****1、在sql查询中为了提高查询效率，我们常常会采取一些措施对查询语句进行sql优化，下面总结的一些方法，有需要的可以参考参考。****

**1.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。      
      
2.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      
      
3.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。      
      
4.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      
      
5.in 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      
      
6.下面的查询也将导致全表扫描：      
select id from t where name like '%abc%'      
      
7.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      
      
8.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      
应改为:      
select id from t where name like 'abc%'      
      
9.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。      
      
10.在使用索引字段作为条件时，如果该索引是复合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，      
否则该索引将不会被使用，并且应尽可能的让字段顺序与索引顺序相一致。      
      
11.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(...)      
      
12.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)      
      
13.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，SQL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。      
      
14.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。      
      
15.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。      
      
16.尽可能的使用 varchar 代替 char ，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。      
      
17.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。      
      
18.避免频繁创建和删除临时表，以减少系统表资源的消耗。  
  
19.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。      
      
20.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。**

****2、基于emp表，列出在部门“SALES”<销售部>工作的员工的姓名（假定不知道销售部门的部门编号）****

**selecte.enamefromemp ewhere  
   e.deptno = (select deptno from dept where dname = 'SALES');**

****3、基于emp表，列出最低薪水大于1500的各种工作及从事此工作的全部雇员人数****

**select job,min(sal),count(\*) from emp group by job having min(sal) > 1500;**

****4、基于emp表，列出薪水比“SMITH”多的所有员工信息****

**select \* from emp where sal > (select sal from emp where ename = 'SMITH');**

****5、基于emp表，取得薪水最高的第六到第十名员工****

**select ename,sal from emp order by sal desc limit 5,5;**

****6、基于emp表，取得薪水最高的前五名员工****

**select \* from emp order by sal desc limit 5;**

****7、基于emp表和dept表,求平均薪水的等级最低的部门的部门名称****

**select   
t.deptno,t.dname,t.avgsal,s.grade  
from  
(select   
e.deptno,d.dname,avg(e.sal) as avgsal   
from   
emp e  
join  
dept d  
on  
e.deptno = d.deptno  
group by   
e.deptno) t  
join  
salgrade s  
on  
t.avgsal between s.losal and s.hisal  
where  
s.grade = (select   
max(s.grade)  
from  
(select   
e.deptno,avg(e.sal) as avgsal   
from   
emp e  
group by   
e.deptno) t  
join  
salgrade s  
on  
t.avgsal between s.losal and s.hisal);**

****8、基于emp表和salgrade表，取得部门中（所有人的）平均薪水的等级****

**select   
e.deptno,avg(e.sal) as avgsal   
from   
emp e   
group by   
e.deptno;  
  
select   
t.\*,s.grade  
from  
salgrade s  
join  
(select deptno,avg(sal) as avgsal from emp e group by deptno) t  
on  
t.avgsal between s.losal and s.hisal;**

****9、基于emp表和dept表，求部门名称中带“S”字符的部门员工的工资合计、部门人数****

**select  
d.dname,  
ifnull(sum(e.sal),0) as deptSumsal,  
count(e.ename) as totalEmp  
from  
emp e  
right join  
dept d  
on  
e.deptno = d.deptno  
where  
d.dname like '%S%'  
group by  
d.dname;**

****10、基于emp表，求出员工领导的薪水超过3000的员工名称和领导名称****

**select  
a.ename as empname,  
a.sal as empsal,  
b.ename as leadername,  
b.sal as leadersal  
from  
emp a  
join  
emp b  
on  
a.mgr = b.empno  
where  
b.sal > 3000;**

****11、基于emp表，列出所有员工的年工资，按年薪从低到高排序****

**select   
ename,(sal+ifnull(comm,0))\*12 as yearsal  
from  
emp  
order by  
yearsal asc;**

****12、基于emp表，列出各个部门MANAGER的最低薪金****

**select  
deptno,min(sal) as minsal  
from  
emp  
where  
job = 'MANAGER'  
group by  
deptno;**

****13、基于emp表，列出各种工作的最低工资及从事此工作的雇员姓名****

**select  
e.ename,e.job  
from  
emp e  
join  
(select job,min(sal) as minsal from emp group by job) t  
on  
e.job = t.job   
where  
e.sal = t.minsal;**

****14、基于emp表和dept表，列出在每个部门工作的员工数量、平均工资和平均服务期限****

**select  
d.deptno,  
count(e.ename) as totalEmp,  
ifnull(avg(e.sal),0) as avgsal,  
avg(ifnull((to\_days(now()) - to\_days(hiredate))/365,0)) as serverTime  
from  
emp e  
right join  
dept d  
on  
e.deptno = d.deptno  
group by  
d.deptno;**

****15、基于emp表和dept表，列出薪金高于在部门30工作的所有员工的薪金的员工姓名和薪金、部门名称****

**select   
e.ename,e.sal,d.dname   
from   
emp e  
join  
dept d  
on  
e.deptno = d.deptno  
where   
e.sal > (select max(sal) from emp where deptno = 30) and e.deptno <> 30;**

****16、基于emp表，列出薪金等于部门30中员工的薪金的其它员工的姓名和薪金****

**select   
ename,sal,deptno   
from   
emp   
where   
sal in(select sal from emp where deptno = 30) and deptno <> 30;**

****17、基于emp表和dept表，列出与“SCOTT”从事相同工作的所有员工及部门名称****

**select   
e.ename,d.dname  
from  
emp e  
join  
dept d  
on  
e.deptno = d.deptno  
where  
e.job = (select job from emp where ename = 'SCOTT');**

****18、基于emp表和dept表，列出所有“CLERK”（办事员）的姓名及其部门名称，部门人数****

**select   
e.ename,d.dname,t.totalEmp  
from  
dept d  
join  
emp e  
on  
e.deptno = d.deptno  
join  
(select deptno,count(\*) as totalEmp from emp group by deptno) t  
on  
d.deptno = t.deptno  
where  
job = 'CLERK';**

****19、基于emp表和dept表，列出至少有5个员工的所有部门****

**select   
e.deptno,d.dname,d.loc,count(e.ename)  
from  
emp e  
join  
dept d  
on  
e.deptno = d.deptno  
group by  
e.deptno,d.dname,d.loc  
having  
count(e.empno) >= 5;**

****20、基于emp表和dept表，列出部门名称和这些部门的员工信息，同时列出那些没有员工的部门****

**select   
d.deptno,d.dname,e.ename  
from  
emp e  
right join  
dept d  
on  
e.deptno = d.deptno;**

****21、基于emp表，列出受雇日期早于其直接上级的所有员工编号、姓名、部门名称****

**select   
a.empno,a.ename,a.hiredate as empHiredate,  
b.empno as leaderNo,b.ename as leaderName,b.hiredate as leaderHiredate  
from   
emp a  
join  
emp b  
on  
a.mgr = b.empno  
where   
a.hiredate < b.hiredate;**

****22、基于emp表和salgrade表，取得每个薪水等级有多少员工****

**select   
s.grade,count(\*)  
from  
emp e  
join   
salgrade s  
on  
e.sal between s.losal and s.hisal  
group by  
s.grade;**

****23、基于emp表，取得最后入职的5名员工****

**select ename,hiredate from emp order by hiredate desc limit 5;**

****24、基于emp表，取得比普通员工（员工代码没有在mgr上出现的）的最高薪水还要高的经理人姓名****

**select \* from emp where   
sal > (select max(sal) as maxsal from emp where empno not in(select distinct mgr from emp where mgr is not null));**

****25、基于emp表，不准用组函数（MAX），取得最高薪水（给出两种解决方案）****

**第一种方案：按照员工薪水降序排列，取第一个  
  
select ename,sal from emp order by sal desc limit 1;  
  
第二种方案：自连接  
  
select   
sal   
from   
emp   
where   
sal not in(select distinct a.sal from emp a join emp b on a.sal < b.sal);**

****26、基于emp表，哪些人的薪水在部门平均薪水之上****

**select   
e.ename,t.\*  
from  
emp e  
join  
(select deptno,avg(sal) as avgsal from emp group by deptno) t  
on  
e.deptno = t.deptno   
where  
e.sal > t.avgsal;**

****27、基于emp表，取得每个部门最高薪水的人员名称****

**select  
e.ename,t.\*  
from  
emp e  
join  
(select deptno,max(sal) as maxsal from emp group by deptno) t  
on  
t.deptno = e.deptno   
where  
t.maxsal = e.sal;**

****28、sql语句应该考虑哪些安全性？****

**（1）防止sql注入，对特殊字符进行转义，过滤或者使用预编译的sql语句绑定变量。**

**（2）最小权限原则，特别是不要用root账户，为不同的类型的动作或者组建使用不同的账户。**

**（3）当sql运行出错时，不要把数据库返回的错误信息全部显示给用户，以防止泄漏服务器和数据库相关信息。**

**OK**

****29、查看数据库版本及当前登录用户是什么****

**查看数据库版本命令：select version();**

**查看当前登录用户：select user();**

****30、共享读锁的特征****

**当前用户为表添加共享读锁后，其它用户可以读取表，但不能写表  
lock table 表名 read local时，其它用户可以读取表，也可以写表，但是写的内容不会被当前用户看到**

****31、独享写锁的特征****

**当前用户为表添加写锁后，其它用户既不能对表进行读操作也不能进行写操作**

****32、MyISAM表可以使用那种锁****

**MyIsam中只能使用表锁。表锁又分为【写锁】和【】读锁】**

****33、字符编码，对于数据库效率的影响****

**1.纯拉丁字符能表示的内容，没必要选择 latin1 之外的其他字符编码，因为这会节省大量的存储空间  
2.如果我们可以确定不需要存放多种语言，就没必要非得使用UTF8或者其他UNICODE字符类型，这回造成大量的存储空间浪费  
3. MySQL的数据类型可以精确到字段，所以当我们需要大型数据库中存放多字节数据的时候，可以通过对不同表不同字段使用不同的数据类型来较大程度减小数据存储量，进而降低 IO 操作次数并提高缓存命中率**

****34、显示的为表添加锁后，是否可以访问其它表****

**不能，因为mysql不支持锁升级**

****35、为什么说myisam不适合作为既有大量更新操作又有大量读取操作的表****

**因为mysql中写锁的优先级远远高于读锁，大量的写锁申请会大大的延迟读锁的实现，导致读取 数据效率非常的低下。**

****36、MyIsam运行时，是否会出现死锁****

**永远不会，因为myisam的执行串行的，不支持并发**

****37、两个用户分别同时向表申请添加【独占写锁】和【共享读锁】，此时mysql会 为当前表添加何种锁****

**在mysql中，【写锁】优先级高于【读锁】，因此当前表添加的是【写锁】**

****38、如何获取表内所有 Field 对象的名称和类型？****

**运行命令 describe table\_name;**

****39、SQL语句的分类****

**数据库查询语言（DQL）、数据库操作语言（DQL）、数据库定义语言（DDL）、事务控制语言（TCL）、数据控制语言（DML）**

**OK**

****40、在DOS窗口中直接通过命令行的形式进行控制，启动/关闭MySQL服务****

**1、启动MySQL服务：net start mysql**

**2、关闭MySQL服务：net stop mysql**

****41、如何得到TabA表的建表语句****

**show create table TabA;**

****42、查看某一用户的权限****

**show grants for UserName;**

****43、查看当前MYSQL支持的存储引擎****

**show engines;**

****44、事务中的提交和回滚是什么意思？****

**1、事务中的提交（COMMIT）是提交事务的所有操作。具体说就是将事务中所有对数据库的更新写回到磁盘上的物理数据库中去，事务正常结束。**

**2、事务中的回滚（ROLLBACK）是数据库滚回到事务开始时的状态。具体地说就是，在事务运行的过程中发生了某种故障，事务不能继续执行，系统将事务中对数据库的所有已完成的更新操作全部撤消，使数据库回滚到事务开始时的状态。**

**OK**

****45、SQL语言的数据操纵语句包括SELECT，INSERT，UPDATE和DELETE，最重要的，也是使用最频繁的语句是\_\_\_\_。****

****A、SELECT****

****B、INSERT****

****C、UPDATE****

****D、DELETE****

**A**

****46、SQL语言具有\_\_\_\_的功能****

****A、关系规范化、数据操纵、数据控制****

****B、数据定义、数据操纵、数据控制****

****C、数据定义、关系规范化、数据控制****

****D、数据定义、关系规范化、数据操纵****

**B**

****47、查看所有数据库语句****

**show databases;**

****48、mysql取得当前时间的函数是?，格式化日期的函数是？****

**取得当前时间用now()函数**

**在数据库中格式化时间用：DATE\_FORMAT(date, format)**

**在数据库中代表年月日时分秒分别是：%Y-%m-%d %H-%i-%S/s**

****49、where 与 having 区别：****

**1、where 和 having 都是为了完成数据的过滤，它们后面都是添加条件；**

**2、where 是在 group by之前完成过滤；**

**3、having 是在 group by 之后完成过滤；**

****50、MYSQL数据库分页****

**页码：pageNo 每页显示条数：pageSize**

**sql = "select \* from 表名 limit " + (pageNo-1)\*pageSize + "," + pageSize;**

****51、数据库中的连接****

**连接分为内连接和外连接两种类型。**

**1、内连接:等值连接、非等值连接和自连接**

**a.等值连接：等值连接是指表之间通过“等于关系”连接起来**

**b.非等值连接：表之间的连接关系不是“等于”**

**c.自连接：在数据查询中有时需要将同一个表进行连接，这种连接称之为自连接，进行自连接如同两个分开的表 一样，可以把一个表的某行与同一表中的另一行连接起来。**

**2、外连接：仅当至少有一个同属于两表的行符合连接条件时，内连接才返回行，所以内连接消除与另一个表中的任何行不匹配的行。**

**外连接分为：左外连接、右外连接和全外连接**

**a.左外连接：左外连接结果包含第一个命名表（"左"表，出现在JOIN子句的左边）中的所有行，不包括右表中不匹配的行**

**b.右外连接：右外连接结果包含第一个命名表（"右"表，出现在JOIN子句的右边）中的所有行，不包括左表中不匹配的行**

**c.全外连接：若要通过在连接结果中包括不匹配的行即保留不匹配的信息，可以使用全连接**

****52、什么是笛卡尔积现象****

**多张表连接查询时，没有任何条件限制的情况下，最终查询条数是多张表记录的乘积**

****53、myisam引擎的特征****

**MyIsam:  
1.每张MyISAM 表被存放在三个文件 ：frm 文件存放表格定义。 数据文件是MYD (MYData) 。 索引文件是MYI (MYIndex) 引伸.  
2.MyISAM 不会出现死锁  
3.MyISAM 不支持事务  
4.MYISAM适合小数据，小并发。适合用于频繁查询的应用   
5.对于AUTO\_INCREMENT类型的字段  
6.在MyISAM表中，可以和其他字段一起建立联合索引  
7.执行select count(\*) from table时,MyISAM只要简单的读出保存好的行数即可.不需要扫描一遍整个表来计算有多少行  
8.MyISAM是文件存储的，可以进行直接在不同操作系统间拷贝使用**

****54、mysql行级锁定的优点?****

**1. 当在许多线程中访问不同的行时只存在少量锁定冲突  
2.回滚时只有少量的更改。  
3.可以长时间锁定单一的行**

****55、内联接,外联接区别？****

**内连接是保证两个表中所有的行都要满足连接条件，而外连接则不然。**

**在外连接中，某些不满条件的列也会显示出来，也就是说，只限制其中一个表的行，而不限制另一个表的行。**

**外连接查询条数 >= 内连接查询条数**

****56、MySQL 检查缓存命中的规则****

**（1）在检查缓存的时候，MySQL 不会对语句进行解析、正则化或者参数化，它精确地使用客户端传来的查询语句和其他数据。只要字符大小写、空格或者注释有一点点不同，查询缓存就认为这是一个不同的查询  
（2）查询缓存不会存储有不确定结果的查询。因此，任何一个包含不确定函数（比如NOW()或CURRENT\_DATE()）的查询不会被缓存。同样地，CURRENT\_USER()或CONNECTION\_ID()这些由不同用户执行，将会产生不同的结果的查询也不会被缓存。事实上，查询缓存不会缓存引用了用户自定义函数、存储函数、用户自定义变量、临时表、mysql 数据库中的表或者任何一个有列级权限的表的查询  
（3）查询必须是完全相同的(逐字节相同)才能够被认为是相同的。另外，同样的查询字符串由于其它原因可能认为是不同的。使用不同的数据库、不同的协议版本或者不同 默认字符集的查询被认为是不同的查询并且分别进行缓存。**

****57、MYSQL支持事务吗？****

**在缺省模式下，MYSQL是autocommit模式的，所有的数据库更新操作都会即时提交，所以在缺省情况下，mysql是不支持事务的。  
但是如果你的MYSQL表类型是使用InnoDB Tables 或 BDB tables的话，你的MYSQL就可以使用事务处理,使用SET AUTOCOMMIT=0就可以使MYSQL允许在非autocommit模式，在非autocommit模式下，你必须使用COMMIT来提交你的更改，或者用ROLLBACK来回滚你的更改。**

****58、查看当前数据库里有哪些用户。****

**mysql> SELECT DISTINCT CONCAT('User: ''',user,'''@''',host,''';') AS query FROM mysql.user;**

****59、如何知道表内行数？****

**select count(\*) from 表名;**

****60、如何删除表？****

**drop table 表名;**

****61、如何列出某个数据库内所有表？****

**show tables;**

****62、如何开启或停止 MySQL 服务？****

**运行命令 service mysqld start 开启服务；  
运行命令 service mysqld stop 停止服务。**

****63、请简述项目中优化sql语句执行效率的方法，从哪些方面。sql语句性能如何分析？****

**（1）尽量选择较小的列   
（2）将where中用的比较频繁的字段建立索引   
（3）select子句中避免使用‘\*’   
（4）避免在索引列上使用计算，not，in和<>等操作   
（5）当只需要一行数据的时候使用limit 1   
（6）保证表单数据不超过200w，适时分割表   
（7）针对查询较慢的语句，可以使用explain来分析该语句具体的执行情况**

****64、mysql默认的慢查询时间是多少?****

**10s**

****65、简单描述MySQL中，索引，主键，唯一索引，联合索引的区别，对数据库的性能有什么影响。****

**1）索引是一种特殊的文件（InnoDB数据表上的索引是表空间的一个组成部分），它们包含着对数据表里所有记录的引用指针。   
（2）普通索引（由关键字KEY或INDEX定义的索引）的唯一任务是加快对数据的访问速度。  
（3）普通索引允许被索引的数据列包含重复的值，如果能确定某个数据列只包含彼此各不相同的值，在为这个数据索引创建索引的时候就应该用关键字UNIQE把它定义为一个唯一所以，唯一索引可以保证数据记录的唯一性。  
（4）主键，一种特殊的唯一索引，在一张表中只能定义一个主键索引，逐渐用于唯一标识一条记录，是用关键字PRIMARY KEY来创建。   
（5）索引可以覆盖多个数据列，如像INDEX索引，这就是联合索引。   
（6）索引可以极大的提高数据的查询速度，但是会降低插入删除更新表的速度，因为在执行这些写操作时，还要操作索引文件。**

****66、sql语句应该考虑哪些安全性？****

**（1）防止sql注入，对特殊字符进行转义，过滤或者使用预编译的sql语句绑定变量。   
（2）最小权限原则，特别是不要用root账户，为不同的类型的动作或者组建使用不同的账户。   
（3）当sql运行出错时，不要把数据库返回的错误信息全部显示给用户，以防止泄漏服务器和数据库相关信息**

**OK**

****67、MySql常用存储引擎有哪些？如何修改mysql存储引擎？****

**常用存储引擎有：**

**1、MyISAM 具有检查和修复表格的大多数工具。表格可以被压缩，而且支持全文搜索，不是事务安全的，而且不支持外键。**

**2、MEMORY 也是以前的(HEAP) 该类型表存储在内存中，表的索引是哈希分布的。**

**3、Innodb 这种表是事务安全的。提供了commit（提交） rollback（实务回滚）支持外键，比MyISAM慢**

**修改mysql存储引擎：**

**alter table tablename type = innodb;**

****68、MySQL相比于其他数据库有哪些特点****

**1、可以处理拥有上千万条记录的大型数据**

**2、支持常见的SQL语句规范**

**3、可移植行高，安装简单小巧**

**4、良好的运行效率，有丰富信息的网络支持**

**5、调试、管理，优化简单（相对其他大型数据库）**

****69、什么是主键?什么是外键?****

**主键是表格里的(一个或多个)字段，只用来定义表格里的行;主键里的值总是唯一的。外键是一个  
用来建立两个表格之间关系的约束。这种关系一般都涉及一个表格里的主键字段与另外一个  
表(可能是同一表)里的字段。那么这些相连的字段就是外键。**

****70、SQL语句优化****

**1、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。**

**2、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**

**3、用Where子句替换HAVING 子句 因为HAVING 只会在检索出所有记录之后才对结果集进行过滤**

****71、一般来说，应该在这些列上创建索引****

**（1）在经常需要搜索的列上，可以加快搜索的速度；**

**（2）在作为主键的列上，强制该列的唯一性和组织表中数据的排列结构；**

**（3）在经常用在连接的列上，这些列主要是一些外键，可以加快连接的速度；**

**（4）在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的；**

**（5）在经常需要排序的列上创建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时间；**

**（6）在经常使用在WHERE子句中的列上面创建索引，加快条件的判断速度。**

****72、什么是SQL注入？****

**所谓SQL注入，就是通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。具体来说，它是利用现有应用程序，将（恶意）的SQL命令注入到后台数据库引擎执行的能力，它可以通过在Web表单中输入（恶意）SQL语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行SQL语句。**

**OK**

****73、简单介绍数据库设计三范式****

**1、数据库设计第一范式：要求有主键，数据库中不能出现重复记录，每一个字段是原子性不能再分;**

**2、数据库设计第二范式：第二范式是建立在第一范式基础之上，要求数据库中所有非主键字段完全依赖主键，不能产生部分依赖；（严格意义上说：尽量不要使用联合主键）**

**3、数据库设计第三范式：建立在第二范式基础之上，要求非主键字段不能产生传递依赖于主键字段；**

****74、SQL语句中修改表结构的命令是****

****A、MODIFY TABLE****

****B、MODIFY STRUCTURE****

****C、ALTER TABLE****

****D、ALTER STRUCTURE****

**C**

****75、试述视图的优点****

**(1) 视图能够简化用户的操作**

**(2) 视图使用户能以多种角度看待同一数据；**

**(3) 视图为数据库提供了一定程度的逻辑独立性；**

**(4) 视图能够对机密数据提供安全保护。**

****76、在一个查询中，使用哪一个关键字能够除去重复列值****

**使用distinct关键字**

****77、请简洁地描述下MySQL中InnoDB支持的事务隔离级别名称以及区别？****

**SQL标准定义的四个隔离级别为：**

**read uncommited：读取未提交内容**

**read committed：读取提交内容**

**repeatable read：可重复读**

**serializable：串行化**

**详细解释如下：**

**Read Uncommitted（读取未提交内容）**

**在该隔离级别，所有事务都可以看到其他未提交事务的执行结果。**

**本隔离级别很少用于实际应用，因为它的性能也不比其他级别好**

**多少。读取未提交的数据，也被称之为脏读（Dirty Read）。**

**Read Committed（读取提交内容）**

**这是大多数数据库系统的默认隔离级别（但不是MySQL默认的）。**

**它满足了隔离的简单定义：一个事务只能看见已经提交事务所做的**

**改变。这种隔离级别也支持 所谓的不可重复读（Nonrepeatable**

**Read），因为同一事务的其他实例在该实例处理其间可能会有新的**

**commit，所以同一select可能返回不同结果。**

**Repeatable Read（可重读）**

**这是MySQL的默认事务隔离级别，它确保同一事务的多个实例在并**

**发读取数据时，会看到同样的数据行。不过理论上，这会导致另一个**

**棘手的问题：幻读 （Phantom Read）。简单的说，幻读指当用户**

**读取某一范围的数据行时，另一个事务又在该范围内插入了新行，当**

**用户再读取该范围的数据行时，会发现有新的“幻影” 行。**

**Serializable（可串行化）**

**这是最高的隔离级别，它通过强制事务排序，使之不可能相互冲突，**

**从而解决幻读问题。简言之，它是在每个读的数据行上加上共享锁。**

**在这个级别，可能导致大量的超时现象和锁竞争。**

**对于不同的事务，采用不同的隔离级别分别有不同的结果。不同的隔**

**离级别有不同的现象。主要有下面3种现在：**

**1、脏读（dirty read）：一个事务可以读取另一个尚未提交事务的修改数据。**

**2、非重复读（nonrepeatable read）：在同一个事务中，同一个查询在T1**

**时间读取某一行，在T2时间重新读取这一行时候，这一行的数据已经发生修改，**

**可能被更新了（update），也可能被删除了（delete）。**

**3、幻像读（phantom read）：在同一事务中，同一查询多次进行时候，由于**

**其他插入操作（insert）的事务提交，导致每次返回不同的结果集。**

**不同的隔离级别有不同的现象，并有不同的锁定/并发机制，隔离级别越高，数**

**据库的并发性就越差。**

****78、MySQL数据库事务的四个特性****

**原子性、一致性、隔离性、持久性**

****79、varchar和char的区别****

**Char是一种固定长度的类型，varchar是一种可变长度的类型**

****80、MYSQL支持事务吗？****

**在缺省模式下，MYSQL是autocommit模式的，所有的数据库更新操作**

**都会即时提交，所以在缺省情况下，mysql是不支持事务的。**

**但是如果你的MYSQL表类型是使用InnoDB Tables的话，你的MYSQL**

**就可以使用事务处理,使用SET AUTOCOMMIT=0就可以使MYSQL允许**

**在非autocommit模式，在非autocommit模式下，你必须使用COMMIT**

**来提交你的更改，或者用ROLLBACK来回滚你的更改。**

****81、请简述项目中优化sql语句执行效率的方法，从哪些方面。sql语句性能如何分析****

**（1）尽量选择较小的列  
（2）将where中用的比较频繁的字段建立索引  
（3）select子句中避免使用‘\*’  
（4）避免在索引列上使用计算，not，in和<>等操作  
（5）当只需要一行数据的时候使用limit 1  
（6）保证表单数据不超过200w，适时分割表  
（7）针对查询较慢的语句，可以使用explain来分析该语句具体的执行情况**

****82、简单描述mysql中，索引，主键，唯一索引，联合索引的区别，对数据库的性能有什么影响****

**（1）索引是一种特殊的文件（InnoDB数据表上的索引是表空间的一个组成部分），它们包含着对数据表里所有记录的引用指针。  
（2）普通索引（由关键字KEY或INDEX定义的索引）的唯一任务是加快对数据的访问速度。  
（3）普通索引允许被索引的数据列包含重复的值，如果能确定某个数据列只包含彼此各不相同的值，在为这个数据索引创建索引的时候就应该用关键字UNIQE把它定义为一个唯一所以，唯一索引可以保证数据记录的唯一性。  
（4）主键，一种特殊的唯一索引，在一张表中只能定义一个主键索引，逐渐用于唯一标识一条记录，是用关键字PRIMARY KEY来创建。  
（5）索引可以覆盖多个数据列，如像INDEX索引，这就是联合索引。  
（6）索引可以极大的提高数据的查询速度，但是会降低插入删除更新表的速度，因为在执行这些写操作时，还要操作索引文件**

****83、MySql中char与varchar区别****

**char:使用指定长度的固定长度表示的字符串；比如char（8），则数据库会使用固定的8个字节 来存储.，不足8为的字符串在其后补空字符；  
varchar:每个值只占用刚好够用的字节，再加上一个用来记录其长度的字节，当长度小于255时，长度记录位占一个字节，大于时，占用两个字节  
char 不需计算，占空间  
varchar 需计算，省空间**