文章
姚 鑫 · 九月 24, 2021 阅读大约需 4 分钟

第二十五章 SQL命令 CREATE VIEW（二）

第二十五章 SQL命令 CREATE VIEW（二）

通过视图更新

视图可用于更新视图所基于的表。可以通过视图插入新行，更新通过视图看到的行中的数据，以及删除通过视图看到的行。如果CREATE VIEW语句指定了此功能，则可以为视图发出INSERT、UPDATE和DELETE语句。要允许通过视图进行更新，请在定义视图时指定WITH CHECK选项(默认值)。

注意:如果视图基于分片表，则不能通过WITH CHECK OPTION视图进行INSERT、UPDATE或DELETE操作。尝试这样做会导致一个SQLCODE -35，其中%msg INSERT/UPDATE/DELETE not allowed for view (sample.myview) based on sharded table with check option conditions。

若要防止通过视图进行更新，请指定WITH READ ONLY。尝试通过使用READ ONLY创建的视图执行插入、更新或删除操作会生成SQLCODE-35错误。

要通过视图进行更新，必须具有要更新表或视图的适当权限，如GRANT命令所指定。

通过视图更新受以下限制：
- 该视图不能是投影为视图的类查询。
- 视图的类不能包含类参数READONLY=1。
- 视图的SELECT语句不能包含DISTINCT、TOP、GROUP BY或HAVING子句，也不能是UNION的一部分。
- 视图的SELECT语句不能包含子查询。
- 视图的SELECT语句只能列出作为列引用的值表达式。
- 视图的SELECT语句只能有一个表引用；它不能在SELECT-LIST或WHERE子句中包含FROM子句、联接语法或箭头语法。表引用必须指定可更新的表或可更新的视图。

WITH CHECK OPTION子句导致INSERT或UPDATE操作根据视图定义的WHERE子句验证结果行。这可确保插入或修改的行是派生视图表格的一部分。有两个可用的检查选项：

- WITH LOCAL CHECK OPTION-仅检查INSERT或UPDATE语句中指定的视图的WHERE子句。
- WITH CASCADED CHECK OPTION-检查INSERT或UPDATE语句中指定的视图的WHERE子句和所有基础视图。这将覆盖这些基础视图中的任何WITH LOCAL CHECK OPTION子句。对于所有可更新的视图，建议使用WITH CASCADED CHECK选项。

如果指定WITH CHECK OPTION，则CHECK选项默认为CASCADED。关键字CASCADE是CASCADED的同义词。

如果插入操作因检查选项验证失败(如上所述)， IRIS将发出SQLCODE-136错误。

如果更新操作因检查选项验证(如上所述)而失败，则 IRIS会发出SQLCODE-137错误。

示例

下面的示例从PhoneBook表中创建了名为 “ CityPhoneBook ” 的视图:

```
CREATE VIEW CityPhoneBook AS
     SELECT Name FROM PhoneBook WHERE City='Boston'
```

下面的示例从Guides表中创建了一个名为 “ GuideHistory ” 的视图。
它列出了所有的Title以及这个人是否已经退休:

```
CREATE VIEW GuideHistory AS
     SELECT Guides, Title, Retired, Date_Retired
     FROM Guides
```

下面的嵌入式SQL示例创建表MyTest，然后为该表创建一个视图MyTestView，该视图从MyTest中选择一个字段：

```
ClassMethod CreateView1()
{
    d $SYSTEM.Security.Login("_SYSTEM","SYS")
    &sql(DROP TABLE Sample.MyTest)
    &sql(DROP VIEW Sample.MyTestView)
CreateTable
    &sql(CREATE TABLE Sample.MyTest
        (
            TestNum     INT NOT NULL,
            FirstWord   CHAR (30) NOT NULL,
            LastWord    CHAR (30) NOT NULL,
            CONSTRAINT MyTestPK PRIMARY KEY (TestNum)
        )
    )
    if SQLCODE = 0 {
        w !,"???"
    } else {
        w "????? SQLCODE=",SQLCODE
    }
CreateView
    &sql(
        CREATE VIEW Sample.MyTestView AS
            SELECT FirstWord FROM Sample.MyTest
            WITH CASCADED CHECK OPTION
    )
    if SQLCODE = 0 {
        w !,"????"
    } else {
        w "?????? SQLCODE=",SQLCODE
    }
}
```

下面的嵌入式SQL示例创建视图MyTestView，该视图从MyTest中选择两个字段。此视图的SELECT查询包含一个TOP子句和一个ORDER BY子句：

```
ClassMethod CreateView2()
{
    d $SYSTEM.Security.Login("_SYSTEM","SYS")
    &sql(DROP TABLE Sample.MyTest)
    &sql(DROP VIEW Sample.MyTestView)
```

```
CreateTable
    &sql(
        CREATE TABLE Sample.MyTest
        (
            TestNum     INT NOT NULL,
            FirstWord   CHAR (30) NOT NULL,
            LastWord    CHAR (30) NOT NULL,
            CONSTRAINT MyTestPK PRIMARY KEY (TestNum)
        )
    )
    if SQLCODE = 0 {
        w !,"???"
    } else {
        w "????? SQLCODE=",SQLCODE
    }
CreateView
    &sql(
        CREATE VIEW Sample.MyTestView AS
        SELECT TOP ALL FirstWord,LastWord FROM Sample.MyTest
        ORDER BY LastWord
    )
    if SQLCODE = 0 {
        w !,"????"
    } else {
        w "?????? SQLCODE=",SQLCODE
    }
}
```

下面的示例从三个表(Proj、Staff和Works)创建了一个名为 “ StaffWorksDesign ” 的视图。
列Name、Cost和Project提供数据。

```
CREATE VIEW StaffWorksDesign (Name,Cost,Project)
     AS SELECT EmpName,Hours*2*Grade,PName
     FROM Proj,Staff,Works
     WHERE Staff.EmpNum=Works.EmpNum
          AND Works.PNum=Proj.PNum AND PType='Design'
```

下面的例子通过使用UNION从b.table2和a.table1中选择，创建了一个名为 “ v3 ” 的视图:

```
CREATE VIEW v_3(fvarchar)
     AS SELECT DISTINCT *
     FROM
       (SELECT fVARCHAR2 FROM b.table2
        UNION ALL
        SELECT fVARCHAR1 FROM a.table1)
```

#SQL #Caché

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E4%BA%8C%E5%8D%81%E4%BA%94%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-create-view%EF%BC%88%E4%BA%8C%EF%BC%89