文章
姚 鑫 · 十一月 10, 2021 阅读大约需 分钟

第七章 SQL命令 SELECT(四)

第七章 SQL命令 SELECT(四)

WHERE子句

WHERE子句限定或取消查询选择中的特定行。
符合条件的行是那些条件表达式为真的行。
条件表达式是逻辑测试(谓词)的列表，可以通过AND和OR逻辑操作符链接这些测试(谓词)。
这些谓词可以使用NOT一元逻辑操作符进行反转。

SQL谓词可分为以下几类:
- Comparison 谓词
- BETWEEN 谓词
- LIKE 谓词
- NULL 谓词
- IN and %INLIST 谓词
- EXISTS 谓词
- FOR SOME 谓词
- FOR SOME %ELEMENT 谓词

条件表达式不能包含聚合函数。
如果要使用聚合函数返回的值指定选择条件，请使用HAVING子句。

WHERE子句可以使用=(内部连接)符号连接操作符指定两个表之间的显式连接。

WHERE子句可以使用箭头语法(->)操作符在表和来自另一个表的字段之间指定隐式连接。

GROUP BY子句

GROUP BY子句将查询的结果行，并根据一个或多个数据库列将它们分成组的组。
当将SELECT与GROUP BY结合使用时，将为GROUP BY字段的每个不同值检索一行。
GROUP BY子句在概念上类似于 IRIS扩展%FOREACH，但是GROUP
BY操作整个查询，而%FOREACH允许在子填充上选择聚合，而不限制整个查询填充。
例如:

```
SELECT Home_State, COUNT(Home_State) AS Population
 FROM Sample.Person
  GROUP BY Home_State
```

这个查询为每个不同的Home_State返回一行。

HAVING 子句

HAVING子句类似于对组进行操的WHERE子句。
它通常与GROUP BY子句或%AFTERHAVING关键字一起使用。
HAVING子句限定或取消查询选择中的特定行。
符合条件的行是那些条件表达式为真的行。
条件表达式是逻辑测试(谓词)的列表，可以通过AND和OR逻辑操符链接这些测试(谓词)。
条件表达式可以包含聚合函数。

ORDER BY 子句

ORDER BY子句由ORDER
BY关键字后面跟着一个选择项或一个以逗号分隔的项列表组成该列表指定显示行的顺序。
每个项目可以有一个可选的ASC(升序)或DESC(降序)。
默认为升序。
ORDER BY子句应用于查询的结果，并且经常与TOP子句配对。

下面的示例返回数据库中所有行的选定字段，并按年龄升序排列这些行:

```
SELECT Home_State, Name, Age
FROM Sample.Person
ORDER BY Age
```

SELECT和事务处理

执行查询的事务被定义为READ COMMITTED或READ UNCOMMITTED。
默认是READ UNCOMMITTED。
不在事务中的查询定义为READ UNCOMMITTED。

- 如果READ
  UNCOMMITTED，则SELECT返回数据的当前状态，包括未提交的正在进行的事务对数据所做的更改。
  这些更改可能随后被回滚。

- 如果READ COMMITTED，则行为取决于SELECT语句的内容。
  通常，在read committed模式下的SELECT语句只会返回对已提交数据的插入和更新更改。
  已被正在进行的事务删除的数据行不会返回，即使这些删除尚未提交并可能回滚。

  但是，如果SELECT语句包含%NOLOCK关键字、DISTINCT子句或GROUP
  BY子句，则SELECT返回数据的当前状态，包括当前事务中尚未提交的对数据的更改。
  SELECT中的聚合函数还返回指定列的数据的当前状态，包括未提交的更改。

Query Metadata

可以使用Dynamic SQL返回关于查询的元数据，例如查询中指定的列数、查询中指定的列的名称(或别名)以及查询中
指定的列的数据类型。

示例

在下面的示例中，在Sample.Person中的所有记录上计算AvgAge computed字段。
HAVING子句管理AvgMiddleAge
computed字段，从Sample.Person中的所有记录中计算那些超过40岁的人的平均年龄。

因此，AvgAge和AvgMiddleAge的每一行都有相同的值。
ORDER BY子句按照Home_State字段值的字母顺序对行进行显示。

```
SELECT Name,Home_State,Age,AVG(Age) AS AvgAge,
 AVG(Age %AFTERHAVING) AS AvgMiddleAge
 FROM Sample.Person
 HAVING Age > 40
 ORDER BY Home_State
```

WHERE/HAVING/ORDER BY

在下面的示例中，WHERE子句将选择限制在七个指定的东北部州。
AvgAge computed字段是根据来自那些Home_States的记录计算的。
HAVING子句管理AvgMiddleAge computed字段，从指定Home_States的记录中计算40岁以上的人的平均年龄。
因此，AvgAge和AvgMiddleAge的每一行都有相同的值。
ORDER BY子句按照Home_State字段值的字母顺序对行进行显示。

```
SELECT Name,Home_State,Age,AVG(Age) AS AvgAge,
 AVG(Age %AFTERHAVING) AS AvgMiddleAge
 FROM Sample.Person
 WHERE Home_State IN ('ME','NH','VT','MA','RI','CT','NY')
 HAVING Age > 40
 ORDER BY Home_State
```

GROUP BY/HAVING/ORDER BY

GROUP BY子句导致对每个Home_State组分别计算AvgAge computed字段。
GROUP BY子句还将输出显示限制为从每个Home_State遇到的第一个记录。
HAVING子句管理AvgMiddleAge computed字段，计算每个Home_State组中40岁以上人群的平均年龄。
ORDER BY子句按照Home_State字段值的字母顺序对行进行显示。

```
 SELECT Name,Home_State,Age,AVG(Age) AS AvgAge,
 AVG(Age %AFTERHAVING) AS AvgMiddleAge
 FROM Sample.Person
 GROUP BY Home_State
 HAVING Age > 40
 ORDER BY Home_State
```

WHERE/GROUP BY/HAVING/ORDER BY

WHERE条款限制了对东北部七州的选择。
GROUP BY子句导致对这七个Home_State组中的每一个单独计算AvgAge computed字段。
GROUP BY子句还将输出显示限制为从每个指定的Home_State遇到的第一个记录。
HAVING子句管理AvgMiddleAge computed字段，计算7个Home_State组中每个组中40岁以上人群的平均年龄。
ORDER BY子句按照Home_State字段值的字母顺序对行进行显示。

```
SELECT Name,Home_State,Age,AVG(Age) AS AvgAge,
 AVG(Age %AFTERHAVING) AS AvgMiddleAge
 FROM Sample.Person
 WHERE Home_State IN ('ME','NH','VT','MA','RI','CT','NY')
 GROUP BY Home_State
```

```
 HAVING Age > 40
 ORDER BY Home_State
```

嵌入式SQL和动态SQL示例

嵌入式SQL和动态SQL可用于从ObjectScript程序中发出SELECT查询。

下面的嵌入式SQL程序从一条记录中检索数据值，并将它们放在INTO子句中指定的输出主机变量中。

```
ClassMethod Select2()
{
    n SQLCODE,%ROWCOUNT
    &sql(
        SELECT Home_State, Name, Age
            INTO :a, :b, :c
        FROM Sample.Person)
    if SQLCODE=0 {
        w !,"  Name=",b
        w !,"  Age=",c
        w !,"  Home State=",a
        w !,"Row count is: ",%ROWCOUNT
    } else {
        w !,"SELECT ??, SQLCODE=",SQLCODE
    }
}
```

```
DHC-APP>d ##class(PHA.TEST.SQLCommand).Select2()

  Name=yaoxin
  Age=31
  Home State=WI
Row count is: 1
```

这个程序检索(最少)一行，因此%ROWCOUNT变量被设置为0或1。
要检索多行，必须声明游标并使用FETCH命令。

下面的动态SQL示例先测试所需表是否存在，并检查当前用户对该表的SELECT特权。
然后执行查询并返回结果集。
它使用WHILE循环对结果集的前10条记录重复调用%Next方法。
它使用%GetData方法显示三个字段值，这些方法指定了SELECT语句中指定的字段位置:

```
ClassMethod Select3()
{
#;  s tname="Sample.Person"
#;  if $SYSTEM.SQL.TableExists(tname) & $SYSTEM.SQL.CheckPrivilege($USERNAME, "1," _
tname, "s"){
#;      GOTO SpecifyQuery
#;  } else {
#;      w "Table unavailable"
#;      q
#;  }
#;SpecifyQuery
    s myquery = 3
```

```
    s myquery(1) = "SELECT Home_State,Name,SSN,Age"
    s myquery(2) = "FROM Sample.Person"
    s myquery(3) = "ORDER BY Name"
    s tStatement = ##class(%SQL.Statement).%New()
    s qStatus = tStatement.%Prepare(.myquery)
    if qStatus '= 1 {
        w "%Prepare failed:"
        d $System.Status.DisplayError(qStatus)
        q
    }
    s rset = tStatement.%Execute()
    if rset.%SQLCODE=0 {
        s x=0
        while x < 10 {
            s x = x + 1
            s status=rset.%Next()
            w rset.%GetData(2)," "   /* Name field */
            w rset.%GetData(1)," "   /* Home_State field */
            w rset.%GetData(4),!     /* Age field */
        }
        w !,"End of Data"
        w !,"SQLCODE=",rset.%SQLCODE," Row Count=",rset.%ROWCOUNT
    } else {
        w !,"SELECT ??, SQLCODE=",rset.%SQLCODE
    }
}
```

#SQL #Caché

源 URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E4%B8%83%E5%8D%81%E4%BA%8C%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-select%EF%BC%88%E5%9B%9B%EF%BC%89