文章
姚 鑫 · 四月 20, 2021 阅读大约需3 分钟

第四章 缓存查询(一)

第四章 缓存查询(一)

系统自动维护已准备好的SQL语句("查询")的缓存。这允许重新执行SQL查询，而无需重复优化查询和开发查询计划的开销。缓存查询是在准备某些SQL语句时创建的。准备查询发生在运行时，而不是在编译包含SQL查询代码的例程时。通常，PREPARE紧跟在SQL语句的第一次执行之后，但在动态SQL中，可以准备查询而不执行它。后续执行会忽略PREPARE语句，转而访问缓存的查询。要强制对现有查询进行新的准备，必须清除缓存的查询。

所有SQL调用都会创建缓存查询，无论是在ObjectScript例程中调用还是在类方法中调用。
- 动态SQL、ODBC、JDBC和$SYSTEM.SQL.DDLImport()方法在准备查询时创建缓存查询。管理门户执行SQL接口、InterSystems SQL Shell和%SYSTEM.SQL.Execute()方法使用动态SQL，因此使用准备操作来创建缓存查询。

它们列在命名空间(或指定方案)的Management Portal常规缓存查询列表、每个正在访问的表的Management Portal Catalog Details缓存查询列表以及SQL语句列表中。动态SQL遵循本章中介绍的缓存查询命名约定。

- 类查询在准备(%PrepareClassQuery()方法)或第一次执行(调用)时创建缓存查询。

它们列在命名空间的管理门户常规缓存查询列表中。如果类查询是在持久类中定义的，则缓存的查询也会列在该类的Catalog Details缓存查询中。它没有列在正在访问的表的目录详细信息中。它没有列在SQL语句清单中。类查询遵循本章中介绍的缓存查询命名约定。

- 嵌入式SQL在第一次执行SQL代码或通过调用声明游标的OPEN命令启动代码执行时创建缓存查询。嵌入式SQL缓存查询列在管理门户缓存查询列表中，查询类型为嵌入式缓存SQL，SQL语句列表。嵌入式SQL缓存查询遵循不同的缓存查询命名约定。

所有清除缓存查询操作都会删除所有类型的缓存查询。

生成缓存查询的SQL查询语句包括：
- SELECT：SELECT缓存查询显示在其表的目录详细资料中。如果查询引用了多个表，则会为每个被引用的表列出相同的缓存查询。从这些表中的任何一个清除缓存的查询都会将其从所有表中清除。从表的目录详细资料中，可以选择缓存的查询名称以显示高速缓存的查询详细资料，包括执行和显示计划选项。由$SYSTEM.SQL.Schema.ImportDDL("IRIS")方法创建的选择缓存查询不提供Execute和Show Plan选项。

SELECT的DECLARE NAME CURSOR创建缓存查询。但是，缓存的查询详细信息不包括执行和显示计划选项。

- CALL：为其结构创建缓存查询列表中显示的缓存查询。
- INSERT、UPDATE、INSERT或UPDATE、DELETE：创建其表的Catalog Details中显示的缓存查询。
- TRUNCATE TABLE:为其表创建一个缓存查询，该查询显示在目录详细信息中。
  注 意，$SYSTEM.SQL.Schema.ImportDDL("IRIS")不支持截断表。
- SET TRANSACTION, START TRANSACTION, %INTRANSACTION, COMMIT, ROLLBACK:为命名空间中的每个模式创建一个缓存查询，显示在缓存查询列表中。

当准备查询时，将创建一个缓存的查询。
因此，不要将%Prepare()方法放入循环结构中是很重要的。
同一个查询的后续%Prepare()(仅在指定的文字值上有所不同)使用现有的缓存查询，而不是创建新的缓存查询。

更改表的SetMapSelectability()值将使所有引用该表的现有缓存查询失效。
现有查询的后续准备将创建一个新的缓存查询，并从清单中删除旧的缓存查询。

清除缓存查询时，缓存查询将被删除。修改表定义会自动清除引用该表的所有查询。在更新查询缓存元数据时，发出准备或清除命令会自动请求独占的系统范围锁。系统管理员可以修改缓存查询锁定的超时值。

创建缓存的查询不是事务的一部分。缓存查询的创建不会被记录下来。

缓存查询提高了性

第一次准备查询时，SQL引擎会对其进行优化，并生成执行该查询的程序(一个或多个InterSystems IRIS®Data Platform例程的集合)。然后将优化的查询文本存储为缓存查询类。如果随后尝试执行相同(或类似)的查询，SQL引擎将找到缓存的查询并直接执行该查询的代码，从而绕过优化和代码生成需

缓存查询提供以下好处：

- 频繁使用的查询的后续执行速度更快。更重要的是，无需编写繁琐的存储过程即可自动获得这种性能提升。大多数关系数据库产品建议仅使用存储过程访问数据库。对于IRIS，这不是必需
- 单个缓存的查询用于类似的查询，这些查询只是在字面值上有所不同。例如，SELECT TOP 5 Name FROM Sample.Person WHERE Name %STARTSWITH 'A' and SELECT TOP 1000 Name FROM Sample.Person WHERE Name %STARTSWITH 'Mc'，只是top和%startswith条件的文本值不同。为第一查询准备的缓存查询自动用于第二查询。
- 查询缓存在所有数据库用户之间共享；如果用户1准备查询，则用户1023可以利用它。
- 查询优化器可以自由地使用更多时间为给定的查询找到最佳解决方案，因为这个代价只需在第一次准备查询时支付。

InterSystems SQL将所有缓存的查询存储在一个位置，即IRISLOCALDATA数据库。但是，缓存查询是特定于名称空间的。每个缓存的查询都由准备(生成)它的名称空间标识。只能从准备缓存查询的命名空间中查看或执行缓存查询。可以清除当前命名空间或所有命名空间的缓存查询。

缓存查询不包括 注 释。但是，它可以在查询文本后面包含 注 释选项，例如/*#OPTIONS {"optionName":value} */。

因为缓存查询使用现有的查询计划，所以它为现有查询提供了很多的连续性，对基础表的更改(如添加索引或重新定义表优化统计信息)不会对现有缓存查询产生任何影响。

创建缓存查询

当InterSystems IRIS准备查询时，它会确定：
- 如果查询与查询缓存中已有的查询匹配。如果不是，则向查询分配递增计数。
- 如果查询准备成功。如果不是，则不会将递增计数分配给缓存的查询名称。
- 否则，递增计数被分配给缓存的查询名称，并且该查询被缓存。

动态SQL的缓存查询名称

SQL引擎为每个缓存查询分配唯一的类名，格式如下：

```
%sqlcq.namespace.clsnnn
```

其中，NAMESPACE为当前名称空间(大写)，NNN为连续整数。例如，%sqlcq.USER.cls16。

缓存的查询以每个命名空间为基础按顺序编号，从1开始。下一个可用的nnn序列号取决于已保留或释放的编号：

- 如果查询与现有缓存查询不匹配，则在开始准备查询时会保留一个数字。如果查询与现有的缓存查询仅在文字值上不同，则查询与现有的缓存查询匹配-
  这取决于某些其他 注 意事项：隐藏的文本替换、不同的 注 释选项或“单独的缓存查询”中描述的情况。
- 如果查询准备不成功，则保留但不分配号码。只有准备成功的查询才会被缓存。
- 如果缓存查询准备成功，则会保留一个编号并将其分配给缓存查询。无论是否从该表访问任何数据，都会为查询中引用的每个表列出该缓存查询。如果查询未引用任何表，则会创建缓存查询，但不能按表列出或清除。
- 清除缓存查询时会释放一个数字。该号码将作为下一个NNN序列号可用。清除与表关联的单个缓存查询或清除表的所有缓存查询将释放分配给这些缓存查询的编号。清除命名空间中的所有缓存查询会释放分配给缓存查询的所有编号，包括未引用表的缓存查询，以及保留但未分配的编号。

清除缓存查询将重置nnn整数。整数会被重复使用，但剩余的缓存查询不会重新编号。例如，缓存查询的部分清除可能会留下cls1、cls3、cls4和cls7。后续缓存查询将编号为cls2、cls5、cls6和cls8。

一条CALL语句可能会导致多个缓存查询。例如，SQL语句CALL Sample.PersonSets('A','MA') 生成缓存查询：

```
%sqlcq.USER.cls1: CALL Sample . PersonSets ( ? , ? )
%sqlcq.USER.cls2: SELECT name , dob , spouse FROM sample . person
                     WHERE name %STARTSWITH ? ORDER BY 1
%sqlcq.USER.cls3: SELECT name , age , home_city , home_state
                     FROM sample . person WHERE home_state = ? ORDER BY 4 , 1
```

在动态SQL中，准备SQL查询(使用%PrepareClassQuery()或%PrepareClassQuery()实例方法)后，可以使用%display()实例方法或%GetImplementationDetails()实例方法返回缓存的查询名称。查看成功准备的结果。

缓存的查询名称也是由%SQL.Statement类的%Execute()实例方法(以及%CurrentResult属性)返回的结果集OREF的一个组件。以下示例显示了这两种确定缓存查询名称的方法：

```
/// w ##class(PHA.TEST.SQL).CacheQuery()
ClassMethod CacheQuery(c)
{
    SET randtop=$RANDOM(10)+1
    SET randage=$RANDOM(40)+1
    SET myquery = "SELECT TOP ? Name,Age FROM Sample.Person WHERE Age < ?"
    SET tStatement = ##class(%SQL.Statement).%New()
    SET qStatus = tStatement.%Prepare(myquery)
    IF qStatus'=1 {
        WRITE "%Prepare failed:"
        DO $System.Status.DisplayError(qStatus)
        QUIT
    }
    SET x = tStatement.%GetImplementationDetails(.class,.text,.args)
    IF x=1 {
        WRITE "cached query name is: ",class,!
    }
    SET rset = tStatement.%Execute(randtop,randage)
    WRITE "result set OREF: ",rset.%CurrentResult,!
    DO rset.%Display()
    WRITE !,"A sample of ",randtop," rows, with age < ",randage
}
```

```
DHC-APP>w ##class(PHA.TEST.SQL).CacheQuery()
cached query name is: %sqlcq.DHCdAPP.cls51
```

```
result set OREF: 5@%sqlcq.DHCdAPP.cls51
Name    Age
??    7
??    7
O'Rielly,Chris H.       7
Orwell,John V.  4
Zevon,Heloisa O.        11
Smith,Kyra P.   7

6 Rows(s) Affected
A sample of 6 rows, with age < 19
```

在本例中，选定的行数(TOP子句)和WHERE子句谓词值会随着每次查询调用而改变，但缓存的查询名称不会改变。

嵌入式SQL的缓存查询名称

SQL引擎为每个嵌入式SQL缓存查询分配一个唯一的类名，格式如下：

```
%sqlcq.namespace.hash
```

其中，NAMESPACE是当前的名称空间(大写)，HASH是唯一的哈希值。例如，%sqlcq.USER.xEM1h5QIeF4l3jhLZr XlnThVJZDh。

管理门户为每个表列出了嵌入式SQL缓存查询，目录详细信息为每个表列出了具有这个类名的缓存查询，查询类型为嵌入式缓存SQL。

单的缓存查询

两个不应该影响查询优的查询之间的差异仍然会生成的缓存查询：
- 同一函数的不同语法形式会生成的缓存查询。因此，ASCII('x')和{fn ASCII('x')}生成的缓存查询，而{fn CURDATE()}和{fn CURDATE}生成的缓存查询。
- 区分大小写的表别名或列别名值以及可选的AS关键字的存在或不存在将生成的缓存查询。因此，ASCII('x'), ASCII('x') AChar, and ASCII('x') AS AChar会生成的缓存查询。
- 使用不同的ORDER BY子句。
- 使用top all代具有整数值的top。

文字换

当SQL引擎缓存一个SQL查询时，它会执行文字换。
查询缓存中的查询用"?"
字符，表示输入参数。
这意味着，仅在文字值上不同的查询由单个缓存的查询表示。
例如，两个查询:

```
SELECT TOP 11 Name FROM Sample.Person WHERE Name %STARTSWITH 'A'
```

```
SELECT TOP 5 Name FROM Sample.Person WHERE Name %STARTSWITH 'Mc'
```

都由单个缓存查询表示：

```
SELECT TOP ? Name FROM Sample.Person WHERE Name %STARTSWITH ?
```

这最小化了查询高速缓存的大小，并且意味着不需要对仅在字面值上不同的查询执行查询优化。

使用输入主机变量(例如:myvar)和?
输入参数也在相应的缓存查询中用"?"
"字符。
因此，SELECT Name FROM t1 WHERE Name='Adam', SELECT Name FROM t1 WHERE Name=?, and SELECT Name FROM t1 WHERE Name=:namevar
，都是匹配查询，并生成一个缓存查询。

可以使用%GetImplementationDetails()方法来确定这些实体的哪些实体在每个"?"特定准备的字符。

以下 注 意事项适用于文字置换：
- 指定为文字一部分的加号和减号将生成独立的缓存查询。因此，ABS(7)、ABS(-7)和ABS(+7)各自生成一个单独的缓存查询。多个符号也会生成独立的缓存查询：ABS(+?)。ABS(++?)。因此，最好使用无符号变量ABS(?)。或ABS(:Num)，可以为其提供有符号或无符号数字，而无需生成独立的缓存查询。
- 精度和小数值通常不被文字置换。因此，ROUND(567.89, 2)被缓存为ROUND(?, 2)。但是，CURRENT_TIME(N)、CURRENT_TIMESTAMP(N)、GETDATE(N)和GETUTCDATE(N)中的可选精度值不被文字置换。
- IS NULL或IS NOT NULL条件中使用的文字不被文字置换。
- ORDER BY子句中使用的任何文字都不被文字置换。这是因为ORDER BY可以使用整数来指定列位置。更改此整数将导致根本不同的查询。
- 字母文字必须用单引号引起来。某些函数允许指定带引号或不带引号的字母格式代码；只有带引号的字母格式代码才被文字置换。因此，DATENAME(MONTER, 64701)和DATENAME('MONTER', 64701)在功能上是相同的，但是对应的缓存查询是DATENAME(MONTER, ?)。和DATENAME(?, ?)
- 被视为可变数量参数的函数会为每个参数计数生成独立的缓存查询。因此，Coalesce(1, 2)和Coalesce(1, 2, 3)会生成独立的缓存查询。

DynamicSQLTypeList Comment Option

当匹配查询时，注 释选项被视为查询文本的一部分。
因此，在 注 释选项中不同于现有缓存查询的查询与现有缓存查询不匹配。
注 释选项可以作为查询的一部分由用户指定，也可以由SQL预处理器在准备查询之前生成并插入。

如果SQL查询包含文字值，SQL预处理器将生成DynamicSQLTypeList 注 释选项，并将其附加到缓存的查询文本的末尾。此 注 释选项为每个文字分配数据类型。数据类型按照文字在查询中出现的顺序列出。只列出实际文字，而不是输入主机变量或?输入参数。下面是一个典型的例子：

```
SELECT TOP 2 Name,Age FROM Sample.MyTest WHERE Name %STARTSWITH 'B' AND Age > 21.5
```

生成缓存的查询文本：

```
SELECT TOP ? Name , Age FROM Sample . MyTest WHERE Name %STARTSWITH ? AND Age > ? /*#
OPTIONS {"DynamicSQLTypeList":"10,1,11"} */
```

在本例中，文字2被列为类型10(整数)，文字"B"被列为类型1(字符串)，而文字21.5被列为类型11(数字)。

请 注 意，数据类型分配仅基于文字值本身，而不是关联字段的数据类型。例如，在上面的示例中，Age被定义为数据类型INTEGER，但是文字值21.5被列为NUMERIC。因为InterSystems IRIS将数字转换为规范形式，所以文字值21.0将被列为整数，而不是数字。

DynamicSQLTypeList返回以下数据类型值：

数字	描述
1	长度为1到32(包括1到32)的字符串
2	长度为33到128(含)的字符串
3	长度为129到512(含)的字符串
4	长度大于512的字符串
10	Integer
11	Numeric

由于DynamicSQLTypeList 注 释选项是查询文本的一部分，因此更改文本以使其产生不同的数据类型会导致创建单独的缓存查询。例如，增加或减少文字字符串的长度，使其落入不同的范围。

文字替换和性能

SQL引擎对IN谓词的每个值执行文字替换。大量IN谓词值可能会对缓存查询性能产生负面影响。可变数量的IN谓词值可能会导致许多缓存查询。将IN谓词转换为%INLIST谓词会导致谓词只有一个文字替换，而不管列出的值有多少。%INLIST还提供了一个数量级大小参数，SQL使用该参数来优化性能。

取消文字替换

可以取消这种文字替换。在某些情况下，可能希望对文字值进行优化，并为具有该文字值的查询创建单独的缓存查询。若要取消文字替换，请将文字值括在双圆括号中。下面的示例显示了这一点：

```
SELECT TOP 11 Name FROM Sample.Person WHERE Name %STARTSWITH (('A'))
```

指定不同的 %STARTSWITH值将生成单独的缓存查询。请 注 意，对每个文字分别指定禁止文字替换。在上面的示例中，指定不同的TOP值不会生成单独的缓存查询。

要取消有符号数字的文字替换，请指定诸如 ABS(-((7)))之类的语法。

 注 意：在某些情况下，不同数量的括号也可能会抑制文字替换。InterSystems建议始终使用双圆括号作为此目的最清晰和最一致的语法。

共分 注 释选项

如果一个SQL查询指定了多个分割表，则SQL预处理器会生成多个共分片 注 释选项，并将该选项附加到缓存的查询文本的末尾。此共分选项显示是否对指定的表进行共分。

在下面的示例中，所有三个指定的表都进行了编码共享：

```
/*#OPTIONS {"Cosharding":[["T1","T2","T3"]]} */
```

在以下示例中，指定的三个表均未进行编码共享：

```
/*#OPTIONS {"Cosharding":[["T1"],["T2"],["T3"]]} */
```

在以下示例中，表T1未被编分，但表T2和T3被编分：

```
/*#OPTIONS {"Cosharding":[["T1"],["T2","T3"]]} */
```

#SQL #Caché #InterSystems IRIS #InterSystems IRIS for Health

源 URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E5%9B%9B%E7%AB%A0-%E7%BC%93%E5%AD%98%E6%9F%A5%E8%AF%A2%EF%BC%88%E4%B8%80%EF%BC%89