文章
姚 鑫 · 十月 11, 2021 阅读大约需 5 分钟

第四十二章 SQL命令 EXPLAIN

第四十二章 SQL命令 EXPLAIN

返回指定查询的查询计划。

大纲

```
EXPLAIN [ALT] [STAT]  [INTO :host-variable] query
```

参数

- ALT - 可选-返回备用查询计划。默认情况下，返回单个查询计划。
- STAT - 可选-(仅限动态SQL)：返回查询计划运行时性能统计信息。默认情况下，返回不带运行时统计信息的查询计划。对于嵌入式SQL，此语法被忽略。
- INTO :host-variable -
  可选-(仅限嵌入式SQL)：放置查询计划的输出主机变量。对于动态SQL，此语法被忽略。
- query - SELECT query

描述

EXPLAIN命令以xml标记文本字符串的形式返回指定查询的查询计划。
该查询计划作为一个结果集返回，该结果集由一个名为plan的字段组成。

查询必须是SELECT查询。
EXPLAIN不能用于创建查询计划的其他SQL操作，例如带SELECT子句的INSERT。
指定不以SELECT关键字开始的查询将导致SQLCODE -51。
可以使用Show Plan显示非select查询的查询计划。

ALT和STAT关键字可以以任何顺序指定。
INTO关键字必须在这些关键字之后指定。

可选的ALT关键字生成备用查询计划。
所有备用查询计划都以相同的xml标记文本字符串返回。
规范化查询文本(标记为<sql>)在每个查询计划之前列出。

可选的STAT关键字生成查询计划中每个模块的运行时性能统计。
这些统计信息包含在包含查询计划的xml标记文本字符串中。
每个模块的统计数据如下:
- <ModuleName>:模块名。
- <TimeSpent>:模块的总执行时间，以秒为单位。
- <GlobalRefs>:全局引用计数。
- <LinesOfCode>:执行的代码行数。
- <DiskWait>:磁盘等待时间，单位为秒。
- <RowCount>:结果集的行数。
- <ModuleCount>:该模块被执行的次数。
- <Counter>:这个程序被执行的次数。

这些统计信息在查询计划的文本中以xml标记的文本字符串的形式返回。
查询计划中所有模块的性能统计信息会在关联查询计划之前返回。
嵌入式SQL不能生成或返回运行时性能统计数据;
忽略STAT关键字，不发出错误。

EXPLAIN命令通过调用$SYSTEM,SQL.Explain()方法返回Show Plan结果，然后将结果集格式化为包含xml标记文本字符串的单个字段。

EXPLAIN ALT命令通过调用带有all=1限定符的$SYSTEM,SQL.Explain()方法返回备用的显示计划结果，然后将结果集格式化为包含xml标记文本字符串的单个字段。

结果集XML结构

下面是用于EXPLAIN ALT STAT查询的xml标记文本字符串的结构。
为了便于解释，这里提供了换行、缩进和注释注释:

```
<plans> /* tag included even if there is only one plan */
   <plan> /* the first query plan */
      <sql> /* the normalzed SELECT statement text */  </sql>
      <cost value="1147000"/>
         /* if STAT, include the following <stats> tags */
      <stats> <ModuleName>MAIN</ModuleName> /* XML-
tagged list of stats (above) for MAIN module */ </stats>
      <stats> <ModuleName>FIRST</ModuleName> /* XML-
tagged list of stats (above) for FIRST module */ </stats>
      <stats>  /* additional modules */ </stats>
   /* text of query plan */
   </plan>
   <plan> /* if ALT, same info for first alternate plan */
      ...
   </plan>
</plans>
```

Explain() 方法

可以使用$SYSTEM.SQL.Explain()方法从ObjectScript返回相同的查询计划信息，示例如下:

```
/// d ##class(PHA.TEST.SQLCommand).Explain()
ClassMethod Explain()
{
    s myquery=2
    s myquery(1) = "SELECT Name,Age FROM Sample.Person WHERE Name %STARTSWITH 'Q' "
    s myquery(2) = "ORDER BY Age"
    s status = $SYSTEM.SQL.Explain(.myquery,{"all":0},,.plan)
    if status '= 1 {
        w "Explain() failed:"
        d $System.Status.DisplayError(status)
        q
    }
    zw plan
}
```

示例

下面的动态SQL示例以XML字符串的形式返回查询计划。
它首先返回SQL查询文本，然后返回查询计划:

```
ClassMethod Explain1()
{
    s myquery = "EXPLAIN SELECT Name,DOB FROM Sample.Person WHERE Name [ 'Q'"
    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()
    while rset.%Next() {
        d rset.%Print("")
    }
}
```

下面的动态SQL示例与前一个相同，只是它使用%Display()来显示结果。
注意，%Display()在XML字符串的开头添加了列名 “ Plan ”，并在XML字符串的末尾添加了 “ 1 Rows(s) Affected ” :

```
ClassMethod Explain2()
{
    s myquery = "EXPLAIN SELECT Name,DOB FROM Sample.Person WHERE Name [ 'Q'"
    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()
    d rset.%Display()
}
```

下面的动态SQL示例以XML字符串的形式返回查询计划和性能统计数据。
它首先返回SQL查询文本，然后返回性能统计数据(按模块)，然后返回查询计划:

```
ClassMethod Explain3()
{
    s myquery = "EXPLAIN STAT SELECT Name,DOB FROM Sample.Person WHERE Name [ 'Q'"
    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()
```

```
    while rset.%Next() {
        w rset.Plan
    }
}
```

下面的动态SQL示例以XML字符串的形式返回备用查询计划。
它在每个查询计划之前返回SQL查询文本:

```
ClassMethod Explain4()
{
    s myquery = "EXPLAIN STAT SELECT Name,DOB FROM Sample.Person WHERE Name [ 'Q'"
    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()
    while rset.%Next() {
        w rset.Plan
    }
}
```

下面的动态SQL示例返回一个更复杂的查询计划。
请注意，在查询计划之前和查询计划中，性能统计是如何显示的:

```
ClassMethod Explain5()
{
    s q1 = "EXPLAIN STAT SELECT p.Name AS Person, e.Name AS Employee "
    s q2 = "FROM Sample.Person AS p,Sample.Employee AS e "
    s q3 = "WHERE p.Name %STARTSWITH 'Q' GROUP BY e.Name ORDER BY p.Name"
    s myquery = q1_q2_q3
    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()
    while rset.%Next() {
        w rset.Plan
    }
}
```

下面的嵌入式SQL示例以XML字符串的形式返回查询计划。
它首先返回SQL查询文本，然后返回查询计划:

```
   #SQLCompile Select=Runtime
   &sql(EXPLAIN INTO :qplan SELECT Name,DOB FROM Sample.Person WHERE Name [ 'Q')
   WRITE qplan
```

下面的嵌入式SQL示例返回查询计划。
STAT关键字被忽略:

```
#SQLCompile Select=Runtime
&sql(EXPLAIN STAT INTO :qplan SELECT Name,DOB FROM Sample.Person WHERE Name [ 'Q')
WRITE qplan
```

注意：Caché没有此关键字。

#SQL #Caché

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E5%9B%9B%E5%8D%81%E4%BA%8C%E7%AB%A0-sql%E5%91%BD%E4%BB%A4-explain