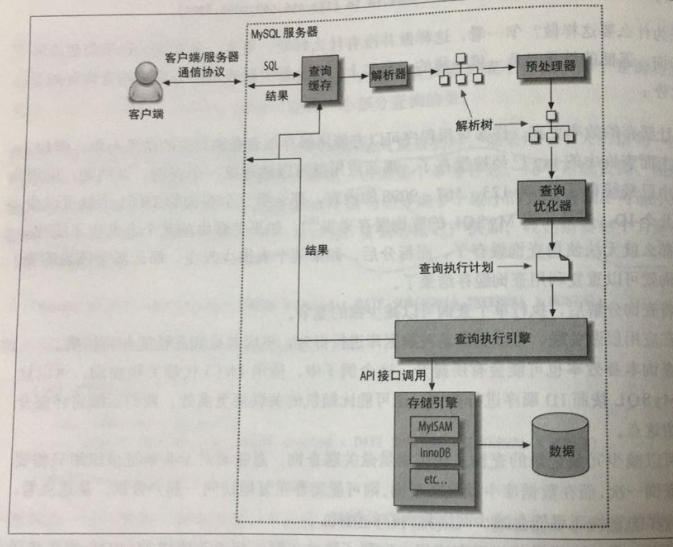
# [MySQL 执行过程与查询缓存](https://segmentfault.com/a/1190000018713591)

MySQL执行一个查询过程:  
当我们向MySQL发送一个请求的时候，MySQL到底做了什么：



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。

mysql 主要是由 server 层和存储层两部分构成的。  
server 层主要包括连接器、查询缓存，分析器、优化器、执行器。  
存储层主要是用来存储和查询数据的，常用的存储引擎有 InnoDB、MyISAM，

（1） MySQL客户端/服务器通信协议

MySQL客户端和服务器之的通信协议是“半双工”的，这就意味着，在任何一个时刻,要么是由服务器向客户端发送数据,要么是由客户端向服务器发送数据,这两个动作不能同时发生。所以我们无法也无须将一个消息切成小块独立来发送。

优缺点：  
这种协议让MySQL通信简单快速,但是也从很多地方限制了 MySQL。  
一个明显的限制是,这意味着没法进行流量控制。一旦一端开始发送消息,另一端要接收完整个消息才能响应它。这就像采回抛球的游戏:在任何时刻,只有一个人能控制球,而且只有控制球的人才能将球抛回去(发送消息)。

（2）.连接器

MySQL客户端和服务端建立连接,获取当前连接用户的权限

（3）查询缓存  
在解析一个查询语句之前,如果查询缓存是打开的，MySQL会检查这个缓存，是否命中查询缓存中的数据。这个检查是通过一个大小写敏感的哈希查找实现的。  
查询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,这种情况下查询  
就会进入下一阶段的处理。

如果当前的查询恰好命中了查询缓存,那么在返回查询结果之前 MySQL会检查一次用  
户权限。这仍然是无须解析查询SQL语句的,因为在查询缓存中已经存放了当前查询需  
要访问的表信息。如果权限没有问题, MySQL会跳过所有其他阶段,直接从缓存中拿  
到结果并返回给客户端。这种情况下,查询不会被解析,不用生成执行计划,不会被执行.

ps:注意在 mysql8 后已经没有查询缓存这个功能了，因为这个缓存非常容易被清空掉，命中率比较低。

(3).分析器

既然没有查到缓存，就需要开始执行 sql 语句了，在执行之前肯定需要先对 sql 语句进行解析。

分析器主要对 sql 语句进行语法和语义分析，检查单词是否拼写错误，还有检查要查询的表或字段是否存在

(4)查询优化

查询的生命周期的下一步是将一个SQL转换成一个执行计划, MySQL再依照这个执行  
计划和存储引擎进行交互。这包括多个子阶段:解析SQL、预处理、优化SQ执行计划。  
这个过程中任何错误(例如语法错误)都可能终止查询。

2.关于查询缓存

（1）  
MySQL 判断缓存命中的方法很简单：缓存存放在一个引用表中，通过一个哈希值引用。  
MySOL查询缓存保存查询返回的完整结果。当查询命中该缓存, MySQL会立刻返回结果跳过了 解析，优化和执行阶段

查询缓存系统会跟踪查迫中涉及的每个表,如果这些表发生变化,那么和这个表相关的的存数据都将失效。

这种机制效率看起来比较低,因为数据表变化时很有可能对查询结果并没有变更,但是这种简单实现代价很小,而这点对于一个非常繁忙的系统来说非常重要。

查询缓存系统对应用程序是完全透明的。应用程序无须关心 MySQL是通过查询缓存返回的结果还是实际执行返回的结果。事实上,这两种方式执行的结果是完全相同的。换句话说,查询缓存无须使用任何语法。无论是 MYSQL开启成关闭查询缓在,对应用程序都是透明的。

（2）判断缓存命中

当判断缓存是否命中时, MySQL不会解析、“正规化”或者参数化查询语句,而是直接使用SQL语句和客户端发送过来的其他原始信息,在字符上不同,例如空格、注释，在何的不同，都会导致缓存的不中。

当查询语句中有一些不确定的数据时，则不会被缓存，例如包含函数NOW()或者 CURRENT\_DATE()  
的查询不会被缓存.

误区：  
我们常听到:“如果查询中包含一个不确定的函数, MySQL则不会检查查询缓存”。这个说法是不正确的。

因为在检查查询缓存的时候,还没有解析SQL语句,所以MySQL并不知道查询语句中是否包含这类函数。

在检查查询缓存之前, MySQL只做一件事情，就是通过一个大小写不敏感的检查看看SQL语句是不是以5EL开头。

准确的说法应该是:“如果查询语句中包含任何的不确定函数,那么在查询缓存中是不可能找到缓存结果的”。

注意点：  
MySQL的查询缓存在很多时候可以提升查询性能,在使用的时候,有一些问题需要特别注意。首先,打开查询缓存对读和写操作都会带来额外的消耗:

1.读查询在开始之前必须先检查是否命中缓存  
2.如果这个读查询可以被缓存,那么当完成执行后, MySQL若发现查询缓存中没有这个查询,会将其结果存入查询缓存,这会带来额外的系统消耗。  
3.这对写操作也会有影响,因为当向某个表写入数据的时候, MySQL必须将对应表的所有缓存都设置失效。如果查询缓存非常大或者碎片很多,这个操作就可能会带来大系统消耗(设置了很多的内存给查询缓存用的时候)

如果查询缓存使用了很大量的内存,缓存失效操作就可能成为一个非常严重的问题瓶颈  
如果缓存中存放了大量的查询结果,那么缓存失效操作时整个系统都可能会僵死一会

因为这个操作是靠一个全局锁操作保护的,所有需要做该操作的查询都要等待这个锁,  
而且无论是检测是否命中缓存、还是缓存失效检测都需要等待这个全局锁。

（3）什么情况下查询缓存能发挥作用  
理论上，可以通过观察打开或者关闭查询缓存时候的系统效率来决定是否需要开启查询。

对手那些需要消耗大量资源的查询通常都是非常适合缓存的。  
例如一些汇总计算查询具体的如 COUNT()等。总地来说,对于复杂的 SELECT语句都可以使用查询缓存,  
例如多表JOIN后还需要做排序和分页,这类查询每次执行消耗都很大,但是返回的结果集却很小,非常适合查询缓存。

不过需要注意的是,涉及的表上 UPDATE、 DELETE和 INSERT操作相比 SELECT来说要非常少才行。

判断查询缓存是否有效的直接数据是命中率。就是使用查询缓存返回结果占总查询的比率

不过缓存中率是一个很难判断的数值。命中率多大才是好的命中率。具体情况，具体分析。

只要查询缓存带来的效率提升大于查询缓存带来的额外消耗,即使30%命中率对系统性能提升也有很大好处。另外,缓存了哪些查询也很重要,例如,被缓存的查询本身消耗非常巨大,那么即使缓存命中率非常低,也仍然会对系统性能提升有好处

缓存未命中可能有如下几种原因:

1.查询语句无法被缓存,可能是因为查询中包含一个不确定的函数(如 CURREN\_DATE)或者查询结果太大而无法缓存。这都会导致状态值 Cache not cached增加。  
2.MySQL从未处理这个查询,所以结果也从不曾被缓存过。

3.还有一种情况是虽然之前缓存了查询结果,但是由于查询缓存的内存用完了,MySQL需要将某些缓存“逐出”,或者由于数据表被修改导致缓存失效。

如果你的服务器上有大量缓存未命中,但是实际上绝大数查询都被缓存了,那么一定是有如下情况发生：

1.查询缓存还没有完成预热。也就是说, MySQL还没有机会将查询结果都缓存起来。  
2.查询语句之前从未执行过。如果你的应用程序不会重复执行一条查询语句,那么即使完成预热仍然会有很多缓存未命中  
3.缓存失效操作太多了。

（4）如何配置 和维护查询缓存

query\_cache\_type

是否打开查询缓存。可以设置成0FN或 DEMAND。 DEMAND表示只有在查询语句中明确写明SQL\_ CACHE的语句才放入查询缓存。这个变量可以是会话级别的也可以是全局级别的

query\_cache\_size

查询缓存使用的总内存空间,单位是字节。这个值必须是1024的整数倍,否则 MySQL实际分配的数据会和你指定的略有不同。

query\_cahce\_min\_res\_unit

在查询缓存中分配内存块时的最小单位。

query\_chache\_limit

MySQL能够缓存的最大査询结果。如果查询结果大于这个值,则不会被缓存。因为査询缓存在数据生成的时候就开始尝试缓存数据,所以只有当结果全部返回后,才知道查询结果是否超出限制

如果超出, MySQL则增加状态值 Cache\_not\_cached,并将结果从查询缓存中删除如果你事先知道有很多这样的情况发生,那么建议在查询语句中加入

（5）替代方案

MySQL查询缓存工作的原则是:执行查询最快的方式就是不去执行,但是查询仍然需要发送到服务器端,服务器也还需要做一点点工作。如果对于某些查询完全不需要与服务器通信效果会如何呢?这时客户端的缓存可以很大程度上帮你分担 MySQL服务器的压力

总结：

完全相同的查询在重复执行的时候,查询缓存可以立即返回结果,而无须在数据库中重新执行一次。根据我们的经验,在高并发压力环境中在询缓存会导致系统性能的下降,甚至僵死。

如果一定要使用查询缓存,那么不要设置太大内存,而且只有在确收益的时候才使用。

那该如何判断是否应该使用查询缓存呢?建议使Percona server.,观察更细致的日志,并做一些简单的计算。还可以查看缓存命中率(并不总是有用)、“ NSERTS和 SELECT比率”(这个参数也并不直观)、或者“命中和写入比率”(这个参考意义较大)。

查询缓存是一个非常方便的缓存,对应用程序完全透明,无须任何额外的编码,但是、如果希望有更高的缓存效率,我们建议使cache 或者其他类似的解决方案。