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。

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。