文章
姚 鑫 · 三月 11, 2021 阅读大约需 10 分钟

# 第八章 SQL修改数据库

# 第八章 SQL修改数据库

可以对现有的表使用SQL语句，也可以对相应的持久化类使用ObjectScript操作来修改InterSystems IRIS®数据平台数据库的内容。
不能修改定义为只读的持久类(表)。

使用SQL命令为维护数据的完整性提供了自动支持。
SQL命令是一个原子操作(全部或没有)。
如果表上定义了索引，SQL将自动更新它们以反映更改。
如果定义了任何数据或引用完整性约束，SQL将自动执行它们。
如果有任何已定义的触发器，执行这些操作将拉动相应的触发器。

## 插入数据

可以使用SQL语句或设置和保存持久化类属性将数据插入表中。

### 使用SQL插入数据

INSERT语句将一条新记录插入SQL表中。
可以插入一条记录或多条记录。

下面的示例插入一条记录。
它是插入单个记录的几种可用语法形式之一:

```
INSERT INTO MyApp.Person
   (Name,HairColor)
   VALUES ('Fred Rogers','Black')
```

以下示例通过查询现有表中的数据插入多条记录:

```
INSERT INTO MyApp.Person
   (Name,HairColor)
    SELECT Name,Haircolor FROM Sample.Person WHERE Haircolor IS NOT NULL
```

还可以发出INSERT或UPDATE语句。
如果SQL表中不存在新记录，则该语句将该记录插入该SQL表中。
如果记录存在，则该语句使用提供的字段值更新记录数据。

### 使用对象属性插入数据

可以使用ObjectScript插入一条或多条数据记录。
创建一个现有持久化类的实例，设置一个或多个属性值，然后使用%Save()插入数据记录:

下面的例子插入一条记录:

```
SET oref=##class(MyApp.Person).%New()
SET oref.Name="Fred Rogers"
SET oref.HairColor="Black"
DO oref.%Save()
```

下面的例子插入多条记录:

```
SET nom=$LISTBUILD("Fred Rogers","Fred Astare","Fred Flintstone")
SET hair=$LISTBUILD("Black","Light Brown","Dark Brown")
FOR i=1:1:$LISTLENGTH(nom) {
    SET oref=##class(MyApp.Person).%New()
    SET oref.Name=$LIST(nom,i)
    SET oref.HairColor=$LIST(hair,i)
    SET status = oref.%Save() }
```

# UPDATE语句

UPDATE语句修改SQL表中的一条或多条现有记录中的值:

```
UPDATE????SQL??????????????:
```

# 在插入或更新时计算字段值

在定义计算字段时，可以指定ObjectScript代码来计算该字段的数据值。
可以在插入、更新行、插入和更新行或查询行时计算此数据值。
下表显示了每种计算操作类型所需的关键字以及字段/属性定义示例:

- 只在插入时计算
  - SQL DDL COMPUTECODE关键字Birthday VARCHAR(50) COMPUTECODE {SET {Birthday}=$PIECE($ZDATE({DOB}，9)，"，")"_changed: "$ZTIMESTAMP}
  - 持久化类定义SqlComputeCode和SqlComputeCode关键字属性生日为%String(MAXLEN = 50) [SqlComputeCode = {SET {Birthday}=$PIECE($ZDATE({DOB}，9)，"，") "_changed: "$ZTIMESTAMP}， SqlComputed];
- 只在更新时计算
  - SQL DDL DEFAULT, COMPUTECODE，和COMPUTEONCHANGE关键字Birthday VARCHAR(50) DEFAULT '' COMPUTECODE {SET {Birthday}=$PIECE($ZDATE({DOB}，9)，"，")"_changed: "$ZTIMESTAMP} COMPUTEONCHANGE (DOB)
- 在插入和更新上都进行计算
  - SQL DDL COMPUTECODE和COMPUTEONCHANGE关键字Birthday VARCHAR(50) COMPUTECODE {SET {Birthday}=$PIECE($ZDATE({DOB}，9)，"，")"_changed: "$ZTIMESTAMP} COMPUTEONCHANGE (DOB)
  - 持久化类定义SqlComputeCode, SqlComputed, and SqlComputeOnChange属性关键字属性Birthday As %String(MAXLEN = 50) [SqlComputeCode = {SET {Birthday}=$PIECE($ZDATE({DOB}，9)，"，") "_changed: "$ZTIMESTAMP}， SqlComputed, SqlComputeOnChange = DOB];
- 计算对查询

- SQL DDL COMPUTECODE和计算或瞬态关键字Birthday VARCHAR(50) COMPUTECODE {SET {Birthday}=$PIECE($ZDATE({DOB}，9)，"，")"_changed: "$ZTIMESTAMP}计算
- 持久类定义SqlComputeCode, SqlComputed, and calculate或瞬态属性关键字属性Birthday为%String(MAXLEN = 50) [SqlComputeCode = {SET {Birthday}=$PIECE($ZDATE({DOB}，9)，"，") "_changed: "$ZTIMESTAMP}， SqlComputed, calculate];

DDL DEFAULT关键字在插入时优先于计算数据值。
DEFAULT必须接受一个数据值，例如空字符串;
不能为空。
在持久类定义中，InitialExpression属性关键字在插入时不会覆盖SqlComputed数据值。

DDL COMPUTEONCHANGE关键字可以使用单个字段名，也可以使用逗号分隔的字段名列表。
这些字段名指定了哪些字段更新时会触发对该字段的计算;
列出的字段名称必须存在于表中，但它们不必出现在计算代码中。
必须指定实际的字段名;
不能指定星号语法。

在修改记录时，可以使用ON UPDATE关键字短语将字段设置为文字或系统变量(如当前时间戳)，而不是使用COMPUTECODE和COMPUTEONCHANGE。
ON UPDATE短语同时修饰INSERT和UPDATE;
若要只在更新时修改，请使用默认短语和更新短语。

每次查询访问该字段时，DDL计算或TRANSIENT关键字都会计算一个数据值。
该字段不需要在选择列表中指定。
例如，SELECT Name FROM MyTable WHERE LENGTH(Birthday)=36在计算条件表达式之前计算生日字段。
管理门户Open Table选项执行一个查询，因此计算计算的和临时的数据值。

计算字段限制:
- 不更新的更新:为记录中的字段提供与它们之前的值相同的值的更新实际上并不更新记录。
如果没有对记录执行真正的更新，则不会调用COMPUTEONCHANGE。
即使没有对一条记录执行真正的更新，也会在更新操作上调用ON UPDATE。
如果希望在更新时总是重新计算已计算字段，而不管记录是否实际更新，请使用更新触发器。
- 用户为计算字段指定的显式值:
- INSERT:在INSERT时，您总是可以向COMPUTECODE、DEFAULT或On UPDATE字段提供显式的值。
InterSystems SQL总是采用显式的值，而不是生成的值。
- 更新COMPUTEONCHANGE:更新操作可以为COMPUTEONCHANGE字段提供显式的值。
InterSystems SQL总是采用显式的值，而不是计算的值。
- 更新时更新:更新操作不能为ON UPDATE字段提供显式值。
InterSystems SQL忽略用户提供的值，并接受ON UPDATE生成的值。
但是，InterSystems
SQL确实会对显式值执行字段验证，例如，如果提供的值大于最大数据大小，就会生成SQLCODE -104错误。
- 计算或暂态:插入或更新操作不能为计算或暂态字段提供显式值，因为计算或暂态字段不存储数据。
但是，InterSystems
SQL确实会对显式值执行字段验证，例如，如果提供的值大于最大数据大小，就会生成SQLCODE -104错误。

# 删除语句

DELETE语句从SQL表中删除一条或多条现有记录:

```
DELETE FROM MyApp.Person
    WHERE HairColor = 'Aqua'
```

可以执行TRUNCATE TABLE命令删除表中的所有记录。
还可以使用delete删除表中的所有记录。

DELETE(默认情况下)提取删除触发器;
TRUNCATE TABLE不拉出删除触发器。
使用DELETE删除所有记录不会重置表计数器;
TRUNCATE TABLE重置这些计数器。

# 事务处理

事务是一系列插入、更新、删除、插入或更新以及截断表数据修改语句，它们组成单个工作单元。

SET TRANSACTION命令用于设置当前进程的事务参数。
还可以使用START TRANSACTION命令设置相同的参数。
这些事务参数在多个事务中继续有效，直到显式更改为止。

START TRANSACTION命令显式地启动事务。
这个命令通常是可选的;
如果事务%COMMITMODE是隐式或显式的，事务从第一个数据库修改操作自动开始。
如果事务%COMMITMODE为NONE，则必须显式指定START transaction来启动事务处理。

如果事务成功，提交其更改可以是隐式(自动)或显式的;
%COMMITMODE值决定是否需要显式地使用COMMIT语句来永久地将数据修改添加到数据库并释放资源。

如果事务失败，可以使用ROLLBACK语句撤消其数据修改，这样这些数据就不会进入数据库。

注意:通过管理门户执行SQL查询接口运行SQL时，不支持SQL事务语句。
这个接口旨在作为开发SQL代码的测试环境，而不是用于修改实际数据。

## 事务和保存点

在InterSystems
SQL中，可以执行两种事务处理：完整事务处理和使用保存点的事务处理。通过完整的事务处理，事务将从START TRANSACTION语句（显式或隐式）开始，一直持续到COMMIT语句（显式或隐式）结束事务并提交所有工作，或者ROLLBACK语句反转事务期间完成的所有工作。

通过保存点，InterSystems SQL支持事务中的级别。可以使用START TRANSACTION语句（显式或隐式）开始事务。然后，在事务期间，可以使用SAVEPOINT在程序中指定一个或多个命名保存点。可以在一个事务中最多指定255个命名保存点。添加一个保存点会增加$TLEVEL事务级别计数器。
- COMMIT提交事务期间执行的所有工作。保存点将被忽略。
- ROLLBACK将回滚事务期间执行的所有工作。保存点将被忽略。
- ROLLBACK TO SAVEPOINT点名将回滚自点名指定的SAVEPOINT以来执行的所有工作，并以适当数量的保存点级别将内部事务级别计数器递减。例如，如果建立了两个保存点svpt1和svpt2，然后回滚到svpt1，则ROLLBACK TO SAVEPOINT svpt1会反转自svpt1以来所做的工作，在这种情况下，将事务级别计数器减2。

## 非事务操作

当事务生效时，以下操作不包括在事务中，因此无法回滚：
- IDKey计数器增量不是事务操作。IDKey由$INCREMENT(或$SEQUENCE)自动生成，它维护独立于SQL事务的计数。例如，如果插入IDKey为17、18和19的记录，然后回滚此插入，则下一条要插入的记录的IDKey将为20。
- 缓存查询的创建、修改和清除不是事务操作。因此，如果在事务期间清除高速缓存的查询，然后回滚该事务，则在回滚操作之后，高速缓存的查询将保持清除状态(不会恢复)。
- 事务内发生的DDL操作或调谐表操作可以创建和运行临时例程。此临时例程被视为与缓存查询相同。也就是说，临时例程的创建、编译和删除不被视为事务的一部分。临时例程的执行被认为是事务的一部分。

## 事务锁

**事务使用锁来保护唯一的数据值。例如，如果进程删除了唯一的数据值，则该值在事务持续时间内被锁定。因此，在第一个事务完成之前，另一个进程无法使用相同的唯一数据值插入记录。这可以防止回滚导致具有唯一性约束的字段出现重复值。这些锁由INSERT、UPDATE、INSERT或UPDATE和DELETE语句自动应用，除非该语句包含%NOLOCK限制参数。**

## 事务大小限制

除了日记文件的空间可用性外，可以在事务中指定的操作数量没有限制。锁表的大小通常不会施加限制，因为InterSystems IRIS提供自动锁升级。

每个表有1000个锁的默认锁阈值。对于当前事务，一个表可以有1000个唯一的数据值锁。第100个锁定操作在事务持续时间内将该表的锁定升级为表锁。

此锁定阈值可使用以下任一选项进行配置：
- 调用$SYSTEM.SQL.SetLockThreshold()方法。此方法更改当前系统范围的值和配置文件设置。要确定当前的锁升级阈值，请使用$SYSTEM.SQL.GetLockThreshold()方法。
- 转到管理门户。从系统管理中，依次选择配置、SQL和对象设置、SQL。在此屏幕上，可以查看和编辑锁定阈值的当前设置。

可以终止的子节点(子表)的数量没有限制。所有子节点终止都被记录下来，因此可以回滚。

## 读取未提交的数据

可以通过为发出查询的进程设置SET TRANSACTION或START TRANSACTION来指定读取隔离级别。

- 提交未提交的隔离级别：对于其他用户进行查询（只读）访问，可以看到未提交的对数据的插入，更新和删除。如果未指定任何事务，则为默认设置。
- 已验证隔离级别：可供其他用户以查询（只读）访问的方式看到未提交的对数据的插入，更新和删除。提供对查询条件所使用并由查询显示的数据的重新检查。
- 读取已提交的隔离级别：未提交的插入和更新对数据所做的更改未显示在查询结果集中。查询结果集仅包含已提交的插入和更新。但是，未提交的删除对数据所做的更改将显示在查询结果集中。

不管当前的隔离级别如何，以下SELECT命令子句始终返回未提交的数据：聚合函数，DISTINCT子句，GROUP BY子句或带有%NOLOCK关键字的SELECT。

## ObjectScript事务命令

ObjectScript和SQL事务命令是完全兼容和可互换的，但以下情况除外：

**如果没有当前事务，则ObjectScript TSTART和SQL START TRANSACTION都将启动一个事务。但是，START TRANSACTION不支持嵌套事务。因此，如果需要（或可能需要）嵌套事务，则最好使用TSTART启动事务。如果需要与SQL标准兼容，请使用START TRANSACTION。**

ObjectScript事务处理为嵌套事务提供了有限的支持。 SQL事务处理为事务中的保存点提供支持。

#SQL #Caché #InterSystems IRIS #InterSystems IRIS for Health

---

**源URL:**
https://cn.community.intersystems.com/post/%E7%AC%AC%E5%85%AB%E7%AB%A0-sql%E4%BF%AE%E6%94%B9%E6%95%B0%E6%8D%AE%E5%BA%93