一、Mysql 面试题（重点）

1.1 数据库中常用术语

ddl:数据定义语言 Create Drop Alter

dml:数据操纵语言 insert update delete select

dcl:数据控制语言 grant revoke

DQL：数据库查询语言

tcl:事务控制语言 commit rollback

1.2 数据库的聚合函数有哪些，连接查询有几种？

sum 、avg 、max、min、count 等等。

连接查询有：内连接和外连接，外连接又分为左连接和右链接。左连接是指满足查询条件的左表数据全部展示，右表不展示。内连接是指两张表的条件都必须满足。

1.3 数据库中的索引是指什么？

索引是对数据库表中一或多个列的值进行排序的结构，是帮助 MySQL 高效获取数据的

数据结构。（是一种数据结构）

MySQL 数据库几个基本的索引类型：普通索引、唯一索引、主键索引、联合索引

索引加快数据库的检索速度、索引降低了插入、删除、修改等维护任务的速度、唯一索引可以确保每一行数据的唯一性，通过使用索引，可以在查询的过程中使用优化隐藏器，

提高系统的性能，索引需要占物理和数据空间 。

1.4 数据库创建索引时候会考虑哪些因素？

我们一般对于查询概率比较高、字段值不固定的，经常作为 where 条件的字段设置索引。

1.5 是否创建过联合索引，联合索引多个字段之间顺序如何选择？

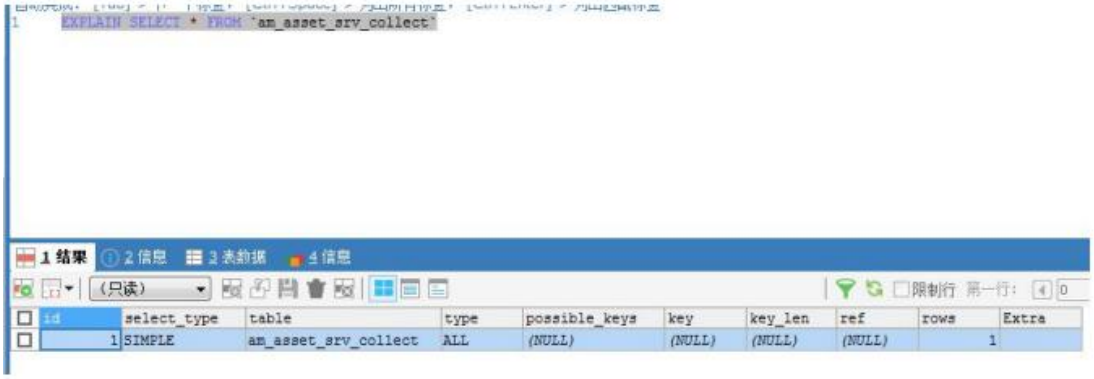
有创建过，当查询是多条件的时候也可以使用联合索引。在创建时候会根据索引使用活跃度来进行排序，活跃度最高的创建时候放在第一个，依次递减。where 子句中使用最 频繁的一列放在最左边，因为 MySQL 索引查询会遵循最左前缀匹配的原则，即最左优先，在检索数据时从联合索引的最左边开始匹配。所以当我们创建一个联合索引的时候， 如(key1,key2,key3)，相当于创建了（key1）、(key1,key2)和(key1,key2,key3)三个索引，这就是最左匹配原则。

1.6 什么时候会添加索引，如何查看索引是否生效

一般在开发的时候，如果某个查询操作特别慢的时候就会考虑添加索引。还有就是线上数据库会统计慢 sql，这个时候也会根据情况进行索引的添加。

通过 explain 查看执行计划可以检查 sql 是否通过索引进行查询。

1.7 如何通过 explain 查看 sql



1.8 数据库的搜索引擎有几种

MyISAM:

不支持事务，但是每次查询都是原子的;

支持表级锁，即每次操作是对整个表加锁;

InnoDb:

支持 ACID 的事务，支持事务的四种隔离级别;

支持行级锁及外键约束：因此可以支持写并发;

不存储总行数;

1.9 什么是事务，Mysql 中的事务隔离级别有几种，事务的特性是什么?

事务（Transaction）是并发控制的基本单位。所谓的事务，它是一个操作序列，这些操作要么都执行，要么都不执行，它是一个不可分割的工作单位。事务是数据库维护数据一致性的单位，在每个事务结束时，都能保持数据一致性。

Mysql 中 InnoDB 支持的四种事务隔离级别： （由高到低）

1.串行化（serializable）：一个事务一个事务的执行

2.可重复读（Repeatable-Read） 可重复读，无论其他事务是否修改并提交了数据，在这个事务中看到的数据值始终不受其他事务影响 （mysql 数据库所默认的级别）

3.读已提交（Read Committed） 读取已提交，其他事务提交了对数据的修改后，本事务就能读取到修改后的数据值（大多数数据库默认的隔离级别）

4.读未提交：(Read Uncommitted) 读取未提交，其他事务只要修改了数据，即使未提交，本事务也能看到修改后的数据值

1.10 数据库的乐观锁和悲观锁是什么？

数据库管理系统（DBMS）中的并发控制的任务是确保在多个事务同时存取数据库中同一数据时不破坏事务的隔离性和统一性以及数据库的统一性。

乐观并发控制(乐观锁)和悲观并发控制（悲观锁）是并发控制主要采用的技术手段。

悲观锁：假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作。在对任意记录进行修改前，先尝试为该记录加上排他锁。

悲观并发控制实际上是“先取锁再访问”的保守策略，为数据处理的安全提供了保证。但是在效率方面，处理加锁的机制会让数据库产生额外的开销，还有增加产生死锁的机会；

乐观锁：假设不会发生并发冲突，只在提交操作时检查是否违反数据完整性。乐观锁假设认为数据一般情况下不会造成冲突，所以在数据进行提交更新的时候，才会正式对数据的冲突与否进行检测，如果发现冲突了，则让返回用户错误的信息，让用户决定如何去做。

一般的实现乐观锁的方式就是记录数据版本。为数据增加的一个版本标识。当读取数据时，将版本标识的值一同读出，数据每更新一次，同时对版本标识进行更新。当我们提交更新的时候，判断数据库表对应记录的当前版本信息与第一次取出来的版本标识进行比对，如果数据库表当前版本号与第一次取出来的版本标识值相等，则予以更新，否则认为是过期数据。

实现数据版本有两种方式，第一种是使用版本号，第二种是使用时间戳。

10.12 什么叫视图？游标是什么？

视图是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增，改，查，操作，视图通常是有一个表或者多个表的行或列的子集。对视图的修改不影响基本表。它使得我们获取数据更容易，相比多表查询。

游标：是对查询出来的结果集作为一个单元来有效的处理。游标可以定在该单元中的特定行，从结果集的当前行检索一行或多行。可以对结果集当前行做修改。一般不使用游标，但是需要逐条处理数据的时候，游标显得十分重要。

10.13 什么是存储过程？用什么来调用？

存储过程是一个预编译的 SQL 语句，优点是允许模块化的设计，就是说只需创建一次，以后在该程序中就可以调用多次。如果某次操作需要执行多次 SQL，使用存储过程比单纯 SQL 语句执行要快。可以用一个命令对象来调用存储过程。

二、sql 语句面试题

基本表结构：

student(sno,sname,sage,ssex)学生表

course(cno,cname,tno) 课程表

sc(sno,cno,score) 成绩表

teacher(tno,tname) 教师表

101，查询课程 1 的成绩比课程 2 的成绩高的所有学生的学号

```
select a.sno from
(select sno,score from sc where cno=1) a,
(select sno,score from sc where cno=2) b
where a.score>b.score and a.sno=b.sno
```

102，查询平均成绩大于 60 分的同学的学号和平均成绩

```
select a.sno as "学号", avg(a.score) as "平均成绩"
from
(select sno,score from sc) a
group by sno having avg(a.score)>60
```

103，查询所有同学的学号、姓名、选课数、总成绩

```
select a.sno as 学号, b.sname as 姓名,
count(a.cno) as 选课数, sum(a.score) as 总成绩
from sc a, student b
where a.sno = b.sno
group by a.sno, b.sname
```

或者：

```
selectstudent.sno as 学号, student.sname as 姓名,
 count(sc.cno) as 选课数, sum(score) as 总成绩
from student left Outer join sc on student.sno = sc.sno
group by student.sno, sname
```

104，查询姓“张”的老师的个数

```
selectcount(distinct(tname)) from teacher where tname like '张%‘
```

或者：

```
select tname as "姓名", count(distinct(tname)) as "人数"
from teacher
where tname like'张%'
group by tname
```

105，查询没学过“张三”老师课的同学的学号、姓名

```
select student.sno,student.sname from student
where sno not in (select distinct(sc.sno) from sc,course,teacher
where sc.cno=course.cno and teacher.tno=course.tno and teacher.tname='张三')
```

106，查询同时学过课程 1 和课程 2 的同学的学号、姓名

```
select sno, sname from student
where sno in (select sno from sc where sc.cno = 1)
and sno in (select sno from sc where sc.cno = 2)
```

或者：

```
selectc.sno, c.sname from
(select sno from sc where sc.cno = 1) a,
(select sno from sc where sc.cno = 2) b,
student c
where a.sno = b.sno and a.sno = c.sno
```

或者：

```
select student.sno,student.sname from student,sc where student.sno=sc.sno and
sc.cno=1
and exists( select * from sc as sc_2 where sc_2.sno=sc.sno and sc_2.cno=2)
```

107，查询学过“李四”老师所教所有课程的所有同学的学号、姓名

```
select a.sno, a.sname from student a, sc b
where a.sno = b.sno and b.cno in
(select c.cno from course c, teacher d where c.tno = d.tno and d.tname = '李四')
```

或者：

```
select a.sno, a.sname from student a, sc b,
(select c.cno from course c, teacher d where c.tno = d.tno and d.tname = '李四') e
where a.sno = b.sno and b.cno = e.cno
```

108，查询课程编号 1 的成绩比课程编号 2 的成绩高的所有同学的学号、姓名

```
select a.sno, a.sname from student a,
(select sno, score from sc where cno = 1) b,
(select sno, score from sc where cno = 2) c
where b.score > c.score and b.sno = c.sno and a.sno = b.sno
```

109，查询所有课程成绩小于 60 分的同学的学号、姓名

```
select sno,sname from student
where sno not in (select distinct sno from sc where score > 60)
```

110，查询至少有一门课程与学号为 1 的同学所学课程相同的同学的学号和姓名

```
select distinct a.sno, a.sname
from student a, sc b
where a.sno <> 1 and a.sno=b.sno and
```

```
b.cno in (select cno from sc where sno = 1)
```

或者：

```
select s.sno,s.sname
from student s,
(select sc.sno
from sc
where sc.cno in (select sc1.cno from sc sc1 where sc1.sno=1)and sc.sno<>1
group by sc.sno)r1
where r1.sno=s.sno
```

111、把“sc”表中“王五”所教课的成绩都更改为此课程的平均成绩

```
update sc set score = (select avg(sc_2.score) from sc sc_2 wheresc_2.cno=sc.cno)
from course,teacher where course.cno=sc.cno and course.tno=teacher.tno
andteacher.tname='王五'
```

112、查询和编号为 2 的同学学习的课程完全相同的其他同学学号和姓名

这一题分两步查：

1，

```
select sno
from sc
where sno <> 2
group by sno
having sum(cno) = (select sum(cno) from sc where sno = 2)
```

2，

```
select b.sno, b.sname
from sc a, student b
where b.sno <> 2 and a.sno = b.sno
group by b.sno, b.sname
having sum(cno) = (select sum(cno) from sc where sno = 2)
```

113、删除学习“王五”老师课的 sc 表记录

```
delete sc from course, teacher
where course.cno = sc.cno and course.tno = teacher.tno and tname = '王五'
```

114、向 sc 表中插入一些记录，这些记录要求符合以下条件：
将没有课程 3 成绩同学的该成绩补齐，其成绩取所有学生的课程 2 的平均成绩

```
insert sc select sno, 3, (select avg(score) from sc where cno = 2)
from student
where sno not in (select sno from sc where cno = 3)
```

115、按平平均分从高到低显示所有学生的如下统计报表：

```
-- 学号,企业管理,马克思,UML,数据库,物理,课程数,平均分
select sno as 学号
,max(case when cno = 1 then score end) AS 企业管理
```

```
,max(case when cno = 2 then score end) AS 马克思
,max(case when cno = 3 then score end) AS UML
,max(case when cno = 4 then score end) AS 数据库
,max(case when cno = 5 then score end) AS 物理
,count(cno) AS 课程数
,avg(score) AS 平均分
FROM sc
GROUP by sno
ORDER by avg(score) DESC
```

116、查询各科成绩最高分和最低分：
以如下形式显示：课程号，最高分，最低分

```
select cno as 课程号, max(score) as 最高分, min(score) 最低分
from sc group by cno

select course.cno as '课程号'
,MAX(score) as '最高分'
,MIN(score) as '最低分'
from sc,course
where sc.cno=course.cno
group by course.cno
```

117、按各科平均成绩从低到高和及格率的百分数从高到低顺序

```
SELECT t.cno AS 课程号,
max(course.cname)AS 课程名,
isnull(AVG(score),0) AS 平均成绩,
100 * SUM(CASE WHEN isnull(score,0)>=60 THEN 1 ELSE 0 END)/count(1) AS 及格率
FROM sc t, course
where t.cno = course.cno
GROUP BY t.cno
ORDER BY 及格率 desc
```

118、查询如下课程平均成绩和及格率的百分数(用"1 行"显示):
企业管理（001），马克思（002），UML （003），数据库（004）

```
select
avg(case when cno = 1 then score end) as 平均分 1,
avg(case when cno = 2 then score end) as 平均分 2,
avg(case when cno = 3 then score end) as 平均分 3,
avg(case when cno = 4 then score end) as 平均分 4,
100 * sum(case when cno = 1 and score > 60 then 1 else 0 end) / sum(casewhen cno = 1
then 1 else 0 end) as 及格率 1,
100 * sum(case when cno = 2 and score > 60 then 1 else 0 end) / sum(casewhen cno = 2
then 1 else 0 end) as 及格率 2,
100 * sum(case when cno = 3 and score > 60 then 1 else 0 end) / sum(casewhen cno = 3
```

```
then 1 else 0 end) as 及格率3,
100 * sum(case when cno = 4 and score > 60 then 1 else 0 end) / sum(casewhen cno = 4
then 1 else 0 end) as 及格率4
from sc
```

119、查询不同老师所教不同课程平均分，从高到低显示

```
select max(c.tname) as 教师, max(b.cname) 课程, avg(a.score) 平均分
from sc a, course b, teacher c
where a.cno = b.cno and b.tno = c.tno
group by a.cno
order by 平均分 desc
```

或者：

```
select r.tname as '教师',r.rname as '课程' , AVG(score) as '平均分'
from sc,
(select  t.tname,c.cno as rcso,c.cname as rname
from teacher t ,course c
where t.tno=c.tno)r
where sc.cno=r.rcso
group by sc.cno,r.tname,r.rname
order by AVG(score) desc
```

120、查询如下课程成绩均在第 3 名到第 6 名之间的学生的成绩：

-- [学生 ID],[学生姓名],企业管理,马克思,UML,数据库,平均成绩

```
select top 6 max(a.sno) 学号, max(b.sname) 姓名,
max(case when cno = 1 then score end) as 企业管理,
max(case when cno = 2 then score end) as 马克思,
max(case when cno = 3 then score end) as UML,
max(case when cno = 4 then score end) as 数据库,
avg(score) as 平均分
from sc a, student b
where a.sno not in

(select top 2 sno from sc where cno = 1 order by score desc)
  and a.sno not in (select top 2 sno from sc where cno = 2 order by scoredesc)
  and a.sno not in (select top 2 sno from sc where cno = 3 order by scoredesc)
  and a.sno not in (select top 2 sno from sc where cno = 4 order by scoredesc)
  and a.sno = b.sno
group by a.sno
```

三、说说对 SQL 语句优化有哪些方法？

（1）Where 子句中：where 表之间的连接必须写在其他 Where 条件之前，那些可以过滤掉最大数量记录的条件必须写在 Where 子句的末尾.HAVING 最后。

(2) 用 EXISTS 替代 IN、用 NOT EXISTS 替代 NOT IN。

(3) 避免在索引列上使用计算

(4) 避免在索引列上使用 IS NULL 和 IS NOT NULL

(5) 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。

(6) 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。

(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。