**在一个千万级的数据库查寻中，如何提高查询效率？**

**1）数据库设计方面：**

a、对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。

b、应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描，如： select id from t where num is null 可以在num上设置默认值0，确保表中num列没有null值，然后这样查询： select id from t where num=0

c、并不是所有索引对查询都有效，SQL是根据表中数据来进行查询优化的，当索引列有大量数据重复时,查询可能不会去利用索引，如一表中有字段sex，male、female几乎各一半，那么即使在sex上建了索引也对查询效率起不了作用。

d、索引并不是越多越好，索引固然可以提高相应的 select 的效率，但同时也降低了 insert 及 update 的效率，因为 insert 或 update 时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。一个表的索引数最好不要超过6个，若太多则应考虑一些不常使用到的列上建的索引是否有必要。

e、应尽可能的避免更新索引数据列，因为索引数据列的顺序就是表记录的物理存储顺序，一旦该列值改变将导致整个表记录的顺序的调整，会耗费相当大的资源。若应用系统需要频繁更新索引数据列，那么需要考虑是否应将该索引建为索引。

f、尽量使用数字型字段，若只含数值信息的字段尽量不要设计为字符型，这会降低查询和连接的性能，并会增加存储开销。这是因为引擎在处理查询和连接时会逐个比较字符串中每一个字符，而对于数字型而言只需要比较一次就够了。

g、尽可能的使用 varchar/nvarchar 代替 char/nchar ，因为首先变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率显然要高些。

h、尽量使用表变量来代替临时表。如果表变量包含大量数据，请注意索引非常有限（只有主键索引）。

i、避免频繁创建和删除临时表，以减少系统表资源的消耗。

j、临时表并不是不可使用，适当地使用它们可以使某些例程更有效，例如，当需要重复引用大型表或常用表中的某个数据集时。但是，对于一次性事件，最好使用导出表。

k、在新建临时表时，如果一次性插入数据量很大，那么可以使用 select into 代替 create table，避免造成大量 log ，以提高速度；如果数据量不大，为了缓和系统表的资源，应先create table，然后insert。

l、如果使用到了临时表，在存储过程的最后务必将所有的临时表显式删除，先 truncate table ，然后 drop table ，这样可以避免系统表的较长时间锁定。

**2)SQL语句方面：**

a、应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。

b、应尽量避免在 where 子句中使用 or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描，如： select id from t where num=10 or num=20 可以这样查询： select id from t where num=10 union all select id from t where num=20

c、n 和 not in 也要慎用，否则会导致全表扫描，如： select id from t where num in(1,2,3) 对于连续的数值，能用 between 就不要用 in 了： select id from t where num between 1 and 3

d、下面的查询也将导致全表扫描： select id from t where name like ‘%abc%’

e、如果在 where 子句中使用参数，也会导致全表扫描。因为SQL只有在运行时

才会解析局部变量，但优化程序不能将访问计划的选择推迟到运行时；它必须在编译时进行选择。然而，如果在编译时建立访问计划，变量的值还是未知的，因而无法作为索引选择的输入项。如下面语句将进行全表扫描： select id from t where num=@num 可以改为强制查询使用索引： select id from t with(index(索引名)) where num=@num

f、应尽量避免在 where 子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描。如： select id from t where num/2=100 应改为: select id from t where num=100\*2

g、应尽量避免在where子句中对字段进行函数操作，这将导致引擎放弃使用索引而进行全表扫描。如： select id from t where substring(name,1,3)=’abc’–name以abc开头的id select id from t where datediff(day,createdate,’2005-11-30′)=0–‘2005-11-30’生成的id 应改为: select id from t where name like ‘abc%’ select id from t where createdate>=’2005-11-30′ and createdate<’2005-12-1′

h、不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。

i、不要写一些没有意义的查询，如需要生成一个空表结构： select col1,col2 into #t from t where 1=0 这类代码不会返回任何结果集，但是会消耗系统资源的，应改成这样： create table #t(…)

j、很多时候用 exists 代替 in 是一个好的选择： select num from a where num in(select num from b) 用下面的语句替换： select num from a where

exists(select 1 from b where num=a.num)

k、任何地方都不要使用 select \* from t ，用具体的字段列表代替“\*”，不要返回用不到的任何字段。

l、尽量避免使用游标，因为游标的效率较差，如果游标操作的数据超过1万行，那么就应该考虑改写。

m、尽量避免向客户端返回大数据量，若数据量过大，应该考虑相应需求是否合理。

n、尽量避免大事务操作，提高系统并发能力。

**3)Java方面：重点内容**

a、尽可能的少造对象。

b、合理摆正系统设计的位置。大量数据操作，和少量数据操作一定是分开的。大量的数据操作，肯定不是ORM框架搞定的。，

c、使用jDBC链接数据库操作数据

d、控制好内存，让数据流起来，而不是全部读到内存再处理，而是边读取边处理；

e、合理利用内存，有的数据要缓存

**如何优化数据库，如何提高数据库的性能?**

**解答：**

**1） 硬件调整性能** 最有可能影响性能的是磁盘和网络吞吐量,解决办法扩大虚拟内存，并保证有足够可以扩充的空间；把数据库服务器上的不必要服务关闭掉；把数据库服务器和主域服务器分开；把SQL数据库服务器的吞吐量调为最大；在具有一个以上处理器的机器上运行SQL。

**2）调整数据库**

若对该表的查询频率比较高，则建立索引；建立索引时，想尽对该表的所有查询搜索操作， 按照where选择条件建立索引，尽量为整型键建立为有且只有一个簇集索引，数据在物理上按顺序在数据页上，缩短查找范围，为在查询经常使用的全部列建立非簇集索引，能最大地覆盖查询；但是索引不可太多，执行UPDATE DELETE INSERT语句需要用于维护这些索引的开销量急剧增加；避免在索引中有太多的索引键；避免使用大型数据类型的列为索引；保证每个索引键值有少数行。

**3）使用存储过程**

应用程序的实现过程中，能够采用存储过程实现的对数据库的操作尽量通过存储过程来实现，因为存储过程是存放在数据库服务器上的一次性被设计、编码、测试，并被再次使用，需要执行该任务的应用可以简单地执行存储过程，并且只返回结果集或者数值，这样不仅可以使程序模块化，同时提高响应速度，减少网络流量，并且通过输入参数接受输入，使得在应用中完成逻辑的一致性实现。

**4）应用程序结构和算法**

建立查询条件索引仅仅是提高速度的前提条件，响应速度的提高还依赖于对索引的使用。因为人们在使用SQL时往往会陷入一个误区，即太关注于所得的结果是否正确，特别是对数据量不是特别大的数据库操作时，是否建立索引和使用索引的好坏对程序的响应速度并不大，因此程序员在书写程序时就忽略了不同的实现方法之间可能存在的性能差异，这种性能差异在数据量特别大时或者大型的或是复杂的数据库环境中（如联机事务处理OLTP或决策支持系统DSS）中表现得尤为明显。在工作实践中发现，不良的SQL往往来自于不恰当的索引设计、不充份的连接条件和不可优化的where子句。在对它们进行适当的优化后，其运行速度有了明显地提高！

**数据库索引的作用：**

索引的意义 ·索引在数据库中的作用相当于目录在书籍中的作用类似，都用来提高查找信息的速度。 ·索引是一个表中所包含值的列表，其中注明了表中包含各个值的行所在的存储位置，使用索引查找数据时，先从索引对象中获得相关列的存储位置，然后再直接去其存储位置查找所需信息，这样就无需对这个表进行扫描，从而可以快速的找到所需数据。

**数据库游标的作用（cursor）：**

游标实际上是一种能从包括多条数据记录的结果集中每次提取一条记录的机制。游标充当指针的作用。尽管游标能遍历结果中的所有行，但他一次只指向一行。  
  
概括来讲，SQL的游标是一种临时的数据库对象，即可以用来存放在数据库表中的数据行副本，也可以指向存储在数据库中的数据行的指针。游标提供了在逐行的基础上操作表中数据的方法。  
  
游标的一个常见用途就是保存查询结果，以便以后使用。游标的结果集是由SELECT语句产生，如果处理过程需要重复使用一个记录集，那么创建一次游标而重复使用若干次，比重复查询数据库要快的多。  
  
大部分程序数据设计语言都能使用游标来检索SQL数据库中的数据，在程序中嵌入游标和在程序中嵌入SQL语句相同