1. 写一个脚本将数据库备份并打包至远程服务器 192.168.1.1/backup 目录下

```
mount 192.168.1.1:/backup /mnt
cd /mnt
/usr/local/mysql/bin/mysqldump -hlocalhost -uroot test > test.sql
tar czf test.sql.tar.gz test.sql
rm -f test.sql
```

2. 请写 mysql 数据库中的 SQL 查询，查找 customer 表中 uid 列内大于 100 的记录并以 uid 排序，正序输出前 10 条记录

```
select * from customer where uid > 100 order by uid asc limit 10
```

3. 数据库读写分离有什么好处

```
1. 将读操作和写操作分离到不同的数据库上，减轻了数据访问的压力，避免出现性能瓶颈；
2. 主服务器进行写操作时，不影响查询应用服务器的查询性能，降低阻塞，提高并发；
3. 数据拥有多个容灾副本，提高数据安全性，同时当主服务器故障时，可立即切换到其他服务器，提高系统可用性
```

4. MongoDB 与 mysql 有什么区别

```
1. MongoDB 在不指定 _id 的情况下插入速度很快，对系统内存利用率很高，适合那些读作业任务很重的任务模型，但是稳定性不如 mysql。

2. 适合数据具体格式不明确的（弱数据结构）的情况下，对开发者友好。但是事务关系支持薄弱（nosql 通病）

3. 自带分布式文件系统，可以很方便的部署到集群上。

4. shard（数据实际存储）概念，每增加一个 shard，插入性能会以接近倍数的方式增长，磁盘容量也可以方便的扩充。

5. shard 划分，包括：mongos（接收 client 需求，shard 集群接口）,config（元数据或者配置信息，如每一个切片的片键范围，mongos 会根据 config 信息去取数据）
```

5. redis 与 memcache 区别以及应用场景

```
1. redis 支持更丰富的数据类型，共有5种（字符串，列表，hash，集合，有序集合），memcache 只支持简单的字符串
2. redis 可以数据持久化，它不仅会将数据放在内存中，还可以存到磁盘里部分，memcache 会将数据存在内存中，最大存储量即内存大小
3. redis 可以做主从
4. redis 最大可以存 1G，memcache 最大 1M

redis 应用场景
```

1. session ：可以做持久化
2. 全页缓存（FPC）：持久化，重启后，用户看到的网页速度依然不会变慢
3. list 和 set 可以让 redis 用来做消息队列
4. 排行榜，set 和 zset 可以很好地进行数字排序
5. 发布/订阅功能

redis 注意事项：
1. redis 的主最好不要做持久化，可以对从机做（AOF）持久化，策略为每秒一次
2. 为了主从稳定，最好主从在同一个网段内
3. 尽量避免在压力过大的 redis 主上添加从机
4. 主从复制，不要用图形结构，用单链表更加稳定，这样方便解决单点故障问题，如果主挂了，直接将 slave1 提升为主

6. redis 的数据结构

string ---- 字符串类型，使用场景：做缓存，计数器，共享 session
hash ---- 哈希类型，使用场景：用户信息存储
list ---- 列表类型，使用场景：消息队列，微博 TimeLine
set ---- 集合类型，使用场景：好友推荐
Sorted Set ---- 有序集合类型，使用场景：排行榜

7. 生产环境 redis 的开源高可用方案有哪些?

1. Twemproxy，大概概念是，它类似于一个代理方式，使用方法和普通 redis 无任何区别，设置好它下属的多个 redis 实例后，使用时在本需要连接 redis 的地方改为连接 twemproxy，它会以一个代理的身份接收请求并使用一致性 hash 算法，将请求转接到具体 redis，将结果再返回 twemproxy。使用方式简便(相对 redis 只需修改连接端口)，对旧项目扩展的首选。 问题：twemproxy 自身单端口实例的压力，使用一致性 hash 后，对 redis 节点数量改变时候的计算值的改变，数据无法自动移动到新的节点。

2. Codis，目前用的最多的集群方案，基本和 twemproxy 一致的效果，但它支持在 节点数量改变情况下，旧节点数据可恢复到新 hash 节点。

3. Redis cluster3.0 自带的集群，特点在于他的分布式算法不是一致性 hash，而是 hash 槽的概念，以及自身支持节点设置从节点。具体看官方文档介绍。

4. 在业务代码层实现，起几个毫无关联的 redis 实例，在代码层，对 key 进行 hash 计算，然后去对应的 redis 实例操作数据。 这种方式对 hash 层代码要求比较高，考虑部分包括，节点失效后的替代算法方案，数据震荡后的自动脚本恢复，实例的监控，等等。

5. Redis哨兵，Redis Sentinal 着眼于高可用，在 master 宕机时会自动将 slave 提升为 master，继续提供服务。

8. Redis 是单线程的，如何提高多核 CPU 的利用率?

可以在同一个服务器部署多个 Redis 的实例，并把他们当作不同的服务器来使用，在某些时候，无论如何一个服务器是不够的， 所以，如果你想使用多个 CPU，你可以考虑一下分片（**shard**）。

9. Redis 常见性能问题和解决方案?

1. Master **最好不要做任何持久化工作，如 RDB** 内存快照和 AOF 日志文件

2. 如果数据比较重要，某个 Slave 开启 AOF 备份数据，策略设置为每秒同步一次

3. 为了主从复制的速度和连接的稳定性，**Master 和 Slave** 最好在同一个局域网内

4. 尽量避免在压力很大的主库上增加从库

5. 主从复制不要用图状结构，用单向链表结构更为稳定，即：**Master <- Slave1** <- Slave2 <- Slave3 ...
这样的结构方便解决单点故障问题，实现 Slave 对 Master 的替换。如果 Master 挂了，可以立刻启用 Slave1 做 Master，其他不变

10. Redis 提供了哪几种持久化方式?

RDB 持久化方式能够在指定的时间间隔能对你的数据进行快照存储.
AOF 持久化方式记录每次对服务器写的操作，当服务器重启的时候会重新执行这些命令来恢复原始的数据，AOF 命令以 redis 协议追加保存每次写的操作到文件末尾。Redis 还能对 AOF 文件进行后台重写，使得 AOF 文件的体积不至于过大。
如果你只希望你的数据在服务器运行的时候存在，你也可以不使用任何持久化方式。
你也可以同时开启两种持久化方式，在这种情况下，当 redis 重启的时候会优先载入 AOF 文件来恢复原始的数据，因为在通常情况下 AOF 文件保存的数据集要比RDB文件保存的数据集要完整。
最重要的事情是了解 RDB 和 AOF 持久化方式的不同，让我们以 RDB 持久化方式开始。

11. 修改配置不重启 Redis 会实时生效吗?

针对运行实例，有许多配置选项可以通过 CONFIG **SET** 命令进行修改，而无需执行任何形式的重启。 从 Redis 2.2 开
始，可以从 AOF 切换到 RDB 的快照持久性或其他方式而不需要重启 Redis。检索 'CONFIG GET *'' 命令获取更多信
息。
但偶尔重新启动是必须的，如为升级 Redis 程序到新的版本，或者当你需要修改某些目前 CONFIG 命令还不支持的配置
参数的时候。

12. Redis 集群会有写操作丢失吗？为什么?

Redis 并不能保证数据的强一致性，这意味这在实际中集群在特定的条件下可能会丢失写操作。

13. 查看 Redis 使用情况及状态信息用什么命令?

```
info
```

14. Mongodb 熟悉吗，一般部署几台?

一般 mongodb 部署主从、或者 mongodb 分片集群；建议 3 台或 5 台服务器来部署。MongoDB 分片的基本思想就是将集合切分成小块。
这些块分散到若干片里面，每个片只负责总数据的一部分。 对于客户端来说，无需知道数据被拆分了，也无需知道服务端哪个分片对应哪些数据。
数据在分片之前需要运行一个路由进程，进程名为 mongos。
这个路由器知道所有数据的存放位置，知道数据和片的对应关系。
对客户端来说，它仅知道连接了一个普通的 mongod，在请求数据的过程中，通过路由器上的数据和片的对应关系，路由到目标数据所在的片上，如果请求有了回应，路由器将其收集起来回送给客户端。

15. Rabbitmq 和 kafka 有什么区别?

1．在架构模型方面
RabbitMQ 遵循 AMQP 协议，RabbitMQ 的 broker 由 Exchange，Binding，queue 组成，其中 exchange 和 binding 组成了消息的路由键；客户端 Producer 通过连接 channel 和 server 进行通信，Consumer 从 queue 获取消息进行消费（长连接，queue 有消息会推送到 consumer 端，consumer 循环从输入流读取数据）。rabbitMQ 以 broker 为中心；有消息的确认机制。
kafka 遵从一般的MQ结构，producer，broker，consumer，以 consumer 为中心，消息的消费信息保存的客户端 consumer 上，consumer 根据消费的点，从 broker 上批量 pull 数据；无消息确认机制。

2．在吞吐量
kafka 具有高的吞吐量，内部采用消息的批量处理，zero-copy 机制，数据的存储和获取是本地磁盘顺序批量操作，具有 O(1) 的复杂度，消息处理的效率很高。
rabbitMQ 在吞吐量方面稍逊于 kafka，他们的出发点不一样，rabbitMQ 支持对消息的可靠的传递，支持事务，不支持批量的操作；基于存储的可靠性的要求存储可以采用内存或者硬盘。

3．在可用性方面
rabbitMQ 支持 miror 的 queue，主 queue 失效，miror queue 接管。
kafka 的 broker 支持主备模式。

4．在集群负载均衡方面
kafka 采用 zookeeper 对集群中的 broker、consumer 进行管理，可以注册 topic 到 zookeeper上；通过 zookeeper 的协调机制，producer 保存对应 topic 的 broker 信息，可以随机或者轮询发送到 broker 上；并且 producer 可以基于语义指定分片，消息发送到 broker 的某分片上。
rabbitMQ 的负载均衡需要单独的 loadbalancer 进行支持。

16. 备份的分类有哪些?

系统备份：针对整个操作系统进行备份；当操作系统损坏或者无法启动时，能通过备份快速恢复。
数据备份：针对用户的数据文件、应用软件、数据库进行备份；当这些数据丢失或损坏时，也能通过备份恢复。

17. 什么是冷/热备份？他们各自有什么优点和缺点?

冷备份
需要备份的文档先关闭停止使用，再执行备份的方式；
优点是简单快速、容易恢复到某个时间点、方便维护；
缺点是只能恢复到某个时间点、备份期间数据不便正常使用。

热备份
指执行备份时不影响备份文档正常使用的方式；

优点是备份速度快、不影响数据使用；
缺点是所有操作都会同步，包括删除。

18. 如果有一个 100G 大小的数据库该如何做备份?

100G 以上的库，可以考虑用 xtranbackup 来做，备份速度明显要比 mysqldump 要快。一般是选择一周一个全备，其
余每天进行增量备份，备份时间为业务低峰期。

19. 备份恢复失败如何处理?

首先在恢复之前就应该做足准备工作，避免恢复的时候出错。比如说备份之后的有效性检查、权限检查、空间检查等。如
果万一报错，再根据报错的提示来进行相应的调整。

20. mysqldump 备份的本质、优点和缺点?

本质：导出的是 **sql** 语句文件
优点：无论是什么存储引擎，都可以用 mysqldump 备成 **sql** 语句
缺点：速度较慢，导入时可能会出现格式不兼容的突发状况，无法直接做增量备份。

21. 逻辑备份是什么?

备份的是建表、建库、插入等操作所执行 **SQL** 语句（DDL DML DCL）。
适用于中小型数据库，效率相对较低。 一般在数据库正常提供服务的前提下进行，如：mysqldump、mydumper、**into**、outfile（表的导出导入）等。

22. 物理备份是什么?

直接复制数据库文件 dbfile、binary、log、**my**.cnf
适用于大型数据库环境，不受存储引擎的限制，但不能恢复到不同的 MySQL 版本。

23. 什么是存储引擎？最常用的存储引擎有哪些?

1. 存储引擎说白了就是如何管理操作数据（存储数据、如何更新、查询数据等）的一种方法和机制。
2. 在 MySql 数据库中提供了多种存储引擎，各个存储引擎的优势各不一样。
3. 用户可以根据不同的需求为数据表选择不同的存储引擎，也可以根据自己的需要编写自己的存储引擎。
4. 甚至一个库中不同的表使用不同的存储引擎，这些都是允许的。

最常用的存储引擎是 MyISAM 和 InnoDB。

24. 什么是 mysql 的二进制日志 (binary log)?

二进制日志记录数据库的所有更改操作（DDL/DML/DCL），不包含 **select** 或者 **show** 这类语句。
用于主从复制中，**master** 主服务器将二进制日志中的更改操作发送给 **slave** 从服务器，从服务器执行这些更改操作是的和主服务器的更改相同。
用于数据的恢复操作。
默认二进制日志是关闭的，可以使用 **log-bin**=xxx 参数开启

25. Binlog 工作模式有哪些？各什么特点，企业如何选择？

1.Row(行模式)；
日志中会记录成每一行数据被修改的形式，然后在 slave 端再对相同的数据进行修改

2.Statement(语句模式)
每一条修改的数据都会完整的记录到主库 **master 的 binlog** 里面，在 slave 上完整执行在 **master 执行的 sql** 语句

3.mixed(混合模式)
结合前面的两种模式，如果在工作中有使用函数 或者触发器等特殊功能需求的时候，使用混合模式
数据量达到比较高时候，它就会选择 statement 模式，而不会选择 Row Level 行模式

26. 如何在线正确清理 MySQL binlog?

MySQL 中的 binlog 日志记录了数据中的数据变动，便于对数据的基于时间点和基于位置的恢复
但日志文件的大小会越来越大，点用大量的磁盘空间，因此需要定时清理一部分日志信息
手工删除：
首先查看主从库正在使用的 binlog 文件名称
```
show master(slave) status\G
```
删除之前一定要备份，删除指定时间前的日志：
```
purge master logs before ' 2017-09-01 00:00:00' ;
```
删除指定的日志文件：
```
purge master logs to ' mysql-bin.000001' ;
```

自动删除：
通过设置 binlog 的过期时间让系统自动删除日志，查看过期时间与设置过期时间
```
show variables like  'expire_logs_days' ;
set global expire_logs_days = 30;
```

27. 什么是 mysql 的中继日志?

用于主从复制，**master 主服务器将自己的二进制日志发送给 slave** 从服务器，slave 先保存在自己的中继日志中，然后再执行自己本地的 relay log 里的 sql 达到数据库更改和 **master 保持一致。**
默认中继日志没有开启，可以使用 `relay`-log` 参数开启

28. 数据库和数据库实例之间的关系是什么?

通常情况下，数据库实例和数据库是一一对应的关系，也就是一个数据库实例对应一个数据库； 但是，在集群环境中存在多个数据库实例共同使用一个数据库。比如：**oracle** RAC

29. 什么叫 mysql 事务?

MySQL 事务主要用于处理操作量大，复杂度高的数据。比如说，在人员管理系统中，你删除一个人员，你即需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作语句就构成一个事务！一般来说，事务是必须满足4个条（ACID）：
原子性（Atomicity，或称不可分割性）、一致性（Consistency）、隔离性（Isolation，又称独立性）、持久性（Durability）。

30. MySQL 密码丢了，请找回?

```
mysqld_safe --skip-grant-tables &
#启动数据库服务
mysql -uroot -ppassowrd -e "use mysql;update user set passowrd = PASSWORD('newpassword')
where user = 'root';flush privileges;"
```

31. 请说出非关系型数据库的典型产品、特点及应用场景?

1. memecached 纯内存
2. redis 持久化缓存
3. mongodb 面向文档

如果需要短时间相应的查询操作，没有良好模式定义的数据存储，或者模式更改频繁的数据存储还是用 NoSQL

32. 什么是 MySQL 多实例，如何配置 MySQL 多实例?

mysql 多实例就是在同一台服务器上启用多个 mysql 服务，它们监听不同的端口，运行多个服务进程，它们相互独立，互不影响的对外提供服务，便于节约服务器资源与后期架构扩展
多实例的配置方法有两种：

1. 一个实例一个配置文件，不同端口
2. 同一配置文件(my.cnf)下配置不同实例，基于 mysqld_multi 工具

33. 如何加强 MySQL 安全，请给出可行的具体措施?

1. 删除数据库不使用的默认用户
2. 配置相应的权限（包括远程连接）
3. 不可在命令行界面下输入数据库的密码
4. 定期修改密码与加强密码的复杂度

34. MySQL Sleep 线程过多如何解决?

1. 可以杀掉 sleep 进程，kill PID
2. 修改配置，重启服务

```
[mysqld]
wait_timeout = 600
interactive_timeout=30
```

注意：如果生产服务器不可随便重启也可以使用下面的方法解决

```
set global wait_timeout=600
set global interactive_timeout=30;
```

35. sort_buffer_size 参数作用？如何在线修改生效?

```
在每个 connection(session) 第一次连接时需要使用到，来提访问性能
set global sort_buffer_size = 2M
```

36. MySQL 中 myisam 与 innodb 的区别?

```
1. InnoDB 支持事物，而 MyISAM 不支持事物
2. InnoDB 支持行级锁，而 MyISAM 支持表级锁
3. InnoDB 支持 MVCC，而 MyISAM 不支持
4. InnoDB 支持外键，而 MyISAM 不支持
5. InnoDB 不支持全文索引，而 MyISAM 支持。
```

37. 如何调整生产线中 MySQL 数据库的字符集?

```
1. 首先导出库的表结构 -d 只导出表结构，然后批量替换
2. 导出库中的所有数据（在不产生新数据的前提下）
3. 然后全局替换 set names = xxxxx
4. 删除原有库与表，并新创建出来，再导入建库与建表语句与所有数据
```

38. 请描述 MySQL 里中文数据乱码原理，如何防止乱码?

```
服务器系统、数据库、客户端三方字符集不一致导致，需要统一字符
```

39. 企业生产 MySQL 如何优化（请多角度描述）？

```
1. 提升服务器硬件资源与网络带宽
2. 优化 mysql 服务配置文件
3. 开启慢查询日志然后分析问题所在
```

40. 生产中误操作执行了一个 drop 库 SQL 语句，如何完整恢复?

```
1. 停止主从复制，在主库上执行锁表并刷新 binlog 操作，接着恢复之前的全备文件（比如 0 点的全备）
2. 将 0 点时的 binlog 文件与全备到故障期间的 binlog 文件合并导出成 sql 语句
mysqlbinlog –no-defaults mysql-bin.000011 mysql-bin.000012 >bin.sql
3. 将导出的 sql 语句中 drop 语句删除，恢复到数据库中
mysql -uroot -pmysql123 < bin.sql
```

41. 工作中遇到过哪些数据库故障，请描述 2 个例子?

1. 开发使用 root 用户在从库上写入数据造成主从数据不一致，并且前端没有展示需要修改的内容（仍旧是老数据）
2. 内网测试环境服务器突然断电造成主从同步故障