、一.平安产险

A.JAVA线程池的参数

B.表查询（找保单号最新的那一条）

保单号 更新时间

1 2022-3-4

1 2022-3-6

2 2022-3-8

2 2022-3-9

SELECT

\*

FROM

(SELECT

\*,

ROW\_NUMBER () OVER (

PARTITION BY INSURANCE\_NO

ORDER BY INSURANCE\_DATE DESC

) ranks

FROM

T\_INSURANCE) a

WHERE a.ranks = 1

C.行转列，列转行

思路：行转列主要适用于对数据作聚合统计，

2.JAVA多线程

二、创势互联：

A.跳槽比较频繁

B.达不到管理岗位

C.搞IT有11年了，技术没有达到一个阶段

连技术面试的机会，都没给。感觉自己方向都没有确定。

（有时想做BA,有时想做电商，有时候想做测试，有时候想做开发）在外面面试，感觉赛意还是不错的。现在华为外包要求是本科学历。不好整。

三、天阳宏业科技股份有限公司

1.@RestControler和@Controller的不同

答：如果想要做页面跳转的化，就使用@Controller，不要做加上@ResposeBody。(json)

2.分布式事务是怎么处理的？

阶段提交

实现的是最终一致性，容忍了数据暂时不一致的情况.BASE定理

三.2022年4月26日

软通（华为外包面试）

a.怎么保证rocket MQ重启后，消息不丢失？

同步复制多Master多Slave模式(可以通过同步的方式阻塞式的发送)

b.innoDB和MyIsam有什么区别？

(1）.innoDB支持事务,MyIsam不支持事务

(2).如果表中绝大多数都只是读查询，可以考虑 MyISAM，如果既有读写也挺频繁，请使用InnoDB。

(3).系统奔溃后，MyISAM恢复起来更困难，能否接受，不能接受就选 InnoDB；

（4）.innoDB支持行级锁，MyIsam支持表级锁

c.事务隔离级别有哪几种？解决什么问题？

Read uncommitted 、Read committed 、Repeatable read 、Serializable,解决并发问题

d.多线程应用场景

后台发大量邮件

6.join()：让并行变成串行

public class JoinTest {

public static void main(String [] args) throws InterruptedException {

ThreadJoinTest t1 = new ThreadJoinTest("小明");

ThreadJoinTest t2 = new ThreadJoinTest("小东");

t1.start();

/\*\*join的意思是使得放弃当前线程的执行，并返回对应的线程，例如下面代码的意思就是：

程序在main线程中调用t1线程的join方法，则main线程放弃cpu控制权，并返回t1线程继续执行直到线程t1执行完毕

所以结果是t1线程执行完后，才到主线程执行，相当于在main线程中同步t1线程，t1执行完了，main线程才有执行的机会

\*/

t1.join();

t2.start();

}

}

class ThreadJoinTest extends Thread{

public ThreadJoinTest(String name){

super(name);

}

@Override

public void run(){

for(int i=0;i<1000;i++){

System.out.println(this.getName() + ":" + i);

}

}

}

————————————————

自己补充：

7.TOPIC:它只是一类数据的集合

RocketMQ 执行流程

1、启动 Namesrv，Namesrv 起 来后监听端口，等待 Broker、Producer、Consumer 连上来，相当于一个路由控制中心。

2、Broker 启动，跟所有的 Namesrv 保持长连接，定时发送心跳包。

3、收发消息前，先创建 Topic 。创建 Topic 时，需要指定该 Topic 要存储在 哪些 Broker 上。也可以在发送消息时自动创建 Topic。

4、Producer 发送消息。

5、Consumer 消费消息

四、深圳杰讯互联科技有限公司

1.多线程生产者和消费者

2.spring boot启动过程和注解

五、法本HR--平安外包

TCP三次握手：

1.三次握手：

第一次握手：客户端发送syn包(syn=x)到服务器，并进入SYN\_SEND状态，等待服务器确认；

第二次握手：服务器收到syn包，必须确认客户的SYN（ack=x+1），同时自己也发送一个SYN包（syn=y），即SYN+ACK包，此时服务器进入SYN\_RECV状态；

第三次握手：客户端收到服务器的SYN＋ACK包，向服务器发送确认包ACK(ack=y+1)，此包发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手。

握手过程中传送的包里不包含数据，三次握手完毕后，客户端与服务器才正式开始传送数据。理想状态下，TCP连接一旦建立，在通信双方中的任何一方主动关闭连接之前，TCP 连接都将被一直保持下去。

确认号：其数值等于发送方的发送序号+1(即接收方期望接收的下一个序列号)。

2.redis持久化机制是怎么样的？

redis持久化分为：RDB持久化(快照持久化)、AOF持久化

RDB持久化：

Redis可以通过创建快照来获得存储在内存里面的数据在某个时间点上的副本。Redis创建快照之后，可以对快照进行备份，可以将快照复制到其他服务器从而创建具有相同数据的服务器副本（Redis主从结构，主要用来提高Redis性能），还可以将快照留在原地以便重启服务器的时候使用(Redis默认采用的持久化方式)。

AOF持久化：

将每条写命令作为日志，以append-only模式写入一个日志文件，在Redis重启时，通过回放日志中的写入指令来重构整个数据

与快照持久化相比，AOF持久化 的实时性更好，因此已成为主流的持久化方案。默认情况下Redis没有开启AOF（append only file）方式的持久化，可以通过appendonly参数开启。

3.Redis和mysql数据怎么保持数据一致的？（重点）

答：一般是是采用延时双删策略。

延时双删策略：

删缓存

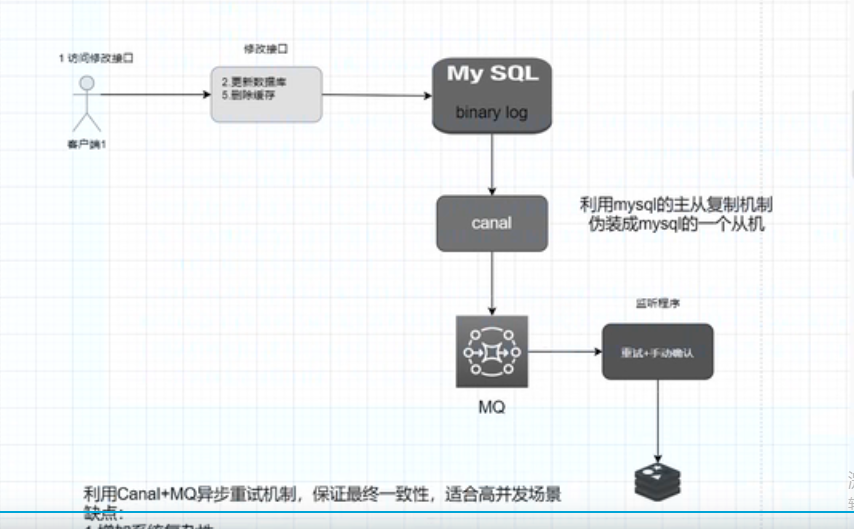
更新数据库

延时500MS

删除缓存

缺点：有延时操作，会造成服务器的阻塞。不适合高并发场景。如果要提高吞吐量，我们可以采用追求最终一致性方案。

先更新数据库，再删除REDIS.确保REDIS（重试+确认）



异步更新缓存(基于订阅binlog的同步机制)

1.技术整体思路：

MySQL binlog增量订阅消费+消息队列+增量数据更新到redis

读Redis：热数据基本都在Redis

写MySQL:增删改都是操作MySQL

更新Redis数据：MySQ的数据操作binlog，来更新到Redis

2.Redis更新

(1）数据操作主要分为两大块：

一个是全量(将全部数据一次写入到redis)

一个是增量（实时更新）

这里说的是增量,指的是mysql的update、insert、delate变更数据。

(2）读取binlog后分析 ，利用消息队列,推送更新各台的redis缓存数据

4.mongoDB有用过吗

5.JVM调优

（1）类加载机制一共有五个步骤，分别是加载、链接、初始化、使用和卸载阶段，这五个阶段的顺序是确定的。

（2）如何判断对象已经死亡？

这里有两种判断方式，首先我们先来说第一种：引用计数法。

在对象中添加一个引用计数器，每当有一个地方引用它时，计数器的值就会加一；当引用失效时，计数器的值就会减一；只要任何时刻计数器为零的对象就是不会再被使用的对象

有一种判断对象无用的方法就是可达性分析算法

可达性分析算法：

通过一系列被称为GC Roots的根对象作为起始节点集，从这些节点开始，根据引用关系向下搜索，搜索过程走过的路径被称为引用链（Reference Chain），如果某个对象到 GC Roots 之间没有任何引用链相连接，或者说从 GC Roots 到这个对象不可达时，则证明此这个对象是无用对象，需要被垃圾回收。

（3）垃圾回收算法

标记-清除算法

标记-复制算法

标记-整理算法

（4）MyISAM和InnoDB引擎的区别？

A. 是否要支持事务，如果要请选择innodb，如果不需要可以考虑MyISAM；

B. 如果表中绝大多数都只是读查询，可以考虑MyISAM，如果既有读也有写，请使用InnoDB。

C. 系统奔溃后，MyISAM恢复起来更困难，能否接受；

(5).SQL优化：

1.使用工具去诊断（pt-query-digest）

2.选择优化的数据类型

a.更小的通常更好

b.索引列，避免有null列

（慷慨不是明智的）

c.将一个复杂的查询，转换为多个简单的查询

d.多用等值查询，少用子查询

e.select \* 用具体的字段代替。

补充：

数据库三大范式：

1、第一范式(确保每列保持原子性)

2、第二范式(确保表中的每列都和主键相关)

3、第三范式(确保每列都和主键列直接相关,而不是间接相关)

六vivo

1.maven冲突怎么处理

答：

（1）：使用maven提供的传递依赖原则

（A）第一声明者优先原则

（B）路径者优先原则

（2）：排除依赖（有时候会用到）

（3）：版本锁定（最常用，推荐使用）

对于版本冲突的问题，我们最常使用的解决方式，是使用dependencyManagement统一对依赖的版本进行定义。这样的话，导入依赖的时候就会直接使用我们锁定的依赖版本。

2.git的一些命令

（1）、用特定的软件打开文件

open -a 软件名称 文件目录

（2）、git查看配置信息

git config --list

git config user.name

（3）、配置个人的用户名称和电子邮件地址

git config --global user.name "runoob"

git config --global user.email test@runoob.com

（4）、git基本概念

Git 工作区、暂存区和版本库概念：

工作区：就是你在电脑里能看到的目录。

暂存区：英文叫 stage 或 index。一般存放在 .git 目录下的 index 文件（.git/index）中，所以我们把暂存区有时也叫作索引（index）。

版本库：工作区有一个隐藏目录 .git，这个不算工作区，而是 Git 的版本库

（5）、git提交代码到仓库

git add \*.c

git add README

git commit -m '初始化项目版本'

git init

git add .

git commit

（6）、拷贝项目到本地

git clone 链接url

（7）、提交与修改

git add 添加文件到仓库

git status 查看仓库当前的状态，显示有变更的文件。

git diff 比较文件的不同，即暂存区和工作区的差异。

git commit 提交暂存区到本地仓库。

git reset 回退版本。

git rm 删除工作区文件。

git mv 移动或重命名工作区文件。

（8）、提交日志

git log 查看历史提交记录

git blame <file> 以列表形式查看指定文件的历史修改记录

（9）、远程操作

git remote 远程仓库操作

git fetch 从远程获取代码库

git pull 下载远程代码并合并

git push 上传远程代码并合并

（10）、Git 分支管理

git branch 查看分支

git branch aa 添加分支aa

git checkout aa 切换到aa分支

git branch -d aa 删除aa分支

git branch -d aa 合并完后就可以删除分支:

3.浅拷贝和深拷贝的具体实现有那些？

所以深拷贝和浅拷贝的根本区别在于：

浅拷贝：只复制了指向堆内存中的内存地址，而并不是赋值对象本身，复制的变量和原变量都指向同一个内存地址。一旦数据发生改变，指向该内存地址的变量都会跟着改变。

深拷贝：深拷贝是复制了整个对象本身，同时新对象和旧对象不是同一个内存地址，二者互不影响。

4.什么时候序列化？

序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决在对对象流进行读写操作时所引发的问题。

5.spring mvc注解

（1）、@ResponseBody：表示方法的返回结果直接写入HTTP response body中

（2）、@Controller：用来定义控制器类

（3）、@RestController：是@ResponseBody和@Controller的合集

（4）、@RequestParam：用在方法的参数前面

（5）、@PathVariable：获取路径参数

（6）、@GetMapping：是@RequestMapping(method = RequestMethod.GET)的缩写。

不支持@RequestMapping的自定义属性。

（7）、@RequestBody

（8）、@RequestMapping：提供路由信息，负责URL到Controller中的具体函数的映射

（9）、@PostMapping：是@RequestMapping(method = RequestMethod.POST)的缩写。不支持@RequestMapping的自定义属性。

（10）、@ExceptionHandler：用在方法上表示遇到这个异常就执行以下方法。

（11）、@ControllerAdvice：统一处理异常

6.spring boot注解，由哪三个注解组成？及常用注解有哪些？

a.@SpringBootApplication

b.@EnableAutoConfiguration

c.@Configuration

d.@ComponentScan

e.@Repository

f.@Service

g.@RestController

h.@ResponseBody

@Component

@Bean

@AutoWired

@Qualifier

@Resource(name=“name”,type=“type”)

@RequestMapping

RequestParam

@PathVariable

@Profiles

@ConfigurationProperties

7.补充设计模式

（1）单例模式：节约资源

（2）工厂模式(IOC)：减少依赖,控制反转（依赖注入）

（3）AOP:简单地说，就是将那些与业务⽆关，却为业务模块所共同调⽤的逻辑或责任封装起来，⽐如⽇志记录，

便于减少系统的重复代码，降低模块间的耦合度，并有利于未来的可操作性和可维护性。

（4）高并发有哪三种解决方法？

a：系统拆分，将一个系统拆分为多个子系统，用dubbo来搞。然后每个系统连一个数据库，这样本来就一个库，现在多个数据库，这样就可以抗高并发。

b：缓存，必须得用缓存。大部分的高并发场景，都是读多写少，那你完全可以在数据库和缓存里都写一份，然后读的时候大量走缓存不就得了。毕竟人家redis轻轻松松单机几万的并发啊。没问题的。所以你可以考的虑考虑你的项目里，那些承载主要请求读场景，怎么用缓存来抗高并发。

c：MQ(消息队列)，必须得用MQ。可能你还是会出现高并发写的场景，比如说一个业务操作里要频繁搞数据库几十次，增删改增删改，疯了。那高并发绝对搞挂你的系统，人家是缓存你要是用redis来承载写那肯定不行，数据随时就被LRU(淘汰掉最不经常使用的)了，数据格式还无比简单，没有事务支持。所以该用mysql还得用mysql啊。那你咋办？用MQ吧，大量的写请求灌入MQ里，排队慢慢玩儿，后边系统消费后慢慢写，控制在mysql承载范围之内。所以你得考虑考虑你的项目里，那些承载复杂写业务逻辑的场景里，如何用MQ来异步写，提升并发性。MQ单机抗几万并发也是ok的。

8.使用线程池的优势

（1）线程和任务分离，提升线程的重用性。

（2）控制线程的并发数量，降低服务器压力，统一管理所有的线程。

（3）提升系统的响应速度，假如创建线程用的时间为T1，执行任务用的时间为T2，销毁线程用的时间为T3，那么使用线程池就免去了T1和T3的时间。

9.线程池的七大参数：

public ThreadPoolExecutor(int corePoolSize,

int maximumPoolSize,

long keepAliveTime,

TimeUnit unit,

BlockingQueue workQueue,

ThreadFactory threadFactory,

RejectedExecutionHandler handler)

七、2022年5月6日

1.spring的一些注解

a.@ComponentScan的作用呢？

告诉Spring哪个pacakge的用注解标识的类会被spring自动扫描并且装配bean容器，param即用来指定扫描包的范围。

b.@Bean和@AutoWired的区别？

@Bean的注解就是替代我们之前配置spring的xml文件里面的

@Autowired修饰变量和方法，自动装配

@Autowired同样也是跟上面这些搭配，但是他的作用是“自动装配”，比如说我们controller层调用service层，首先这个service要注册过或者被@Service注解修饰过，容器里面有才行，要不然没办法去装配。（所以我们使用SpringBoot构建项目的时候用的都是@Autowired,@Bean人家已经帮我们完成了）

c.自动装配的原理

d.熔断的注解

@HystrixCommand注解

@HystrixProperty

e.AOP的注解

@Aspect

@Before

@After

@Around

@Pointcut

@AfterReturning

@AfterThrowing

AOP注解

八、2020/05/06

1.

空间复杂度

时间复杂度

2.sql优化

3.线程池使用

2020/05/07

八、金正睿服面试：

1.先谈感受：主要问的是，项目中遇到的问题，怎么解决？没有实际处理过，很难回答

主要问的题目是：

a.生产者发送一个消息到broker中，如果丢失了，怎么处理？

b.事务隔离4种级别

c.断路器的作用

d.发邮件

九、2020/05/09

事务传播的特性：

事务的传播特性是Spring的事务应用策略。

事务的七大传播特性

（1）PROPAGATION\_REQUIRED

支持当前事务，如果当前没有事务，就新建一个事务。这是最常见的选择。

（2）PROPAGATION\_SUPPORTS

支持当前事务，如果当前没有事务，就以非事务方式执行。

（3）PROPAGATION\_MANDATORY

支持当前事务，如果当前没有事务，就抛出异常。

（4）PROPAGATION\_REQUIRES\_NEW

新建事务，如果当前存在事务，把当前事务挂起。

（5）PROPAGATION\_NOT\_SUPPORTED

以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。

（6）PROPAGATION\_NEVER

以非事务方式执行，如果当前存在事务，则抛出异常。

（7）PROPAGATION\_NESTED

如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则新建一个事务。

Spring事务

十、2020/5/22

1.文件上传流的过程

2.springMVC的执行流程

3.mybits的一个调用过程

4.springBoot和springCloud的区别

5.支付有微信，支付宝，网银，怎么实现呢？

十一、深圳中兴网信科技有限公司

2022/5/13

1.消息中间件

2.K8S

3.消息积压处理：

（1）、发送端优化，增加批量和线程并发两种方式处理

（2）、消费端优化，优化业务逻辑代码、水平扩容增加并发并同步扩容分区数量

大部分MQ内置监控功能，通过监控数据，很容易确定哪种原因。若单位时间发送消息增多，如秒杀，短时内不大可能优化消费端代码以提升性能，只能通过扩容消费端实例数提升总体消费能力。

若短时内无足够服务器资源扩容，那就将系统降级，关闭一些不重要业务，减少发送方发送的数据量，最低限度让系统还能正常运转，服务一些核心业务。

十二、2022/5/16

1.如何查询系统慢查询

MySQL通过慢查询⽇志定位那些执⾏效率较低的SQL 语句.

2.内存溢出怎么查？及怎么解决

a.用Jprofiler、 Memory Analyzer (MAT)、jvisualvm(jdk自带)分析

根据dump文件进行装入 分析占比较大的类

内存溢出是指应⽤系统中存在⽆法回收的内存或使⽤的内存过多，最终使得程序运⾏要⽤到的内存⼤于虚拟机能提供的最⼤内存。

引起内存溢出的原因有很多种，常见的有以下⼏种：

1.内存中加载的数据量过于庞⼤，如⼀次从数据库取出过多数据；

2.集合类中有对对象的引⽤，使⽤完后未清空，使得JVM不能回收；

3.代码中存在死循环或循环产⽣过多重复的对象实体；

4.使⽤的第三⽅软件中的BUG；

5.启动参数内存值设定的过⼩；

内存溢出的解决⽅案：

第⼀步，修改JVM启动参数，直接增加内存。(-Xms，-Xmx参数⼀定不要忘记加。)

第⼆步，检查错误⽇志，查看“OutOfMemory”错误前是否有其它异常或错误。

第三步，对代码进⾏⾛查和分析，找出可能发⽣内存溢出的位置。

3.oracle优化

（1）.选取最适用的字段属性

(表字段的宽度设得尽可能小)，应该尽量把字段设置为NOTNULL

（2）、使用连接（JOIN）来代替子查询(Sub-Queries)，（子查询会产生临时表）

（3）、使用索引

一般说来，索引应建立在那些将用于JOIN,WHERE判断和ORDERBY排序的字段上。尽量不要对数据库中某个含有大量重复的值的字段建立索引

（4）.复合索引

如果我们创建了(area, age, salary)的复合索引，那么其实相当于创建了(area,age,salary)、(area,age)、(area)三个索引，这被称为最佳左前缀特性。因此我们在创建复合索引时应该将最常用作限制条件的列放在最左边，依次递减。

（5）.使用短索引

不使用NOT IN和<>操作

（6）.优化的查询语句

首先，最好是在相同类型的字段间进行比较的操作。

其次，在建有索引的字段上尽量不要使用函数进行操作。

第三，在搜索字符型字段时，我们有时会使用LIKE关键字和通配符，这种做法虽然简单，但却也是以牺牲系统性能为代价的。

用union all 代替union

（7）.任何地方都不要使用 select \* from t ，用具体的字段列表代替“\*”，不要返回用不到的任何字段。

（8）分表（重点）

单张表的访问及写入很频繁，这时候就可以根据一定的业务规则来分表，这样就能减轻单表压力

4.分区、分库，分表的瓶颈在哪里呢？

分库

IO瓶颈：

第一种: 磁盘读的IO，数据库缓存放不下，每次查询是产生大量IO，降低查询速度。

第二种：网络IO，请求数据太多，带宽不够。

CPU瓶颈：

第一种：sql语句，SQL语句中包含函数（join，group by，count…），非索引字段条件查询等等，增加CPU运算操作。

第二种: 单表数据量太大，查询是扫描的数据太多，SQL执行效率低下，CPU瓶颈就出现了。

(1)、水平分库（按关键字取模）

(2)、垂直分库

（3）、分库分表带来的复杂性

a.跨库关联查询

b.分布式ID

c.排序、分页、函数计算问题)

分层

6.索引失效：

（1）查询条件中有or,即使有部分条件带索引也会失效

（2）like查询是以%开头

（3）如果列类型是字符串，那在查询条件中需要将数据用引号引用起来，否则不走索引

（4）索引列上参与计算会导致索引失效

（5）.违背最左匹配原则

十三、

2022/5/16（腾祥公司）

1.分面式事务是怎么实现的？

2.enurka集群是怎么相互注册的？

十四、

2022/5/18

（1）.java虚拟机

（2）.消息中间件，消费端出问题怎么办呢？

（3）.联合索引查询，是不用回表查询的。如果根据主键查询，是需要回表查询的。

（4）.分布式事务

5.mybits一级缓存和二级缓存指的是什么呢？

答：一级缓存作用域是sqlsession级别的，同一个sqlsession中执行相同的sql查询（相同的sql和参数），第一次会去查询数据库并写到缓存中，第二次从一级缓存中取。

二级缓存,它指的是Mybatis中SqlSessionFactory对象的缓存。由同一个SqlSessionFactory对象创建的SqlSession共享其缓存。

在关闭sqlsession后(close)，才会把该sqlsession一级缓存中的数据添加到namespace的二级缓存中。

当 Mybatis 调用 Dao 层查询数据库时，先查询二级缓存，二级缓存中无对应数据，再去查询一级缓存，一级缓存中也没有，最后去数据库查找。

6.数据库优化

7.消息过期了，怎么办呢？

8.rocketMQ和kafka有什么区别呢？

十五、天阳科技：

关注点：需要关注下： 多线程、分布式、sql优化、docker，rocketmq，性能优化的，分库分表，jvm调优的， 自己要熟悉和总结下，结合自己项目。

（1）线程和进程的区别：

线程是程序运行的基本单位，进程是资源分配的最小单位。

（2）并发

为什么要使用多线程呢?

充分利用CPU资源，提高程序的执行效率和运行速度

（3）使用多线程可能带来什么问题?

内存泄露，死锁，线程不安全

（4）创建线程有哪几种方式？

a.继承 Thread 类;b.实现 Runnable 接口;c. 实现Callable接口;d. 使用线程池。

（5）. 什么是上下文切换?

任务从保存到再加载的过程就是一次上下文切换。

（6）什么是线程死锁?

多个线程被阻塞，并且相互等待某一个资源被释放。

（7）说说 sleep() 方法和 wait() 方法区别和共同点?

sleep 方法没有释放锁，而 wait 方法释放了锁 。

（8）说一说对于 synchronized 关键字的了解？

synchronized 关键字解决的是多个线程之间访问资源的同步性，synchronized关键字可以保证被它修饰的方法或者代码块在任意时刻只能有一个线程执行。

（9）volatile ：

当一个变量对共享变量进行了修改，那么另外的线程是立即可以看到修改后的最新值。

volatile:保证共享变量的可见性

（10）ThreadLocal

此类主要解决的是让每个线程绑定自己的值，可以将ThreadLocal类形象比喻为存放数据的盒子，盒子可以存储每个线程的私有数据。避免线程安全

（11）使用线程池的好处

降低资源消耗

提高响应速度

提高线程可管理性

（12）.执行 execute()方法和 submit()方法的区别是什么呢？

execute()方法用于提交不需要返回值的任务，所以无法判断任务是否被线程池执行成功与否；

submit()方法用于提交需要返回值的任务。线程池会返回一个 Future 类型的对象，通过这个 Future 对象可以判断任务是否执行成功，

（13）.ThreadPoolExecutor 七大参数

corePoolSize

maximumPoolSize

workQueue

keepAliveTime

TimeUnit

threadFactory

handler

(14).HashMap底层原理:

1、map.put(k,v)实现原理

（1）⾸先将k,v封装到Node对象当中（节点）。

（2）然后它的底层会调⽤K的hashCode()⽅法得出hash值。

（3）通过哈希表函数/哈希算法，将hash值转换成数组的下标，下标位置上如果没有任何元素，就把Node添加到这个位置上。如果说

下标对应的位置上有链表。此时，就会拿着k和链表上每个节点的k进⾏equal。如果所有的equals⽅法返回都是false，那么这个新的

节点将被添加到链表的末尾。如其中有⼀个equals返回了true，那么这个节点的value将会被覆盖。

2、map.get(k)实现原理

(1)先调⽤k的hashCode()⽅法得出哈希值，并通过哈希算法转换成数组的下标。

(2)通过上⼀步哈希算法转换成数组的下标之后，在通过数组下标快速定位到某个位置上。如果这个位置上什么都没有，则返回null。如

果这个位置上有单向链表，那么它就会拿着K和单向链表上的每⼀个节点的K进⾏equals，如果所有equals⽅法都返回false，则get⽅

法返回null。如果其中⼀个节点的K和参数K进⾏equals返回true，那么此时该节点的value就是我们要找的value了，get⽅法最终返回

这个要找的value。

2022/5/19

(1)JAVA集合有哪些特性？

List: 有顺序，可重复

Set:无序的，不允许重复

Map:无序的，键是唯一的，它的值允许重复

(2).分布式锁

(3)基于Redis的实现方式

a.获取锁的时候，使用setnx加锁，并使用expire命令为锁添加一个超时时间，超过该时间则自动释放锁，锁的value值为一个随机生成的UUID，通过此在释放锁的时候进行判断。

b.获取锁的时候还设置一个获取的超时时间，若超过这个时间则放弃获取锁。

c.释放锁的时候，通过UUID判断是不是该锁，若是该锁，则执行delete进行锁释放。

d.使用命令介绍：

i.SETNX

SETNX key val：当且仅当key不存在时，set一个key为val的字符串，返回1；若key存在，则什么都不做，返回0。

ii.expire

expire key timeout：为key设置一个超时时间，单位为second，超过这个时间锁会自动释放，避免死锁。

iii.delete

delete key：删除key

(4).在使用Redis实现分布式锁的时候，主要就会使用到这三个命令。

基于数据库的实现方式:

基于数据库的实现方式的核心思想是：在数据库中创建一个表，表中包含方法名等字段，并在方法名字段上创建唯一索引，想要执行某个方法，就使用这个方法名向表中插入数据，成功插入则获取锁，执行完成后删除对应的行数据释放锁。

synchronized 可以对方法和语句块进行修饰。从而实现同一时刻只有一个线程能够执行。

volatile可以对变量进行修饰。保证线程在每次使用变量的时候，都会读取变量修改后的最的值。

（5）、多线程同步有哪几种方法？

Synchronized关键字，Lock锁实现，分布式锁等。

(6)redis集群

Btree和B+Tree有什么区别？

（7）java中关于锁的关键字有2个 ， synchronized 和volatile，LOCK

（8）悲观锁和乐观锁

a.悲观锁:,就是很悲观,每次去拿数据的时候都认为别人会修改,所以每次在拿数据的时候都会上锁,这样想拿这个数据就会block直到它拿到锁。

b.乐观锁:,就是很乐观,每次去拿数据的时候都认为别人不会修改,所以不会上锁,但是在更新的时候会判断一下在此期间别人有没有去更新这个数据,可以使用版本号等机制。

生产者-消费者模式的简介：

在实际的软件开发过程中，我们将产生数据的模块称为生产者，处理数据的模块成为消费者。但仅有这两者还不够成为一个生产者-消费者模式，还需要有一个缓冲区（一段内存区域）作为中介，生产者产生的数据放入缓冲区，消费者从缓冲区读取数据并处理。（注：上述所说的模块是广义的，可以是类，函数，线程，进程等）

生产者-消费者模式之间的三种关系：

模拟实现生产者-消费者模式之前，我们需要先捋清除这之间的关系：

生产者-生产者：很明显，这两者之间必定是一种竞争关系，也就是说一个生产者往缓冲区放数据时另一个生产者就不能去访问这块空间

消费者-消费者：同样，两个消费者之间也是竞争的关系，这就好比两个人同时看中一件商品时，他们之间就是一种竞争的关系

生产者-消费者：生产者与消费者之间其实是一种同步与互斥的关系，假设只有一个生产者一个消费者时，只有生产者放入数据后消费者才能读取，消费者拿到数据后生产者才去生产，这就是一种同步；但当生产者生产数据的时候消费者就不能从缓冲区拿数据，或者消费者读数据的时候生产者就不能往缓冲区里写数据，否则很可能会导致两者都存/取数据失败，产生二义性问题。

消息队列如何保证消息丢失？

（1）生产者发送消息不丢失

消息发送+回调，事务机制

（2）MQ主从消息同步不丢失

同步同步，Dledger集群，两阶段提交

（3）MQ消息存盘不丢失

同步刷盘，异步刷盘，同步刷盘消息不会丢失，效率会很低，异步刷盘消息会丢失，效率会很高

（4）MQ消费消息不丢失，采用同步消费方式

在消息端增加幂等性校验，只有消费完了，才通知broker，删除消息。

rocketMq:采用事务机制保证消失不丢失，

（5）消息队列有哪些作用？

解藕，异步，剥峰平谷

（6）.保证消息不重复消费？

1.ACK机制

2.幂等

公司1：天阳科技

（1）.有没有自己写过注解，怎么样自定义注解？

公司2：中软（中信银行）

（3）redis面试题

（4）.哨兵，集群模式

哨兵模式：

通过发送命令，让Redis服务器返回监控其运行状态，包括主服务器和从服务器。

当哨兵监听到Redis主机宕机，会自动将slave切换成master模式，然后通过发布订阅模式，通知其他服务器，修改配置文件，让他们换主机。

公司3：龙博通招聘HR

（1）@resource和@Autowired有什么区别？

（2）@resource是JAVA的注解

可以按类型和按名称进行注入

@Autowired是spring的注解

只能按类型注入

@resource是JAVA注解

公司4：2022.5.26 绛门科技面试

（1）.spring boot的启动原理

（2）.rocketMQ,怎么保证消息的顺序

mq的topic下会存在多个queue，要保证消息的顺序存储，同一个业务编号的消息需要被发送到一个queue中。对应到mq中，需要使用MessageQueueSelector来选择要发送的queue，即对业务编号进行hash，然后根据队列数量对hash值取余，将消息发送到一个queue中

（3）.spring怎么避免循环依赖

@lazy注解

（1）三个map

第一个map:完整生命周期的bean对象

第二个map:不完整生命周期的bean对象，保证单例

第二个map:打破循环依赖 LAMBD表达式（BEAN普通对象(无参构参对象)）AOP

CreateingSet:判断是否在循环依赖中

earlyProxyReference:是否提前进行AOP

（2）

@LAZY:直接生成代理对象（有参构造方法）

（4）.spring的生命周期

Spring Bean的生命周期只有四个阶段：实例化 Instantiation --> 属性填充 Populate --> 初始化 Initialization -->AOP----🡪 销毁 Destruction

公司5：长亮科技HR

（1）bean是如何创建？

（2）跨域是怎么解决的？

第一种：添加@CrossOrigin注解

第二种：添加CORS过滤器

第三种：实现WebMvcConfigurer，重写addCorsMappings方法

（3）.spring mvc 的工作流程：

a、用户发送请求至前端控制器DispatcherServlet。

b、DispatcherServlet收到请求调用HandlerMapping处理器映射器。

c、处理器映射器找到具体的处理器(可以根据xml配置、注解进行查找)，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。

d、 DispatcherServlet调用HandlerAdapter处理器适配器。

d、HandlerAdapter经过适配调用具体的处理器(Controller，也叫后端控制器)。

e、Controller执行完成返回ModelAndView。

f、HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet。

g、DispatcherServlet将ModelAndView传给ViewReslover视图解析器。

h、ViewReslover解析后返回具体View。

h、DispatcherServlet根据View进行渲染视图（即将模型数据填充至视图中）。

i、DispatcherServlet响应用户。

(3).创建线程池的四种方式

1）newCachedThreadPool 创建一个可缓存的线程池，如果线程池长度超过处理需求，可灵活回收空闲线程，若无可回收，则新建线程。

2）newFixedThreadPool 创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待

3）newScheduledThreadPool 创建一个定长线程池，支持定时及周期性任务执行

4）newSingleThreadExecutor 创建一个单线程化的线程池，它只会唯一的工作线程来执行任务，保证所有任务按照指定顺序（FIFO，LIFO，优先级）执行

（4）synchronized可以作用在方法和对象上，有什么区别？

（5）.spring boot启动原理

（6）.java 代理有三种角色

(7).如何让LIKE前置%也走索引

其实就是利用覆盖索引

（8）.prepareStatement和prepareStatement有什么区别？

答：1）.都是接口，但PrepareStatement 继承了Statement。

2）.prepareStatement使用的是占用符，而Statement只能执行静态sql

公司5:法本（三一重工1）

(1).spring boot与spring mvc有什么不同呢？

spring boot主要用于提升项目搭建

不同：A.以前web应用要使用到tomat服务器启动，而springboot内置服务器容器，通过@SpringBootApplication中注解类中main函数启动即可。

B.提供生产指标,健壮检查和外部化配置, 之前spring 是没有的

(2).一个spring boot主要的配置文件有哪些呢？

bootstrap.properties和application.properties

属于spring-cloud的一个环境配置

(3).法本（三一重工2）

1)如何避免死锁？

2)死锁的原因：

互斥使用，占有且等待，不可抢占，循环等待。

占有且等待：一次申请所用资源，就可以不用再等待了

不可抢占：如果某一个线程，获取不到资源。那么必须释放它占有的所有资源。（不可抢占）

循环等待：按序申请资源，先申请序号小的资源，再申请序号大的资源

3).什么是redis?

第一，它是一个高性能基于key\_value的高性能NO\_SQL开源数据库。

第二、目前绝大部分公司都采用redis来实现分布式缓存。从而去提高检索数据的一个效率。

它有以下特点：

基于内存存储： 在进行数据IO操作时，能够达到一个非常高的QPS .官方提供的是10W

二、它提供丰富的数据结构string,list,hash,set,zset

三，它底层采用的是单线程实现数据IO,在数据算法层面并不需要考虑并发安全性，从而导致底层算法时间复杂度基本上都是常量复杂度。

第四、redis虽然它是内存存储，但它可以支持持久化。避免因服务器故障，而造成数据丢失的问题。

最后基于这些特点，redis一般用来实现分布式缓存，从而去降低应用程序对关系型数据库检索带来的影响。除此之外，redis还可以实现分布式锁，分布式队列，排行榜，查找附近的人等功能。为复杂应用提供非常方便和成熟的解决方案

(4).设计模式

[面向对象编程中，都有哪些设计原则](https://blog.csdn.net/zhaohongfei_358/article/details/115085887#_46)

* [开闭原则](https://blog.csdn.net/zhaohongfei_358/article/details/115085887#_47)

对扩展开放，对修改关闭。就是如果修改原有的功能或者是扩展功能，尽量去扩展原有的功能，而不是修改原有的代码

* 里氏替换原则

任务子类对象都应该可以替换其派生的超类对象。即子类可以扩展父类的功能，但不要修改父类的功能。

* 依赖转置（依赖倒置）原则

面向接口编程，而不是面向实现编程

* 单一职责原则

一个对象要专注一件事情，不要让它担任太多责任

* 接口隔离原则

一个接口尽量只包含用户关心的内容。就是一个接口不要太庞大。

* 迪米特法则

两件软件之间不是特别必要，尽量不要让他们直接通信。而是找一个第三方进行转发，比如使用MQ(消息队列)

SIE:

SPRING BOOT过滤器

SPRING BOOT 全局异常

（1）.对异常进行拦截

@ControllerAdvice：对异常拦截

2）.对异常类型进行处理

@ExceptionHandler(value =Exception.class)

@ControllerAdvice

public class MyExceptionHandler {

@ExceptionHandler(value =Exception.class)

public String exceptionHandler(Exception e){

System.out.println("未知异常！原因是:"+e);

return e.getMessage();

}

}

@DeleteMapping("/user")

public boolean delete(@RequestBody User user) {

System.out.println("开始删除...");

//这里故意造成一个异常，并且不进行处理

Integer.parseInt("abc123");

return true;

}

SPRING MVC和SPRING BOOT 的区别

2022.6.3：中金电信面试题

1.LOCK和synclize的区别？

答：

1).synchronized是一个关键字而lock是一个接口（lock、lockInterruptibly、tryLock、unlock、newCondition）。

2).synchronized是隐式的加锁，lock是显示的加锁。

3).synchronized可以作用在方法和代码块上，而lock只能作用在代码块上。

synchronized作用在静态方法上锁的是当前类的class，作用在普通方法上锁的是当前类的对象。

在javap反编译成字节码后，synchronized关键字需要有一个代码块进入的点monitorenter，代码块退出和代码块异常的出口点monitorexit。

4）.synchronized是阻塞式加锁，而lock中的trylock支持非阻塞式加锁。

5）.synchronized没有超时机制，而lock中的trylcok可以支持超时机制。

6）.synchronized不可中断，而lock中的lockInterruptibly可中断的获取锁。（ReentrantLock.lockInterruptibly允许在等待时由其它线程调用等待线程的Thread.interrupt方法来中断等待线程的等待而直接返回，这时不用获取锁，而会抛出一个InterruptedException。 ReentrantLock.lock方法不允许Thread.interrupt中断,即使检测到Thread.isInterrupted,一样会继续尝试获取锁，失败则继续休眠。只是在最后获取锁成功后再把当前线程置为interrupted状态,然后再中断线程。）

7）.synchronized采用的是monitor对象监视器，lock的底层原理是AQS

8）.synchronized只有一个同步队列和一个等待队列，而lock有一个同步队列，可以有多个等待队列。

同步队列：排队取锁的线程所在的队列。

等待队列：调用 wait 方法后，线程会从同步队列转移到等待队列。

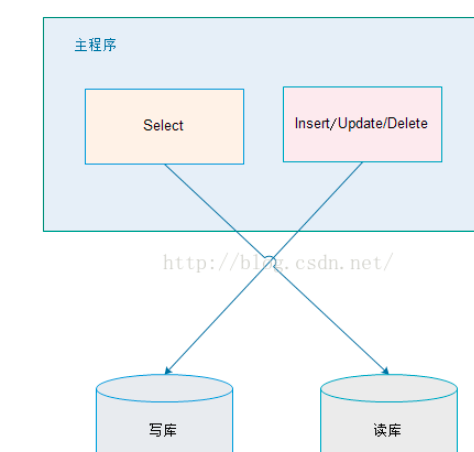
9）.synchronized是非公平锁，而lock可以是公平锁也可以是非公平锁。

10）.synchronized用object的notify方法进行唤醒，而lock用condition进行唤醒。

11）.lock有ReadWriteLock支持并发读。

2.项目总结

3.读写分离用应用层怎么实现？



优点：

1、多数据源切换方便，由程序自动完成；

2、不需要引入中间件；

3、理论上支持任何数据库；

缺点：

1、由程序员完成，运维参与不到；

2、不能做到动态增加数据源；

## 2.2. 中间件解决

4.redis击穿，怎么预防？

5.缓存雪崩

6.缓存穿透

答：既使REDIS,查不到。也可以放一个默认值（如空字符串）。为了防止REDIS中有大批量的空字符串，可以给REDIS设置一个过期时间。

当REDIS查到数据，也设置一个过期时间。为了前面的过期时间失效，去查数据库。

5.JVM内存模型

轻松科技

1. SPRING BOOT启动原理
2. 读配置文件内容方法
3. HASHMAP原理
4. 部署

2022-6-9

你在项目中遇到最大的难题是什么？

数据同步通用框架

公司考察性能问题（智能推荐）

2020 /9/9

1. 华炎科技有限公司（外派）

（1）MYBITS分页是怎么分页的？

答：MYBITS分页查询使用RowBounds

A.接口：

List<User> getUserByRowBounds(RowBounds rowBounds);

B.mapper.xml

<select id="getUserByRowBounds" resultType="com.hzy.pojo.User">

select \* from user

</select>

TEST.JAVA

RowBounds rowBounds = new RowBounds(1,2);

List<User> userByRowBounds = userMapper.getUserByRowBounds(rowBounds);

# （2）设计原则的分类：

单一职责原则：一个类应该只做一个职责相关的业务

开闭原则 ：一个软件实体应该对扩展开放，对修改关闭

里式替换原则：

聚合/组合原则:

接口隔离原则：

迪米特法则：

依赖倒置原则：

汉得（得逸）

1. Ribbon是客户端还是服务端负载均衡呢？

答：是客户端

2.负载均衡有哪些算法呢？

轮询法、随机法、源地址哈希法、加权轮询法、加权随机法、最小连接数法。

FEIGIN里面含有ROBIN

1.服务降级：不管在什么情况下，服务降级的流程都是先调用正常的方法，再调用fallback的方法。 也就是服务器繁忙，请稍后再试，不让客户端等待并立刻返回一个友好提示。

1. 服务熔断：假设服务宕机或者在单位时间内调用服务失败的次数过多，即服务降级的次数太多，那么则服务熔断。 并且熔断以后会跳过正常的方法，会直接调用fallback方法，即所谓“服务熔断后不可用”。 类似于家里常见的保险丝，当达到最大服务访问后，会直接拒绝访问，拉闸限电，然后调用服务降级的fallback方法，返回友好提示。

ZUUL:权限校验，服务的转发

网新软件

6到9月干嘛去了？

照顾爸妈身体

订菜系统的一些情况:

订菜系统现在有在使用吗？

没有，做得不美观。

数据库的优化

你换了几家公司？答：我换了4、5家。

2022/9/23

中软国际（平安银行）

1. 有没有使用分布式事务
2. 数据库优化
3. 设计模式又没有在项目中用过，有用过哪些？

工厂模式，代理模式（AOP），单例模式

ThreadPool 就是单例模式

代理模式（AOP）:记录日志，事务管理，监控性能，缓存优化

工厂模式：MYSQL shujuku, PG KU, ORACLE , 根据传参，获取不同的链接对象

1. 线程池业务场景和线程池的参数和流程
2. 消息中间件

2022/9/25

艾融科技

1. SPRING BOOT 注解有哪些？

@CONFIGRATION,@COMPONENT SCAN,@ENABLEAUTOCONFIGRATION

1. SPRING 事务传播有哪几种

A.PROPAGATION\_REQUIRED

B.PROPAGATION\_REQUIRES\_NEW

C.PROPAGATION\_SUPPORTS

D.PROPAGATION\_NOT\_SUPPORTED

E.PROPAGATION\_MANDATORY

F.PROPAGATION\_NEVER

G.PROPAGATION\_NESTED

1. 如何实现表单防重复提交？

答：1、前端防抖

答：用户提交后，把按钮改为不可用。

2、token机制

例如订单提交场景：

当用户跳到订单提交页面时，后台会生成一个TOKEN,先存入到REDIS,同时又返回给前端。当用户点提交按钮时，前端会传一个TOKEN会与REDIS中的TOKEN进行比较，如果相同，则允许提交，如果过期或不同，那么不允许提交。

如果多线程请求持有相同的TOKEN，来进行提交。那么使用分布锁和LUA脚本。从REDIS获取令牌，对比令牌，生成订单号，删除令牌。不可分隔，保证4步原子性，防止另一线程操作同一订单。

3、幂等

支付幂等，通过订单号，防止同一订单支付多次。

使用分布锁，保证同一订单，同一时间只有一个服务在处理。

验证订单状态，防止重复支付。支付订单只一进入后端，都要改为一个支付中的中间状态，有了中间状态。及时后续没完成，我们也可以防止重复支付。

用户支付完成，则会通知第三方支付。一般来说，第三方支付处理成功后，会给我们发送消息的回调。接受到，会进行整体状态的变更，还有一种情况，如果第三方没有给我们回调信息，或者各种异常导致我们没有收到，我们则要提供主动查询逻辑，去第三方查询当前订单的支付状态，防止卡单。主动查询可以做成异步扫描机制，防止前端用户等太久。

1. 最近的一个项目是什么呢？
2. 做过JAVA组长吗？

做过

1. 答应过的面试，不重视

面试官评价：

主要是表达方面的，技术问题倒不大