第六章 查询性能优化

#### 一．为什么查询速度会变慢

1. 真正重要的是响应时间  
 把查询看成任务，它由一系列子任务组成，每个子任务都会消耗一定时间。

对查询的优化，实际上是优化子任务：

要么消除子任务；

要么减少子任务的执行次数；

要么让子任务运行的更快；

2. 查询的生命周期

客户端🡪服务器🡪在服务器上解析🡪生成执行计划🡪**执行**🡪返回给客户端

其中执行是最重要的阶段：

执行包括：大量的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。

3. 查询花费的时间

* 网络
* CPU计算
* 生成统计信息
* 执行计划
* 锁等待
* 等操作

尤其是向底层存储引擎检索数据的调用操作，这些调用需要在内存操作、CPU操作和内存不足时导致的I/O操作上消耗时间。可能会产生大量的上下文切换以及系统调用。

优化查询就是要减少这些时间。

#### 二．慢查询基础：优化数据访问

1. 查询性能低下最基本的原因是访问的数据太多。

2. 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。

3. 分析步骤：

（1）应用程序是否在检索大量超过需要的数据。访问太多行或者太多列。

（2）确认MySQL服务器层是否在分析大量超过需要的数据行。

4. 关键问题

* 是否向数据库请求了不需要的数据

1. 查询不需要的记录

可以使用limit避免

1. 多表关联时返回全部列

select \* …

1. 总是取出全部列

select \* …

1. 重复查询相同的数据

需要的时候，使用缓存

* MySQL是否在扫描额外的记录

查看以下指标

1. 响应时间
2. 扫描的行数
3. 返回的行数

如果发现查询需要扫描大量的数据但只返回少数的行，那么通常可以尝试下面的技巧去优化它：

1. 使用索引覆盖扫描，把所有需要用到的列都放到索引中，这样存储引擎不需要回表查询。
2. 改变库表结构，使用汇总表。
3. 重写这个复杂的查询。

#### 三．重构查询的方式

1. 一个复杂查询还是多个简单查询

多个小查询不是问题：

MySQL从设计上让连接和断开连接都很轻量级，在返回一个小的查询结果方面很高效。每秒上千次，上万数十万次的查询都能够满足。

一个查询能解决的尽量不要多次查询，但是要衡量哪个更优。

2. 切分查询

有时候对于一个大的查询我们需要分而治之，将大查询切分成小查询，每个查询功能一样，只完成一小部分，每次只返回一小部分结果。

一次性删除大量数据的时候，会锁住很多数据、占满整个事务日志、耗尽系统资源、阻塞很多小的但重要的查询。

将大的delete语句拆分成小的会尽可能小的影响MySQL性能，同时可以减少MySQL复制的延迟。

3. 分解关联查询

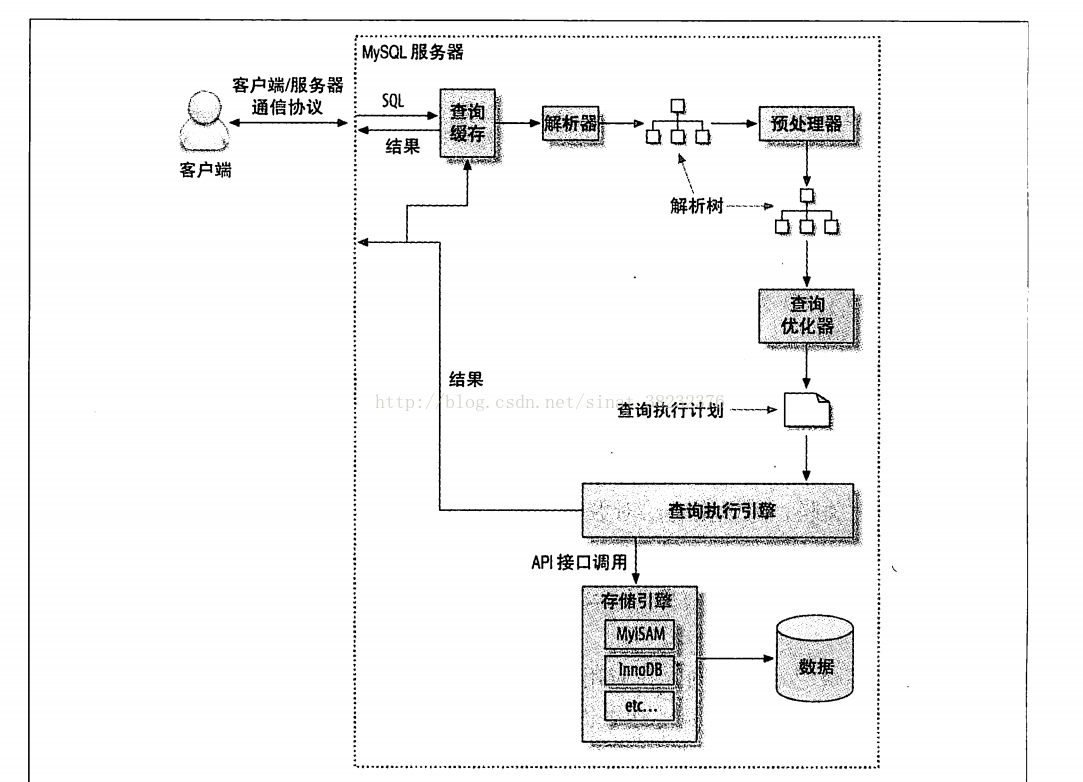
优点：

* 使缓存效率更高
* 单个查询可以减少锁的竞争
* 更容易对数据库拆分，更容易做到高性能、可扩展
* 减少冗余数据

应用层做关联：某条应用记录只需要查找一次

数据库做关联：可能需要重复访问数据

#### 四．查询执行的基础



1. 查询过程

1. 客户端发送一条查询给服务器
2. 服务器先检查查询缓存，如果命中了缓存，则立刻返回存储在缓存中的结果。否则进入下一阶段。
3. 服务器端进行SQL解析、预处理，再由优化器生成对应的执行计划。
4. MySQL根据优化器生成的执行计划，调用存储引擎的API来执行查询。
5. 将结果返回给客户端。

2. MySQL客户端/服务器通信协议

半双工：任意时刻，只能由一侧向另一侧发数据，不能同时发生。

（1） 查询状态：

* Sleep：线程在等待客户端发送新的请求。
* Query：线程正在执行查询或者正在将结果发送给客户端。
* Locked：线程正在等待表锁。
* Analyzing and statistics：线程正在收集存储引擎的统计信息，并生成查询的执行计划。
* Copying to tmp table [on disk]：线程正在执行查询，并且将结果复制到一个临时表中，这种状态表示在执行group by或者文件排序或者union操作。
* Sorting result：线程正在对结果集进行排序。
* Sending data：多种情况，传送数据、生成结果集、向客户端返回数据。

（2） 查询缓存：

在解析语句之前，如果缓存是打开的，那么MySQL会优先检查这个查询是否命中查询缓存中的数据。查询语句和缓存中的查询即使只有一个字节不同，那也不会匹配缓存结果。

如果命中缓存，在返回结果前MySQL会检查一次用户权限。权限正确则直接返回数据。

3. 查询优化处理

（1）语法解析和预处理

MySQL通过关键字将SQL语句进行解析，生成解析树。解析器会验证语法规则和解析查询。例如，关键字是否正确。

（2）查询优化器

一条语句可能有多种执行方式，最后返回相同的结果。优化器就是找到其中最优的执行计划。

最优不代表最快。

（3）排序优化

#### 四．查询优化器的局限性

1. 关联子查询

where in 的子查询，

并不是先查内部🡪再查外部

而是将外部的查询压到子查询中

尽量不要用in子查询。

替代方法：

（1）用group\_concat()在in()中构造一个由逗号分隔的列表。

（2）使用exists()通常会比in()更快

（3）使用连接（join）重写in子查询

2. UNION的限制

UNION只取前20条的时候，两个查询会分别把所有的记录都存放到一个临时表中，然后再从临时表中取出前20条；

优化：分别在两个查询中用limit

3. 索引合并优化

4. 最大值和最小值优化

Select min(actor\_id) from sakila.actor where first\_name = ‘PENE’;

因为first\_name没有索引，所以会进行全表扫描

但是Mysql主键严格按照大小顺序排列，所以改写如下

Select actor\_id from sakila.actor use index(primary) where first\_name = ‘PENE’ limit 1;

5. 优化特定类型的查询

* 优化count()查询

Count有两个作用：

（1）统计某个列值的数量

（2）统计表的行数：count(\*)最好，加where时尽量扫描较少的行数。

优化1：select count(\*) from world.city where id>5

扫描过多，可以改成下面的

select (select count(\*) from world.city) – count(\*) from world.city where id < 5;

优化2：查询同一列的不同值的数量

select count(color=’blue’ or null) as blue,count(color=’red’ or null) as red from item;

（3）

* 优化关联查询
* 优化子查询
* 优化group by和distinct
* 优化limit分页
* 优化union查询