MySql 面试题

1、 数据库事务的四个特性及含义

数据库事务 transanction 正确执行的四个基本要素。ACID：原子性(Atomicity)、 一致性(Correspondence)、隔离性(Isolation)、持久性(Durability)。

• 原子性:整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback） 到事务开始前的状态，就像这个事务从来没有执行过一样。

• 一致性:在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。

• 隔离性:隔离状态执行事务，使它们好像是系统在给定时间内执行的唯一操作。如果有两个事务，运行在相同的时间内，执行相同的功能，事务的隔离性将确保每一事务在系统中认为只有该事务在使用系统。这种属性有时称为串行化，为了防止事务操作间的混淆，必须串行化或序列化请求，使得在同一时间仅有一个请求用于同一数据。

• 持久性:在事务完成以后，该事务所对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。

2、视图的作用，视图可以更改么?

视图是虚拟的表，与包含数据的表不一样，视图只包含使用时动态检索数据的查询；不包含任何列或数据。使用视图可以简化复杂的 sql 操作，隐藏具体的细节，保护数据；视图创建后，可以使用与表相同的方式利用它们。视图不能被索引，也不能有关联的触发器或默认值，如果视图本身内有 order by 则对视图再次 order by 将被覆盖。

创建视图：create view XXX as XXXXXXXXXXXXX;

对于某些视图比如未使用联结子查询分组聚集函数 Distinct Union 等，是可以对其更新的，对视图的更新将对基表进行更新；但是视图主要用于简化检索，保护数据，并不用于更新，而且大部分视图都不可以更新。

3、 drop,delete 与 truncate 的区别

参考答案： drop 直接删掉表 truncate 删除表中数据，再插入时自增长 id 又从 1 开始 delete 删除表中数据，可以加 where 字句。

（1） DELETE 语句执行删除的过程是每次从表中删除一行，并且同时将该行的删除操作作为事务记录在日志中保存以便进行进行回滚操作。TRUNCATE TABLE 则一次性地从表中删除所有的数据并不把单独的删除操作记录记入日志保存，删除行是不能恢复的。并且在删除的过程中不会激活与表有关的删除触发器。执行速度快。

（2） 表和索引所占空间。当表被 TRUNCATE 后，这个表和索引所占用的空间会恢复到初始大小，而 DELETE 操作不会减少表或索引所占用的空间。drop 语句将表所占用的空间全释放掉。

（3） 一般而言，drop > truncate > delete

（4） 应用范围。TRUNCATE 只能对 TABLE；DELETE 可以是 table 和 view

（5） TRUNCATE 和 DELETE 只删除数据，而 DROP 则删除整个表（结构和数据）。

（6） truncate 与不带 where 的 delete ： 只删除数据，而不删除表的结构（定义）drop 语句将删除表的结构被依赖的约束（constrain),触发器（trigger)索引 （index);依赖于该表的存储过程/函数将被保留，但其状态会变为：invalid。

（7）delete 语句为 DML（data maintain Language),这个操作会被放到 rollback segment 中,事务提交后才生效。如果有相应的 tigger,执行的时候将被触发。

（8） truncate、drop 是 DLL（data define language),操作立即生效，原数据不放到 rollback segment 中，不能回滚

（9） 在没有备份情况下，谨慎使用 drop 与 truncate。要删除部分数据行采用 delete 且注意结合 where 来约束影响范围。回滚段要足够大。要删除表用 drop; 若想保留表而将表中数据删除，如果于事务无关，用 truncate 即可实现。如果和事务有关，或老师想触发 trigger,还是用 delete。

（10） Truncate table 表名速度快,而且效率高,因为: truncate table 在功能上与不带 WHERE 子句的 DELETE 语句相同：二者均删除表中的全部行。但 TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少。 DELETE 语句每次删除一行，并在事务日志中为所删除的每行记录一项。 TRUNCATE TABLE 通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放。

（11） TRUNCATE TABLE 删除表中的所有行，但表结构及其列、约束、索引等保持不变。新行标识所用的计数值重置为该列的种子。如果想保留标识计数值，请改用 DELETE。如果要删除表定义及其数据，请使用 DROP TABLE 语句。

（12） 对于由 FOREIGN KEY 约束引用的表，不能使用 TRUNCATE TABLE，而应使用不带 WHERE 子句的 DELETE 语句。由于 TRUNCATE TABLE 不记录在 日志中，所以它不能激活触发器。

4、 索引的工作原理及其种类

数据库索引，是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。索引的实现通常使用 B 树及其变种 B+**树。**

在数据之外，数据库系统还维护着满足特定查找算法的数据结构，这些数据结构以某种方式引用（指向）数据，这样就可以在这些数据结构上实现高级查找算法。这种数据结构，就是索引。

为表设置索引要付出代价的：一是增加了数据库的存储空间，二是在插入和修改数据时要花费较多的时间(因为索引也要随之变动)。

![01](C:/Users/LXR/Documents/WeChat Files/wxid_gsvna0kjo9bj22/FileStorage/File/2021-03/img/01.png)

图展示了一种可能的索引方式。左边是数据表，一共有两列七条记录，最左边的 是数据记录的物理地址（注意逻辑上相邻的记录在磁盘上也并不是一定物理相邻的）。为了加快 Col2 的查找，可以维护一个右边所示的二叉查找树，每个节点分别包含索引键值和一个指向对应数据记录物理地址的指针，这样就可以运用二 叉查找在 **O(log2n)**的复杂度内获取到相应数据。

创建索引可以大大提高系统的性能。

第一，通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

第二，可以大大加快数据的检索速度，这也是创建索引的最主要的原因。

第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。

第四，在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。

第五，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。

也许会有人要问：增加索引有如此多的优点，为什么不对表中的每一个列创建一个索引呢？因为，增加索引也有许多不利的方面。

第一，创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。

第二，索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。

第三，当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。

索引是建立在数据库表中的某些列的上面。在创建索引的时候，应该考虑在哪些列上可以创建索引，在哪些列上不能创建索引。一般来说，应该在这些列上创建索引：在经常需要搜索的列上，可以加快搜索的速度；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；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；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；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；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。

同样，对于有些列不应该创建索引。一般来说，不应该创建索引的的这些列具有下列特点

第一，对于那些在查询中很少使用或者参考的列不应该创建索引。这是因为，既然这些列很少使用到，因此有索引或者无索引，并不能提高查询速度。相反，由于增加了索引，反而降低了系统的维护速度和增大了空间需求。

第二，对于那些只有很少数据值的列也不应该增加索引。这是因为，由于这些列的取值很少，例如人事表的性别列，在查询的结果中，结果集的数据行占了表中数据行的很大比例，即需要在表中搜索的数据行的比例很大。增加索引，并不能明显加快检索速度。

第三，对于那些定义为 text, image 和 bit 数据类型的列不应该增加索引。这是因为，这些列的数据量要么相当大，要么取值很少。

第四，当修改性能远远大于检索性能时，不应该创建索引。这是因为，修改性能和检索性能是互相矛盾的。当增加索引时，会提高检索性能，但是会降低修改性能。当减少索引时，会提高修改性能，降低检索性能。因此，当修改性能远远大于检索性能时，不应该创建索引。根据数据库的功能，可以在数据库设计器中创建三种索引：唯一索引、主键索引和聚集索引。

唯一索引

唯一索引是不允许其中任何两行具有相同索引值的索引。当现有数据中存在重复的键值时，大多数数据库不允许将新创建的唯一索引与表一起保存。数据库还可能防止添加将在表中创建重复键值的新数据。例如，如果在employee 表中职员的姓(**lname)**上创建了唯一索引，则任何两个员工都不能同姓。主键索引数据库表经常有一列或列组合，其值唯一标识表中的每一行。该列称为表的主键。在数据库关系图中为表定义主键将自动创建主键索引，主键索引是唯一索引的特定类型。该索引要求主键中的每个值都唯一。当在查询中使用主键索引时，它还允许对数据的快速访问。聚集索引在聚集索引中，表中行的物理顺序与键值的逻辑（索引）顺序相同。一个表只能包含一个聚集索引。

如果某索引不是聚集索引，则表中行的物理顺序与键值的逻辑顺序不匹配。与非聚集索引相比，聚集索引通常提供更快的数据访问速度。

局部性原理与磁盘预读由于存储介质的特性，磁盘本身存取就比主存慢很多，再加上机械运动耗费，磁盘的存取速度往往是主存的几百分分之一，因此为了提高效率，要尽量减少磁盘 I/O。为了达到这个目的，磁盘往往不是严格按需读取，而是每次都会预读，即使只需要一个字节，磁盘也会从这个位置开始，顺序向后读取一定长度的数据放入内存。这样做的理论依据是计算机科学中著名的局部性原理：当一个数据被用到时，其附近的数据也通常会马上被使用。程序运行期间所需要的数据通常比较集中。

由于磁盘顺序读取的效率很高（不需要寻道时间，只需很少的旋转时间），因此对于具有局部性的程序来说，预读可以提高 I/O 效率。

预读的长度一般为页（page）的整倍数。页是计算机管理存储器的逻辑块，硬件及操作系统往往将主存和磁盘存储区分割为连续的大小相等的块，每个存储块称为一页（在许多操作系统中，页得大小通常为 4k），主存和磁盘以页为单位交换数据。当程序要读取的数据不在主存中时，会触发一个缺页异常，此时系统会向磁盘发出读盘信号，磁盘会找到数据的起始位置并向后连续读取一页或几页载入内存中，然后异常返回，程序继续运行。

B-/+Tree 索引的性能分析到这里终于可以分析 B-/+Tree 索引的性能了。

上文说过一般使用磁盘 I/O 次数评价索引结构的优劣。先从 B-Tree 分析，根据 B-Tree 的定义，可知检索一次最多需要访问 h 个节点。数据库系统的设计者巧妙利用了磁盘预读原理，将一个节点的大小设为等于一个页，这样每个节点只需要一次 I/O 就可以完全载入。为了达到这个目的，在实际实现 B-Tree 还需要使用如下技巧：

每次新建节点时，直接申请一个页的空间，这样就保证一个节点物理上也存储在 一个页里，加之计算机存储分配都是按页对齐的，就实现了一个 node 只需一次 I/O。

B-Tree 中一次检索最多需要 h-1 次 I/O（根节点常驻内存），渐进复杂度为 O(h)=O(logdN)**。一般实际应用中，出度**d 是非常大的数字，通常超过 100，因此 h 非常小（通常不超过 3）。

而红黑树这种结构，h 明显要深的多。由于逻辑上很近的节点（父子）物理上可能很远，无法利用局部性，所以红黑树的 I/O 渐进复杂度也为 O(h)**，效率明显比 **B-Tree 差很多。

综上所述，用 B-Tree 作为索引结构效率是非常高的。

5、 连接的种类

参考答案：

查询分析器中执行：

```
--建表 table1,table2：
create table table1(id int,name varchar(10)) create table table2(id int,score int)
insert into table1 select 1,'lee'
insert into table1 select 2,'zhang'
insert into table1 select 4,'wang' insert into table2 select 1,90 insert into table2 select 2,100 insert into
```

如表:

```
---------------------------------------------- table1 | table2 |
---------------------------------------------- id name |id score |
1 lee |1 90|
2 zhang| 2 100|
4 wang| 3 70|
-----------------------------------------------
```

以下均在查询分析器中执行

一、外连接

1.概念：包括左向外联接、右向外联接 或完整外部联接

2.左连接：**left join 或 left outer join (1)****左向外联接的结果集包括 **LEFT OUTER** 子句中指定的左表的所有行，而不仅仅是联接列所匹配的行。如果左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为空值 (null)**。 (2)**sql** 语句

```
select * from table1 left join table2 on table1.id=table2.id
-------------结果-------------
idnameidscore
------------------------------
1lee190
2zhang2100
4wangNULLNULL ------------------------------
```

注释：包含 table1 的所有子句，根据指定条件返回 table2 相应的字段，不符合的以 null 显示

3.右连接：right join 或 right outer join (1)右向外联接是左向外联接的反向联接。 将返回右表的所有行。如果右表的某行在左表中没有匹配行，则将为左表返回空 值。

(2)sql语句

```
elect * from table1 right join table2 on table1.id=table2.id
-------------结果-------------
idnameidscore
------------------------------ 1lee190
2zhang2100
NULLNULL370 ------------------------------
```

注释：包含 table2 的所有子句，根据指定条件返回 table1 相应的字段，不符合 的以 null 显示

4.完整外部联接:**full join** 或 **full outer join **

(1)完整外部联接返回左表和右表中的所有行。当某行在另一个表中没有匹配行时，则另一个表的选择列表列包含空值。如果表之间有匹配行，则整个结果集行包含基表的数据值。

(2)sql 语句

```
select * from table1 full join table2 on table1.id=table2.id
-------------结果-------------
idnameidscore
------------------------------
1lee190
2zhang2100
4wangNULLNULL
NULLNULL370
------------------------------
```

注释：返回左右连接的和（见上左、右连接）

二、内连接

1.概念：内联接是用比较运算符比较要联接列的值的联接

2.内连接：join 或 inner join

3.sql 语句

```
select * from table1 join table2 on table1.id=table2.id
-------------结果-------------
idnameidscore
------------------------------
1lee190
2zhang2100
------------------------------
```

注释：只返回符合条件的 table1 和 table2 的列

4.等价（与下列执行效果相同）

```
A:select a.*,b.* from table1 a,table2 b where a.id=b.id
B:select * from table1 cross join table2 where table1.id=table2.id (注：cross join 后加条件只能用 where,不能用 o
```

三、交叉连接(完全)

1.概念：没有 **WHERE** 子句的交叉联接将产生联接所涉及的表的笛卡尔积。第一个表的行数乘以第二个表的行数等于笛卡尔积结果集的大小。（**table1** 和 **table2** 交叉连接产生 *3*3=9** 条记录）

2.交叉连接：**cross join** (不带条件 **where...)**

3.sql 语句

```
select * from table1 cross join table2
-------------结果-------------
idnameidscore
------------------------------
1lee190
2zhang190
4wang190
1lee2100
2zhang2100
4wang2100
1lee370
2zhang370
4wang370
------------------------------
```

注释：返回 *33=9** 条记录，即笛卡尔积

4.等价（与下列执行效果相同）

```
select * from table1,table2
```

6 、数据库范式

1 第一范式（1NF）

在任何一个关系数据库中，第一范式（1NF）是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系数据库。所谓第一范式（1NF）是指数据库表的每一列都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。如果出现重复的属性，就可能需要定义一个新的实体，新的实体由重复的属性构成，新实体与原实体之间为一对多关系。在第一范式（1NF）中表的每一行只包含一个实例的信息。简而言之，第一范式就是无重复的列。

2 第二范式（2NF） 第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。第二范式（2NF）要求数据库表中的每个实例或行必须可以被惟一地区分。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。这个惟一属性列被称为主关键字或主键、主码。第二 范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，第二范式就是非主属性非部分依赖于主关键字。

3 第三范式（3NF） 满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，第三范式（b）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。例如，存在一个部门信息表，其中每个部门有部门编号（dept_id）、部门名称、部门简介等信息。那么在员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。如果不存在部门信息表，则根据第三范式（3NF）也应该构建它，否则就会有大量的数据冗余。简而言之，第三范式就是属性不依赖于其它非主属性。（我的理解是消除冗余）

7、数据库优化的思路

1.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;
```

不是非我杠精，关于 null,isNull,isNotNull 其实是要看成本的，是否回表等因素总和考虑，才会决定是要走索引还是走全表扫描

也给大家找了一个作者的博文（MySQL 中 IS NULL、IS NOT NULL、!=不能用索 引？胡扯！），仅供参考！！！

[zhiyong0804d 的意见] 之所以未把第二条删除还是考虑可能很多人都被误导了。那这样的组织能让大家兼听则明。

3）很多时候用 exists 代替 in 是一个好的选择

4）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

**2.索引优化 **

3.数据库结构优化

1）范式优化： 比如消除冗余（节省空间。。）

2）反范式优化：比如适当加冗余等（减少 join）

3）拆分表： 分区将数据在物理上分隔开，不同分区的数据可以制定保存在处于不同磁盘上的数据文件里。这样，当对这个表进行查询时，只需要在表分区中进行扫描，而不必进行全表扫描，明显缩短了查询时间，另外处于不同磁盘的分区也将对这个表的数据传输分散在不同的磁盘 I/O，一个精心设置的分区可以将数据传输对磁盘 I/O 竞争均匀地分散开。对数据量大的时时表可采取此方法。可按月自动建表分区。

4）拆分其实又分垂直拆分和水平拆分： 案例： 简单购物系统暂设涉及如下表：

```
1.产品表（数据量 10w，稳定）
2.订单表（数据量 200w，且有增长趋势）
3.户表 （数据量 100w，且有增长趋势）
```

以 mysql 为例讲述下水平拆分和垂直 拆分，mysql 能容忍的数量级在百万静态数据可以到千万垂直拆分：解决问题：表与表之间的 io 竞争不解决问题：单表中数据量增长出现的压力方案： 把产品表和用户表放到一个 server 上订单表单独放到一个 server 上水平拆分： 解决问题：单表中数据量增长出现的压力不解决问题：表与表之间的 io 争夺方案：用户表通过性别拆分为男用户表和女用户表订单表通过已完成和完成中拆分为已完成订单和未完成订单产品表未完成订单放一个 server 上已完成订单表盒男用户表放一个 server 上 女用户表放一个 server 上(女的爱购物 哈哈)

4.服务器硬件优化 这个么多花钱咯!

8、 存储过程与触发器的区别

触发器与存储过程非常相似，触发器也是 SQL 语句集，两者唯一的区别是触发 器不能用 EXECUTE 语句调用，而是在用户执行 Transact-SQL 语句时自动触发（激 活）执行。触发器是在一个修改了指定表中的数据时执行的存储过程。通常通过 创建触发器来强制实现不同表中的逻辑相关数据的引用完整性和一致性。由于用 户不能绕过触发器，所以可以用它来强制实施复杂的业务规则，以确保数据的完 整性。触发器不同于存储过程，触发器主要是通过事件执行触发而被执行的，而 存储过程可以通过存储过程名称名字而直接调用。当对某一表进行诸如 UPDATE、 INSERT、DELETE 这些操作时，SQLSERVER 就会自动执行触发器所定义的 SQL 语句，从而确保对数据的处理必须符合这些 SQL 语句所定义的规则。

9、数据库三范式是什么?

1. 第一范式（1NF）：字段具有原子性,不可再分。(所有关系型数据库系统都满足第一范式数据库表中的字段都是单一属性的，不可再分)
2. 第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。要求数据库表中的每个实例或行必须可以被惟一地区分。通常需要为表加上一个列，以存储各个实例的惟一标识。这个惟一属性列被称为主关键字或主键。
3. 满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。

   所以第三范式具有如下特征：

   i. 每一列只有一个值
   ii. 每一行都能区分。
   iii. 每一个表都不包含其他表已经包含的非主关键字信息。

10、有哪些数据库优化方面的经验?

1. 用 PreparedStatement， 一般来说比 Statement 性能高：一个 sql 发给服务器去执行，涉及步骤：语法检查、语义分析， 编译，缓存。

2. 有外键约束会影响插入和删除性能，如果程序能够保证数据的完整性，那在设计数据库时就去掉外键。
3. 表中允许适当冗余，譬如，主题帖的回复数量和最后回复时间等
4. UNION ALL 要比 UNION 快很多，所以，如果可以确认合并的两个结果集中不包含重复数据且不需要排序时的话，那么就使用 UNION ALL。 >>UNION 和 UNION ALL 关键字都是将两个结果集合并为一个，但这两者从使用和效率上来说都有所不同。

> i. 对重复结果的处理：UNION 在进行表链接后会筛选掉重复的记录，Union All 不会去除重复记录。
> ii. 对排序的处理：Union 将会按照字段的顺序进行排序；UNION ALL 只是简单的将两个结果合并后就返回

11、请简述常用的索引有哪些种类?

1. 普通索引: 即针对数据库表创建索引
2. 唯一索引: 与普通索引类似，不同的就是：MySQL 数据库索引列的值必须唯一，但允许有空值
3. 主键索引: 它是一种特殊的唯一索引，不允许有空值。一般是在建表的时候同时创建主键索引
4. 组合索引: 为了进一步榨取 MySQL 的效率，就要考虑建立组合索引。即将数据库表中的多个字段联合起来作为一个组合索引

12、以及在mysql数据库中索引的工作机制是什么?

数据库索引，是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。索引的实现通常使用 B 树及其变种 B+树

13、MySQL的基础操作命令

1. MySQL 是否处于运行状态:Debian 上运行命令 service mysql status，在 RedHat 上运行命令 service mysqld status
2. 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
3. Shell 登入 MySQL: 运行命令 mysql -u root -p
4. 列出所有数据库:运行命令 show databases;
5. 切换到某个数据库并在上面工作:运行命令 use database*name;* 进入名为 *database*name 的数据库
6. 列出某个数据库内所有表: show tables;
7. 获取表内所有 Field 对象的名称和类型 :describe table_name;

14、Mysql 的复制原理以及流程。

Mysql 内建的复制功能是构建大型，高性能应用程序的基础。将 Mysql 的数据分布到多个系统上去，这种分布的机制，是通过将 Mysql 的某一台主机的数据复制到其它主机（slaves）上，并重新执行一遍来实现的。 * 复制过程中一个服务器充当主服务器，而一个或多个其它服务器充当从服务器。主服务器将更新写入二进制日志文件，并维护文件的一个索引以跟踪日志循环。这些日志可以记录发送到从服务器的更新。 当一个从服务器连接主服务器时，它通知主服务器在日志中读取的最后一次成功更新的位置。从服务器接收从那时起发生的任何更新，然后封锁并等待主服务器通知新的更新。 过程如下 1. 主服务器把更新记录到二进制日志文件中。 2. 从服务器把主服务器的二进制日志拷贝到自己的中继日志（replay log）中。 3. 从服务器重做中继日志中的时间，把更新应用到自己的数据库上。

15、mysql 支持的复制类型

1. 基于语句的复制： 在主服务器上执行的 SQL 语句，在从服务器上执行同样的语句。MySQL 默认采用基于语句的复制，效率比较高。 一旦发现没法精确复制时，会自动选着基于行的复制。
2. 基于行的复制：把改变的内容复制过去，而不是把命令在从服务器上执行一遍. 从 mysql5.0 开始支持

3. 混合类型的复制: 默认采用基于语句的复制，一旦发现基于语句的无法精确的复制时，就会采用基于行的复制。

16、mysql中 myisam 与 innodb 的区别?

1. 事务支持

> MyISAM：强调的是性能，每次查询具有原子性,其执行数度比InnoDB类型更快，但是不提供事务支持。
>
> InnoDB：提供事务支持事务，外部键等高级数据库功能。 具有事务(commit)、回滚(rollback)和崩溃修复能力(crash recovery capabilities)的事务安全(transaction-safe (ACID compliant))型表。

2. InnoDB 支持行级锁，而 MyISAM 支持表级锁. >> 用户在操作 myisam 表时，select，update，delete，insert 语句都会给表自动加锁，如果加锁以后的表满足 insert 并发的情况下，可以在表的尾部插入新的数据。

3. InnoDB 支持 MVCC, 而 MyISAM 不支持

4. InnoDB 支持外键，而 MyISAM 不支持

5. 表主键

   MyISAM：允许没有任何索引和主键的表存在，索引都是保存行的地址。

   > InnoDB：如果没有设定主键或者非空唯一索引，就会自动生成一个 6 字节的主键(用户不可见)，数据是主索引的一部分，附加索引保存的是主索引的值。

6. InnoDB 不支持全文索引，而 MyISAM 支持。

7. 可移植性、备份及恢复

> MyISAM：数据是以文件的形式存储，所以在跨平台的数据转移中会很方便。在备份和恢复时可单独针对某个表进行操作。
>
> InnoDB：免费的方案可以是拷贝数据文件、备份 binlog，或者用 mysqldump，在数据量达到几十 G 的时候就相对痛苦了

8. 存储结构

> MyISAM：每个MyISAM在磁盘上存储成三个文件。第一个文件的名字以表的名字开始，扩展名指出文件类型。.frm文件存储表定义。数据文件的扩展名为.MYD (MYData)。索引文件的扩展名是.MYI (MYIndex)。
>
> InnoDB：所有的表都保存在同一个数据文件中（也可能是多个文件，或者是独立的表空间文件），InnoDB 表的大小只受限于操作系统文件的大小，一般为 2GB。

17、mysql 中 varchar 与 char 的区别以及 varchar(50)中的 50 代表的涵义?

1. varchar 与 char 的区别: char 是一种固定长度的类型，varchar 则是一种可变长度的类型.

2. varchar(50)中 50 的涵义 : 最多存放 50 个字节

3. int（20）中 20 的涵义: int(M)中的 M indicates the maximum display width (最大显示宽度)for integer types. The maximum legal display width is 255.

18、MySQL 中 InnoDB 支持的四种事务隔离级别名称，以及逐级之间的区别?

1. 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）。

2. 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。

3. 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。InnoDB 和 Falcon 存储引擎通过多版本并发控制

   （MVCC，Multiversion Concurrency Control 间隙锁）机制解决了该问题。注：其实多版本只是解决不可重复读 问题，而加上间隙锁（也就是它这里所谓的并发控制）才解决了幻读问题。

4. 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。

19、表中有大字段 X（例如：text 类型），且字段 X 不会经常更新，以读为为主，将该字段拆成子表好处是什么?

如果字段里面有大字段（text,blob)类型的，而且这些字段的访问并不多，这时候放在一起就变成缺点了。 MYSQL 数据库的记录存储是按行存储的，数据块大小又是固定的（16K），每条记录越小，相同的块存储的记录就越多。此时应该把大字段拆走，这样应付大部分小字段的查询时，就能提高效率。当需要查询大字段时，此时的关联查询是不可避免的，但也是值得的。拆分开后，对字段的 UPDAE 就要 UPDATE 多个表了 。

20、MySQL中 InnoDB 引擎的行锁是通过加在什么上完成（或称实现）的?

InnoDB 行锁是通过给索引上的索引项加锁来实现的，这一点 MySQL 与 Oracle 不同，后者是通过在数据块中对相应数据行加锁来实现的。InnoDB 这种行锁实现特点意味着：只有通过索引条件检索数据，InnoDB 才使用行级锁，否则，InnoDB 将使用表锁!

21、MySQL中控制内存分配的全局参数，有哪些?

1. Keybuffersize：

   keybuffersize 指定索引缓冲区的大小，它决定索引处理的速度，尤其是索引读的速度。通过检查状态值

   Keyreadrequests 和 Keyreads，可以知道*keybuffersize*设置是否合理。比例*keyreads /keyreadrequests* 应该尽可能的低，至少是*1:100*，*1:1000* 更好（上述状态值可以使用 *SHOW STATUS LIKE 'keyread%'***获得）。 *keybuffersize*只对*MyISAM*表起作用。即使你不使用*MyISAM*表，但是内部的临时磁盘表是*MyISAM*表，也要使用该值。可以使用检查状态值*createdtmpdisk*tables 得知详情。

   对于 1G 内存的机器，如果不使用 MyISAM 表，推荐值是 16M（8-64M）

keybuffersize 设置注意事项

1）单个 keybuffer的大小不能超过4G，如果设置超过4G，就有可能遇到下面3个

*bug: >>>>> http://bugs.mysql.com/bug.php?id=29446

*/> >>>>> http://bugs.mysql.com/bug.php?id=29419

*/> >>>>> http://bugs.mysql.com/bug.php?id=5731

2）建议keybuffer 设置为物理内存的 1/4(针对 MyISAM 擎)，甚至是物理内存的 30%~40%，如果 keybuffersize 设置太大，系统就会频繁的换页，降低系统性能。因为 MySQL 使用操作系统的缓存来 缓存数据，所以我们得为系统留够足够的内存；在很多情况下数据要比索引大得多。

3）如果机器性能优越，可以设置多个 keybuffer,分别让不同的keybuffer 来缓存专门的索引

2. innodbbufferpool_size

   表示缓冲池字节大小，InnoDB缓存表和索引数据的内存区域。mysql默认的值是128M。最大值与你的CPU体系结构有关，在32位操作系统，最大值是4294967295(2^32-1)，在64位操作系统，最大值为18446744073709551615 (2^64-1)*。> 在 *32* 位操作系统中，*CPU* 和操作系统实用的最大大小低于设置的最大值。如果设定的缓冲池的大小大于 *1G*，设置 *innodbbuffer*poolinstances的值大于1. > 数据读写在内存中非常快**, innodbbufferpoolsize 减少了对磁盘的读写。 当数据提交或满足检查点条件后才一次性将内存数据刷新到磁盘中。然而内存还有操作系统或数据库其他进程使用, 一般设置 buffer pool 大小为总内存的 3/4 至 4/5。 若设置不当, 内存使用可能浪费或者使用过多。 对于繁忙的服务器, buffer pool 将划分为多个实例以提高系统并发性, 减少线程间读写缓存的争用。buffer pool 的大小首先受 innodbbufferpool_instances 影响, 当然影响较小。

3. query*cache*size

   当mysql接收到一条select类型的query时，mysql会对这条query进行hash计算而得到一个hash值，然后通过该hash值到query cache中去匹配，如果没有匹配中，则将这个hash值存放在一个hash链表中，同时将query的结果集存放进cache中，存放hash值的链表的每一个hash节点存放了相应query结果集在cache中的地址，以及该query所涉及到的一些table的相关信息；如果通过hash值匹配到了一样的query，则直接将cache中相应的query结果集返回给客户端。如果mysql任何一个表中的任何一条数据发生了变化，便会通知query cache需要与该ctable相关的 query的cache全部失效，并释放占用的内存地址。

   query cache 优缺点

   > i. query 语句的 hash 计算和 hash 查找带来的资源消耗。mysql 会对每条接收到的 select 类型的 query 进行 hash 计算然后查找该 query 的 cache 是否存在，虽然 hash 计算和查找的效率已经足够高了，一条 query 所带来的消耗可以忽略，但一旦涉及到高并发，有成千上万条 query 时，hash 计算和查找所带来的开销就的重视了；
   > ii. query cache 的失效问题。如果表变更比较频繁，则会造成 query cache 的失效率非常高。表变更不仅仅指表中的数据发生变化，还包括结构或者索引的任何变化；
   > iii. 对于不同 sql 但同一结果集的 query 都会被缓存，这样便会造成内存资源的过渡消耗。sql 的字符大小写、空格或者注释的不同，缓存都是认为是不同的 sql（因为他们的 hash 值会不同）；
   > iv. 相关参数设置不合理会造成大量内存碎片，相关的参数设置会稍后介绍。

4. read*buffer*size

   是 MySQL 读入缓冲区大小。对表进行顺序扫描的请求将分配一个读入缓冲区，MySQL 会为它分配一段内存缓冲区。

read*buffer*size 变量控制这一缓冲区的大小。如果对表的顺序扫描请求非常频繁，并且你认为频繁扫描进行得太慢，可以通过增加该变量值以及内存缓冲区大小提高其性能。

22、若一张表中只有一个字段VARCHAR(N)类型，utf8 编码，则 N 最大值为多少(精确到数量级即可)?

由于 utf8 的每个字符最多占用 3 个字节。而 MySQL 定义行的长度不能超过 65535，因此 N 的最大值计算方法为：(65535-1-2)/3。减去 1 的原因是实际存储从第二个字节开始，减去 2 的原因是因为要在列表长度存储实际的字符长度，除以 3 是因为 utf8 限制：每个字符最多占用 3 个字节。

23、[SELECT] 和[SELECT全部字段]的2 种写法有何优缺点?

1） 前者要解析数据字典，后者不需要

2） 结果输出顺序，前者与建表列顺序相同，后者按指定字段顺序。

3） 表字段改名，前者不需要修改，后者需要改

4） 后者可以建立索引进行优化，前者无法优化

5） 后者的可读性比前者要高

24、HAVNG子句 和 WHERE 的异同点?

1） 语法上：where 用表中列名，having 用 select 结果别名

2） 影响结果范围：where 从表读出数据的行数，having 返回客户端的行数

3） 索引：where 可以使用索引，having 不能使用索引，只能在临时结果集操作

4） where 后面不能使用聚集函数，having 是专门使用聚集函数的。

25、MySQL当记录不存在时insert,当记录存在时update，语句怎么写?

```
INSERT INTO table (a,b,c) VALUES (1,2,3) ON DUPLICATE KEY UPDATE c=c+1;
```

26、MySQL 的 insert 和 update 的 select 语句语法

```
insert into student (stuid,stuname,deptid) select 10,'xzm',3 from student where stuid > 8;

update student a inner join student b on b.stuID=10 set a.stuname=concat(b.stuname, b.stuID) where a.stuID=10
```

27、一张表，里面有 ID 自增主键，当 insert 了 17 条记录之后，删除了第 15,16,17 条记录，再把 Mysql 重启，再 insert 一条记录，这条记录的 ID 是 18 还是 15 ？

（1）如果表的类型是 MyISAM，那么是 18

因为 MyISAM 表会把自增主键的最大 ID 记录到数据文件里，重启 MySQL 自增主键的最大ID 也不会丢失

（2）如果表的类型是 InnoDB，那么是 15

InnoDB 表只是把自增主键的最大 ID 记录到内存中，所以重启数据库或者是对表进行OPTIMIZE 操作，都会导致最大 ID 丢失

28、Mysql 的技术特点是什么?

Mysql 数据库软件是一个客户端或服务器系统，其中包括：支持各种客户端程序和库的多

线程 SQL 服务器、不同的后端、广泛的应用程序编程接口和管理工具。

29、Heap 表是什么?

HEAP 表存在于内存中，用于临时高速存储。

BLOB 或 TEXT 字段是不允许的

只能使用比较运算符=，<，>，=>，= <

HEAP 表不支持 AUTO_INCREMENT 索引不可为 NULL

30、Mysql 服务器默认端口是什么?

Mysql 服务器的默认端口是 3306。

31、与 Oracle 相比，Mysql 有什么优势?

Mysql 是开源软件，随时可用，无需付费。

Mysql 是便携式的带有命令提示符的 GUI。

使用 Mysql 查询浏览器支持管理

32、如何区分 FLOAT 和 DOUBLE?

以下是 FLOAT 和 DOUBLE 的区别：

浮点数以 8 位精度存储在 FLOAT 中，并且有四个字节。浮点数存储在 DOUBLE 中，精度为 18 位，有八个字节。

33、区分 CHAR_LENGTH 和 LENGTH?

CHAR_LENGTH 是字符数，而 LENGTH 是字节数。Latin 字符的这两个数据是相同的，但是对于 Unicode 和其他编码，它们是不同的。

34、 请简洁描述 Mysql 中 InnoDB 支持的四种事务隔离级别名称，以及逐级之间的区别?

SQL 标准定义的四个隔离级别为：

read uncommited ： 读到未提交数据

read committed：脏读，不可重复读

repeatable read：可重读 serializable ： 串行事物

35、在 Mysql 中 ENUM 的用法是什么?

ENUM 是一个字符串对象，用于指定一组预定义的值，并可在创建表时使用。

```
create table size(name ENUM('Smail,'Medium','Large');
```

36、如何定义 REGEXP?

REGEXP 是模式匹配，其中匹配模式在搜索值的任何位置。

37、CHAR 和 VARCHAR 的区别?

以下是 CHAR 和 VARCHAR 的区别：

CHAR 和 VARCHAR 类型在存储和检索方面有所不同

CHAR 列长度固定为创建表时声明的长度，长度值范围是 1 到 255

当 CHAR 值被存储时，它们被用空格填充到特定长度，检索 CHAR 值时需删除尾随空格。

38、 列的字符串类型可以是什么?

字符串类型是：

SET

BLOB

ENUM

CHAR

TEXT

VARCHAR

39、如何获取当前的 Mysql 版本?

SELECT VERSION();用于获取当前 Mysql 的版本。

40、Mysql 中使用什么存储引擎?存储引擎称为表类型,数据使用各种技术存储在文件中。

技术涉及:

Storage mechanism

Locking levels

Indexing

Capabilities and functions.

41、Mysql 驱动程序是什么?

以下是 Mysql 中可用的驱动程序:

PHP 驱动程序

JDBC 驱动程序

ODBC 驱动程序

CWRAPPER

PYTHON 驱动程序

PERL 驱动程序

RUBY 驱动程序

CAP11PHP 驱动程序

Ado.net5.mxj

42、TIMESTAMP 在 UPDATE CURRENT_TIMESTAMP 数据类型上做什么?

创建表时 TIMESTAMP 列用 Zero 更新。只要表中的其他字段发生更改,UPDATE CURRENT_TIMESTAMP 修饰符就将时间戳字段更新为当前时间。

43、 主键和候选键有什么区别?

表格的每一行都由主键唯一标识,一个表只有一个主键。

主键也是候选键。按照惯例,候选键可以被指定为主键,并且可以用于任何外键引用。

44、如何使用 Unix shell 登录 Mysql?

我们可以通过以下命令登录:

[mysql dir]/bin/mysql -h hostname -u

45、myisamchk 是用来做什么的?

它用来压缩 MyISAM 表,这减少了磁盘或内存使用。

46、如何控制 HEAP 表的最大尺寸?

Heal 表的大小可通过称为 max_heap_table_size 的 Mysql 配置变量来控制。

47、MyISAM Static 和 MyISAM Dynamic 有什么区别?

在 MyISAM Static 上的所有字段有固定宽度。动态 MyISAM 表将具有像 TEXT，BLOB 等字段，以适应不同长度的数据类型。点击这里有一套最全阿里面试题总结。

MyISAM Static 在受损情况下更容易恢复。

48、federated 表是什么?

federated 表，允许访问位于其他服务器数据库上的表。

49、 如果一个表有一列定义为 TIMESTAMP，将发生什么?

每当行被更改时，时间戳字段将获取当前时间戳。

50、列设置为 AUTO INCREMENT 时，如果在表中达到最大值，会发生什么情况?

它会停止递增，任何进一步的插入都将产生错误，因为密钥已被使用。

51、怎样才能找出最后一次插入时分配了哪个自动增量?

LAST_INSERT_ID 将返回由 Auto_increment 分配的最后一个值，并且不需要指定表名称。

52、你怎么看到为表格定义的所有索引?

索引是通过以下方式为表格定义的：

SHOW INDEX FROM

53、LIKE 声明中的%和_是什么意思?

%对应于 0 个或更多字符，_只是 LIKE 语句中的一个字符。

54、如何在 Unix 和 Mysql 时间戳之间进行转换?

UNIX_TIMESTAMP 是从 Mysql 时间戳转换为 Unix 时间戳的命令

FROM_UNIXTIME 是从 Unix 时间戳转换为 Mysql 时间戳的命令

55、列对比运算符是什么?

在 SELECT 语句的列比较中使用=，<>，<=，<，> =，>，<<，>>，<=>，AND，OR 或 LIKE 运算符。

56、我们如何得到受查询影响的行数?

行数可以通过以下代码获得：

```
SELECT COUNT(user_id)FROM users;
```

57、Mysql 查询是否区分大小写?

不区分

```
SELECT VERSION(), CURRENT_DATE;

SeLect version(), current_date;

seleCt vErSiOn(), current_DATE;
```

所有这些例子都是一样的，Mysql 不区分大小写。

58、LIKE 和 REGEXP 操作有什么区别?

LIKE 和 REGEXP 运算符用于表示^和%。

```
SELECT * FROM employee WHERE emp_name REGEXP "^b";

SELECT * FROM employee WHERE emp_name LIKE "%b";
```

59、BLOB 和 TEXT 有什么区别?

BLOB 是一个二进制对象，可以容纳可变数量的数据。有四种类型的 BLOB

TINYBLOB

BLOB

MEDIUMBLOB 和 LONGBLOB 它们只能在所能容纳价值的最大长度上有所不同。

TEXT 是一个不区分大小写的 BLOB。四种 TEXT 类型

TINYTEXT

TEXT

MEDIUMTEXT 和 LONGTEXT

它们对应于四种 BLOB 类型，并具有相同的最大长度和存储要求。

BLOB 和 TEXT 类型之间的唯一区别在于对 BLOB 值进行排序和比较时区分大小写，对 TEXT 值不区分大小写。

60、mysql_fetch_array 和 mysql_fetch_object 的区别是什么?

以下是 mysql_fetch_array 和 mysql_fetch_object 的区别：

mysql_fetch_array（） - 将结果行作为关联数组或来自数据库的常规数组返回。

mysql_fetch_object - 从数据库返回结果行作为对象。

61、我们如何在 mysql 中运行批处理模式?

以下命令用于在批处理模式下运行：

mysql;

mysql mysql.out

62、MyISAM 表格将在哪里存储，并且还提供其存储格式?

每个 MyISAM 表格以三种格式存储在磁盘上：

·".frm"文件存储表定义

·数据文件具有".MYD"（MYData）扩展名

索引文件具有".MYI"（MYIndex）扩展名

63、Mysql 中有哪些不同的表格?

共有 5 种类型的表格：

MyISAM

Heap

Merge

INNODB

ISAM

MyISAM 是 Mysql 的默认存储引擎。

64、ISAM 是什么?

ISAM 简称为索引顺序访问方法。它是由 IBM 开发的，用于在磁带等辅助存储系统上存储和检索数据。

65、InnoDB 是什么?

InnoDB 是一个由 Oracle 公司开发的 Innobase Oy 事务安全存储引擎。

66、Mysql 如何优化 DISTINCT?

DISTINCT 在所有列上转换为 GROUP BY，并与 ORDER BY 子句结合使用。

```
SELECT DISTINCT t1.a FROM t1,t2 where t1.a=t2.a;
```

67、如何输入字符为十六进制数字?

如果想输入字符为十六进制数字，可以输入带有单引号的十六进制数字和前缀（X），或者只用（Ox）前缀输入十六进制数字。 如果表达式上下文是字符串，则十六进制数字串将自动转换为字符串。

68、如何显示前 50 行?

在 Mysql 中，使用以下代码查询显示前 50 行：

```
SELECT * FROM LIMIT 0,50;
```

69、可以使用多少列创建索引?

任何标准表最多可以创建 16 个索引列。

70、NOW（）和 CURRENT_DATE（）有什么区别?

NOW（）命令用于显示当前年份，月份，日期，小时，分钟和秒。

CURRENT_DATE（）仅显示当前年份，月份和日期。

71、什么样的对象可以使用 CREATE 语句创建?

以下对象是使用 CREATE 语句创建的：

DATABASE

EVENT

FUNCTION

INDEX

PROCEDURE

TABLE

TRIGGER

USER

VIEW

72、Mysql 表中允许有多少个 TRIGGERS?

在 Mysql 表中允许有六个触发器，如下：

BEFORE INSERT

AFTER INSERT

BEFORE UPDATE

AFTER UPDATE

BEFORE DELETE

AFTER DELETE

73、什么是非标准字符串类型?

以下是非标准字符串类型：

TINYTEXT

TEXT

MEDIUMTEXT

LONGTEXT

74、什么是通用 SQL 函数?

CONCAT(A, B) - 连接两个字符串值以创建单个字符串输出。通常用于将两个或多个字段合并为一个字段。

FORMAT(X, D)- 格式化数字 X 到 D 有效数字。

CURRDATE(), CURRTIME()- 返回当前日期或时间。

NOW（） - 将当前日期和时间作为一个值返回。

MONTH（），DAY（），YEAR（），WEEK（），WEEKDAY（） - 从日期值中提取给定数据。 HOUR（），MINUTE（），SECOND（） - 从时间值中提取给定数据。 DATEDIFF（A，B） - 确定两个日期之间的差异，通常用于计算年龄 SUBTIMES（A，B） - 确定两次之间的差异。

FROMDAYS（INT） - 将整数天数转换为日期值。

75、解释访问控制列表

ACL（访问控制列表）是与对象关联的权限列表。这个列表是 Mysql 服务器安全模型的基础，它有助于排除用户无法连接的问题。

Mysql 将 ACL（也称为授权表）缓存在内存中。当用户尝试认证或运行命令时，Mysql 会按照预定的顺序检查 ACL 的认证信息和权限。

76、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。

示例如下：

START TRANSACTION;

SELECT @A:=SUM(salary) FROM table1 WHERE type=1;

UPDATE table2 SET summmary=@A WHERE type=1;

COMMIT;

77、mysql 里记录货币用什么字段类型好？

NUMERIC 和 DECIMAL 类型被 Mysql 实现为同样的类型，这在 SQL92 标准允许。他们被用于保存值，该值的准确精度是极其重要的值，例如与金钱有关的数据。当声明一个类是这些类型之一时，精度和规模的能被(并且通常是)指定；点击这里有一套最全阿里面试题总结。

例如：

salary DECIMAL(9,2)

在这个例子中，9(precision)代表将被用于存储值的总的小数位数，而 2(scale)代表将被用于存储小数点后的位数。 因此，在这种情况下，能被存储在 salary 列中的值的范围是从-9999999.99 到 9999999.99。在 ANSI/ISO SQL92 中，句法 DECIMAL(p)等价于 DECIMAL(p,0)。

同样，句法 DECIMAL 等价于 DECIMAL(p,0)，这里实现被允许决定值 p。Mysql 当前不支持 DECIMAL/NUMERIC 数据类型的这些变种形式的任一种。

这一般说来不是一个严重的问题，因为这些类型的主要益处得自于明显地控制精度和规模的能力。

DECIMAL 和 NUMERIC 值作为字符串存储，而不是作为二进制浮点数，以便保存那些值的小数精度。

一个字符用于值的每一位、小数点(如果 scale>0)和“-”符号(对于负值)。如果 scale 是 0，

DECIMAL 和 NUMERIC 值不包含小数点或小数部分。

DECIMAL 和 NUMERIC 值得最大的范围与 DOUBLE 一样，但是对于一个给定的 DECIMAL 或

NUMERIC 列，实际的范围可由制由给定列的 precision 或 scale 限制。

当这样的列赋给了小数点后面的位超过指定 scale 所允许的位的值，该值根据 scale 四舍五入。

当一个 DECIMAL 或 NUMERIC 列被赋给了其大小超过指定(或缺省的）precision 和 scale 隐含的范围的值，Mysql 存储表示那个范围的相应的端点值。

我希望本文可以帮助你提升技术水平。那些，感觉学的好难，甚至会令你沮丧的人，别担心，我认为，如果你愿意试一试本文介绍的几点，会向前迈进，克服这种感觉。这些要点也许对你不适用，但你会明确一个重要的道理：接受自己觉得受困这个事实是摆脱这个困境的第一步。

78、MYSQL 数据表在什么情况下容易损坏?

服务器突然断电导致数据文件损坏。

强制关机，没有先关闭 mysql 服务等。

79、mysql 有关权限的表都有哪几个?

Mysql 服务器通过权限表来控制用户对数据库的访问，权限表存放在 mysql 数据库里，由 mysql_install_db 脚本初始化。这些权限表分别 user，db，table_priv，columns_priv 和 host。

80、Mysql 中有哪几种锁?

MyISAM 支持表锁，InnoDB 支持表锁和行锁，默认为行锁

表级锁：开销小，加锁快，不会出现死锁。锁定粒度大，发生锁冲突的概率最高，并发量最低

行级锁：开销大，加锁慢，会出现死锁。锁力度小，发生锁冲突的概率小，并发度最高