT Name,DOB FROM Sample.Person WHERE Age <= '21'"
    s tinsert1 = "INSERT INTO SQLUser.MyStudents1 (StudentName,StudentDOB) VALUES ('B
ob Jones',60123)"
    s tdelete = "DELETE SQLUser.MyStudents1"
    s tStatement = ##class(%SQL.Statement).%New()
    s qStatus = tStatement.%Prepare(tcreate)
    if qStatus '= 1 {
        w "%Prepare failed:"
        d $System.Status.DisplayError(qStatus)
        q
    }
    s rset = tStatement.%Execute()
    w rset.%StatementTypeName,!

    n %ROWCOUNT,%ROWID
    s qStatus = tStatement.%Prepare(tinsert)
    if qStatus '= 1 {
        w "%Prepare failed:"
        d $System.Status.DisplayError(qStatus)
        q
    }
    s rset = tStatement.%Execute()
    w rset.%StatementTypeName," rowcount ",rset.%ROWCOUNT,!

    s qStatus = tStatement.%Prepare(tdelete)
```

```
    if qStatus '= 1 {
        w "%Prepare failed:"
        d $System.Status.DisplayError(qStatus)
        q
    }
    s rset = tStatement.%Execute()
    w rset.%StatementTypeName," rowcount ",rset.%ROWCOUNT,!

    s qStatus = tStatement.%Prepare(tinsert1)
    if qStatus '= 1 {
        w "%Prepare failed:"
        d $System.Status.DisplayError(qStatus)
        q
    }
    s rset = tStatement.%Execute()
    w rset.%StatementTypeName," rowcount ",rset.%ROWCOUNT," RowID ",rset.%ROWID,!
    &sql(DROP TABLE SQLUser.MyStudents1)
}


DHC-APP>d ##class(PHA.TEST.SQLCommand).TruncateTable()
CREATE TABLE
INSERT rowcount 41
DELETE rowcount 41
INSERT rowcount 1 RowID 42
```

第二个示例使用TRUNCATE TABLE删除表中的所有记录。
注意，%StatementTypeName对TRUNCATE表返回“ DELETE ”。
注意，TRUNCATE TABLE会重置RowID计数器:

```
ClassMethod TruncateTable1()
{
    s tcreate = "CREATE TABLE SQLUser.MyStudents2 (StudentName VARCHAR(32),StudentDOB
 DATE)"
    s tinsert = "INSERT INTO SQLUser.MyStudents2 (StudentName,StudentDOB) "_
    "SELECT Name,DOB FROM Sample.Person WHERE Age <= '21'"
    s tinsert1 = "INSERT INTO SQLUser.MyStudents2 (StudentName,StudentDOB) VALUES ('B
ob Jones',60123)"
    s ttrunc = "TRUNCATE TABLE SQLUser.MyStudents2"
    s tStatement = ##class(%SQL.Statement).%New()
    s qStatus = tStatement.%Prepare(tcreate)
    if qStatus '= 1 {
        w "%Prepare failed:"
        d $System.Status.DisplayError(qStatus)
        q
    }
    s rset = tStatement.%Execute()
    w rset.%StatementTypeName,!

    n %ROWCOUNT,%ROWID
    s qStatus = tStatement.%Prepare(tinsert)
    if qStatus '= 1 {
        w "%Prepare failed:"
        d $System.Status.DisplayError(qStatus)
        q
    }
    s rset = tStatement.%Execute()
```

```
    w rset.%StatementTypeName," rowcount ",rset.%ROWCOUNT,!

    s qStatus = tStatement.%Prepare(ttrunc)
    if qStatus '= 1 {
        w "%Prepare failed:"
        d $System.Status.DisplayError(qStatus)
        q
    }
    s rset = tStatement.%Execute()
    w rset.%StatementTypeName," (TRUNCATE TABLE) rowcount ",rset.%ROWCOUNT,!

    s qStatus = tStatement.%Prepare(tinsert1)
    if qStatus '= 1 {
        w "%Prepare failed:"
        d $System.Status.DisplayError(qStatus)
        q
    }
    s rset = tStatement.%Execute()
    w rset.%StatementTypeName," rowcount ",rset.%ROWCOUNT," RowID ",rset.%ROWID,!
    //&sql(DROP TABLE SQLUser.MyStudents2)
}
```

#SQL #Caché

源
URL:
https://cn.community.intersystems.com/post/%E7%AC%AC%E4%B8%83%E5%8D%81%E4%B8%83%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-truncate-table