文章
姚 鑫 · 九月 27, 2021 阅读大约需 13 分钟

第二十七章 SQL命令 DELETE（一）

第二十七章 SQL命令 DELETE（一）

从表中删除行。

大纲

```
DELETE [%keyword] [FROM] table-ref [[AS] t-alias]
     [FROM [optimize-option] select-table [[AS] t-alias]
               {,select-table2 [[AS] t-alias]} ]
     [WHERE condition-expression]

DELETE [%keyword] [FROM] table-ref [[AS] t-alias]
     [WHERE CURRENT OF cursor]
```

参数

- %keyword - 可选-以下一个或多个关键字选项，以空格分隔：%NOCHECK、%NOFPLAN、%NOINDEX、%NOJOURN、%NOLOCK、%NOTRIGGER、%PROFILE、%PROFILEALL。
- FROM table-ref - 要从中删除行的表。这不是FROM子句；它是一个FROM关键字，后跟一个表引用。(FROM关键字是可选的；table-ref是必需的。)表名(或视图名)可以是限定的(schema.table)，也可以是不限定的(Table)。使用架构搜索路径(如果提供)或默认架构名称将非限定名称与其架构匹配。可以指定可通过其删除表行的视图，而不是表引用，也可以指定括在圆括号中的子查询。与SELECT语句FROM子句不同，不能在此处指定Optimize-Option关键字。不能在此参数中指定表值函数或联接语法。
- FROM clause - 可选-FROM子句，在table-ref之后指定。此FROM可用于指定一个或多个选择表，用于选择要删除的行。可以将多个表指定为逗号分隔的列表或与ANSI联接关键字关联。可以指定表或视图的任意组合。如果在此处的两个选择表之间指定逗号， IRIS将对这两个表执行交叉联接，并从联接操作的结果表中检索数据。如果在此处的两个选择表之间指定ANSI联接关键字，则 IRIS将执行指定的联接操作。可以选择指定一个或多个OPTIMIZE-OPTION关键字来优化查询执行。可用选项为：%ALLINDEX、%FIRSTTABLE TABLE TABNAME、%FULL、%INORDER、%IGNOREINDICES、%NOFLATTEN、%NOMERGE、%NOSVSO、%NOTOPOPT、%NOUNIONOROPT、%PARALLEL和%STARTTABLE。
- AS t-alias - 可选-表或视图名称的别名。别名必须是有效的标识符。AS关键字是可选的。
- WHERE condition-expression - 可选-
  指定一个或多个布尔谓词，用于限制要删除的行。可以指定WHERE子句或WHERE CURRENT OF子句，但不能同时指定两者。如果未提供WHERE子句(或WHERE CURRENT OF子句)，则DELETE将从表中删除所有行。
- WHERE CURRENT OF cursor - 可选：仅嵌入式SQL-
  指定删除操作删除游标当前位置的记录。可以指定WHERE CURRENT OF子句或WHERE子句，但不能同时指定两者。如果未提供WHERE CURRENT OF子句(或WHERE子句)，则DELETE将从表中删除所有行。

描述

DELETE命令从满足指定条件的表中删除行。可以直接从表中删除行、通过视图删除或删除使用子查询选择的行。通过视图删除受要求和限制的约束，如创建视图中所述。

删除操作将%ROWCOUNT局部变量设置为已删除行数，并将%ROWID局部变量设置为已删除最后一行的RowID值。如果没有删除任何行，则%ROWCOUNT=0和%ROWID未定义或保持设置为其先前的值。

必须指定table-ref；table-ref前的from关键字是可选的。要从表中删除所有行，只需指定：

```
DELETE FROM tablename
```

或

```
DELETE tablename
```

这将从表中删除所有行数据，但不会重置RowID、Identity、流字段OID值和序列(%Library.Counter)字段计数器。TRUNCATE TABLE命令既删除表中的所有行数据，又重置这些计数器。默认情况下，DELETE FROM TABLENAME将拉取DELETE触发器；可以指定DELETE %NOTRIGGER FROM TABLENAME不拉取DELETE触发器。TRUNCATE TABLE不拉取删除触发器。

更常见的情况是，删除指定基于条件表达式的特定行(或多行)的删除。默认情况下，删除操作遍历表的所有行，并删除满足条件表达式的所有行。如果没有满足条件表达式的行，则DELETE成功完成，并设置SQLCODE=100(没有更多数据)。

可以指定WHERE子句或WHERE CURRENT OF子句(但不能同时指定两者)。如果使用WHERE CURRENT OF子句，删除操作将删除游标当前位置的记录。

默认情况下，DELETE是一个全有或全无事件：要么完全删除所有指定的行，要么不执行任何删除。IRIS设置状态变量SQLCODE，指示删除是成功还是失败。

要从表中删除行，请执行以下操作：

- 该表必须存在于当前(或指定的)命名空间中。如果找不到指定的表， IRIS将发出SQLCODE-30错误。
- 用户必须具有对指定表的删除权限。如果用户是表的所有者(创建者)，则会自动授予该用户对该表的删除权限。否则，必须授予用户对该表的删除权限。否则将导致SQLCODE-99错误，因为%msg用户‘ name ’没有该操作的特权。可以通过调用%CHECKPRIV命令来确定当前用户是否具有删除权限。可以使用GRANT命令将删除权限分配给指定表。
- 表不能被另一个进程以独占模式锁定。尝试从锁定表中删除行将导致SQLCODE-110错误，错误代码为%msg，如下所示：无法获取用于删除行ID为‘ 10 ’的行的表‘ Sample.Person ’的锁。请注意，只有当DELETE语句找到第一条要删除的记录，然后无法在超时期限内锁定它时，才会出现SQLCODE-110错误。
- 如果DELETE命令的WHERE子句指定了一个不存在的字段，则会发出SQLCODE-29。要如果该字段存在，但没有一个字段值满足DELETE命令的WHERE子句，则不会影响任何行，并发出SQLCODE 100(数据结束)。
- 不能将该表定义为READONLY。尝试编译引用只读表的删除会导致SQLCODE-115错误。请注意，此错误现在在编译时发出，而不是仅在执行时发出。
- 如果通过视图删除，则不能将该视图定义为只读。尝试这样做会导致SQLCODE-35错误。如果视图基于分割表，则不能通过使用CHECK OPTION定义的视图进行删除。尝试这样做会导致SQLCODE-35，其中不允许基于带有CHECK选项条件的切片表的视图(sample.myview)使用%msg INSERT/UPDATE/DELETE。同样，如果试图通过子查询进行删除，则子查询必须是可更新的；例如，以下子查询会导致SQLCODE-35错误：DELETE FROM (SELECT COUNT(*) FROM Sample.Person) AS x。
- 要删除的行必须存在。通常，尝试删除不存在的行会导致SQLCODE 100(没有更多数据)，因为找不到指定的行。但是，在极少数情况下，DELETE WITH%NOLOCK会找到要删除的行，但随后该行会被另一个进程立即删除；这种情况会导致SQLCODE-106错误。此错误的%msg列出了表名和RowID。
- 指定要删除的所有行都必须可供删除。默认情况下，如果无法删除一行或多行，则删除操作将失败，并且不会删除任何行。如果要删除的行已被另一个并发进程锁定，则DELETE会发出SQLCODE-110错误。如果删除指定行之一会违反外键引用完整性(并且未指定%NOCHECK)，则删除操作将发出SQLCODE-124错误。此默认行为是可修改的，如下所述。

- 某些%SYS命名空间系统提供的工具受到保护，不会被删除。例如，DELETE FROM Security.Users 不能用于DELETE SYSTEM、PUBLIC或UNKNOWnUser。尝试这样做会导致SQLCODE-134错误。

From语法

一个DELETE命令可以包含两个指定表的FROM关键字。From的这两种用法从根本上说是不同的：
- FROM BEFORE TABLE-REF指定要从中删除行的表(或视图)。它是FROM关键字，而不是FROM子句。只能指定一个表。不能指定联接语法或优化选项关键字。FROM关键字本身是可选的；table-ref是必需的。
- FROM AFTER TABLE-REF是一个可选的FROM子句，可用于确定应该删除哪些行。它可以指定一个或多个表。它支持SELECT语句可用的所有FROM子句语法，包括联接语法和优化选项关键字。此FROM子句通常(但不总是)与WHERE子句一起使用。

因此，以下任何一种都是有效的语法形式：

```
DELETE FROM table WHERE ...
DELETE table WHERE ...
DELETE FROM table FROM table2 WHERE ...
DELETE table FROM table2 WHERE ...
```

此语法以与Transact-SQL兼容的方式支持复杂的选择条件。

下面的示例显示如何使用这两个FROM关键字。它从Employees表中删除那些记录，在Replrees表中也可以找到相同的EmpId：

```
DELETE FROM Employees AS Emp
       FROM Retirees AS Rt
       WHERE Emp.EmpId = Rt.EmpId
```

如果两个FROM关键字引用了同一个表，则这些引用可以是对同一个表的引用，也可以是对该表的两个实例的联接。这取决于如何使用表别名：

- 如果两个表引用都没有别名，则两者都引用同一个表：

```
DELETE FROM table1 FROM table1,table2   /* join of 2 tables */
```

- 如果两个表引用具有相同的别名，则两者引用同一个表：

```
DELETE FROM table1 AS x FROM table1 AS x,table2   /* join of 2 tables */
```

- 如果两个表引用都有别名，并且别名不同，则 IRIS将执行表的两个实例的联接：

```
DELETE FROM table1 AS x FROM table1 AS y,table2   /* join of 3 tables */
```

- 如果第一个表引用具有别名，而第二个表引用没有别名，则 IRIS将执行表的两个实例的联接：

```
DELETE FROM table1 AS x FROM table1,table2   /* join of 3 tables */
```

- 如果第一个表引用没有别名，而第二个表引用具有别名的表只有一个引用，则这两个表都引用同一个表，并且此表具有指定的别名：

```
DELETE FROM table1 FROM table1 AS x,table2   /* join of 2 tables */
```

- 如果第一个表引用没有别名，而第二个表引用有多个对表的引用，则IRIS会将每个别名实例视为单独的表，并对这些表执行联接：

```
DELETE FROM table1 FROM table1,table1 AS x,table2   /* join of 3 tables */
DELETE FROM table1 FROM table1 AS x,table1 AS y,table2   /* join of 4 tables */
```

%Keyword 选项

指定%Keyword参数将按如下方式限制处理：
- %NOCHECK-禁止对引用要删除的行的外键进行参照完整性检查。用户必须具有当前命名空间的相应%NOCHECK管理权限才能应用此限制。否则将导致SQLCODE-99错误，因为%msg用户‘ name ’没有%NOCHECK权限。
- %NOFPLAN-忽略此操作的冻结计划(如果有)；该操作将生成新的查询计划。冻结的计划将保留，但不会使用。
- %NOINDEX -禁止删除要删除行的所有索引中的索引项。使用时应格外小心，因为它会在表索引中留下孤立值。用户必须具有当前命名空间的相应%noindex管理权限才能应用此限制。否则将导致SQLCODE-99错误，因为%msg用户‘ name ’没有%noindex权限。
- %NOJOURN-在删除操作期间禁止日志记录。任何行中所做的任何更改都不会被记录下来，包括拉出的任何触发器。如果在使用%NOJOURN的语句之后执行ROLLBACK，则不会回滚该语句所做的更改。
- %NOLOCK-禁止对要删除的行进行行锁定。这应该仅在单个用户/进程更新数据库时使用。用户必须具有当前命名空间的相应%NOLOCK管理权限才能应用此限制。否则将导致SQLCODE-99错误，因为%msg用户‘ name ’没有%NOLOCK权限。
- %NOTRIGGER-禁止拉取基表触发器，否则将在删除处理期间拉取这些触发器。用户必须具有当前命名空间的相应%NOTRIGGER管理权限才能应用此限制。否则将导致SQLCODE-99错误，因为%msg用户‘ name ’没有%NOTRIGGER权限。
- %PROFILE或%PROFILEALL-如果指定了其中一个关键字指令，则生成SQLStats收集代码。这与启用PTools时生成的代码相同。不同之处在于，SQLStats收集代码只为该特定语句生成。正在编译的例程/类中的所有其他SQL语句将生成代码，就像PTools已关闭一样。这使用户能够分析/检查应用程序中的特定问题SQL语句，而无需收集未被调查的SQL语句的无关统计信息。

%PROFILE收集主查询模块的SQLStat。%PROFILEALL收集主查询模块及其所有子查询模块的SQLStat。

如果在删除父记录时指定%KEYWORD参数，则删除相应的子记录时也会应用相同的%KEYWORD参数。

参照完整性

如果不指定%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；默认情况下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，请调用$SYSTEM.SQL.CurrentSettings()。

在删除操作期间，对于每个外键引用，都会在被引用表中的相应行上获取一个共享锁。此行将被锁定，直到事务结束。这可确保引用的行在可能回滚删除之前不会更改。

如果将一系列外键引用定义为级联，则删除操作可能会导致循环引用。
IRIS防止DELETE与级联引用操作一起执行循环引用循环递归。 IRIS在返回到原始表时结束级联序列。

如果使用%NOLOCK对使用CASCADE、SET NULL或SET DEFAULT定义的外键字段执行DELETE操作，则也会使用%NOLOCK执行相应的更改外键表的引用操作。

原子性

默认情况下，DELETE、UPDATE、INSERT和TRUNCATE TABLE是原子操作。删除要么成功完成，要么回滚整个操作。如果无法删除任何指定的行，则不会删除任何指定的行，并且数据库将恢复到发出DELETE之前的状态。

可以通过调用SET TRANSACTION %COMMITMODE来修改SQL中当前进程的此默认值。可以通过调用SetOption()方法在ObjectScript中修改当前进程的此默认值，如下所示 SET status=$SYSTEM.SQL.Util.SetOption("AutoCommit",intval,.oldval)。以下整型整数选项可用：

- 1或隐式(自动提交打开)-如上所述的默认行为。每次删除都构成一个单独的事务。
- 2或EXPLICIT(AUTOCOMMIT OFF)-如果没有正在进行的事务，则DELETE会自动启动一个事务，但必须显式提交或回滚才能结束该事务。在显式模式下，每个事务的数据库操作数由用户定义。
- 0或None(无自动事务)-调用DELETE时不会启动任何事务。失败的删除操作可能会使数据库处于不一致的状态，其中一些指定的行已删除，另一些未删除。要在此模式下提供事务支持，必须使用START TRANSACTION来启动事务，并使用COMMIT或ROLLBACK来结束事务。

切片表始终处于非自动事务模式，这意味着对切片表的所有插入、更新和删除都在事务范围之外执行。

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

事务锁定

如果未指定%NOLOCK，系统将自动对INSERT、UPDATE和DELETE操作执行标准记录锁定。在当前事务期间锁定每个受影响的记录(行)。

默认锁定阈值是每个表1000个锁。这意味着如果在事务期间从表中删除1000条以上的记录，就会达到锁定阈值，IRIS会自动将锁定级别从记录锁升级为表锁。这允许在事务期间进行大规模删除，而不会使锁表溢出。

IRIS应用以下两种锁升级策略之一：
- “ E ”-类型的锁升级：如果满足以下条件， IRIS将使用这种类型的锁升级：(1)类使用%Storage.Persistent(可以从管理门户SQL架构显示中的目录详细信息确定)。(2)该类不指定IDKey索引，或者指定单属性IDKey索引。
- 传统的SQL锁升级：类不使用 “ E ” 类型锁升级的最可能原因是多属性IDKey索引的存在。在这种情况下，每个%Save都会递增锁定计数器。这意味着如果在一个事务内对单个对象执行1001次保存， IRIS将尝试升级锁。

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

- 使用$SYSTEM.SQL.Util.SetOption(“ LockThreshold ”)方法。
- 使用管理门户：依次选择系统管理、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。默认值为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，当升级到表锁的尝试与持有该表中的记录锁的另一个进程冲突时，可能会发生死锁情况。有几种可能的策略可以避免这种情况：(1)提高锁升级阈值，使锁升级不太可能在事务内发生。(2)大幅降低锁升级阈值，使锁升级几乎立即发生，从而降低其他进程锁定同一表中记录的机会。(3)在事务期间应用表锁，不要执行记录锁。这可以在事务开始时完成，方法是指定LOCK TABLE，然后解锁TABLE(不使用IMMEDIATE关键字，这样表锁将一直持续到事务结束)，然后使用%NOLOCK选项执行删除。

自动锁升级旨在防止锁表溢出。但是，如果执行的删除次数太多，以至于出<LOCKTABLEFULL>
错误，则DELETE会发出SQLCODE-110错误。

#SQL #Caché

源URL:
https://cn.community.intersystems.com/post/%E7%AC%AC%E4%BA%8C%E5%8D%81%E4%B8%83%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-delete%EF%BC%88%E4%B8%80%EF%BC%89