getArticleList

无特殊参数查询

getArticleList

getTotal优化(TODO)

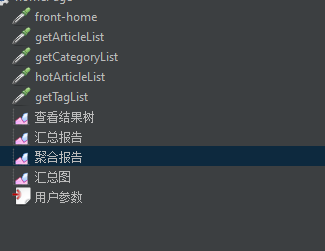
getTagListByArticleId优化(TODO)

Limit优化(doing)

TODO 首页getList为啥不是数据库前十--order by

关于本博客在本地运行环境下的性能优化。

1. 关于getArticleList的性能优化。
   1. 背景：用户在“访问首页”时，除前端项目资源外，还会调用后端项目的一些接口，要想对“访问首页”进行优化，其中少不了对后端接口的优化。
   2. 问题分析：用50线程数/秒对“访问首页”接口进行拟并发请求，结果如下图：



（内容为模拟访问“访问首页”接口的线程组，其中front-home是对前端服务的取样器，其余为后端取样器）

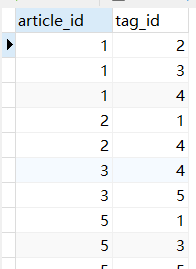
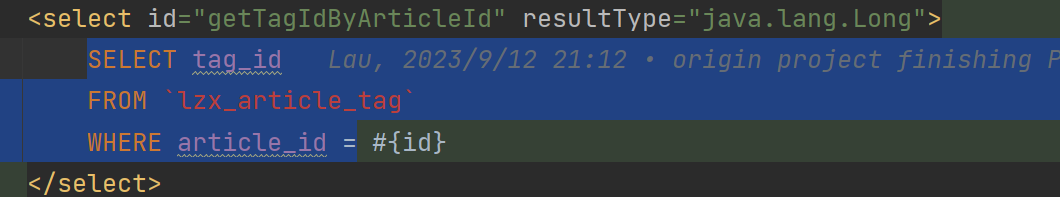


（内容为吞吐量平稳之后，各取样器的相应请求数据）

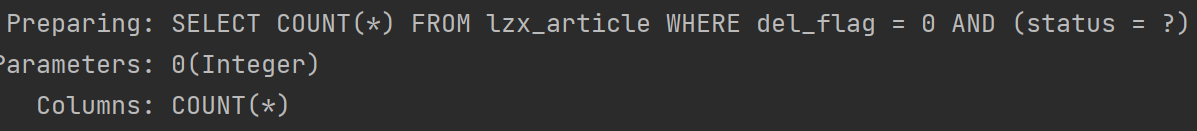
从上面两图不难看出，在后端接口中，getArticleList接口的平均访问时间为23ms，中位数为14ms，都明显高于其他后端接口，因此，优化getArticleList接口成了优化“访问首页”接口的重要一环。那么如何优化？该接口的代码逻辑比较简单，主要是从数据库获取内容，因此重点放在优化查询数据库上。而优化查询数据库，首先想到的是利用索引，然后就是关闭后端mybatis日志，当然关闭日志（TODO）可以暂时放一放。

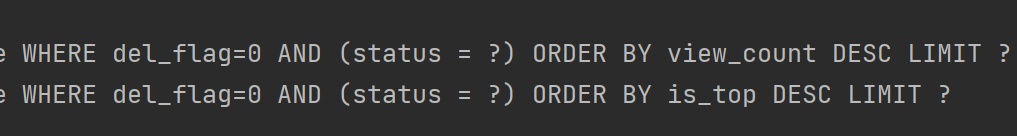
* 1. 问题分析：尝试使用索引，发现并不合适，考虑缓存；同时对getArticleList的另一种

首先我们需要看getArticleList包括哪些查询。从代码不难得知，该接口的查询主要包括getCategoryName(categoryId)，getTagList(articleId),以及分页查询page()。getCategoryName查询的是根据lzx\_category表的主键进行查询，而这张表又很小（少于100行），所以暂不考虑优化。对于getTagList(articleId),该查询又分为从article\_tag这张2个字段的relation表根据主键查询TagId（如下面的两张图），以及从lzx\_tag这张不大（少于100行）表根据主键查询，所以也暂时不考虑优化。



那么我们就来看分页查询page（）。从前端项目进入首页，然后查看后端日志，如下两图所示。从下两图不难看出，“进入首页”接口涉及到2条sql语句，其中第一条（TODO，第一条也尝试优化）是用来查询所有记录数。第二条包含del\_flag,status,view\_count,is\_top字段。

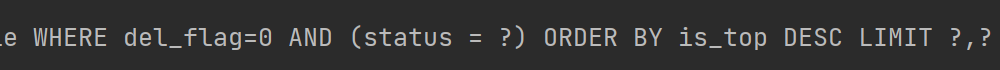




（“进入首页”接口的后端sql语句）

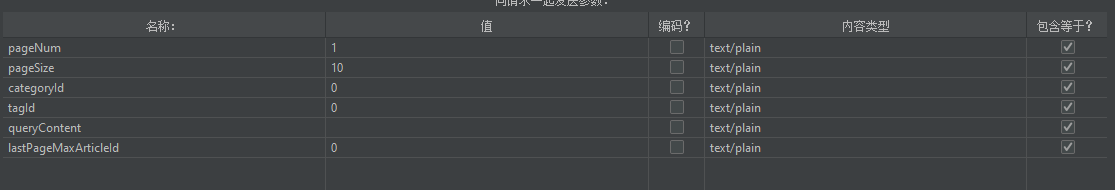
首先看del\_flag,删除字段只有两个值0和1，因此没有必要对此建立索引，status，is\_top同理。因此，想要使用索引优化，并不合适。但是鉴于访问首页的这些article经常访问，而且对于数据一致性要求并不高，我们可以考虑将这些article先放入redis缓存（TODO）中。

同时，从首页还可以点击“查看更多”按钮来进行getArticleList接口的调用。这时的sql仍然为2条，与“进入首页”接口不同的是，第二条sql语句变为了如下图：

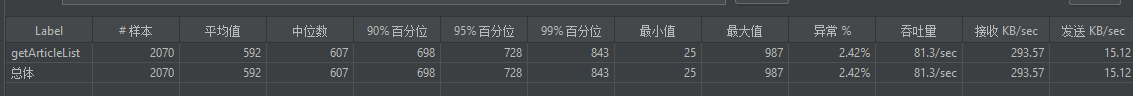


Sql从limit m变成了 limit m, n。这种sql在article数据量特别大时会导致全局扫描太多数据并丢弃，很浪费性能。因此我们对此尝试优化。我们需要自定义分页查询，但这时我发现原代码并不好仅仅针对首页中的getArticleList进行自定义分页查询，因此我对其进行了一定程度的更改。

-----------------------------

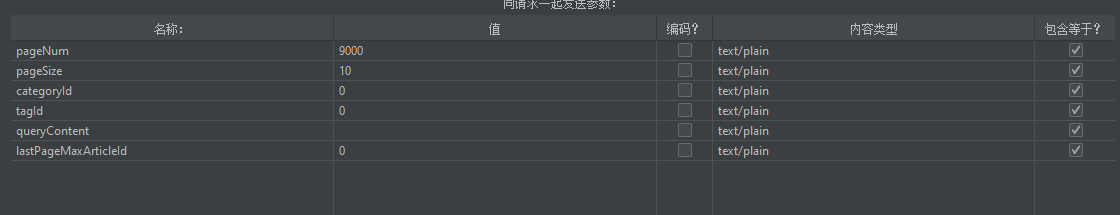






=============

-----------------------------

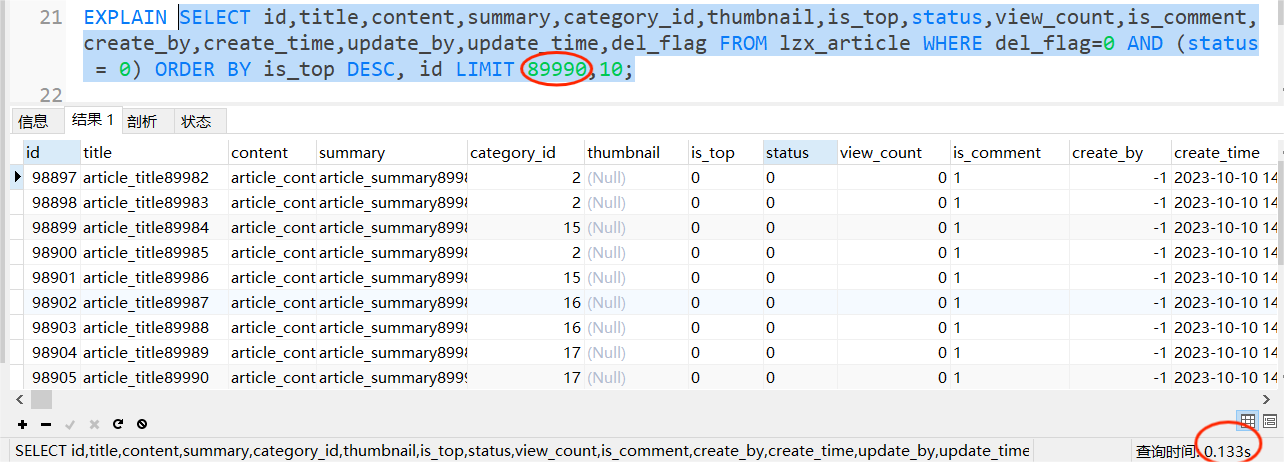


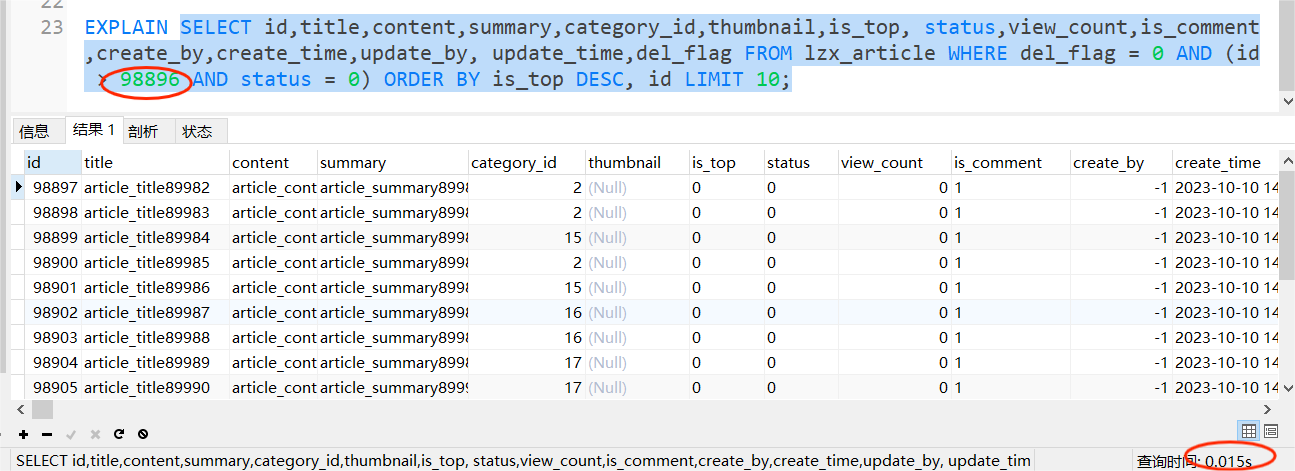




======================

-------------------





=============