计算机学院实验报告

课 程　 数据库

实验名称　 数据库查询优化及索引设计

专 业 软件工程

班 级 1602

学 号　 161060009

学生姓名 黄佳俊

嘉应学院计算机学院

2018 年 12月 23日

一、实验目的

通过本实验掌握MYSQL 数据库查询优化方法及索引设计方法。

二、实验原理

在实际应用中，数据库系统优化是很重要的课题。好的数据库管理系统有好的性能优化器，应该尽量使用性能优化器去分析系统的瓶颈。

但是性能优化器并不能解决所有问题，除应用性能优化器外，还应该重点关注下面几个问题：

（一） 数据库模式及数据类型的优化

（二） 数据库索引设计

（三） 查询优化

（四） 应用系统的优化

三、实验环境

操作系统： win710

开发环境： MYSQL 8.0， MySQL Workbench 8.0 CE

四、实验内容方法

（一） 数据库模式及数据类型的优化

1.范式化-反范式化 Normalization and Denormalization

例如下面的表

员工 部门 负责人

Jones Accounting Jones

Smith Engineering Smith

Brown Accounting Jones

Green Engineering Smith

存在传递依赖

员工 -〉部门， 部门-〉负责人

通过模式分解后得到2张表，都是BCNF

员工 部门

Jones Accounting

Smith Engineering

Brown Accounting

Green Engineering

部门 负责人

Accounting Jones

Engineering Smith

范式化的好处：

消除了冗余数据； 比非范式化更新速度快； 范式化后表通常较小，更加适合调入内存处理性能高。

范式化的缺点

在查询数据时，需要更加多的连接开销。不仅增加JOIN数，并且会使本可以属于一个索引的列分隔到不同的表中。

2. 优化数据类型

1）尽量使用可以正确存储数据的最小数据类型。

占用更少磁盘，内存，cup。

2）使用简单的。

整型比字符操作代价低，原因：字符集和校对规则。

3）避免null -通常使用 not null 限制

可为null的列索引统计更复杂，更多存储空间，如果确实需要才使用。

4）时间类型

datetime类型适合用来记录数据的原始的创建时间，timestamp类型适合用来记录数据的最后修改时间，只要修改记录，timestamp字段的值都会被自动更新。

5）小数类型。float和double近似小数，decimal精确小数，尽量只在对小数精确计算时使用，数据量大时使用bigint替代。

6）字符型。varchar可变长字符串适合长的字符串（需要额外的1或2个字节记录长度），char定长的，长度不够用空格填充，适合短的字符串及定值的如MD5值，或者经常变更的。

7）存储很大的字符串数据。使用blob（二进制方式）和text（字符方式，有排序规则和字符集），性能低下，尽量避免。

8）存储IPv4使用整型而不是varchar(15)，因为它实际就是32位无符号整数，加小数点只是方便阅读。

所以我们应当用Integer等内建类型而非字符串来保存日期时间或IP地址。

（二） 数据库索引设计

索引优化应该是对查询性能优化最有效的手段了，索引能够轻易将查询性能提高几个数量级。

MySQL主要有以下几种索引类型：

1.普通索引 create index ind on student(ssex) ,一张表< 16 个索引

2.唯一索引 sname char(20) UNIQUE not null

3.主键索引 sno int primary key

4.组合索引（多列索引）

5.全文索引，6.空间索引

create index ind on student(ssex)

创建索引方法

CREATE INDEX语法

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX|key index\_name

[USING index\_type ]

ON tbl\_name ( index\_col\_name ,...)

（1）直接创建索引

CREATE INDEX index\_name ON table(column(length))

（2）修改表结构的方式添加索引

ALTER TABLE table\_name ADD INDEX index\_name ON (column(length))

（3）创建表的时候同时创建索引

CREATE TABLE table\_name[col\_name data type]

[unique|fulltext][index|key][index\_name](col\_name)

1. unique|fulltext为可选参数，分别表示唯一索引、全文索引

2.index和key为同义词，两者作用相同，用来指定创建索引

3.col\_name为需要创建索引的字段列，该列必须从数据表中该定义的多个列中选择

4.index\_name指定索引的名称，为可选参数，如果不指定，默认col\_name为索引值

选择合适的索引列

1. 查询频繁的列，在where，group by，order by，on从句中出现的列

2. where条件中<，<=，=，>，>=，between，in，以及like 字符串+通配符（%）出现的列

3. 长度小的列，索引字段越小越好，因为数据库的存储单位是页，一页中能存下的数据越多越好

例如：

create table stu(

sno int primary key,

sname char(10) unique not null,

ssex char(2),

sage int,

index ind1(sage));

查看索引信息

show index from stu \G;

SHOW INDEX 会返回以下字段：

· Table 表的名称。

· Non\_unique 如果索引不能包括重复词，则为0。如果可以，则为1。

· Key\_name 索引的名称。

· Seq\_in\_index 索引中的列序列号，从1开始。

· Column\_name 列名称。

· Collation 列以什么方式存储在索引中。在MySQL中，有值‘A’（升序）或NULL（无分类）。

· Cardinality 索引中唯一值的数目的估计值。

· Sub\_part 如果列只是被部分地编入索引，则为被编入索引的字符的数目。如果整列被编入索引，则为NULL。

· Packed 指示关键字如何被压缩。如果没有被压缩，则为NULL。

· Null 如果列含有NULL，则含有YES。如果没有，则该列含有NO。

· Index\_type 用过的索引方法（BTREE, FULLTEXT, HASH, RTREE）。

· Comment 多种评注。

B 树 , hash ,R 树

多列索引

CREATE TABLE employees (

id INT NOT NULL,

last\_name CHAR(30) NOT NULL,

first\_name CHAR(30) NOT NULL,

PRIMARY KEY (id),

INDEX mindx (last\_name,first\_name)

);

create index mindx on employees(lastname,firstname);

检查索引信息

mysql> show index from employees;

删除索引

DROP INDEX index\_name ON table

（三） 查询优化

1 为什么查询速度会慢

通常来说，查询的生命周期大致可以按照顺序来看：从客户端>>服务器>>在服务器上进行解析>>生成执行计划>>执行>>返回结果给客户端。其中执行可以认为是整个生命周期中最重要的阶段，这其中包括了大量为了检索数据到存储引擎的调用以及调用后的数据处理，包括排序、分组等。了解查询的生命周期、清楚查询的时间消耗情况对于优化查询有很大的意义。

查询性能低下的最基本的原因是访问的数据太多。大部分性能低下的查询都可以通过减少访问的数据量的方式进行优化。

SQL语句优化的一些原则

1. 使用limit对查询结果的记录进行限定

2. 避免select \*，将需要查找的字段列出来

3. 使用连接（join）来代替子查询

4. 拆分大的delete或insert语句

数据库系统都有查询优化器，可以通过explain 命令进行分析查询。

explain 命令

例如分析下列

explain select \* from products \G;

explain select productid from products \G;

id SELECT识别符。这是SELECT的查询序列号

select\_type SELECT类型,可以为以下任何一种:

SIMPLE:简单SELECT(不使用UNION或子查询)

PRIMARY:最外面的SELECT

UNION:UNION中的第二个或后面的SELECT语句

DEPENDENT UNION:UNION中的第二个或后面的SELECT语句,取决于外面的查询

UNION RESULT:UNION 的结果

SUBQUERY:子查询中的第一个SELECT

DEPENDENT SUBQUERY:子查询中的第一个SELECT,取决于外面的查询

DERIVED:导出表的SELECT(FROM子句的子查询)

table 输出的行所引用的表

type 联接类型。下面给出各种联接类型,按照从最佳类型到最坏类型进行排序（system,const ...ALL):

system:表仅有一行(=系统表)。这是const联接类型的一个特例。

const:表最多有一个匹配行,它将在查询开始时被读取。因为仅有一行,在这行的列值可被优化器剩余部分认为是常数。const表很快,因为它们只读取一次!

eq\_ref:对于每个来自于前面的表的行组合,从该表中读取一行。这可能是最好的联接类型,除了const类型。

ref:对于每个来自于前面的表的行组合,所有有匹配索引值的行将从这张表中读取。

ref\_or\_null:该联接类型如同ref,但是添加了MySQL可以专门搜索包含NULL值的行。

index\_merge:该联接类型表示使用了索引合并优化方法。

unique\_subquery:该类型替换了下面形式的IN子查询的ref: value IN (SELECT primary\_key FROM single\_table WHERE some\_expr) unique\_subquery是一个索引查找函数,可以完全替换子查询,效率更高。

index\_subquery:该联接类型类似于unique\_subquery。可以替换IN子查询,但只适合下列形式的子查询中的非唯一索引: value IN (SELECT key\_column FROM single\_table WHERE some\_expr)

range:只检索给定范围的行,使用一个索引来选择行。

index:该联接类型与ALL相同,除了只有索引树被扫描。这通常比ALL快,因为索引文件通常比数据文件小。

ALL:对于每个来自于先前的表的行组合,进行完整的表扫描。这个连接类型对于前面的每一个记录联合进行完全扫描，这一般比较糟糕，应该尽量避免

possible\_keys 指出MySQL能使用哪个索引在该表中找到行

key 显示MySQL实际决定使用的键(索引)。如果没有选择索引,键是NULL。

key\_len 显示MySQL决定使用的键长度。如果键是NULL,则长度为NULL。

ref 显示使用哪个列或常数与key一起从表中选择行。

rows 显示MySQL认为它执行查询时必须检查的行数。多行之间的数据相乘可以估算要处理的行数。

filtered 显示了通过条件过滤出的行数的百分比估计值。

Extra 该列包含MySQL解决查询的详细信息

Distinct:MySQL发现第1个匹配行后,停止为当前的行组合搜索更多的行。

Not exists:MySQL能够对查询进行LEFT JOIN优化,发现1个匹配LEFT JOIN标准的行后,不再为前面的的行组合在该表内检查更多的行。

range checked for each record (index map: #):MySQL没有发现好的可以使用的索引,但发现如果来自前面的表的列值已知,可能部分索引可以使用。

Using filesort:MySQL需要额外的一次传递,以找出如何按排序顺序检索行。(糟糕情况）

Using index:从只使用索引树中的信息而不需要进一步搜索读取实际的行来检索表中的列信息。

Using temporary:为了解决查询,MySQL需要创建一个临时表来容纳结果。(糟糕情况）

Using where:WHERE 子句用于限制哪一个行匹配下一个表或发送到客户。

Using sort\_union(...), Using union(...), Using intersect(...):这些函数说明如何为index\_merge联接类型合并索引扫描。

Using index for group-by:类似于访问表的Using index方式,Using index for group-by表示MySQL发现了一个索引,可以用来查询

GROUP BY或DISTINCT查询的所有列,而不要额外搜索硬盘访问实际的表。

explain SELECT

Orders.ShipName as 货主名,

Orders.ShipAddress,

Orders.ShipCity,

Orders.ShipRegion,

Orders.ShipPostalCode,

Orders.ShipCountry,

Orders.CustomerID,

Customers.CompanyName AS 客户公司名,

Customers.Address,

Customers.City,

Customers.Region,

Customers.PostalCode,

Customers.Country,

(Employees.FirstName + ' ' + Employees.LastName) AS 售货人,

Orders.OrderID,

Orders.OrderDate,

Orders.RequiredDate,

Orders.ShippedDate,

Shippers.CompanyName As 运货商,

`OrderDetails`.ProductID,

Products.ProductName,

`OrderDetails`.UnitPrice,

`OrderDetails`.Quantity,

`OrderDetails`.Discount,

ROUND(`OrderDetails`.UnitPrice\*Quantity\*(1-Discount)) AS 销售量,

Orders.Freight as 运货费

FROM Customers

JOIN Orders ON Customers.CustomerID = Orders.CustomerID

JOIN Employees ON Employees.EmployeeID = Orders.EmployeeID

JOIN `OrderDetails` ON Orders.OrderID = `OrderDetails`.OrderID

JOIN Products ON Products.ProductID = `OrderDetails`.ProductID

JOIN Shippers ON Shippers.ShipperID = Orders.ShipVia \G;

（四） 应用系统的优化

1 重用数据库连接

建立数据库连接是一种昂贵的操作。(建立 连接池， tomcat ）

2 最小化SQL语句解析

SQL语句解析是CPU密集型操作。

3 尽可能一次处理多行 - 批处理 -(java 调用 存储过程）

在批量中获取、处理和写入行要比逐行执行要快得多。单一SQL处理所有行并执行所有操作的语句提供最佳性能。这是因为它只需要一个

网络往返，并且使用面向高度的面向集合的处理。

4 减少数据冲突（锁）

数据冲突会极大地损害应用程序性能。

应用程序将数据分发到多个表空间/表/分区中，避免对同一行数据进行反复更新（例如计算平衡），并运行定期报告代替，以减少数据冲突。

5 谨慎使用面向对象 （sql -> 对象转换）

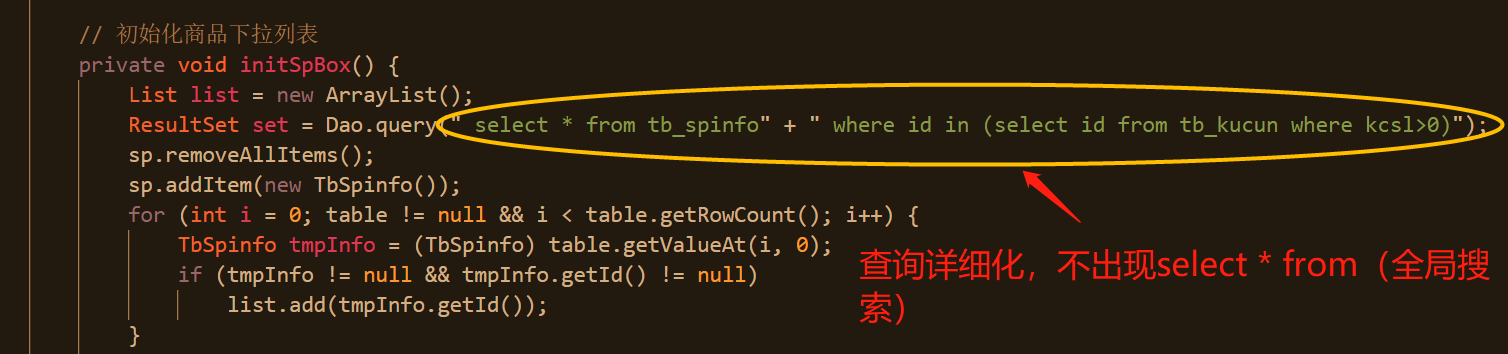
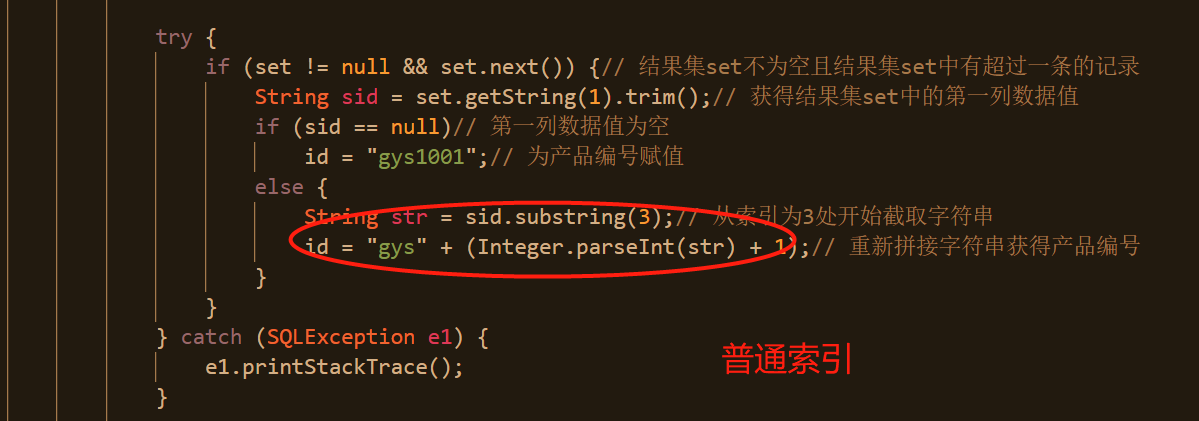
对于案例数据库进行优化及索引设计。

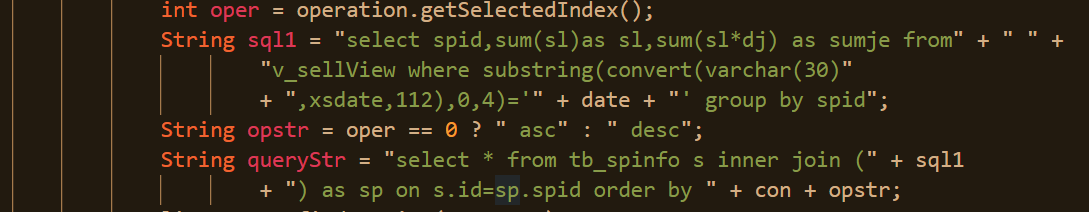
本次实验使用自己的数据库进行优化以及索引查询设计

索引设计大概有一下：

唯一索引，主键索引，组合索引，空间索引。

代码截图如下：





以及建立相应的数据库缓冲池如下图：



进行时间戮的样式转换如下图：

