MySql - 基础

基础问题

1.列举常见的关系型数据库和非关系型都有那些?

```
关系型数据库(需要有表结构)
    mysql、oracle 、 spl、server、db2、sybase
非关系型数据库(是以key - value存储的,没有表结构)
    mongod : mongod 是一个高性能,开源,无模式的文档型数据库,开发语言是c++,它在许多场景下替代传统的关系型数据库或键/值的存储方式
    redis:是一个开源的,使用ANSI C语言编写,支持网络,可基于内存亦可持久化的日志型,并提供多种语言的API。目前由VMware主持开发工作。
```

2.MySQL常见数据库引擎及比较?

```
MySql常见的引擎是
    myisam:不支持事务,支持表锁
    innoDB:支持事务,支持表锁和行锁
```

3. 简述事务及其特性,mysql 如何实现事务

```
事务简介:
    事务是应用程序中一系列严密的操作,所有操作必须成功完成,否则在每个操作中所有的操作都会被撤销
事务特性 (ACID):
    原子性(Atomicity): 表示组成一个事务的多个数据库操作是一个不可分割的原子单位,只有所有的操作执行成功,整个事务才提交,事务中任何一个数据库操作失败,已经执行的任何操作都必须撤销,让数据 库返回到初始状态
    一致性(Consistency):事务操作成功后，数据库所处的状态和它的业务规则是一致的，即数据不会被破坏。
    隔离性(Isolation):在并发数据操作时，不同的事务拥有各自数据空间，它们的操作不会对对方产生干扰。数据库规定了多种事务隔 离级别，不同隔离级别对应不同的干扰程度，隔离级别越高，数据一 致性越好，但并发性越弱
    持续性(Durability):一旦事务提交成功后，事务中所有的数据操作都必须被持久化到数据库中，即使提交事务后，数据库马上崩溃，在数据库重启时，也必须能保证能够通过某种机制恢复数据
```

```
Mysql实现事务:(使用innoDB)
    开启事务 start transactions
    关闭事务 commit
```

4. 简述触发器,函数,视图,存储过程

1.触发器:
　　对某个表进行(增/删/改)操作的前后触发一些操作即为触发器,(触发器用于自定义用户对表的行进行(增/删/改)前后的行为)
　　触发器必须定义在特定的表上
2.函数:
　　内置函数
　　自定义函数
 3.视图
　　视图是查询命令结果构成的一个虚拟表（非真实存在），其本质是【根据SQL语句获取动态的数据集，并为其命名】，用户使用时只需使用【名称】即可获取结果集合，并可以当作表来查询使用。
 4.存储过程:存储过程(procedure)，概念类似于函数，就是把一段代码封装起来，当要执行这一段代码的时候，可以通过调用该存储过程来实现。在封装的语句体里面，可以同if/else ,case,while等控制结构，可以进行sql编程，查看现有的存储过程。

5. mysql常见的函数

```
ABS()        BIN()
CEILING向上取整
FLOOR()
EXP(x)    e的x次方
RAND()    0-1的随机数
PI()    圆周率
LEAST(X1, X2 ...)    返回集合中最小的值
GREATEST(X1, X2)    返回集合中最大的值
LN()    返回x的自然对数
LOG(x, y)    返回x的以y为底的对数
MOD()    取模(余数)
round()    四舍五入
SIGN(x)    返回代表数字x的符合的值
SQRT(X)    返回一个数的平方根
TRUNCATE(X,Y)    返回数字x截短为y位小数的结果

聚合函数
AVG
COUNT
MIN
MAX
SUM
GROUP_CONCATE 返回集合中最小的值

日期和时间函数
curdate()    current_date()    当前日期
curtime()    current_time()    当前时间
```

6. 数据库导入命令(结构+数据)

```
数据库备份与恢复
mysqldump命令用于备份数据库数据

[root@master ~]# mysqldump -u root -p --all-databases > /tmp/db.dump
2.导出db1、db2两个数据库的所有数据
```

```
mysqldump -uroot -proot --databases db1 db2 >/tmp/user.sql
进入mariadb数据库，删除一个db

[root@master ~]# mysql -uroot -p

MariaDB [(none)]> drop database s11;
进行数据恢复，吧刚才重定向备份的数据库文件导入到mysql中

[root@master ~]# mysql -uroot -p < /tmp/db.dump
```

7. char 和varchar的区别

```
char定长字段，创建所有记录的值存储的长度一致，读取速度快
varchar 变长  记录的值存储长度为本身的长度，省空间

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；而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，两者的存储数据都非unicode的字符数据。
```

8. mysql执行计划的作用和使用方法

```
EXPLAIN  SELECT ……
变体：
1. EXPLAIN EXTENDED SELECT ……
将执行计划“反编译”成SELECT语句，运行SHOW WARNINGS 可得到被MySQL优化器优化后的查询语句
2. EXPLAIN PARTITIONS SELECT ……
用于分区表的EXPLAIN

预估执行语句的性能，查询速度
```

9. 1000w条数据, 使用limit分页, 为什么越往后越慢

```
LIMIT 451350 , 30 扫描了45万多行，怪不得慢的都堵死了
当一个数据库表过于庞大，LIMIT offset, length中的offset值过大，则SQL查询语句会非常缓慢，你需增加order by，并且order by字段需要建立索引。
如果使用子查询去优化LIMIT的话，则子查询必须是连续的，某种意义来讲，子查询不应该有where条件，where会过滤数据，使数据失去连续性。
如果你查询的记录比较大，并且数据传输量比较大，比如包含了text类型的field，则可以通过建立子查询。
SELECT id,title,content FROM items WHERE id IN (SELECT id FROM items ORDER BY id limit 900000, 10);
如果limit语句的offset较大，你可以通过传递pk键值来减小offset = 0，这个主键最好是int类型并且auto_increment
SELECT * FROM users WHERE uid > 456891 ORDER BY uid LIMIT 0, 10;
这条语句，大意如下：
SELECT * FROM users WHERE uid >=  (SELECT uid FROM users ORDER BY uid limit 895682, 1) limit 0, 10;
如果limit的offset值过大，用户也会翻页疲劳，你可以设置一个offset最大的，超过了可以另行处理，一般连续翻页过大，用户体验很差，则应该提供更优的用户体验给用户。
```

10 . 读写分离

https://baijiahao.baidu.com/s?id=1614304400276051465&wfr=spider&for=pc

11. char varchar

```
1 首先明确的是，char的长度是不可变的，而varchar的长度是可变的，
2 定义一个char[10]和varchar[10],如果存进去的是‘abcd’,那么char所占的长度依然为10，除了字符‘abcd’外，后面跟六个空格，而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，
3 char的存取数度还是要比varchar要快得多，因为其长度固定，方便程序的存储与查找；但是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，而varchar是以空间效率为首位的。
4 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；而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，两者的存储数据都非unicode的字符数据。
```

12 sql半同步复制原理。

```
异步复制（Asynchronous replication）

MySQL默认的复制即是异步的，主库在执行完客户端提交的事务后会立即将结果返给给客户端，并不关心从库是否已经接收并处理，这样就会有一个问题，主如果crash掉了，此时主上已经提交的事务可能并没有传到从上，如果此时，强行将从提升为主，可能导致新主上的数据不完整。

全同步复制（Fully synchronous replication）

指当主库执行完一个事务，所有的从库都执行了该事务才返回给客户端。因为需要等待所有从库执行完该事务才能返回，所以全同步复制的性能必然会收到严重的影响。

半同步复制（Semisynchronous replication）

介于异步复制和全同步复制之间，主库在执行完客户端提交的事务后不是立刻返回给客户端，而是等待至少一个从库接收到并写到relay log中才返回给客户端。相对于异步复制，半同步复制提高了数据的安全性，同时它也造成了一定程度的延迟，这个延迟最少是一个TCP/IP往返的时间。所以，半同步复制最好在低延时的网络中使用。
```

13. sql注入攻击原理, 代码层面防止sql注入

```
# 原理
通过前端的表单提交的数据中携带sql语句， 欺骗服务器， 在后端对数据进行存储时， 执行恶意的sql语句

# 出现在哪里
比如在使用pymysql操作数据库是使用字符串拼接生产sql语句， 就会出现sql注入漏洞

# 如何防止
1. 使用pymysql时， 不要使用字符串拼接， 而是使用execute方法， pymysql模块已经对提交的数据做了处理， 不会出现sql注入
2. 不要相信前端提交的任何数据， 要严格校验
3. 永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。
4. 不要把机密信息直接存放，加密或者hash掉密码和敏感的信息
5. 应用的异常信息应该给出尽可能少的提示，最好使用自定义的错误信息对原始错误信息进行包装
6. 使用辅助软件对网站进行检测， 软件一般采用sql注入检测工具jsky， 网站平台就有亿思网站安全平台检测
```

14.. python实现将数据库的student表中提取的数据写入db.txt

```
pymysql.Connect()参数说明
```

```
host(str):      MySQL服务器地址
port(int):      MySQL服务器端口号
user(str):      用户名
passwd(str):    密码
db(str):        数据库名称
charset(str):   连接编码

connection对象支持的方法
cursor()        使用该连接创建并返回游标
commit()        提交当前事务
rollback()      回滚当前事务
close()         关闭连接

cursor对象支持的方法
execute(op)     执行一个数据库的查询命令
fetchone()      取得结果集的下一行
fetchmany(size) 获取结果集的下几行
fetchall()      获取结果集中的所有行
rowcount()      返回数据条数或影响行数
close()         关闭游标对象


db = pymysql.Connect(host="127.0.0.1", port=3306, user="root", passod="", db="test")
# 使用cursor()方法获取操作游标
cursor = db.cursor()
# SQL 查询语句
sql = "SELECT * FROM student"
try:
   # 执行SQL语句
   cursor.execute(sql)
   # 获取所有记录列表
   results = cursor.fetchall()
   with open("db.txt", "w", encoding="utf8") as f:
       for row in results:
            f.write(row)
except:
   print ("Error: unable to fetch data")

# 关闭数据库连接
db.close()
```

15. 慢日志

MySQL的慢查询日志是MySQL提供的一种日志记录，它用来记录在MySQL中响应时间超过阀值的语句，具体指运行时间超过long_query_time值的SQL，则会被记录到慢查询日志中。long_query_time的默认值为10，意思是运行10s以上的语句。

默认情况下，MySQL数据库并不启动慢查询日志，需要我们手动来设置这个参数，当然，如果不是调优需要的话，一般不建议启动该参数，因为开启慢查询日志或多或少会带来一定的性能影响。慢查询日志支持将日志记录写入文件，也支持将日志记录写入数据库表。

Sql查询

1.存在的表有:

```
products(商品表) columns为id,nane, price
orders(商城订单表) columns为id, reservation_id, product id,quentity(购买数量)
reservations(酒店订单表) columns为id,user_id, price, created

 ---------------------------------------------------------------------------
需要查询的
1.各个商品的售卖情况,需要字段商品名购买总量商品收入
2.所有用户在2018-01-01至2018-82-01下单次数,下单金额,商城下单次数,商城下单金额
3.历月下单用户数:下单一次用户数,下单两次用户数,下单三次及以上用户数
```

2.考虑如下表结构，写出建表语句

id (自增主键)	name(非空)	Balance(非空)
1	A	19
2	A	20
3	A	100

3.假设学生 Students和教师 Teachers关系模型如下所示:

```
1.Student:(学号,姓名,性别,类别,身份证号)
2.Teacher:(教师号,姓名,性别,身份证号,工资)
其中,学生关系中的类别分别为”本科生”和”研究生两类”,性别分为”男”和”女”
两类

查询研究生教师平均工资(显示为平均工资),最高工资与最低工资之间的差值
(显示为差值)的SQL语句

select(1)as平均工资,(2)as差值 from Students, Teacher where(3);

查询工资少于10900元的女研究生教师的身份证号和姓名的SOL语句(非激套查
面方式);
select 身份证号,姓名 from Students where (4) (5)
select 身份证号,姓名 from Teachers where (6)
```

4.主从同步原理

1. 简述数据三大范式?
2. 什么是事务？MySQL如何支持事务?
3. 简述数据库设计中一对多和多对多的应用场景?
4. 如何基于数据库实现商城商品计数器?
5. 常见SQL（必备） 详见武沛齐博客：https://www.cnblogs.com/wupeiqi/articles/5729934.html
6. 简述触发器、函数、视图、存储过程?
7. MySQL索引种类
8. 索引在什么情况下遵循最左前缀的规则?
9. 主键和外键的区别?
10. MySQL常见的函数?
11. 列举 创建索引但是无法命中索引的8种情况。
12. 如何开启慢日志查询?
13. 数据库导入导出命令（结构+数据）?
14. 数据库优化方案?
15. char和varchar的区别?
16. 简述MySQL的执行计划?
17. 在对name做了唯一索引前提下，简述以下区别： select * from tb where name = ‘Oldboy-Wupeiqi’ select * from tb where name = ‘Oldboy-Wupeiqi’ limit 1

18. 1000w条数据，使用limit offset 分页时，为什么越往后翻越慢？如何解决？
19. 什么是索引合并？
20. 什么是覆盖索引？
21. 简述数据库读写分离？
22. 简述数据库分库分表？（水平、垂直）

1. redis和memcached比较？
2. redis中数据库默认是多少个db 及作用？
3. python操作redis的模块？
4. 如果redis中的某个列表中的数据量非常大，如果实现循环显示每一个值？
5. redis如何实现主从复制？以及数据同步机制？
6. redis中的sentinel的作用？
7. 如何实现redis集群？
8. redis中默认有多少个哈希槽？
9. 简述redis的有哪几种持久化策略及比较？
10. 列举redis支持的过期策略。
11. MySQL 里有 2000w 数据，redis 中只存 20w 的数据，如何保证 redis 中都是热点数据？
12. 写代码，基于redis的列表实现 先进先出、后进先出队列、优先级队列。
13. 如何基于redis实现消息队列？
14. 如何基于redis实现发布和订阅？以及发布订阅和消息队列的区别？
15. 什么是codis及作用？
16. 什么是twemproxy及作用？
17. 写代码实现redis事务操作。
18. redis中的watch的命令的作用？
19. 基于redis如何实现商城商品数量计数器？
20. 简述redis分布式锁和redlock的实现机制。
21. 什么是一致性哈希？Python中是否有相应模块？
22. 如何高效的找到redis中所有以oldboy开头的key？