**【Mycat流式查询的实现】**

**项目说明书**

完成时间:2020年9月12日

目 录

[【Mycat流式查询的实现】](#_Toc27626)

[一、 Demand analysis 需求分析](#_Toc12237)

[1、现状与问题](#_Toc15027)

[2、解决思路](#_Toc10694)

[二、 Outline design 概要设计](#_Toc13034)

[1、流式查询的开关](#_Toc17234)

[2、获取内存大小](#_Toc16850)

[3、判断是否进入死循环(阻塞)的位置](#_Toc23964)

[4、limit分页功能](#_Toc4365)

[三、Function realization 功能实现](#_Toc15424)

[1、开关的具体实现](#_Toc7234)

[2、内存大小的获取](#_Toc217)

[3、阻塞位置的确定](#_Toc2286)

[4、limit分页功能的实现](#_Toc17428)

[四、Advantages of Featur 功能的优势](#_Toc31136)

[1、适用范围](#_Toc16503)

[2、影响](#_Toc8241)

[3、性能的改善](#_Toc25654)

## 需求分析

### 1、现状与问题

云计算去IOE(ibm,orale ,emc)浪潮中目前多采用sharding proxy数据库sharding中间件代理架构，实现对常见开源数据库如MySQL的横向扩展，现有开源组件主要有MyCat，dble，这种share nothing架构相对简单，主要针对写扩展，即OLTP场景，但同时也需要兼顾少量统计查询类，如select x from tb大数据量查询，如百万，千万级别，现有MyCat，dble对此查询是一边读取后端MySQL数据到其内存缓存，一边缓存被客户端读取，如果某些业务场景处理慢读取记录速度较慢，数据会在代理层中间件内存积压，对代理中间件的内存大小要求较高，也容易导致代理中间件内存溢出，即缺乏流式控制机制。

### 2、解决思路

针对上述情况所出现的问题，我们设想可以通过实时获取中间件MyCat堆内存的使用情况，以做到监控作用。监控堆内存的使用情况，当将要达到顶峰的时候，为了防止内存溢出想，程序进入堵塞状态，直至客户端开始读取数据，内存的使用率降低之后，程序结束堵塞状态，开始正常运行。这样在资源最大化利用的同时也保证了中间件MyCat的正常运转。

## 概要设计

### 1、流式查询的开关

流式查询，相当于一个附加功能，我们修改的MyCat版本中还没有开发这个功能，但我们是在MyCat的原有功能基础之上进行优化修改实现的这个功能。我们设置一个流式查询开关，可以用来开启和关闭流式查询功能，这样可以方便用户，还能将流式查询作为一个完整的附加功能来看待。我们可以通过在配置文件中模仿其他功能，来完成流式查询开关的实现。

### 2、监控内存大小

实时监控内存大小是整个修改方案的关键点，只有实现了监控内存才能准确的进入和关闭阻塞状态。我们通过互联网查询相关代码实现的方法，理解当中原理，以达到自己使用的目的---监视内存的变化。

### 3、判断进入无限循环(阻塞)的最佳位置

在设计这项功能时，为了使性能更加良好、快捷，我们不断的寻找、判断进入无限循环的最佳位置。我们从两个方面进行判断的，一是从查询的效率方面考虑，二是看对MyCat其他功能有没有产生影响。我们通过不断的尝试，更改位置，测试效果，最后我们找到了最佳的阻塞位置。

### 4、limit分页功能

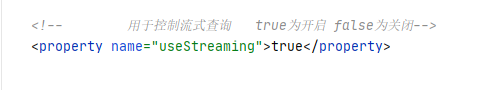
在实现流式查询功能的基础上，我们会选择去突破一下这项功能。MyCat的原理是除了基本查询以外，像分组、排序、limit等含有附加条件的查询，都会将所有的数据全部拉到内存中，进行归并处理，在归并结束之后才会筛选需要的数据，然后返回给客户端，在这样的情况下，我们现在实现的流式查询功能并不能满足这些查询。

但是，经过我们的思考，limit分页查询在省略归并处理之后，同样可以查询的到正确结果，所以我们通过跳过归并，直接将数据返回给客户端。通过这样的方式来实现limit分页查询，但是这项功能同样也包括在流式查询里，并且还是通过流式查询开关来开启或关闭这项功能，所以不会影响到其他功能。

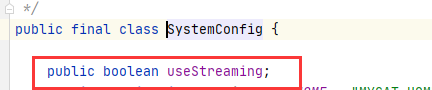
## 三、功能实现

### 1、开关的具体实现

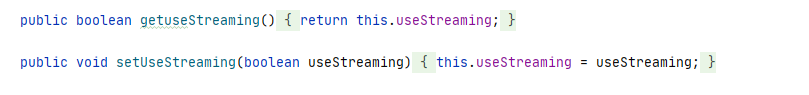
我们在开关的设计上，主要是模仿了MyCat自带开关的设定方法，通过server.xml配置文件中的property标签添加参数并设置参数值，来控制是否要开启流式查询:



在属性值中，true代表开启，false代表关闭。并且在SystemConfig类(系统基础配置项)中声明变量:



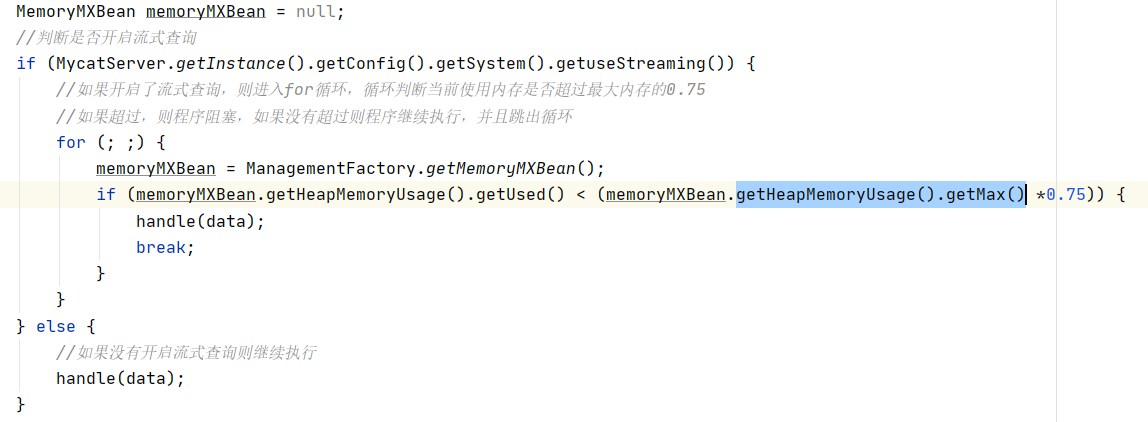
然后还设置set，get方法用于设置和获取配置文件中参数具体的值：



### 2、监控内存大小的具体实现

监控内存大小，我们通过互联网寻找了相关资料，理清原理之后，便使用在了我们需要的地方，通过接口MemoryMXBean声明变量，通过ManagementFactory.*getMemoryMXBean*()可以将接口指向它的实现类对象，通过实现类的getHeapMemoryUsage().getUsed()方法获取正在使用的内存大小，通过getHeapMemoryUsage().getMax()获取最大的内存大小

## 3、阻塞位置的确定

经过我们的判断和综合考虑，我们决定在AbstractConnection类的onReadData方法中读取Socket字节流的位置(data字节流下发出去之前)做流式查询控制，这里可以在不影响MyCat其他功能的条件下，还能提升查询速度。 先通过MemoryMXBean接口声明一个变量，用于监控内存大小；然后判断是否开启流式查询。如果开启,程序会先进入到for无限循环，然后通过MemoryMXBean接口的实现类获取到内存大小的数值，通过if条件判断，如果使用内存小于最大内存(用户设定好的堆内内存)的0.75倍(设置小于最大内存的0.75倍的目的是给MyCat预留一部分内存，这样可以保护MyCat在特殊情况下，也不会导致内存溢出)，则data字节流正常下发，break跳出死循环；使用内存一旦超出最大内存的0.75倍，则一直循环判断，直到客户端拉取数据，内存使用降低。具体实现如下:  


### 4、limit分页功能的实现

按照MyCat原有的查询机制，limit分页查询是会将所有数据拉到堆内内存中归并，然后筛选出结果返回给客户端的。然而我们通过对limit开始的位置和结束的位置，结合进行判断之后，不进入归并，直接存入缓冲区发送给客户端，这样可以在内存不溢出的情况下，得到正确的结果。具体实现如下:



## 四、功能的优势

### 1、实用范围广

此项功能，适用于全平台，全数据库，任意的开发环境。

### 2、连带影响小

对于MyCat其他功能，不会造成任何影响。

### 3、性能提升

不仅对内存做到了最大化利用，还对MyCat做了一个保护。而且，在查询效率方面也有不小的提升。