文章
姚 鑫 · 四月 21, 2021 阅读大约需 13 分钟

第四章 缓存查询（二）

第四章 缓存查询（二）

运行时计划选择

运行时计划选择(RTPC)是一个配置选项，它允许SQL优化器利用运行时(查询执行时)的离群值信息。运行时计划选择是系统范围的SQL配置选项。

当RTPC被激活时，准备查询包括检测查询是否包含具有离群值的字段上的条件。如果PREPARE检测到一个或多个异常值字段条件，则不会将查询发送到优化器。相反，SQL会生成一个运行时计划选择存根。在执行时，优化器使用此存根选择要执行的查询计划：忽略离群值状态的标准查询计划，或针对离群值状态进行优化的替代查询计划。如果有多个异常值条件，优化器可以从多个备选运行时查询计划中进行选择。
- 准备查询时，SQL将确定它是否包含离群值字段条件。如果是这样，它将推迟选择查询计划，直到执行查询。在准备时，它创建一条标准SQL语句和(对于动态SQL)相应的缓存查询，但将选择是使用此查询计划还是创建不同的查询计划，直到查询执行。在准备时，它创建看起来像是标准SQL语句的内容，如下所示：DECLARE QRS CURSOR FOR SELECT Top ? Name,HaveContactInfo FROM Sample.MyTest WHERE HaveContactInfo=?，用问号表示文字替代变量。但是，如果查看SQL语句详细资料，则查询计划在准备时包含语句 “ 执行可能导致创建不同的计划 ”，动态SQL查询还会创建看似标准的缓存查询；但是，缓存查询显示计划选项使用SELECT %NORUNTIME关键字显示查询文本，表明这是不使用RTPC的查询计划。
- 执行查询(在嵌入式SQL中打开)时，SQL将创建第二个SQL语句和相应的缓存查询。SQL语句具有散列生成的名称并生成RTPC存根，如下所示： DECLARE C CURSOR FOR %NORUNTIME SELECT Top :%CallArgs(1) Name,HaveContactInfo FROM Sample.MyTest WHERE HaveContactInfo=:%CallArgs(2).然后，优化器使用它来生成相应的缓存查询。如果优化器确定离群值信息没有提供性能优势，它将创建一个与准备时创建的缓存查询相同的缓存查询，并执行该缓存查询。但是，如果优化器确定使用离群值信息可提供性能优势，则它会创建一个缓存查询，以禁止对缓存查询中的离群值字段进行文字替换。例如，如果HaveContactInfo字段是异常值字段(绝大多数记录的值为 ‘ Yes ’)，查询SELECT Name,HaveContactInfo FROM t1 WHERE HaveContactInfo=?将导致缓存查询：SELECT Name,HaveContactInfo FROM t1 WHERE HaveContactInfo=(('Yes')).

请注意，RTPC查询计划的显示根据SQL代码的源代码而有所不同：

管理门户SQL界面显示计划按钮可能会显示另一个运行时查询计划，因为此显示计划从SQL界面文本框中获取其SQL代码。

选中该SQL语句后，将显示包括查询计划的语句详细资料。此查询计划不显示替代运行时查询计划，而是包含文本 “ 执行可能导致创建不同的计划 ”，因为它从语句索引中获取其SQL代码。

如果RTPC未激活，或者查询不包含适当的离群值字段条件，优化器将创建标准SQL语句和相应的缓存查询。

如果一个RTPC存根被冻结，那么所有相关的备用运行时查询计划也会被冻结。
即使关闭了RTPC配置选项，对于冻结的查询，RTPC处理仍然是活动的。

在写查询时，可以通过指定圆括号来手动抑制文字替换: SELECT Name,HaveContactInfo FROM t1 WHERE HaveContactInfo=(('Yes')).如果在条件中抑制离群值字段的文字替换，则RTPC不会应用于查询。
优化器创建一个标准的缓存查询。

激活RTPC

可以使用管理门户或类方法在系统范围内配置RTPC。

注意，更改RTPC配置设置将清除所有缓存的查询。

使用管理门户，根据参数值SQL设置配置系统范围的优化查询。
该选项将运行时计划选择(RTPC)优化和作为离群值(BQO)优化的偏差查询设置为合适的组合。
选择系统管理、配置、SQL和对象设置、SQL来查看和更改此选项。
可用的选择有:
- 假设查询参数值不是字段离群值(BQO=OFF, RTPC=OFF，初始默认值)
- 假设查询参数值经常匹配字段离群值(BQO=ON, RTPC=OFF)
- 在运行时优化实际查询参数值(BQO=OFF, RTPC=ON)

要确定当前设置，调用$SYSTEM.SQL.CurrentSettings()。

$SYSTEM.SQL.Util.SetOption()方法可以在系统范围内激活所有进程的RTPC，如下所示:SET status=$SYSTEM.SQL.Util.SetOption("RTPC"，flag，.oldval)。
flag参数是一个布尔值，用于设置(1)或取消设置(0)RTPC。
oldvalue参数以布尔值的形式返回之前的RTPC设置。

应用RTPC

系统对SELECT和CALL语句应用RTPC。
它不应用RTPC插入、更新或删除语句。

当在以下查询上下文中指定了一个离群值时，系统将RTPC应用于调优表确定的任何字段。

在与文字比较的条件中指定离群值字段。
这个比较条件可以是:
- 使用相等(=)、非相等(!=)、IN或%INLIST谓词的WHERE子句条件。
- 具有相等(=)、非相等(!=)、IN或%INLIST谓词的ON子句连接条件。

如果应用了RTPC，优化器将在运行时确定是应用标准查询计划还是备选查询计划。

如果查询中包含unresolved ？
输入参数。

如果查询指定了用双括号括起来的文字值，则不应用RTPC，从而抑制了文字替换。

如果文字是由子查询提供给离群字段条件的，则RTPC不会被应用。
但是，如果子查询中存在离群字段条件，则应用RTPC。

Overriding RTPC

通过指定%NORUNTIME restrict关键字，可以覆盖特定查询的RTPC。如果查询SELECT Name,HaveContactInfo FROM t1 WHERE HaveContactInfo=? 会导致RTPC处理，查询 SELECT %NORUNTIME Name,HaveContactInfo FROM t1 WHERE HaveContactInfo=?将覆盖RTPC，从而产生一个标准的查询计划。

缓存查询结果集

当执行缓存的查询时，它会创建一个结果集。
缓存的查询结果集是一个对象实例。
这意味着为文字替换输入参数指定的值被存储为对象属性。
这些对象属性使用i%PropName语法引用。

List缓存查询

计算缓存查询

通过调用%Library.SQLCatalog类的GetCachedQueryTableCount()方法，可以确定表的当前缓存查询数。下面的示例显示了这一点：

```
/// w ##class(PHA.TEST.SQL).CountingCachedQueries()
ClassMethod CountingCachedQueries()
{
    SET tbl="Sample.Person"
    SET num=##class(%Library.SQLCatalog).GetCachedQueryTableCount(tbl)
    IF num=0 {
        WRITE "??????? ",tbl
    } ELSE {
        WRITE tbl," ???????? ",num," ????"
    }
    q ""
}
```

```
DHC-APP>w ##class(PHA.TEST.SQL).CountingCachedQueries()
Sample.Person ???????? 2 ????
```

请注意，引用多个表的查询将创建单个缓存查询。但是，这些表中的每一个都单独计算该缓存查询的数量。因此，按表计数的缓存查询数可能大于实际缓存查询数。

显示缓存的查询

可以使用IRIS管理门户查看(和管理)查询缓存的内容。从系统资源管理器中，选择SQL。使用页面顶部的切换选项选择一个命名空间；这将显示可用命名空间的列表。在屏幕左侧打开Cached Queries文件夹。选择其中一个缓存查询将显示详细信息。

查询类型可以是下列值之一：

- %SQL.Statement Dynamic SQL:使用%SQL.Statement的动态SQL查询。
- Embedded cached SQL :嵌入式缓存SQL
- ODBC/JDBC Statement:来自ODBC或JDBC的动态查询。

成功准备SQL语句后，系统会生成一个实现该语句的新类。如果已经设置了Retention Cached Query Source-System-wide配置选项，那么这个生成的类的源代码将被保留，并且可以使用Studio打开以供检查。要执行此操作，请转到IRIS管理门户。从系统管理中，依次选择配置、SQL和对象设置、SQL。在此屏幕上，可以设置保留缓存的查询源选项。如果未设置此选项(默认设置)，系统将生成并部署类，并且不保存源代码。

也可以使用$SYSTEM.SQL.Util.SetOption()方法设置这个系统范围的选项，如下所示：SET status=$SYSTEM.SQL.Util.SetOption("CachedQuerySaveSource",flag,.oldval)。Flag参数是一个布尔值，用于在编译缓存查询后保留(1)或不保留(0)查询源代码；默认值为0。要确定当前设置，请调用$SYSTEM.SQL.CurrentSettings()。

使用^rINDEXSQL列出缓存查询

```
  ZWRITE ^rINDEXSQL("sqlidx",2)
```

此列表中的典型全局变量如下所示：

```
^rINDEXSQL("sqlidx",2,"%sqlcq.USER.cls4.1","oRuYrsuQDz72Q6dBJHa8QtWT/rQ=")="".
```

第三个下标是位置。例如，"%sqlcq.USER.cls4.1"是用户名称空间中的缓存查询；"Sample.MyTable.1"是一条SQL语句。第四个下标是语句散列。

将缓存查询导出到文件

以下实用程序将当前名称空间的所有缓存查询列出到文本文件中。

```
ExportSQL^%qarDDLExport(file,fileOpenParam,eos,cachedQueries,classQueries,classMethod
s,routines,display)
```

- file 要列出缓存查询的文件路径名。指定为带引号的字符串。如果该文件不存在，系统将创建该文件。如果该文件已存在，则InterSystems IRIS会覆盖该文件。
- fileOpenParam 可选-文件的打开模式参数。指定为带引号的字符串。默认值为 “ WNS ”。 “ W ”指定正在打开文件以进行写入。 “ N ”指定如果该文件不存在，则使用此名称创建一个新的顺序文件。 “ S ”指定以回车符、换行符或换页符作为默认终止符的流格式。
- eos 可选-用于分隔清单中各个缓存查询的语句结尾分隔符。指定为带引号的字符串。默认值为 “ GO ”。
- cachedQueries 可选—从查询缓存导出所有SQL查询到文件。一个布尔标志。默认值为1。
- classQueries 可选-从SQL类查询导出所有SQL查询到文件。一个布尔标志。默认值为1。
- classMethods 可选-从类方法导出嵌入式SQL查询到文件。一个布尔标志。默认值为1。
- routines 可选-从MAC例程导出嵌入式SQL查询到文件。这个清单不包括系统例程、缓存查询或生成的例程。一个布尔标志。默认值为1。
- display 可选-在终端屏幕上显示导出进度。一个布尔标志。默认值为0。

下面是一个调用这个缓存查询导出工具的示例:

```
  DO ExportSQL^%qarDDLExport("C:\temp\test\qcache.txt","WNS","GO",1,1,1,1,1)
```

当在终端命令行中执行display=1时，导出进度显示在终端屏幕上，示例如下:

```
Export SQL Text for Cached Query: %sqlcq.USER.cls14..                Done
Export SQL Text for Cached Query: %sqlcq.USER.cls16..                Done
Export SQL Text for Cached Query: %sqlcq.USER.cls17..                Done
Export SQL Text for Cached Query: %sqlcq.USER.cls18..                Done
Export SQL Text for Cached Query: %sqlcq.USER.cls19..                Done
Export SQL statement for Class Query: Cinema.Film.TopCategory...        Done
Export SQL statement for Class Query: Cinema.Film.TopFilms...           Done
Export SQL statement for Class Query: Cinema.FilmCategory.CategoryName...Done
Export SQL statement for Class Query: Cinema.Show.ShowTimes...          Done

20 SQL statements exported to script file C:\temp\test\qcache.txt
```

创建的导出文件包含如下条目:

```
  -- SQL statement from Cached Query %sqlcq.USER.cls30
  SELECT TOP ? Name , Home_State , Age , AVG ( Age ) AS AvgAge FROM Sample . Person O
RDER BY Home_State
GO
```

```
  -- SQL statement from Class Query Cinema.Film.TopCategory
#import Cinema
SELECT TOP 3 ID, Description, Length, Rating, Title, Category->CategoryName
  FROM Film
  WHERE (PlayingNow = 1) AND (Category = :P1)
  ORDER BY TicketsSold DESC
GO


  -- SQL statement(s) from Class Method Aviation.EventCube.Fact.%Count
#import Aviation.EventCube
SELECT COUNT(*) INTO :tCount FROM Aviation_EventCube.Fact
GO
```

这个缓存的查询列表可以用作查询优化计划实用程序的输入。

执行缓存查询

- 从动态SQL：%SQL.Statement准备操作(%PrepareClassQuery()或%ExecDirect())创建缓存查询。使用同一实例的动态SQL%Execute()方法执行最近准备的缓存查询。
- 从终端：可以使用$SYSTEM.SQL类的ExecuteCachedQuery()方法直接执行缓存查询。此方法允许指定输入参数值并限制要输出的行数。可以从终端命令行执行动态SQL%SQL.Statement缓存查询或xDBC缓存查询。此方法主要用于测试有限数据子集上的现有缓存查询。
- 在管理门户SQL界面中：按照上面的 “ 显示缓存的查询 ” 说明进行操作。从所选缓存查询的目录详细资料选项卡中，单击执行链接。

缓存查询锁

在更新缓存的查询元数据时，发出PREPARE或PURCESS语句会自动请求独占的系统范围锁。SQL支持$SYSTEM.SQL.Util.SetOption()方法的系统范围CachedQueryLockTimeout选项。此选项控制在尝试获取对缓存查询元数据的锁定时的锁定超时。默认值为120秒。这比标准的SQL锁定超时(默认为10秒)要长得多。系统管理员可能需要在具有大量并发准备和清除操作的系统上修改此缓存查询锁定超时，尤其是在执行涉及大量(数千)缓存查询的批量清除的系统上。

SET
status=$SYSTEM.SQL.Util.SetOption("CachedQueryLockTimeout",seconds,.oldval)方法设置系统范围的超时值：

```
SetCQTimeout
   SET status=$SYSTEM.SQL.Util.SetOption("CachedQueryLockTimeout",150,.oldval)
   WRITE oldval," initial value cached query seconds",!!
SetCQTimeoutAgain
   SET status=$SYSTEM.SQL.Util.SetOption("CachedQueryLockTimeout",180,.oldval2)
   WRITE oldval2," prior value cached query seconds",!!
ResetCQTimeoutToDefault
   SET status=$SYSTEM.SQL.Util.SetOption("CachedQueryLockTimeout",oldval,.oldval3)
```

CachedQueryLockTimeout设置系统范围内所有新进程的缓存查询锁定超时。它不会更改现有进程的缓存查询锁定超时。

清除缓存的查询

每当修改(更改或删除)表定义时，基于该表的任何查询都会自动从本地系统上的查询缓存中清除。如果重新编译持久

类，则使用该类的任何查询都会自动从本地系统上的查询缓存中清除。

可以使用清除缓存查询选项之一通过管理门户显式清除缓存查询。可以使用SQL命令PURGE Cached Queries显式清除缓存查询。可以使用SQL Shell清除命令显式清除缓存查询。

可以使用$SYSTEM.SQL.Push(N)方法显式清除最近未使用的缓存查询。指定n天数将清除当前命名空间中在过去n天内未使用(准备)的所有缓存查询。将n值指定为0或“ ”将清除当前命名空间中的所有缓存查询。例如，如果在2018年5月11日发出$SYSTEM.SQL.Push(30)方法，则它将仅清除在2018年4月11日之前最后准备的缓存查询。不会清除恰好在30天前(在本例中为4月11日)上次准备的缓存查询。

还可以使用以下方法清除缓存的查询：
- $SYSTEM.SQL.PurgeCQClass()按名称清除当前命名空间中的一个或多个缓存查询。可以将缓存的查询名称指定为逗号分隔的列表。缓存查询名称区分大小写；命名空间名称必须以全大写字母指定。指定的缓存查询名称或缓存查询名称列表必须用引号引起来。
- $SYSTEM.SQL.PurgeForTable()清除当前命名空间中引用指定表的所有缓存查询。架构和表名称不区分大小写。
- $SYSTEM.SQL.PurgeAllNamespaces()清除当前系统上所有名称空间中的所有缓存查询。请注意，删除命名空间时，不会清除与其关联的缓存查询。执行PurgeAllNamespaces()检查是否有任何与不再存在的名称空间相关联的缓存查询；如果有，则清除这些缓存查询。

要清除当前命名空间中的所有缓存查询，请使用管理门户清除此命名空间的所有查询选项。

清除缓存的查询还会清除相关的查询性能统计信息。

清除缓存的查询还会清除相关的SQL语句列表条目。管理门户中列出的SQL语句可能不会立即清除，可能需要按清除陈旧按钮才能从SQL语句列表中清除这些条目。

注意：当您更改系统范围的默认架构名称时，系统会自动清除系统上所有名称空间中的所有缓存查询。

远程系统

在本地系统上清除缓存的查询不会清除该缓存查询在镜像系统上的副本。
必须手动清除远程系统上已清除的缓存查询的副本。

当修改和重新编译持久性类时，基于该类的本地缓存查询将被自动清除。
IRIS不会自动清除远程系统上缓存的查询的副本。
这可能意味着远程系统上缓存的一些查询是“ 过时的 ”(不再有效)。
但是，当远程系统尝试使用缓存的查询时，远程系统会检查查询引用的任何持久类是否已重新编译。
如果重新编译了本地系统上的持久化类，则远程系统在尝试使用它之前会自动清除并重新创建过时的缓存查询。

没有缓存的SQL命令

以下非查询SQL命令不会缓存；它们在使用后会立即清除：
- 数据定义语言(DDL)：CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE VIEW, ALTER VIEW, DROP VIEW, CREATE INDEX, DROP INDEX, CREATE FUNCTION, CREATE METHOD, CREATE PROCEDURE, CREATE QUERY, DROP FUNCTION, DROP METHOD, DROP PROCEDURE, DROP QUERY, CREATE TRIGGER, DROP TRIGGER, CREATE DATABASE, USE DATABASE, DROP DATABASE
- 用户、角色和权限：CREATE USER, ALTER USER, DROP USER, CREATE ROLE, DROP ROLE, GRANT, REVOKE, %CHECKPRIV
- 锁 ：LOCK TABLE, UNLOCK TABLE
- 其他: SAVEPOINT, SET OPTION

请注意，如果从管理门户执行查询界面发出这些SQL命令之一，性能信息将包括如下文本：缓存查询：%sqlcq.USER.cls16。这将显示在中，表示已分配缓存的查询名称。但是，此缓存查询名称不是链接。未创建缓存查询，并且未保留增量缓存查询编号.cls16。 SQL将此缓存的查询号分配给下一个发出的SQL命令。

#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%BA%8C%EF%BC%89