分析：

1.观察，至少跑一天，看看生产的慢sql情况。

2.开启慢查询日志，设置阈值，比如超过5秒的就是慢sql，并将它抓出来。

3.explain+慢sql分析

4.show profile

5.运维经理或者DBA，进行数据库服务器的参数调优。

==总结

1. 慢查询的开启并捕获。
2. Explain+慢查询分析
3. Show profile查询sql在mysql服务器里面的执行细节和生命周期情况。
4. Sql数据库服务器参数调优

**第一式explain：**

Select \* from a left join b on a.id = b.id；通常最好的建立索引方式是建立在b表的id上。

Select \* from b left join a on a.id = b.id；通常最好的建立索引方式是建立在a表的id上。

Select \* from a right join b on a.id = b.id;通常最好的建立索引方式是建立在a表的id上。

Select \* from b right join a on a.id = b.id;通常最好的建立索引方式是建立在b表的id上。

**索引优化：**

1. 尽可能减少join与剧中nestedloop的循环总次数：永远用小结果驱动大的结果集。

Select \* from a where id in (select id from b) 等价于：

For select id from b for select \* from a where a.id = b.id;当B表的数据集必须小于a表的数据集时，用in优于exists。

Select \* from a where exists(select 1 from b where b.id = a.id)等价于：

For select \* from a for select \* from b where b.id = a.id，当a表的数据集小于b表的数据集时，用exists由于in

提示：

1. exists只返回true或false，因此子查询中的select \* 也可以是select 1或select ‘x’，官方说法是实际执行会忽略select清单，因此没有区别。
2. exists子查询的实际执行过程可能经过了优化而不是我们理解上的逐条对比，如果担忧效率问题，可进行实际检验。
3. exists子查询往往也可以用条件表达式、其他子查询或者join替代，具体问题具体分析。

Filesort效率会低于index

Where高于having

2.如果索引了多列，要遵守最左前缀法则，指的是查询从索引的最左列开始并且不调过索引中的列。

如：create index index\_a on a(id,name); select \* from a where name = 'a';带头id断了，直接找name，会导致索引失效。

3.不在索引列上做任何操作（计算、函数、（自动or手动）类型转化），会导致索引失效引发全表扫描。

如：select \* from staffs where left(name,4) = ‘july’; 会导致索引失效全表扫描

4.存储引擎不能使用索引范围条件右边的列。

使用>这样范围的列，其右边的索引会失效

如select \* from staffs where name =’july’ and book>2 and id=1。Id的索引将会失效。

5．尽量使用覆盖索引（只访问索引的查询（索引列和查询列一致）），减少使用select \*

如：Select name from a where name = ‘a’;

6.mysql在使用不等于（！=或者<>）的时候无法使用索引会导致全表扫描。

7．Is null,is not null也无法使用索引

8.like以通配符开头（‘%abc…’）mysql索引失效，会编程全表扫描。（abc%）这种则不会。

9.字符串不加单引号会导致索引失效。

10.少用or，用它来连接时会导致索引失效。

**第二式：查询截取分析**

**开启慢查询日志**

Set global slow\_query\_log = 1开启了慢查询日志只对当前数据库生效。如果mysql重启则失效。如果要永久生效，必须修改my.cnf：slow\_query\_log=1 slow\_query\_log\_file=/var/lib/mysql/atguigu-slow.log

慢查询日志的时间：long\_query\_time。默认是10秒。大于10秒以上会被记录下来，不包括10秒，设置时间set global long\_query\_time=3.设置为3秒。需要重新连接或新开一个会话才能查看修改的值。

得到返回记录集最多的10个sql：

mysqldumpslow -s r -t 10 /var/lib/mysql/ubuntu-slow.log

得到访问次数最多的10gesql：

Mysqldumpslow -s c -t 10 /var/lib/mysql/ubuntu-slow.log

得到按照时间顺序的前10条里面含有左连接的查询语句:

Mysqldumpslow -s t -t 10 -g “legt join” /var/lib/mysql/ubuntu-slow.log

另外建议在使用这些命令时结合|和more使用，否则有可能出现爆屏的情况

Mysqldumpslow -s r -t 10 /var/lib/mysql/ubuntu-slow.log | more

查看mysqldumpslow的帮助信息：

s：是表示按照何种方式排序

c:访问次数

l：锁定时间

r：返回记录

t：查询时间

al：平均锁定时间

ar：平均返回记录数

at：平均查询时间

t：返回前面多少条的数据

g：后边搭配一个正则表达式，大小写不敏感

**show Profile(查看sql具体执行时间)**

profile是mysql提供可以用来分析当前会话中语句执行的资源消耗情况。可以用于sql调优的测量。

Show variables like ‘profiling’

开启show profile:set profiling=on;

show profiles:查看结果

show profile cpu,block io for query 序号（上一步前面问题sql数字号码）

,日志开发中需要注意的结论：

Converting HEAP to MyISAM 查询结果太大，内存溢出

Creating tmp table 创建临时表：拷贝数据，用完再删

Copying to tmp table on disk把内存中的临时表复制到磁盘中

Locked

遇到以上这些都要处理

**全局查询日志**（永远不要在生产环境中开启该功能）

**第三式：数据库的锁机制**

**1.MyISAM**在执行查询语句（select）前，会自动给涉及的所有表加读锁，在执行增删改操作前，会自动给涉及的表加写锁。

Mysql的表级锁有两种模式：表共享读锁，表独占写锁

1.针对myisam表的读操作（加读锁），不会阻塞其他进程对同一表的读请求，但会阻塞对同一表的写请求。只有当读锁释放后，才会执行其他进程的写操作。

2.针对myisam表的写操作（加写锁），会阻塞其他进程对同一表的读和写操作，只有当写锁释放后，其他进程才能进行读写操作。

总结：读锁会阻塞写，写锁读写都会阻塞。

使用样例：lock table testb read; unlock tables；

1. **InnoDB**存储引擎，开销大，加锁慢，会出现死锁，锁定粒度最小，发生锁冲突的概率最低，并发度也最高。与MyISAM的最大不同是：支持事务，采用行级锁(自动锁定索引)。

索引失效，行锁会变表锁；

间隙锁存在危害。

如何锁定一行（直到该会话commit）：select \* from test where a=8 for update;

查看行锁的争夺情况：show status like ‘innodb\_row\_lock%’;

事务：具有四个属性，通常称为事务的ACID属性

原子性（Atomicity）：事务是一个原子操作单元，对数据修改要么全部不执行，要么全部执行。

一致性（Consistent）：在事务开始和完成时，数据都必须保持一致状态。这意味着所有相关的数据规则都必须用于事务的修改，以保持数据的完整性；事务结束时，所有的内部数据结构（如B树索引或双向链表）也都必须是正确的。

隔离性（IsoIation）：数据库系统提供一定的隔离机制，保证事务在不受外部并发操作影响的“独立”环境执行。这意味着事务处理过程中的中间状态对外部是不可见的，反之亦然。

持久性（Durable）：事务完成之后，它对于数据的修改是永久性的，即使出现系统故障也能够保持。