最新百度、阿里、字节 Java 面试题集

一、 百度篇.....7

1.JAVA 中的几种基本数据类型是什么，各自占用多少字节。.....7

2.String 类能被继承吗，为什么。.....7

3.String，Stringbuffer，StringBuilder 的区别。.....7

4.ArrayList 和 LinkedList 有什么区别。.....7

5.讲讲类的实例化顺序，比如父类静态数据，构造函数，字段，子类静态数据，构造函数，字段，当 new 的时候，他们的执行顺序。.....7

6.sql 优化有哪些?如何创建索引?创建索引的原则?索引的优缺点?.....7

7.sql 如何去重?.....10

8.内连接和外连接的区别.....10

9.sql 语句关键词的执行顺序?.....11

10.什么情况下会发生栈内存溢出。.....11

11.JVM 的内存结构，Eden 和 Survivor 比例。.....12

12.JVM 内存为什么要分成新生代，老年代，持久代。新生代中为什么要分为 Eden 和 Survivor。.....12

13.JVM 中一次完整的 GC 流程是怎样的，对象如何晋升到老年代，说说你知道的几种主要的 JVM 参数。.....12

14.Java 中如何使用 redis？redis 支持的数据类型及各种数据类型的使用场景？redis 如何解决数据过期?.....12

15.存储过程的了解和使用？ 13

16.消息队列的使用场景。 14

17.消息的重发，补充策略。 14

18.如何保证消息的有序性。 14

用过哪些 MQ，和其他 mq 比较有什么优缺点，MQ 的连接是线程安全的吗， . 14

19.数据库表的设计注意事项有哪些？三大范式的了解？ 14

20.百万级量的数据分页查询如何优化？ 16

21.数据库的乐观锁和悲观锁的理解和使用？ 18

22.数据库如何实现分页？ 18

23.#{}和${}的区别是什么？ 19

24.数据库中字符串和日期的相互转换？ 19

25.MySQL 的存储引擎有哪些？ 20

26.事务隔离级别有哪些？mysql 和 oracle 默认的隔离级别是什么？ 20

27.SQL 如何行转列和列转行？ 21

28.如何查看 sql 的执行计划？ 22

29.union 和 unionAll 区别？ 22

31.oracle 中的分析函数有哪些？ 22

32.数据库中除了聚合函数之外还有哪些常用的函数？oracle 数据库的 merge（）

函数的作用和使用？ 23

33.MySQL 中如何忽略表名的大小写？ 25

34.Aspect 切面 25

35.常见的高并发场景有哪些，对应的架构设计方案是什么。 26

36.介绍完整的分布式中间件有哪些，各自的应用场景和作用。 26

37.双 11 秒杀活动，你的技术架构设计思路。 26

38.用 java 自己实现一个 LRU。 26

二、阿里篇 26

1.讲讲类的实例化顺序，比如父类静态数据，构造函数，字段，子类静态数据，构造函数，字段，当 new 的时候，他们的执行顺序。 26

2.用过哪些 Map 类，都有什么区别，HashMap 是线程安全的吗,并发下使用的 Map 是什么，他们内部原理分别是什么，比如存储方式，hashcode，扩容，默认容量等。 26

3.JAVA8 的 ConcurrentHashMap 为什么放弃了分段锁，有什么问题吗，如果你来设计，你如何设计。 26

4.有没有有顺序的 Map 实现类，如果有，他们是怎么保证有序的。 26

5.having 和 where 的区别？ 26

6.游标的作用和使用？ 27

7.如何使用数据库中的定时器？触发器？定时任务？Oracle 中如何实现递归查询？ 28

8.高并发下如何保证修改数据安全 30

9.MyBatis 实现一对一有几种方式?具体怎么操作的？ 31

10. linux 命令，如何改文件权限。 31

11.数组、链表（单向、双向、双端）、栈和队列、二叉树、红黑树、哈希表、堆（最大和最小） 31

12. 个人经验：栈和队列、哈希表、链表、二叉树的题较多，图的较少 31

13. 查找：二分查找及其变形.....31

14. 二叉树：前序、中序、后序遍历，按规定方式打印，两个节点之间操作（最近公共祖先、距离）等问题。.....32

15.Mybatis 是如何进行分页的？分页插件的原理是什么？.....32

16.Delete 误删数据没有备份怎么恢复？.....32

17.Mybatis 是否支持延迟加载？如果支持，它的实现原理是什么？.....33

18.什么是 MyBatis 的接口绑定？有哪些实现方式？.....33

19.使用 MyBatis 的 mapper 接口调用时有哪些要求？.....34

20. 介绍下你理解的操作系统中线程切换过程。.....34

21.. 进程和线程的区别。.....34

22. top 命令之后有哪些内容，有什么作用。.....34

23. 线上 CPU 爆高，请问你如何找到问题所在。.....34

24.什么是 Spring MVC 框架的控制器？.....35

25.数据传输的事务定义有哪三种？.....35

26.谈谈对 Synchronized 关键字，类锁，方法锁，重入锁的理解。.....35

27.volatile 的原理。.....35

28.使用 kafka 有没有遇到什么问题，怎么解决的。.....35

29.MQ 有可能发生重复消费，如何避免，如何做到幂等。.....35

30.MQ 的消息延迟了怎么处理，消息可以设置过期时间么，过期了你们一般怎么处理。.....35

31. Redis 集群方案应该怎么做？都有哪些方案？.....36

32.谈谈你的参与的项目？能否画出整个项目的架构设计图，尽量包含流程、部署

等.....36

33. 一次 web 请求响应中，那个部分最耗时，tcp 握手？业务逻辑处理？网络延迟？数据库查询？浏览器解析？ 34、分布式系统设计你会考虑哪些策略？.....36

35、最常见的数据分布方式是什么？.....36

36、谈一谈一致性哈希算法。.....36

37、paxos 是什么？.....36

38、什么是 Lease 机制？.....36

39、如何理解选主算法？.....36

40、OSI 有哪七层模型？TCP/IP 是哪四层模型。 41 正常情况下，当在 try 块或 catch 块中遇到 return 语句时，finally 语句块在方法返回之前还是之后被执行？36

42、try、catch、finally 语句块的执行顺序。.....36

43、Java 虚拟机中，数据类型可以分为哪几类？.....36

44、怎么理解栈、堆？堆中存什么？栈中存什么？.....36

45、为什么要把堆和栈区分出来呢？栈中不是也可以存储数据吗？.....37

三、字节篇.....37

1.IO 模型有哪些，讲讲你理解的 nio ，他和 bio，aio 的区别是啥，谈谈 reactor 模型。.....37

2.反射的原理，反射创建类实例的三种方式是什么。.....37

3.反射中，Class.forName 和 ClassLoader 区别 。.....37

4.描述动态代理的几种实现方式，分别说出相应的优缺点。.....37

5.动态代理与 cglib 实现的区别。.....37

6.为什么 CGlib 方式可以对接口实现代理。.....37

7.final 的用途。 37

8.简述 Mybatis 的插件运行原理，以及如何编写一个插件。 37

9.ZAB 协议？ 38

10.客户端注册 Watcher 实现 38

11.Chroot 特性 39

12.会话管理 39

13.服务器角色 40

14.数据同步 40

15.如何理解 Spring 中的代理？ 42

16.分布式集群中为什么会有 Master？ 42

17.Zookeeper 对节点的 watch 监听通知是永久的吗？为什么不是永久的？ 43

18.chubby 是什么，和 zookeeper 比你怎么看？ 43

19.说几个 zookeeper 常用的命令。 43

20.ZAB 和 Paxos 算法的联系与区别？ 43

21.服务调用是阻塞的吗？ 44

22.Dubbo 推荐用什么协议？ 44

23.Dubbo Monitor 实现原理？ 45

24.Dubbo 用到哪些设计模式？ 46

25.Dubbo SPI 和 Java SPI 区别？ 48

26.简述 Http 请求 get 和 post 的区别以及数据包格式。 48

27.HTTP 有哪些 method 48

28.简述 HTTP 请求的报文格式。 48

29.HTTP 的长连接是什么意思。 48

30.Thread 类的 sleep()方法和对象的 wait()方法都可以让线程暂停执行，它们有什么区别? 49

31. Java 服务端问题排查（OOM，CPU 高，Load 高，类冲突） 49

32. 如何查看 Java 应用的线程信息 50

33.Redis hash 算法用的是什么？ 50

一、百度篇

1.JAVA 中的几种基本数据类型是什么，各自占用多少字节。

2.String 类能被继承吗，为什么。

3.String，Stringbuffer，StringBuilder 的区别。

4.ArrayList 和 LinkedList 有什么区别。

5.讲讲类的实例化顺序，比如父类静态数据，构造函数，字段，子类静态数据，构造函数，字段，当 new 的时候，他们的执行顺序。

6.sql 优化有哪些?如何创建索引?创建索引的原则?索引的优缺点?

sql 的优化有:

尽量避免使用 select * ，返回无用的字段会降低效率。优化方式：只能使用具体的字段代替 select 具体字段，只返回使用到的字段。

尽量避免使用 in 和 not in，会导致数据库引擎放弃索引进行全表扫描。优化方式：如果是连续数值，可以用 betwween 代替，如果是子查询，可以用 exists 代替。

尽量避免在字段开头模糊查询，会导致数据库引擎放弃索引进行全表扫描。优化方式:尽量在字段后面使用模糊查询。

尽量避免进行 null 值的判断，会导致数据库引擎放弃索引进行全表扫描。优化方式:可以给字段添加默认值 0，对 0 值进行判断。

如何创建索引:

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。

创建索引有两种:

直接创建索引，例如使用 CREATE INDEX 语句或者使用创建索引向导。

间接创建索引，例如在表中定义主键约束或者唯一性键约束时，同时也创建了索引。

索引的作用:

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）。

创建索引可以大大提高系统的性能（优点）：

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。

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。

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。

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。

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):

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。

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。

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。

创建索引的原则：

最左前缀匹配原则（一直向右匹配直到遇到范围查询就停止匹配）；

=和 in 可以乱序（建立索引是可以任意顺序的，mysql 的查询优化器会帮你优化成索引可以识别的形式）；

尽量选择区分度高的列作为索引（区分度的公式是 count(distinct col)/count(*)，表示字段不重复的比例，比例越大我们扫描的记录数越少，唯一键的区分度是 1，）；

索引列不能参与计算（保持列“干净”，）；

尽量的扩展索引，不要新建索引（如 a->（a，b）只需要修改原来的索引）；

选择唯一性索引（唯一性索引的值是唯一的，可以更快速的通过该索引来确定某条记录。）；

为经常需要排序、分组和联合操作的字段建立索引；

为常作为查询条件的字段建立索引；

限制索引的数目；

尽量使用数据量少的索引；

尽量使用前缀来索引；

删除不再使用或者很少使用的索引。

7.sql 如何去重?

总的思路就是先找出表中重复数据中的一条数据，插入临时表中，删除所有的重复数据，然后再将临时表中的数据插入表中。

实现：

重复数据完全一样，使用 distinct；

id 列不同,id 类型为 int,自增字段,使用聚合函数 max 或其他；

id 列不同,id 类型为 uniqueidentifier；使用 row_number() over()和 partition by 给每一组添加行号；将行号=1 的数据插入临时表中。

8.内连接和外连接的区别

自然连接：是一种特殊的等值连接，他要求两个关系表中进行比较的必须是相同的属性列，无须添加连接条件，并且在结果中消除重复的属性列。

sql 语句：Select …… from 表 1 natural join 表 2

内连接：基本与自然连接相同，不同之处在于自然连接要求是同名属性列的比较，而内连接则不要求两属性列同名，可以用 using 或 on 来指定某两列字段相同的连接条件。

sql 语句：Select …… from 表 1 inner join 表 2 on 表 1.A=表 2.E

左外连接：是在两表进行自然连接，左边表数据行全部保留，右边表保留符合连接条件的行。

sql 语句：Select …… from 表 1 left outer join 表 2 on 表 1.C=表 2.C

右外连接：是在两表进行自然连接，右边表数据行全部保留，左边表保留符合连接条件的行。

Select …… from 表 1 rignt outer join 表 2 on 表 1.C=表 2.C

9.sql 语句关键词的执行顺序?

FROM 子句，组装来自不同数据源的数据；

WHERE 子句，基于指定的条件对记录进行筛选

GROUP BY 子句，将数据划分为多个分组

使用聚合函数进行计算

使用 HAVING 子句筛选分组

计算所有的表达式

使用 ORDER BY 对结果集进行排序

即：from—>where—>group by—>having—>计算所有的表达式—>order by—>select 输出

10.什么情况下会发生栈内存溢出。

11.JVM 的内存结构，Eden 和 Survivor 比例。

12.JVM 内存为什么要分成新生代，老年代，持久代。新生代中为什么要分为 Eden 和 Survivor。

13.JVM 中一次完整的 GC 流程是怎样的，对象如何晋升到老年代，说说你知道的几种主要的 JVM 参数。

14.Java 中如何使用 redis？redis 支持的数据类型及各种数据类型的使用场景？redis 如何解决数据过期？

如何使用 redis：

redis 的安装(windows)；启动 redis，默认端口 6379；连接 redis 输入

redis-cli.exe -h 127.0.0.1 -p 6379

java 中利用 jedis 连接 redis

spring 集成 redis：

引入 spring-data-redis.jar 包；

redis.properties 配置文件；

spring-redis 配置文件(此处包含了数据库和 mybatis 的配置)；

redis 工具类(此处 try catch 为防止 redis 宕机等问题时 方法可以继续执行)；

将 spring-redis.xml 包含到 web.xml 中；

在需要 redis 的业务类中注入 redisUtil。

redis 支持的数据类型及各种数据类型的使用场景：

数据类型	可以存储的值	操作	应用场景
STRING	字符串、整数或者浮点数	对整个字符串或者字符串的其中一部分执行操作 对整数和浮点数执行自增或者自减操作	做简单的键值对缓存
LIST	列表	从两端压入或者弹出元素 对单个或者多个元素进行修剪，只保留一个范围内的元素	存储一些列表型的数据结构，类似粉丝列表、文章的评论列表之类的数据
SET	无序集合	添加、获取、移除单个元素 检查一个元素是否存在于集合中 计算交集、并集、差集 从集合里面随机获取元素	交集、并集、差集的操作，比如交集，可以把两个人的粉丝列表整一个交集
HASH	包含键值对的无序散列表	添加、获取、移除单个键值对 获取所有键值对 检查某个键是否存在	结构化的数据，比如一个对象
ZSET	有序集合	添加、获取、删除元素 根据分值范围或者成员来获取元素 计算一个键的排名	去重但可以排序，如获取排名前几名的用户

redis 如何解决数据过期：

Redis 提供了两种的方式，用于删除过期的数据：

定期删除：Redis 默认 100ms 随即抽取部分设置过期时间的 key，过期了就删除。优点是避免长时间的在扫描过期 key，缺点是有些过期 key 无法被删除。

惰性删除：如果查询了某个过期 key，但定期删除没有删除掉，那就将其删除了。key 没过期就正常返回。

15.存储过程的了解和使用?

存储过程：就是作为可执行对象存放在数据库中的一个或多个 SQL 命令。通俗来讲：存储过程其实就是能完成一定操作的一组 SQL 语句。

优点：

存储过程可封装，并隐藏复杂的商业逻辑。

存储过程可以回传值，并可以接受参数。

存储过程无法使用 select 指令来运行，因为它是子程序，与查看表，数据表或用户定义函数不同。

存储过程可以用在数据检验，强制实行商业逻辑等。

缺点：

存储过程，往往定制化于特定的数据库上，因为支持的编程语言不同。当切换到其他厂商的数据库系统时，需要重写原有的存储过程。

存储过程的性能调校和撰写，受限于各种数据库系统。

使用：

创建存储过程保存在数据库的数据字典中。

查询所有存储过程状态

查看对应数据库下所有存储过程状态

mysql 存储过程用 call 和过程名以及一个括号，括号里面根据需要，加入参数，参数包括输入参数、输出参数、输入输出参数调用。

16.消息队列的使用场景。

17.消息的重发，补充策略。

18.如何保证消息的有序性。

用过哪些 MQ，和其他 mq 比较有什么优缺点，MQ 的连接是线程安全的吗，

19.数据库表的设计注意事项有哪些？三大范式的了解？

数据库设计的注意事项：

字段的原子性：保证每列的原子性，不可分解，能用一个字段表达清楚的绝不使用第二个字段。

主键设计：主键不要与业务逻辑有所关联，最好是毫无意义的一串独立不重复的数字。

字段使用次数：对于频繁修改的字段（一般是指状态类字段）最好用独立的数字或者单个字母去表示，不用使用汉字或长字符的英文。

字段长度：建表的时候，字段长度尽量要比实际业务的字段大 3-5 个字段左右，最好是 2 的 n 次方幂值。

关于外键：尽量不要建立外键，保证每个表的独立性。

动静分离：最好做好静态表和动态表的分离。

关于 code 的值：使用数字码或者字母去代替实际的名字，也就是尽量把 name 转换为 code。

关于 null 的值：尽量不要有 null 值，有 null 值的话，数据库在进行索引的时候查询的时间更久，从而浪费更多的时间！可以在建表的时候设置一个默认值！

关于引擎的选择：myisam 的实际查询速度要比 innodb 快，因为它不扫面全表，但是 myisam 不支持事务，没办法保证数据的 Acid。

资源存储：数据库不要存储任何资源文件。

与主键相关：根据数据库设计三大范式，尽量保证列数据和主键直接相关而不是间接相关

关系映射：多对一或者一对多的关系,关联一张表最好通过 id 去建立关系。

预留字段：在设计一张表的时候应该预制一个空白字段，用于以后的扩展。

留下单一字段确定是否可用：通过一个单一字段去控制表是否可用。

删除字段：数据库是禁止使用 delete 命令的,一般都不会真正删除数据，都是采用改状态的方式，设置 state 字段，通过修改状态赋予它是否有效的逻辑含义！

三大范式的了解：

第一范式（1NF）:确保每一列的原子性

数据表中的每一列都是最小的不可分割的单元

第二范式（2NF）：表中的记录是唯一的

表中的数据是可以通过主键来区分的

第三范式（3NF）：表中数据不要有冗余

在一个表中不要出现其他表中除了关键字段（主键）的其他字段，用外键进行关联

20.百万级量的数据分页查询如何优化?

--方法 1: 直接使用数据库提供的 SQL 语句

--语句样式: MySQL 中,可用如下方法: SELECT * FROM 表名称 LIMIT M,N

--适应场景: 适用于数据量较少的情况(元组百/千级)

--原因/缺点: 全表扫描,速度会很慢 且 有的数据库结果集返回不稳定(如某次返回 1,2,3,另外的一次返回 2,1,3). Limit 限制的是从结果集的 M 位置处取出 N 条输出,其余抛弃.

--方法 2: 建立主键或唯一索引, 利用索引(假设每页 10 条)

--语句样式: MySQL 中,可用如下方法: SELECT FROM 表名称 WHERE

id_pk > (pageNum10) LIMIT M

--适应场景: 适用于数据量多的情况(元组数上万)

--原因: 索引扫描,速度会很快. 有朋友提出: 因为数据查询出来并不是按照 pk_id 排序的，所以会有漏掉数据的情况，只能方法 3

--方法 3: 基于索引再排序

--语句样式: MySQL 中,可用如下方法: SELECT FROM 表名称 WHERE id_pk > (pageNum10) ORDER BY id_pk ASC LIMIT M

--适应场景: 适用于数据量多的情况(元组数上万). 最好 ORDER BY 后的列对象是主键或唯一所以,使得 ORDERBY 操作能利用索引被消除但结果集是稳定的(稳定的含义,参见方法 1)

--原因: 索引扫描,速度会很快. 但 MySQL 的排序操作,只有 ASC 没有 DESC(DESC 是假的,未来会做真正的 DESC,期待...).

--方法 4: 基于索引使用 prepare（第一个问号表示 pageNum，第二个？表示每页元组数)

--语句样式: MySQL 中,可用如下方法: PREPARE stmt_name FROM SELECT FROM 表名称 WHERE id_pk > (? ？) ORDER BY id_pk ASC LIMIT M

--适应场景: 大数据量

--原因: 索引扫描,速度会很快. prepare 语句又比一般的查询语句快一点。

--方法 5: 利用 MySQL 支持 ORDER 操作可以利用索引快速定位部分元组,避免全表扫描

比如: 读第 1000 到 1019 行元组(pk 是主键/唯一键).

SELECT * FROM your_table WHERE pk>=1000 ORDER BY pk ASC LIMIT 0,20

--方法 6: 利用"子查询/连接+索引"快速定位元组的位置,然后再读取元组. 道理同方法 5

如(id 是主键/唯一键,蓝色字体时变量):

21.数据库的乐观锁和悲观锁的理解和使用?

悲观锁,就是对数据的冲突采取一种悲观的态度,也就是说假设数据肯定会冲突,所以在数据开始读取的时候就把数据锁定住。(数据锁定:数据将暂时不会得到修改)

乐观锁,认为数据一般情况下不会造成冲突,所以在数据进行提交更新的时候,才会正式对数据的冲突与否进行检测,如果发现冲突了,则让用户返回错误的信息。让用户决定如何去做。

使用:

悲观锁通常依靠数据库提供的锁机制,比如 mysql 的排他锁,select ... for update 来实现悲观锁。

乐观锁不依靠数据库提供的锁机制,需要我们自已实现,实现方式一般是记录数据版本,一种是通过版本号,一种是通过时间戳。

22.数据库如何实现分页?

SQL Server

在分页查询上,我感觉 SQL Server 比较费劲,没有一个专门的分页的语句,靠

的是一种巧妙的方法实现分页查询。

MySQL

MySQL 有个专门针对查询出一段数据的语句 limit，使用起来非常的方便。

Oracle

Oracle 中有个 rownum，其含义更加明显，就是第几行的意思，这样我们就可以通过 where 条件来进行分段查询了。

23.#{}和${}的区别是什么?

#{}是预编译处理，${}是字符串替换。

Mybatis 在处理#{}时，会将 sql 中的#{}替换为?号，调用 PreparedStatement 的 set 方法来赋值；

Mybatis 在处理${}时，就是把${}替换成变量的值。

使用#{}可以有效的防止 SQL 注入，提高系统安全性。

24.数据库中字符串和日期的相互转换?

Oracle

时间转字符串 to_char(date,format)

select to_char(sysdata,'YYYY"年"MM"月"DD"日"') 时间转字符串 from dual

字符串转时间 to_date(str,format)

select to_date('2019-10-25 17:15:20','yyyy-MM-dd HH24:mi:ss') 字符串转时间 from dual

select to_date('2019-10-25 17:15:20','yyyy-MM-dd HH24:mi:ss') 字符串转时间 from dual

MySQL

MySQL 内置函数，在 MySQL 里面利用 str_to_date() 把字符串转换为日期

示例：分隔符一致，年月日要一致

字符串转日期

select str_to_date('2019-10-25 15:43:28','%Y-%m-%d %H:%i:%s');

日期转字符串

select DATE_FORMAT(SYSDATE(),'%Y 年%m 月%d 日') MySQL 日期转字符串 from DUAL;

25.MySQL 的存储引擎有哪些?

InnoDB,MyISAM,Memory,Merge,Archive,Federate,CSV,BLACKHOLE

26.事务隔离级别有哪些? mysql 和 oracle 默认的隔离级别是什么?

隔离级别	脏读	不可重复读	幻读
读未提交（Read uncommitted）	√	√	√
读已提交（Read committed）	×	√	√
可重复读（Repeated read）	×	×	√
可串行化（Serializable）	×	×	×

Oracle数据库只支持Serializable (串行化) 级别和 Read committed (读已提交) 两种级别，默认隔离级别为 Read committed（读已提交）级别；

MySQL数据库中支持上面四种隔离级别，默认隔离级别为Repeatable read (可重复读)。

事务的四大特性：

原子性：事务是最小的执行单位，不允许分割。事务的原子性确保动作要么全部

完成，要么完全不起作用。

一致性：执行事务前后，数据保持一致，多个事务对同一个数据读取的结果是相同的；

隔离性：并发访问数据库时，一个用户的事务不被其他事务所干扰，各并发事务之间数据库是独立的；

持久性：一个事务被提交之后。它对数据库中数据的改变是持久的，即使数据库发生故障也不应该对其有任何影响。

脏读？幻读？不可重复读？

脏读(Drity Read)：某个事务已更新一份数据，另一个事务在此时读取了同一份数据，由于某些原因，前一个 RollBack 了操作，则后一个事务所读取的数据就会是不正确的。

幻读(Phantom Read):在一个事务的两次查询中数据笔数不一致，例如有一个事务查询了几列(Row)数据，而另一个事务却在此时插入了新的几列数据，先前的事务在接下来的查询中，就会发现有几列数据是它先前所没有的。

不可重复读(Non-repeatable read):在一个事务的两次查询之中数据不一致，这可能是两次查询过程中间插入了一个事务更新的原有的数据。

27.SQL 如何行转列和列转行?

行转列：

case when 以及 sum（max 也可以）

列转行：

使用 union 和 case when 以及 concat 来完成

28.如何查看 sql 的执行计划?

set statistics profile on

explain

29.union 和 unionAll 区别?

union 和 union all 的区别是,union 会自动压缩多个结果集合中的重复结果，而 union all 则将所有的结果全部显示出来，不管是不是重复。

Union：对两个结果集进行并集操作，不包括重复行，同时进行默认规则的排序；---在进行表链接后会筛选掉重复的记录，所以在表链接后会对所产生的结果集进行排序运算，删除重复的记录再返回结果

Union All：对两个结果集进行并集操作，包括重复行，不进行排序；---如果返回的两个结果集中有重复的数据，那么返回的结果集就会包含重复的数据了。

30.为什么说 Mybatis 是半自动 ORM 映射工具？它与全自动的区别在哪里?

Hibernate 属于全自动 ORM 映射工具，使用 Hibernate 查询关联对象或者关联集合对象时，可以根据对象关系模型直接获取，所以它是全自动的。而 Mybatis 在查询关联对象或关联集合对象时，需要手动编写 sql 来完成，所以，称之为半自动 ORM 映射工具。

31.oracle 中的分析函数有哪些?

分析函数是 Oracle 专门用于解决复杂报表统计需求的功能强大的函数，它可以在数据中进行分组然后计算基于组的某种统计值，并且每一组的每一行都可以返回一个统计值。

分析函数带有一个开窗函数 over()，包含三个分析子句:分组(partition by)，排序(order by)，窗口(rows)；

first_value()与 last_value()：求最值对应的其他属性

rank(),dense_rank()与 row_number()：求排序

lag()与 lead()：求之前或之后的第 N 行

rollup()与 cube()：排列组合分组

max(),min(),sun()与 avg()：求移动的最值总和与平均值

32.数据库中除了聚合函数之外还有哪些常用的函数？oracle 数据库的 merge（）函数的作用和使用？

常用函数 聚合函数

函数名	作用
AVG()	返回某字段的平均值
COUNT()	返回某字段的行数
MAX()	返回某字段的最大值
MIN()	返回某字段的最小值
SUM()	返回字段的和

数学函数

函数名	作用	举例	结果
ceil(x)	返回大于或等于数值x的最小整数	SELECT CEIL(2.3);	返回：3
floor(x)	返回小于或等于数值x的最大整数	SELECT FLOOR(2.3);	返回:2
round(X)	四舍五入运算法则	SELECT ROUND(-10.53)	返回：-11
rand()	返回0-1之间的随机数	SELECT RAND();	返回：0.8652411800485488
truncate(x,d)	保留小数点后d位，直接截取。	SELECT TRUNCATE(10.05727,2)	结果：10.05
round(x,d)	保留小数点后d位，四舍五入	SELECT ROUND(10.05727,2)	结果：10.06
sign(x)	返回x的符号	SELECT SIGN(-10.53)	结果：-1

字符串函数

函数名	作用	举例	返回结果
concat(str1,str2...)	字符串连接	SELECT CONCAT('my','s','ql');	返回：mysql
insert(str,pos,len,newstr)	字符串替换	SELECT INSERT('这是SQL Server数据库',3,10,'MySQL');	返回：这是MySQL数据库
lower(str)	将字符串转为小写	SELECT LOWER(MySQL);	返回：mysql
upper(str)	将字符串转为大写	SELECT UPPER(MySQL);	返回：MYSQL
substring(str,num,len)	字符串截取	SELECT SUBSTRING('JavaMySQLOracle',5,5);	返回：MySQL
substr(str FROM pos FOR len)	字符串截取	SELECT SUBSTR("JavaMySQLOracle" FROM 5 FOR 5);	返回：MySQL

时间日期函数

函数名	作用	举例	结果
curdate()	获取当前日期	SELECT CURDATE();	返回：2018-08-03
curtime()	获取当前时间	SELECT CURTIME();	返回：10:38:49
now()	获取当前日期和时间	SELECT NOW();	返回：2018-08-03 10:38:49
week(date)	返回日期date为一年	SELECT WEEK(NOW());	返回：30
year(date)	返回日期date的年份	SELECT YEAR(NOW());	返回：2018
hour(time)	返回时间time的小时	SELECT HOUR(NOW());	返回：10
minute(time)	返回时间time的分钟	SELECT MINUTE(NOW());	返回：38
datediff(expr1,expr2)	返回日期参数date1和	SELECT DATEDIFF(CURDATE(),"2089-10-09");	返回：-26000
adddate(date,n)	结算参数date加上n 天后的日期	SELECT ADDDATE(NOW(),100);	返回：2018-11-11 10:56:33

oracle 数据库 merge（）函数的作用和使用：

通常我们对数据库数据进行插入的时候，会判断数据是否已经存在，如果存在就修改，不存在就插入，一般我们会写两个 sql，一个 insert 一个 update,那么其实 oracle 中存在 merge 函数，可以一次性解决这个问题

33.MySQL 中如何忽略表名的大小写?

mysql 在 windows 系统下安装好后，默认是对表名大小写不敏感的。

在 linux 下，一些系统需要手动设置。 用 root 登录，打开并修改 /etc/my.cnf；在[mysqld]节点下，加入一行： lower_case_table_names=1。重启 mysql 服务 systemctl restart mysqld

34.Aspect 切面

AOP 核心就是切面，它将多个类的通用行为封装成可重用的模块，该模块含有一组 API 提供横切功能。比如，一个日志模块可以被称作日志的 AOP 切面。

根据需求的不同，一个应用程序可以有若干切面。在 Spring AOP 中，切面通过带有@Aspect 注解的类实现。

35.常见的高并发场景有哪些，对应的架构设计方案是什么。

36.介绍完整的分布式中间件有哪些，各自的应用场景和作用。

37.双 11 秒杀活动，你的技术架构设计思路。

38.用 java 自己实现一个 LRU。

二、阿里篇

1.讲讲类的实例化顺序，比如父类静态数据，构造函数，字段，子类静态数据，构造函数，字段，当 new 的时候，他们的执行顺序。

2.用过哪些 Map 类，都有什么区别，HashMap 是线程安全的吗,并发下使用的 Map 是什么，他们内部原理分别是什么，比如存储方式，hashcode，扩容，默认容量等。

3.JAVA8 的 ConcurrentHashMap 为什么放弃了分段锁，有什么问题吗，如果你来设计，你如何设计。

4.有没有有顺序的 Map 实现类，如果有，他们是怎么保证有序的。

5.having 和 where 的区别?

Where 过滤子句是在查询过程中对表中数据的过滤条件，不允许使用聚合函数作为过滤条件，原因在于时机不对。聚合函数是对表中数据查询后的结果集进行统计处理的，两者的执行时机不一致。

Having 过滤子句是对查询结果集进行过滤的，可以在该子句后使用聚合函数，因为时机相同，聚合函数是处理结果集的，having 子句又是过滤结果集的，可以在一起使用，另外 having 不能单独使用，只能跟在 group by 分组子句后面使用。

6.游标的作用和使用?

游标实际上是一种能从包括多条数据记录的结果集中每次提取一条记录的机制。SQL 的游标是一种临时的数据库对象，即可以用来存放在数据库表中的数据行副本，也可以指向存储在数据库中的数据行的指针。游标提供了在逐行的基础上操作表中数据的方法。

游标的一个常见用途就是保存查询结果，以便以后使用。

游标的结果集是由 SELECT 语句产生，如果处理过程需要重复使用一个记录集，那么创建一次游标而重复使用若干次，比重复查询数据库要快的多。

使用:

%FOUND--判断游标中是否还有数据，若有，返回 true，否则，返回 false。

%NOTFOUND--与%FOUND 相反

%ISOPEN--判断游标是否为打开状态

%ROWCOUNT--记录已从游标中取出的记录数

7.如何使用数据库中的定时器？触发器？定时任务？Oracle 中如何实现递归查询?

start with 条件一 connect by prior 条件二 where 条件三

第一种:start with 子节点 ID=’...’ connect by prior 子节点 ID=父节点 ID---按照条件对条件(包括自己)及其子节点进行递归查询,查询结果自己所有后代节点(包括自己)

第二种:start with 子节点 ID=’...’ connect by 子节点 ID = prior 父节点 ID----按照条件对于条件(包括自己)及其父节点进行递归查询,查询结果自己所有的前代节点(包括自己)

第三种: start with 父节点 ID=’...’ connect by prior 子节点 ID=父节点 ID---按照条对条件（不包括自己）子节点进行递归查询,查询结果自己所有的后代节点,(不包括自己)

第四种: start with 父节点 ID= ‘... ‘ connect by 子节点 ID = prior 父节点 ID---按照条件对条件（包括自己）的第一代孩子们及其父节点进行递归查询,查询结果是自己的第一代后节点,和所有的前代节点(包括自己)如果有 where 条件,(4)执行顺序为先执行 start with connect by prior,然后再按照 where 条件进过滤

如何执行批量插入?

首先,创建一个简单的 insert 语句:

```
<insert id=" insertname" >
insert into names (name) values (#{value})
```

</insert>

然后在 java 代码中像下面这样执行批处理插入:

```
list < string > names = new arraylist();
names.add( “fred” );
names.add( “barney” );
names.add( “betty” );
names.add( “wilma” );
// 注意这里 executortype.batch
sqlsession sqlsession =
sqlsessionfactory.opensession(executortype.batch);
try {
namemapper mapper = sqlsession.getmapper(namemapper.class);
for (string name: names) {
mapper.insertname(name);
}
sqlsession.commit();
} catch (Exception e) {
e.printStackTrace();
sqlSession.rollback();
throw e;
}
finally {
```

```
sqlsession.close();
}
```

8.高并发下如何保证修改数据安全

使用 Synchronized 解决，给生成 ID 的代码加上同步代码块，成功解决问题；

作用是：同一时刻，只有一个线程可以执行该代码块

使用 Lock 锁解决问题：给生成 ID 的代码加上 Lock 锁，成功解决问题；

悲观锁---在修改数据的时候,采用锁定状态,排斥外部请求的修改,遇到加锁的状态,就必须等待

弊端: 在高并发下,每个请求都需要等待’锁’,某些线程可能永远都没有机会抢到这个锁,请求就会死在那里,这种请求会很多,瞬间增系统的平均响应时间,结果时可用链接数被耗尽,系统陷入异常

FIFO 队列----采用 FIFO(First Input First Output，先进先出),这样就不会导致某些请求永远获取不到锁

弊端:请求很多,很有可能一瞬间将队列内存”撑爆”,系统陷入到异常状态,或者设计一个极大的内存队列,但是系统处理完一个队列,内请求的速度根本无法和疯狂涌入队列中的数目相比,也就是说,队列内的请求会越积累越多,最终 WEB 系统平均响应时候还是会大幅下降，系统还是陷入异常。

乐观锁---相对于”悲观锁”采用更为宽松的加锁机制,大都是采用带版本号更新,实现就是,这个数据所有请求都有资格去修改,但会获得一个该数据的版本号,只有版本号符合才能更新成功,其他的返回抢购失败,这样就不用了考虑队列的问题,

会增大 CPU 的计算开销,

类别	synchronized	Lock
存在层次	Java的关键字，在jvm层面上	是一个类
锁的释放	1、以获取锁的线程执行完同步代码，释放锁 2、线程执行发生异常，jvm会让线程释放锁	在finally中必须释放锁，不然容易造成线
锁的获取	假设A线程获得锁，B线程等待。如果A线程阻塞，B线程会一直等待	分情况而定，Lock有多个锁获取的方式，具体下面会说 获得锁，线程可以不用一直等待
锁状态	无法判断	可以判断
锁类型	可重入 不可中断 非公平	可重入 可判断 可公平（两者皆可）
性能	少量同步	大量同步

9.MyBatis 实现一对一有几种方式?具体怎么操作的?

有联合查询和嵌套查询,联合查询是几个表联合查询,只查询一次，通过在 resultMap 里面配置 association 节点配置一对一的类就可以完成；嵌套查询是先查一个表，根据这个表里面的结果的 外键 id，去再另外一个表里面查询数据,也是通过 association 配置，但另外一个表的查询通过 select 属性配置。

10. linux 命令，如何改文件权限。

11.数组、链表（单向、双向、双端）、栈和队列、二叉树、红黑树、哈希表、堆（最大和最小）

12. 个人经验：栈和队列、哈希表、链表、二叉树的题较多，图的较少

13. 查找：二分查找及其变形

14. 二叉树：前序、中序、后序遍历，按规定方式打印，两个节点之间操作（最近公共祖先、距离）等问题。

15.Mybatis 是如何进行分页的？分页插件的原理是什么？

Mybatis 使用 RowBounds 对象进行分页，它是针对 ResultSet 结果集执行的内存分页，而非物理分页。可以在 sql 内直接书写带有物理分页的参数来完成物理分页功能，也可以使用分页插件来完成物理分页。

分页插件的基本原理是使用 Mybatis 提供的插件接口，实现自定义插件，在插件的拦截方法内拦截待执行的 sql，然后重写 sql，根据 dialect 方言，添加对应的物理分页语句和物理分页参数。

16.Delete 误删数据没有备份怎么恢复？

flashback query 闪回查询

--尝试使用 Oracle 10g 以后的 flashback Query 特性 闪回查询可以查询若干时间之前的数据

logmnr 日志挖掘

--使用 logminer 日志挖掘 把 delete 的 redo 挖出来看有没有 对应的 undo 回滚 SQL 可用

使用 ORACLE PRM-DUL 工具

--Oracle PRM-DUL 工具可以恢复 数据库中表上 被删除的记录。

17.MyBatis 实现一对多有几种方式,怎么操作的？

有联合查询和嵌套查询。联合查询是几个表联合查询,只查询一次,通过在 resultMap 里面的 collection 节点配置一对多的类就可以完成;嵌套查询是先查一个表,根据这个表里面的 结果的外键 id,去再另外一个表里面查询数据,也是通过配置 collection,但另外一个表的查询通过 select 节点配置。

17.Mybatis 是否支持延迟加载?如果支持,它的实现原理是什么?

答:Mybatis 仅支持 association 关联对象和 collection 关联集合对象的延迟加载,association 指的就是一对一,collection 指的就是一对多查询。在 Mybatis 配置文件中,可以配置是否启用延迟加载 lazyLoadingEnabled=true|false。

它的原理是,使用 CGLIB 创建目标对象的代理对象,当调用目标方法时,进入拦截器方法,比如调用 a.getB().getName(),拦截器 invoke()方法发现 a.getB()是 null 值,那么就会单独发送事先保存好的查询关联 B 对象的 sql,把 B 查询上来,然后调用 a.setB(b),于是 a 的对象 b 属性就有值了,接着完成 a.getB().getName()方法的调用。这就是延迟加载的基本原理。

当然了,不光是 Mybatis,几乎所有的包括 Hibernate,支持延迟加载的原理都是一样的。

18.什么是 MyBatis 的接口绑定?有哪些实现方式?

接口绑定，就是在 MyBatis 中任意定义接口,然后把接口里面的方法和 SQL 语句绑定，我们直接调用接口方法就可以,这样比起原来了 SqlSession 提供的方法我们可以有更加灵活的选择和设置。

接口绑定有两种实现方式,一种是通过注解绑定，就是在接口的方法上面加上 @Select、@Update 等注解，里面包含 Sql 语句来绑定；另外一种就是通过 xml 里面写 SQL 来绑定，在这种情况下,要指定 xml 映射文件里面的 namespace 必须为接口的全路径名。当 Sql 语句比较简单时候,用注解绑定,当 SQL 语句比较复杂时候,用 xml 绑定,一般用 xml 绑定的比较多。

19.使用 MyBatis 的 mapper 接口调用时有哪些要求?

1、Mapper 接口方法名和 mapper.xml 中定义的每个 sql 的 id 相同；

2、Mapper 接口方法的输入参数类型和 mapper.xml 中定义的每个 sql 的 parameterType 的类型相同；

3、Mapper 接口方法的输出参数类型和 mapper.xml 中定义的每个 sql 的 resultType 的类型相同；

4、Mapper.xml 文件中的 namespace 即是 mapper 接口的类路径。

20. 介绍下你理解的操作系统中线程切换过程。

21.. 进程和线程的区别。

22. top 命令之后有哪些内容，有什么作用。

23. 线上 CPU 爆高，请问你如何找到问题所在。

24.什么是 Spring MVC 框架的控制器?

控制器提供一个访问应用程序的行为，此行为通常通过服务接口实现。控制器解析用户输入并将其转换为一个由视图呈现给用户的模型。Spring 用一个非常抽象的方式实现了一个控制层，允许用户创建多种用途的控制器。

25.数据传输的事务定义有哪三种?

和 MQTT 的事务定义一样都是 3 种。

（1）最多一次: 消息不会被重复发送，最多被传输一次，但也有可能一次不传输

（2）最少一次: 消息不会被漏发送，最少被传输一次，但也有可能被重复传输.

（3）精确的一次（Exactly once）：不会漏传输也不会重复传输,每个消息都传输被一次而且仅仅被传输一次，这是大家所期望的

26.谈谈对 Synchronized 关键字，类锁，方法锁，重入锁的理解。

27.volatile 的原理。

28.使用 kafka 有没有遇到什么问题，怎么解决的。

29.MQ 有可能发生重复消费，如何避免，如何做到幂等。

30.MQ 的消息延迟了怎么处理，消息可以设置过期时间么，过期了你们一般怎么处理。

31. Redis 集群方案应该怎么做？都有哪些方案？

32.谈谈你的参与的项目？能否画出整个项目的架构设计图，尽量包含流程、部署等

33. 一次 web 请求响应中，那个部分最耗时，tcp 握手？业务逻辑处理？网络延迟？数据库查询？浏览器解析？

34、分布式系统设计你会考虑哪些策略？

35、最常见的数据分布方式是什么？

36、谈一谈一致性哈希算法。

37、paxos 是什么？

38、什么是 Lease 机制？

39、如何理解选主算法？

40、OSI 有哪七层模型？TCP/IP 是哪四层模型。

41 正常情况下，当在 try 块或 catch 块中遇到 return 语句时，finally 语句块在方法返回之前还是之后被执行？

42、try、catch、finally 语句块的执行顺序。

43、Java 虚拟机中，数据类型可以分为哪几类？

44、怎么理解栈、堆？堆中存什么？栈中存什么？

45、为什么要把堆和栈区分出来呢？栈中不是也可以存储数据吗？

三、字节篇

1.IO 模型有哪些，讲讲你理解的 nio ，他和 bio，aio 的区别是啥，谈谈 reactor 模型。

2.反射的原理，反射创建类实例的三种方式是什么。

3.反射中，Class.forName 和 ClassLoader 区别 。

4.描述动态代理的几种实现方式，分别说出相应的优缺点。

5.动态代理与 cglib 实现的区别。

6.为什么 CGlib 方式可以对接口实现代理。

7.final 的用途。

8.简述 Mybatis 的插件运行原理，以及如何编写一个插件。

答：Mybatis 仅可以编写针对 ParameterHandler、ResultSetHandler、StatementHandler、Executor 这 4 种接口的插件，Mybatis 使用 JDK 的动态代 理，为需要拦截的接口生成代理对象以实现接口方法拦截功能，每当执行

这 4 种接口对象的方法时，就会进入拦截方法，具体就是 InvocationHandler 的 invoke()方法，当然，只会拦截那些你指定需要拦截的方法。

编写插件：实现 Mybatis 的 Interceptor 接口并复写 intercept()方法，然后在给插件编写注解，指定要拦截哪一个接口的哪些方法即可，记住，别忘了在配置文件中配置你编写的插件

9.ZAB 协议?

ZAB 协议是为分布式协调服务 Zookeeper 专门设计的一种支持崩溃恢复的原子广播协议。ZAB 协议包括两种基本的模式：崩溃恢复和消息广播。

当整个 zookeeper 集群刚刚启动或者 Leader 服务器宕机、重启或者网络故障导致不存在过半的服务器与 Leader 服务器保持正常通信时，所有进程（服务器）进入崩溃恢复模式，首先选举产生新的 Leader 服务器，然后集群中 Follower 服务器开始与新的 Leader 服务器进行数据同步，当集群中超过半数机器与该 Leader 服务器完成数据同步之后，退出恢复模式进入消息广播模式，Leader 服务器开始接收客户端的事务请求生成事物提案来进行事务请求处理

10.客户端注册 Watcher 实现

1、调用 getData()/getChildren()/exist()三个 API，传入 Watcher 对象

2、标记请求 request，封装 Watcher 到 WatchRegistration

3、封装成 Packet 对象，发服务端发送 request

4、收到服务端响应后，将 Watcher 注册到 ZKWatcherManager 中进行管理

5、请求返回，完成注册。

11.Chroot 特性

3.2.0 版本后，添加了 Chroot 特性，该特性允许每个客户端为自己设置一个命名 空间。如果一个客户端设置了 Chroot，那么该客户端对服务器的任何操作，都将会被限制在其自己的命名空间下。通过设置 Chroot，能够将一个客户端应用于 Zookeeper 服务端的一颗子树相对应，在那些多个应用公用一个 Zookeeper 进群的场景下，对实现不同应用间的相互隔离非常有帮助。

12.会话管理

分桶策略：将类似的会话放在同一区块中进行管理，以便于 Zookeeper 对会话进行不同区块的隔离处理以及同一区块的统一处理。

分配原则：每个会话的“下次超时时间点”（ExpirationTime）

计算公式：

ExpirationTime_ = currentTime + sessionTimeout

ExpirationTime = (ExpirationTime_ / ExpirationInrerval + 1) * ExpirationInterval , ExpirationInterval 是指 Zookeeper 会话超时检查时间间隔，默认 tickTime

13.服务器角色

Leader

1、事务请求的唯一调度和处理者，保证集群事务处理的顺序性

2、集群内部各服务的调度者

Follower

1、处理客户端的非事务请求，转发事务请求给 Leader 服务器

2、参与事务请求 Proposal 的投票

3、参与 Leader 选举投票

Observer

1、3.0 版本以后引入的一个服务器角色，在不影响集群事务处理能力的基础上提升集群的非事务处理能力

2、处理客户端的非事务请求，转发事务请求给 Leader 服务器

3、不参与任何形式的投票

14.数据同步

整个集群完成 Leader 选举之后，Learner（Follower 和 Observer 的统称）回向 Leader 服务器进行注册。当 Learner 服务器想 Leader 服务器完成注册后，进入数据同步环节。

数据同步流程：（均以消息传递的方式进行）

Learner 向 Learder 注册

数据同步

同步确认

Zookeeper 的数据同步通常分为四类：

1、直接差异化同步（DIFF 同步）

2、先回滚再差异化同步（TRUNC+DIFF 同步）

3、仅回滚同步（TRUNC 同步）

4、全量同步（SNAP 同步）

在进行数据同步前，Leader 服务器会完成数据同步初始化：

peerLastZxid：

第 59 页 共 485 页第 60 页 共 485 页

- 从 learner 服务器注册时发送的 ACKEPOCH 消息中提取 lastZxid（该 Learner 服务器最后处理的 ZXID）

minCommittedLog：

- Leader 服务器 Proposal 缓存队列 committedLog 中最小 ZXID

maxCommittedLog：

- Leader 服务器 Proposal 缓存队列 committedLog 中最大 ZXID

直接差异化同步（DIFF 同步）

- **场景**：peerLastZxid 介于 minCommittedLog 和 maxCommittedLog 之间

先回滚再差异化同步（TRUNC+DIFF 同步）

-

场景：当新的 Leader 服务器发现某个 Learner 服务器包含了一条自己没有的事务记录，那么就需要让该 Learner 服务器进行事务回滚--回滚到 Leader 服务器上存在的，同时也是最接近于 peerLastZxid 的 ZXID

仅回滚同步（TRUNC 同步）第 61 页 共 485 页

- **场景**：peerLastZxid 大于 maxCommittedLog

全量同步（SNAP 同步）

- **场景一**：peerLastZxid 小于 minCommittedLog
- **场景二**：Leader 服务器上没有 Proposal 缓存队列且 peerLastZxid 不等

15.如何理解 Spring 中的代理?

将 Advice 应用于目标对象后创建的对象称为代理。在客户端对象的情况下，目标对象和代理对象是相同的。

Advice + Target Object = Proxy

16.分布式集群中为什么会有 Master?

在分布式环境中，有些业务逻辑只需要集群中的某一台机器进行执行，其他的机器可以共享这个结果，这样可以大大减少重复计算，提高性能，于是就需要进行 leader 选举

17.Zookeeper 对节点的 watch 监听通知是永久的吗？为什么不是永久的?

不是。官方声明：一个 Watch 事件是一个一次性的触发器，当被设置了 Watch 的数据发生了改变的时候，则服务器将这个改变发送给设置了 Watch 的客户端，以便通知它们。

为什么不是永久的，举个例子，如果服务端变动频繁，而监听的客户端很多情况下，每次变动都要通知到所有的客户端，给网络和服务器造成很大压力。

一般是客户端执行 getData(“/节点 A”,true)，如果节点 A 发生了变更或删除，客户端会得到它的 watch 事件，但是在之后节点 A 又发生了变更，而客户端又没有设置 watch 事件，就不再给客户端发送。

在实际应用中，很多情况下，我们的客户端不需要知道服务端的每一次变动，我只要最新的数据即可。

18.chubby 是什么，和 zookeeper 比你怎么看?

chubby 是 google 的，完全实现 paxos 算法，不开源。zookeeper 是 chubby 的开源实现，使用 zab 协议，paxos 算法的变种。

19.说几个 zookeeper 常用的命令。

常用命令：ls get set create delete 等。

20.ZAB 和 Paxos 算法的联系与区别?

相同点:

1、两者都存在一个类似于 Leader 进程的角色，由其负责协调多个 Follower 进程的运行

2、Leader 进程都会等待超过半数的 Follower 做出正确的反馈后，才会将一个提案进行提交

3、ZAB 协议中，每个 Proposal 中都包含一个 epoch 值来代表当前的 Leader 周期，Paxos 中名字为 Ballot

不同点:

ZAB 用来构建高可用的分布式数据主备系统（Zookeeper），Paxos 是用来构建分布式一致性状态机系统。

21.服务调用是阻塞的吗?

默认是阻塞的，可以异步调用，没有返回值的可以这么做。

Dubbo 是基于 NIO 的非阻塞实现并行调用，客户端不需要启动多线程即可完成并行调用多个远程服务，相对多线程开销较小，异步调用会返回一个 Future 对象

22.Dubbo 推荐用什么协议?

- dubbo://（推荐）
- rmi://
-

hessian://

•

http://

•

webservice://

•

thrift://

•

memcached://

•

redis://

•

23.Dubbo Monitor 实现原理?

Consumer 端在发起调用之前会先走 filter 链；provider 端在接收到请求时也是先走 filter 链，然后才进行真正的业务逻辑处理。

默认情况下，在 consumer 和 provider 的 filter 链中都会有 Monitorfilter。

1、MonitorFilter 向 DubboMonitor 发送数据

2、DubboMonitor 将数据进行聚合后（默认聚合 1min 中的统计数据）暂存到 ConcurrentMap<Statistics, AtomicReference> statisticsMap，然后使用一个含有 3 个线程（线程名字：DubboMonitorSendTimer）的线程池每隔 1min 钟，调用 SimpleMonitorService 遍历发送 statisticsMap 中的统计数据，每发送完毕一个，就重置当前的 Statistics 的 AtomicReference

3、SimpleMonitorService 将这些聚合数据塞入 BlockingQueue queue 中（队

列大写为 100000）4、SimpleMonitorService 使用一个后台线程（线程名为：DubboMonitorAsyncWriteLogThread）将 queue 中的数据写入文件（该线程以死循环的形式来写）

5、SimpleMonitorService 还会使用一个含有 1 个线程（线程名字：DubboMonitorTimer）的线程池每隔 5min 钟，将文件中的统计数据画成图表

24.Dubbo 用到哪些设计模式?

Dubbo 框架在初始化和通信过程中使用了多种设计模式，可灵活控制类加载、权限控制等功能。

工厂模式

Provider 在 export 服务时，会调用 ServiceConfig 的 export 方法。ServiceConfig 中有个字段：

```
private static final Protocol protocol =
ExtensionLoader.getExtensionLoader(Protocol.class).getAdaptiveExtensi
on();
```

Dubbo 里有很多这种代码。这也是一种工厂模式，只是实现类的获取采用了 JDK SPI 的机制。这么实现的优点是可扩展性强，想要扩展实现，只需要在 classpath 下增加个文件就可以了，代码零侵入。另外，像上面的 Adaptive 实现，可以做到调用时动态决定调用哪个实现，但是由于这种实现采用了动态代理，会造成代码调试比较麻烦，需要分析出实际调用的实现类。

装饰器模式

Dubbo 在启动和调用阶段都大量使用了装饰器模式。以 Provider 提供的调用链为例，具体的调用链代码是在 ProtocolFilterWrapper 的 buildInvokerChain 完成的，具体是将注解中含有 group=provider 的 Filter 实现，按照 order 排序，最后的调用顺序是：

 EchoFilter -> ClassLoaderFilter -> GenericFilter -> ContextFilter -> ExecuteLimitFilter -> TraceFilter -> TimeoutFilter -> MonitorFilter -> ExceptionFilter

更确切地说，这里是装饰器和责任链模式的混合使用。例如，EchoFilter 的作用是判断是否是回声测试请求，是的话直接返回内容，这是一种责任链的体现。而像 ClassLoaderFilter 则只是在主功能上添加了功能，更改当前线程的 ClassLoader，这是典型的装饰器模式。

观察者模式

Dubbo 的 Provider 启动时，需要与注册中心交互，先注册自己的服务，再订阅自己的服务，订阅时，采用了观察者模式，开启一个 listener。注册中心会每 5 秒定时检查是否有服务更新，如果有更新，向该服务的提供者发送一个 notify 消息，provider 接受到 notify 消息后，即运行 NotifyListener 的 notify 方法，执行监听器方法。

动态代理模式

Dubbo 扩展 JDK SPI 的类 ExtensionLoader 的 Adaptive 实现是典型的动态代理实现。Dubbo 需要灵活地控制实现类，即在调用阶段动态地根据参数决定调用哪个实现类，所以采用先生成代理类的方法，能够做到灵活的调用。生成代理类的

代码是 ExtensionLoader 的 createAdaptiveExtensionClassCode 方法。代理类的主要逻辑是，获取 URL 参数中指定参数的值作为获取实现类的 key。

25.Dubbo SPI 和 Java SPI 区别?

JDK SPI

JDK 标准的 SPI 会一次性加载所有的扩展实现，如果有的扩展吃实话很耗时，但也没用上，很浪费资源。

所以只希望加载某个的实现，就不现实了

DUBBO SPI

1，对 Dubbo 进行扩展，不需要改动 Dubbo 的源码

2，延迟加载，可以一次只加载自己想要加载的扩展实现。

3，增加了对扩展点 IOC 和 AOP 的支持，一个扩展点可以直接 setter 注入其它扩展点。

3，Dubbo 的扩展机制能很好的支持第三方 IoC 容器，默认支持 Spring Bean。

26.简述 Http 请求 get 和 post 的区别以及数据包格式。

27.HTTP 有哪些 method

28.简述 HTTP 请求的报文格式。

29.HTTP 的长连接是什么意思。

30.Thread 类的 sleep()方法和对象的 wait()方法都可以让线程暂停执行，它们有什么区别?

答:

sleep()方法（休眠）是线程类（Thread）的静态方法，调用此方法会让当前线程暂停执行指定的时间，将执行机会（CPU）让给其他线程，但是对象的锁依然保持，因此休眠时间结束后会自动恢复（线程回到就绪状态，请参考第 66 题中的线程状态转换图）。wait()是 Object 类的方法，调用对象的 wait()方法导致当前线程放弃对象的锁（线程暂停执行），进入对象的等待池（wait pool），只有调用对象的 notify()方法（或 notifyAll()方法）时才能唤醒等待池中的线程进入等锁池（lock pool），如果线程重新获得对象的锁就可以进入就绪状态。

补充：可能不少人对什么是进程，什么是线程还比较模糊，对于为什么需要多线程编程也不是特别理解。简单的说：进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动，是操作系统进行资源分配和调度的一个独立单位；线程是进程的一个实体，是 CPU 调度和分派的基本单位，是比进程更小的能独立运行的基本单位。线程的划分尺度小于进程，这使得多线程程序的并发性高；进程在执行时通常拥有独立的内存单元，而线程之间可以共享内存。使用多线程的编程通常能够带来更好的性能和用户体验，但是多线程的程序对于其他程序是不友好的，因为它可能占用了更多的 CPU 资源。当然，也不是线程越多，程序的性能就越好，因为线程之间的调度和切换也会浪费 CPU 时间。时下很时髦的 Node.js 就采用了单线程异步 I/O 的工作模式。

31. Java 服务端问题排查（OOM，CPU 高，Load 高，类冲突）

32. 如何查看 Java 应用的线程信息

33.Redis hash 算法用的是什么?