门诊医疗预约系统

优化报告

许正霖-518030910163
组长：周义天-518030910237
杨宇晗-518030910208
焦明胜-518030910249
曹韫琪-518030910242

目录

1. 业务背景简述

建立医院门诊预约系统，对提高工作效率，减少工作量以及解放劳动力具有重要意义。医院作为关系民生的重要机构，每天需要接待的人员基数巨大，尤其是现在人们健康意识的提高，因此，我们需要一个好的设计来构造医院的医疗门诊预约系统。本次实验在原来构建的项目系统基础上做出了优化，使得性能较原先有些许的提升。

2. 优化目标描述

本次优化目标如下：

1、使得长时间的查询过程能够缩短时间，希望对于长时间的查询能够尽量减少其查询时间。

2、高并发的情况下减少查询时间，增加系统健壮度，保证系统的稳定性正确性。

3、希望能够尽量降低数据库中表的复杂度，让对数据库操作的人员能够简便上手便于管理。

3. 性能分析与测试过程、结果

3.1 测试过程

测试过程首先是使用 sql 的语句产生了大部分的数据量，其中大部分数据量主要是属于 appointment patient 部分（考虑到一个现行的医疗数据系统最有可能出现大数据量的地方便是预约和病人部分），其他处则采用了规定个数，类型的方式产生了一定的数据（比如规定科室有哪几种，有哪些门诊，产生了一百条随机的医生信息之类）。

其次采用 mysqlslap 增大了并行量进行查询，对之前编写过的函数或是简单的一条查询语句进行查询时间分析，利用 explain 分析一个查询子句进行时遍历了多少元组，是否使用了索引等操作。

3.2 测试结果

以下展示了几个原先未优化版本的 mysqlslap 分析结果：

1、将已完成的预约条目更新

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=100 --iterations=1 --create-s
chema=proj1.1 --query="call updateIsFinished('700003','2020-06-01', 'Monday 10:00-12:00');" --engine=innodb --number-of-
queries=50000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 3.906 seconds
        Minimum number of seconds to run all queries: 3.906 seconds
        Maximum number of seconds to run all queries: 3.906 seconds
        Number of clients running queries: 100
        Average number of queries per client: 500
```

2、更新某一天 time_slot 状态

```
PS C:\Users\Fisher Chris> mysqlslap --no-defaults -h localhost -uroot -p12345678 --concurrency=100 --iterations=1 --crea
te-schema=project --query="call update_timeslot_status('12','1','2019-06-10',10);" --engine=innodb --number-of-queries=5
000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 463.953 seconds
        Minimum number of seconds to run all queries: 463.953 seconds
        Maximum number of seconds to run all queries: 463.953 seconds
        Number of clients running queries: 100
        Average number of queries per client: 50
```

3、查询一个医生的工作的所有时段

```
PS C:\Users\Fisher Chris> mysqlslap --no-defaults -h localhost -uroot -p12345678 --concurrency=100 --iterations=1 --crea
te-schema=project --query='call Doc_All_timeSlot('100');' --engine=innodb --number-of-queries=5000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 5.515 seconds
        Minimum number of seconds to run all queries: 5.515 seconds
        Maximum number of seconds to run all queries: 5.515 seconds
        Number of clients running queries: 100
        Average number of queries per client: 50
```

4、根据某一条件查询一个医生工作时段的信息

```
PS C:\Users\Fisher Chris> mysqlslap --no-defaults -h localhost -uroot -p12345678 --concurrency=100 --iterations=1 --crea
te-schema=project --query="call Doc_timeSlot_info('65','Monday 08:00-10:00','2019-06-10');" --engine=innodb --number-of-
queries=5000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 0.344 seconds
        Minimum number of seconds to run all queries: 0.344 seconds
        Maximum number of seconds to run all queries: 0.344 seconds
        Number of clients running queries: 100
        Average number of queries per client: 50
```

5、根据某一时段查询该时段预约人数信息

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=100 --iterations=1 --create-s
chema=proj1.1 --query="select getRemainNum('2019-06-10', 'Monday 08:00-10:00', '内科', 'bJYum');" --engine=innodb --numb
er-of-queries=50000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 9.953 seconds
        Minimum number of seconds to run all queries: 9.953 seconds
        Maximum number of seconds to run all queries: 9.953 seconds
        Number of clients running queries: 100
        Average number of queries per client: 500
```

6、改变某一时段的最大人数

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=100 --iterations=1 --create-
chema=proj1.1 --query="call change_maxnum('1', 'Monday 08:00-10:00', '2019-06-10', 99);" --engine=innodb --number-of-qu
ries=50000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 49.937 seconds
        Minimum number of seconds to run all queries: 49.937 seconds
        Maximum number of seconds to run all queries: 49.937 seconds
        Number of clients running queries: 100
        Average number of queries per client: 500
```

7、将已完成的 appoint 设置成绩

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=10 --iterations=1 --create-sc
hema=proj1.1 --query="call setGrade1('622489','2016-09-26','Monday 18:00-20:00','999');" --engine=innodb --number-of-que
ries=5
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 11.844 seconds
        Minimum number of seconds to run all queries: 11.844 seconds
        Maximum number of seconds to run all queries: 11.844 seconds
        Number of clients running queries: 10
        Average number of queries per client: 0
```

8、得到 time_slot_id

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=100 --iterations=1 --create-s
chema=proj1.1 --query="select get_time_slot_id('2020-06-02','Tuesday 08:00-10:00','内科','MVec');" --engine=innodb --nu
ber-of-queries=5000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 1.422 seconds
        Minimum number of seconds to run all queries: 1.422 seconds
        Maximum number of seconds to run all queries: 1.422 seconds
        Number of clients running queries: 100
        Average number of queries per client: 50
```

9、增加一条预约

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=100 --iterations=1 --create-s
chema=proj1.1 --query="call addAppoint('1000','内科','2020-06-02','Tuesday 08:00-10:00','MVec');" --engine=innodb --numb
er-of-queries=5000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 1.656 seconds
        Minimum number of seconds to run all queries: 1.656 seconds
        Maximum number of seconds to run all queries: 1.656 seconds
        Number of clients running queries: 100
        Average number of queries per client: 50
```

3.3 性能分析

从测试结果看出，查询的时间代价基本都是较大的，很多处查询都超过了至少 5s，这对于一个能够正常工作的系统来说的确是令人遗憾的。其中时间过长的地方基本集中在关于 appointment 的操作上(appointment 中的数据为一百万条）。

4. 优化方法描述和优化后性能

4.1 优化方法

4.1.1 分表

首先想到的优化方法就是分表。

当一张的数据达到几百万时，查询一次所花的时间会变多，如果有联合查询的话，会花费更多时间。

分表的目的就在于此，减小数据库的负担，缩短查询时间。尤其是对一

个表中适用区域明显不同的两部分数据。

在我们的项目中，最明显的可用于分表的表为 appointment：可以将未完成的预约和已完成的预约分成两个表，这样查询相关预约时只需要从某一个表中查询即可。

分表后新修改的更新预约状态的函数性能：

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=100 --iterations=1 --create-s
chema=proj1.1 --query="call updateIsFinished0('700003','2020-06-01', 'Monday 10:00-12:00');" --engine=innodb --number-of
-queries=50000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 11.734 seconds
        Minimum number of seconds to run all queries: 11.734 seconds
        Maximum number of seconds to run all queries: 11.734 seconds
        Number of clients running queries: 100
        Average number of queries per client: 500
```

可以看出性能比原来差了许多，这是因为在原先表的基础上只需要将一个名为 isfinished 从 0 改为 1 即代表 appointment 完成，但是在新表中需要将未完成预约表中的一项删除，再将这项的内容插入已完成预约表中成为新的一项。操作的繁琐带来的自然也是性能的下降。但一旦更新完成，appointment 的查询时间会大大降低。

在助教学长的帮助下，我们把更新预约状态函数从即时调整数据所在表，改为每隔一周把未完成表中已完成的数据移入已完成表中。由于病人需要在完成的订单状态被更新至 appointment_finished 表之前完成评分，所以必须限制病人在就诊完成后的一周内完成评分。

4.1.2 添加索引

添加索引之后，DB 在根据 query 进行查询的过程中可以根据现有的一些 index 达到快速查询的效果。

4.1.3 子查询优化

子查询可以一次性完成逻辑上需要多个步骤才能完成的 SQL 操作，但是

在两种情况下性能可能较差：

1、select count(*)语句后接子查询

2、In 和 exist 后接子查询

此次优化过程中我们主要对 In 和 exist 中的子查询进行了优化。

利用之前的一个业务 SQL 进行实验：改变科室

使用 in 方法所需时间：94s

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=10 --iterations=1 --create
chema=proj1.1 --query=" call change_dept('3','2');" --engine=innodb --number-of-queries=50
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 93.875 seconds
        Minimum number of seconds to run all queries: 93.875 seconds
        Maximum number of seconds to run all queries: 93.875 seconds
        Number of clients running queries: 10
        Average number of queries per client: 5
```

使用 exists 方法所需时间：108s

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=10 --iterations=1 --create-s
chema=proj1.1 --query=" call change_dept('3','2');" --engine=innodb --number-of-queries=50
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 108.281 seconds
        Minimum number of seconds to run all queries: 108.281 seconds
        Maximum number of seconds to run all queries: 108.281 seconds
        Number of clients running queries: 10
        Average number of queries per client: 5
```

实验表明：

形如 SELECT * FROM A WHERE id IN (SELECT id FROM B);in 会将 B 的所有查询做完后进行缓存，再将每次 A 的查询结果与缓存的结果对比

而形如 SELECT * FROM A WHERE EXISTS(SELECT 1 FROM B WHERE B.id = A.id) 则会先进行 A 的查询，在再将查询结果带入 B 查询中进行判断

如果内表数据量大而外表数据量小，那么使用 exists 的子查询就更容易得到 true 的返回值，效果相对较好；如果内表数据量小而外表数据量大，

那么使用 in 查询只对内表查询一次，可以避免反复读取内表中数据影响性能。

4.1.4 对 sql 语句进行优化

这一过程主要体现在优化先前写过的功能函数，过程和触发器上，通过将 where 子句提前和利用前三种方法，我们最终得到了优化后的业务 SQL 版本。

4.2 优化后性能

以下展示了优化后版本的 mysqlslap 分析结果：

1、利用 In 和 exists 思想进行优化的函数

更改某一天 time_slot 状态

```
PS C:\Users\Fisher Chris> mysqlslap --no-defaults -h localhost -uroot -p12345678 --concurrency=100 --iterations=1 --crea
te-schema=project --query="call update_timeslot_status('12','1','2019-06-10',10);" --engine=innodb --number-of-queries=5
000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 348.234 seconds
        Minimum number of seconds to run all queries: 348.234 seconds
        Maximum number of seconds to run all queries: 348.234 seconds
        Number of clients running queries: 100
        Average number of queries per client: 50
```

对这一过程的优化主要体现的是 In 和 exist 的思想，优化后的版本选择了 exists，适合于外表小内表大的情况。

2、把 where 提前到子查询思想进行优化的函数

查询一个医生的工作的所有时段

```
PS C:\Users\Fisher Chris> mysqlslap --no-defaults -h localhost -uroot -p12345678 --concurrency=100 --iterations=1 --crea
te-schema=project --query='call Doc_All_timeSlot('100');' --engine=innodb --number-of-queries=5000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 0.328 seconds
        Minimum number of seconds to run all queries: 0.328 seconds
        Maximum number of seconds to run all queries: 0.328 seconds
        Number of clients running queries: 100
        Average number of queries per client: 50
```

根据某一条件查询一个医生工作时段的信息

```
>> mysqlslap --no-defaults -h localhost -uroot -p12345678 --concurrency=100 --iterations=1 --create-schema=project --que
ry="call Doc_timeSlot_info('65','Monday 08:00-10:00','2019-06-10');" --engine=innodb --number-of-queries=5000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 0.281 seconds
        Minimum number of seconds to run all queries: 0.281 seconds
        Maximum number of seconds to run all queries: 0.281 seconds
        Number of clients running queries: 100
        Average number of queries per client: 50
```

得到 time_slot_id

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=100 --iterations=1 --create-s
chema=proj1.1 --query="select get_timeslotid('Tuesday 08:00-10:00','MVec');" --engine=innodb --number-of-queries=5000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 0.359 seconds
        Minimum number of seconds to run all queries: 0.359 seconds
        Maximum number of seconds to run all queries: 0.359 seconds
        Number of clients running queries: 100
        Average number of queries per client: 50
```

调用了 get_timeslotid 的增加一条 appoint 的业务 sql。

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=100 --iterations=1 --create-s
chema=proj1.1 --query="call add_Appoint('1000','内科','2020-06-02','Tuesday 08:00-10:00','MVec');" --engine=innodb --num
ber-of-queries=5000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 0.625 seconds
        Minimum number of seconds to run all queries: 0.625 seconds
        Maximum number of seconds to run all queries: 0.625 seconds
        Number of clients running queries: 100
        Average number of queries per client: 50
```

调用 setGrade 在病人完成就诊后给医生评分

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=10 --iterations=1 --create-sc
hema=proj1.1 --query="call setGrade1('622489','2016-09-26','Monday 18:00-20:00','999');" --engine=innodb --number-of-que
ries=5
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 11.844 seconds
        Minimum number of seconds to run all queries: 11.844 seconds
        Maximum number of seconds to run all queries: 11.844 seconds
        Number of clients running queries: 10
        Average number of queries per client: 0
```

对这一过程的优化主要体现的是利用把 where 提前到子查询中减少

natural join 之前表的行数，进而减少了检索的次数。

3、用子查询的嵌套代替多次 natural join

原先函数

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=100 --iterations=1 --create-s
chema=proj1.1 --query="select getRemainNum('2019-06-10', 'Monday 08:00-10:00', '内科', 'bJYum');" --engine=innodb --numb
er-of-queries=50000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 9.953 seconds
        Minimum number of seconds to run all queries: 9.953 seconds
        Maximum number of seconds to run all queries: 9.953 seconds
        Number of clients running queries: 100
        Average number of queries per client: 500
```

优化后函数

```
PS C:\WINDOWS\system32> mysqlslap --no-defaults -h localhost -uroot -p998998 --concurrency=100 --iterations=1 --create-s
chema=proj1.1 --query="select GetRemainNum1('2019-06-10', 'Monday 08:00-10:00', '内科', 'bJYum');" --engine=innodb --num
ber-of-queries=50000
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Running for engine innodb
        Average number of seconds to run all queries: 8.156 seconds
        Minimum number of seconds to run all queries: 8.156 seconds
        Maximum number of seconds to run all queries: 8.156 seconds
        Number of clients running queries: 100
        Average number of queries per client: 500
```

这一优化思想为：由于参与 natural join 的表数量太多，每次 natural join 时都很多列都在同时重复的被使用，同时，每进行一次 natural join 表的规模都会扩大，可能会造成性能上的问题，故考虑用子查询的嵌套取代 natural join，在具体的尝试中发现，在某些时候使用嵌套查询代替多个表 natural join 可以一定程度上的提升效率，但这可能与表的规模与数据表的列之间的关系相关。同时使用嵌套查询的 sql 语句可读性不如使用 natural join

5. 其他发现

5.1 缓存

为什么第一次查询比第二次慢很多？

在我们进行实验的过程中，我们发现对于一条 select 语句，第一次查询要比第二次查询慢得多：

第一次查询

```
[SQL]select *
from appointment
where patient_id = '95996'

受影响的行: 0
时间: 1.354s
```

第二次查询

```
[SQL]select *
from appointment
where patient_id = '95996'

受影响的行: 0
时间: 0.615s
```

这是为什么呢？联系我们在别的课学到的知识，我们猜想是不是第一次查询是从磁盘读取数据而此后第二次是从类似的 cache 或者别的缓存区读取数据才会带来这么大的差距.

打开 mysql,输入 show global variables like '%query_cache%’；

```
mysql> show global variables like '%query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| have_query_cache             | YES     |
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 1048576 |
| query_cache_type             | OFF     |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
6 rows in set, 1 warning (0.03 sec)
```

我们发现 have_query_cache 这一行确实是 YES，证明 cache 确实是存在的，印证了我们之前的猜测。

此外，输入 show variables like '%storage_engine%'；，发现表的引擎是 innodb，查询资料得知 innodb 有 buffer_pool，也起到了缓冲区的作用.

```
mysql> show variables like '%storage_engine%'
    -> ;
+---------------------------------+--------+
| Variable_name                   | Value  |
+---------------------------------+--------+
| default_storage_engine          | InnoDB |
| default_tmp_storage_engine      | InnoDB |
| disabled_storage_engines        |        |
| internal_tmp_disk_storage_engine | InnoDB |
+---------------------------------+--------+
4 rows in set, 1 warning (0.00 sec)
```

对于一次查询，只要经过之后，就会将其缓冲在缓冲区中，第二次只需要 cache hit 就可以从缓冲区中取出内容，而非再进行一次 IO 读取，因此读取的时间大大减少。

5.2 数据库引擎

我们对 MyISAM 和 InnoDB 两个数据库引擎进行了调查和实验比较，得出以下结果。

MyISAM 基于 ISAM 存储引擎，并对其进行扩展。它是在 Web、数据仓储和其他应用环境下最常使用的存储引擎之一。MyISAM 拥有较高的插入、查询速度，但不支持事物。主要特性有：

1、myisam 只支持表级锁，用户在操作 myisam 表时，select，update，delete，insert 语句都会给表自动加锁，如果加锁以后的表满足 insert 并发的情况下，可以在表的尾部插入新的数据。也可以通过 lock table 命令来锁表，这样操作主要是可以模仿事务，但是消耗非常大，一般只在实验演示中使用。

2、myisam 属于堆表

3、当把删除和更新及插入操作混合使用的时候，动态尺寸的行产生更少碎片。这要通过合并相邻被删除的块，以及若下一个块被删除，就扩展到下一块自动完成

InnoDB 是事务型数据库的首选引擎，支持事务安全表，支持行锁定和外键。主要特性有：

1、Innodb 支持事务和行级锁，InnoDB 给 MySQL 提供了具有提交、回滚和崩溃恢复能力的事物安全（ACID 兼容）存储引擎。

2、Innodb 属于索引组织表。如果使用共享表空间，那么所有表的数据文件和索引文件都保存在一个表空间里，一个表空间可以有多个文件，通过 innodb_data_file_path 和 innodb_data_home_dir 参数设置共享表空间的位置和名字。

3、InnoDB 是为处理巨大数据量的最大性能设计。它的 CPU 效率可能是任何其他基于磁盘的关系型数据库引擎锁不能匹敌的

4、InnoDB 存储引擎完全与 MySQL 服务器整合，InnoDB 存储引擎为在主内存中缓存数据和索引而维持它自己的缓冲池。InnoDB 将它的表和索引在一个逻辑表空间中，表空间可以包含数个文件（或原始磁盘文件）。这与 MyISAM 表不同，比如在 MyISAM 表中每个表被存放在分离的文件中。InnoDB 表可以是任何尺寸，即使在文件尺寸被限制为 2GB 的操作系统上

5、InnoDB 支持外键完整性约束，存储表中的数据时，每张表的存储都按主键顺序存放，如果没有显示在表定义时指定主键，InnoDB 会为每一行生成一个 6 字节的 ROWID，并以此作为主键

据此，我们得出结论：由于 MyISAM 不是事务安全的，而且不支持外键，如果执行大量的 select，insert 操作 MyISAM 比较适合；而 InnoDB 是支持事务安全的引擎，支持外键、行锁、事务是他的最大特点。如果有大量的 update 和 insert 操作，建议使用 InnoDB，特别是针对多个并发和 QPS 较高的情况。