周测

选择题:每题4分,共40分

1. 关于redis持久化描述错误的有(D)
2. RDB是全量备份
3. AOF是增量备份
4. RDB备份的是全部的命令
5. AOF备份的是增量的命令
6. 关于RDB持久化描述错误的是(C)
7. RDB可在redis终端中触发
8. RDB可以在配置中配置
9. RDB默认是关闭的，需要手动在配置中打开
10. RDB可能会丢失一部分数据
11. 以下针对于AOF说法错误的是(B)
12. AOF相对于RDB备份，能减少数据丢失
13. AOF可以做到每个命令都写入到磁盘
14. AOF默认为每秒刷新缓冲区的内容到磁盘中
15. redis重启时，AOF数据还原速度比RDB更快
16. 以下对redis得主从描述错误的是(C)
17. redis得主负责写入，从负责读取
18. redis配置主从后，只能用从读取数据，主只能进行写入
19. 从服务可以直接在任意redis终端中直接配置
20. slave of no one 可将从脱离主
21. 以下对redis有序集合描述错误的是？ (C)
22. 有序集合得元素不支持复杂嵌套
23. 有序集合每个元素要有一个分值，分值关系到元素顺序
24. zrange命令默认获取升序得数据
25. 有序集合可以做 差集 并集 交集得操作
26. 以下对redis集合描述错误的是？（D）
27. SRANDMEMBER命令可随机弹出一个元素，但是不删除元素
28. SPOP命令可随机弹出一个元素，且删除元素
29. 集合具备无序，去重得特点
30. 集合可以嵌套集合

1. 关于redis位图操作说法错误的是(A)
2. 位图操作实际操作的是字符串结构
3. bitset可以设置字符串的2进制位
4. bitset得返回值是该位2进制set以前得值
5. bitcount可根据参数得不同统计出 0 或者 1 得数量
6. 关于redis得hash结构说法错误的是(D)
7. hash可以按字段取值，非常灵活
8. hash结构可以做缓存
9. hash 可以给每个字段设置不同的过期时间
10. hash 结构是无序得
11. 以下说法错误的是(C)
12. redis支持弱事务
13. redis的事务不具备原子性
14. redis的事务具备一致性
15. pyredis需要用流水线调用redis的事务
16. 以下属于事务的特性为(CD) 【多选】
17. 原子性
18. 一致性
19. 持久性
20. 可用性

简答题 每题5分；共30分：

1. 谈谈redis基础五大数据结构中，具体都分别适用于哪些场景中？

String——字符串 GETSET 命令的妙用，请于清空旧值的同时设置一个新值，配合原子计数器使用  
Hash——字典 存储、读取、修改用户属性  
List——列表 1.微博 TimeLine 2.消息队列  
Set——集合 Set——集合  
Sorted Set——有序集合 带有权重的元素，比如一个游戏的用户得分排行榜

1. 简述一下jwt是的组成？

jwt是信息加密的一种方式，一个JWT由三个部分组成：header，payload，signatur

1. 请简述一下cors跨域原理

CORS原理只需要向响应头header中注入Access-Control-Allow-Origin，这样浏览器检测到header中的Access-Control-Allow-Origin，则就可以跨域操作了。

1. CORS中的预检请求每次都发出两个请求，会加大服务器压力，这块您有什么解决方案？

那么面对这种跨域预检机制造成的多次请求问题，我们可以在后台设置Access-Control-Max-Age来控制浏览器在多长时间内（单位s）无需在请求时发送预检请求，从而减少不必要的预检请求。或者将非简单请求降级成简单请求。

1. 请简述一下hash算法的特点及使用场景？

* 从哈希值不能反向推导出原始数据（所以哈希算法也叫单向算法，不可逆）；
* 对输入数据非常敏感，哪怕原始数据只修改了一个比特，最后得到的哈希值也大不相同；
* 散列冲突的概率要很小，对于不同的原始数据，哈希值相同的概率非常小；
* 哈希算法的执行效率要尽量高效，针对较长的文本，也能快速地计算出哈希值

**哈希算法的应用**

   1、场景一：安全加密

       日常用户密码加密通常使用的都是 md5、sha等哈希函数，因为不可逆，而且微小的区别加密之后的结果差距很大，所以安全性更好。

   2、场景二：唯一标识

       比如 URL 字段或者图片字段要求不能重复，这个时候就可以通过对相应字段值做 md5 处理，将数据统一为 32 位长度从数据库索引构建和查询角度效果更好，此外，还可以对文件之类的二进制数据做 md5 处理，作为唯一标识，这样判定重复文件的时候更快捷。

   3、场景三：数据校验

       比如从网上下载的很多文件（尤其是P2P站点资源），都会包含一个 MD5 值，用于校验下载数据的完整性，避免数据在中途被劫持篡改。

   4、场景五：散列函数

       前面已经提到，PHP 中的 md5、sha1、hash 等函数都是基于哈希算法计算散列值

   5、场景五：负载均衡

       对于同一个客户端上的请求，尤其是已登录用户的请求，需要将其会话请求都路由到同一台机器，以保证数据的一致性，这可以借助哈希算法来实现，通过用户 ID 尾号对总机器数取模（取多少位可以根据机器数定），将结果值作为机器编号。

   6、场景六：分布式缓存

       分布式缓存和其他机器或数据库的分布式不一样，因为每台机器存放的缓存数据不一致，每当缓存机器扩容时，需要对缓存存放机器进行重新索引（或者部分重新索引），这里应用到的也是哈希算法的思想。

1. 请简述一下base64的计算流程？

第一步：每3个字节（8位二进制）为一组，一共24个二进制位

　　第二步：将这个24个二进制位分成4组，每个组有6个二进制位，不足6位的，后面补0。

　　第三步：在每个组前面加两个0，这样每个组就又变成了8位，即每个组一个字节，4个组就4个字节了。

　　第四步：根据Base64的转码表找到每个字节对应的符号，这个符号就是Base64的编码值

编程题（30分）：

编写一个小说网站 阅读量排行榜 功能，要求如下：

1. 视图函数： topic\_detail(request, topic\_id)

作用：返回topic\_id的具体文章详情；即论坛文章详情页对应视图

返回值：返回字符串 ‘%s topic detail’%(topic\_id) 即可

编码要求：执行该试图函数后 再全网的 阅读排行榜 中给该topic\_id文章计数+1

def topic\_detail(request, topic\_id) :

1. 试图函数：topic\_index(request)

作用：模拟小说网站首页

返回值：输出全网 小说阅读量最多的 前十名 文章id

编码要求：返回排名即可，形式不限