数据库MySql笔试题：共计71道12910字

一、数据库知识（通用）篇

1.说说非关系型数据库和关系型数据库区别，优势比较?

非关系型数据库的优势：
性能：NOSQL是基于键值对的，可以想象成表中的主键和值的对应关系，而且不需要经过SQL层的解析，所以性能非常高。
可扩展性：同样也是因为基于键值对，数据之间没有耦合性，所以非常容易水平扩展。
关系型数据库的优势：
复杂查询：可以用SQL语句方便的在一个表以及多个表之间做非常复杂的数据查询。
事务支持：使得对于安全性能很高的数据访问要求得以实现。
其他：
1.对于这两类数据库，对方的优势就是自己的弱势，反之亦然。
2.NOSQL数据库慢慢开始具备SQL数据库的一些复杂查询功能，比如MongoDB。
3.对于事务的支持也可以用一些系统级的原子操作来实现例如乐观锁之类的方法来曲线救国，比如Redis set nx。

2.说说drop、truncate、 delete区别

最基本：
1) drop直接删掉表。
2) truncate删除表中数据，再插入时自增长id又从1开始。
3) 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，如果与事务无关，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。

3.什么叫视图？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，试图通常是有一个表或者多个表的行或列的子集。对视图的修改会影响基本表。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。

4.视图的优缺点有哪些?

优点：
1对数据库的访问，因为视图可以有选择性的选取数据库里的一部分。
2)用户通过简单的查询可以从复杂查询中得到结果。
3)维护数据的独立性，试图可从多个表检索数据。
4)对于相同的数据可产生不同的视图。
缺点：
性能：查询视图时，必须把视图的查询转化成对基本表的查询，如果这个视图是由一个复杂的多表查询所定义，那么，那么就无法更改数据

5.说说主键、外键、超键、候选键

超键：在关系中能唯一标识元组的属性集称为关系模式的超键。一个属性可以为作为一个超键，多个属性组合在一起也可以作为一个超键。超键包含候选键和主键。
候选键：是最小超键，即没有冗余元素的超键。
主键：数据库表中对储存数据对象予以唯一和完整标识的数据列或属性的组合。一个数据列只能有一个

主键，且主键的取值不能缺失，即不能为空值（Null）。
外键：在一个表中存在的另一个表的主键称此表的外键。

6.为什么用自增列作为主键?

如果我们定义了主键(PRIMARY KEY)，那么InnoDB会选择主键作为聚集索引、
如果没有显式定义主键，则InnoDB会选择第一个不包含有NULL值的唯一索引作为主键索引、
如果也没有这样的唯一索引，则InnoDB会选择内置6字节长的ROWID作为隐含的聚集索引(ROWID随着行记录的写入而主键递增，这个ROWID不像ORACLE的ROWID那样可引用，是隐含的)。
数据记录本身被存于主索引（一颗B+Tree）的叶子节点上。这就要求同一个叶子节点内（大小为一个内存页或磁盘页）的各条数据记录按主键顺序存放，因此每当有一条新的记录插入时，MySQL会根据其主键将其插入适当的节点和位置，如果页面达到装载因子（InnoDB默认为15/16），则开辟一个新的页（节点）
如果表使用自增主键，那么每次插入新的记录，记录就会顺序添加到当前索引节点的后续位置，当一页写满，就会自动开辟一个新的页
如果使用非自增主键（如果身份证号或学号等），由于每次插入主键的值近似于随机，因此每次新记录都要被插到现有索引页的中间某个位置，此时MySQL不得不为了将新记录插到合适位置而移动数据，甚至目标页面可能已经被回写到磁盘上而从缓存中清掉，此时又要从磁盘上读回来，这增加了很多开销，同时频繁的移动、分页操作造成了大量的碎片，得到了不够紧凑的索引结构，后续不得不通过OPTIMIZE TABLE来重建表并优化填充页面。

7.varchar和char的使用场景?

1.char的长度是不可变的，而varchar的长度是可变的。
定义一个char[10]和varchar[10]。
如果存进去的是‘csdn’,那么char所占的长度依然为10，除了字符‘csdn’外，后面跟六个空格，varchar就立马把长度变为4了，取数据的时候，char类型的要用trim()去掉多余的空格，而varchar是不需要的。
2.char的存取速度还是要比varchar要快得多，因为其长度固定，方便程序的存储与查找。
char也为此付出的是空间的代价，因为其长度固定，所以难免会有多余的空格占位符占据空间，可谓是以空间换取时间效率。
varchar是以空间效率为首位。
3.char的存储方式是：对英文字符（ASCII）占用1个字节，对一个汉字占用两个字节。
varchar的存储方式是：对每个英文字符占用2个字节，汉字也占用2个字节。
4.两者的存储数据都非unicode的字符数据。

8.触发器的作用是什么?

触发器是一种特殊的存储过程，主要是通过事件来触发而被执行的。它可以强化约束，来维护数据的完整性和一致性，可以跟踪数据库内的操作从而不允许未经许可的更新和变化。可以联级运算。如，某表上的触发器上包含对另一个表的数据操作，而该操作又会导致该表触发器被触发。

9.什么是存储过程？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。
调用：
1）可以用一个命令对象来调用存储过程。
2）可以供外部程序调用，比如：java程序。

10.说说存储过程的优缺点?

优点:
1）存储过程是预编译过的，执行效率高。
2）存储过程的代码直接存放于数据库中，通过存储过程名直接调用，减少网络通讯。
3）安全性高，执行存储过程需要有一定权限的用户。
4）存储过程可以重复使用，可减少数据库开发人员的工作量。
缺点:
移植性差

11.说说存储过程与函数的区别

（1）存储过程用户在数据库中完成特定操作或者任务（如插入，删除等），函数用于返回特定的数据。
（2）存储过程声明用procedure，函数用function。
（3）存储过程不需要返回类型，函数必须要返回类型。
（4）存储过程可作为独立的pl-sql执行，函数不能作为独立的plsql执行，必须作为表达式的一部分。
（5）存储过程只能通过out和in/out来返回值，函数除了可以使用out，in/out以外，还可以使用return返回值。
（6）sql语句（DML或SELECT)中不可用调用存储过程，而函数可以。

12.什么是临时表，临时表什么时候删除?

临时表可以手动删除:
DROP TEMPORARY TABLE IF EXISTS temp_tb;
临时表只在当前连接可见，当关闭连接时，MySQL会自动删除表并释放所有空间。因此在不同的连接中可以创建同名的临时表，并且操作属于本连接的临时表。
创建临时表的语法与创建表语法类似，不同之处是增加关键字TEMPORARY，
如:

```
CREATE TEMPORARY TABLE tmp_table (
        NAME VARCHAR (10) NOT NULL,
        time date NOT NULL
);
select * from tmp_table;
```

13.什么是数据库范式，根据某个场景设计数据表?

第一范式:(确保每列保持原子性)所有字段值都是不可分解的原子值。
第一范式是最基本的范式。如果数据库表中的所有字段值都是不可分解的原子值，就说明该数据库表满足了第一范式。
第一范式的合理遵循需要根据系统的实际需求来定。比如某些数据库系统中需要用到“地址”这个属性，本来直接将“地址”属性设计成一个数据库表的字段就行。但是如果系统经常会访问“地址”属性中的“城市”部分，那么就非要将“地址”这个属性重新拆分为省份、城市、详细地址等多个部分进行存储，这样在对地址中某一部分操作的时候将非常方便。这样设计才算满足了数据库的第一范式，如下表所示。
上表所示的用户信息遵循了第一范式的要求，这样在对用户使用城市进行分类的时候就非常方便，也提高了数据库的性能。

第二范式:(确保表中的每列都和主键相关)在一个数据库表中，一个表中只能保存一种数据，不可以把多种数据保存在同一张数据库表中。
第二范式在第一范式的基础之上更进一层。第二范式需要确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。也就是说在一个数据库表中，一个表中只能保存一种数据，不可以把多种数据保存在同一张数据库表中。
比如要设计一个订单信息表，因为订单中可能会有多种商品，所以要将订单编号和商品编号作为数据库表的联合主键。
第三范式:(确保每列都和主键列直接相关,而不是间接相关) 数据表中的每一列数据都和主键直接相关，而不能间接相关。
第三范式需要确保数据表中的每一列数据都和主键直接相关，而不能间接相关。
比如在设计一个订单数据表的时候，可以将客户编号作为一个外键和订单表建立相应的关系。而不可以在订单表中添加关于客户其它信息（比如姓名、所属公司等）的字段。
BCNF:符合3NF，并且，主属性不依赖于主属性。
若关系模式属于第二范式，且每个属性都不传递依赖于键码，则R属于BC范式。
通常BC范式的条件有多种等价的表述：每个非平凡依赖的左边必须包含键码；每个决定因素必须包含键码。
BC范式既检查非主属性，又检查主属性。当只检查非主属性时，就成了第三范式。满足BC范式的关系都必然满足第三范式。
还可以这么说：若一个关系达到了第三范式，并且它只有一个候选码，或者它的每个候选码都是单属性，则该关系自然达到BC范式。
一般，一个数据库设计符合3NF或BCNF就可以了。
第四范式:要求把同一表内的多对多关系删除。
第五范式:从最终结构重新建立原始结构。

14.什么是 内连接、外连接、交叉连接、笛卡尔积等?

内连接: 只连接匹配的行
左外连接: 包含左边表的全部行（不管右边的表中是否存在与它们匹配的行），以及右边表中全部匹配的行
右外连接: 包含右边表的全部行（不管左边的表中是否存在与它们匹配的行），以及左边表中全部匹配的行
例如1：

```
SELECT a.,b. FROM luntan LEFT JOIN usertable as b ON a.username=b.username
```

例如2：

```
SELECT a.,b. FROM city as a FULL OUTER JOIN user as b ON a.username=b.username
```

全外连接: 包含左、右两个表的全部行，不管另外一边的表中是否存在与它们匹配的行。
交叉连接: 生成笛卡尔积 - 它不使用任何匹配或者选取条件，而是直接将一个数据源中的每个行与另一个数据源的每个行都一一匹配
例如：

```
SELECT type,pub_name FROM titles CROSS JOIN publishers ORDER BY type
```

15.SQL语言分类

SQL语言共分为四大类：
一、数据查询语言DQL
二、数据操纵语言DML
三、数据定义语言DDL
四、数据控制语言DCL。

1. 数据查询语言DQL
   数据查询语言DQL基本结构是由SELECT子句，FROM子句，WHERE子句组成的查询块：
   SELECT
   FROM
   WHERE
2. 数据操纵语言DML
   数据操纵语言DML主要有三种形式：
   1) 插入：INSERT
   2) 更新：UPDATE
   3) 删除：DELETE
3. 数据定义语言DDL
   数据定义语言DDL用来创建数据库中的各种对象-----表、视图、索引、同义词、聚簇等如：
   CREATE TABLE/VIEW/INDEX/SYN/CLUSTER
   表 视图 索引 同义词 簇
   DDL操作是隐性提交的！不能rollback
4. 数据控制语言DCL
   数据控制语言DCL用来授予或回收访问数据库的某种特权，并控制数据库操纵事务发生的时间及效果，对数据库实行监视等。如：
   1) GRANT：授权。
   2) ROLLBACK [WORK] TO [SAVEPOINT]：回退到某一点。回滚---ROLLBACK；回滚命令使数据库状态回到上次最后提交的状态。其格式为：
   SQL>ROLLBACK;
   3) COMMIT [WORK]：提交。
   在数据库的插入、删除和修改操作时，只有当事务在提交到数据库时才算完成。在事务提交前，只有操作数据库的这个人才能有权看到所做的事情，别人只有在最后提交完成后才可以看到。
   提交数据有三种类型：显式提交、隐式提交及自动提交。下面分别说明这三种类型。
   (1) 显式提交
   用COMMIT命令直接完成的提交为显式提交。其格式为：SQL>COMMIT；
   (2) 隐式提交
   用SQL命令间接完成的提交为隐式提交。这些命令是：
   ALTER，AUDIT，COMMENT，CONNECT，CREATE，DISCONNECT，DROP，
   EXIT，GRANT，NOAUDIT，QUIT，REVOKE，RENAME。
   (3) 自动提交
   若把AUTOCOMMIT设置为ON，则在插入、修改、删除语句执行后，
   系统将自动进行提交，这就是自动提交。
   其格式为：SQL>SET AUTOCOMMIT ON；

16.说说like %和-的区别

通配符的分类
%百分号通配符:表示任何字符出现任意次数(可以是0次).
下划线通配符:表示只能匹配单个字符,不能多也不能少,就是一个字符.
like操作符: LIKE作用是指示mysql后面的搜索模式是利用通配符而不是直接相等匹配进行比较.

*注意: 如果在使用like操作符时,后面的没有使用通用匹配符效果是和= 一致的,SELECT * FROM products WHERE products.prod_name like '1000';只能匹配的结果为1000,而不能匹配像etPack 1000这样的结果.*
%通配符使用: 匹配以"yves"开头的记录:(包括记录"yves") SELECT FROM products WHERE products.prod_name like 'yves%';
匹配包含"yves"的记录(包括记录"yves") SELECT FROM products WHERE products.prod_name like '%yves%';
*匹配以"yves"结尾的记录(包括记录"yves",不包括记录"yves ",也就是yves后面有空格的记录,这里需要注意) SELECT * FROM products WHERE products.prod_name like '%yves';*
通配符使用: SELECT FROM products WHERE products.prod_name like 'yves'; 匹配结果为: 像"yyves"这样记录. SELECT FROM products WHERE products.prodname like 'yves'; 匹配结果为: 像"yvesHe"这样的记录.(一个下划线只能匹配一个字符,不能多也不能少)
注意事项:
注意大小写,在使用模糊匹配时,也就是匹配文本时,mysql是可能区分大小的,也可能是不区分大小写的,这个结果是取决于用户对MySQL的配置方式.如果是区分大小写,那么像YvesHe这样记录是不能被"yves__"这样的匹配条件匹配的.
注意尾部空格,"%yves"是不能匹配"heyves "这样的记录的.
注意NULL,%通配符可以匹配任意字符,但是不能匹配NULL,也就是说SELECT * FROM products WHERE products.prod_name like '%;是匹配不到products.prod_name为NULL的的记录.
技巧与建议:
正如所见， MySQL的通配符很有用。但这种功能是有代价的：通配符搜索的处理一般要比前面讨论的其他搜索所花时间更长。这里给出一些使用通配符要记住的技巧。
不要过度使用通配符。如果其他操作符能达到相同的目的，应该 使用其他操作符。
在确实需要使用通配符时，除非绝对有必要，否则不要把它们用 在搜索模式的开始处。把通配符置于搜索模式的开始处，搜索起 来是最慢的。
仔细注意通配符的位置。如果放错地方，可能不会返回想要的数.

17.说说count(*)、count(1)、count(column)的区别

count()*对行的数目进行计算,包含NULL*
count(column)对特定的列的值具有的行数进行计算,不包含NULL值。
*count()还有一种使用方式,count(1)这个用法和count()*的结果是一样的。
性能问题:
1.任何情况下SELECT COUNT() *FROM tablename是最优选择,*
2.尽量减少SELECT COUNT() FROM tablename WHERE COL = ‘value’ 这种查询;
3.杜绝SELECT COUNT(COL) FROM tablename WHERE COL2 = ‘value’ 的出现。
如果表没有主键,那么count(1)比count()*快。*
*如果有主键,那么count(主键,联合主键)比count()*快。
如果表只有一个字段,count()*最快。*
*count(1)跟count(主键)一样,只扫描主键。count()*跟count(非主键)一样,扫描整个表。明显前者更快一些。

18.什么是最左前缀原则?

多列索引:

```
ALTER TABLE people ADD INDEX lname_fname_age (lame,fname,age);
```

为了提高搜索效率，我们需要考虑运用多列索引,由于索引文件以B - Tree格式保存，所以我们不用扫描任何记录，即可得到最终结果。
注：在mysql中执行查询时，只能使用一个索引，如果我们在lname,fname,age上分别建索引,执行查询时，只能使用一个索引，mysql会选择一个最严格(获得结果集记录数最少)的索引。
最左前缀原则：顾名思义，就是最左优先，上例中我们创建了lname_fname_age多列索引,相当于创建了(lname)单列索引，(lname,fname)组合索引以及(lname,fname,age)组合索引。

19.什么是索引?

何为索引:
数据库索引，是数据库管理系统中一个排序的数据结构，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。

20.索引的作用？它的优点缺点是什么?

索引作用:
协助快速查询、更新数据库表中数据。
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)。

21.说说聚集索引和非聚集索引区别?

聚合索引(clustered index):
聚集索引表记录的排列顺序和索引的排列顺序一致，所以查询效率快，只要找到第一个索引值记录，其余就连续性的记录在物理也一样连续存放。聚集索引对应的缺点就是修改慢，因为为了保证表中记录的物理和索引顺序一致，在记录插入的时候，会对数据页重新排序。
聚集索引类似于新华字典中用拼音去查找汉字，拼音检索表于书记顺序都是按照a~z排列的，就像相同的逻辑顺序于物理顺序一样，当你需要查找a,ai两个读音的字，或是想一次寻找多个傻(sha)的同音字时，也许向后翻几页，或紧接着下一行就得到结果了。
非聚合索引(nonclustered index):
非聚集索引指定了表中记录的逻辑顺序，但是记录的物理和索引不一定一致，两种索引都采用B+树结构，非聚集索引的叶子层并不和实际数据页相重叠，而采用叶子层包含一个指向表中的记录在数据页中的指针方式。非聚集索引层次多，不会造成数据重排。
非聚集索引类似在新华字典上通过偏旁部首来查询汉字，检索表也许是按照横、竖、撇来排列的，但是由于正文中是a~z的拼音顺序，所以就类似于逻辑地址于物理地址的不对应。同时适用的情况就在于分组，大数目的不同值，频繁更新的列中，这些情况即不适合聚集索引。
根本区别:
聚集索引和非聚集索引的根本区别是表记录的排列顺序和与索引的排列顺序是否一致。

22.索引的优缺点有哪些?

创建索引可以大大提高系统的性能（优点）：
（1）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。
（2）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。
（3）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。
（4）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。
（5）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。
增加索引也有许多不利的方面(缺点):
（1）.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。
（2）.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。

（3）.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。
（4）.哪些列适合建立索引、哪些不适合建索引?
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：
（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。

对于有些列不应该创建索引：
（1）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。
（2）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。
（3）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。
(4)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。
索引详解：带你从头到尾捋一遍MySQL索引结构！

23.什么样的字段适合建索引?

唯一、不为空、经常被查询的字段

24.说说MySQL B+Tree索引和Hash索引的区别?

Hash索引和B+树索引的特点：
Hash索引结构的特殊性，其检索效率非常高，索引的检索可以一次定位;
B+树索引需要从根节点到枝节点，最后才能访问到页节点这样多次的IO访问;
为什么不都用Hash索引而使用B+树索引?
Hash索引仅仅能满足"=","IN"和""查询，不能使用范围查询,因为经过相应的Hash算法处理之后的Hash值的大小关系，并不能保证和Hash运算前完全一样；
Hash索引无法被用来避免数据的排序操作，因为Hash值的大小关系并不一定和Hash运算前的键值完全一样；
Hash索引不能利用部分索引键查询，对于组合索引，Hash索引在计算Hash值的时候是组合索引键合并后再一起计算Hash值，而不是单独计算Hash值，所以通过组合索引的前面一个或几个索引键进行查询的时候，Hash索引也无法被利用；
Hash索引在任何时候都不能避免表扫描，由于不同索引键存在相同Hash值，所以即使取满足某个Hash键值的数据的记录条数，也无法从Hash索引中直接完成查询，还是要回表查询数据；
Hash索引遇到大量Hash值相等的情况后性能并不一定就会比B+树索引高。
补充：

（1）.MySQL中，只有HEAP/MEMORY引擎才显示支持Hash索引。
（2）.常用的InnoDB引擎中默认使用的是B+树索引，它会实时监控表上索引的使用情况，如果认为建立哈希索引可以提高查询效率，则自动在内存中的“自适应哈希索引缓冲区”建立哈希索引（在InnoDB中默认开启自适应哈希索引），通过观察搜索模式，MySQL会利用index key的前缀建立哈希索引，如果一个表几乎大部分都在缓冲池中，那么建立一个哈希索引能够加快等值查询。
B+树索引和哈希索引的明显区别是：
（3）.如果是等值查询，那么哈希索引明显有绝对优势，因为只需要经过一次算法即可找到相应的键值；当然了，这个前提是，键值都是唯一的。如果键值不是唯一的，就需要先找到该键所在位置，然后再根据链表往后扫描，直到找到相应的数据；
（4）.如果是范围查询检索，这时候哈希索引就毫无用武之地了，因为原先是有序的键值，经过哈希算法后，有可能变成不连续的了，就没办法再利用索引完成范围查询检索；
同理，哈希索引没办法利用索引完成排序，以及like ‘xxx%’ 这样的部分模糊查询（这种部分模糊查询，其实本质上也是范围查询）；
（5）.哈希索引也不支持多列联合索引的最左匹配规则；
（6）.B+树索引的关键字检索效率比较平均，不像B树那样波动幅度大，在有大量重复键值情况下，哈希索引的效率也是极低的，因为存在所谓的哈希碰撞问题。
（7）在大多数场景下，都会有范围查询、排序、分组等查询特征，用B+树索引就可以了。

25.说说B树和B+树的区别

B树，每个节点都存储key和data，所有节点组成这棵树，并且叶子节点指针为nul，叶子结点不包含任何关键字信息。
B+树，所有的叶子结点中包含了全部关键字的信息，及指向含有这些关键字记录的指针，且叶子结点本身依关键字的大小自小而大的顺序链接，所有的非终端结点可以看成是索引部分，结点中仅含有其子树根结点中最大（或最小）关键字。(而B 树的非终节点也包含需要查找的有效信息)

26.为什么说B+比B树更适合实际应用中操作系统的文件索引和数据库索引?

1.B+的磁盘读写代价更低
B+的内部结点并没有指向关键字具体信息的指针。因此其内部结点相对B树更小。如果把所有同一内部结点的关键字存放在同一盘块中，那么盘块所能容纳的关键字数量也越多。一次性读入内存中的需要查找的关键字也就越多。相对来说IO读写次数也就降低了。
2.B+tree的查询效率更加稳定
由于非终结点并不是最终指向文件内容的结点，而只是叶子结点中关键字的索引。所以任何关键字的查找必须走一条从根结点到叶子结点的路。所有关键字查询的路径长度相同，导致每一个数据的查询效率相当。

二、数据库MySql基础篇

1.函数的分类？经常使用的函数有哪些?

lower
upper
substr
length
trim(去首尾空格，不会去除中间的空格)
str_to_date(%Y-%m-%d)

date_format
format(保留小数)
round
rand()随机数
ifnull（如果为空，则替换为0）
聚合函数/分组函数
分组函数自动忽略空值
count
sum
avg
min
max

2.分组查询需要注意条件?

如果使用了order by，order by 必须放到group by后面。
在sql语句中，select语句后面只能跟分组函数+参与分组的字段。
如果想要对分组数据再进行过滤需要使用having子句。

3.limit使用方法?

```
select * from emp limit m,n;
```

4.mysql常见数据类型?

char:定长字符串，适合做主键或者外键
varchar：可变长字符串
double/float
int/bigint
date

5.如何增加删除修改表结构

```
 alter table 表名 add 字段名 数据类型(长度) --添加字段
 alter table 表名 modify 字段名 数据类型(长度) --修改字段长度
 alter table 表名 change 原字段名 现在字段名 数据类型(长度) --修改字段名称
 alter table 表名 drop 字段名 --删除字段
```

6.如何开启MySQL服务,关闭My服务

开启服务：

```
service mysqld start

/init.d/mysqld start

safe_mysql &
```

关闭服务：

```
service mysqld stop

/etc/init.d/mysqld stop

mysqladmin -uroot -p123456 shutdown
```

7.检测端口是否运行

```
lsof -i:3306

netstat -tunlp|grep 3306

ss -tulnp|grep 3306
```

8.如何为MySQL设置密码或者修改密码。

```
方法一

mysqladmin -u root -p123456 password 'abc123'    #比较常用

方法二（sql语句修改）

update mysql.user set password=password(123456) where user='root' and
host='localhost';

flush privileges;

方法三（sql语句修改）

set password=password('abc123');
```

9.如何登陆MySQL数据库。

```
单实例登陆

mysql -uroot -p123456

多实例登陆

mysql -uroot -p123456 -S /data/3306/mysql.sock
```

10.查看当前数据库的字符集

```
mysql> show variables like "%charac%";
```

11.如何查看当前数据库版本

```
# mysql -V
mysql> select version();
```

12.如何 查看当前登录的用户。

```
mysql> select user();
```

13.查看T1数据库中有哪儿些表

```
mysql> use T1;

mysql> show tables;
```

14.创建GBK字符集的数据库oldboy，并查看已建库完整语句

```
mysql> create database oldboy default character set gbk;
mysql> show create database oldboy;
```

15.创建用户oldboy，使之可以管理数据库oldboy

```
mysql> grant select,update,insert,delete,alter on oldboy.* to oldboy@'localhost'
identified by '123456';
```

16.查看创建的用户oldboy拥有哪些权限

```
mysql> show grants for oldboy@'localhost';
```

17.查看当前数据库里有哪些用户

```
mysql> select user,host from mysql.user;
```

18.如何进入oldboy数据库

```
mysql> use oldboy();
```

19.请写一个脚本： 创建一个innodb GBK表test，字段id int(4)和name varchar(16)

```
mysql> create table test (id int(4),name varchar(16)) engine=InnoDB default
charset=gbk;
```

20.查看建表结构及表结构的SQL语句

```
mysql> desc test;
mysql> show create table test\G
```

21.请使用脚本查询一条数据：插入一条数据“1,oldboy”

```
mysql> insert into test (id,name) values (1,'oldboy');
```

22.再批量插入2行数据 “2,老男孩”，“3,oldboyedu”

```
mysql> insert into test (id,name) values (2,'老男孩'),(3,'oldboyedu');
```

23.查询名字为oldboy的记录

```
mysql> select * from test where name='oldboy';
```

24.把数据id等于1的名字oldboy更改为oldgirl

```
mysql> update test set name='oldgirl' where id=1;
```

25.在字段name前插入age字段，类型tinyint(2)

```
mysql> alter table test add age tinyint(2) after id;
```

26.不退出数据库,完成备份oldboy数据库

```
mysql> system mysqldump -uroot -p123456 -B -x -F --events oldboy >/opt/bak.sql
```

27.删除test表中的所有数据，并查看

```
mysql> delete from test;
```

三、数据库MySql篇

1.说一下 MySQL 的行锁和表锁?

MyISAM 只支持表锁，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。

2.MySQL的默认事务隔离级别是?

读未提交(RU): 一个事务还没提交时, 它做的变更就能被别的事务看到.
读提交(RC): 一个事务提交之后, 它做的变更才会被其他事务看到.
可重复读(RR): 一个事务执行过程中看到的数据, 总是跟这个事务在启动时看到的数据是一致的. 当然在可重复读隔离级别下, 未提交变更对其他事务也是不可见的.
串行化(S): 对于同一行记录, 读写都会加锁. 当出现读写锁冲突的时候, 后访问的事务必须等前一个事务执行完成才能继续执行.

3.Mysql数据库表类型有哪些?

MyISAM、InnoDB、HEAP、BOB,ARCHIVE,CSV等。
MyISAM：成熟、稳定、易于管理，快速读取。一些功能不支持（事务等），表级锁。
InnoDB：支持事务、外键等特性、数据行锁定。空间占用大，不支持全文索引等。

4.MySQL怎么恢复半个月前的数据?

通过整库备份+binlog进行恢复. 前提是要有定期整库备份且保存了binlog日志.

5.一张自增表里面总共有 7 条数据，删除了最后 2 条数据，重启 MySQL 数据库，又插入了一条数据，此时 id 是几?

表类型如果是 MyISAM ，那 id 就是 8。
表类型如果是 InnoDB，那 id 就是 6。
InnoDB 表只会把自增主键的最大 id 记录在内存中，所以重启之后会导致最大 id 丢失。

6.MySQL 的内连接、左连接、右连接有什么区别?

内连接关键字：inner join；左连接：left join；右连接：right join。 内连接是把匹配的关联数据显示出来；左连接是左边的表全部显示出来，右边的表显示出符合条件的数据；右连接正好相反。

7.MySQL 问题排查都有哪些手段?

使用 show processlist 命令查看当前所有连接信息。 使用 explain 命令查询 SQL 语句执行计划。 开启慢查询日志，查看慢查询的 SQL。

8.如何做 MySQL 的性能优化?

为搜索字段创建索引。
避免使用 select *，列出需要查询的字段。
垂直分割分表。
选择正确的存储引擎。

读写分离

9.MySQL数据库作发布系统的存储，一天五万条以上的增量，预计运维三年,怎么优化?

（1）设计良好的数据库结构，允许部分数据冗余，尽量避免join查询，提高效率。
（2） 选择合适的表字段数据类型和存储引擎，适当的添加索引。
（3） 做mysql主从复制读写分离。
（4）对数据表进行分表，减少单表中的数据量提高查询速度。
（5）添加缓存机制，比如redis，memcached等。
（6）对不经常改动的页面，生成静态页面（比如做ob缓存）。
（7）书写高效率的SQL。比如 SELECT * FROM TABEL 改为 SELECT field_1, field_2, field_3 FROM TABLE.

10.MySQL由哪些部分组成, 分别用来做什么?

（1）Server
（2）连接器: 管理连接, 权限验证.
（3）分析器: 词法分析, 语法分析.
（4）优化器: 执行计划生成, 索引的选择.

（5）执行器: 操作存储引擎, 返回执行结果.
（6）存储引擎: 存储数据, 提供读写接口.

11.怎么验证 MySQL 的索引是否满足需求?

使用 explain 查看 SQL 是如何执行查询语句的，从而分析你的索引是否满足需求。 explain 语法：explain select * from table where type=1。

12.请你介绍一下 mysql的主从复制?

考察点：数据库

MySQL主从复制是其最重要的功能之一。主从复制是指一台服务器充当主数据库服务器，另一台或多台服务器充当从数据库服务器，主服务器中的数据自动复制到从服务器之中。对于多级复制，数据库服务器即可充当主机，也可充当从机。MySQL主从复制的基础是主服务器对数据库修改记录二进制日志，从服务器通过主服务器的二进制日志自动执行更新。

MySQL主从复制的两种情况：同步复制和异步复制，实际复制架构中大部分为异步复制。

复制的基本过程如下：

Slave上面的IO进程连接上Master，并请求从指定日志文件的指定位置（或者从最开始的日志）之后的日志内容。

Master接收到来自Slave的IO进程的请求后，负责复制的IO进程会根据请求信息读取日志指定位置之后的日志信息，返回给Slave的IO进程。返回信息中除了日志所包含的信息之外，还包括本次返回的信息已经到Master端的bin-log文件的名称以及bin-log的位置。

Slave的IO进程接收到信息后，将接收到的日志内容依次添加到Slave端的relay-log文件的最末端，并将读取到的Master端的bin-log的文件名和位置记录到master-info文件中，以便在下一次读取的时候能够清楚的告诉Master“我需要从某个bin-log的哪个位置开始往后的日志内容，请发给我”。

Slave的Sql进程检测到relay-log中新增加了内容后，会马上解析relay-log的内容成为在Master端真实执行时候的那些可执行的内容，并在自身执行。

13.请你介绍一下mysql的MVCC机制

MVCC是一种多版本并发控制机制，是MySQL的InnoDB存储引擎实现隔离级别的一种具体方式，用于实现提交读和可重复读这两种隔离级别。MVCC是通过保存数据在某个时间点的快照来实现该机制，其在每行记录后面保存两个隐藏的列，分别保存这个行的创建版本号和删除版本号，然后Innodb的MVCC使用到的快照存储在Undo日志中，该日志通过回滚指针把一个数据行所有快照连接起来。

14.常用的Mysql复制架构有哪些?

（1）一主多从 在主库读取请求压力非常大的场景下, 可以通过配置一主多从复制架构实现读写分离, 把大量对实时性要求不是特别高的读请求通过负载均衡分布到多个从库上, 降低主库的读取压力，在主库出现异常宕机的情况下, 可以把一个从库切换为主库继续提供服务 。

（2）多级复制 一主多从的架构能够解决大部分读请求压力特别大的场景的需求, 考虑到 MysQL的复制是主库“推送” Binlog日志到从库,主库的 I/0压力和网络压力会随着从库的增加而增长(每个从库都会在主库上有一个独立的 Binlog Dump线程来发送事件), 而多级复制架构解决了一主多从场景下,主库额外的I/0和网络压力。

（3）双主复制/Dual Master 其实就是主库 Master和 Master2互为主从， client客户端的写请求都访问主库 Master,而读请求可以选择访问主库 Master或 Master2。

15.Mysql 的存储引擎,myisam和innodb的区别?

（1）InnoDB支持事务, MyISAM不支持.

（2）InnoDB支持行级锁, MyISAM支持表级锁.

（3）InnoDB支持多版本并发控制(MVVC), MyISAM不支持.

（4）InnoDB支持外键, MyISAM不支持.

（5）MyISAM支持全文索引, InnoDB不支持(但可以使用Sphinx插件)

16.请问MySQL的端口号是多少，如何修改这个端口号

查看端口号：

使用命令show global variables like 'port';查看端口号，mysql的默认端口是3306。（补充：sqlserver默认端口号为：1433；oracle默认端口号为：1521；DB2默认端口号为：5000；PostgreSQL默认端口号为：5432）

修改端口号：

修改端口号：编辑/etc/my.cnf文件，早期版本有可能是my.conf文件名，增加端口参数，并且设定端口，注意该端口未被使用，保存退出。

17.Mysql如何为表字段添加索引?

（1）添加PRIMARY KEY（主键索引）

```
ALTER TABLE `table_name` ADD PRIMARY KEY ( `column` )
```

（2）添加UNIQUE(唯一索引)

```
ALTER TABLE `table_name` ADD UNIQUE ( `column` )
```

（3）添加INDEX(普通索引)

```
ALTER TABLE `table_name` ADD INDEX index_name ( `column` )
```

（4）添加FULLTEXT(全文索引)

```
ALTER TABLE `table_name` ADD FULLTEXT ( `column`)
```

（5）添加多列索引

```
ALTER TABLE `table_name` ADD INDEX index_name ( `column1`, `column2`, `column3`
)
```

18.说说自己对于 MySQL 常见的两种存储引擎：MyISAM与InnoDB的理解?

InnoDB 引擎：InnoDB 引擎提供了对数据库 acid 事务的支持，并且还提供了行级锁和外键的约束，它的设计的目标就是处理大数据容量的数据库系统。MySQL 运行的时候，InnoDB 会在内存中建立缓冲池，用于缓冲数据和索引。但是该引擎是不支持全文搜索，同时启动也比较的慢，它是不会保存表的行数的，所以当进行 select count() from table 指令的时候，需要进行扫描全表。由于锁的粒度小，写操作是不会锁定全表的,所以在并发度较高的场景下使用会提升效率的。

MyIASM 引擎：MySQL 的默认引擎，但不提供事务的支持，也不支持行级锁和外键。因此当执行插入和更新语句时，即执行写操作的时候需要锁定这个表，所以会导致效率会降低。不过和 InnoDB 不同的是，MyIASM 引擎是保存了表的行数，于是当进行 select count() from table 语句时，可以直接的读取已经保存的值而不需要进行扫描全表。所以，如果表的读操作远远多于写操作时，并且不需要事务的支持的，可以将 MyIASM 作为数据库引擎的首选。