

文章

[姚鑫](#) · 十月 12, 2021 阅读大约需 3 分钟

第四十三章 SQL命令 FETCH

第四十三章 SQL命令 FETCH

重新定位游标，并从中检索数据。

大纲

`FETCH cursor-name [INTO host-variable-list ]`

参数

- `cursor-name` - 当前打开的游标的名称。
游标名称是在`DECLARE`命令中指定的。
游标名称区分大小写。
- `INTO host-variable-list` - 可选—将取操作列中的数据放入局部变量中。
`host-variable-list`指定一个主机变量或一个逗号分隔的主机变量列表，它们是包含与游标关联的数据的目标。
`INTO`句是可选的。
如果没有指定，`FETCH`语句只定位游标。

描述

在嵌入式SQL应用程序中，`FETCH`语句从游标检索数据。
所需的操作顺序是：`DECLARE`、`OPEN`、`FETCH`、`CLOSE`。
在未打开的游标上尝试`FETCH`会导致`SQLCODE -102`错误。

作为SQL语句，这只在嵌入式SQL中得到支持。
通过ODBC使用ODBC API支持等价的操作。

`INTO`子句可以指定为`DECLARE`语句的子句，也可以指定为`FETCH`语句的子句，或者两者都指定。
`INTO`子句允许将`fetch`列中的数据放到本地主机变量中。
列表中的每个主机变量，从左到右，都与游标结果集中的相应列相关联。
每个变量的数据类型必须匹配或支持对应结果集列的数据类型的隐式转换。
变量的数量必须与游标选择列表中的列数匹配。

当游标前进到数据的末尾时，`FETCH`操作就完成了。
这将设置`SQLCODE=100`(没有更多数据)。
它还将`%ROWCOUNT`变量设置为获取的行数。

注意:只有当`SQLCODE=0`时，`INTO`子句宿主变量返回的值才是可靠的。
如果`SQLCODE=100`(没有更多数据)，则不应该使用主机变量值。

游标名称不是特定于名称空间的。
更改当前名称空间对声明游标的使用没有影响。
唯一需要考虑的名称空间是`FETCH`必须出现在包含要查询的表的名称空间中。

%ROWID

当FETCH检索可更新游标的行时，它将%ROWID设置为所获取行的ROWID值。
可更新游标是指顶部FROM子句只包含一个元素(表名或可更新视图名)的游标。

为检索到的每一行设置%ROWID受以下条件的限制:

- DECLARE cursorname CURSOR和OPEN cursorname语句不初始化%ROWID;
%ROWID值与之前的值不变。
第一个成功的FETCH设置%ROWID。
每个后续的FETCH检索行都会将%ROWID重置为当前的ROWID。
FETCH如果检索可更新游标的行，则设置%ROWID。
如果游标不可更新，%ROWID将保持不变。
如果没有匹配查询选择条件的行，FETCH不会更改之前的%ROWID值。
在CLOSE或FETCH发出SQLCODE 100 (No Data, or No More Data)时，%ROWID包含检索到的最后一行的ROWID。
- 带有DISTINCT关键字或GROUP BY子句的基于游标的SELECT不会设置%ROWID。
%ROWID值与之前的值(如果有的话)保持不变。
- 基于游标的SELECT只执行聚合操作，不设置%ROWID。
%ROWID值与之前的值(如果有的话)保持不变。

没有声明游标的嵌入式SQL SELECT不会设置%ROWID。
在完成一个简单的SELECT语句后，%ROWID值是不变的。

FETCH for UPDATE or DELETE

可以使用FETCH来检索要进行更新或删除的行。
UPDATE或DELETE必须指定WHERE CURRENT OF子句。
DECLARE应该指定FOR UPDATE子句。
下面的示例显示了一个基于游标的删除操作，它删除所有选中的行:

```
ClassMethod FETCH()
{
    s $NAMESPACE="Samples"
    &sql(
        DECLARE MyCursor CURSOR FOR SELECT %ID,Status
        FROM Sample.Quality WHERE Status='Bad' FOR UPDATE
    )
    &sql(
        OPEN MyCursor
    )
    if SQLCODE<0 {
        w "SQL Open????:",SQLCODE, " ",%msg
        q
    }
    n %ROWCOUNT,%ROWID
    for {
        &sql(
            FETCH MyCursor
        )
        q:SQLCODE'=0
        &sql(
            DELETE FROM Sample.Quality WHERE CURRENT OF MyCursor
        )
    }
    w !,"?????=",%ROWCOUNT
```

```

    &sql(
        CLOSE MyCursor
    )
    if SQLCODE<0 {
        w "SQL??????:", SQLCODE, " ", %msg
        q
    }
}

```

示例

下面的嵌入式SQL示例显示了一个无参数的FOR循环调用FETCH，从名为EmpCursor的游标检索数据。INTO子句在DECLARE语句中指定：

```

ClassMethod FETCH1()
{
    &sql(
        DECLARE EmpCursor CURSOR FOR
            SELECT Name, Home_State
            INTO :name,:state FROM Sample.Employee
            WHERE Home_State %STARTSWITH 'M'
    )
    &sql(
        OPEN EmpCursor
    )
    if SQLCODE<0 {
        w "SQL Open?????:", SQLCODE, " ", %msg
        q
    }
    n %ROWCOUNT,%ROWID
    for {
        &sql(
            FETCH EmpCursor
        )
        q:SQLCODE'=0
        w "count: ",%ROWCOUNT," RowID: ",%ROWID,!
        w "  Name=",name," State=",state,!
    }
    w !,"Final Fetch SQLCODE: ",SQLCODE
    &sql(
        CLOSE EmpCursor
    )
    if SQLCODE<0 {
        w "SQL??????:", SQLCODE, " ", %msg
        q
    }
}

```

下面的嵌入式SQL示例显示了一个无参数的FOR循环调用FETCH，从名为EmpCursor的游标检索数据。INTO子句被指定为FETCH语句的一部分：

```

ClassMethod FETCH2()
{
    &sql(

```

```
        DECLARE C1 CURSOR FOR
            SELECT Name,Home_State INTO :name,:state FROM Sample.Person
            WHERE Home_State %STARTSWITH 'M'
    )
    &sql(OPEN C1)
    if SQLCODE<0 {
        w "SQL Open????:",SQLCODE," ",%msg
        q
    }
    &sql(FETCH C1)
    while (SQLCODE = 0) {
        w "count: ",%ROWCOUNT," RowID: ",%ROWID,!
        w "  Name=",name," State=",state,!
        &sql(FETCH C1)
    }
    w !,"Final Fetch SQLCODE: ",SQLCODE
    &sql(CLOSE C1)
    if SQLCODE<0 {
        w "SQL??????:",SQLCODE," ",%msg
        q
    }
}
```

下面的嵌入式SQL示例显示FETCH检索聚合函数值。
%ROWID没有设置:

```
ClassMethod FETCH3()
{
    &sql(
        DECLARE PersonCursor CURSOR FOR
        SELECT COUNT(*),AVG(Age) FROM Sample.Person
    )
    &sql(OPEN PersonCursor)
    if SQLCODE<0 {
        w "SQL Open????:",SQLCODE," ",%msg
        q
    }
    n %ROWCOUNT
    for {
        &sql(
            FETCH PersonCursor INTO :cnt,:avg
        )
        q:SQLCODE'=0
        w %ROWCOUNT," Num People=",cnt," Average Age=",avg,!
    }
    w !,"Final Fetch SQLCODE: ",SQLCODE
    &sql(CLOSE PersonCursor)
    if SQLCODE<0 {
        w "SQL??????:",SQLCODE," ",%msg
        q
    }
}
```

下面的嵌入式SQL示例显示FETCH检索DISTINCT值。
%ROWID没有设置:

```
ClassMethod FETCH4()
{
    &sql(
        DECLARE EmpCursor CURSOR FOR
        SELECT DISTINCT Home_State FROM Sample.Employee
        WHERE Home_State %STARTSWITH 'M'
        ORDER BY Home_State
    )
    &sql(OPEN EmpCursor)
    if SQLCODE<0 {
        w "SQL Open????:",SQLCODE," ",%msg
        q
    }
    n %ROWCOUNT
    for {
        &sql(
            FETCH EmpCursor INTO :state
        )
        q:SQLCODE'=0
        w %ROWCOUNT," State=",state,!
    }
    w !,"Final Fetch SQLCODE: ",SQLCODE
    &sql(CLOSE EmpCursor)
    if SQLCODE<0 {
        w "SQL??????:",SQLCODE," ",%msg
        q
    }
}
}
```

下面的嵌入式SQL示例显示了游标跨名称空间持久存在。

该游标在%SYS中声明，在USER中打开和获取，在SAMPLES中关闭。

注意，OPEN必须在包含要查询的表的名称空间中执行，FETCH必须能够访问输出主机变量，这些变量是特定于名称空间的：

```
ClassMethod FETCH5()
{
    &sql(USE DATABASE %SYS)
    w $ZNSPACE,!
    &sql(DECLARE NSCursor CURSOR FOR SELECT Name INTO :name FROM Sample.Employee)
    &sql(USE DATABASE "USER")
    w $ZNSPACE,!
    &sql(OPEN NSCursor)
    if SQLCODE<0 {
        w "SQL Open????:",SQLCODE," ",%msg
        q
    }
    n SQLCODE,%ROWCOUNT,%ROWID
    for {
        &sql(FETCH NSCursor)
        q:SQLCODE
        w "Name=",name,!
    }
    &sql(USE DATABASE SAMPLES)
    w $ZNSPACE,!
    &sql(CLOSE NSCursor)
    if SQLCODE<0 {
        w "SQL??????:",SQLCODE," ",%msg
    }
}
```

```
    q  
  }  
}
```

[#SQL](#) [#Caché](#)

源

URL:

<https://cn.community.intersystems.com/post/%E7%AC%AC%E5%9B%9B%E5%8D%81%E4%B8%89%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-fetch>