1. 目标

- 了解什么是优化
- 掌握优化查询的方法
- 掌握优化数据库结构的方法
- 掌握优化 MySQL 服务器的方法

2. 什么是优化?

- 合理安排资源、调整系统参数使 MySQL 运行更快、更节省资源。
- 优化是多方面的，包括查询、更新、服务器等。
- 原则：减少系统瓶颈，减少资源占用，增加系统的反应速度。

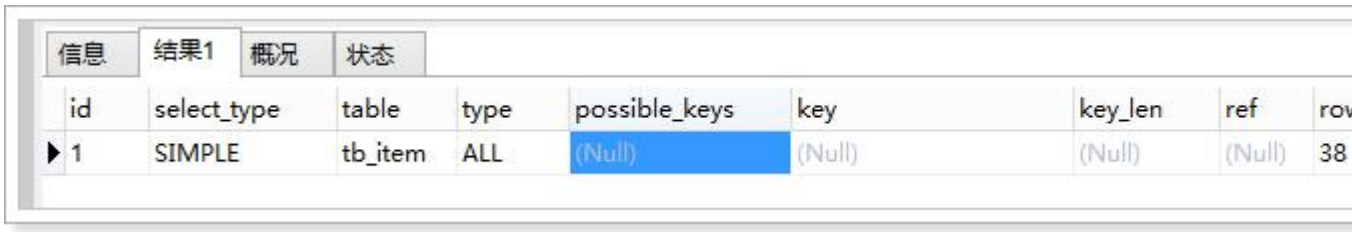
3. 数据库性能参数

- 使用 SHOW STATUS 语句查看 MySQL 数据库的性能参数
  - SHOW STATUS LIKE 'value'
- 常用的参数：
  - Slow_queries　慢查询次数
  - Com_(CRUD) 操作的次数
  - Uptime　上线时间

4. 查询优化

4.1. EXPLAIN

在 MySQL 中可以使用 EXPLAIN 查看 SQL 执行计划，用法：EXPLAIN SELECT * FROM tb_item



4.2. 结果说明

4.2.1. id

SELECT 识别符。这是 SELECT 查询序列号。这个不重要。

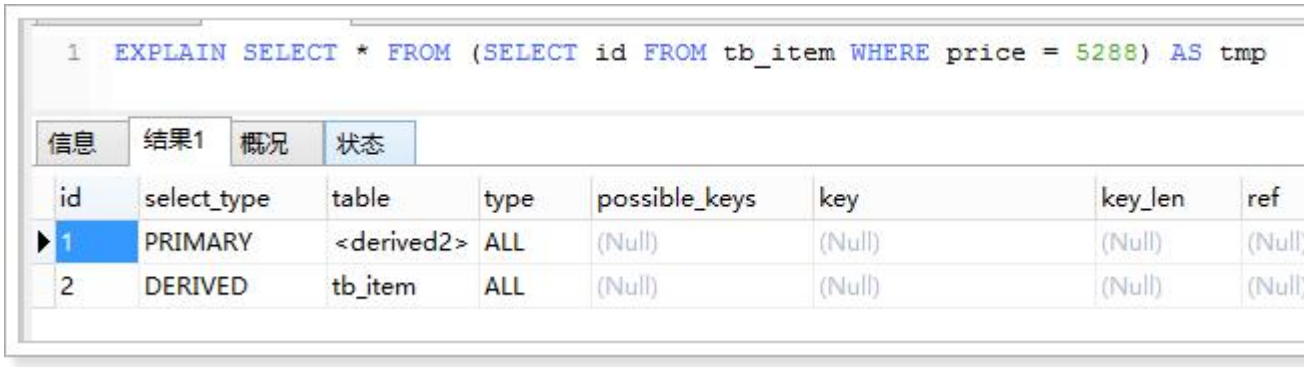
4.2.2. select_type

表示 SELECT 语句的类型。

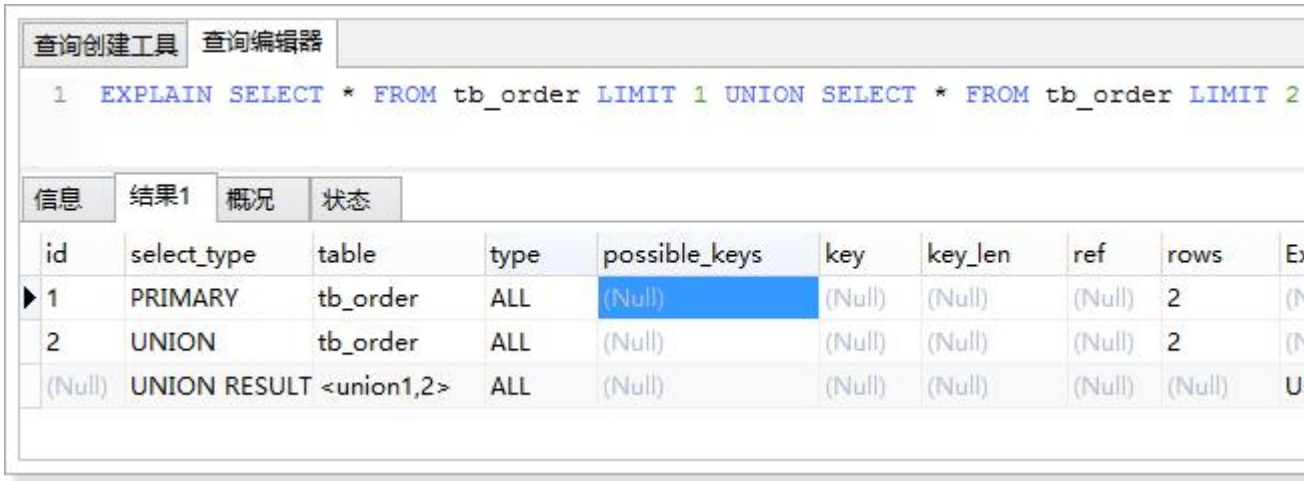
有以下几种值:

1、 SIMPLE
表示简单查询，其中不包含连接查询和子查询。

2、 PRIMARY
表示主查询，或者是最外面的查询语句。



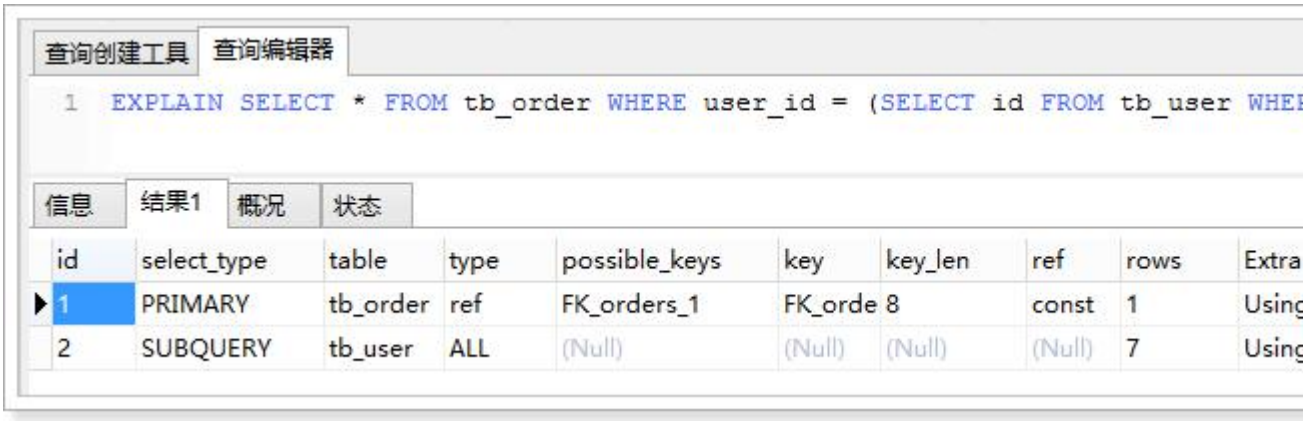
3、 UNION
表示连接查询的第 2 个或后面的查询语句。



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

5、 UNION RESULT
连接查询的结果。

6、SUBQUERY
子查询中的第 1 个 SELECT 语句。



7、DEPENDENT SUBQUERY
子查询中的第 1 个 SELECT 语句，取决于外面的查询。

8、DERIVED
SELECT(FROM 子句的子查询)。

4.2.3. table

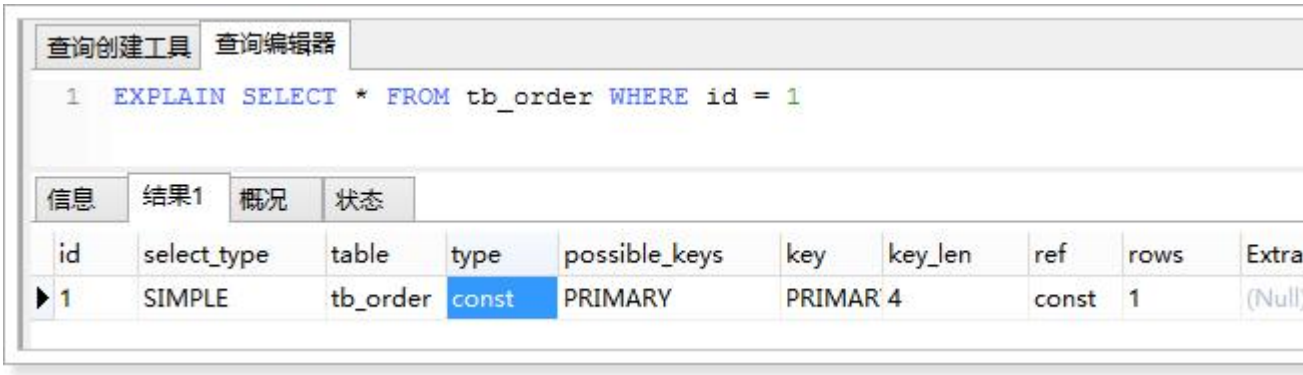
表示查询的表。

4.2.4. type（重要）

表示表的连接类型。
以下的连接类型的顺序是从最佳类型到最差类型：

1、system
表仅有一行，这是 const 类型的特列，平时不会出现，这个也可以忽略不计。

2、const
数据表最多只有一个匹配行，因为只匹配一行数据，所以很快，常用于 PRIMARY KEY 或者 UNIQUE 索引的查询，可理解为 const 是最优化的。



3、eq_ref
mysql 手册是这样说的:"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。它用在一个索引的所有部分被联接使用并且索

引是 UNIQUE 或 PRIMARY KEY"。eq_ref 可以用于使用=比较带索引的列。

```
1  EXPLAIN SELECT * FROM tb_order o,tb_user u WHERE u.id = o.user_id
```

信息 | 结果1 | 概况 | 状态

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	o	ALL	FK_orders_1	(Null)	(Null)	(Null)	2	(Null)
1	SIMPLE	u	eq_ref	PRIMARY	PRIMAR	8	mybat	1	(Null)

4、 ref

查询条件索引既不是 UNIQUE 也不是 PRIMARY KEY 的情况。ref 可用于=或<或>操作符的带索引的列。

```
1  EXPLAIN SELECT * FROM tb_order WHERE order_number = '20140921001'
```

信息 | 结果1 | 概况 | 状态

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	tb_order	ref	order_number	order_n	62	const	1	Using

5、 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。在解决子查询中经常使用该联接类型的优化。

上面这五种情况都是很理想的索引使用情况。

6、 index_merge

该联接类型表示使用了索引合并优化方法。在这种情况下，key 列包含了使用的索引的清单，key_len 包含了使用的索引的最长的关键元素。

7、 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)

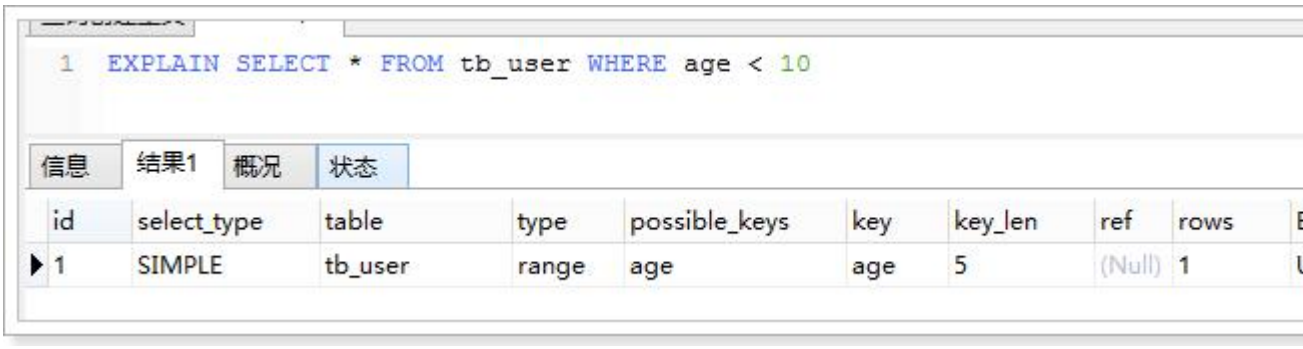
unique_subquery 是一个索引查找函数,可以完全替换子查询,效率更高。

8、 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)

9、 range

只检索给定范围的行,使用一个索引来选择行。



10、 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。

11、 ALL

对于每个来自于先前的表的行组合,进行完整的表扫描。(性能最差)

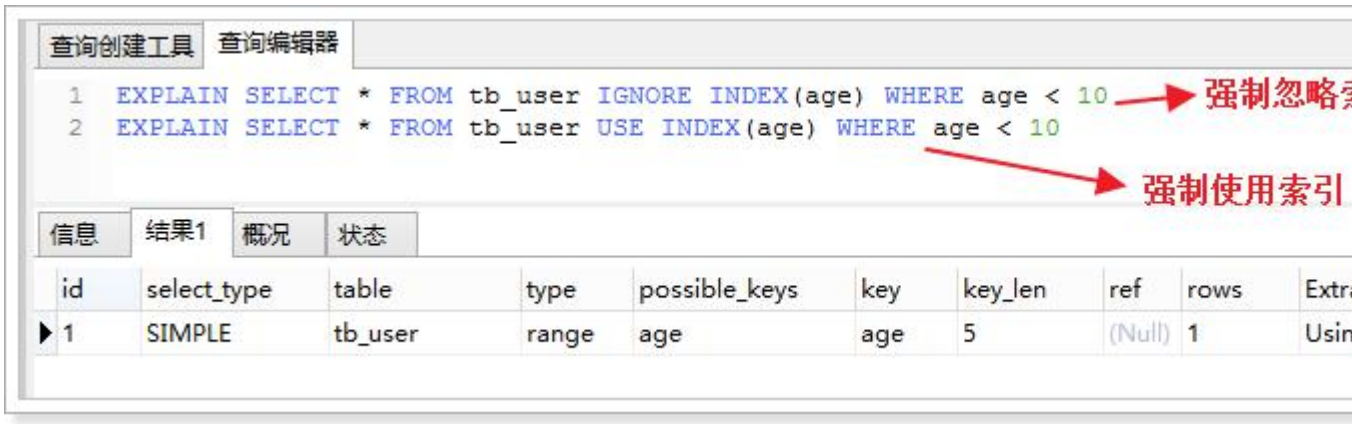
4.2.5. possible_keys

指出 MySQL 能使用哪个索引在该表中找到行。
如果该列为 NULL，说明没有使用索引，可以对该列创建索引来提高性能。

4.2.6. 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。

可以强制使用索引或者忽略索引:



4.2.7. key_len

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

注意：key_len 是确定了 MySQL 将实际使用的索引长度。

4.2.8. ref

显示使用哪个列或常数与 key 一起从表中选择行。

4.2.9. rows

显示 MySQL 认为它执行查询时必须检查的行数。

4.2.10. Extra

该列包含 MySQL 解决查询的详细信息

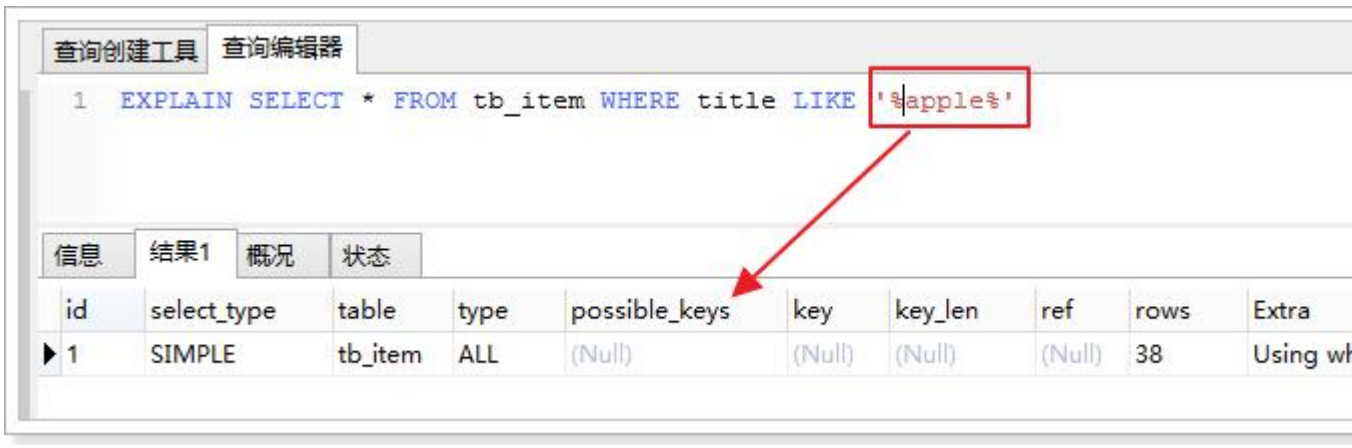
- Distinct:MySQL 发现第 1 个匹配行后,停止为当前的行组合搜索更多的行。
- Not exists:MySQL 能够对查询进行 LEFT JOIN 优化,发现 1 个匹配 LEFT JOIN 标准的行后,不再为前面的的行组合在该表内检查更多的行。
- range checked for each record (index map: #):MySQL 没有发现好的可以使用的索引,但发现如果来自前面的表的列值已知,可能部分索引可以使用。
- Using filesort:MySQL 需要额外的一次传递,以找出如何按排序顺序检索行。
- Using index:从只使用索引树中的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。
- 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。

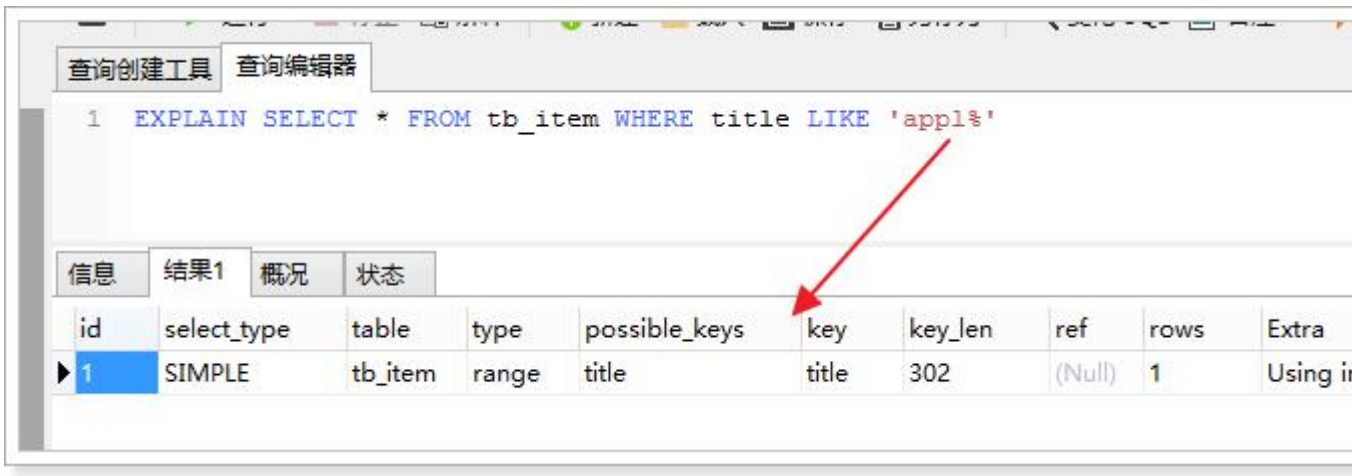
4.3. 使用索引查询需要注意

索引可以提供查询的速度，但并不是使用了带有索引的字段查询都会生效，有些情况下是不生效的，需要注意!

4.3.1. 使用 LIKE 关键字的查询

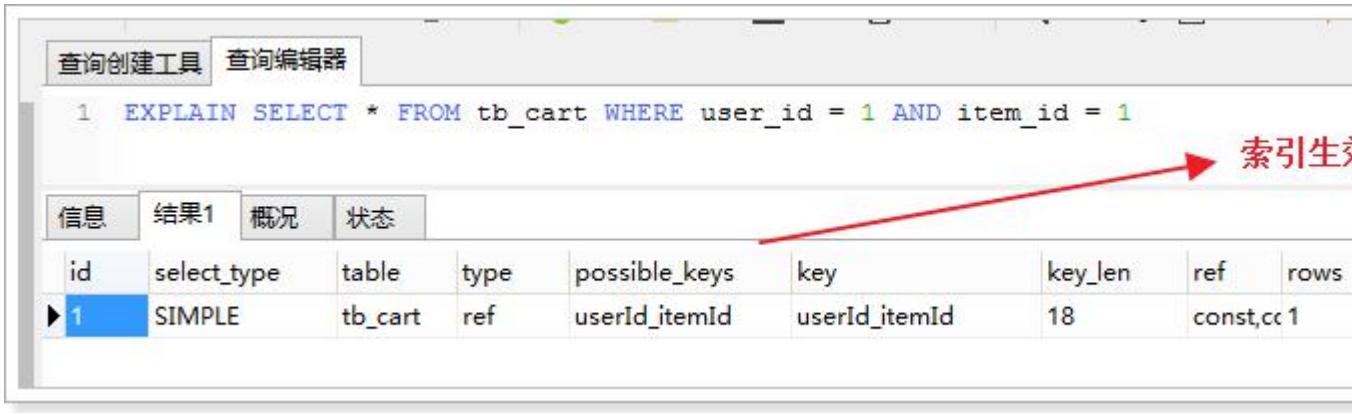
在使用 LIKE 关键字进行查询的查询语句中，如果匹配字符串的第一个字符为“%”，索引不起作用。只有“%”不在第一个位置，索引才会生效。

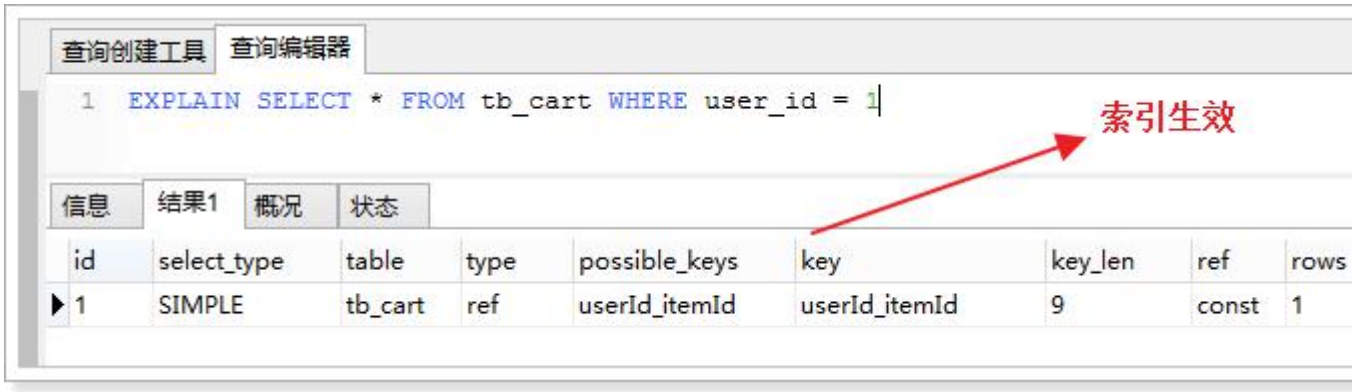


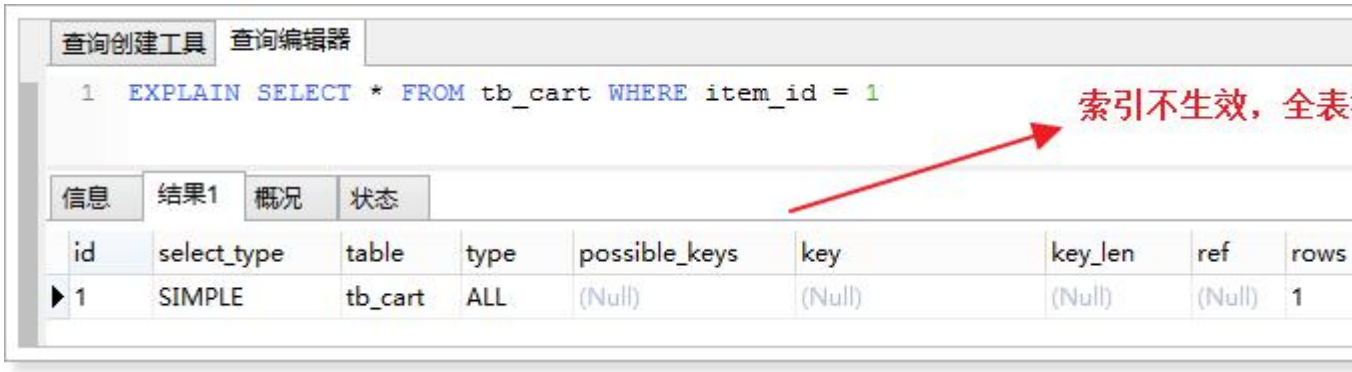


4.3.2. 使用联合索引的查询

MySQL 可以为多个字段创建索引，一个索引可以包括 16 个字段。对于联合索引，只有查询条件中使用了这些字段中第一个字段时，索引才会生效。

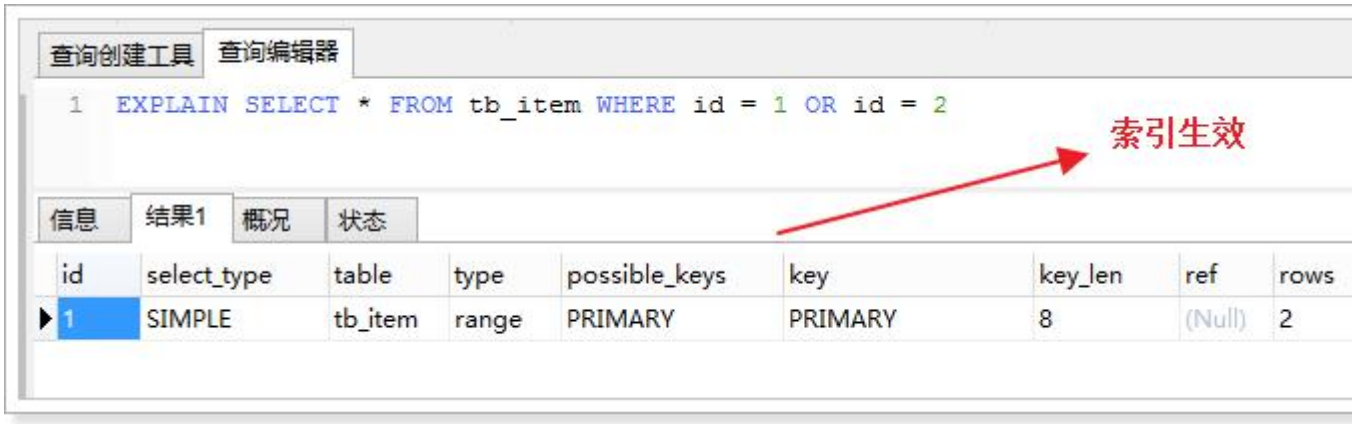


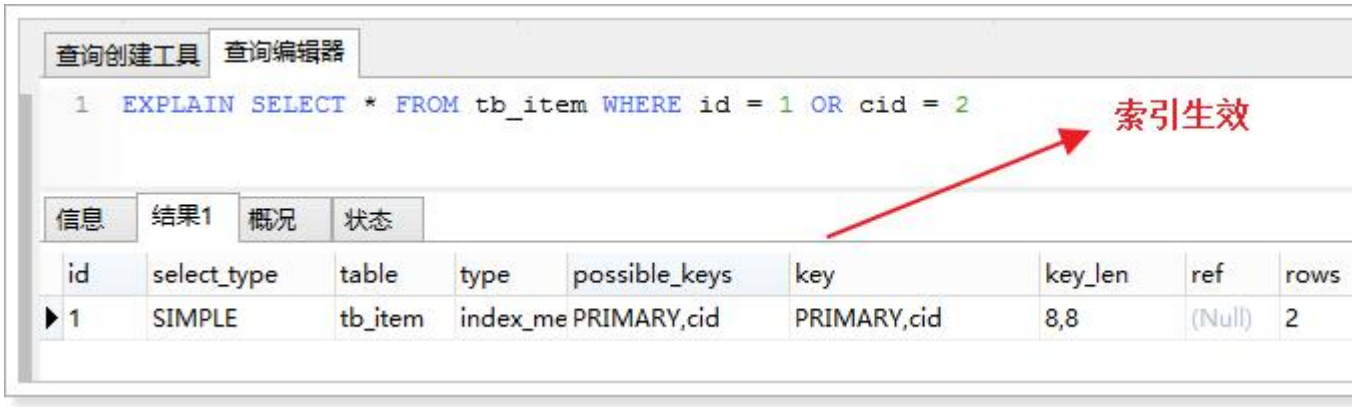


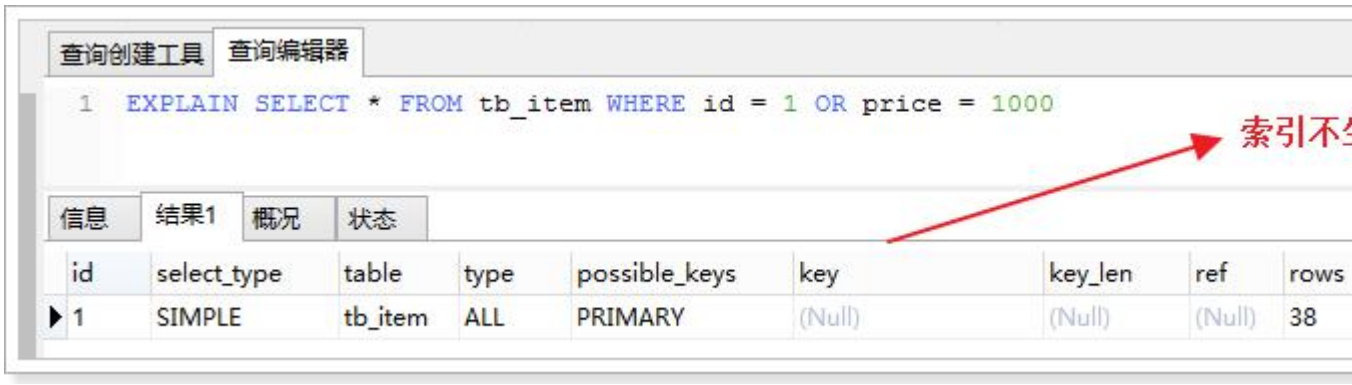


4.3.3. 使用 OR 关键字的查询

查询语句的查询条件中只有 OR 关键字，且 OR 前后的两个条件中的列都是索引时，索引才会生效，否则，索引不生效。







4.4. 子查询优化

MySQL 从 4.1 版本开始支持子查询，使用子查询进行 SELECT 语句嵌套查询，可以一次完成很多逻辑上需要多个步骤才能完成的 SQL 操作。

子查询虽然很灵活，但是执行效率并不高。

执行子查询时，MYSQL 需要创建临时表，查询完毕后再删除这些临时表，所以，子查询的速度会受到一定的影响。

优化：

可以使用连接查询（JOIN）代替子查询，连接查询时不需要建立临时表，其速度比子查询快。

5. 数据库结构优化

一个好的数据库设计方案对于数据库的性能往往会起到事半功倍的效果。

需要考虑数据冗余、查询和更新的速度、字段的数据类型是否合理等多方面的内容。

5.1. 将字段很多的表分解成多个表

对于字段较多的表，如果有些字段的使用频率很低，可以将这些字段分离出来形成新表。

因为当一个表的数据量很大时，会由于使用频率低的字段的存在而变慢。

5.2. 增加中间表

对于需要经常联合查询的表，可以建立中间表以提高查询效率。

通过建立中间表，将需要通过联合查询的数据插入到中间表中，然后将原来的联合查询改为对中间表的查询。

5.3. 增加冗余字段

设计数据表时应尽量遵循范式理论的规约，尽可能的减少冗余字段，让数据库设计看起来精致、优雅。但是，合理的加入冗余字段可以提高查询速度。

表的规范化程度越高，表和表之间的关系越多，需要连接查询的情况也就越多，性能也就越差。

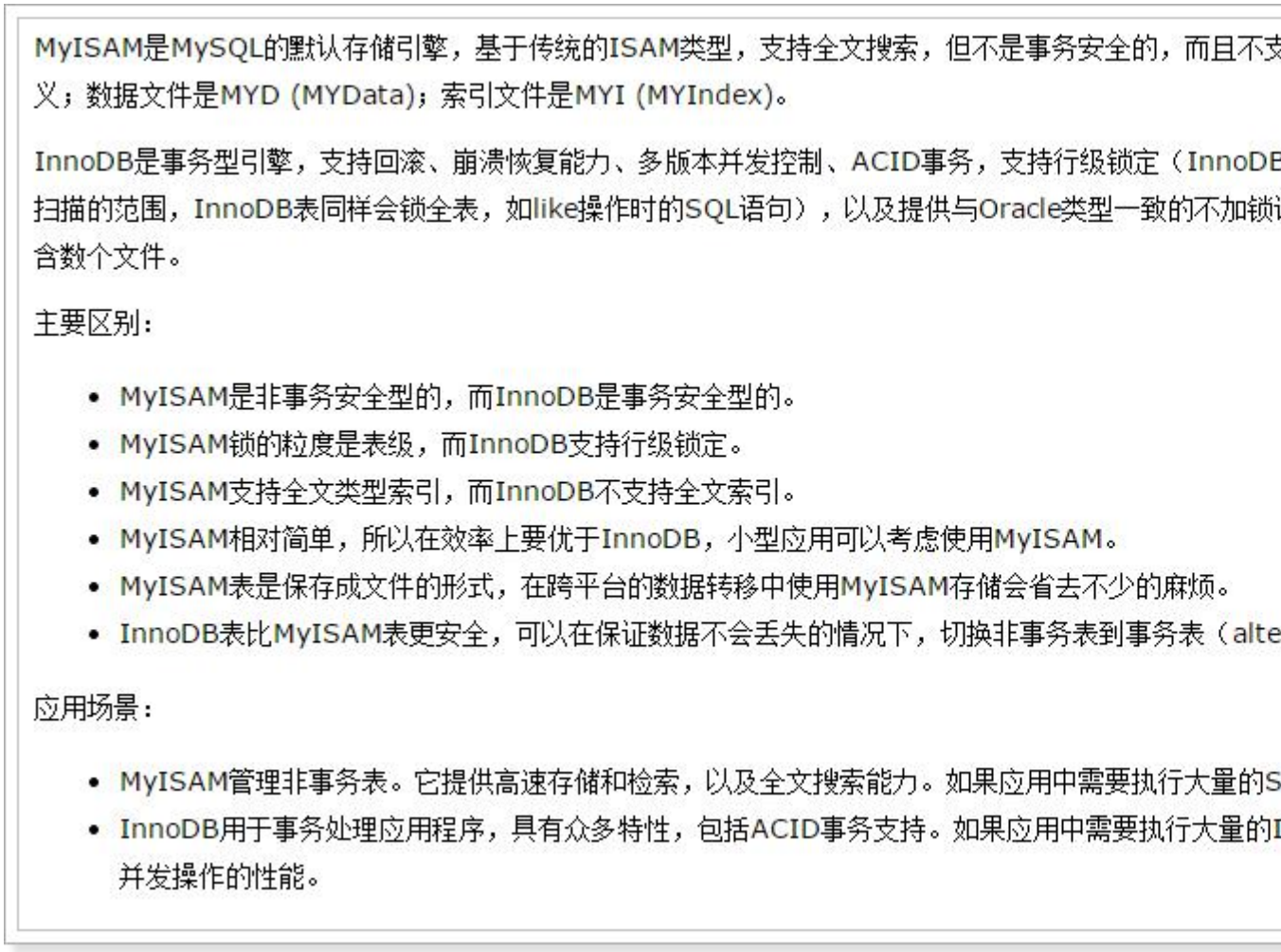
注意：
冗余字段的值在一个表中修改了，就要想办法在其他表中更新，否则就会导致数据不一致的问题。

6. 插入数据的优化

插入数据时，影响插入速度的主要是索引、唯一性校验、一次插入的数据条数等。

插入数据的优化，不同的存储引擎优化手段不一样，在 MySQL 中常用的存储引擎有，MyISAM 和 InnoDB，两者的区别：

http://www.cnblogs.com/panfeng412/archive/2011/08/16/2140364.html



MyISAM是MySQL的默认存储引擎，基于传统的ISAM类型，支持全文搜索，但不是事务安全的，而且不支
义；数据文件是MYD (MYData)；索引文件是MYI (MYIndex)。

InnoDB是事务型引擎，支持回滚、崩溃恢复能力、多版本并发控制、ACID事务，支持行级锁定（InnoDB
扫描的范围，InnoDB表同样会锁全表，如like操作时的SQL语句），以及提供与Oracle类型一致的不加锁
含数个文件。

主要区别：

- MyISAM是非事务安全型的，而InnoDB是事务安全型的。
- MyISAM锁的粒度是表级，而InnoDB支持行级锁定。
- MyISAM支持全文类型索引，而InnoDB不支持全文索引。
- MyISAM相对简单，所以在效率上要优于InnoDB，小型应用可以考虑使用MyISAM。
- MyISAM表是保存成文件的形式，在跨平台的数据转移中使用MyISAM存储会省去不少的麻烦。
- InnoDB表比MyISAM表更安全，可以在保证数据不会丢失的情况下，切换非事务表到事务表（alte

应用场景：

- 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。如果应用中需要执行大量的S
- InnoDB用于事务处理应用程序，具有众多特性，包括ACID事务支持。如果应用中需要执行大量的I
  并发操作的性能。

6.1. MyISAM

6.1.1. 禁用索引

对于非空表，插入记录时，MySQL 会根据表的索引对插入的记录建立索引。如果插入大量数据，建立索引会降低插入数据速度。

为了解决这个问题，可以在批量插入数据之前禁用索引，数据插入完成后再开启索引。

禁用索引的语句：
ALTER TABLE table_name DISABLE KEYS

开启索引语句：
ALTER TABLE table_name ENABLE KEYS

对于空表批量插入数据，则不需要进行操作，因为 MyISAM 引擎的表是在导入数据后才建立索引。

6.1.2. 禁用唯一性检查

唯一性校验会降低插入记录的速度，可以在插入记录之前禁用唯一性检查，插入数据完成后再开启。

禁用唯一性检查的语句：SET UNIQUE_CHECKS = 0;

开启唯一性检查的语句：SET UNIQUE_CHECKS = 1;

6.1.3. 批量插入数据

插入数据时，可以使用一条 INSERT 语句插入一条数据，也可以插入多条数据。

```
INSERT INTO tb_cart
VALUES
  (
    '40',
    '6',
    '36',
    '苹果（APPLE）iPhone 6 A1586 16G版 4G手机（深空灰）',
    'http://image.taotao.com/images/2014/10/23/20141023045240028070 9.jpg',
    '5288000',
    '1',
    '2014-12-11 18:07:46',
    '2014-12-12 16:52:55'
  );
INSERT INTO tb_cart
VALUES
  (
    '41',
    '6',
    '36',
    '苹果（APPLE）iPhone 6 A1586 16G版 4G手机（深空灰）',
    'http://image.taotao.com/images/2014/10/23/20141023045240028070 9.jpg',
    '5288000',
    '1',
    '2014-12-11 18:07:46',
    '2014-12-12 16:52:55'
  );

```

```sql
INSERT INTO tb_cart
VALUES
  (
    '40',
    '6',
    '36',
    '苹果（APPLE）iPhone 6 A1586 16G版 4G手机（深空灰）',
    'http://image.taotao.com/images/2014/10/23/20141023045240028070 9.jpg',
    '5288000',
    '1',
    '2014-12-11 18:07:46',
    '2014-12-12 16:52:55'
  )
,
  (
    '40',
    '6',
    '36',
    '苹果（APPLE）iPhone 6 A1586 16G版 4G手机（深空灰）',
    'http://image.taotao.com/images/2014/10/23/20141023045240028070 9.jpg',
    '5288000',
    '1',
    '2014-12-11 18:07:46',
    '2014-12-12 16:52:55'
  );

```

第二种方式的插入速度比第一种方式快。

6.1.4. 使用 LOAD DATA INFILE

当需要批量导入数据时，使用 LOAD DATA INFILE 语句比 INSERT 语句插入速度快很多。

6.2. InnoDB

6.2.1. 禁用唯一性检查

用法和 MyISAM 一样。

6.2.2. 禁用外键检查

插入数据之前执行禁止对外键的检查，数据插入完成后再恢复，可以提供插入速度。

禁用：SET foreign_key_checks = 0;
开启：SET foreign_key_checks = 1;

6.2.3. 禁止自动提交

插入数据之前执行禁止事务的自动提交，数据插入完成后再恢复，可以提高插入速度。

禁用：SET autocommit = 0;
开启：SET autocommit = 1;

7. 服务器优化

7.1. 优化服务器硬件

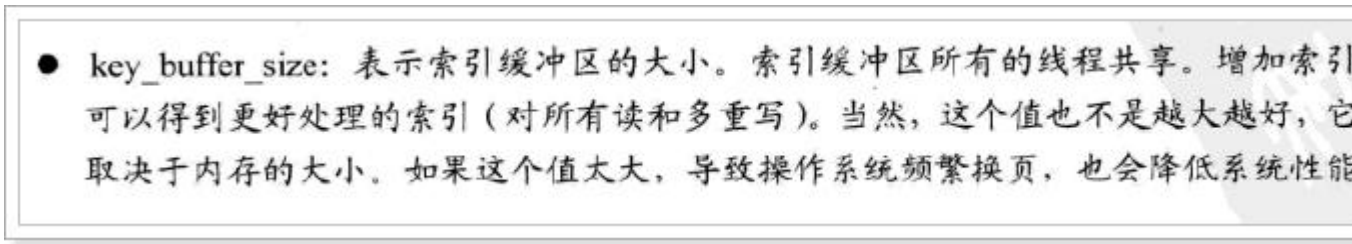
服务器的硬件性能直接决定着 MySQL 数据库的性能，硬件的性能瓶颈，直接决定 MySQL 数据库的运行速度和效率。

需要从以下几个方面考虑：
1、配置较大的内存。足够大的内存，是提高 MySQL 数据库性能的方法之一。内存的 IO 比硬盘快的多，可以增加系统的缓冲区容量，使数据在内存停留的时间更长，以减少磁盘的 IO。
2、配置高速磁盘，比如 SSD。
3、合理分配磁盘 IO，把磁盘 IO 分散到多个设备上，以减少资源的竞争，提高并行操作能力。
4、配置多核处理器，MySQL 是多线程的数据库，多处理器可以提高同时执行多个线程的能力。

7.2. 优化 MySQL 的参数

通过优化 MySQL 的参数可以提高资源利用率，从而达到提高 MySQL 服务器性能的目的。

MySQL 的配置参数都在 my.conf 或者 my.ini 文件的[mysqld]组中，常用的参数如下：



- table_cache：表示同时打开的表的个数。这个值越大，能够同时打开的表的个数越
个值不是越大越好，因为同时打开的表太多会影响操作系统的性能。
- query_cache_size：表示查询缓冲区的大小。该参数需要和 query_cache_type 配合使
query_cache_type 值是 0 时，所有的查询都不使用查询缓冲区。但是 query_cache_ty
不会导致 MySQL 释放 query_cache_size 所配置的缓冲区内存。当 query_cache_type
所有的查询都将使用查询缓冲区，除非在查询语句中指定 SQL_NO_CACHE，如 S
SQL_NO_CACHE * FROM tbl_name。当 query_cache_type=2 时，只有在查询语句
SQL_CACHE 关键字，查询才会使用查询缓冲区。使用查询缓冲区可以提高查询的
这种方式只适用于修改操作少且经常执行相同的查询操作的情况。
- sort_buffer_size：表示排序缓存区的大小。这个值越大，进行排序的速度越快。
- read_buffer_size：表示每个线程连续扫描时为扫描的每个表分配的缓冲区的大小（
当线程从表中连续读取记录时需要用到这个缓冲区。SET SESSION read_buffer_size
临时设置该参数的值。
- read_rnd_buffer_size：表示为每个线程保留的缓冲区的大小，与 read_buffer_size 不
主要用于存储按特定顺序读取出来的记录。也可以用 SET SESSION read_rnd_buffe
来临时设置该参数的值。如果频繁进行多次连续扫描，可以增加该值。
- innodb_buffer_pool_size：表示 InnoDB 类型的表和索引的最大缓存。这个值越大，
速度就会越快。但是这个值太大会影响操作系统的性能。

- max_connections：表示数据库的最大连接数。这个连接数不是越大越好，因为这些连接会浪费内存的资源。过多的连接可能会导致 MySQL 服务器僵死。
- innodb_flush_log_at_trx_commit：表示何时将缓冲区的数据写入日志文件，并且将日志文件写入磁盘中。该参数对于 innoDB 引擎非常重要。该参数有 3 个值，分别为 0、1 和 2。0 值为 0 时表示每隔 1 秒将数据写入日志文件并将日志文件写入磁盘；值为 1 时表示每次提交事务时将数据写入日志文件并将日志文件写入磁盘；值为 2 时表示每次提交事务时将数据写入日志文件，每隔 1 秒将日志文件写入磁盘。该参数的默认值为 1。默认值 1 的数据最高，但是每次事务提交或事务外的指令都需要把日志写入（flush）硬盘，是比较费时的。0 值更快一点，但安全方面比较差；2 值日志仍然会每秒写入到硬盘，所以即使出现故障，一般也不会丢失超过 1~2 秒的更新。
- back_log：表示在 mysql 暂时停止回答新请求之前的短时间内，多少个请求可以被存在堆栈中。换句话说，该值表示对到来的 Tcp/Ip 连接的侦听队列的大小。只有期望在一个短时间内有很多连接，才需要增加该参数的值。操作系统在这个队列大小上也有限制，将 back_log 高于操作系统的限制将是无效的。
- interactive_timeout：表示服务器在关闭连接前等待行动的秒数。
- sort_buffer_size：表示每个需要进行排序的线程分配的缓冲区的大小。增加这个参数可以提高 ORDER BY 或 GROUP BY 操作的速度。默认数值是 2 097 144（2MB）。
- thread_cache_size：表示可以复用的线程的数量。如果有很多新的线程，为了提高性能可以增大该参数的值。
- wait_timeout：表示服务器在关闭一个连接时等待行动的秒数。默认数值是 28 800

要求：必须记忆至少 3 个。