SQL性能与查询优化

一、为何要进行SQL性能与查询优化

在Web应用程序体系架构中，数据持久层（通常是一个关系数据库）是关键的核心部分，它对系统的性能有非常重要的影响。MySQL是目前使用最多的开源数据库，但是MySQL数据库的默认设置性能比较差，因此在产品中使用MySQL数据库必须进行必要的优化。

同时，当用SQL处理大量数据表（比如几百上千万的流水数据表）、表连接频繁的表或是进行批量操作时，如果未对SQL进行性能与查询优化，则会极大的增大工作量，降低处理效率。由此，SQL性能与查询优化就变得尤为重要。

二、如何进行性能优化

首先来看如何进行性能优化。性能优化包括三个方面的优化：数据库表设计优化、性能服务器端优化、存储引擎优化。

1.数据库表的设计优化

在对SQL进行性能优化时，首先要考虑如何设计数据库（Database Schema设计）。数据库设计受到需求与预期数据集大小的影响。因此设计时主要考虑标准化以及数据类型。

1.1标准化

标准化是在数据库中组织数据的过程。标准化的特点有：所有的“对象”都在它自己的table中，没有冗余；数据库通常由E-R图生成；简洁，更新属性通常只需要更新很少的记录；Select，sort优化措施比较少；Join操作比较耗时。标准化与非标准化都有各自的优缺点，在数据库中可以混合使用。例如：冗余一些数据在表格中，例如将id和name存在同一张表中；对于一些简单的对象，直接使用value作为建。例如IP address等；对于Reference by PRIMARY/UNIQUE KEY此类操作可以优化为：

select city\_name

from city,state

where state\_id=state.id and state.code=‘CA’” converted to “select city\_name from city where state\_id=12

1.2数据类型优化

数据类型对查询的最基本优化之一就是通过对数据类型的进行选择以减少表在磁盘上占据的空间，磁盘读入更快，并且在查询过程中表内容被处理所占用的内存更少。同时，在更小的列上建索引，索引也会占用更少的资源。通过对数据类型调整以实现查询优化有以下方式：尽量减少将数字存储为字符串，尽量使用小的数据类型；如果可能，声明列为NOT NULL；使用 ENUM 而不是 VARCHAR；把IP地址存成 UNSIGNED INT

2.性能服务器端优化

2.1MYSQL安装

MySQL有很多发行版本，最好使用MySQL AB发布的二进制版本。使用intel C++编译器编译的MySQL查询速度比正常版本快30%左右。

2.2服务器设置优化

可以通过参数调整以对性能进行优化。在字符集设置时，尽量减少使用Utf-8，因为utf-8占用空间较多；设置最大连接数，若连接数过大则可能造成out of memory；限制MYSQL在同一时间保持打开表的数量，因为打开表的开销较大；MYSQL存放数据的根目录和安装文件分开在不同的磁盘可以提高性能。

3.存储引擎优化

MySQL支持不同的存储引擎，主要使用的有MyISAM和InnoDB。

3.1 MyISAM优化

MyISAM管理非事务表。它提供高速存储和检索，以及全文搜索能力。在对其进行优化时，可将列为NOT NULL，可以减少磁盘存储；使用optimize table做碎片整理，回收空闲空间（仅仅在非常大的数据变化后运行）；Deleting/updating/adding大量数据的时候禁止使用index。使用ALTER TABLE t DISABLE KEYS；设置myisam\_max\_[extra]\_sort\_file\_size足够大，以更好提高repair table的速度；

3.2 InnoDB优化

InnoDB是为在处理巨大数据量时获得最大性能而设计的，它提供了具有提交，回滚和崩溃恢复能力的事务安全（ACID兼容）存储引擎。进行优化时，尽量使用short，integer的主键；加载或插入数据时按主键顺序操作，如果数据没有按主键排序，先排序然后再进行数据库操作；在加载数据时，设置SET UNIQUE\_CHECKS=0，这样可以避免外键和唯一性约束检查的开销；使用prefix keys。因为InnoDB没有key压缩功能。

三、如何进行查询优化

查询优化主要是通过优化查询语句以及建立索引实现。

1.查询语句优化

减少不必要的子查询，建立合适的子查询

mysql> select count(\*) from (select \* from EE\_PawnBill where corporationid = 2494) bill left join mb\_ransomrequestform rrf on rrf.corporationid = 2494 and bill.pawncode=rrf.pawncode , EE\_PawnerInfo pawner, EE\_PawnRansom ransom where bill.pawnerid = pawner.pawnerid and bill.billid = ransom.billid and bill.printflag = 1 and ransom.corporationid = 2494;

+----------+

| count(\*) |

+----------+

| 28798 |

+----------+

1 row in set (1.02 sec)

减少子查询后：

mysql> select count(\*) from EE\_PawnBill bill left join mb\_ransomrequestform rrf on bill.corporationid = 2494 and rrf.corporationid = 2494 and bill.pawncode=rrf.pawncode , EE\_PawnerInfo pawner, EE\_PawnRansom ransom where bill.pawnerid = pawner.pawnerid and bill.billid = ransom.billid and bill.printflag = 1 and ransom.corporationid = 2494;

+----------+

| count(\*) |

+----------+

| 28798 |

+----------+

1 row in set (0.32 sec)

避免全表扫描，如避免在 where 子句中对字段进行 null 值判断，避免在 where 子句中使用or 来连接条件，使用参数，对字段进行表达式操作或对字段进行函数操作。减少不必要的查询。注意不要使用 select \* from t，用具体的字段列表代替“\*”以避免返回用不到的任何字段，提高查询效率。

2.利用索引

建立索引能够极大提高查询速度，优化查询。优化途径有：考虑在where以及order by上建立索引；避免全表扫描，如避免在where子句上使用不能使用索引的操作符（比如!=、<>、like “%xx%”、or）；

mysql> select bill.billId from EE\_PawnBill bill, EE\_PawnerInfo pawner where bill.pawnerid = pawner.pawnerid and bill.businessstatus in (0,1,5) and bill.corporationid = 14077 and bill.status = 0 and pawner.username like '%薛建%' and (bill.printflag = '1' or bill.printstatus=0) order by bill.crtTime desc limit 15;

+----------------------+

| billId |

+----------------------+

| DD330150020160421002 |

+----------------------+

1 row in set (2.58 sec)

**将like “%xx%”改成like “xxx%”可以使用索引**

mysql> select bill.billId from EE\_PawnBill bill, EE\_PawnerInfo pawner where bill.pawnerid = pawner.pawnerid and bill.businessstatus in (0,1,5) and bill.corporationid = 14077 and bill.status = 0 and pawner.username like '薛建%' and (bill.printflag = '1' or bill.printstatus=0) order by bill.crtTime desc limit 15;

+----------------------+

| billId |

+----------------------+

| DD330150020160421002 |

+----------------------+

1 row in set (0.01 sec)

此外，在建立索引时还要注意避免建立用不到的索引、重复的索引、选择性的使用索引，如减少在变化很少的列（例如性别列）上使用索引、以及在已有索引的前缀上建立索引，例如：如果存在index（a，b）则去掉index（a）。避免在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引；如果该索引是复合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使用，并且应尽可能的让字段顺序与索引顺序相一致。

参考网页：

1.<https://dev.mysql.com/doc/refman/5.7/en/order-by-optimization.html>

2. <https://blog.csdn.net/gaowenhui2008/article/details/53905911>