**mysql 缓存机制**

2018年05月23日 12:08:35 [qzqanlhy1314](https://me.csdn.net/qzqanzc) 阅读数：5094

mysql缓存机制就是缓存sql 文本及缓存结果，用KV形式保存再服务器内存中，如果运行相同的sql,服务器直接从缓存中去获取结果，不需要在再去解析、优化、执行sql。 如果这个表修改了，那么使用这个表中的所有缓存将不再有效，查询缓存值得相关条目将被清空。表中得任何改变是值表中任何数据或者是结构的改变，包括insert,update,delete,truncate,alter table,drop table或者是drop database 包括那些映射到改变了的表的使用merge表的查询，显然，者对于频繁更新的表，查询缓存不合适，对于一些不变的数据且有大量相同sql查询的表，查询缓存会节省很大的性能。

**命中条件**

缓存存在一个hash表中，通过查询SQL，查询数据库，客户端协议等作为key,在判断命中前，mysql不会解析SQL，而是使用SQL去查询缓存，SQL上的任何字符的不同，如空格，注释，都会导致缓存不命中。如果查询有不确定的数据like now(),current\_date()，那么查询完成后结果者不会被缓存，包含不确定的数的是不会放置到缓存中。

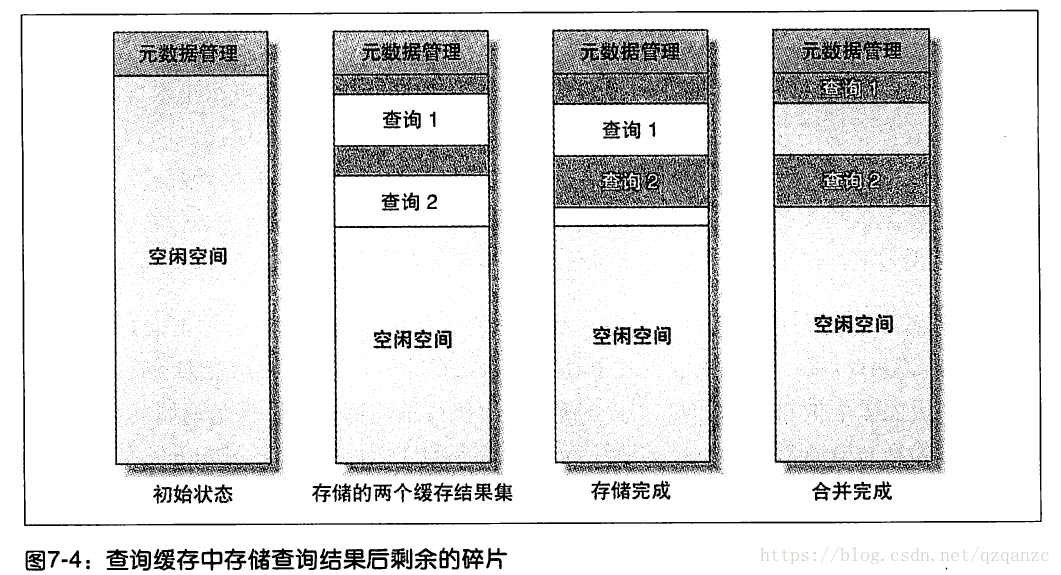
**工作流程**

1.服务器接收SQL，以SQL和一些其他条件为key查找缓存表   
2.如果找到了缓存，则直接返回缓存   
3.如果没有找到缓存，则执行SQL查询，包括原来的SQL解析，优化等。   
4.执行完SQL查询结果以后，将SQL查询结果缓存入缓存表

**缓存失败**

当某个表正在写入数据，则这个表的缓存（命中缓存，缓存写入等）将会处于失效状态，在Innodb中，如果某个事务修改了这张表，则这个表的缓存在事务提交前都会处于失效状态，在这个事务提交前，这个表的相关查询都无法被缓存。

**缓存的内存管理**

缓存会在内存中开辟一块内存（query\_cache\_size）来维护缓存数据，其中大概有40K的空间是用来维护缓存数据的元数据的，例如空间内存，例如空间内存，数据表和查询结果映射，SQL和查询结果映射的。   
mysql将这个大内存块分为小内存块（query\_cache\_min\_res\_unit),每个小块中存储自身的类型、大小和查询结果数据，还有前后内存块的指针。   
mysql需要设置单个小存储块大小，在SQL查询开始（还未得到结果）时就去申请一块内存空间，所以即使你的缓存数据没有达到这个大小也需要这个大小的数据块去保存（like linux filesystem’s block)。如果超出这个内存块的大小，则需要再申请一个内存块。当查询完成发现申请的内存有富余，则会将富余的内存空间是放点，这就会造成内存碎片的问题，见下图   


**缓存的使用时机**

衡量打开缓存是否对系统有性能提升是一个很难的话题

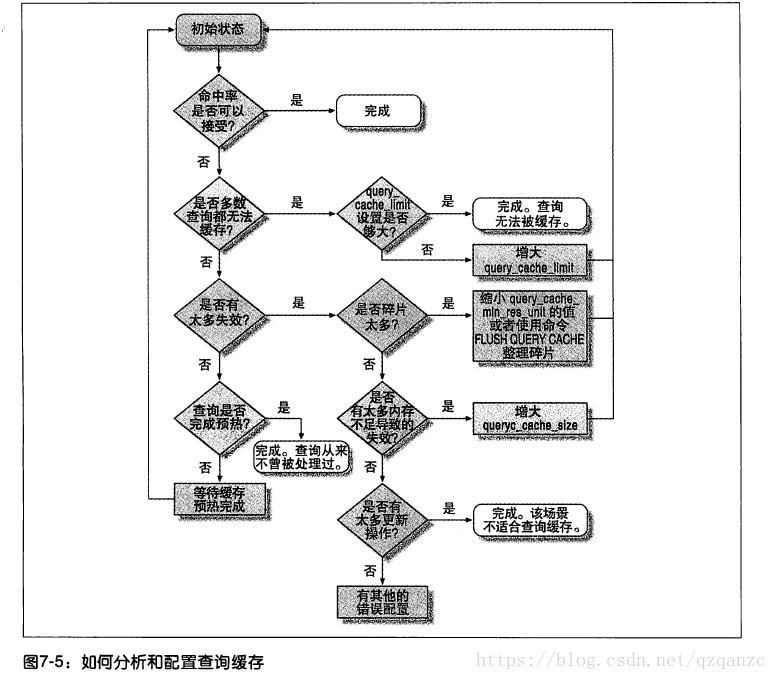
1. 通过缓存命中率判断, 缓存命中率 = 缓存命中次数 (Qcache\_hits) / 查询次数 (Com\_select)
2. 通过缓存写入率, 写入率 = 缓存写入次数 (Qcache\_inserts) / 查询次数 (Qcache\_inserts)
3. 通过 命中-写入率 判断, 比率 = 命中次数 (Qcache\_hits) / 写入次数 (Qcache\_inserts), 高性能MySQL中称之为比较能反映性能提升的指数,一般来说达到3:1则算是查询缓存有效,而最好能够达到10:1

**缓存参数配置**

1. query\_cache\_type: 是否打开缓存   
   可选项   
   1) OFF: 关闭   
   2) ON: 总是打开   
   3) DEMAND: 只有明确写了SQL\_CACHE的查询才会吸入缓存
2. query\_cache\_size: 缓存使用的总内存空间大小,单位是字节,这个值必须是1024的整数倍,否则MySQL实际分配可能跟这个数值不同(感觉这个应该跟文件系统的blcok大小有关)
3. query\_cache\_min\_res\_unit: 分配内存块时的最小单位大小
4. query\_cache\_limit: MySQL能够缓存的最大结果,如果超出,则增加 Qcache\_not\_cached的值,并删除查询结果
5. query\_cache\_wlock\_invalidate: 如果某个数据表被锁住,是否仍然从缓存中返回数据,默认是OFF,表示仍然可以返回

GLOBAL STAUS 中 关于 缓存的参数解释:   
Qcache\_free\_blocks: 缓存池中空闲块的个数   
Qcache\_free\_memory: 缓存中空闲内存量   
Qcache\_hits: 缓存命中次数   
Qcache\_inserts: 缓存写入次数   
Qcache\_lowmen\_prunes: 因内存不足删除缓存次数   
Qcache\_not\_cached: 查询未被缓存次数,例如查询结果超出缓存块大小,查询中包含可变函数等   
Qcache\_queries\_in\_cache: 当前缓存中缓存的SQL数量   
Qcache\_total\_blocks: 缓存总block数

**减少碎片策略**

1. 选择合适的block大小
2. 使用 FLUSH QUERY CACHE 命令整理碎片.这个命令在整理缓存期间,会导致其他连接无法使用查询缓存   
   PS: 清空缓存的命令式 RESET QUERY CACHE   
   

**InnoDB与查询缓存**

Innodb会对每个表设置一个事务计数器,里面存储当前最大的事务ID.当一个事务提交时,InnoDB会使用MVCC中系统事务ID最大的事务ID跟新当前表的计数器.   
只有比这个最大ID大的事务能使用查询缓存,其他比这个ID小的事务则不能使用查询缓存.   
另外,在InnoDB中,所有有加锁操作的事务都不使用任何查询缓存   
查询必须是完全相同的(逐字节相同)才能够被认为是相同的。另外，同样的查询字符串由于其它原因可能认为是不同的。使用不同的数据库、不同的协议版本或者不同 默认字符集的查询被认为是不同的查询并且分别进行缓存。