文章
姚 鑫 · 十一月 21, 2021 阅读大约需 9 分钟

第八十三章 SQL命令 UPDATE（二）

第八十三章 SQL命令 UPDATE（二）

显示到逻辑数据转换

数据以逻辑模式格式存储。
例如，日期存储为整数天数，时间存储为从午夜开始的秒数，%List存储为编码字符串。
大多数其他数据，如字符串和数字，不需要转换;
无论当前模式如何，它们都以相同的格式输入、更新和存储。

对于已转换的数据，可以在LOGICAL模式(默认)中更新数据值，或者通过指定选择模式，使用更易于阅读的格式(DISPLAY模式或ODBC模式)更新数据值。
例如，通过指定选择模式，可以以DISPLAY格式(例如2/22/2018)、ODBC格式(例如2018-02-22)或逻辑格式(例如64701)更新日期。
对于某些数据类型，还可以在ODBC或DISPLAY选择模式下以LOGICAL格式指定数据。

列表结构

IRIS支持列表结构数据类型%list(数据类型类%Library.List)。
这是一种压缩的二进制格式，并不映射到 SQL的相应本机数据类型。
它对应的数据类型为VARBINARY，默认MAXLEN为32749。
因此，动态SQL不能使用UPDATE或INSERT来设置类型为%List的属性值。

流值

可以按照如下方法更新流字段中的数据值:
- 对于任何表:字符串字面值或包含字符串字面值的主机变量，例如:

```
SET literal="update stream string value"
   //do the update; use a string
&sql(UPDATE MyStreamTable SET MyStreamField = :literal WHERE %ID=21)
```

- 对于非分片表:流对象的对象引用(OREF)。
  IRIS打开这个对象并复制它的内容，更新stream字段。
  例如:

```
SET oref=##class(%Stream.GlobalCharacter).%New()
DO oref.Write("Update stream string value non-shard 1")
   //do the update; use an actual OREF
&sql(UPDATE MyStreamTable SET MyStreamField = :oref WHERE %ID=22)
```

或流的OREF的字符串版本，例如:

```
SET oref=##class(%Stream.GlobalCharacter).%New()
DO oref.Write("Update stream string value non-shard 2")
   //next line converts OREF to a string OREF
set string=oref_""
   //do the update
&sql(UPDATE MyStreamTable SET MyStreamField = :string WHERE %ID=23)
```

- 对于分片表:使用存储在^IRIS.Stream中的临时流对象的对象ID (OID)。
  碎片全球:

```
SET clob=##class(%Stream.GlobalCharacter).%New("Shard")
DO clob.Write("Update sharded table stream string value")
SET sc=clob.%Save() // Handle $$$ISERR(sc)
  set ClobOid=clob.%Oid()
//do the update
&sql(UPDATE MyStreamTable SET MyStreamField = :ClobOid WHERE %ID=24)
```

不能使用流字段的内容更新非流字段。
这将导致一个SQLCODE -303错误: “ 不支持在UPDATE赋值中隐式地将流值转换为非流字段 ” 。
要用Stream数据更新字符串字段，必须首先使用SUBSTRING函数将Stream数据的前n个字符转换为字符串，如下面的示例所示:

```
UPDATE MyTable
     SET MyStringField=SUBSTRING(MyStreamField,1,2000)
```

计算字段

用COMPUTECODE定义的字段可以作为UPDATE操作的一部分重新计算它的值，如下所示:

- COMPUTECODE:值在INSERT时计算并存储，在UPDATE时不更改值。
- 带有COMPUTEONCHANGE的COMPUTECODE:值在INSERT时计算并存储，在UPDATE时重新计算并存储。
- COMPUTECODE with DEFAULT and COMPUTEONCHANGE:默认值在INSERT时存储，值在UPDATE时计算并存储。
  如果计算代码包含一个编程错误(例如，除以0)，UPDATE操作将失败，并出现SQLCODE -415错误。
- COMPUTECODE WITH COMPUTECODE WITH COMPUTED或TRANSPENT：不能更新此字段的值，因为没有存储值。查询时会计算该值。但是，如果尝试更新计算字段中的值， IRIS会对提供的值执行验证，如果值无效，则会发出错误。如果该值有效，则IRIS不执行更新操作，不发出SQLCODE错误，并递增ROWCOUNT。

当没有实际更新发生时，COMPUTEONCHANGE计算字段不会重新计算:当update操作的新字段值与之前的字段值相同时。

在大多数情况下，将计算字段定义为只读。
这防止更新操作直接更改一个值，该值是涉及其他字段值的计算结果。
在本例中，试图使用UPDATE覆盖计算字段的值将导致SQLCODE -138错误。

但是，可能希望修改一个计算字段值，以反映对一个(或多个)源字段值的更新。
可以通过使用更新触发器来实现这一点，该更新触发器在您更新了指定的源字段之后重新计算计算过的字段值。
例如，对Salary数据字段的更新可能触发重新计算Bonus computed字段的触发器。
这个更新触发器重新计算Bonus并成功完成，即使Bonus是一个只读字段。

可以使用CREATE TABLE ON
UPDATE关键字短语来定义一个字段，当记录被更新时，该字段被设置为文字或系统变量(例如当前时间戳)。

%SerialObject属性

当更新%SerialObject中的数据时，必须更新引用嵌入%SerialObject的表(持久化类);
不能直接更新%SerialObject。
从引用表中，可以:

- 使用引用字段将多个%SerialObject属性的值更新为%List结构。
  例如,如果持久化类有一个属性PAddress引用一个串行对象包含属性,城市和国家(依次),SET PAddress=$LISTBUILD('123 Main St.','Newtown','USA') or (PAddress) VALUES ($LISTBUILD('123 Main St.','Newtown','USA')) or (PAddress) VALUES (:vallist)。
  %List必须包含串行对象(或占位符逗号)的属性值，其顺序与串行对象中指定的属性的顺序一致。

此类型的更新可能不会执行%SerialObject属性值的验证。因此，强烈建议在使用%List结构更新%SerialObject属性值之后，使用$SYSTEM.SQL.Schema.ValidateTable()方法执行表数据验证。

- 使用下划线语法以任意顺序更新单个%SerialObject属性的值。例如，如果持久类的特性PAddress引用了包含特性Street、City和Country的序列对象，则可以更新集合PAddress_City= ‘ Newtown ’ ，PAddress_Street= ‘ 123 Main St. ’ ，PAddress_Country= ‘ USA ’ 。

此类型的更新执行%SerialObject属性值的验证。

FROM子句

UPDATE命令可能没有FROM关键字。它可以简单地指定要更新的表(或视图)，并使用WHERE子句选择要更新的行。

但是，还可以在value-assignment-语句之后包括一个可选的FROM子句。此FROM子句指定用于确定要更新哪些记录的一个或多个表。FROM子句通常(但并非总是)与涉及多个表的WHERE子句一起使用。FROM子句可以很复杂，并且可以包括ANSI联接语法。UPDATE FROM子句允许SELECT FROM子句中支持的任何语法。此UPDATE FROM子句提供与Transact-SQL的功能兼容性。

以下示例显示如何使用此FROM子句。它更新Employees表中的那些记录，其中也可以在Requirees表中找到相同的EmpId：

```
UPDATE Employees AS Emp
     SET retired='Yes'
     FROM Retirees AS Rt
     WHERE Emp.EmpId = Rt.EmpId
```

如果UPDATE TABLE-REF和FROM子句引用同一个表，则这些引用可能是引用同一个表，也可能是引用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：

```
UPDATE table1 value-assignment 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：

```
UPDATE table1 AS x value-
```

```
assignment 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：

```
  UPDATE table1 AS x value-
assignment 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：

```
  UPDATE table1 AS x value-assignment 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：

```
  UPDATE table1 value-assignment 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：

```
  UPDATE table1 value-
assignment FROM table1,table1 AS x,table2        /* join of 3 tables */
  UPDATE table1 value-
assignment FROM table1 AS x,table1 AS y,table2   /* join of 4 tables */
```

%Keyword 参数

指定%Keyword参数将按如下方式限制处理：
- %NOCHECK-不执行唯一值检查和外键引用完整性检查。也不执行针对数据类型、最大长度、数据约束和其他验证条件的列数据验证。通过视图执行更新时，不执行视图的WITH CHECK选项验证。

注意：由于使用%NOCHECK可能导致无效数据，因此只有在从可靠的数据源执行批量插入或更新时才应使用此%关键字参数。

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。

如果希望在指定%NOCHECK时阻止导致非唯一数据值的更新，请在更新之前执行EXISTS检查。

如果只希望禁用外键引用完整性检查，请使用$SYSTEM.SQL.Util.SetOption(“ FilerRefIntegrity ”)方法，而不是指定%NOCHECK。或者，可以使用NOCHECK关键字定义外键，这样就永远不会执行外键引用完整性检查。

- %NOFPLAN - FROM子句语法仅:此操作忽略冻结的计划(如果有);
  该操作将生成一个新的查询计划。
  冻结的计划被保留，但不使用。
- %NOINDEX -在UPDATE处理期间没有设置索引映射。
  用户必须对当前名称空间具有相应的%NOINDEX管理权限才能应用此限制。
  如果不这样做，会出现一个带有%msg的SQLCODE -99错误，用户 “ name ” 没有%NOINDEX权限。
- %NOJOURN -在更新操作期间抑制日志记录。
  任何行中所做的更改都不会被记录到日志中，包括任何被拉出的触发器。
  如果在带有%NOJOURN的语句之后执行ROLLBACK，则该语句所做的更改将不会回滚。
- %NOLOCK -在UPDATE时未锁定行。
  这应该只在单个用户/进程更新数据库时使用。

用户必须拥有当前命名空间相应的%NOLOCK管理权限才能应用此限制。
如果不这样做，将导致%msg的SQLCODE -99错误，用户 “ name ” 没有%NOLOCK权限。

- %NOTRIGGER—在UPDATE处理过程中不会提取基表触发器。
  BEFORE和AFTER触发器都不执行。
  用户必须对当前命名空间具有相应的%NOTRIGGER管理权限才能应用此限制。
  如果不这样做，将导致%msg的SQLCODE -99错误。User 'name' does not have %NOTRIGGER privileges。
- %PROFILE或%PROFILEALL—如果指定了这些关键字指令中的一个，将生成SQLStats收集代码。
  这与打开PTools时生成的代码相同。
  不同之处在于，SQLStats收集代码仅为该特定语句生成。
  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被关闭一样。
  这使用户能够分析/检查应用程序中的特定问题SQL语句，而不必为未被调查的SQL语句收集无关的统计信息。

%PROFILE为主查询模块收集SQLStats。
%PROFILEALL为主查询模块及其所有子查询模块收集SQLStats。

可以以任何顺序指定多个%keyword 参数。
多个参数由空格分隔。

#SQL #Caché

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