

分库分表必会-跨库分页查询看此一篇就够了

π大星的日常 于 2022-09-14 13:56:47 发布



java 专栏收录该内容

11 订阅

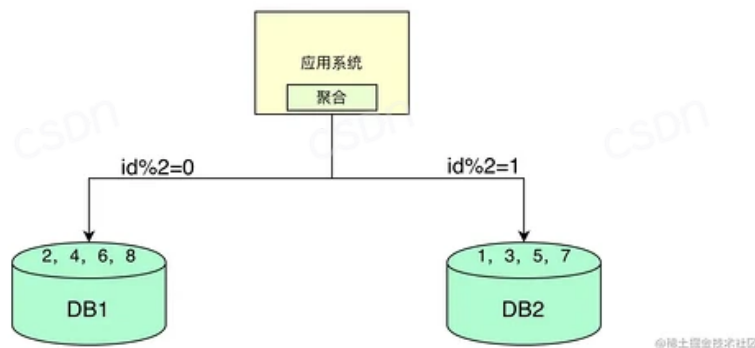
1390 篇文章

订阅专栏

跨库分页查询的问题

概述

随着数据库中数据量日益增多，不得进行分库分表，在分库后将数据分布到不同的数据库实例（甚至物理机器）上，以达到降低数据量，提高系统的处理能力，但是这种架构也带来其他问题，比如本文要讲解的跨库查询



全局查询法

test表有数据[1,2,3,4,5,6,7,8]，在单库的时候，查询第2页数据并且显示2条，语句是这样的

```
select * from test order by id limit 2 offset 2
```

复制代码

数据返回[3,4],但是数据切分以后，如果要查询，这样语句就可能会有问题，例如：在节点1执行此语句，返回【6,8】，节点2返回【5,7】，然后进行排序取前二条返回了【5,6】，可以看到此结果与实际结果不一致，所以应该对sql语句改写为：

```
select * from test order by id limit 0 offset 4;
```

复制代码

内容来源: csdn.net

作者昵称: π大星的日常

原文链接: https://blog.csdn.net/m0_73311735/article/details/126851214

作者主页: https://blog.csdn.net/m0_73311735

然后在根据各节点返回的数据，在进行排序，筛选出第2页的2条

缺点

1. 每个节点返回更多的数据，增大了网络传输量
2. 服务层还需要进行二次排序，增大了服务层的计算量
3. 随着页码的增大，性能会急剧下降

优点

查询简单，数据准确，不用做业务兼容，数据库中间件都支持

禁止跳页查询法

在数据量很大，翻页数很多的时候，很多产品并不提供“直接跳到指定页面”的功能，而只提供“下一页”的功能，这一个小小的业务折衷，就能极大的降低技术方案的复杂度

假设db1中值为【2、4、6、8】，db2中值为【1、3、5、7】，根据id进行排序，返回对应的条数，在内存中对各个节点返回的数据进行排序，得到需要的数据，执行以下语句，查询第一页数据，返回结果集为【1,2】

```
select * from test where id>0 order by id limit 2;
```

复制代码

相比以前的方案，貌似跟以前处理流程一样，但是在查询第二页时，要根据上一页的id的最大值 **id_max**（第一页的最大id_max为2），作为第二页的最小值，那么会将如下语句

```
select * from test order by id limit 2,2;
```

复制代码

改写成：

```
select * from test order by id> 2 limit 2
```

复制代码

这样每个节点不用返回4页数据了，只需要返回跟第一页一样页数的数据，可以看到通过对业务的折中，性能得到大大的提升。

缺点

此种方案需要业务层进行处理，而且不能跳页查询，比如当前页是第一页，直接调到第五页，因无法获取到第四页的最大ID，所以无法查询第五页的数据

内容来源：csdn.net

作者昵称：π大星的日常

原文链接：https://blog.csdn.net/m0_73311735/article/details/126851214

IT资讯：https://blog.csdn.net/m0_73311735/

优点

不会随着页数的增大而影响查询性能

允许数据精度损失查询法

使用partition key进行分库，在数据量较大，数据分布足够随机的情况下，各分库所有非partition key属性，在各个分库上的数据分布，统计概率情况是一致的。

例如，在uid随机的情况下，使用uid取模分两库，db0和db1：

(1) **性别** 属性，如果db0库上的男性用户占比70%，则db1上男性用户占比也应为70% (2) **年龄** 属性，如果db0库上18-28岁少女用户比例占比15%，则db1上少女用户比例也应为15% (3) **时间** 属性，如果db0库上每天10:00之前登录的用户占比为20%，则db1上应该是相同的统计规律



利用这一原理，如上图要查询全局第二页数据，limit 2 offset 2 改写为 limit 1 offset 1，每个分库偏移 1（一半），获取1条数据（半页），得到的数据集的并集，那么结果为【3,4】基本能够认为，是全局数据的limit 2 offset 2的数据，当然这里只是为了所以返回的准确数据，但是实际并不是精准的。

根据实际业务经验，用户都要查询第100页网页、帖子、邮件的数据了，这一页数据的精准性损失，业务上往往是可以接受的，但此时技术方案的复杂度便大大降低了，既不需要返回更多的数据，也不需要进行服务内存排序了

终极大招-二次查询法

以上介绍的方案或多或少都有一定缺点，那么有没有一种方式能够满足业务需要，也能满足性能要求的方法呢，有，那就是二次查询法。

因此方案相比前三个方案理解起来相对复杂点，为了方便说明，所以先单一DB说起，以下单一DB中保存用户年龄数据，1到30岁，总共30条，如果要查询

```
select * from T order by age limit 5 offset 10
```

复制代码

那么会返回以下粉色标识数据，即【11-15】，请记住此结果，下面会讲解怎么分库查询以下结果。

单一DB

内容来源: csdn.net

作者昵称: π大星的日常

原文链接: https://blog.csdn.net/m0_73311735/article/details/126851214

作者主页: https://blog.csdn.net/m0_73311735

1
2
3
...
11
12
13
14
15
..

内容来源: csdn.net
作者昵称: π大星的日常
原文链接: https://blog.csdn.net/m0_73311735/article/details/126851214
作者主页: https://blog.csdn.net/m0_73311735

把以上所有数据进行拆分打散存放到3个分库中，如下，注意下面数据只是用户属性年龄，不是分片键：

DB1	DB2	DB3
1	2	8
3	4	9
7	5	11
10	6	12
14	13	15
16	17	18
21	19	23
22	20	25
24	26	27
28	29	30

@稀土掘金技术社区

通过上文介绍，在单一DB中查询 `limit 5 offset 10`，返回了【11-15】结果，那如果在以上三个分库全局查询 `limit 5 offset 10` 怎么做？

第一步：语句改写

将 `select * from T order by age limit 5 offset 10` 改写为 `select * from T order by age limit 5 offset 3`，并投递给所有的分库，注意，这个 `offset` 的 3，来自于全局 `offset` 的总偏移量 10，除以水平切分数据库个数 3。

执行 `select * from T order by age limit 5 offset 3`，结果如下（粉色标识数据），为了便于理解用青黄色标识库表前三条数据：

内容来源：csdn.net

作者昵称：π大星的日常

原文链接：https://blog.csdn.net/m0_73311735/article/details/126851214

作者主页：https://blog.csdn.net/m0_73311735

DB1	DB2	DB3
1	2	8
3	4	9
7	5	11
10	6	12
14	13	15
16	17	18
21	19	23
22	20	25

@稀土掘金技术社区

第二步：找到返回数据的最小值

1. 第一个库，5 条数据的 age 最小值是10；
2. 第二个库，5 条数据的 age 最小值是 6；
3. 第三个库，5 条数据的 age 最小值是 12；

DB1	DB2	DB3
1	2	8
3	4	9
7	5	11
10	6	12
14	13	15
16	17	18
21	19	23
22	20	25

@稀土掘金技术社区

故，三页数据中，age最小值来自第二个库，age_min=6，这个过程只需要比较各个分库第一条数据，时间复杂度很低

第三步：查询二次改写

内容来源：csdn.net

作者昵称：π大星的日常

原文链接：https://blog.csdn.net/m0_73311735/article/details/126851214

作者主页：https://blog.csdn.net/m0_73311735

第一次改写的SQL语句是select * from T order by age limit 5 offset 3 第二次要改写成between语句，between的起点是age_min，between的终点是原来每个分库各自返回数据的最大值：

第一个分库，第一次返回数据的最大值是22 所以查询改写为select * from T order by age where age between age_min and 22

第二个分库，第一次返回数据的最大值是20 所以查询改写为select * from T order by age where age between age_min and 20

第三个分库，第一次返回数据的最大值是25 所以查询改写为select * from T order by age where age between age_min and 25

相对第一次查询，第二次查询条件放宽了，故第二次查询会返回比第一次查询结果集更多的数据，假设这三个分库返回的数据如下：

DB1	DB2	DB3
1	2	6
3	4	8
6	5	9
7	6	11
10	13	12
14	17	15
16	19	18
21	20	23
22		25

可以看到：

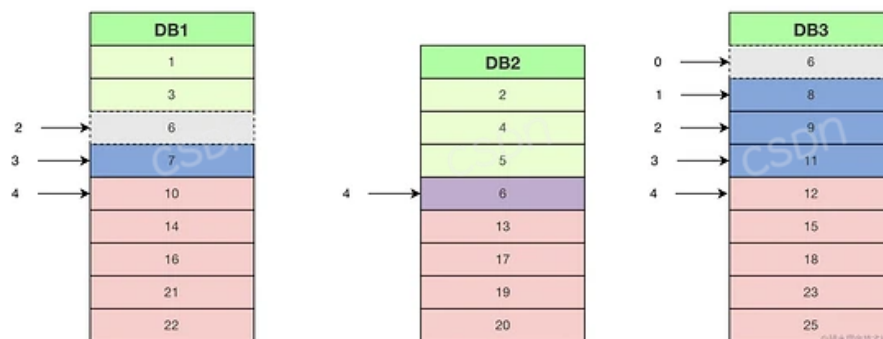
分库一的结果集，比第一次多返回了1条数据，上图中深蓝色记录 7

由于age_min来自原来的分库二，所以分库二的返回结果集和第一次查询相同,其实这次查询可以省掉

分库三的结果集，比第一次多返回了3条数据，上图中深蓝色记录 8,9,11

第四步：找到age_min在全局的offset

在每个结果集中虚拟一个age_min记录，找到age_min在全局的offset



因为查询语句为 `limit 5 offset 3`，所以查询结果集中每个分库的第一条数据offset为4；

分库一中，根据第一次查询条件得出的10的offset是4，查询又返回了一条数据向前推进一位索引，故虚拟age_min在第一个库的offset是2

分库二没有数据变化所以age_min的offset=4

分库三中，根据第一次查询条件得出的12的offset是4，查询又返回了三条数据向前推进三位索引，故虚拟age_min在第三个库的offset是0

因此age_min的全局offset为： **$2+4+0=6$**

第五步：查找最终数据

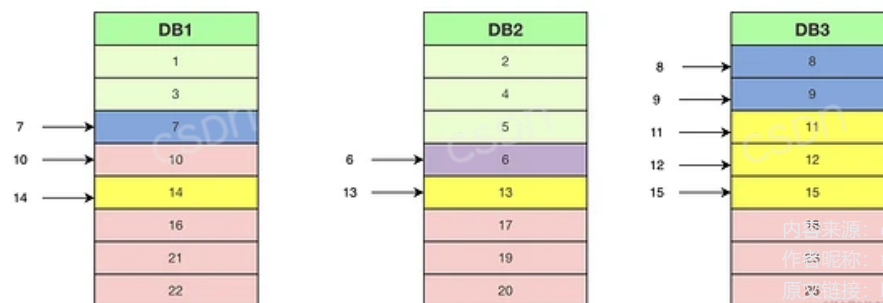
既然得到了age_min在全局的offset为6，就有了全局视野，根据第二次的结果集，就能够得到全局`limit 5 offset 10`的记录（下图黄色标识数据），具体计算如下，各分库二次查询结果如下：

分库1：7、10、14、16、21、22

分库2：6、13、17、19、20

分库3：8、9、11、12、15、18、23、25

统一放到list排序后：**【6、7、8、9、10、11、12、13、14、15、16、17、18、19、20、21、22、23、25】**，得知最小值全局offset为6，最终结果要取 `offset 10 limit 5`，那就 $10-6=4$ ，把排序后结果，向后推移4位，然后再取5位，那就是**【11、12、13、14、15】**



内容来源：csdn.net
 作者昵称：大星的日常
 原文链接：https://blog.csdn.net/m0_73311735/article/details/126851214
 作者主页：https://blog.csdn.net/m0_73311735

优点

精确返回数据，不会随着页数变大而丢失数据


缺点

需要进行两次数据库查询

 文章知识点与官方知识档案匹配，可进一步学习相关知识

MySQL入门技能树 > 查询进阶 > 分页查询 37915 人正在系统学习中

戳一戳货取资料

 微信名片

>

内容来源：csdn.net
作者昵称：π大星的日常
原文链接：https://blog.csdn.net/m0_73311735/article/details/126851214
作者主页：https://blog.csdn.net/m0_73311735